CComp

Generated by Doxygen 1.8.3.1

Thu May 9 2013 14:45:49

Contents

Chapter 1

University of Nevada, Reno <br/><br/> CS 660 - Compilers

Authors: Alex Fiannaca & Sandeep Mathew

Due Date: 05/15/2013

Submitted Materials: GitHub

Contents:

- Description
- Language Features
- Implementation Details
- Assumptions
- Restrictions
- How-To Section (compiling and running the project)

Description:

This is the front end of a simple C compiler being written for the CS660 Compiler course at the University of Nevada, Reno (Spring 2013). In order to compile the scanner and parser, you must use the most recent versions of Flex and Bison, the GNU scanner and parser generators.

Links to dependencies:

- Bison 2.7
- Flex 2.5.37

For more information, contact fiannac4@live.com

Language Features: The C Programming Language

This compiler is for the C programming language and is based off the specification description and grammar from K&R. Advanced features of C such as the auto keyword functionality and the register keyword functionality have not been implemented.

Implementation Details

This compiler is broken down into several classes that work together to generate assembly code (currently the compiler only goes through the steps of scanning and parsing the input C file and stops before generating 3AC code). The code for this project can be accessed through the GitHub repository for the project. The main classes are as follows:

- **CCompiler**: This class is the driver for the compiler. It brings together the scanner and parser in a unified manner by providing a single set of error and warning functions so that the scanner and parser do not have to implement their own. In addition, it redirects the output from the scanner and parser in a unified manner so that debugging is easier to perform. The instance of this class that is created when the program is run, can be found in the main.cpp file.
- **CParser.yy**: This class is automatically generated by Bison from the CParser.yy file.
- **AVLTree**: This class is used by the symbol table to maintain a balanced tree of symbols, thereby guaranteeing efficient interaction with the symbol table. This was implemented instead of using the STL map class, since the map class does not allow any method for printing the physical structure of the tree, and therefore cannot show that the tree is balanced (a project requirement).
- **SymTab**: This class is the symbol table. It allows for inserting and search for symbols and manages the scoping restrictions of the compiler.
- **Type** (and all derived classes): This set of classes is used for maintaining information about the type of a symbol. There are 8 derived classes from the base Type class: pointers, functions, unions, structs, arrays, enums, and plain old data types (PODs). These classes allow for efficient management of descriptive information such as the capacity of an array or the size of a datatype (in bytes).

Other Files:

- **main.cpp**: This file creates an instance of the compiler and parses the command line arguments.
- **Platform.h**: This file specifies the size of base types in our compiler.
- **CScanner.ll**: This file is passed to Flex in order to generate a scanner for the compiler. The output scanner is written in C++ compatible C and is therefore able to be compiled into the compiler.
- **tests/**: This folder contains several test cases which can be run with the run_tests bash script.

Assumptions

There were several assumptions when creating this compiler:

- The input file will be in ASCII encoding.

Restrictions

- The main restriction in this compiler is that the compiler will abort after the first error (it will continue after warnings). We decided to do this in order to ensure that the output error is the correct error rather than outputting a series of errors in which only the first error is the real error (umm, gcc...).
- When using the run_tests script (the preferred method for testing this compiler), the output of scanner and parser debug information are written to the .ydb and .ldb files, the tokens and reductions are written to the .tok and .red files, and symbol table output is written to the standard output stream.

How To: Compile this project

Clone the repository using the following command:

```
git clone https://github.com/afiannac2/cs660.git
```

This repository is a private repository, meaning that you must be granted access rights before you can clone the repository, but since you are reading this, I am guessing you have been given rights.
Execute the following commands to compile the project:

```
cd cs660/flexbison/
make
```

This will run flex and bison and then compile all of the sources to produce the C Compiler front end executable ∗__ccomp__∗

How To: Run this project (direct use)

The C Compiler front end can either be invoked by using the following command

```
./ccomp -d lsp -v -o <filename>.out <filename>
```

How To: Run this project (using the test script)

A script has been written which allows you to automatically run any test ∗.c∗ files located in the *tests/* folder. The script can be invoked through the makefile by running the command (after you have compiled the project)

```
make test
```

This will copy the compiler executable into the *tests/* folder and will run the compiler on every ∗.c∗ file in the *tests/* folder. This will invoke the compiler with the ∗-d∗ flag (with the l, s, and p parameters - lexer, symbol table, and parser respectively) and the ∗-o∗ flag (with an output filename of <filename>.out). In addition, it will group the results of each attempted compilation into a folder with the same name as the input C file. Some of the test files in the *tests/* folder are designed to fail (to test error output and whatnot) and some are designed to compile correctly (to test things like proper input into the symbol table). The ∗-v∗ flag can also be provided to the *ccomp* executable in order to print general debug info (this is for the developers).

How To: Generate the GraphVis Visualization of the AST

Running the compiler will generate a ∗.dot∗ output file which represents the AST in a GraphViz format. In order to generate a PNG image of the graph, run this command

```
    dot -Tpng vis.dot -o vis.png
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- Address . . . . . ??
- AddressTable . . . . . ??
- AST . . . . . ??
  - AstAbstractDecl . . . . . ??
  - AstAbstractDecl . . . . . ??
  - AstAbstractDecl . . . . . ??
  - AstAbstractDecl . . . . . ??
  - AstAddExpr . . . . . ??
  - AstAddExpr . . . . . ??
  - AstAddExpr . . . . . ??
  - AstAddExpr . . . . . ??
  - AstAndExpr . . . . . ??
  - AstAndExpr . . . . . ??
  - AstAndExpr . . . . . ??
  - AstAndExpr . . . . . ??
  - AstArgExprList . . . . . ??
  - AstArgExprList . . . . . ??
  - AstArgExprList . . . . . ??
  - AstArgExprList . . . . . ??
  - AstAssignExpr . . . . . ??
  - AstAssignExpr . . . . . ??
  - AstAssignExpr . . . . . ??
  - AstAssignExpr . . . . . ??
  - AstAssignOp . . . . . ??
  - AstAssignOp . . . . . ??
  - AstAssignOp . . . . . ??
  - AstAssignOp . . . . . ??
  - AstBreak . . . . . ??
  - AstBreak . . . . . ??
  - AstBreak . . . . . ??
  - AstBreak . . . . . ??
  - AstCastExpr . . . . . ??
  - AstCastExpr . . . . . ??
  - AstCastExpr . . . . . ??
  - AstCastExpr . . . . . ??
  - AstCompoundStmt . . . . . ??
  - AstCompoundStmt . . . . . ??
  - AstCompoundStmt . . . . . ??

AstCompoundStmt . . . . . ??
AstConditionalExpr . . . . . ??
AstConditionalExpr . . . . . ??
AstConditionalExpr . . . . . ??
AstConditionalExpr . . . . . ??
AstConstant . . . . . ??
AstConstant . . . . . ??
AstConstant . . . . . ??
AstConstant . . . . . ??
AstConstantExpr . . . . . ??
AstConstantExpr . . . . . ??
AstConstantExpr . . . . . ??
AstConstantExpr . . . . . ??
AstContinue . . . . . ??
AstContinue . . . . . ??
AstContinue . . . . . ??
AstContinue . . . . . ??
AstDecl . . . . . ??
AstDecl . . . . . ??
AstDecl . . . . . ??
AstDecl . . . . . ??
AstDeclarationList . . . . . ??
AstDeclarationList . . . . . ??
AstDeclarationList . . . . . ??
AstDeclarationList . . . . . ??
AstDeclarator . . . . . ??
AstDeclarator . . . . . ??
AstDeclarator . . . . . ??
AstDeclarator . . . . . ??
AstDeclList . . . . . ??
AstDeclList . . . . . ??
AstDeclList . . . . . ??
AstDeclList . . . . . ??
AstDecSpeci . . . . . ??
AstDecSpeci . . . . . ??
AstDecSpeci . . . . . ??
AstDecSpeci . . . . . ??
AstDirectAbsDecl . . . . . ??
AstDirectAbsDecl . . . . . ??
AstDirectAbsDecl . . . . . ??
AstDirectAbsDecl . . . . . ??
AstDirectDecl . . . . . ??
AstDirectDecl . . . . . ??
AstDirectDecl . . . . . ??
AstDirectDecl . . . . . ??
AstDoWhile . . . . . ??
AstDoWhile . . . . . ??
AstDoWhile . . . . . ??
AstDoWhile . . . . . ??
AstEnumerator . . . . . ??
AstEnumerator . . . . . ??
AstEnumerator . . . . . ??
AstEnumerator . . . . . ??
AstEnumList . . . . . ??
AstEnumList . . . . . ??
AstEnumList . . . . . ??
AstEnumList . . . . . ??
AstEqExpr . . . . . ??

AstEqExpr . . . . ??
AstEqExpr . . . . ??
AstEqExpr . . . . ??
AstExpression . . . . ??
AstExpression . . . . ??
AstExpression . . . . ??
AstExpression . . . . ??
AstExprStmt . . . . ??
AstExprStmt . . . . ??
AstExprStmt . . . . ??
AstExprStmt . . . . ??
AstFor . . . . ??
AstFor . . . . ??
AstFor . . . . ??
AstFor . . . . ??
AstFuncDef . . . . ??
AstFuncDef . . . . ??
AstFuncDef . . . . ??
AstFuncDef . . . . ??
AstGoto . . . . ??
AstGoto . . . . ??
AstGoto . . . . ??
AstGoto . . . . ??
AstID . . . . ??
AstID . . . . ??
AstID . . . . ??
AstID . . . . ??
AstIDList . . . . ??
AstIDList . . . . ??
AstIDList . . . . ??
AstIDList . . . . ??
AstIfElse . . . . ??
AstIfElse . . . . ??
AstIfElse . . . . ??
AstIfElse . . . . ??
AstInitDeclarator . . . . ??
AstInitDeclarator . . . . ??
AstInitDeclarator . . . . ??
AstInitDeclarator . . . . ??
AstInitDeclList . . . . ??
AstInitDeclList . . . . ??
AstInitDeclList . . . . ??
AstInitDeclList . . . . ??
AstInitializer . . . . ??
AstInitializer . . . . ??
AstInitializer . . . . ??
AstInitializer . . . . ??
AstInitList . . . . ??
AstInitList . . . . ??
AstInitList . . . . ??
AstInitList . . . . ??
AstIteration . . . . ??
AstIteration . . . . ??
AstIteration . . . . ??
AstIteration . . . . ??
AstJump . . . . ??
AstJump . . . . ??
AstJump . . . . ??

AstJump . . . . ??
AstLabeledStmt . . . . ??
AstLabeledStmt . . . . ??
AstLabeledStmt . . . . ??
AstLabeledStmt . . . . ??
AstLogicAndExpr . . . . ??
AstLogicAndExpr . . . . ??
AstLogicAndExpr . . . . ??
AstLogicAndExpr . . . . ??
AstLogicOrExpr . . . . ??
AstLogicOrExpr . . . . ??
AstLogicOrExpr . . . . ??
AstLogicOrExpr . . . . ??
AstMultExpr . . . . ??
AstMultExpr . . . . ??
AstMultExpr . . . . ??
AstMultExpr . . . . ??
AstORExpr . . . . ??
AstORExpr . . . . ??
AstORExpr . . . . ??
AstORExpr . . . . ??
AstParamDec . . . . ??
AstParamDec . . . . ??
AstParamDec . . . . ??
AstParamDec . . . . ??
AstParamList . . . . ??
AstParamList . . . . ??
AstParamList . . . . ??
AstParamList . . . . ??
AstPointer . . . . ??
AstPointer . . . . ??
AstPointer . . . . ??
AstPointer . . . . ??
AstPostfixExpr . . . . ??
AstPostfixExpr . . . . ??
AstPostfixExpr . . . . ??
AstPostfixExpr . . . . ??
AstPrimaryExpr . . . . ??
AstPrimaryExpr . . . . ??
AstPrimaryExpr . . . . ??
AstPrimaryExpr . . . . ??
AstRelExpr . . . . ??
AstRelExpr . . . . ??
AstRelExpr . . . . ??
AstRelExpr . . . . ??
AstReturn . . . . ??
AstReturn . . . . ??
AstReturn . . . . ??
AstReturn . . . . ??
AstSelection . . . . ??
AstSelection . . . . ??
AstSelection . . . . ??
AstSelection . . . . ??
AstShiftExpr . . . . ??
AstShiftExpr . . . . ??
AstShiftExpr . . . . ??
AstShiftExpr . . . . ??
AstSpeciQualList . . . . ??

AstSpeciQualList . . . . ??
AstSpeciQualList . . . . ??
AstSpeciQualList . . . . ??
AstStatement . . . . ??
AstStatement . . . . ??
AstStatement . . . . ??
AstStatement . . . . ??
AstStatementList . . . . ??
AstStatementList . . . . ??
AstStatementList . . . . ??
AstStatementList . . . . ??
AstString . . . . ??
AstString . . . . ??
AstString . . . . ??
AstString . . . . ??
AstStructDecl . . . . ??
AstStructDecl . . . . ??
AstStructDecl . . . . ??
AstStructDecl . . . . ??
AstStructDeclarator . . . . ??
AstStructDeclarator . . . . ??
AstStructDeclarator . . . . ??
AstStructDeclarator . . . . ??
AstStructDeclatorList . . . . ??
AstStructDeclatorList . . . . ??
AstStructDeclatorList . . . . ??
AstStructDeclatorList . . . . ??
AstStructDeclList . . . . ??
AstStructDeclList . . . . ??
AstStructDeclList . . . . ??
AstStructDeclList . . . . ??
AstStructUniSpeci . . . . ??
AstStructUniSpeci . . . . ??
AstStructUniSpeci . . . . ??
AstStructUniSpeci . . . . ??
AstSwitch . . . . ??
AstSwitch . . . . ??
AstSwitch . . . . ??
AstSwitch . . . . ??
AstTrans . . . . ??
AstTrans . . . . ??
AstTrans . . . . ??
AstTrans . . . . ??
AstTypeName . . . . ??
AstTypeName . . . . ??
AstTypeName . . . . ??
AstTypeName . . . . ??
AstTypeParamList . . . . ??
AstTypeParamList . . . . ??
AstTypeParamList . . . . ??
AstTypeParamList . . . . ??
AstTypeQualList . . . . ??
AstTypeQualList . . . . ??
AstTypeQualList . . . . ??
AstTypeQualList . . . . ??
AstTypeSpeci . . . . ??
AstTypeSpeci . . . . ??
AstTypeSpeci . . . . ??

  - AstTypeSpeci . . . . **??**
  - AstUnaryExpr . . . . **??**
  - AstUnaryExpr . . . . **??**
  - AstUnaryExpr . . . . **??**
  - AstUnaryExpr . . . . **??**
  - AstUnaryOp . . . . **??**
  - AstUnaryOp . . . . **??**
  - AstUnaryOp . . . . **??**
  - AstUnaryOp . . . . **??**
  - AstWhile . . . . **??**
  - AstWhile . . . . **??**
  - AstWhile . . . . **??**
  - AstWhile . . . . **??**
  - AstXORExpr . . . . **??**
  - AstXORExpr . . . . **??**
  - AstXORExpr . . . . **??**
  - AstXORExpr . . . . **??**
  - EnumSpecifier . . . . **??**
  - EnumSpecifier . . . . **??**
  - EnumSpecifier . . . . **??**
  - EnumSpecifier . . . . **??**
- AVLTree< DataItem > . . . . **??**
- CCompiler . . . . **??**
- GoFPatterns::Event< SourceType, EventArgType > . . . . **??**
- GoFPatterns::Event< RegAllocTable, void $*$ > . . . . **??**
- Function . . . . **??**
- FunctionTable . . . . **??**
- InputLine . . . . **??**
- AVLTree< DataItem >::Node . . . . **??**
- Parameter . . . . **??**
- RegAllocTable . . . . **??**
- Register . . . . **??**
- GoFPatterns::Event< SourceType, EventArgType >::SubscriberRecord . . . . **??**
- SymbolInfo . . . . **??**
- SymTab . . . . **??**
- tac2mips . . . . **??**
- TAC_Generator . . . . **??**
- Type . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - ArrayType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**
  - EnumType . . . . **??**

EnumType . . . ??
EnumType . . . ??
EnumType . . . ??
EnumType . . . ??
EnumType . . . ??
EnumType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
FunctionType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PODType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
PointerType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??
StructType . . . ??

TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
TypedefType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
UnionType . . . . **??**
Visualizer . . . . **??**

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Address
This struct describes the entire location information of a variable in the program including the memory location flag, the name of the register it is in (if it is in a register), and the memory offset location it is in (if it is also in memory) . . . ??
AddressTable . . . ??
ArrayType . . . ??
AST
Abstract syntax tree node type . . . ??
AstAbstractDecl . . . ??
AstAddExpr . . . ??
AstAndExpr . . . ??
AstArgExprList . . . ??
AstAssignExpr . . . ??
AstAssignOp . . . ??
AstBreak . . . ??
AstCastExpr . . . ??
AstCompoundStmt . . . ??
AstConditionalExpr . . . ??
AstConstant . . . ??
AstConstantExpr . . . ??
AstContinue . . . ??
AstDecl . . . ??
AstDeclarationList . . . ??
AstDeclarator . . . ??
AstDeclList . . . ??
AstDecSpeci . . . ??
AstDirectAbsDecl . . . ??
AstDirectDecl . . . ??
AstDoWhile . . . ??
AstEnumerator . . . ??
AstEnumList . . . ??
AstEqExpr . . . ??
AstExpression . . . ??
AstExprStmt . . . ??
AstFor . . . ??
AstFuncDef . . . ??
AstGoto . . . ??
AstID . . . ??

AstIDList . . . . . ??
AstIfElse . . . . . ??
AstInitDeclarator . . . . . ??
AstInitDeclList . . . . . ??
AstInitializer . . . . . ??
AstInitList . . . . . ??
AstIteration . . . . . ??
AstJump . . . . . ??
AstLabeledStmt . . . . . ??
AstLogicAndExpr . . . . . ??
AstLogicOrExpr . . . . . ??
AstMultExpr . . . . . ??
AstORExpr . . . . . ??
AstParamDec . . . . . ??
AstParamList . . . . . ??
AstPointer . . . . . ??
AstPostfixExpr . . . . . ??
AstPrimaryExpr . . . . . ??
AstRelExpr . . . . . ??
AstReturn . . . . . ??
AstSelection . . . . . ??
AstShiftExpr . . . . . ??
AstSpeciQualList . . . . . ??
AstStatement . . . . . ??
AstStatementList . . . . . ??
AstString . . . . . ??
AstStructDecl . . . . . ??
AstStructDeclarator . . . . . ??
AstStructDeclatorList . . . . . ??
AstStructDeclList . . . . . ??
AstStructUniSpeci . . . . . ??
AstSwitch . . . . . ??
AstTrans . . . . . ??
AstTypeName . . . . . ??
AstTypeParamList . . . . . ??
AstTypeQualList . . . . . ??
AstTypeSpeci . . . . . ??
AstUnaryExpr . . . . . ??
AstUnaryOp . . . . . ??
AstWhile . . . . . ??
AstXORExpr . . . . . ??
AVLTree< DataItem >
An implementation of a balanced binary tree called an AVL tree . . . . . ??
CCompiler
A minimalist C programming language compiler class . . . . . ??
EnumSpecifier . . . . . ??
EnumType . . . . . ??
GoFPatterns::Event< SourceType, EventArgType > . . . . . ??
Function . . . . . ??
FunctionTable . . . . . ??
FunctionType . . . . . ??
InputLine
A structure for buffering lines input code . . . . . ??
AVLTree< DataItem >::Node
A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node . . . . . ??
Parameter . . . . . ??
PODType . . . . . ??

PointerType . . . . ??
RegAllocTable . . . . ??
Register . . . . ??
StructType . . . . ??
GoFPatterns::Event< SourceType, EventArgType >::SubscriberRecord
This inner class, for each EventHandler, stores the associated context information - pointer . . ??
SymbolInfo . . . . ??
SymTab . . . . ??
tac2mips . . . . ??
TAC_Generator
A class for generating three address code . . . . ??
Type . . . . ??
TypedefType . . . . ??
UnionType . . . . ??
Visualizer
A class for visualizing the generation of the AST . . . . ??

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Ast.cpp . . . . . . ??
Ast.h . . . . . . ??
AvlTree.h . . . . . . ??
CCompiler.cpp . . . . . . ??
CCompiler.h . . . . . . ??
CParser.yy
NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATED DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE BISON FILES! . . . . . . ??
CScanner.ll
NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATE-D DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE FLEX FILES! . . . . . . ??
main.cpp
This file serves as an entry point to the compiler . . . . . . ??
Platform.h
This file simply lists the size in bytes of the base data types . . . . . . ??
SymTab.cpp . . . . . . ??
SymTab.h . . . . . . ??
TAC_Generator.cpp . . . . . . ??
TAC_Generator.h . . . . . . ??
Type.cpp . . . . . . ??
Type.h . . . . . . ??
Visualizer.cpp . . . . . . ??
Visualizer.h . . . . . . ??
mips/**AddressTable.cpp** . . . . . . ??
mips/**AddressTable.h** . . . . . . ??
mips/**AddressTest.cpp** . . . . . . ??
mips/**Event.h** . . . . . . ??
mips/**FunctionTable.cpp** . . . . . . ??
mips/**FunctionTable.h** . . . . . . ??
mips/**main.cpp** . . . . . . ??
mips/**RegAllocTable.cpp** . . . . . . ??
mips/**RegAllocTable.h** . . . . . . ??
mips/**SpillTest.cpp** . . . . . . ??
mips/**tac2mips.cpp** . . . . . . ??
mips/**tac2mips.h** . . . . . . ??
mips/**TAC_Parser.yy** . . . . . . ??

mips/**TAC_Scanner.ll** . . . . **??**
tests/**test1.c** . . . . **??**
tests/**test2.c** . . . . **??**
tests/**test3.c** . . . . **??**
tests/**test4.c** . . . . **??**
tests/**test5.c** . . . . **??**
tests/**test6.c** . . . . **??**
tests/**test7.c** . . . . **??**

Chapter 5

Class Documentation

5.1 Address Struct Reference

This struct describes the entire location information of a variable in the program including the memory location flag, the name of the register it is in (if it is in a register), and the memory offset location it is in (if it is also in memory).

`#include <AddressTable.h>`

Public Member Functions

- Address ()

  Default constructor.
- Address (const Address &rhs)

  Standard copy constructor.
- Address ()

  Default constructor.
- Address (const Address &rhs)

  Standard copy constructor.
- Address ()

  Default constructor.
- Address (const Address &rhs)

  Standard copy constructor.

Public Attributes

- string varName

  The name of the variable.
- MemLocation loc

  Flag indicating where the variable is (memory, register, or both).
- string reg

  The register the variable is located in.
- int memOffset

  The memory offset of the variable.

5.1.1 Detailed Description

This struct describes the entire location information of a variable in the program including the memory location flag, the name of the register it is in (if it is in a register), and the memory offset location it is in (if it is also in memory).

Definition at line 30 of file AddressTable.h.

The documentation for this struct was generated from the following file:

- mips/AddressTable.h

5.2 AddressTable Class Reference

Public Member Functions

- AddressTable (RegAllocTable ∗rt)

  The default constructor for the address table.
- ∼AddressTable ()

  Destructor for the address table.
- void Add (string name, string reg)

  Adds a variable to the address table and sets the variable's location to the provided register.
- void Add (string name, int offset)

  Adds a variable to the address table and sets the variable's location to the provided memory offset.
- void Add (string name, string reg, int offset)

  Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.
- void Remove (string name)

  Removes the record for the given variable from the address table.
- Address ∗ Lookup (string name)

  This function looks up the variable of the given name and then returns a pointer to the record for that variable.
- Address ∗ **LookupReg** (string reg)
- string Load (string name)

  This function is responsible for loading variables from memory into a register.
- string Load (Address ∗addr)

  This function is responsible for loading variables from memory into a register.
- void Store (string name)

  This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).
- void Store (string reg, string name)

  This function stores the value of a register into a variable in memory.
- void UpdateRegisters (RegAllocTable ∗src)

  This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.
- void clear ()

  Clears out all the records in the address table.
- int size ()

  Returns the number of records in the address table.
- void SetFstream (fstream ∗fs)

  Sets the output stream.
- void Print ()

  This function is intended for debugging uses only.
- void SetVerbose (bool flag)

  Turns verbose comments on or off.

- AddressTable (RegAllocTable ∗rt)

    The default constructor for the address table.
- ∼AddressTable ()

    Destructor for the address table.
- void Add (string name, string reg)

    Adds a variable to the address table and sets the variable's location to the provided register.
- void Add (string name, int offset)

    Adds a variable to the address table and sets the variable's location to the provided memory offset.
- void Add (string name, string reg, int offset)

    Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.
- void Remove (string name)

    Removes the record for the given variable from the address table.
- Address ∗ Lookup (string name)

    This function looks up the variable of the given name and then returns a pointer to the record for that variable.
- Address ∗ **LookupReg** (string reg)
- string Load (string name)

    This function is responsible for loading variables from memory into a register.
- string Load (Address ∗addr)

    This function is responsible for loading variables from memory into a register.
- void Store (string name)

    This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).
- void Store (string reg, string name)

    This function stores the value of a register into a variable in memory.
- void UpdateRegisters (RegAllocTable ∗src)

    This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.
- void clear ()

    Clears out all the records in the address table.
- int size ()

    Returns the number of records in the address table.
- void SetFstream (fstream ∗fs)

    Sets the output stream.
- void Print ()

    This function is intended for debugging uses only.
- void SetVerbose (bool flag)

    Turns verbose comments on or off.
- AddressTable (RegAllocTable ∗rt)

    The default constructor for the address table.
- ∼AddressTable ()

    Destructor for the address table.
- void Add (string name, string reg)

    Adds a variable to the address table and sets the variable's location to the provided register.
- void Add (string name, int offset)

    Adds a variable to the address table and sets the variable's location to the provided memory offset.
- void Add (string name, string reg, int offset)

    Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.
- void Remove (string name)

    Removes the record for the given variable from the address table.
- Address ∗ Lookup (string name)

    This function looks up the variable of the given name and then returns a pointer to the record for that variable.

- Address ∗ **LookupReg** (string reg)
- string Load (string name)

    This function is responsible for loading variables from memory into a register.
- string Load (Address ∗addr)

    This function is responsible for loading variables from memory into a register.
- void Store (string name)

    This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).
- void Store (string reg, string name)

    This function stores the value of a register into a variable in memory.
- void UpdateRegisters (RegAllocTable ∗src)

    This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.
- void clear ()

    Clears out all the records in the address table.
- int size ()

    Returns the number of records in the address table.
- void SetFstream (fstream ∗fs)

    Sets the output stream.
- void Print ()

    This function is intended for debugging uses only.
- void SetVerbose (bool flag)

    Turns verbose comments on or off.

Static Public Member Functions

- static void Update (RegAllocTable ∗src, void ∗data, void ∗context)

    This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.
- static void Update (RegAllocTable ∗src, void ∗data, void ∗context)

    This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.
- static void Update (RegAllocTable ∗src, void ∗data, void ∗context)

    This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.

Private Attributes

- map< string, Address ∗ > Variables

    The internal data structure for the table.
- RegAllocTable ∗ regtab

    A reference to the register allocation table owned by the tac2mips object which also owns this object.
- fstream ∗ fout

    The output stream to print MIPS to.
- bool verbose

    Indicates if verbose comments should be output in the MIPS.

5.2.1 Detailed Description

Definition at line 74 of file AddressTable.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AddressTable::AddressTable (**RegAllocTable** ∗ *rt*)

The default constructor for the address table.

Parameters

rt	A reference to the register allocation table.

Definition at line 3 of file AddressTable.cpp.

5.2.2.2 AddressTable::AddressTable (**RegAllocTable** ∗ *rt*)

The default constructor for the address table.

Parameters

rt	A reference to the register allocation table.

5.2.2.3 AddressTable::AddressTable (**RegAllocTable** ∗ *rt*)

The default constructor for the address table.

Parameters

rt	A reference to the register allocation table.

5.2.3 Member Function Documentation

5.2.3.1 void AddressTable::Add (string *name,* string *reg*)

Adds a variable to the address table and sets the variable's location to the provided register.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.

Definition at line 14 of file AddressTable.cpp.

5.2.3.2 void AddressTable::Add (string *name,* string *reg*)

Adds a variable to the address table and sets the variable's location to the provided register.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.

5.2.3.3 void AddressTable::Add (string *name,* string *reg*)

Adds a variable to the address table and sets the variable's location to the provided register.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.

5.2.3.4 void AddressTable::Add (string *name*, int *offset*)

Adds a variable to the address table and sets the variable's location to the provided memory offset.

Parameters

name	The name of the variable.
offset	The memory offset the variable is in.

Definition at line 25 of file AddressTable.cpp.

5.2.3.5 void AddressTable::Add (string *name*, int *offset*)

Adds a variable to the address table and sets the variable's location to the provided memory offset.

Parameters

name	The name of the variable.
offset	The memory offset the variable is in.

5.2.3.6 void AddressTable::Add (string *name*, int *offset*)

Adds a variable to the address table and sets the variable's location to the provided memory offset.

Parameters

name	The name of the variable.
offset	The memory offset the variable is in.

5.2.3.7 void AddressTable::Add (string *name*, string *reg*, int *offset*)

Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.
offset	The memory offset the variable is in.

5.2.3.8 void AddressTable::Add (string *name*, string *reg*, int *offset*)

Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.
offset	The memory offset the variable is in.

5.2.3.9 void AddressTable::Add (string *name*, string *reg*, int *offset*)

Adds a variable to the address table and sets the variable's location to both the provided register and memory offset.

Parameters

name	The name of the variable.
reg	The name of the register the variable is in.
offset	The memory offset the variable is in.

Definition at line 36 of file AddressTable.cpp.

5.2.3.10 string AddressTable::Load (string *name*)

This function is responsible for loading variables from memory into a register.

Parameters

name	The name of the variable to load

Returns

The name of the register the variable has been loaded into

5.2.3.11 string AddressTable::Load (string *name*)

This function is responsible for loading variables from memory into a register.

Parameters

name	The name of the variable to load

Returns

The name of the register the variable has been loaded into

5.2.3.12 string AddressTable::Load (string *name*)

This function is responsible for loading variables from memory into a register.

Parameters

name	The name of the variable to load

Returns

The name of the register the variable has been loaded into

Definition at line 74 of file AddressTable.cpp.

5.2.3.13 string AddressTable::Load (Address $*$ *addr*)

This function is responsible for loading variables from memory into a register.

Parameters

addr	A reference to the address struct of the variable to load

Returns

The name of the register the variable has been loaded into

Definition at line 104 of file AddressTable.cpp.

5.2.3.14 string AddressTable::Load (Address ∗ *addr*)

This function is responsible for loading variables from memory into a register.

Parameters

addr	A reference to the address struct of the variable to load

Returns

The name of the register the variable has been loaded into

5.2.3.15 string AddressTable::Load (Address ∗ *addr*)

This function is responsible for loading variables from memory into a register.

Parameters

addr	A reference to the address struct of the variable to load

Returns

The name of the register the variable has been loaded into

5.2.3.16 Address∗ AddressTable::Lookup (string *name*)

This function looks up the variable of the given name and then returns a pointer to the record for that variable.

This function can also be used to determine if a string refers to a variable or a temp register. If the return value is NULL, then the string passed in refers to a temp register since the string is not a name of a variable in the table.

Parameters

name	The name of the variable in the table to look up.

Returns

A pointer to the address table record for the requested variable.

5.2.3.17 Address ∗ AddressTable::Lookup (string *name*)

This function looks up the variable of the given name and then returns a pointer to the record for that variable.

This function can also be used to determine if a string refers to a variable or a temp register. If the return value is NULL, then the string passed in refers to a temp register since the string is not a name of a variable in the table.

Parameters

name	The name of the variable in the table to look up.

Returns

A pointer to the address table record for the requested variable.

Definition at line 53 of file AddressTable.cpp.

5.2.3.18 Address∗ AddressTable::Lookup (string *name*)

This function looks up the variable of the given name and then returns a pointer to the record for that variable.

This function can also be used to determine if a string refers to a variable or a temp register. If the return value is NULL, then the string passed in refers to a temp register since the string is not a name of a variable in the table.

Parameters

name	The name of the variable in the table to look up.

Returns

A pointer to the address table record for the requested variable.

5.2.3.19 void AddressTable::Print ( )

This function is intended for debugging uses only.

This function prints out the current contents of the address table.

5.2.3.20 void AddressTable::Print ( )

This function is intended for debugging uses only.

This function prints out the current contents of the address table.

Definition at line 219 of file AddressTable.cpp.

5.2.3.21 void AddressTable::Print ( )

This function is intended for debugging uses only.

This function prints out the current contents of the address table.

5.2.3.22 void AddressTable::Remove (string *name*)

Removes the record for the given variable from the address table.

In essence, by calling this function, it is assumed that the variable will never again be used in the program.

Parameters

name	The name of the variable to remove from the table.

Definition at line 48 of file AddressTable.cpp.

5.2.3.23 void AddressTable::Remove (string *name*)

Removes the record for the given variable from the address table.

In essence, by calling this function, it is assumed that the variable will never again be used in the program.

Parameters

name	The name of the variable to remove from the table.

5.2.3.24 void AddressTable::Remove (string *name*)

Removes the record for the given variable from the address table.

In essence, by calling this function, it is assumed that the variable will never again be used in the program.

Parameters

name	The name of the variable to remove from the table.

5.2.3.25 void AddressTable::SetFstream (fstream ∗ *fs*)

Sets the output stream.

Parameters

fs	A reference to the filestream.

5.2.3.26 void AddressTable::SetFstream (fstream ∗ *fs*)

Sets the output stream.

Parameters

fs	A reference to the filestream.

5.2.3.27 void AddressTable::SetFstream (fstream ∗ *fs*)

Sets the output stream.

Parameters

fs	A reference to the filestream.

Definition at line 214 of file AddressTable.cpp.

5.2.3.28 int AddressTable::size ( )

Returns the number of records in the address table.

Returns

The number of records in the address table.

5.2.3.29 int AddressTable::size ( )

Returns the number of records in the address table.

Returns

The number of records in the address table.

Definition at line 209 of file AddressTable.cpp.

5.2.3.30 int AddressTable::size ( )

Returns the number of records in the address table.

Returns

The number of records in the address table.

5.2.3.31 void AddressTable::Store (string *name*)

This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).

Parameters

name	The name of the variable to store back to memory.

Definition at line 130 of file AddressTable.cpp.

5.2.3.32 void AddressTable::Store (string *name*)

This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).

Parameters

name	The name of the variable to store back to memory.

5.2.3.33 void AddressTable::Store (string *name*)

This function is responsible for storing the given variable back into memory (no effect will occur if the provided variable is only currently in memory and not in a register).

Parameters

name	The name of the variable to store back to memory.

5.2.3.34 void AddressTable::Store (string *reg,* string *name*)

This function stores the value of a register into a variable in memory.

In addition, this requires that if the variable was already in a register, the register must be freed and the location of the variable must be set to MEMORY.

Parameters

reg	The register to store to the variable
name	The variable name to store a value to

5.2.3.35 void AddressTable::Store (string *reg*, string *name*)

This function stores the value of a register into a variable in memory.

In addition, this requires that if the variable was already in a register, the register must be freed and the location of the variable must be set to MEMORY.

Parameters

reg	The register to store to the variable
name	The variable name to store a value to

Definition at line 157 of file AddressTable.cpp.

5.2.3.36 void AddressTable::Store (string *reg*, string *name*)

This function stores the value of a register into a variable in memory.

In addition, this requires that if the variable was already in a register, the register must be freed and the location of the variable must be set to MEMORY.

Parameters

reg	The register to store to the variable
name	The variable name to store a value to

5.2.3.37 static void AddressTable::Update (RegAllocTable ∗ *src*, void ∗ *data*, void ∗ *context*) [static]

This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.

Parameters

src	A reference to the register allocation table
data	This will ALWAYS be a null pointer
context	This is a pointer to the AddressTable object

5.2.3.38 static void AddressTable::Update (RegAllocTable ∗ *src*, void ∗ *data*, void ∗ *context*) [static]

This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.

Parameters

src	A reference to the register allocation table
data	This will ALWAYS be a null pointer
context	This is a pointer to the AddressTable object

5.2.3.39 void AddressTable::Update (RegAllocTable ∗ *src*, void ∗ *data*, void ∗ *context*) [static]

This static function receives the event notification from the register allocation table and fires the update function for the provided address table context object.

Parameters

src	A reference to the register allocation table
data	This will ALWAYS be a null pointer
context	This is a pointer to the AddressTable object

Definition at line 183 of file AddressTable.cpp.

5.2.3.40 void AddressTable::UpdateRegisters (RegAllocTable ∗ *src*)

This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.

Parameters

src	A reference to the register allocation table

5.2.3.41 void AddressTable::UpdateRegisters (RegAllocTable ∗ *src*)

This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.

Parameters

src	A reference to the register allocation table

Definition at line 190 of file AddressTable.cpp.

5.2.3.42 void AddressTable::UpdateRegisters (RegAllocTable ∗ *src*)

This function is responsible for updating the register locations stored in the address table for any variables currently in a register after any operations which require register changes.

Parameters

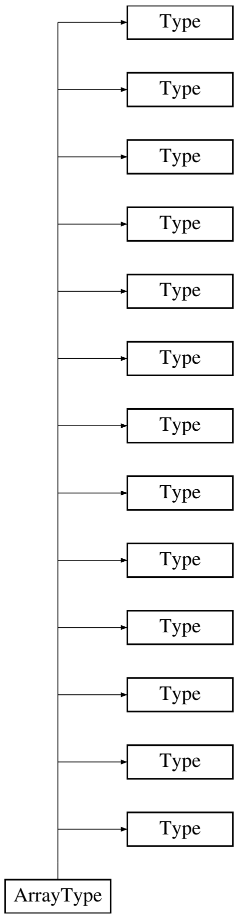
src	A reference to the register allocation table

The documentation for this class was generated from the following files:

- mips/AddressTable.h
- mips/AddressTable.cpp

5.3 ArrayType Class Reference

Inheritance diagram for ArrayType:


Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
ArrayType

Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **ArrayType** (Type ∗baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type ∗ **GetBase** ()
- void **SetBase** (Type ∗base)
- bool **CheckType** (ArrayType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type ∗baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type ∗ **GetBase** ()
- void **SetBase** (Type ∗base)
- bool **CheckType** (ArrayType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type ∗baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type ∗ **GetBase** ()
- void **SetBase** (Type ∗base)
- bool **CheckType** (ArrayType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type ∗baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type ∗ **GetBase** ()
- void **SetBase** (Type ∗base)

- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **ArrayType** (Type *baseType, string name, int dims)
- int **SetCapacity** (int cap)
- int **GetCapacity** (int dim)
- Type * **GetBase** ()
- void **SetBase** (Type *base)
- bool **CheckType** (ArrayType *rhs, bool &isConvertable, CONVERSIONTYPE &t)

- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- Type $*$ **baseType**
- int **dimensions**
- vector$<$ int $>$ **capacities**
- string **name**
- int **size**

5.3.1 Detailed Description

Definition at line 128 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.4 AST Class Reference

Abstract syntax tree node type.

`#include <Ast.h>`

Inheritance diagram for AST:

Public Member Functions

- AST ()

  AST default constructor.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- virtual void Visit ()

  This function is responsible for tree traversals.
- AST ()

  AST default constructor.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- virtual void Visit ()

  This function is responsible for tree traversals.
- AST ()

  AST default constructor.

- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- virtual void Visit ()

  This function is responsible for tree traversals.
- AST ()

  AST default constructor.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- virtual void Visit ()

  This function is responsible for tree traversals.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.4.1 Detailed Description

Abstract syntax tree node type.

Definition at line 19 of file Ast.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AST::AST ( ) [inline]

AST default constructor.

This default constructor ensures that every node has a label and a unique identifier (used for visualizing the tree)

Definition at line 27 of file Ast.h.

5.4.2.2 AST::AST ( ) [inline]

AST default constructor.

This default constructor ensures that every node has a label and a unique identifier (used for visualizing the tree)

Definition at line 27 of file CParser.yy.

5.4.2.3 AST::AST ( ) [inline]

AST default constructor.

This default constructor ensures that every node has a label and a unique identifier (used for visualizing the tree)

Definition at line 27 of file CParser.yy.

5.4.2.4 AST::AST ( ) [inline]

AST default constructor.

This default constructor ensures that every node has a label and a unique identifier (used for visualizing the tree)

Definition at line 27 of file CScanner.ll.

5.4.3 Member Function Documentation

5.4.3.1 string AST::getLabel ( ) [inline]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.4.3.2 string AST::getLabel ( ) [inline]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.4.3.3 string AST::getLabel ( ) `[inline]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.4.3.4 string AST::getLabel ( ) `[inline]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.4.3.5 int AST::getUID ( ) `[inline]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.4.3.6 int AST::getUID ( ) `[inline]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.4.3.7 int AST::getUID ( ) `[inline]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.4.3.8 int AST::getUID ( ) `[inline]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.4.3.9 void AST::setLabel (string *l*) `[inline]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.4.3.10 void AST::setLabel (string *l*) `[inline]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.4.3.11 void AST::setLabel (string *l*) `[inline]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.4.3.12 void AST::setLabel (string *l*) `[inline]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.4.3.13 virtual void AST::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented in AstFuncDef, AstFuncDef, AstFuncDef, AstFuncDef, AstDeclarator, AstDeclarator, AstDeclarator, AstDeclarator, AstStructDecl, AstStructDecl, AstStructDecl, AstStructDecl, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, EnumSpecifier, EnumSpecifier, EnumSpecifier, EnumSpecifier, AstEnumList, AstEnumList, AstEnumList, AstEnumList, AstEnumerator, AstEnumerator, AstEnumerator, AstEnumerator, AstInitializer, AstInitializer, AstInitializer, AstInitializer, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstDecl, AstDecl, AstDecl, AstDecl, AstDeclList, AstDeclList, AstDeclList, AstDeclList, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstStatementList, AstStatementList, AstStatementList, AstStatementList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstTrans, AstTrans, AstTrans, AstTrans, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeSpeci, AstTypeSpeci, AstTypeSpeci, AstTypeSpeci, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstIDList, AstIDList, AstIDList, AstIDList, AstInitList, AstInitList, AstInitList, AstInitList, AstParamDec, AstParamDec, AstParamDec, AstParamDec, AstParamList, AstParamList, AstParamList, AstParamList, AstPointer, AstPointer, AstPointer, AstPointer, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStatement, AstStatement, AstStatement, AstStatement, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstSelection, AstSelection, AstSelection, AstSelection, AstIfElse, AstIfElse, AstIfElse, AstIfElse, AstSwitch, AstSwitch, AstSwitch, AstSwitch, AstIteration, AstIteration, AstIteration, AstIteration, AstFor, AstFor, AstFor, AstFor, AstWhile, AstWhile, AstWhile, AstWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstJump, AstJump, AstJump, AstJump, AstGoto, AstGoto, AstGoto, AstGoto, AstBreak, AstBreak, AstBreak, AstBreak, AstContinue, AstContinue, AstContinue, AstContinue, AstReturn, AstReturn, AstReturn, AstReturn, AstExpression, AstExpression, AstExpression, AstExpression, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignOp, AstAssignOp, AstAssignOp, AstAssignOp, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstORExpr, AstORExpr, AstORExpr, AstORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstCastExpr, AstCastExpr, AstCastExpr, AstCastExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstPostfixExpr, AstPostfixExpr, AstPostfixExpr, AstPostfixExpr, AstArgExprList, AstArgExprList, AstArgExprList, AstArgExprList, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstID, AstID, AstID, AstID, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstConstant, AstConstant, AstConstant, AstConstant, AstString, AstString, AstString, AstString, AstTypeName, AstTypeName, AstTypeName, and AstTypeName.

Definition at line 74 of file CParser.yy.

5.4.3.14 virtual void AST::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented in AstFuncDef, AstFuncDef, AstFuncDef, AstFuncDef, AstDeclarator, AstDeclarator, AstDeclarator, AstDeclarator, AstStructDecl, AstStructDecl, AstStructDecl, AstStructDecl, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, EnumSpecifier, EnumSpecifier, EnumSpecifier, EnumSpecifier, AstEnumList, AstEnumList, AstEnumList, AstEnumList, AstEnumerator, AstEnumerator, AstEnumerator, AstEnumerator, AstInitializer, AstInitializer, AstInitializer, AstInitializer, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstDecl, AstDecl, AstDecl, AstDecl, AstDeclList, AstDeclList, AstDeclList, AstDeclList, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstStatementList, AstStatementList, AstStatementList, AstStatementList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstTrans, AstTrans,

AstTrans, AstTrans, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeSpeci, AstType-Speci, AstTypeSpeci, AstTypeSpeci, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstIDList, AstIDList, AstIDList, AstIDList, AstInitList, AstInitList, AstInitList, AstInitList, AstParamDec, AstParam-Dec, AstParamDec, AstParamDec, AstParamList, AstParamList, AstParamList, AstParamList, AstPointer, Ast-Pointer, AstPointer, AstPointer, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructUniSpeci, Ast-StructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStatement, AstStatement, AstStatement, AstStatement, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstSelection, Ast-Selection, AstSelection, AstSelection, AstIfElse, AstIfElse, AstIfElse, AstIfElse, AstSwitch, AstSwitch, AstSwitch, AstSwitch, AstIteration, AstIteration, AstIteration, AstIteration, AstFor, AstFor, AstFor, AstFor, AstWhile, Ast-While, AstWhile, AstWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstJump, AstJump, AstJump, AstJump, AstGoto, AstGoto, AstGoto, AstGoto, AstBreak, AstBreak, AstBreak, AstBreak, AstContinue, Ast-Continue, AstContinue, AstContinue, AstReturn, AstReturn, AstReturn, AstReturn, AstExpression, AstExpression, AstExpression, AstExpression, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignOp, Ast-AssignOp, AstAssignOp, AstAssignOp, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstLogicOrExpr, AstLogicOr-Expr, AstLogicOrExpr, AstLogicOrExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstORExpr, AstORExpr, AstORExpr, AstORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstAnd-Expr, AstAndExpr, AstAndExpr, AstAndExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstRelExpr, AstRel-Expr, AstRelExpr, AstRelExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstCastExpr, AstCastExpr, Ast-CastExpr, AstCastExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstPostfixExpr, AstPostfix-Expr, AstPostfixExpr, AstPostfixExpr, AstArgExprList, AstArgExprList, AstArgExprList, AstArgExprList, AstPrimary-Expr, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstID, AstID, AstID, AstID, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstConstant, AstConstant, AstConstant, AstConstant, AstString, AstString, AstString, AstString, AstTypeName, AstTypeName, AstTypeName, and AstTypeName.

Definition at line 74 of file Ast.h.

5.4.3.15 virtual void AST::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented in AstFuncDef, AstFuncDef, AstFuncDef, AstFuncDef, AstDeclarator, AstDeclarator, AstDeclarator, AstDeclarator, AstStructDecl, AstStructDecl, AstStructDecl, AstStructDecl, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, EnumSpecifier, EnumSpecifier, EnumSpecifier, EnumSpecifier, AstEnumList, AstEnumList, AstEnumList, AstEnumList, AstEnumerator, AstEnumerator, AstEnumerator, AstEnumerator, Ast-Initializer, AstInitializer, AstInitializer, AstInitializer, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInit-Declarator, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstTypeParam-List, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstDecl, AstDecl, AstDecl, AstDecl, AstDeclList, AstDeclList, AstDeclList, AstDeclList, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDecSpeci, Ast-DirectDecl, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstStatementList, AstStatementList, AstStatementList, AstStatementList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstTrans, AstTrans, AstTrans, AstTrans, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeSpeci, AstType-Speci, AstTypeSpeci, AstTypeSpeci, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstIDList, AstIDList, AstIDList, AstIDList, AstInitList, AstInitList, AstInitList, AstInitList, AstParamDec, AstParam-Dec, AstParamDec, AstParamDec, AstParamList, AstParamList, AstParamList, AstParamList, AstPointer, Ast-Pointer, AstPointer, AstPointer, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructUniSpeci, Ast-StructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStatement, AstStatement, AstStatement, AstStatement, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstSelection, Ast-Selection, AstSelection, AstSelection, AstIfElse, AstIfElse, AstIfElse, AstIfElse, AstSwitch, AstSwitch, AstSwitch, AstSwitch, AstIteration, AstIteration, AstIteration, AstIteration, AstFor, AstFor, AstFor, AstFor, AstWhile, Ast-

While, AstWhile, AstWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstJump, AstJump, AstJump, AstJump, AstGoto, AstGoto, AstGoto, AstGoto, AstBreak, AstBreak, AstBreak, AstBreak, AstContinue, AstContinue, AstContinue, AstContinue, AstReturn, AstReturn, AstReturn, AstReturn, AstExpression, AstExpression, AstExpression, AstExpression, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignOp, AstAssignOp, AstAssignOp, AstAssignOp, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstORExpr, AstORExpr, AstORExpr, AstORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstCastExpr, AstCastExpr, AstCastExpr, AstCastExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstPostfixExpr, AstPostfixExpr, AstPostfixExpr, AstPostfixExpr, AstArgExprList, AstArgExprList, AstArgExprList, AstArgExprList, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstID, AstID, AstID, AstID, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstConstant, AstConstant, AstConstant, AstConstant, AstString, AstString, AstString, AstString, AstTypeName, AstTypeName, AstTypeName, and AstTypeName.

Definition at line 74 of file CParser.yy.

5.4.3.16 virtual void AST::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented in AstFuncDef, AstFuncDef, AstFuncDef, AstFuncDef, AstDeclarator, AstDeclarator, AstDeclarator, AstDeclarator, AstStructDecl, AstStructDecl, AstStructDecl, AstStructDecl, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, AstSpeciQualList, EnumSpecifier, EnumSpecifier, EnumSpecifier, EnumSpecifier, AstEnumList, AstEnumList, AstEnumList, AstEnumList, AstEnumerator, AstEnumerator, AstEnumerator, AstEnumerator, AstInitializer, AstInitializer, AstInitializer, AstInitializer, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclarator, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstInitDeclList, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstDirectAbsDecl, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstTypeParamList, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstAbstractDecl, AstDecl, AstDecl, AstDecl, AstDecl, AstDeclList, AstDeclList, AstDeclList, AstDeclList, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDecSpeci, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstDirectDecl, AstStatementList, AstStatementList, AstStatementList, AstStatementList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstStructDeclList, AstTrans, AstTrans, AstTrans, AstTrans, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeQualList, AstTypeSpeci, AstTypeSpeci, AstTypeSpeci, AstTypeSpeci, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstDeclarationList, AstIDList, AstIDList, AstIDList, AstIDList, AstInitList, AstInitList, AstInitList, AstInitList, AstParamDec, AstParamDec, AstParamDec, AstParamDec, AstParamList, AstParamList, AstParamList, AstParamList, AstPointer, AstPointer, AstPointer, AstPointer, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclarator, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructDeclatorList, AstStructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStructUniSpeci, AstStatement, AstStatement, AstStatement, AstStatement, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstLabeledStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstExprStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstCompoundStmt, AstSelection, AstSelection, AstSelection, AstSelection, AstIfElse, AstIfElse, AstIfElse, AstIfElse, AstSwitch, AstSwitch, AstSwitch, AstSwitch, AstIteration, AstIteration, AstIteration, AstIteration, AstFor, AstFor, AstFor, AstFor, AstWhile, AstWhile, AstWhile, AstWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstDoWhile, AstJump, AstJump, AstJump, AstJump, AstGoto, AstGoto, AstGoto, AstGoto, AstBreak, AstBreak, AstBreak, AstBreak, AstContinue, AstContinue, AstContinue, AstContinue, AstReturn, AstReturn, AstReturn, AstReturn, AstExpression, AstExpression, AstExpression, AstExpression, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignExpr, AstAssignOp, AstAssignOp, AstAssignOp, AstAssignOp, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConstantExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstConditionalExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicOrExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstLogicAndExpr, AstORExpr, AstORExpr, AstORExpr, AstORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstXORExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstAndExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstEqExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstRelExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstShiftExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstAddExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstMultExpr, AstCastExpr, AstCastExpr, Ast-

CastExpr, AstCastExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstUnaryExpr, AstPostfixExpr, AstPostfix-Expr, AstPostfixExpr, AstPostfixExpr, AstArgExprList, AstArgExprList, AstArgExprList, AstArgExprList, AstPrimary-Expr, AstPrimaryExpr, AstPrimaryExpr, AstPrimaryExpr, AstID, AstID, AstID, AstID, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstUnaryOp, AstConstant, AstConstant, AstConstant, AstConstant, AstString, AstString, AstString, AstString, AstTypeName, AstTypeName, AstTypeName, and AstTypeName.

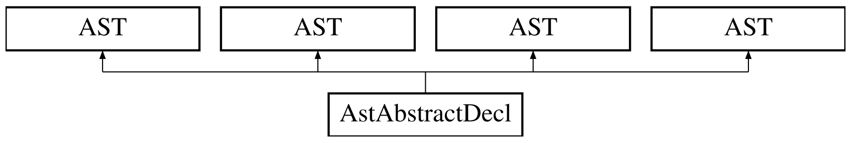
Definition at line 74 of file CScanner.ll.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.5 AstAbstractDecl Class Reference

Inheritance diagram for AstAbstractDecl:



Public Member Functions

- **AstAbstractDecl** (AstPointer ∗pointer, AstDirectAbsDecl ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAbstractDecl** (AstPointer ∗pointer, AstDirectAbsDecl ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAbstractDecl** (AstPointer ∗pointer, AstDirectAbsDecl ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAbstractDecl** (AstPointer ∗pointer, AstDirectAbsDecl ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstPointer $*$ **pointer**
- AstDirectAbsDecl $*$ **dec**

5.5.1 Detailed Description

Definition at line 1297 of file Ast.h.

5.5.2 Member Function Documentation

5.5.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.5.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.5.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.5.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.5.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.5.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.5.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.5.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.5.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.5.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.5.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.5.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.5.2.13 void AstAbstractDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1310 of file Ast.h.

5.5.2.14 void AstAbstractDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1310 of file CParser.yy.

5.5.2.15 void AstAbstractDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1310 of file CParser.yy.

5.5.2.16 void AstAbstractDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

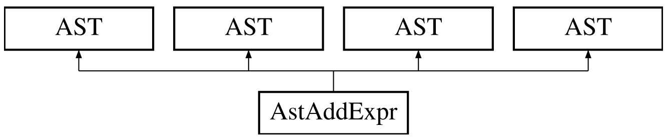
Definition at line 1310 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.6 AstAddExpr Class Reference

Inheritance diagram for AstAddExpr:



Public Types

- enum **Operator** {
  NONE, **PLUS**, **MINUS**, **NONE**,
  PLUS, **MINUS**, **NONE**, **PLUS**,
  MINUS, **NONE**, **PLUS**, **MINUS** }
- enum **Operator** {
  NONE, **PLUS**, **MINUS**, **NONE**,
  PLUS, **MINUS**, **NONE**, **PLUS**,
  MINUS, **NONE**, **PLUS**, **MINUS** }
- enum **Operator** {
  NONE, **PLUS**, **MINUS**, **NONE**,
  PLUS, **MINUS**, **NONE**, **PLUS**,
  MINUS, **NONE**, **PLUS**, **MINUS** }
- enum **Operator** {
  NONE, **PLUS**, **MINUS**, **NONE**,
  PLUS, **MINUS**, **NONE**, **PLUS**,
  MINUS, **NONE**, **PLUS**, **MINUS** }

Public Member Functions

- **AstAddExpr** (AstMultExpr ∗m)
- **AstAddExpr** (AstAddExpr ∗a, Operator o, AstMultExpr ∗m)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAddExpr** (AstMultExpr ∗m)
- **AstAddExpr** (AstAddExpr ∗a, Operator o, AstMultExpr ∗m)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAddExpr** (AstMultExpr ∗m)
- **AstAddExpr** (AstAddExpr ∗a, Operator o, AstMultExpr ∗m)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAddExpr** (AstMultExpr ∗m)
- **AstAddExpr** (AstAddExpr ∗a, Operator o, AstMultExpr ∗m)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstAddExpr::Operator **op**
- Type $*$ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstMultExpr ∗ **mult**
- AstAddExpr ∗ **add**

5.6.1 Detailed Description

Definition at line 387 of file Ast.h.

5.6.2 Member Function Documentation

5.6.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.6.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.6.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.6.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.6.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.6.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.6.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.6.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.6.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.6.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.6.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.6.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.6.2.13 void AstAddExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 932 of file Ast.cpp.

5.6.2.14 void AstAddExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.6.2.15 void AstAddExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.6.2.16 void AstAddExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

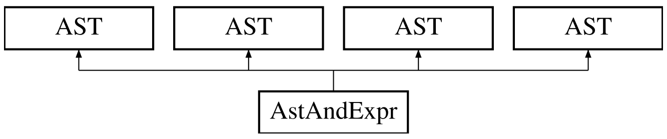
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.7 AstAndExpr Class Reference

Inheritance diagram for AstAndExpr:



Public Member Functions

- **AstAndExpr** (AstEqExpr ∗e)
- **AstAndExpr** (AstAndExpr ∗a, AstEqExpr ∗e)
- void Visit ()

    This function is responsible for tree traversals.
- **AstAndExpr** (AstEqExpr ∗e)
- **AstAndExpr** (AstAndExpr ∗a, AstEqExpr ∗e)
- void Visit ()

    This function is responsible for tree traversals.
- **AstAndExpr** (AstEqExpr ∗e)
- **AstAndExpr** (AstAndExpr ∗a, AstEqExpr ∗e)
- void Visit ()

    This function is responsible for tree traversals.
- **AstAndExpr** (AstEqExpr ∗e)
- **AstAndExpr** (AstAndExpr ∗a, AstEqExpr ∗e)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type $*$ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstEqExpr $*$ **eq**
- AstAndExpr $*$ **a**

5.7.1 Detailed Description

Definition at line 485 of file Ast.h.

5.7.2 Member Function Documentation

5.7.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.7.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.7.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.7.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.7.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.7.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.7.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.7.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.7.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.7.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.7.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.7.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.7.2.13 void AstAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1271 of file Ast.cpp.

5.7.2.14 void AstAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.7.2.15 void AstAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.7.2.16 void AstAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

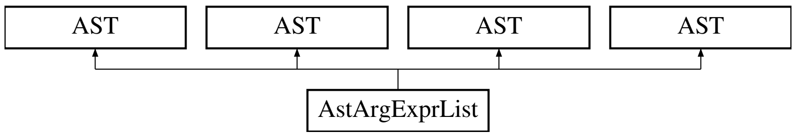
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.8 AstArgExprList Class Reference

Inheritance diagram for AstArgExprList:



Public Member Functions

- **AstArgExprList** (AstArgExprList ∗list, AstAssignExpr ∗expr)
- **AstArgExprList** (AstAssignExpr ∗expr)
- void Visit ()

  This function is responsible for tree traversals.
- **AstArgExprList** (AstArgExprList ∗list, AstAssignExpr ∗expr)
- **AstArgExprList** (AstAssignExpr ∗expr)
- void Visit ()

  This function is responsible for tree traversals.
- **AstArgExprList** (AstArgExprList ∗list, AstAssignExpr ∗expr)
- **AstArgExprList** (AstAssignExpr ∗expr)
- void Visit ()

  This function is responsible for tree traversals.
- **AstArgExprList** (AstArgExprList ∗list, AstAssignExpr ∗expr)
- **AstArgExprList** (AstAssignExpr ∗expr)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstArgExprList $*$ **list**
- AstAssignExpr $*$ **expr**
- bool **isLastItem**

5.8.1 Detailed Description

Definition at line 242 of file Ast.h.

5.8.2 Member Function Documentation

5.8.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.8.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.8.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.8.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.8.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.8.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.8.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.8.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.8.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.8.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.8.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.8.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.8.2.13 void AstArgExprList::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 145 of file Ast.cpp.

5.8.2.14 void AstArgExprList::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.8.2.15 void AstArgExprList::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.8.2.16 void AstArgExprList::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

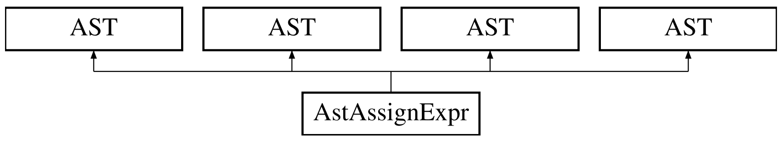
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.9 AstAssignExpr Class Reference

Inheritance diagram for AstAssignExpr:



Public Member Functions

- **AstAssignExpr** (AstConditionalExpr ∗c)
- **AstAssignExpr** (AstUnaryExpr ∗u, AstAssignOp ∗a, AstAssignExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAssignExpr** (AstConditionalExpr ∗c)
- **AstAssignExpr** (AstUnaryExpr ∗u, AstAssignOp ∗a, AstAssignExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.

- **AstAssignExpr** (AstConditionalExpr ∗c)
- **AstAssignExpr** (AstUnaryExpr ∗u, AstAssignOp ∗a, AstAssignExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAssignExpr** (AstConditionalExpr ∗c)
- **AstAssignExpr** (AstUnaryExpr ∗u, AstAssignOp ∗a, AstAssignExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstConditionalExpr ∗ **cond**
- AstUnaryExpr ∗ **uni**
- AstAssignOp ∗ **op**
- AstAssignExpr ∗ **expr**

5.9.1 Detailed Description

Definition at line 614 of file Ast.h.

5.9.2 Member Function Documentation

5.9.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.9.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.9.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.9.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.9.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.9.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.9.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.9.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.9.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.9.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.9.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.9.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.9.2.13 void AstAssignExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1728 of file Ast.cpp.

5.9.2.14 void AstAssignExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.9.2.15 void AstAssignExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.9.2.16 void AstAssignExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

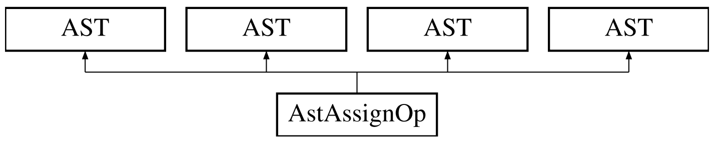
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.10 AstAssignOp Class Reference

Inheritance diagram for AstAssignOp:



Public Types

- enum **Operator** {
EQ, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**,
ADD_ASSIGN, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**,
AND_ASSIGN, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**,
MUL_ASSIGN, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**,
SUB_ASSIGN, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**,
XOR_ASSIGN, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**,
DIV_ASSIGN, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**,
LEFT_ASSIGN, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**,
OR_ASSIGN, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**,
MOD_ASSIGN, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**,
RIGHT_ASSIGN, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN** }
- enum **Operator** {

EQ, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN** }

- enum **Operator** {
  EQ, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN** }
- enum **Operator** {
  EQ, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN**, **EQ**, **MUL_ASSIGN**, **DIV_ASSIGN**, **MOD_ASSIGN**, **ADD_ASSIGN**, **SUB_ASSIGN**, **LEFT_ASSIGN**, **RIGHT_ASSIGN**, **AND_ASSIGN**, **XOR_ASSIGN**, **OR_ASSIGN** }

Public Member Functions

- **AstAssignOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAssignOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAssignOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstAssignOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Operator **op**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.10.1 Detailed Description

Definition at line 583 of file Ast.h.

5.10.2 Member Function Documentation

5.10.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.10.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.10.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.10.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.10.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.10.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.10.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.10.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.10.2.9 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.10.2.10 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.10.2.11 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.10.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.10.2.13 void AstAssignOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1652 of file Ast.cpp.

5.10.2.14 void AstAssignOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.10.2.15 void AstAssignOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.10.2.16 void AstAssignOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

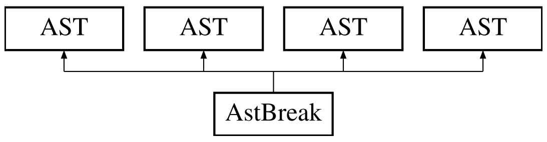
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.11 AstBreak Class Reference

Inheritance diagram for AstBreak:



Public Member Functions

- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

5.11.1 Detailed Description

Definition at line 658 of file Ast.h.

5.11.2 Member Function Documentation

5.11.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.11.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.11.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.11.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.11.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.11.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.11.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.11.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.11.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.11.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.11.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.11.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.11.2.13 void AstBreak::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1924 of file Ast.cpp.

5.11.2.14 void AstBreak::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.11.2.15 void AstBreak::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.11.2.16 void AstBreak::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

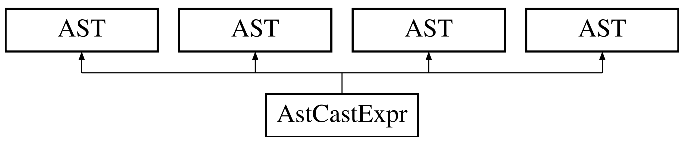
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.12 AstCastExpr Class Reference

Inheritance diagram for AstCastExpr:



Public Member Functions

- **AstCastExpr** (AstUnaryExpr ∗u)
- **AstCastExpr** (AstTypeName ∗t, AstCastExpr ∗c)
- void Visit ()

    This function is responsible for tree traversals.
- **AstCastExpr** (AstUnaryExpr ∗u)
- **AstCastExpr** (AstTypeName ∗t, AstCastExpr ∗c)
- void Visit ()

    This function is responsible for tree traversals.

- **AstCastExpr** (AstUnaryExpr ∗u)
- **AstCastExpr** (AstTypeName ∗t, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- **AstCastExpr** (AstUnaryExpr ∗u)
- **AstCastExpr** (AstTypeName ∗t, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstUnaryExpr ∗ **uniexpr**
- AstCastExpr ∗ **cast**
- AstTypeName ∗ **tname**

5.12.1 Detailed Description

Definition at line 344 of file Ast.h.

5.12.2 Member Function Documentation

5.12.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.12.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.12.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.12.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.12.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.12.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.12.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.12.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.12.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.12.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.12.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.12.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.12.2.13 void AstCastExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 598 of file Ast.cpp.

5.12.2.14 void AstCastExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.12.2.15 void AstCastExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.12.2.16 void AstCastExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

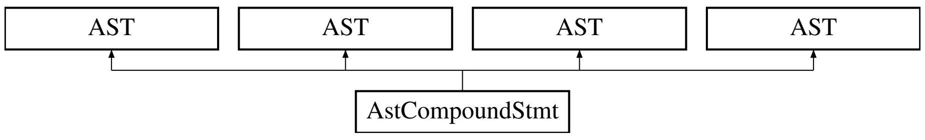
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.13 AstCompoundStmt Class Reference

Inheritance diagram for AstCompoundStmt:



Public Member Functions

- **AstCompoundStmt** (AstDeclarationList ∗d, AstStatementList ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstCompoundStmt** (AstDeclarationList ∗d, AstStatementList ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstCompoundStmt** (AstDeclarationList ∗d, AstStatementList ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstCompoundStmt** (AstDeclarationList ∗d, AstStatementList ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstStatementList ∗ **stmtList**
- AstDeclarationList ∗ **declList**

5.13.1 Detailed Description

Definition at line 794 of file Ast.h.

5.13.2 Member Function Documentation

5.13.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.13.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.13.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.13.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.13.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.13.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.13.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.13.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.13.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.13.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.13.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.13.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.13.2.13 void AstCompoundStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2443 of file Ast.cpp.

5.13.2.14 void AstCompoundStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.13.2.15 void AstCompoundStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.13.2.16 void AstCompoundStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

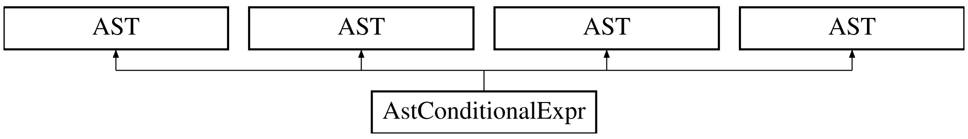
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.14 AstConditionalExpr Class Reference

Inheritance diagram for AstConditionalExpr:



Public Member Functions

- **AstConditionalExpr** (AstLogicOrExpr ∗o)
- **AstConditionalExpr** (AstLogicOrExpr ∗o, AstExpression ∗e, AstConditionalExpr ∗ce)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConditionalExpr** (AstLogicOrExpr ∗o)
- **AstConditionalExpr** (AstLogicOrExpr ∗o, AstExpression ∗e, AstConditionalExpr ∗ce)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConditionalExpr** (AstLogicOrExpr ∗o)
- **AstConditionalExpr** (AstLogicOrExpr ∗o, AstExpression ∗e, AstConditionalExpr ∗ce)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConditionalExpr** (AstLogicOrExpr ∗o)
- **AstConditionalExpr** (AstLogicOrExpr ∗o, AstExpression ∗e, AstConditionalExpr ∗ce)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type $*$ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstLogicOrExpr $*$ **o**
- AstExpression $*$ **e**
- AstConditionalExpr $*$ **ce**

5.14.1 Detailed Description

Definition at line 555 of file Ast.h.

5.14.2 Member Function Documentation

5.14.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.14.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.14.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.14.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.14.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.14.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.14.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.14.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.14.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.14.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.14.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.14.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.14.2.13 void AstConditionalExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1593 of file Ast.cpp.

5.14.2.14 void AstConditionalExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.14.2.15 void AstConditionalExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.14.2.16 void AstConditionalExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

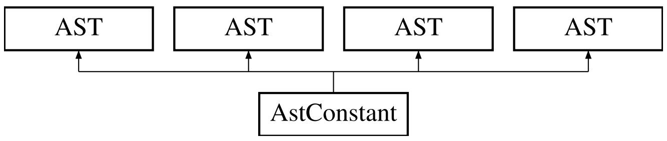
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.15 AstConstant Class Reference

Inheritance diagram for AstConstant:



Public Types

- enum **ConstType** {
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM** }
- enum **ConstType** {
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM** }
- enum **ConstType** {
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM** }
- enum **ConstType** {
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM**,
  INT, **CHAR**, **FLOAT**, **ENUM** }

Public Member Functions

- **AstConstant** (int val)
- **AstConstant** (string val)
- **AstConstant** (double val)
- **AstConstant** (int val, string name, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConstant** (int val)
- **AstConstant** (string val)
- **AstConstant** (double val)
- **AstConstant** (int val, string name, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConstant** (int val)
- **AstConstant** (string val)
- **AstConstant** (double val)
- **AstConstant** (int val, string name, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.

- **AstConstant** (int val)
- **AstConstant** (string val)
- **AstConstant** (double val)
- **AstConstant** (int val, string name, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- ConstType **type**
- int **ival**
- string **str**
- double **dval**
- Type ∗ **etype**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.15.1 Detailed Description

Definition at line 140 of file Ast.h.

5.15.2 Member Function Documentation

5.15.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.15.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.15.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.15.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.15.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.15.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.15.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.15.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.15.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.15.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.15.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.15.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.15.2.13 void AstConstant::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 719 of file Ast.cpp.

5.15.2.14 void AstConstant::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.15.2.15 void AstConstant::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.15.2.16 void AstConstant::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

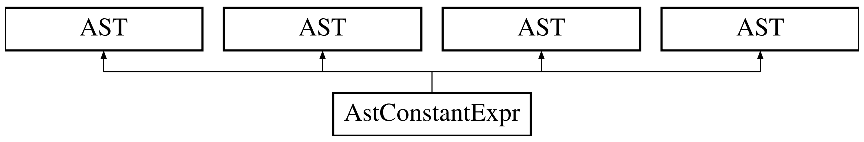
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.16 AstConstantExpr Class Reference

Inheritance diagram for AstConstantExpr:



Public Member Functions

- **AstConstantExpr** (AstConditionalExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConstantExpr** (AstConditionalExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConstantExpr** (AstConditionalExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstConstantExpr** (AstConditionalExpr ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstConditionalExpr ∗ **expr**

5.16.1 Detailed Description

Definition at line 571 of file Ast.h.

5.16.2 Member Function Documentation

5.16.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.16.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.16.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.16.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.16.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.16.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.16.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.16.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.16.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.16.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.16.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.16.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.16.2.13 void AstConstantExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1634 of file Ast.cpp.

5.16.2.14 void AstConstantExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.16.2.15 void AstConstantExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.16.2.16 void AstConstantExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

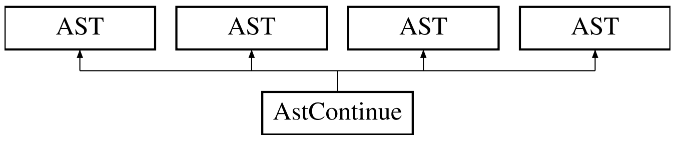
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.17 AstContinue Class Reference

Inheritance diagram for AstContinue:



Public Member Functions

- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

5.17.1 Detailed Description

Definition at line 651 of file Ast.h.

5.17.2 Member Function Documentation

5.17.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.17.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.17.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.17.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.17.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.17.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.17.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.17.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.17.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.17.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.17.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.17.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.17.2.13 void AstContinue::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1910 of file Ast.cpp.

5.17.2.14 void AstContinue::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.17.2.15 void AstContinue::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.17.2.16 void AstContinue::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

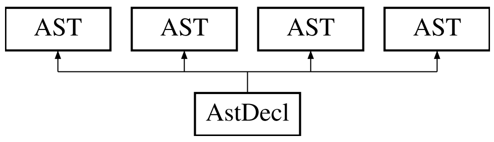
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.18 AstDecl Class Reference

Inheritance diagram for AstDecl:



Public Member Functions

- **AstDecl** (AstDecSpeci ∗speci, AstInitSpecList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDecl** (AstDecSpeci ∗speci, AstInitSpecList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDecl** (AstDecSpeci ∗speci, AstInitSpecList ∗list)
- void Visit ()

This function is responsible for tree traversals.
- **AstDecl** (AstDecSpeci ∗speci, AstInitSpecList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDecSpeci ∗ **speci**
- AstInitSpecList ∗ **list**

5.18.1 Detailed Description

Definition at line 1275 of file Ast.h.

5.18.2 Member Function Documentation

5.18.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.18.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.18.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.18.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.18.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.18.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.18.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.18.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.18.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.18.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.18.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.18.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.18.2.13 void AstDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1288 of file Ast.h.

5.18.2.14 void AstDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1288 of file CParser.yy.

5.18.2.15 void AstDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1288 of file CParser.yy.

5.18.2.16 void AstDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

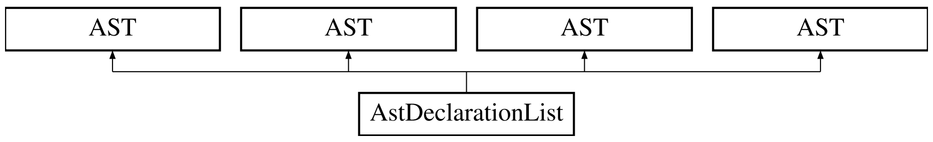
Definition at line 1288 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.19 AstDeclarationList Class Reference

Inheritance diagram for AstDeclarationList:



Public Member Functions

- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.

- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.19.1 Detailed Description

Definition at line 1061 of file Ast.h.

5.19.2 Member Function Documentation

5.19.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.19.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.19.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.19.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.19.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.19.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.19.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.19.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.19.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.19.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.19.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.19.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.19.2.13 void AstDeclarationList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1064 of file Ast.h.

5.19.2.14 void AstDeclarationList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1064 of file CParser.yy.

5.19.2.15 void AstDeclarationList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1064 of file CParser.yy.

5.19.2.16 void AstDeclarationList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

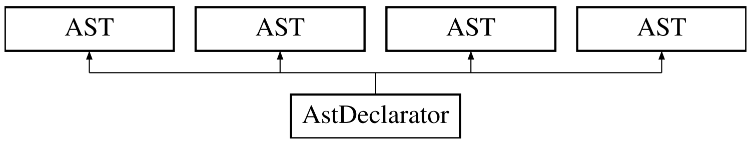
Definition at line 1064 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.20 AstDeclarator Class Reference

Inheritance diagram for AstDeclarator:



Public Member Functions

- **AstDeclarator** (AstPointer ∗pointer, AstDirectDecl ∗decl)
- AstDirectDecl ∗ **GetDirectDecl** ()
- void Visit ()

  This function is responsible for tree traversals.
- **AstDeclarator** (AstPointer ∗pointer, AstDirectDecl ∗decl)
- AstDirectDecl ∗ **GetDirectDecl** ()
- void Visit ()

  This function is responsible for tree traversals.
- **AstDeclarator** (AstPointer ∗pointer, AstDirectDecl ∗decl)
- AstDirectDecl ∗ **GetDirectDecl** ()
- void Visit ()

  This function is responsible for tree traversals.
- **AstDeclarator** (AstPointer ∗pointer, AstDirectDecl ∗decl)
- AstDirectDecl ∗ **GetDirectDecl** ()
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstPointer $*$ **pointer**
- AstDirectDecl $*$ **decl**

5.20.1 Detailed Description

Definition at line 1516 of file Ast.h.

5.20.2 Member Function Documentation

5.20.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.20.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.20.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.20.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.20.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.20.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.20.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.20.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.20.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.20.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.20.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.20.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.20.2.13 void AstDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1534 of file Ast.h.

5.20.2.14 void AstDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1534 of file CParser.yy.

5.20.2.15 void AstDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1534 of file CParser.yy.

5.20.2.16 void AstDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

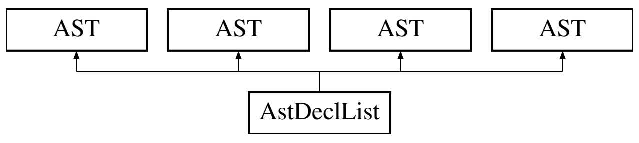
Definition at line 1534 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.21 AstDeclList Class Reference

Inheritance diagram for AstDeclList:



Public Member Functions

- **AstDeclList** (AstDeclList ∗list, AstDecl ∗decl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDeclList** (AstDeclList ∗list, AstDecl ∗decl)
- void Visit ()

This function is responsible for tree traversals.
- **AstDeclList** (AstDeclList ∗list, AstDecl ∗decl)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDeclList** (AstDeclList ∗list, AstDecl ∗decl)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""

- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDecl ∗ **decl**
- AstDeclList ∗ **list**

5.21.1 Detailed Description

Definition at line 1251 of file Ast.h.

5.21.2 Member Function Documentation

5.21.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.21.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.21.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.21.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.21.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.21.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.21.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.21.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.21.2.9 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.21.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.21.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.21.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.21.2.13 void AstDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1264 of file Ast.h.

5.21.2.14 void AstDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1264 of file CParser.yy.

5.21.2.15 void AstDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1264 of file CParser.yy.

5.21.2.16 void AstDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

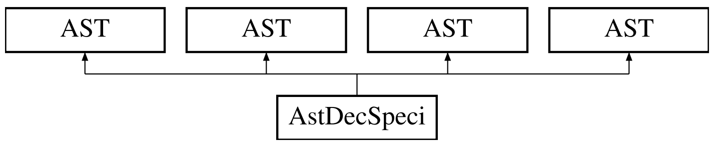
Definition at line 1264 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.22 AstDecSpeci Class Reference

Inheritance diagram for AstDecSpeci:



Public Member Functions

- **AstDecSpeci** (string str_class, string typeq, AstDecSpeci ∗speci, AstTypeSpeci ∗typeSpeci)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDecSpeci** (string str_class, string typeq, AstDecSpeci ∗speci, AstTypeSpeci ∗typeSpeci)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDecSpeci** (string str_class, string typeq, AstDecSpeci ∗speci, AstTypeSpeci ∗typeSpeci)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDecSpeci** (string str_class, string typeq, AstDecSpeci ∗speci, AstTypeSpeci ∗typeSpeci)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- string **storage_class**
- string **type_qual**
- AstDecSpeci ∗ **speci**
- AstTypeSpeci ∗ **typeSpeci**

5.22.1 Detailed Description

Definition at line 1226 of file Ast.h.

5.22.2 Member Function Documentation

5.22.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.22.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.22.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.22.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.22.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.22.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.22.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.22.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.22.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.22.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.22.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.22.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.22.2.13 void AstDecSpeci::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1243 of file Ast.h.

5.22.2.14 void AstDecSpeci::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1243 of file CParser.yy.

5.22.2.15 void AstDecSpeci::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1243 of file CParser.yy.

5.22.2.16 void AstDecSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

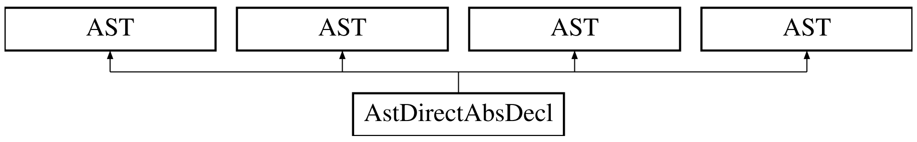
Definition at line 1243 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.23 AstDirectAbsDecl Class Reference

Inheritance diagram for AstDirectAbsDecl:



Public Member Functions

- **AstDirectAbsDecl** (int type, AstAbstractDecl ∗decl, AstExpression ∗exp, AstDirectAbsDecl ∗dabsdecl, AstTypeParamList ∗pList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDirectAbsDecl** (int type, AstAbstractDecl ∗decl, AstExpression ∗exp, AstDirectAbsDecl ∗dabsdecl, AstTypeParamList ∗pList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDirectAbsDecl** (int type, AstAbstractDecl ∗decl, AstExpression ∗exp, AstDirectAbsDecl ∗dabsdecl, AstTypeParamList ∗pList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstDirectAbsDecl** (int type, AstAbstractDecl ∗decl, AstExpression ∗exp, AstDirectAbsDecl ∗dabsdecl, AstTypeParamList ∗pList)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- int **type**
- AstAbstractDecl ∗ **decl**
- AstExpression ∗ **exp**
- AstDirectAbsDecl ∗ **dabsdecl**
- AstTypeParamList ∗ **pList**

5.23.1 Detailed Description

Definition at line 1336 of file Ast.h.

5.23.2 Member Function Documentation

5.23.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.23.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.23.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.23.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.23.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.23.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.23.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.23.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.23.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.23.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.23.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.23.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.23.2.13 void AstDirectAbsDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1353 of file Ast.h.

5.23.2.14 void AstDirectAbsDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1353 of file CParser.yy.

5.23.2.15 void AstDirectAbsDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1353 of file CParser.yy.

5.23.2.16 void AstDirectAbsDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

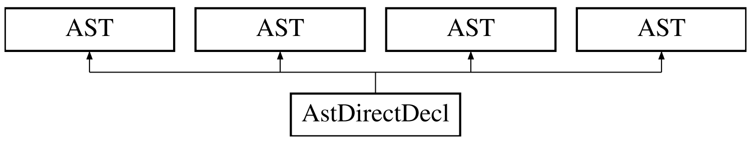
Definition at line 1353 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.24 AstDirectDecl Class Reference

Inheritance diagram for AstDirectDecl:



Public Member Functions

- **AstDirectDecl** (AstID ∗id, AstDirectDecl ∗ddecl, AstExpression ∗exp, AstDeclarator ∗decl, AstTypeParamList ∗plist, AstIDList ∗idList, int type)
- void Visit ()

  This function is responsible for tree traversals.
- AstID ∗ **GetID** ()
- AstDirectDecl ∗ **GetDirectDecl** ()
- **AstDirectDecl** (AstID ∗id, AstDirectDecl ∗ddecl, AstExpression ∗exp, AstDeclarator ∗decl, AstTypeParamList ∗plist, AstIDList ∗idList, int type)
- void Visit ()

  This function is responsible for tree traversals.
- AstID ∗ **GetID** ()
- AstDirectDecl ∗ **GetDirectDecl** ()
- **AstDirectDecl** (AstID ∗id, AstDirectDecl ∗ddecl, AstExpression ∗exp, AstDeclarator ∗decl, AstTypeParamList ∗plist, AstIDList ∗idList, int type)
- void Visit ()

  This function is responsible for tree traversals.
- AstID ∗ **GetID** ()
- AstDirectDecl ∗ **GetDirectDecl** ()
- **AstDirectDecl** (AstID ∗id, AstDirectDecl ∗ddecl, AstExpression ∗exp, AstDeclarator ∗decl, AstTypeParamList ∗plist, AstIDList ∗idList, int type)
- void Visit ()

  This function is responsible for tree traversals.
- AstID ∗ **GetID** ()
- AstDirectDecl ∗ **GetDirectDecl** ()
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.

- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- int **type**
- AstID ∗ **id**
- AstDirectDecl ∗ **ddecl**
- AstExpression ∗ **exp**
- AstDeclarator ∗ **decl**
- AstTypeParamList ∗ **pList**
- AstIDList ∗ **idList**

5.24.1 Detailed Description

Definition at line 1187 of file Ast.h.

5.24.2 Member Function Documentation

5.24.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.24.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.24.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.24.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.24.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.24.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.24.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.24.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.24.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.24.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.24.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.24.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.24.2.13 void AstDirectDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1210 of file CParser.yy.

5.24.2.14 void AstDirectDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1210 of file CParser.yy.

5.24.2.15 void AstDirectDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1210 of file CScanner.ll.

5.24.2.16 void AstDirectDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

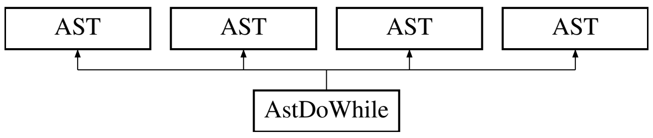
Definition at line 1210 of file Ast.h.

The documentation for this class was generated from the following file:

- Ast.h

5.25 AstDoWhile Class Reference

Inheritance diagram for AstDoWhile:



Public Member Functions

- **AstDoWhile** (AstStatement ∗s, AstExpression ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDoWhile** (AstStatement ∗s, AstExpression ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDoWhile** (AstStatement ∗s, AstExpression ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstDoWhile** (AstStatement ∗s, AstExpression ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstExpression ∗ **test**
- AstStatement ∗ **statement**

5.25.1 Detailed Description

Definition at line 701 of file Ast.h.

5.25.2 Member Function Documentation

5.25.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.25.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.25.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.25.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.25.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.25.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.25.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.25.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.25.2.9 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.25.2.10 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.25.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.25.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.25.2.13 void AstDoWhile::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2089 of file Ast.cpp.

5.25.2.14 void AstDoWhile::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.25.2.15 void AstDoWhile::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.25.2.16 void AstDoWhile::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

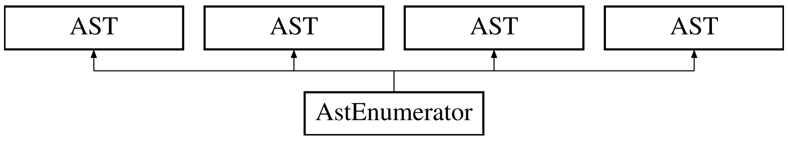
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.26 AstEnumerator Class Reference

Inheritance diagram for AstEnumerator:



Public Member Functions

- **AstEnumerator** (AstID ∗id, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumerator** (AstID ∗id, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumerator** (AstID ∗id, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumerator** (AstID ∗id, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.

- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstID $*$ **id**
- AstExpression $*$ **exp**

5.26.1 Detailed Description

Definition at line 1419 of file Ast.h.

5.26.2 Member Function Documentation

5.26.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.26.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.26.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.26.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.26.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.26.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.26.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.26.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.26.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.26.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.26.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.26.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.26.2.13 void AstEnumerator::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1431 of file Ast.h.

5.26.2.14 void AstEnumerator::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1431 of file CParser.yy.

5.26.2.15 void AstEnumerator::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1431 of file CParser.yy.

5.26.2.16 void AstEnumerator::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

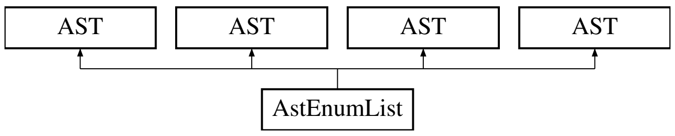
Definition at line 1431 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.27 AstEnumList Class Reference

Inheritance diagram for AstEnumList:



Public Member Functions

- **AstEnumList** (AstEnumerator ∗en, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumList** (AstEnumerator ∗en, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumList** (AstEnumerator ∗en, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEnumList** (AstEnumerator ∗en, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstEnumerator ∗ **en**
- AstEnumList ∗ **list**

5.27.1 Detailed Description

Definition at line 1437 of file Ast.h.

5.27.2 Member Function Documentation

5.27.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.27.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.27.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.27.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.27.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.27.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.27.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.27.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.27.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.27.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.27.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.27.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.27.2.13 void AstEnumList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1449 of file Ast.h.

5.27.2.14 void AstEnumList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1449 of file CParser.yy.

5.27.2.15 void AstEnumList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1449 of file CParser.yy.

5.27.2.16 void AstEnumList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

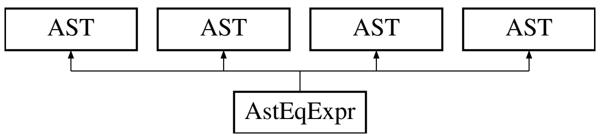
Definition at line 1449 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.28 AstEqExpr Class Reference

Inheritance diagram for AstEqExpr:



Public Types

- enum **Operator** {
  NONE, **EQ_OP**, **NE_OP**, **NONE**,
  EQ_OP, **NE_OP**, **NONE**, **EQ_OP**,
  NE_OP, **NONE**, **EQ_OP**, **NE_OP** }

- enum **Operator** {
  NONE, **EQ_OP**, **NE_OP**, **NONE**,
  EQ_OP, **NE_OP**, **NONE**, **EQ_OP**,
  NE_OP, **NONE**, **EQ_OP**, **NE_OP** }
- enum **Operator** {
  NONE, **EQ_OP**, **NE_OP**, **NONE**,
  EQ_OP, **NE_OP**, **NONE**, **EQ_OP**,
  NE_OP, **NONE**, **EQ_OP**, **NE_OP** }
- enum **Operator** {
  NONE, **EQ_OP**, **NE_OP**, **NONE**,
  EQ_OP, **NE_OP**, **NONE**, **EQ_OP**,
  NE_OP, **NONE**, **EQ_OP**, **NE_OP** }

Public Member Functions

- **AstEqExpr** (AstRelExpr ∗r)
- **AstEqExpr** (AstEqExpr ∗e, Operator o, AstRelExpr ∗r)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEqExpr** (AstRelExpr ∗r)
- **AstEqExpr** (AstEqExpr ∗e, Operator o, AstRelExpr ∗r)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEqExpr** (AstRelExpr ∗r)
- **AstEqExpr** (AstEqExpr ∗e, Operator o, AstRelExpr ∗r)
- void Visit ()

  This function is responsible for tree traversals.
- **AstEqExpr** (AstRelExpr ∗r)
- **AstEqExpr** (AstEqExpr ∗e, Operator o, AstRelExpr ∗r)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstEqExpr::Operator **op**
- Type $*$ **type**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstRelExpr $*$ **rel**
- AstEqExpr $*$ **eq**

5.28.1 Detailed Description

Definition at line 461 of file Ast.h.

5.28.2 Member Function Documentation

5.28.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.28.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.28.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.28.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.28.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.28.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.28.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.28.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.28.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.28.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.28.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.28.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.28.2.13 void AstEqExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1192 of file Ast.cpp.

5.28.2.14 void AstEqExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.28.2.15 void AstEqExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.28.2.16 void AstEqExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

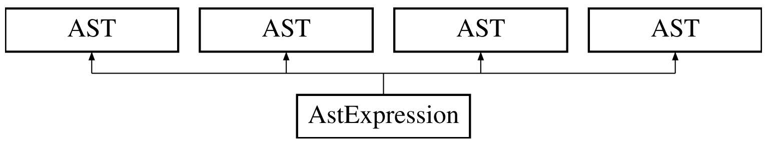
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.29 AstExpression Class Reference

Inheritance diagram for AstExpression:



Public Member Functions

- **AstExpression** (AstAssignExpr ∗a)
- **AstExpression** (AstExpression ∗e, AstAssignExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.
- **AstExpression** (AstAssignExpr ∗a)
- **AstExpression** (AstExpression ∗e, AstAssignExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.

- **AstExpression** (AstAssignExpr ∗a)
- **AstExpression** (AstExpression ∗e, AstAssignExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.
- **AstExpression** (AstAssignExpr ∗a)
- **AstExpression** (AstExpression ∗e, AstAssignExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstAssignExpr $*$ **ass**
- AstExpression $*$ **expr**

5.29.1 Detailed Description

Definition at line 627 of file Ast.h.

5.29.2 Member Function Documentation

5.29.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.29.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.29.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.29.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.29.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.29.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.29.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.29.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.29.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.29.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.29.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.29.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.29.2.13 void AstExpression::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1853 of file Ast.cpp.

5.29.2.14 void AstExpression::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.29.2.15 void AstExpression::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.29.2.16 void AstExpression::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

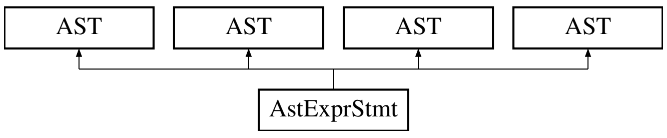
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.30 AstExprStmt Class Reference

Inheritance diagram for AstExprStmt:



Public Member Functions

- **AstExprStmt** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstExprStmt** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstExprStmt** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstExprStmt** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstExpression ∗ **expr**

5.30.1 Detailed Description

Definition at line 804 of file Ast.h.

5.30.2 Member Function Documentation

5.30.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.30.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.30.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.30.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.30.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.30.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.30.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.30.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.30.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.30.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.30.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.30.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.30.2.13 void AstExprStmt::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2473 of file Ast.cpp.

5.30.2.14 void AstExprStmt::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.30.2.15 void AstExprStmt::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.30.2.16 void AstExprStmt::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

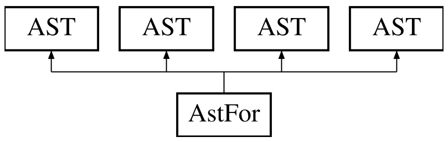
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.31 AstFor Class Reference

Inheritance diagram for AstFor:



Public Member Functions

- **AstFor** (AstExpression ∗init, AstExpression ∗test, AstExpression ∗increment, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstFor** (AstExpression ∗init, AstExpression ∗test, AstExpression ∗increment, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstFor** (AstExpression ∗init, AstExpression ∗test, AstExpression ∗increment, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstFor** (AstExpression ∗init, AstExpression ∗test, AstExpression ∗increment, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstExpression ∗ **init**
- AstExpression ∗ **test**
- AstExpression ∗ **increment**
- AstStatement ∗ **statement**

5.31.1 Detailed Description

Definition at line 721 of file Ast.h.

5.31.2 Member Function Documentation

5.31.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.31.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.31.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.31.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.31.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.31.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.31.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.31.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.31.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.31.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.31.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.31.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.31.2.13 void AstFor::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2164 of file Ast.cpp.

5.31.2.14 void AstFor::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.31.2.15 void AstFor::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.31.2.16 void AstFor::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

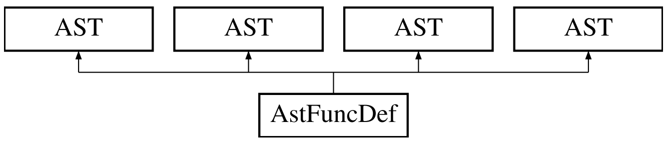
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.32 AstFuncDef Class Reference

Inheritance diagram for AstFuncDef:



Public Member Functions

- **AstFuncDef** (AstDeclarator ∗decl, AstCompound ∗comp, AstDeclList ∗dlist, AstDecSpeci ∗speci)
- string **GetFunctionName** ()
- void Visit ()

  This function is responsible for tree traversals.
- **AstFuncDef** (AstDeclarator ∗decl, AstCompound ∗comp, AstDeclList ∗dlist, AstDecSpeci ∗speci)
- string **GetFunctionName** ()
- void Visit ()

  This function is responsible for tree traversals.

- **AstFuncDef** (AstDeclarator ∗decl, AstCompound ∗comp, AstDeclList ∗dlist, AstDecSpeci ∗speci)
- string **GetFunctionName** ()
- void Visit ()

  This function is responsible for tree traversals.
- **AstFuncDef** (AstDeclarator ∗decl, AstCompound ∗comp, AstDeclList ∗dlist, AstDecSpeci ∗speci)
- string **GetFunctionName** ()
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDeclarator ∗ **decl**
- AstCompound ∗ **comp**
- AstDeclList ∗ **dlist**
- AstDecSpeci ∗ **speci**
- AstDecl ∗ **dec**

5.32.1 Detailed Description

Definition at line 1543 of file Ast.h.

5.32.2 Member Function Documentation

5.32.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.32.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.32.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.32.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.32.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.32.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.32.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.32.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.32.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.32.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.32.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.32.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.32.2.13 void AstFuncDef::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1579 of file Ast.h.

5.32.2.14 void AstFuncDef::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1579 of file CParser.yy.

5.32.2.15 void AstFuncDef::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1579 of file CParser.yy.

5.32.2.16 void AstFuncDef::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

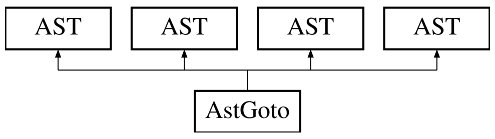
Definition at line 1579 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.33 AstGoto Class Reference

Inheritance diagram for AstGoto:



Public Member Functions

- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.

- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.33.1 Detailed Description

Definition at line 665 of file Ast.h.

5.33.2 Member Function Documentation

5.33.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.33.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.33.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.33.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.33.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.33.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.33.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.33.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.33.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.33.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.33.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.33.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.33.2.13 void AstGoto::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1938 of file Ast.cpp.

5.33.2.14 void AstGoto::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.33.2.15 void AstGoto::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.33.2.16 void AstGoto::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

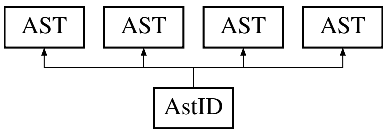
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.34 AstID Class Reference

Inheritance diagram for AstID:



Public Member Functions

- **AstID** (string s, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstID** (string s, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstID** (string s, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstID** (string s, Type ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- string **str**
- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

5.34.1 Detailed Description

Definition at line 196 of file Ast.h.

5.34.2 Member Function Documentation

5.34.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.34.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.34.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.34.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.34.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.34.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.34.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.34.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.34.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.34.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.34.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.34.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.34.2.13 void AstID::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 801 of file Ast.cpp.

5.34.2.14 void AstID::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.34.2.15 void AstID::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.34.2.16 void AstID::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

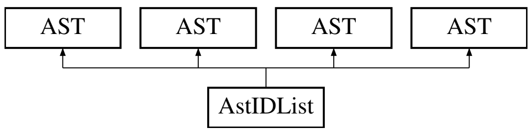
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.35 AstIDList Class Reference

Inheritance diagram for AstIDList:



Public Member Functions

- **AstIDList** (AstID ∗id, AstIDList ∗idlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIDList** (AstID ∗id, AstIDList ∗idlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIDList** (AstID ∗id, AstIDList ∗idlist)
- void Visit ()

This function is responsible for tree traversals.
- **AstIDList** (AstID ∗id, AstIDList ∗idlist)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstID ∗ **id**
- AstIDList ∗ **idlist**

5.35.1 Detailed Description

Definition at line 1036 of file Ast.h.

5.35.2 Member Function Documentation

5.35.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.35.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.35.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.35.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.35.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.35.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.35.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.35.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.35.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.35.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.35.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.35.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.35.2.13 void AstIDList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1049 of file Ast.h.

5.35.2.14 void AstIDList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1049 of file CParser.yy.

5.35.2.15 void AstIDList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1049 of file CParser.yy.

5.35.2.16 void AstIDList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

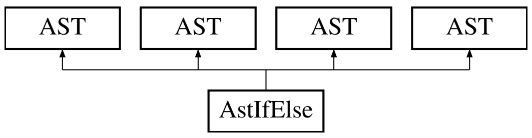
Definition at line 1049 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.36 AstIfElse Class Reference

Inheritance diagram for AstIfElse:



Public Member Functions

- **AstIfElse** (AstExpression ∗test, AstStatement ∗statement, AstStatement ∗elseStatement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIfElse** (AstExpression ∗test, AstStatement ∗statement, AstStatement ∗elseStatement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIfElse** (AstExpression ∗test, AstStatement ∗statement, AstStatement ∗elseStatement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIfElse** (AstExpression ∗test, AstStatement ∗statement, AstStatement ∗elseStatement)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstExpression ∗ **test**
- AstStatement ∗ **statement**
- AstStatement ∗ **elseStatement**

5.36.1 Detailed Description

Definition at line 763 of file Ast.h.

5.36.2 Member Function Documentation

5.36.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.36.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.36.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.36.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.36.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.36.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.36.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.36.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.36.2.9 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.36.2.10 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.36.2.11 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.36.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.36.2.13 void AstIfElse::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2308 of file Ast.cpp.

5.36.2.14 void AstIfElse::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.36.2.15 void AstIfElse::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.36.2.16 void AstIfElse::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

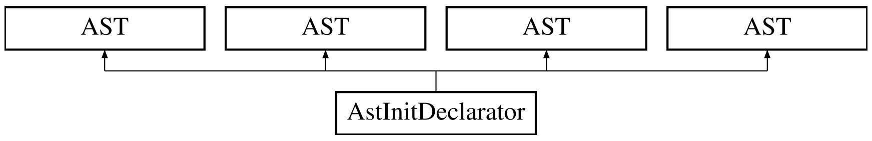
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.37 AstInitDeclarator Class Reference

Inheritance diagram for AstInitDeclarator:



Public Member Functions

- **AstInitDeclarator** (AstDeclarator ∗decl, AstInitializer ∗init)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitDeclarator** (AstDeclarator ∗decl, AstInitializer ∗init)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitDeclarator** (AstDeclarator ∗decl, AstInitializer ∗init)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitDeclarator** (AstDeclarator ∗decl, AstInitializer ∗init)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDeclarator ∗ **decl**
- AstInitializer ∗ **init**

5.37.1 Detailed Description

Definition at line 1380 of file Ast.h.

5.37.2 Member Function Documentation

5.37.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.37.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.37.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.37.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.37.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.37.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.37.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.37.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.37.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.37.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.37.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.37.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.37.2.13 void AstInitDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1392 of file Ast.h.

5.37.2.14 void AstInitDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1392 of file CParser.yy.

5.37.2.15 void AstInitDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1392 of file CParser.yy.

5.37.2.16 void AstInitDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

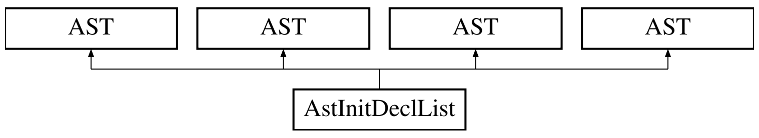
Definition at line 1392 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.38 AstInitDeclList Class Reference

Inheritance diagram for AstInitDeclList:



Public Member Functions

- **AstInitDeclList** (AstInitDeclarator ∗decl, AstInitDeclList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitDeclList** (AstInitDeclarator ∗decl, AstInitDeclList ∗list)
- void Visit ()

This function is responsible for tree traversals.

- **AstInitDeclList** (AstInitDeclarator ∗decl, AstInitDeclList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitDeclList** (AstInitDeclarator ∗decl, AstInitDeclList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""

- static string returnLabel =""
  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid
  The unique id.
- string label
  The label to be printed in the visualization.

Private Attributes

- AstInitDeclarator ∗ **decl**
- AstInitDeclList ∗ **list**

5.38.1 Detailed Description

Definition at line 1362 of file Ast.h.

5.38.2 Member Function Documentation

5.38.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.38.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.38.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.38.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.38.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.38.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.38.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.38.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.38.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.38.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.38.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.38.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.38.2.13 void AstInitDeclList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1374 of file Ast.h.

5.38.2.14 void AstInitDeclList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1374 of file CParser.yy.

5.38.2.15 void AstInitDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1374 of file CParser.yy.

5.38.2.16 void AstInitDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

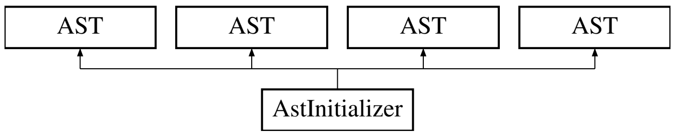
Definition at line 1374 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.39 AstInitializer Class Reference

Inheritance diagram for AstInitializer:



Public Member Functions

- **AstInitializer** (AstAssignExpr ∗expr, AstInitList ∗list, int type)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitializer** (AstAssignExpr ∗expr, AstInitList ∗list, int type)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitializer** (AstAssignExpr ∗expr, AstInitList ∗list, int type)
- void Visit ()

  This function is responsible for tree traversals.
- **AstInitializer** (AstAssignExpr ∗expr, AstInitList ∗list, int type)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstAssignExpr ∗ **expr**
- AstInitList ∗ **list**
- int **type**

5.39.1 Detailed Description

Definition at line 1398 of file Ast.h.

5.39.2 Member Function Documentation

5.39.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.39.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.39.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.39.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.39.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.39.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.39.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.39.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.39.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.39.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.39.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.39.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.39.2.13 void AstInitializer::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1412 of file Ast.h.

5.39.2.14 void AstInitializer::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1412 of file CParser.yy.

5.39.2.15 void AstInitializer::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1412 of file CParser.yy.

5.39.2.16 void AstInitializer::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

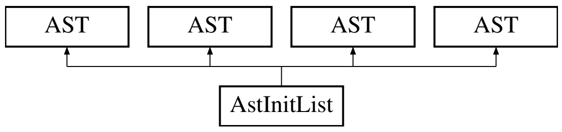
Definition at line 1412 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.40 AstInitList Class Reference

Inheritance diagram for AstInitList:



Public Member Functions

- **AstInitList** (AstInitializer ∗intializer, AstInitList ∗list)
- void Visit ()

    This function is responsible for tree traversals.
- **AstInitList** (AstInitializer ∗intializer, AstInitList ∗list)
- void Visit ()

    This function is responsible for tree traversals.
- **AstInitList** (AstInitializer ∗intializer, AstInitList ∗list)
- void Visit ()

    This function is responsible for tree traversals.
- **AstInitList** (AstInitializer ∗intializer, AstInitList ∗list)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.

- int getUID ()
  Gets the node's unique ID.
- string getLabel ()
  Gets the node's label.
- string getLabel ()
  Gets the node's label.
- string getLabel ()
  Gets the node's label.
- string getLabel ()
  Gets the node's label.

Public Attributes

- bool needsCast
  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv
  Indicates is a conversion is possible.
- CONVERSIONTYPE convType
  If needsCast is true, then this indicates what the cast should be.
- int operandToCast
  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis
  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen
  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""
  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid
  The unique id.
- string label
  The label to be printed in the visualization.

Private Attributes

- AstInitializer $*$ **intializer**
- AstInitList $*$ **list**

5.40.1 Detailed Description

Definition at line 1014 of file Ast.h.

5.40.2 Member Function Documentation

5.40.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.40.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.40.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.40.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.40.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.40.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.40.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.40.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.40.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.40.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.40.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.40.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.40.2.13 void AstInitList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1028 of file Ast.h.

5.40.2.14 void AstInitList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1028 of file CParser.yy.

5.40.2.15 void AstInitList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1028 of file CParser.yy.

5.40.2.16 void AstInitList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

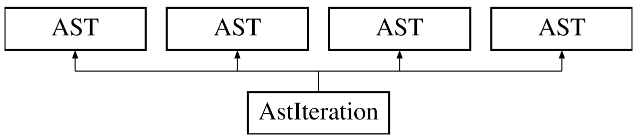
Definition at line 1028 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.41 AstIteration Class Reference

Inheritance diagram for AstIteration:



Public Types

- enum **Type** {
  DOWHILE, **WHILE**, **FOR**, **DOWHILE**,
  WHILE, **FOR**, **DOWHILE**, **WHILE**,
  FOR, **DOWHILE**, **WHILE**, **FOR** }
- enum **Type** {
  DOWHILE, **WHILE**, **FOR**, **DOWHILE**,
  WHILE, **FOR**, **DOWHILE**, **WHILE**,
  FOR, **DOWHILE**, **WHILE**, **FOR** }
- enum **Type** {
  DOWHILE, **WHILE**, **FOR**, **DOWHILE**,
  WHILE, **FOR**, **DOWHILE**, **WHILE**,
  FOR, **DOWHILE**, **WHILE**, **FOR** }
- enum **Type** {
  DOWHILE, **WHILE**, **FOR**, **DOWHILE**,
  WHILE, **FOR**, **DOWHILE**, **WHILE**,
  FOR, **DOWHILE**, **WHILE**, **FOR** }

Public Member Functions

- **AstIteration** (AstDoWhile ∗d)
- **AstIteration** (AstWhile ∗w)
- **AstIteration** (AstFor ∗f)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIteration** (AstDoWhile ∗d)
- **AstIteration** (AstWhile ∗w)
- **AstIteration** (AstFor ∗f)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIteration** (AstDoWhile ∗d)
- **AstIteration** (AstWhile ∗w)
- **AstIteration** (AstFor ∗f)
- void Visit ()

  This function is responsible for tree traversals.
- **AstIteration** (AstDoWhile ∗d)
- **AstIteration** (AstWhile ∗w)
- **AstIteration** (AstFor ∗f)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.

- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstIteration::Type **t**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDoWhile ∗ **dwl**
- AstWhile ∗ **wl**
- AstFor ∗ **fr**

5.41.1 Detailed Description

Definition at line 733 of file Ast.h.

5.41.2 Member Function Documentation

5.41.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.41.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.41.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.41.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.41.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.41.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.41.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.41.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.41.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.41.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.41.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.41.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.41.2.13 void AstIteration::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2235 of file Ast.cpp.

5.41.2.14 void AstIteration::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.41.2.15 void AstIteration::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.41.2.16 void AstIteration::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

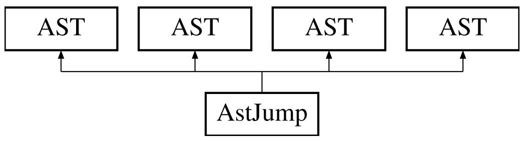
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.42 AstJump Class Reference

Inheritance diagram for AstJump:



Public Types

- enum **Type** {
GOTO, **CONTINUE**, **BREAK**, **EMPTY_RETURN**,
RETURN, **GOTO**, **CONTINUE**, **BREAK**,
EMPTY_RETURN, **RETURN**, **GOTO**, **CONTINUE**,
BREAK, **EMPTY_RETURN**, **RETURN**, **GOTO**,
CONTINUE, **BREAK**, **EMPTY_RETURN**, **RETURN** }
- enum **Type** {
GOTO, **CONTINUE**, **BREAK**, **EMPTY_RETURN**,
RETURN, **GOTO**, **CONTINUE**, **BREAK**,
EMPTY_RETURN, **RETURN**, **GOTO**, **CONTINUE**,
BREAK, **EMPTY_RETURN**, **RETURN**, **GOTO**,
CONTINUE, **BREAK**, **EMPTY_RETURN**, **RETURN** }
- enum **Type** {
GOTO, **CONTINUE**, **BREAK**, **EMPTY_RETURN**,
RETURN, **GOTO**, **CONTINUE**, **BREAK**,
EMPTY_RETURN, **RETURN**, **GOTO**, **CONTINUE**,
BREAK, **EMPTY_RETURN**, **RETURN**, **GOTO**,
CONTINUE, **BREAK**, **EMPTY_RETURN**, **RETURN** }
- enum **Type** {
GOTO, **CONTINUE**, **BREAK**, **EMPTY_RETURN**,
RETURN, **GOTO**, **CONTINUE**, **BREAK**,
EMPTY_RETURN, **RETURN**, **GOTO**, **CONTINUE**,
BREAK, **EMPTY_RETURN**, **RETURN**, **GOTO**,
CONTINUE, **BREAK**, **EMPTY_RETURN**, **RETURN** }

Public Member Functions

- **AstJump** (AstGoto ∗g, AstID ∗i)
- **AstJump** (AstContinue ∗c)
- **AstJump** (AstBreak ∗b)
- **AstJump** (AstReturn ∗r)
- **AstJump** (AstReturn ∗r, AstExpression ∗e)
- void Visit ()

This function is responsible for tree traversals.

- **AstJump** (AstGoto ∗g, AstID ∗i)
- **AstJump** (AstContinue ∗c)
- **AstJump** (AstBreak ∗b)
- **AstJump** (AstReturn ∗r)
- **AstJump** (AstReturn ∗r, AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstJump** (AstGoto ∗g, AstID ∗i)
- **AstJump** (AstContinue ∗c)
- **AstJump** (AstBreak ∗b)
- **AstJump** (AstReturn ∗r)
- **AstJump** (AstReturn ∗r, AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstJump** (AstGoto ∗g, AstID ∗i)
- **AstJump** (AstContinue ∗c)
- **AstJump** (AstBreak ∗b)
- **AstJump** (AstReturn ∗r)
- **AstJump** (AstReturn ∗r, AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstJump::Type **t**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstGoto ∗ **go**
- AstID ∗ **id**
- AstContinue ∗ **cont**
- AstBreak ∗ **br**
- AstReturn ∗ **ret**
- AstExpression ∗ **expr**

5.42.1 Detailed Description

Definition at line 672 of file Ast.h.

5.42.2 Member Function Documentation

5.42.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.42.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.42.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.42.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.42.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.42.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.42.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.42.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.42.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.42.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.42.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.42.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.42.2.13 void AstJump::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2012 of file Ast.cpp.

5.42.2.14 void AstJump::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.42.2.15 void AstJump::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.42.2.16 void AstJump::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

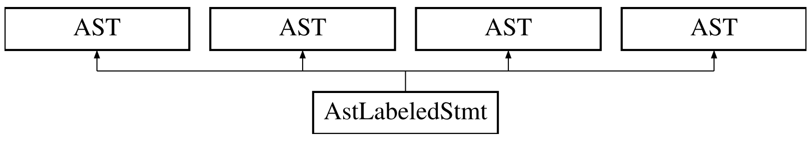
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.43 AstLabeledStmt Class Reference

Inheritance diagram for AstLabeledStmt:



Public Types

- enum **Type** {
NO_CASE, **CASE**, **DEFAULT**, **NO_CASE**,
CASE, **DEFAULT**, **NO_CASE**, **CASE**,
DEFAULT, **NO_CASE**, **CASE**, **DEFAULT** }
- enum **Type** {
NO_CASE, **CASE**, **DEFAULT**, **NO_CASE**,
CASE, **DEFAULT**, **NO_CASE**, **CASE**,
DEFAULT, **NO_CASE**, **CASE**, **DEFAULT** }
- enum **Type** {
NO_CASE, **CASE**, **DEFAULT**, **NO_CASE**,
CASE, **DEFAULT**, **NO_CASE**, **CASE**,
DEFAULT, **NO_CASE**, **CASE**, **DEFAULT** }
- enum **Type** {
NO_CASE, **CASE**, **DEFAULT**, **NO_CASE**,
CASE, **DEFAULT**, **NO_CASE**, **CASE**,
DEFAULT, **NO_CASE**, **CASE**, **DEFAULT** }

Public Member Functions

- **AstLabeledStmt** (AstID ∗i, AstStatement ∗s)
- **AstLabeledStmt** (AstConstantExpr ∗c, AstStatement ∗s)
- **AstLabeledStmt** (AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLabeledStmt** (AstID ∗i, AstStatement ∗s)
- **AstLabeledStmt** (AstConstantExpr ∗c, AstStatement ∗s)
- **AstLabeledStmt** (AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLabeledStmt** (AstID ∗i, AstStatement ∗s)
- **AstLabeledStmt** (AstConstantExpr ∗c, AstStatement ∗s)
- **AstLabeledStmt** (AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLabeledStmt** (AstID ∗i, AstStatement ∗s)
- **AstLabeledStmt** (AstConstantExpr ∗c, AstStatement ∗s)
- **AstLabeledStmt** (AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstLabeledStmt::Type **t**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstID $*$ **id**
- AstStatement $*$ **stmt**
- AstConstantExpr $*$ **constExpr**

5.43.1 Detailed Description

Definition at line 813 of file Ast.h.

5.43.2 Member Function Documentation

5.43.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.43.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.43.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.43.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.43.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.43.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.43.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.43.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.43.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.43.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.43.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.43.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.43.2.13 void AstLabeledStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2520 of file Ast.cpp.

5.43.2.14 void AstLabeledStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.43.2.15 void AstLabeledStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.43.2.16 void AstLabeledStmt::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

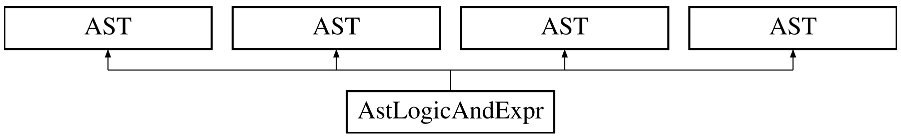
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.44 AstLogicAndExpr Class Reference

Inheritance diagram for AstLogicAndExpr:



Public Member Functions

- **AstLogicAndExpr** (AstORExpr ∗o)
- **AstLogicAndExpr** (AstLogicAndExpr ∗a, AstORExpr ∗o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLogicAndExpr** (AstORExpr ∗o)
- **AstLogicAndExpr** (AstLogicAndExpr ∗a, AstORExpr ∗o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLogicAndExpr** (AstORExpr ∗o)
- **AstLogicAndExpr** (AstLogicAndExpr ∗a, AstORExpr ∗o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLogicAndExpr** (AstORExpr ∗o)
- **AstLogicAndExpr** (AstLogicAndExpr ∗a, AstORExpr ∗o)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstORExpr ∗ **o**
- AstLogicAndExpr ∗ **a**

5.44.1 Detailed Description

Definition at line 527 of file Ast.h.

5.44.2 Member Function Documentation

5.44.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.44.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.44.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.44.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.44.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.44.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.44.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.44.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.44.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.44.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.44.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.44.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.44.2.13 void AstLogicAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1468 of file Ast.cpp.

5.44.2.14 void AstLogicAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.44.2.15 void AstLogicAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.44.2.16 void AstLogicAndExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

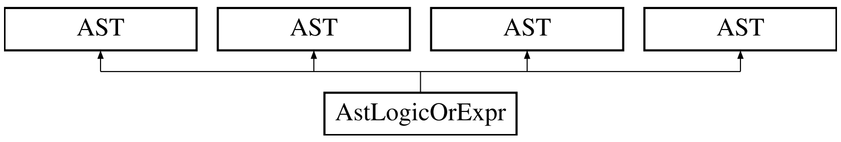
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.45 AstLogicOrExpr Class Reference

Inheritance diagram for AstLogicOrExpr:



Public Member Functions

- **AstLogicOrExpr** (AstLogicAndExpr ∗a)
- **AstLogicOrExpr** (AstLogicOrExpr ∗o, AstLogicAndExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.
- **AstLogicOrExpr** (AstLogicAndExpr ∗a)
- **AstLogicOrExpr** (AstLogicOrExpr ∗o, AstLogicAndExpr ∗a)
- void Visit ()

    This function is responsible for tree traversals.

- **AstLogicOrExpr** (AstLogicAndExpr ∗a)
- **AstLogicOrExpr** (AstLogicOrExpr ∗o, AstLogicAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstLogicOrExpr** (AstLogicAndExpr ∗a)
- **AstLogicOrExpr** (AstLogicOrExpr ∗o, AstLogicAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstLogicAndExpr $*$ **a**
- AstLogicOrExpr $*$ **o**

5.45.1 Detailed Description

Definition at line 541 of file Ast.h.

5.45.2 Member Function Documentation

5.45.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.45.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.45.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.45.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.45.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.45.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.45.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.45.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.45.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.45.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.45.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.45.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.45.2.13 void AstLogicOrExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1533 of file Ast.cpp.

5.45.2.14 void AstLogicOrExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.45.2.15 void AstLogicOrExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.45.2.16 void AstLogicOrExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

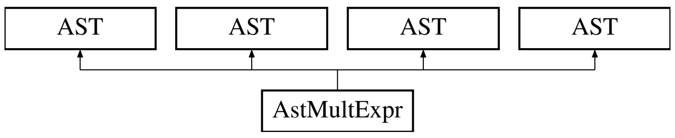
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.46 AstMultExpr Class Reference

Inheritance diagram for AstMultExpr:



Public Types

- enum **Operator** {
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD** }
- enum **Operator** {
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD** }
- enum **Operator** {
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD** }
- enum **Operator** {
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD**,
  NONE, **STAR**, **DIV**, **MOD** }

Public Member Functions

- **AstMultExpr** (AstCastExpr ∗c)
- **AstMultExpr** (AstMultExpr ∗m, Operator o, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- **AstMultExpr** (AstCastExpr ∗c)
- **AstMultExpr** (AstMultExpr ∗m, Operator o, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- **AstMultExpr** (AstCastExpr ∗c)
- **AstMultExpr** (AstMultExpr ∗m, Operator o, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- **AstMultExpr** (AstCastExpr ∗c)
- **AstMultExpr** (AstMultExpr ∗m, Operator o, AstCastExpr ∗c)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstMultExpr::Operator **op**
- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.

- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstCastExpr $*$ **cast**
- AstMultExpr $*$ **mult**

5.46.1 Detailed Description

Definition at line 362 of file Ast.h.

5.46.2 Member Function Documentation

5.46.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.46.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.46.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.46.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.46.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.46.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.46.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.46.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.46.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.46.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.46.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.46.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.46.2.13 void AstMultExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 843 of file Ast.cpp.

5.46.2.14 void AstMultExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.46.2.15 void AstMultExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.46.2.16 void AstMultExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

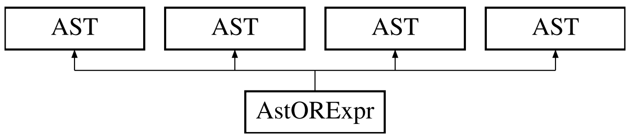
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.47 AstORExpr Class Reference

Inheritance diagram for AstORExpr:



Public Member Functions

- **AstORExpr** (AstXORExpr ∗x)
- **AstORExpr** (AstORExpr ∗o, AstXORExpr ∗x)
- void Visit ()

  This function is responsible for tree traversals.
- **AstORExpr** (AstXORExpr ∗x)
- **AstORExpr** (AstORExpr ∗o, AstXORExpr ∗x)
- void Visit ()

  This function is responsible for tree traversals.

- **AstORExpr** (AstXORExpr ∗x)
- **AstORExpr** (AstORExpr ∗o, AstXORExpr ∗x)
- void Visit ()

    This function is responsible for tree traversals.
- **AstORExpr** (AstXORExpr ∗x)
- **AstORExpr** (AstORExpr ∗o, AstXORExpr ∗x)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstXORExpr $*$ **x**
- AstORExpr $*$ **o**

5.47.1 Detailed Description

Definition at line 513 of file Ast.h.

5.47.2 Member Function Documentation

5.47.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.47.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.47.2.3 string AST::getLabel ( ) [inline],[inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.47.2.4 string AST::getLabel ( ) [inline],[inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.47.2.5 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.47.2.6 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.47.2.7 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.47.2.8 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.47.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.47.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.47.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.47.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.47.2.13 void AstORExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1403 of file Ast.cpp.

5.47.2.14 void AstORExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.47.2.15 void AstORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.47.2.16 void AstORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

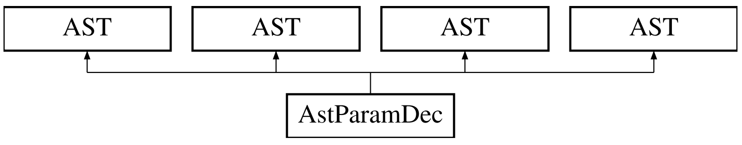
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.48 AstParamDec Class Reference

Inheritance diagram for AstParamDec:



Public Member Functions

- **AstParamDec** (AstDecSpeci ∗speci, AstDeclarator ∗declarator, AstAbstractDecl ∗adecl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamDec** (AstDecSpeci ∗speci, AstDeclarator ∗declarator, AstAbstractDecl ∗adecl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamDec** (AstDecSpeci ∗speci, AstDeclarator ∗declarator, AstAbstractDecl ∗adecl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamDec** (AstDecSpeci ∗speci, AstDeclarator ∗declarator, AstAbstractDecl ∗adecl)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstDecSpeci ∗ **speci**
- AstDeclarator ∗ **declarator**
- AstAbstractDecl ∗ **adecl**

5.48.1 Detailed Description

Definition at line 990 of file Ast.h.

5.48.2 Member Function Documentation

5.48.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.48.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.48.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.48.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.48.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.48.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.48.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.48.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.48.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.48.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.48.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.48.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.48.2.13 void AstParamDec::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1004 of file Ast.h.

5.48.2.14 void AstParamDec::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1004 of file CParser.yy.

5.48.2.15 void AstParamDec::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1004 of file CParser.yy.

5.48.2.16 void AstParamDec::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

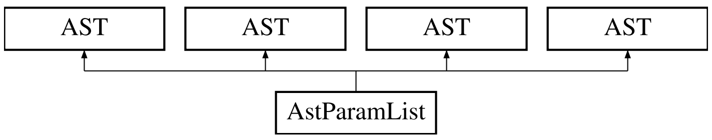
Definition at line 1004 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.49 AstParamList Class Reference

Inheritance diagram for AstParamList:



Public Member Functions

- **AstParamList** (AstParamDec ∗dec, AstParamList ∗plist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamList** (AstParamDec ∗dec, AstParamList ∗plist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamList** (AstParamDec ∗dec, AstParamList ∗plist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstParamList** (AstParamDec ∗dec, AstParamList ∗plist)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstParamDec $*$ **dec**
- AstParamList $*$ **plist**

5.49.1 Detailed Description

Definition at line 970 of file Ast.h.

5.49.2 Member Function Documentation

5.49.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.49.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.49.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.49.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.49.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.49.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.49.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.49.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.49.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.49.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.49.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.49.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.49.2.13 void AstParamList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 983 of file Ast.h.

5.49.2.14 void AstParamList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 983 of file CParser.yy.

5.49.2.15 void AstParamList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 983 of file CParser.yy.

5.49.2.16 void AstParamList::Visit ( ) `[inline], [virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

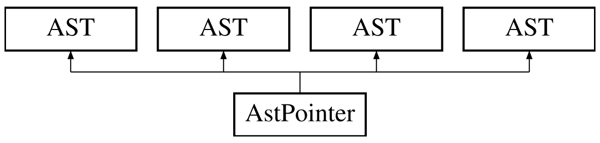
Definition at line 983 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.50 AstPointer Class Reference

Inheritance diagram for AstPointer:



Public Member Functions

- **AstPointer** (AstPointer ∗pointer, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPointer** (AstPointer ∗pointer, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPointer** (AstPointer ∗pointer, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPointer** (AstPointer ∗pointer, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstPointer $*$ **pointer**
- AstTypeQualList $*$ **list**

5.50.1 Detailed Description

Definition at line 949 of file Ast.h.

5.50.2 Member Function Documentation

5.50.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.50.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.50.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.50.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.50.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.50.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.50.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.50.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.50.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.50.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.50.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.50.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.50.2.13 void AstPointer::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 962 of file Ast.h.

5.50.2.14 void AstPointer::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 962 of file CParser.yy.

5.50.2.15 void AstPointer::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 962 of file CParser.yy.

5.50.2.16 void AstPointer::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

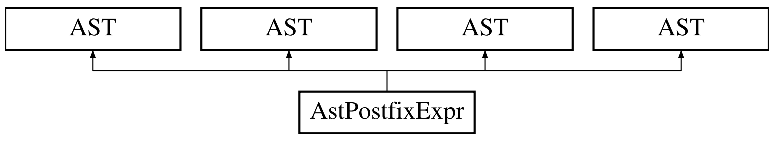
Definition at line 962 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.51 AstPostfixExpr Class Reference

Inheritance diagram for AstPostfixExpr:



Public Types

- enum **Operator** {
  NONE, **DOT_OP**, **PTR_OP**, **INC_OP**,
  DEC_OP, **NONE**, **DOT_OP**, **PTR_OP**,
  INC_OP, **DEC_OP**, **NONE**, **DOT_OP**,
  PTR_OP, **INC_OP**, **DEC_OP**, **NONE**,
  DOT_OP, **PTR_OP**, **INC_OP**, **DEC_OP** }

- enum **ExprType** {
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC** }
- enum **Operator** {
  NONE, **DOT_OP**, **PTR_OP**, **INC_OP**,
  DEC_OP, **NONE**, **DOT_OP**, **PTR_OP**,
  INC_OP, **DEC_OP**, **NONE**, **DOT_OP**,
  PTR_OP, **INC_OP**, **DEC_OP**, **NONE**,
  DOT_OP, **PTR_OP**, **INC_OP**, **DEC_OP** }
- enum **ExprType** {
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC** }
- enum **Operator** {
  NONE, **DOT_OP**, **PTR_OP**, **INC_OP**,
  DEC_OP, **NONE**, **DOT_OP**, **PTR_OP**,
  INC_OP, **DEC_OP**, **NONE**, **DOT_OP**,
  PTR_OP, **INC_OP**, **DEC_OP**, **NONE**,
  DOT_OP, **PTR_OP**, **INC_OP**, **DEC_OP** }
- enum **ExprType** {
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC** }
- enum **Operator** {
  NONE, **DOT_OP**, **PTR_OP**, **INC_OP**,
  DEC_OP, **NONE**, **DOT_OP**, **PTR_OP**,
  INC_OP, **DEC_OP**, **NONE**, **DOT_OP**,
  PTR_OP, **INC_OP**, **DEC_OP**, **NONE**,
  DOT_OP, **PTR_OP**, **INC_OP**, **DEC_OP** }
- enum **ExprType** {
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC**,
  PRIMARY, **BRACKETS**, **EMPTY_PARENS**, **PARENS**,
  DOT, **PTR**, **INC**, **DEC** }

Public Member Functions

- **AstPostfixExpr** (AstPrimaryExpr ∗p)

- **AstPostfixExpr** (AstPostfixExpr ∗p, AstExpression ∗e)
- **AstPostfixExpr** (AstPostfixExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstArgExprList ∗a)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o, AstID ∗i)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o)
- void Visit ()

    This function is responsible for tree traversals.
- **AstPostfixExpr** (AstPrimaryExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstExpression ∗e)
- **AstPostfixExpr** (AstPostfixExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstArgExprList ∗a)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o, AstID ∗i)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o)
- void Visit ()

    This function is responsible for tree traversals.
- **AstPostfixExpr** (AstPrimaryExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstExpression ∗e)
- **AstPostfixExpr** (AstPostfixExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstArgExprList ∗a)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o, AstID ∗i)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o)
- void Visit ()

    This function is responsible for tree traversals.
- **AstPostfixExpr** (AstPrimaryExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstExpression ∗e)
- **AstPostfixExpr** (AstPostfixExpr ∗p)
- **AstPostfixExpr** (AstPostfixExpr ∗p, AstArgExprList ∗a)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o, AstID ∗i)
- **AstPostfixExpr** (AstPostfixExpr ∗p, Operator o)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstPrimaryExpr ∗ **priexpr**
- AstPostfixExpr ∗ **ptfExpr**
- AstExpression ∗ **brakExpr**
- AstArgExprList ∗ **argExprList**
- AstID ∗ **id**
- Operator **op**
- ExprType **t**

5.51.1 Detailed Description

Definition at line 254 of file Ast.h.

5.51.2 Member Function Documentation

5.51.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.51.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.51.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.51.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.51.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.51.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.51.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.51.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.51.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.51.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.51.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.51.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.51.2.13 void AstPostfixExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 273 of file Ast.cpp.

5.51.2.14 void AstPostfixExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.51.2.15 void AstPostfixExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.51.2.16 void AstPostfixExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

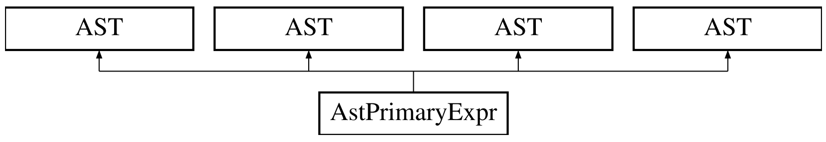
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.52 AstPrimaryExpr Class Reference

Inheritance diagram for AstPrimaryExpr:



Public Member Functions

- **AstPrimaryExpr** (AstID ∗id)
- **AstPrimaryExpr** (AstConstant ∗c)
- **AstPrimaryExpr** (AstString ∗s)
- **AstPrimaryExpr** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPrimaryExpr** (AstID ∗id)
- **AstPrimaryExpr** (AstConstant ∗c)
- **AstPrimaryExpr** (AstString ∗s)
- **AstPrimaryExpr** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPrimaryExpr** (AstID ∗id)
- **AstPrimaryExpr** (AstConstant ∗c)
- **AstPrimaryExpr** (AstString ∗s)
- **AstPrimaryExpr** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- **AstPrimaryExpr** (AstID ∗id)
- **AstPrimaryExpr** (AstConstant ∗c)
- **AstPrimaryExpr** (AstString ∗s)
- **AstPrimaryExpr** (AstExpression ∗e)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **etype**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Types

- enum **ExprType** {
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR** }
- enum **ExprType** {
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR** }
- enum **ExprType** {
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR** }
- enum **ExprType** {
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR**,
  ID, **CONST**, **STRING**, **EXPR** }

Private Attributes

- ExprType **type**
- AstID ∗ **id**
- AstConstant ∗ **constant**
- AstString ∗ **str**
- AstExpression ∗ **expr**

5.52.1 Detailed Description

Definition at line 212 of file Ast.h.

5.52.2 Member Function Documentation

5.52.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.52.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.52.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.52.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.52.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.52.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.52.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.52.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.52.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.52.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.52.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.52.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.52.2.13 void AstPrimaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 66 of file Ast.cpp.

5.52.2.14 void AstPrimaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.52.2.15 void AstPrimaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.52.2.16 void AstPrimaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

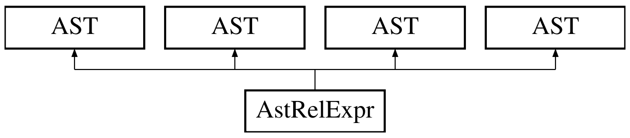
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.53 AstRelExpr Class Reference

Inheritance diagram for AstRelExpr:



Public Types

- enum **Operator** {
  NONE, **LT_OP**, **GT_OP**, **LE_OP**,
  GE_OP, **NONE**, **LT_OP**, **GT_OP**,
  LE_OP, **GE_OP**, **NONE**, **LT_OP**,
  GT_OP, **LE_OP**, **GE_OP**, **NONE**,
  LT_OP, **GT_OP**, **LE_OP**, **GE_OP** }
- enum **Operator** {
  NONE, **LT_OP**, **GT_OP**, **LE_OP**,
  GE_OP, **NONE**, **LT_OP**, **GT_OP**,
  LE_OP, **GE_OP**, **NONE**, **LT_OP**,
  GT_OP, **LE_OP**, **GE_OP**, **NONE**,
  LT_OP, **GT_OP**, **LE_OP**, **GE_OP** }
- enum **Operator** {
  NONE, **LT_OP**, **GT_OP**, **LE_OP**,
  GE_OP, **NONE**, **LT_OP**, **GT_OP**,
  LE_OP, **GE_OP**, **NONE**, **LT_OP**,
  GT_OP, **LE_OP**, **GE_OP**, **NONE**,
  LT_OP, **GT_OP**, **LE_OP**, **GE_OP** }
- enum **Operator** {
  NONE, **LT_OP**, **GT_OP**, **LE_OP**,
  GE_OP, **NONE**, **LT_OP**, **GT_OP**,
  LE_OP, **GE_OP**, **NONE**, **LT_OP**,
  GT_OP, **LE_OP**, **GE_OP**, **NONE**,
  LT_OP, **GT_OP**, **LE_OP**, **GE_OP** }

Public Member Functions

- **AstRelExpr** (AstShiftExpr ∗s)
- **AstRelExpr** (AstRelExpr ∗r, Operator o, AstShiftExpr ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstRelExpr** (AstShiftExpr ∗s)

- **AstRelExpr** (AstRelExpr ∗r, Operator o, AstShiftExpr ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstRelExpr** (AstShiftExpr ∗s)
- **AstRelExpr** (AstRelExpr ∗r, Operator o, AstShiftExpr ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstRelExpr** (AstShiftExpr ∗s)
- **AstRelExpr** (AstRelExpr ∗r, Operator o, AstShiftExpr ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstRelExpr::Operator **op**
- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab * **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstShiftExpr * **shift**
- AstRelExpr * **rel**

5.53.1 Detailed Description

Definition at line 435 of file Ast.h.

5.53.2 Member Function Documentation

5.53.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.53.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.53.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.53.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.53.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.53.2.6 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.53.2.7 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.53.2.8 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.53.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.53.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.53.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.53.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.53.2.13 void AstRelExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1105 of file Ast.cpp.

5.53.2.14 void AstRelExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.53.2.15 void AstRelExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.53.2.16 void AstRelExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

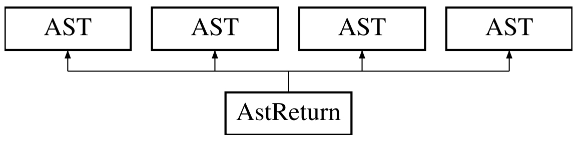
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.54 AstReturn Class Reference

Inheritance diagram for AstReturn:



Public Member Functions

- **AstReturn** (AstExpression ∗r)
- void Visit ()

    This function is responsible for tree traversals.
- **AstReturn** (AstExpression ∗r)
- void Visit ()

    This function is responsible for tree traversals.
- **AstReturn** (AstExpression ∗r)
- void Visit ()

    This function is responsible for tree traversals.
- **AstReturn** (AstExpression ∗r)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstExpression * **expr**

5.54.1 Detailed Description

Definition at line 641 of file Ast.h.

5.54.2 Member Function Documentation

5.54.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.54.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.54.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.54.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.54.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.54.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.54.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.54.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.54.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.54.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.54.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.54.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.54.2.13 void AstReturn::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1888 of file Ast.cpp.

5.54.2.14 void AstReturn::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.54.2.15 void AstReturn::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.54.2.16 void AstReturn::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

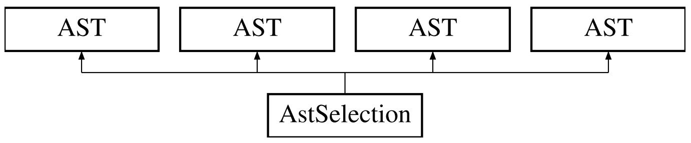
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.55 AstSelection Class Reference

Inheritance diagram for AstSelection:



Public Types

- enum **Type** {
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE**,
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE** }
- enum **Type** {
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE**,
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE** }
- enum **Type** {
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE**,
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE** }
- enum **Type** {
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE**,
  SWITCH, **IFELSE**, **SWITCH**, **IFELSE** }

Public Member Functions

- **AstSelection** (AstSwitch ∗s)
- **AstSelection** (AstIfElse ∗ie)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSelection** (AstSwitch ∗s)
- **AstSelection** (AstIfElse ∗ie)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSelection** (AstSwitch ∗s)
- **AstSelection** (AstIfElse ∗ie)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSelection** (AstSwitch ∗s)
- **AstSelection** (AstIfElse ∗ie)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstSelection::Type **t**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstSwitch ∗ **swtch**
- AstIfElse ∗ **ifelse**

5.55.1 Detailed Description

Definition at line 774 of file Ast.h.

5.55.2 Member Function Documentation

5.55.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.55.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.55.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.55.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.55.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.55.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.55.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.55.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.55.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.55.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.55.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.55.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.55.2.13 void AstSelection::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2371 of file Ast.cpp.

5.55.2.14 void AstSelection::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.55.2.15 void AstSelection::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.55.2.16 void AstSelection::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

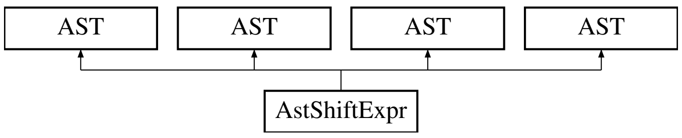
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.56 AstShiftExpr Class Reference

Inheritance diagram for AstShiftExpr:



Public Types

- enum **Operator** {
  NONE, **LEFT_OP**, **RIGHT_OP**, **NONE**,
  LEFT_OP, **RIGHT_OP**, **NONE**, **LEFT_OP**,
  RIGHT_OP, **NONE**, **LEFT_OP**, **RIGHT_OP** }
- enum **Operator** {
  NONE, **LEFT_OP**, **RIGHT_OP**, **NONE**,
  LEFT_OP, **RIGHT_OP**, **NONE**, **LEFT_OP**,
  RIGHT_OP, **NONE**, **LEFT_OP**, **RIGHT_OP** }
- enum **Operator** {
  NONE, **LEFT_OP**, **RIGHT_OP**, **NONE**,
  LEFT_OP, **RIGHT_OP**, **NONE**, **LEFT_OP**,
  RIGHT_OP, **NONE**, **LEFT_OP**, **RIGHT_OP** }
- enum **Operator** {
  NONE, **LEFT_OP**, **RIGHT_OP**, **NONE**,
  LEFT_OP, **RIGHT_OP**, **NONE**, **LEFT_OP**,
  RIGHT_OP, **NONE**, **LEFT_OP**, **RIGHT_OP** }

Public Member Functions

- **AstShiftExpr** (AstAddExpr ∗a)
- **AstShiftExpr** (AstShiftExpr ∗s, Operator o, AstAddExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstShiftExpr** (AstAddExpr ∗a)
- **AstShiftExpr** (AstShiftExpr ∗s, Operator o, AstAddExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstShiftExpr** (AstAddExpr ∗a)
- **AstShiftExpr** (AstShiftExpr ∗s, Operator o, AstAddExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.

- **AstShiftExpr** (AstAddExpr ∗a)
- **AstShiftExpr** (AstShiftExpr ∗s, Operator o, AstAddExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstShiftExpr::Operator **op**
- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab * **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstAddExpr * **add**
- AstShiftExpr * **shift**

5.56.1 Detailed Description

Definition at line 411 of file Ast.h.

5.56.2 Member Function Documentation

5.56.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.56.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.56.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.56.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.56.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.56.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.56.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.56.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.56.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.56.2.10 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.56.2.11 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.56.2.12 void AST::setLabel (string *l*) `[inline], [inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.56.2.13 void AstShiftExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1019 of file Ast.cpp.

5.56.2.14 void AstShiftExpr::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.56.2.15 void AstShiftExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.56.2.16 void AstShiftExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

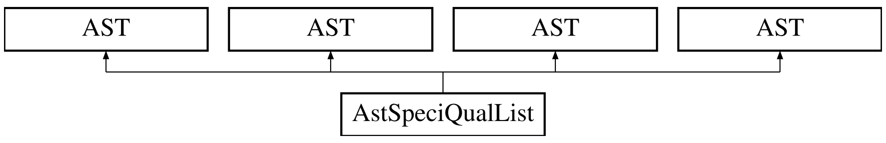
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.57 AstSpeciQualList Class Reference

Inheritance diagram for AstSpeciQualList:



Public Member Functions

- **AstSpeciQualList** (AstTypeSpeci ∗typespeci, string qual, AstSpeciQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSpeciQualList** (AstTypeSpeci ∗typespeci, string qual, AstSpeciQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSpeciQualList** (AstTypeSpeci ∗typespeci, string qual, AstSpeciQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSpeciQualList** (AstTypeSpeci ∗typespeci, string qual, AstSpeciQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstTypeSpeci ∗ **typespeci**
- string **qual**
- AstSpeciQualList ∗ **list**

5.57.1 Detailed Description

Definition at line 1475 of file Ast.h.

5.57.2 Member Function Documentation

5.57.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.57.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.57.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.57.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.57.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.57.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.57.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.57.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.57.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.57.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.57.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.57.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.57.2.13 void AstSpeciQualList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1490 of file Ast.h.

5.57.2.14 void AstSpeciQualList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1490 of file CParser.yy.

5.57.2.15 void AstSpeciQualList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1490 of file CParser.yy.

5.57.2.16 void AstSpeciQualList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

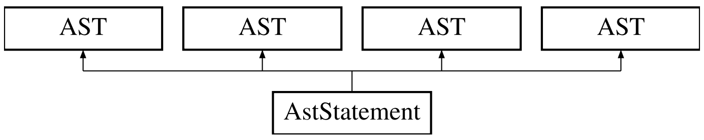
Definition at line 1490 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.58 AstStatement Class Reference

Inheritance diagram for AstStatement:



Public Types

- enum **Type** {
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP**,
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP** }
- enum **Type** {
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP**,
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP** }
- enum **Type** {
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP**,
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP** }
- enum **Type** {
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP**,
LABELED, **COMPOUND**, **EXPR**, **SELECT**,
ITER, **JUMP**, **LABELED**, **COMPOUND**,
EXPR, **SELECT**, **ITER**, **JUMP** }

Public Member Functions

- **AstStatement** (AstLabeledStmt ∗l)
- **AstStatement** (AstCompoundStmt ∗c)
- **AstStatement** (AstExprStmt ∗e)
- **AstStatement** (AstSelection ∗s)
- **AstStatement** (AstIteration ∗i)
- **AstStatement** (AstJump ∗j)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatement** (AstLabeledStmt ∗l)
- **AstStatement** (AstCompoundStmt ∗c)
- **AstStatement** (AstExprStmt ∗e)
- **AstStatement** (AstSelection ∗s)
- **AstStatement** (AstIteration ∗i)
- **AstStatement** (AstJump ∗j)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatement** (AstLabeledStmt ∗l)
- **AstStatement** (AstCompoundStmt ∗c)
- **AstStatement** (AstExprStmt ∗e)
- **AstStatement** (AstSelection ∗s)
- **AstStatement** (AstIteration ∗i)
- **AstStatement** (AstJump ∗j)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatement** (AstLabeledStmt ∗l)
- **AstStatement** (AstCompoundStmt ∗c)
- **AstStatement** (AstExprStmt ∗e)
- **AstStatement** (AstSelection ∗s)
- **AstStatement** (AstIteration ∗i)
- **AstStatement** (AstJump ∗j)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

Gets the node's label.

- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstStatement::Type **t**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstLabeledStmt $*$ **lbl**
- AstCompoundStmt $*$ **cmp**
- AstExprStmt $*$ **expr**
- AstSelection $*$ **slct**
- AstIteration $*$ **iter**
- AstJump $*$ **jump**

5.58.1 Detailed Description

Definition at line 833 of file Ast.h.

5.58.2 Member Function Documentation

5.58.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.58.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.58.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.58.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.58.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.58.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.58.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.58.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.58.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.58.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.58.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.58.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.58.2.13 void AstStatement::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2639 of file Ast.cpp.

5.58.2.14 void AstStatement::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.58.2.15 void AstStatement::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.58.2.16 void AstStatement::Visit ( ) `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

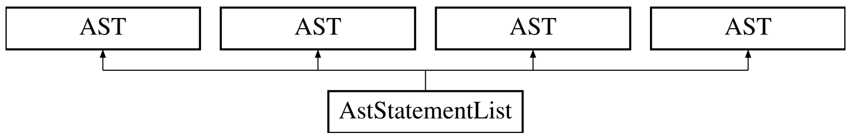
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.59 AstStatementList Class Reference

Inheritance diagram for AstStatementList:



Public Member Functions

- **AstStatementList** (AstStatement ∗s)
- **AstStatementList** (AstStatementList ∗l, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatementList** (AstStatement ∗s)
- **AstStatementList** (AstStatementList ∗l, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatementList** (AstStatement ∗s)
- **AstStatementList** (AstStatementList ∗l, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStatementList** (AstStatement ∗s)
- **AstStatementList** (AstStatementList ∗l, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstStatement $*$ **stmt**
- AstStatementList $*$ **list**

5.59.1 Detailed Description

Definition at line 1175 of file Ast.h.

5.59.2 Member Function Documentation

5.59.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.59.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.59.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.59.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.59.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.59.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.59.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.59.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.59.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.59.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.59.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.59.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.59.2.13 void AstStatementList::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2415 of file Ast.cpp.

5.59.2.14 void AstStatementList::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.59.2.15 void AstStatementList::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.59.2.16 void AstStatementList::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

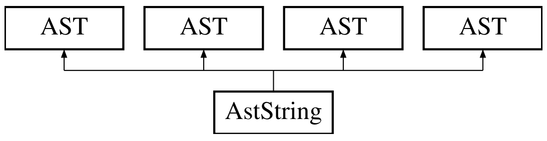
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.60 AstString Class Reference

Inheritance diagram for AstString:



Public Member Functions

- **AstString** (string str, Type ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstString** (string str, Type ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstString** (string str, Type ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- **AstString** (string str, Type ∗t)
- void Visit ()

    This function is responsible for tree traversals.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

    Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- string **val**

5.60.1 Detailed Description

Definition at line 127 of file Ast.h.

5.60.2 Member Function Documentation

5.60.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.60.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.60.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.60.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.60.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.60.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.60.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.60.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.60.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.60.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.60.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.60.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.60.2.13 void AstString::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 663 of file Ast.cpp.

5.60.2.14 void AstString::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.60.2.15 void AstString::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.60.2.16 void AstString::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

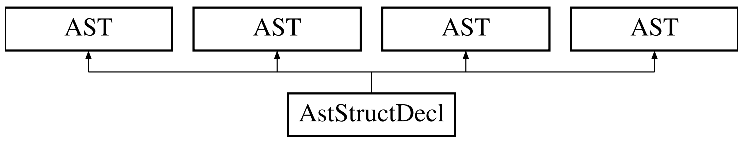
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.61 AstStructDecl Class Reference

Inheritance diagram for AstStructDecl:



Public Member Functions

- **AstStructDecl** (AstSpeciQualList ∗list, AstStructDeclList ∗declList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDecl** (AstSpeciQualList ∗list, AstStructDeclList ∗declList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDecl** (AstSpeciQualList ∗list, AstStructDeclList ∗declList)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDecl** (AstSpeciQualList ∗list, AstStructDeclList ∗declList)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstSpeciQualList $*$ **list**
- AstStructDeclList $*$ **declList**

5.61.1 Detailed Description

Definition at line 1497 of file Ast.h.

5.61.2 Member Function Documentation

5.61.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.61.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.61.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.61.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.61.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.61.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.61.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.61.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.61.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.61.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.61.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.61.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.61.2.13 void AstStructDecl::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1509 of file Ast.h.

5.61.2.14 void AstStructDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1509 of file CParser.yy.

5.61.2.15 void AstStructDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1509 of file CParser.yy.

5.61.2.16 void AstStructDecl::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

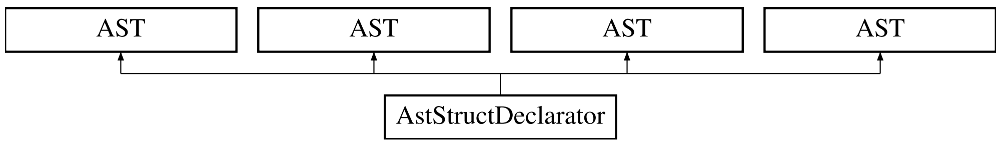
Definition at line 1509 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.62 AstStructDeclarator Class Reference

Inheritance diagram for AstStructDeclarator:



Public Member Functions

- **AstStructDeclarator** (AstDeclarator ∗decl, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclarator** (AstDeclarator ∗decl, AstExpression ∗exp)
- void Visit ()

This function is responsible for tree traversals.
- **AstStructDeclarator** (AstDeclarator ∗decl, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclarator** (AstDeclarator ∗decl, AstExpression ∗exp)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""

- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstDeclarator ∗ **decl**
- AstExpression ∗ **exp**

5.62.1 Detailed Description

Definition at line 930 of file Ast.h.

5.62.2 Member Function Documentation

5.62.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.62.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.62.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.62.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.62.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.62.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.62.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.62.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.62.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.62.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.62.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.62.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.62.2.13 void AstStructDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 941 of file Ast.h.

5.62.2.14 void AstStructDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 941 of file CParser.yy.

5.62.2.15 void AstStructDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 941 of file CParser.yy.

5.62.2.16 void AstStructDeclarator::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

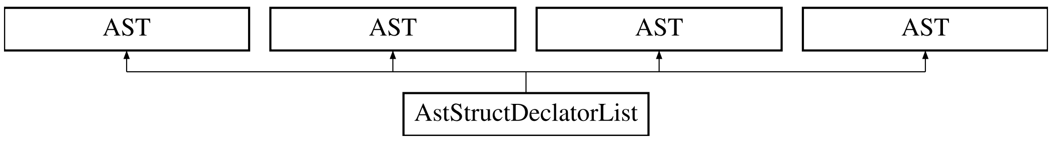
Definition at line 941 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.63 AstStructDeclatorList Class Reference

Inheritance diagram for AstStructDeclatorList:



Public Member Functions

- **AstStructDeclatorList** (AstStructDeclarator ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclatorList** (AstStructDeclarator ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclatorList** (AstStructDeclarator ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclatorList** (AstStructDeclarator ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstStructDeclarator ∗ **sdecl**
- AstStructDeclList ∗ **stdlist**

5.63.1 Detailed Description

Definition at line 911 of file Ast.h.

5.63.2 Member Function Documentation

5.63.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.63.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.63.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.63.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.63.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.63.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.63.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.63.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.63.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.63.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.63.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.63.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.63.2.13 void AstStructDeclatorList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 922 of file Ast.h.

5.63.2.14 void AstStructDeclatorList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 922 of file CParser.yy.

5.63.2.15 void AstStructDeclatorList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 922 of file CParser.yy.

5.63.2.16 void AstStructDeclatorList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

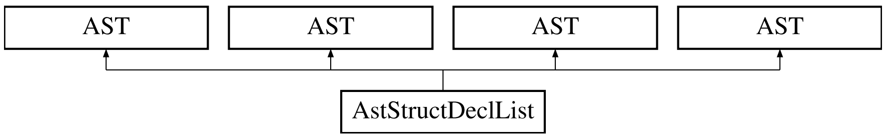
Definition at line 922 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.64 AstStructDeclList Class Reference

Inheritance diagram for AstStructDeclList:



Public Member Functions

- **AstStructDeclList** (AstStructDecl ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclList** (AstStructDecl ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclList** (AstStructDecl ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructDeclList** (AstStructDecl ∗sdecl, AstStructDeclList ∗stdlist)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstStructDecl $*$ **sdecl**
- AstStructDeclList $*$ **stdlist**

5.64.1 Detailed Description

Definition at line 1153 of file Ast.h.

5.64.2 Member Function Documentation

5.64.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.64.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.64.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.64.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.64.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.64.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.64.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.64.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.64.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.64.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.64.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.64.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.64.2.13 void AstStructDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1165 of file Ast.h.

5.64.2.14 void AstStructDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1165 of file CParser.yy.

5.64.2.15 void AstStructDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1165 of file CParser.yy.

5.64.2.16 void AstStructDeclList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

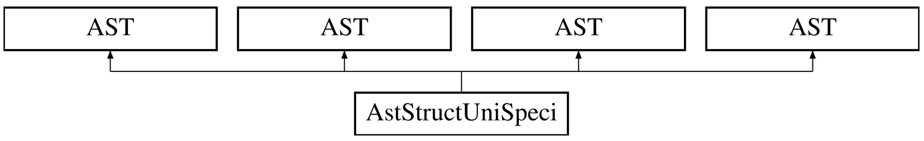
Definition at line 1165 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.65 AstStructUniSpeci Class Reference

Inheritance diagram for AstStructUniSpeci:



Public Member Functions

- **AstStructUniSpeci** (string structOrUnion, AstID ∗sName, AstStructDeclList ∗sdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructUniSpeci** (string structOrUnion, AstID ∗sName, AstStructDeclList ∗sdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructUniSpeci** (string structOrUnion, AstID ∗sName, AstStructDeclList ∗sdlist)
- void Visit ()

  This function is responsible for tree traversals.
- **AstStructUniSpeci** (string structOrUnion, AstID ∗sName, AstStructDeclList ∗sdlist)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- string **structOrUnion**
- AstID $*$ **structName**
- AstStructDeclList $*$ **sdlist**

5.65.1 Detailed Description

Definition at line 885 of file Ast.h.

5.65.2 Member Function Documentation

5.65.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.65.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.65.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.65.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.65.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.65.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.65.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.65.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.65.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.65.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.65.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.65.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.65.2.13 void AstStructUniSpeci::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 900 of file Ast.h.

5.65.2.14 void AstStructUniSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 900 of file CParser.yy.

5.65.2.15 void AstStructUniSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 900 of file CParser.yy.

5.65.2.16 void AstStructUniSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

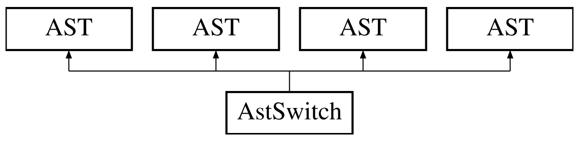
Definition at line 900 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.66 AstSwitch Class Reference

Inheritance diagram for AstSwitch:



Public Member Functions

- **AstSwitch** (AstExpression ∗e, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSwitch** (AstExpression ∗e, AstStatement ∗s)
- void Visit ()

This function is responsible for tree traversals.

- **AstSwitch** (AstExpression ∗e, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- **AstSwitch** (AstExpression ∗e, AstStatement ∗s)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""

- static string returnLabel =""
  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid
  The unique id.
- string label
  The label to be printed in the visualization.

Private Attributes

- AstExpression $*$ **expr**
- AstStatement $*$ **stmt**

5.66.1 Detailed Description

Definition at line 753 of file Ast.h.

5.66.2 Member Function Documentation

5.66.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.66.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.66.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.66.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.66.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.66.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.66.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.66.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.66.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.66.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.66.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.66.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.66.2.13 void AstSwitch::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2285 of file Ast.cpp.

5.66.2.14 void AstSwitch::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.66.2.15 void AstSwitch::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.66.2.16 void AstSwitch::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

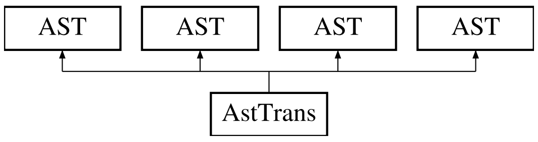
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.67 AstTrans Class Reference

Inheritance diagram for AstTrans:



Public Member Functions

- **AstTrans** (AstTrans ∗trans, AstExternDec ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTrans** (AstTrans ∗trans, AstExternDec ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTrans** (AstTrans ∗trans, AstExternDec ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTrans** (AstTrans ∗trans, AstExternDec ∗dec)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstTrans ∗ **trans**
- AstExternDec ∗ **dec**

5.67.1 Detailed Description

Definition at line 1132 of file Ast.h.

5.67.2 Member Function Documentation

5.67.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.67.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.67.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.67.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.67.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.67.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.67.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.67.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.67.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.67.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.67.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.67.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.67.2.13 void AstTrans::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1144 of file Ast.h.

5.67.2.14 void AstTrans::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1144 of file CParser.yy.

5.67.2.15 void AstTrans::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1144 of file CParser.yy.

5.67.2.16 void AstTrans::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

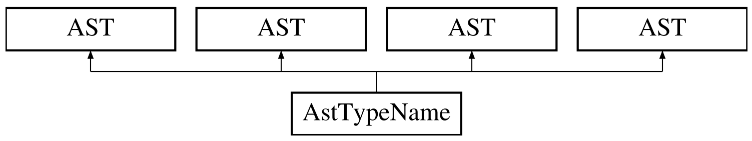
Definition at line 1144 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.68 AstTypeName Class Reference

Inheritance diagram for AstTypeName:



Public Member Functions

- **AstTypeName** (AstSpeciQualList ∗list, AstAbstractDecl ∗decl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeName** (AstSpeciQualList ∗list, AstAbstractDecl ∗decl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeName** (AstSpeciQualList ∗list, AstAbstractDecl ∗decl)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeName** (AstSpeciQualList ∗list, AstAbstractDecl ∗decl)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstSpeciQualList $*$ **list**
- AstAbstractDecl $*$ **decl**

5.68.1 Detailed Description

Definition at line 100 of file Ast.h.

5.68.2 Member Function Documentation

5.68.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.68.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.68.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.68.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.68.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.68.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.68.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.68.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.68.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.68.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.68.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.68.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.68.2.13 void AstTypeName::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 640 of file Ast.cpp.

5.68.2.14 void AstTypeName::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.68.2.15 void AstTypeName::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.68.2.16 void AstTypeName::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

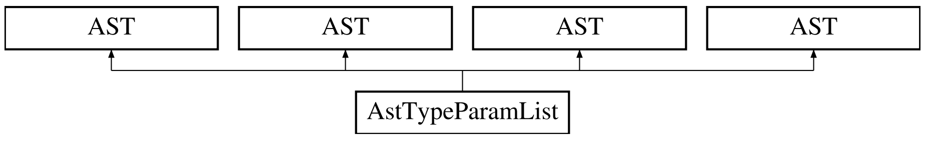
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.69 AstTypeParamList Class Reference

Inheritance diagram for AstTypeParamList:



Public Member Functions

- **AstTypeParamList** (int type, AstParamList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeParamList** (int type, AstParamList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeParamList** (int type, AstParamList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeParamList** (int type, AstParamList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- int **type**
- AstParamList $*$ **list**

5.69.1 Detailed Description

Definition at line 1317 of file Ast.h.

5.69.2 Member Function Documentation

5.69.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.69.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.69.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.69.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.69.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.69.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.69.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.69.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.69.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.69.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.69.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.69.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.69.2.13 void AstTypeParamList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1328 of file Ast.h.

5.69.2.14 void AstTypeParamList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1328 of file CParser.yy.

5.69.2.15 void AstTypeParamList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1328 of file CParser.yy.

5.69.2.16 void AstTypeParamList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

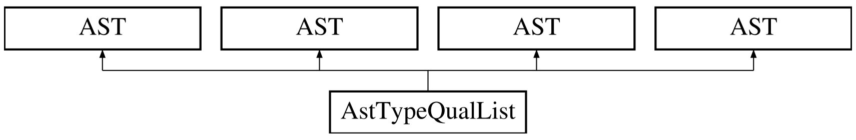
Definition at line 1328 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.70 AstTypeQualList Class Reference

Inheritance diagram for AstTypeQualList:



Public Member Functions

- **AstTypeQualList** (string type_qual, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeQualList** (string type_qual, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeQualList** (string type_qual, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeQualList** (string type_qual, AstTypeQualList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.

- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- string **type_qual**
- AstTypeQualList ∗ **list**

5.70.1 Detailed Description

Definition at line 1110 of file Ast.h.

5.70.2 Member Function Documentation

5.70.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.70.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.70.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.70.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.70.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.70.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.70.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.70.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.70.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.70.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.70.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.70.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.70.2.13 void AstTypeQualList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1122 of file Ast.h.

5.70.2.14 void AstTypeQualList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1122 of file CParser.yy.

5.70.2.15 void AstTypeQualList::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1122 of file CParser.yy.

5.70.2.16 void AstTypeQualList::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

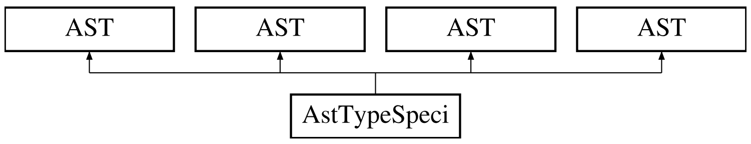
Definition at line 1122 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.71 AstTypeSpeci Class Reference

Inheritance diagram for AstTypeSpeci:



Public Member Functions

- **AstTypeSpeci** (string stypeName, AstStructUniSpeci ∗stspeci, EnumSpecifier ∗espci)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeSpeci** (string stypeName, AstStructUniSpeci ∗stspeci, EnumSpecifier ∗espci)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeSpeci** (string stypeName, AstStructUniSpeci ∗stspeci, EnumSpecifier ∗espci)
- void Visit ()

  This function is responsible for tree traversals.
- **AstTypeSpeci** (string stypeName, AstStructUniSpeci ∗stspeci, EnumSpecifier ∗espci)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- string **stypeName**
- AstStructUniSpeci ∗ **stspeci**
- EnumSpecifier ∗ **espci**

5.71.1 Detailed Description

Definition at line 1086 of file Ast.h.

5.71.2 Member Function Documentation

5.71.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.71.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.71.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.71.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.71.2.5 int AST::getUID ( ) `[inline], [inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.71.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.71.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.71.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.71.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.71.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.71.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.71.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.71.2.13 void AstTypeSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1102 of file Ast.h.

5.71.2.14 void AstTypeSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1102 of file CParser.yy.

5.71.2.15 void AstTypeSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1102 of file CParser.yy.

5.71.2.16 void AstTypeSpeci::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

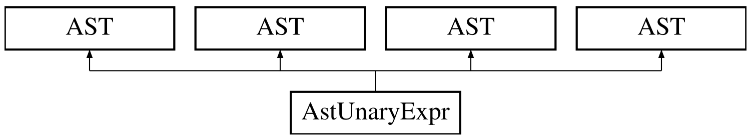
Definition at line 1102 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.72 AstUnaryExpr Class Reference

Inheritance diagram for AstUnaryExpr:



Public Types

- enum **ExprType** {
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE**,
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE** }
- enum **ExprType** {
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE**,
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE** }
- enum **ExprType** {
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE**,
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE** }
- enum **ExprType** {
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE**,
  POSTFIX, **INC**, **DEC**, **CAST**,
  SIZEOF, **SIZEOF_TYPE**, **POSTFIX**, **INC**,
  DEC, **CAST**, **SIZEOF**, **SIZEOF_TYPE** }

Public Member Functions

- **AstUnaryExpr** (AstPostfixExpr ∗e)
- **AstUnaryExpr** (AstUnaryExpr ∗e, bool inc)
- **AstUnaryExpr** (AstUnaryOp ∗o, AstCastExpr ∗c)
- **AstUnaryExpr** (AstUnaryExpr ∗e)
- **AstUnaryExpr** (AstTypeName ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryExpr** (AstPostfixExpr ∗e)
- **AstUnaryExpr** (AstUnaryExpr ∗e, bool inc)
- **AstUnaryExpr** (AstUnaryOp ∗o, AstCastExpr ∗c)
- **AstUnaryExpr** (AstUnaryExpr ∗e)

- **AstUnaryExpr** (AstTypeName ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryExpr** (AstPostfixExpr ∗e)
- **AstUnaryExpr** (AstUnaryExpr ∗e, bool inc)
- **AstUnaryExpr** (AstUnaryOp ∗o, AstCastExpr ∗c)
- **AstUnaryExpr** (AstUnaryExpr ∗e)
- **AstUnaryExpr** (AstTypeName ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryExpr** (AstPostfixExpr ∗e)
- **AstUnaryExpr** (AstUnaryExpr ∗e, bool inc)
- **AstUnaryExpr** (AstUnaryOp ∗o, AstCastExpr ∗c)
- **AstUnaryExpr** (AstUnaryExpr ∗e)
- **AstUnaryExpr** (AstTypeName ∗t)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- enum AstUnaryExpr::ExprType **t**
- Type ∗ **type**
- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstPostfixExpr ∗ **expr**
- bool **isINC**
- bool **isDEC**
- AstUnaryOp ∗ **op**
- AstCastExpr ∗ **cast**
- AstUnaryExpr ∗ **uniexpr**
- AstTypeName ∗ **tname**

5.72.1 Detailed Description

Definition at line 309 of file Ast.h.

5.72.2 Member Function Documentation

5.72.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.72.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.72.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.72.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.72.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.72.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.72.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.72.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.72.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.72.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.72.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.72.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.72.2.13 void AstUnaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 479 of file Ast.cpp.

5.72.2.14 void AstUnaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.72.2.15 void AstUnaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.72.2.16 void AstUnaryExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

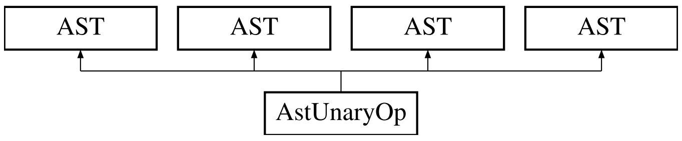
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.73 AstUnaryOp Class Reference

Inheritance diagram for AstUnaryOp:



Public Types

- enum **Operator** {
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,
  PLUS, **MINUS**, **TILDE**, **BANG**,
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,
  PLUS, **MINUS**, **TILDE**, **BANG** }
- enum **Operator** {
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,
  PLUS, **MINUS**, **TILDE**, **BANG**,
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,
  PLUS, **MINUS**, **TILDE**, **BANG** }
- enum **Operator** {
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,
  PLUS, **MINUS**, **TILDE**, **BANG**,
  BIN_AND, **STAR**, **PLUS**, **MINUS**,
  TILDE, **BANG**, **BIN_AND**, **STAR**,

PLUS, **MINUS**, **TILDE**, **BANG** }

- enum **Operator** {
BIN_AND, **STAR**, **PLUS**, **MINUS**,
TILDE, **BANG**, **BIN_AND**, **STAR**,
PLUS, **MINUS**, **TILDE**, **BANG**,
BIN_AND, **STAR**, **PLUS**, **MINUS**,
TILDE, **BANG**, **BIN_AND**, **STAR**,
PLUS, **MINUS**, **TILDE**, **BANG** }

Public Member Functions

- **AstUnaryOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- **AstUnaryOp** (Operator o)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- Operator **op**

5.73.1 Detailed Description

Definition at line 171 of file Ast.h.

5.73.2 Member Function Documentation

5.73.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.73.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.73.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.73.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.73.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.73.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.73.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.73.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.73.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.73.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.73.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.73.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.73.2.13 void AstUnaryOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 779 of file Ast.cpp.

5.73.2.14 void AstUnaryOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.73.2.15 void AstUnaryOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.73.2.16 void AstUnaryOp::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

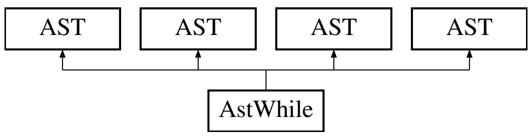
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.74 AstWhile Class Reference

Inheritance diagram for AstWhile:



Public Member Functions

- **AstWhile** (AstExpression ∗test, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstWhile** (AstExpression ∗test, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- **AstWhile** (AstExpression ∗test, AstStatement ∗statement)
- void Visit ()

This function is responsible for tree traversals.

- **AstWhile** (AstExpression ∗test, AstStatement ∗statement)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstExpression ∗ **test**
- AstStatement ∗ **statement**

5.74.1 Detailed Description

Definition at line 711 of file Ast.h.

5.74.2 Member Function Documentation

5.74.2.1 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.74.2.2 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.74.2.3 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.74.2.4 string AST::getLabel ( ) `[inline]`, `[inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.74.2.5 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.74.2.6 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.74.2.7 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.74.2.8 int AST::getUID ( ) [inline],[inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.74.2.9 void AST::setLabel (string *l*) [inline],[inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.74.2.10 void AST::setLabel (string *l*) [inline],[inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.74.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.74.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.74.2.13 void AstWhile::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 2125 of file Ast.cpp.

5.74.2.14 void AstWhile::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.74.2.15 void AstWhile::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.74.2.16 void AstWhile::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

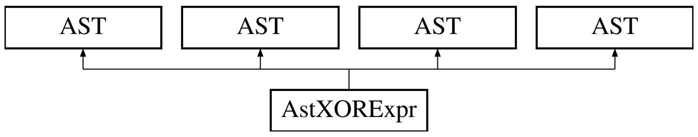
Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.75 AstXORExpr Class Reference

Inheritance diagram for AstXORExpr:



Public Member Functions

- **AstXORExpr** (AstAndExpr ∗a)
- **AstXORExpr** (AstXORExpr ∗x, AstAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstXORExpr** (AstAndExpr ∗a)
- **AstXORExpr** (AstXORExpr ∗x, AstAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstXORExpr** (AstAndExpr ∗a)
- **AstXORExpr** (AstXORExpr ∗x, AstAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- **AstXORExpr** (AstAndExpr ∗a)
- **AstXORExpr** (AstXORExpr ∗x, AstAndExpr ∗a)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.

- int getUID ()

    Gets the node's unique ID.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.
- string getLabel ()

    Gets the node's label.

Public Attributes

- Type ∗ **type**
- bool needsCast

    This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

    Indicates is a conversion is possible.
- CONVERSIONTYPE convType

    If needsCast is true, then this indicates what the cast should be.
- int operandToCast

    This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

    Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

    Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

    This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list< string > **tempStack**
- static string **lastID** =""
- static SymTab ∗ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

    The unique id.
- string label

    The label to be printed in the visualization.

Private Attributes

- AstAndExpr ∗ **a**
- AstXORExpr ∗ **x**

5.75.1 Detailed Description

Definition at line 499 of file Ast.h.

5.75.2 Member Function Documentation

5.75.2.1 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.75.2.2 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.75.2.3 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.75.2.4 string AST::getLabel ( ) [inline], [inherited]

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.75.2.5 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.75.2.6 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.75.2.7 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.75.2.8 int AST::getUID ( ) [inline], [inherited]

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.75.2.9 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.75.2.10 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.75.2.11 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.75.2.12 void AST::setLabel (string *l*) [inline], [inherited]

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.75.2.13 void AstXORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1336 of file Ast.cpp.

5.75.2.14 void AstXORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.75.2.15 void AstXORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

5.75.2.16 void AstXORExpr::Visit ( ) [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

The documentation for this class was generated from the following files:

- Ast.h
- Ast.cpp

5.76 AVLTree< DataItem > Class Template Reference

An implementation of a balanced binary tree called an AVL tree.

```
#include <AvlTree.h>
```

Classes

- struct Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.

Public Member Functions

- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.
- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()
- AVLTree ()

  Default AVL tree constructor.
- void Insert (DataItem item)

  Inserts a new node into the tree.

- void Insert (DataItem item, Node ∗&node, int &change)

  Inserts a new node into the tree.
- DataItem ∗ Fetch (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- Node ∗ Find (DataItem itemToFind)

  Searches the AVL tree for a given DataItem.
- bool Contains (DataItem itemToFind)

  Checks if the given data item is in the tree or not.
- void Dump ()

  Outputs the AVL tree to stdout recursively using the insertion operator for the DataItem template type.
- list< DataItem > **GetElements** ()

Private Member Functions

- int SingleRotate (Node ∗&rootNode, int direction)

  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)

  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)

  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)

  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)

  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)

  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)

  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)

  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)

  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)

  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)

  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)

  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)

  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)

  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)

  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)

  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)

Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)
  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)
  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)
  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)
  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)
  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)
  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)
  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)
  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)
  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)
  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)
  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)
  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)
  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)
  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)
  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)
- int SingleRotate (Node ∗&rootNode, int direction)
  Performs a single rotation in the indicated direction and about the specified node.
- int DoubleRotate (Node ∗&rootNode, int direction)
  Performs a double rotation in the indicated direction and about the specified node.
- int Balance (Node ∗&rootNode)
  Balances the tree beginning at the provided root node using single and double rotations.
- void Dump (Node ∗node)
  Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.
- list< DataItem > **GetElements** (Node ∗node)

Private Attributes

- Node ∗ root

  Root node of the AVL tree.
- list< DataItem > **elements**

5.76.1 Detailed Description

template<class DataItem>class AVLTree< DataItem >

An implementation of a balanced binary tree called an AVL tree.

This implementation assumes that the DataItem template class implements the operators < (less than), > (greater than), == (equality), and << (insertion).

Definition at line 34 of file AvlTree.h.

5.76.2 Constructor & Destructor Documentation

5.76.2.1 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) `[inline]`

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 191 of file AvlTree.h.

5.76.2.2 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) `[inline]`

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 214 of file CParser.yy.

5.76.2.3 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) `[inline]`

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CParser.yy.

5.76.2.4 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) `[inline]`

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CParser.yy.

5.76.2.5 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) `[inline]`

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 260 of file CParser.yy.

5.76.2.6 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) [inline]

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CParser.yy.

5.76.2.7 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) [inline]

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CParser.yy.

5.76.2.8 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) [inline]

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 229 of file CScanner.ll.

5.76.2.9 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) [inline]

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CScanner.ll.

5.76.2.10 template<class DataItem> AVLTree< DataItem >::AVLTree ( ) [inline]

Default AVL tree constructor.

Only sets the root of the tree to NULL since there are no nodes in the initial tree.

Definition at line 192 of file CScanner.ll.

5.76.3 Member Function Documentation

5.76.3.1 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 126 of file AvlTree.h.

5.76.3.2 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CParser.yy.

5.76.3.3 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CParser.yy.

5.76.3.4 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CParser.yy.

5.76.3.5 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CScanner.ll.

5.76.3.6 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CScanner.ll.

5.76.3.7 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 127 of file CParser.yy.

5.76.3.8 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 149 of file CParser.yy.

5.76.3.9 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 164 of file CScanner.ll.

5.76.3.10 template<class DataItem> int AVLTree< DataItem >::Balance (Node *& *rootNode*) [inline], [private]

Balances the tree beginning at the provided root node using single and double rotations.

Parameters

rootNode	The root node of a tree to balance (not necessarily the root of the entire AVL tree)

See Also

SingleRotate()
DoubleRotate()

Returns

The height change

Definition at line 195 of file CParser.yy.

5.76.3.11 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 333 of file AvlTree.h.

5.76.3.12 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CParser.yy.

5.76.3.13 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CParser.yy.

5.76.3.14 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) [inline]

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CParser.yy.

5.76.3.15 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) [inline]

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CParser.yy.

5.76.3.16 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) [inline]

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CScanner.ll.

5.76.3.17 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) [inline]

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 334 of file CScanner.ll.

5.76.3.18 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 356 of file CParser.yy.

5.76.3.19 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 371 of file CScanner.ll.

5.76.3.20 template<class DataItem> bool AVLTree< DataItem >::Contains (DataItem *itemToFind*) `[inline]`

Checks if the given data item is in the tree or not.

Parameters

itemToFind	A data item to search for

Returns

TRUE if the item is in the tree, FALSE otherwise

Definition at line 402 of file CParser.yy.

5.76.3.21 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node *& *rootNode*, int *direction*) `[inline]`, `[private]`

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 93 of file AvlTree.h.

5.76.3.22 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node *& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CParser.yy.

5.76.3.23 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node *& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CParser.yy.

5.76.3.24 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node *& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CScanner.ll.

5.76.3.25 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CParser.yy.

5.76.3.26 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CScanner.ll.

5.76.3.27 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode,* int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 94 of file CParser.yy.

5.76.3.28 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode,* int *direction*) `[inline]`, `[private]`

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 116 of file CParser.yy.

5.76.3.29 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode,* int *direction*) `[inline]`, `[private]`

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 131 of file CScanner.ll.

5.76.3.30 template<class DataItem> int AVLTree< DataItem >::DoubleRotate (Node ∗& *rootNode*, int *direction*) [inline], [private]

Performs a double rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

See Also

SingleRotate()

Returns

The height change (always 1)

Definition at line 162 of file CParser.yy.

5.76.3.31 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 163 of file AvlTree.h.

5.76.3.32 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CParser.yy.

5.76.3.33 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CParser.yy.

5.76.3.34 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CParser.yy.

5.76.3.35 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CScanner.ll.

5.76.3.36 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CScanner.ll.

5.76.3.37 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 164 of file CParser.yy.

5.76.3.38 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 186 of file CParser.yy.

5.76.3.39 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 201 of file CScanner.ll.

5.76.3.40 template<class DataItem> void AVLTree< DataItem >::Dump (Node ∗ *node*) [inline], [private]

Outputs the AVL tree to stdout using the << operator of the provided DataItem template type.

Parameters

node	The current node to print and recursively print the children of

Definition at line 232 of file CParser.yy.

5.76.3.41 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 269 of file AvlTree.h.

5.76.3.42 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CParser.yy.

5.76.3.43 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CParser.yy.

5.76.3.44 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CParser.yy.

5.76.3.45 template<class DataItem> DataItem* AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CParser.yy.

5.76.3.46 template<class DataItem> DataItem* AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CScanner.ll.

5.76.3.47 template<class DataItem> DataItem* AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 270 of file CScanner.ll.

5.76.3.48 template<class DataItem> DataItem* AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 292 of file CParser.yy.

5.76.3.49 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 307 of file CScanner.ll.

5.76.3.50 template<class DataItem> DataItem∗ AVLTree< DataItem >::Fetch (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the item data in the tree otherwise

Definition at line 338 of file CParser.yy.

5.76.3.51 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 303 of file AvlTree.h.

5.76.3.52 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CParser.yy.

5.76.3.53 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CParser.yy.

5.76.3.54 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CParser.yy.

5.76.3.55 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CParser.yy.

5.76.3.56 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CScanner.ll.

5.76.3.57 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 304 of file CScanner.ll.

5.76.3.58 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 326 of file CParser.yy.

5.76.3.59 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) [inline]

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 341 of file CScanner.ll.

5.76.3.60 template<class DataItem> Node∗ AVLTree< DataItem >::Find (DataItem *itemToFind*) `[inline]`

Searches the AVL tree for a given DataItem.

Returns a pointer to the node containing the data rather than the a pointer to the actual data.

Parameters

itemToFind	The item to search for

Returns

NULL if the item wasn't found, or a pointer to the node which contains the given data item otherwise

Definition at line 372 of file CParser.yy.

5.76.3.61 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 202 of file AvlTree.h.

5.76.3.62 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CParser.yy.

5.76.3.63 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CParser.yy.

5.76.3.64 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CScanner.ll.

5.76.3.65 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CParser.yy.

5.76.3.66 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CParser.yy.

5.76.3.67 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 203 of file CScanner.ll.

5.76.3.68 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 217 of file AvlTree.h.

5.76.3.69 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item*, **Node** ∗& *node*, int & *change*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CParser.yy.

5.76.3.70 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item*, **Node** ∗& *node*, int & *change*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CScanner.ll.

5.76.3.71 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item*, **Node** ∗& *node*, int & *change*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CParser.yy.

5.76.3.72 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CParser.yy.

5.76.3.73 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CScanner.ll.

5.76.3.74 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 218 of file CParser.yy.

5.76.3.75 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 225 of file CParser.yy.

5.76.3.76 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 240 of file CParser.yy.

5.76.3.77 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 240 of file CScanner.ll.

5.76.3.78 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item,* Node ∗& *node,* int & *change*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 255 of file CScanner.ll.

5.76.3.79 template<class DataItem> void **AVLTree**< DataItem >::Insert (DataItem *item*) `[inline]`

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree

Definition at line 271 of file CParser.yy.

5.76.3.80 template<class DataItem> void AVLTree< DataItem >::Insert (DataItem *item*, Node ∗& *node*, int & *change*) [inline]

Inserts a new node into the tree.

Parameters

item	The new data to create a node for and to insert into the tree
node	The root of the tree to insert the node into
change	Boolean indicating if the tree has been changed or not

See Also

Insert(DataItem)

Definition at line 286 of file CParser.yy.

5.76.3.81 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node ∗& *rootNode*, int *direction*) [inline], [private]

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 69 of file AvlTree.h.

5.76.3.82 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node ∗& *rootNode*, int *direction*) [inline], [private]

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CParser.yy.

5.76.3.83 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node ∗& *rootNode*, int *direction*) [inline], [private]

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CScanner.ll.

5.76.3.84 template<class DataItem> int **AVLTree**< DataItem >::SingleRotate (**Node** ∗& *rootNode,* int *direction*) `[inline]`, `[private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CParser.yy.

5.76.3.85 template<class DataItem> int **AVLTree**< DataItem >::SingleRotate (**Node** ∗& *rootNode,* int *direction*) `[inline]`, `[private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CParser.yy.

5.76.3.86 template<class DataItem> int **AVLTree**< DataItem >::SingleRotate (**Node** ∗& *rootNode,* int *direction*) `[inline]`, `[private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CScanner.ll.

5.76.3.87 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node *& *rootNode,* int *direction*) `[inline], [private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 70 of file CParser.yy.

5.76.3.88 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node *& *rootNode,* int *direction*) `[inline], [private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 92 of file CParser.yy.

5.76.3.89 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node *& *rootNode,* int *direction*) `[inline], [private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 107 of file CScanner.ll.

5.76.3.90 template<class DataItem> int AVLTree< DataItem >::SingleRotate (Node *& *rootNode,* int *direction*) `[inline], [private]`

Performs a single rotation in the indicated direction and about the specified node.

Parameters

rootNode	Node to rotate about
direction	the direction in which to rotate

Returns

The height change (either 1 or 0)

Definition at line 138 of file CParser.yy.

The documentation for this class was generated from the following files:

- AvlTree.h
- CParser.yy
- CScanner.ll

5.77 CCompiler Class Reference

A minimalist C programming language compiler class.

```
#include <CCompiler.h>
```

Public Member Functions

- CCompiler ()

  Default constructor.
- virtual ∼CCompiler ()

  Destructor.
- void scan_begin (bool debug_scanning)

  Initializes the scanning process.
- void scan_end ()

  Closes the different output streams used in the scanner.
- int parse (const std::string &fname)

  Runs the parsing (and consequently the scanning) process.
- void setOutfile (std::string fname)

  Sets the filename of the standard compiler output file stream and opens the stream.
- yy::CParser::token::yytokentype checkType (char ∗key, const yy::location &loc, SymbolInfo ∗sym)

  Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.
- void allocateSymbol ()

  Allocates a new current symbol.
- void globalScope ()

  Initializes the global scope for the input program.
- void enterScope ()

  Enters a new scope in the input program.
- void leaveScope ()

  Leaves a scope in the input program.
- void set_insert_mode (bool iMode)

  Sets the insert mode flag to the provided value.
- bool get_insert_mode ()

  Gets TRUE if the compiler is currently in insert mode and false otherwise.
- void error (const yy::location &loc, const std::string &msg)

Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.

- void error (const std::string &msg)

  Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.
- void warning (const yy::location &loc, const std::string &msg)

  Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.
- void warning (const std::string &msg)

  Prints a warning to the standard compiler output stream.
- void printTok (std::string ttxt)

  Prints the recognized token to the token file.
- void printTok (std::string ttxt, char ∗yytext)

  Prints the recognized token to the token file with the matched text.
- void printRed (std::string ptxt)

  Prints a reduction to the reduction file.
- void turnDebugOn (bool flag)

  Sets the debug output flag to the value provided.
- void printDebug (std::string txt)

  Prints a debug string to the standard compiler output stream if the debug output flag is set to true.
- void save_line (int i, string s)

  Stores a line of code in the input code buffer.
- CCompiler ()

  Default constructor.
- virtual ∼CCompiler ()

  Destructor.
- void scan_begin (bool debug_scanning)

  Initializes the scanning process.
- void scan_end ()

  Closes the different output streams used in the scanner.
- int parse (const std::string &fname)

  Runs the parsing (and consequently the scanning) process.
- void setOutfile (std::string fname)

  Sets the filename of the standard compiler output file stream and opens the stream.
- yy::CParser::token::yytokentype checkType (char ∗key, const yy::location &loc, SymbolInfo ∗sym)

  Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.
- void allocateSymbol ()

  Allocates a new current symbol.
- void globalScope ()

  Initializes the global scope for the input program.
- void enterScope ()

  Enters a new scope in the input program.
- void leaveScope ()

  Leaves a scope in the input program.
- void set_insert_mode (bool iMode)

  Sets the insert mode flag to the provided value.
- bool get_insert_mode ()

  Gets TRUE if the compiler is currently in insert mode and false otherwise.
- void error (const yy::location &loc, const std::string &msg)

  Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.
- void error (const std::string &msg)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

- void warning (const yy::location &loc, const std::string &msg)

  Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.
- void warning (const std::string &msg)

  Prints a warning to the standard compiler output stream.
- void printTok (std::string ttxt)

  Prints the recognized token to the token file.
- void printTok (std::string ttxt, char ∗yytext)

  Prints the recognized token to the token file with the matched text.
- void printRed (std::string ptxt)

  Prints a reduction to the reduction file.
- void turnDebugOn (bool flag)

  Sets the debug output flag to the value provided.
- void printDebug (std::string txt)

  Prints a debug string to the standard compiler output stream if the debug output flag is set to true.
- void save_line (int i, string s)

  Stores a line of code in the input code buffer.
- CCompiler ()

  Default constructor.
- virtual ∼CCompiler ()

  Destructor.
- void scan_begin (bool debug_scanning)

  Initializes the scanning process.
- void scan_end ()

  Closes the different output streams used in the scanner.
- int parse (const std::string &fname)

  Runs the parsing (and consequently the scanning) process.
- void setOutfile (std::string fname)

  Sets the filename of the standard compiler output file stream and opens the stream.
- yy::CParser::token::yytokentype checkType (char ∗key, const yy::location &loc, SymbolInfo ∗sym)

  Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.
- void allocateSymbol ()

  Allocates a new current symbol.
- void globalScope ()

  Initializes the global scope for the input program.
- void enterScope ()

  Enters a new scope in the input program.
- void leaveScope ()

  Leaves a scope in the input program.
- void set_insert_mode (bool iMode)

  Sets the insert mode flag to the provided value.
- bool get_insert_mode ()

  Gets TRUE if the compiler is currently in insert mode and false otherwise.
- void error (const yy::location &loc, const std::string &msg)

  Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.
- void error (const std::string &msg)

  Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.
- void warning (const yy::location &loc, const std::string &msg)

Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.

- void warning (const std::string &msg)

  Prints a warning to the standard compiler output stream.
- void printTok (std::string ttxt)

  Prints the recognized token to the token file.
- void printTok (std::string ttxt, char ∗yytext)

  Prints the recognized token to the token file with the matched text.
- void printRed (std::string ptxt)

  Prints a reduction to the reduction file.
- void turnDebugOn (bool flag)

  Sets the debug output flag to the value provided.
- void printDebug (std::string txt)

  Prints a debug string to the standard compiler output stream if the debug output flag is set to true.
- void save_line (int i, string s)

  Stores a line of code in the input code buffer.

Public Attributes

- int result

  A flag indicating the result of an attempted compilation.
- bool trace_scanning

  Scanner trace state.
- std::string fname

  Standard compiler output stream filename.
- bool trace_parsing

  Trace parsing flag.
- SymbolInfo ∗ currentSymbol

  The current symbol table symbol.
- Type ∗ **structMemberType**
- SymTab **SymbolTable**
- bool **anonymousEnum**
- int **structUnionMode**
- list< string > **enumConsts**
- list< SymbolInfo > **structUnionTypes**
- EnumType ∗ **enumType**
- SymbolInfo ∗ **enumSym**
- int **structVarCount**
- string **currentStorageType**
- string **currentTypeQual**
- string **currentFunctionName**
- AST ∗ **source_ast**
- bool **isFuncDef**
- bool trace_symtab

  Flag indicating whether or not trace data should be output for the symbol table.
- fstream ydbFile

  Bison debug output file.
- map< int, string > input_text

  Input code buffer.

Private Attributes

- bool debug_on

  Debug output flag.
- bool insert_mode

  Insert mode flag.
- bool outfile_set

  Output to file flag.
- fstream tFile

  Token output stream.
- fstream rFile

  Reduction output stream.
- fstream outfile

  Standard compiler output stream.

5.77.1 Detailed Description

A minimalist C programming language compiler class.

This class handles the entire operation of compiling C code, from lexing and parsing, to walking the AST, generating 3AC, and generating ASM.

Definition at line 49 of file CCompiler.h.

5.77.2 Member Function Documentation

5.77.2.1 yy::CParser::token::yytokentype CCompiler::checkType (char ∗ *key,* const yy::location & *loc,* **SymbolInfo** ∗ *sym*)

Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.

Parameters

key	The string key for the token
loc	The location of the token
sym	a pointer to the symbol in the symbol table

Returns

Returns either IDENTIFIER, ENUM_CONSTANT, or TYPEDEF_NAME

5.77.2.2 yy::CParser::token::yytokentype CCompiler::checkType (char ∗ *key,* const yy::location & *loc,* **SymbolInfo** ∗ *sym*)

Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.

Parameters

key	The string key for the token
loc	The location of the token
sym	a pointer to the symbol in the symbol table

Returns

Returns either IDENTIFIER, ENUM_CONSTANT, or TYPEDEF_NAME

Definition at line 98 of file CCompiler.cpp.

5.77.2.3 yy::CParser::token::yytokentype CCompiler::checkType (char ∗ *key,* const yy::location & *loc,* SymbolInfo ∗ *sym*)

Should check the type of the symbol with the input key and insert any symbol not currently in the symbol table.

Parameters

key	The string key for the token
loc	The location of the token
sym	a pointer to the symbol in the symbol table

Returns

Returns either IDENTIFIER, ENUM_CONSTANT, or TYPEDEF_NAME

5.77.2.4 void CCompiler::error (const yy::location & *loc,* const std::string & *msg*)

Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.

Parameters

loc	The coordinates of the error
msg	The error message

5.77.2.5 void CCompiler::error (const yy::location & *loc,* const std::string & *msg*)

Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.

Parameters

loc	The coordinates of the error
msg	The error message

Definition at line 193 of file CCompiler.cpp.

5.77.2.6 void CCompiler::error (const yy::location & *loc,* const std::string & *msg*)

Prints an error to the standard compiler output stream with the coordinates of the code that caused the error and terminates the program with with an EXIT_FAILURE return value.

Parameters

loc	The coordinates of the error
msg	The error message

5.77.2.7 void CCompiler::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

5.77.2.8 void CCompiler::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

Definition at line 223 of file CCompiler.cpp.

5.77.2.9 void CCompiler::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

5.77.2.10 bool CCompiler::get_insert_mode ( )

Gets TRUE if the compiler is currently in insert mode and false otherwise.

Returns

The value of the insert mode flag;

5.77.2.11 bool CCompiler::get_insert_mode ( )

Gets TRUE if the compiler is currently in insert mode and false otherwise.

Returns

The value of the insert mode flag;

Definition at line 93 of file CCompiler.cpp.

5.77.2.12 bool CCompiler::get_insert_mode ( )

Gets TRUE if the compiler is currently in insert mode and false otherwise.

Returns

The value of the insert mode flag;

5.77.2.13 int CCompiler::parse (const std::string & *fname*)

Runs the parsing (and consequently the scanning) process.

This is equivalent to a call to a Run() like function.

Parameters

fname	The input code file filename

Returns

The result value of the parse process

Definition at line 37 of file CCompiler.cpp.

5.77.2.14 int CCompiler::parse (const std::string & *fname*)

Runs the parsing (and consequently the scanning) process.

This is equivalent to a call to a Run() like function.

Parameters

fname	The input code file filename

Returns

The result value of the parse process

5.77.2.15 int CCompiler::parse (const std::string & *fname*)

Runs the parsing (and consequently the scanning) process.

This is equivalent to a call to a Run() like function.

Parameters

fname	The input code file filename

Returns

The result value of the parse process

5.77.2.16 void CCompiler::printDebug (std::string *txt*)

Prints a debug string to the standard compiler output stream if the debug output flag is set to true.

Parameters

txt	The debug string to print

Definition at line 289 of file CCompiler.cpp.

5.77.2.17 void CCompiler::printDebug (std::string *txt*)

Prints a debug string to the standard compiler output stream if the debug output flag is set to true.

Parameters

txt	The debug string to print

5.77.2.18 void CCompiler::printDebug (std::string *txt*)

Prints a debug string to the standard compiler output stream if the debug output flag is set to true.

Parameters

txt	The debug string to print

5.77.2.19 void CCompiler::printRed (std::string *ptxt*)

Prints a reduction to the reduction file.

Parameters

ptxt	Reduction string

Definition at line 279 of file CCompiler.cpp.

5.77.2.20 void CCompiler::printRed (std::string *ptxt*)

Prints a reduction to the reduction file.

Parameters

ptxt	Reduction string

5.77.2.21 void CCompiler::printRed (std::string *ptxt*)

Prints a reduction to the reduction file.

Parameters

ptxt	Reduction string

5.77.2.22 void CCompiler::printTok (std::string *ttxt*)

Prints the recognized token to the token file.

Parameters

ttxt	The token

Definition at line 269 of file CCompiler.cpp.

5.77.2.23 void CCompiler::printTok (std::string *ttxt*)

Prints the recognized token to the token file.

Parameters

ttxt	The token

5.77.2.24 void CCompiler::printTok (std::string *ttxt*)

Prints the recognized token to the token file.

Parameters

ttxt	The token

5.77.2.25 void CCompiler::printTok (std::string *ttxt*, char ∗ *yytext*)

Prints the recognized token to the token file with the matched text.

Parameters

ttxt	The token
yytext	The matched text

5.77.2.26 void CCompiler::printTok (std::string *ttxt*, char ∗ *yytext*)

Prints the recognized token to the token file with the matched text.

Parameters

ttxt	The token
yytext	The matched text

Definition at line 274 of file CCompiler.cpp.

5.77.2.27 void CCompiler::printTok (std::string *ttxt*, char ∗ *yytext*)

Prints the recognized token to the token file with the matched text.

Parameters

ttxt	The token
yytext	The matched text

5.77.2.28 void CCompiler::save_line (int *i*, string *s*)

Stores a line of code in the input code buffer.

Parameters

i	Line number
s	Line of input code

Definition at line 300 of file CCompiler.cpp.

5.77.2.29 void CCompiler::save_line (int *i*, string *s*)

Stores a line of code in the input code buffer.

Parameters

i	Line number
s	Line of input code

5.77.2.30 void CCompiler::save_line (int *i*, string *s*)

Stores a line of code in the input code buffer.

Parameters

i	Line number
s	Line of input code

5.77.2.31 void CCompiler::scan_begin (bool *debug_scanning*)

Initializes the scanning process.

This function sets up the scanner output streams, sets the scanner's debug level, and begins the FIRSTLINE state.

Parameters

debug_scanning	Indicates if the scanner should output debug info

5.77.2.32 void CCompiler::scan_begin (bool *debug_scanning*)

Initializes the scanning process.

This function sets up the scanner output streams, sets the scanner's debug level, and begins the FIRSTLINE state.

Parameters

debug_scanning	Indicates if the scanner should output debug info

5.77.2.33 void CCompiler::scan_begin (bool *debug_scanning*)

Initializes the scanning process.

This function sets up the scanner output streams, sets the scanner's debug level, and begins the FIRSTLINE state.

Parameters

debug_scanning	Indicates if the scanner should output debug info

5.77.2.34 void CCompiler::set_insert_mode (bool *iMode*)

Sets the insert mode flag to the provided value.

Parameters

iMode	Value to set the insert mode flag to

5.77.2.35 void CCompiler::set_insert_mode (bool *iMode*)

Sets the insert mode flag to the provided value.

Parameters

iMode	Value to set the insert mode flag to

Definition at line 88 of file CCompiler.cpp.

5.77.2.36 void CCompiler::set_insert_mode (bool *iMode*)

Sets the insert mode flag to the provided value.

Parameters

iMode	Value to set the insert mode flag to

5.77.2.37 void CCompiler::setOutfile (std::string *fname*)

Sets the filename of the standard compiler output file stream and opens the stream.

Parameters

fname	The filename

Definition at line 60 of file CCompiler.cpp.

5.77.2.38 void CCompiler::setOutfile (std::string *fname*)

Sets the filename of the standard compiler output file stream and opens the stream.

Parameters

fname	The filename

5.77.2.39 void CCompiler::setOutfile (std::string *fname*)

Sets the filename of the standard compiler output file stream and opens the stream.

Parameters

fname	The filename

5.77.2.40 void CCompiler::turnDebugOn (bool *flag*)

Sets the debug output flag to the value provided.

Parameters

flag	Flag value

Definition at line 284 of file CCompiler.cpp.

5.77.2.41 void CCompiler::turnDebugOn (bool *flag*)

Sets the debug output flag to the value provided.

Parameters

flag	Flag value

5.77.2.42 void CCompiler::turnDebugOn (bool *flag*)

Sets the debug output flag to the value provided.

Parameters

flag	Flag value

5.77.2.43 void CCompiler::warning (const yy::location & *loc,* const std::string & *msg*)

Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.

Parameters

loc	The coordinates of the warning
msg	The warning message

Definition at line 233 of file CCompiler.cpp.

5.77.2.44 void CCompiler::warning (const yy::location & *loc,* const std::string & *msg*)

Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.

Parameters

loc	The coordinates of the warning
msg	The warning message

5.77.2.45 void CCompiler::warning (const yy::location & *loc,* const std::string & *msg*)

Prints a warning to the standard compiler output stream with the coordinates of the code that caused the warning.

Parameters

loc	The coordinates of the warning
msg	The warning message

5.77.2.46 void CCompiler::warning (const std::string & *msg*)

Prints a warning to the standard compiler output stream.

Parameters

msg	The warning message to print

5.77.2.47 void CCompiler::warning (const std::string & *msg*)

Prints a warning to the standard compiler output stream.

Parameters

msg	The warning message to print

Definition at line 261 of file CCompiler.cpp.

5.77.2.48 void CCompiler::warning (const std::string & *msg*)

Prints a warning to the standard compiler output stream.

Parameters

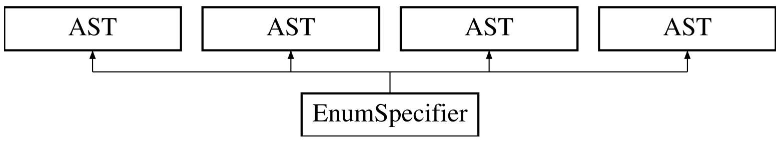
msg	The warning message to print

The documentation for this class was generated from the following files:

- CCompiler.h
- CCompiler.cpp

5.78 EnumSpecifier Class Reference

Inheritance diagram for EnumSpecifier:



Public Member Functions

- **EnumSpecifier** (AstID ∗id, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **EnumSpecifier** (AstID ∗id, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **EnumSpecifier** (AstID ∗id, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- **EnumSpecifier** (AstID ∗id, AstEnumList ∗list)
- void Visit ()

  This function is responsible for tree traversals.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- void setLabel (string l)

  Sets the label for the node.
- int getUID ()

Gets the node's unique ID.

- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- int getUID ()

  Gets the node's unique ID.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.
- string getLabel ()

  Gets the node's label.

Public Attributes

- bool needsCast

  This indicates if cast 3AC needs to be output, and is only relevant for expressions.
- bool isConv

  Indicates is a conversion is possible.
- CONVERSIONTYPE convType

  If needsCast is true, then this indicates what the cast should be.
- int operandToCast

  This indicates if the first or second operand should be the one that is cast.

Static Public Attributes

- static Visualizer vis

  Static visualizer instance for generating the visualization of the AST.
- static TAC_Generator tacGen

  Three address code generator.
- static string **currentTemp** =""
- static string returnLabel =""

  This is for storing the string id of any temporary result register that may be created during 3AC generation.
- static list$<$ string $>$ **tempStack**
- static string **lastID** =""
- static SymTab $*$ **symbolTable** =NULL
- static string **currentFunction** =""

Protected Attributes

- int uid

  The unique id.
- string label

  The label to be printed in the visualization.

Private Attributes

- AstID ∗ **id**
- AstEnumList ∗ **list**

5.78.1 Detailed Description

Definition at line 1456 of file Ast.h.

5.78.2 Member Function Documentation

5.78.2.1 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file Ast.h.

5.78.2.2 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CScanner.ll.

5.78.2.3 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.78.2.4 string AST::getLabel ( ) `[inline], [inherited]`

Gets the node's label.

Returns

The label

Definition at line 60 of file CParser.yy.

5.78.2.5 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.78.2.6 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CParser.yy.

5.78.2.7 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file CScanner.ll.

5.78.2.8 int AST::getUID ( ) `[inline]`, `[inherited]`

Gets the node's unique ID.

Returns

The unique id

Definition at line 53 of file Ast.h.

5.78.2.9 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CScanner.ll.

5.78.2.10 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.78.2.11 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file Ast.h.

5.78.2.12 void AST::setLabel (string *l*) `[inline]`, `[inherited]`

Sets the label for the node.

Parameters

l	The label string

Definition at line 43 of file CParser.yy.

5.78.2.13 void EnumSpecifier::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1468 of file Ast.h.

5.78.2.14 void EnumSpecifier::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1468 of file CParser.yy.

5.78.2.15 void EnumSpecifier::Visit ( ) `[inline]`, `[virtual]`

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

Definition at line 1468 of file CParser.yy.

5.78.2.16 void EnumSpecifier::Visit ( ) [inline], [virtual]

This function is responsible for tree traversals.

This function will call the Visit functions of each of it's children nodes, call the visualization code for itself, and output any 3AC that can be generated at the current node.

Reimplemented from AST.

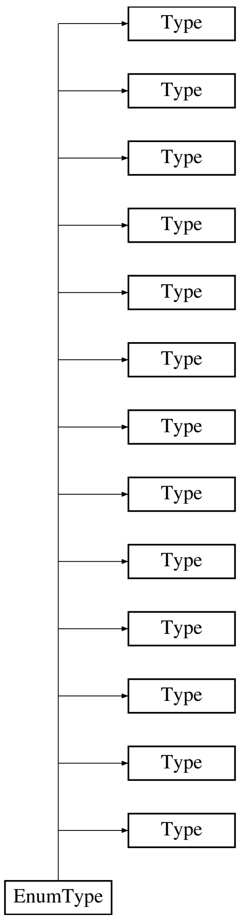
Definition at line 1468 of file CScanner.ll.

The documentation for this class was generated from the following file:

- Ast.h

5.79 EnumType Class Reference

Inheritance diagram for EnumType:



Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)

- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **EnumType** (string n, int startVal)
- int **GetConstVal** (string s)
- void **AddEnumConst** (string s)
- void **AddEnumConst** (string s, int val)
- bool **CheckType** (EnumType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()

- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- map< string, int > **enumConsts**
- int **currentVal**
- string **name**
- int **size**

5.79.1 Detailed Description

Definition at line 114 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.80 GoFPatterns::Event< SourceType, EventArgType > Class Template Reference

Classes

- class SubscriberRecord

  This inner class, for each EventHandler, stores the associated context information - pointer.

Public Member Functions

- Event (SourceType ∗source)

  Constructor - sets the Event source.
- virtual ∼Event ()

  Virtual destructor - perform clean up if any.
- void operator+= (EventHandler handler)

  Operator used to subscribe a handler C# style event subscription.
- void operator-= (EventHandler handler)

  Operator used to unsubscribe a handler C# style event subscription.
- void Subscribe (EventHandler handler, void ∗context=0)

  Function used to subscribe a handler with optional context information.
- void Unsubscribe (EventHandler handler, void ∗context=0)

  Function used to unsubscribe a handler with optional context information.
- void FireEvent (EventArgType eventArg)

  Fire the event and notify all observers with event argument, - source and context information if any provided.
- Event (SourceType ∗source)

  Constructor - sets the Event source.
- virtual ∼Event ()

  Virtual destructor - perform clean up if any.
- void operator+= (EventHandler handler)

  Operator used to subscribe a handler C# style event subscription.
- void operator-= (EventHandler handler)

  Operator used to unsubscribe a handler C# style event subscription.
- void Subscribe (EventHandler handler, void ∗context=0)

  Function used to subscribe a handler with optional context information.
- void Unsubscribe (EventHandler handler, void ∗context=0)

  Function used to unsubscribe a handler with optional context information.
- void FireEvent (EventArgType eventArg)

  Fire the event and notify all observers with event argument, - source and context information if any provided.
- Event (SourceType ∗source)

  Constructor - sets the Event source.
- virtual ∼Event ()

  Virtual destructor - perform clean up if any.
- void operator+= (EventHandler handler)

Operator used to subscribe a handler C# style event subscription.
- void operator-= (EventHandler handler)

    Operator used to unsubscribe a handler C# style event subscription.
- void Subscribe (EventHandler handler, void ∗context=0)

    Function used to subscribe a handler with optional context information.
- void Unsubscribe (EventHandler handler, void ∗context=0)

    Function used to unsubscribe a handler with optional context information.
- void FireEvent (EventArgType eventArg)

    Fire the event and notify all observers with event argument, - source and context information if any provided.

Protected Types

- typedef void(∗ EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context)

    Event handler function pointer definition source - Subject - the object which fired the event.
- typedef void(∗ EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context)

    Event handler function pointer definition source - Subject - the object which fired the event.
- typedef void(∗ EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context)

    Event handler function pointer definition source - Subject - the object which fired the event.

Protected Attributes

- vector< SubscriberRecord > **Subscribers**
- SourceType ∗ **eventSource**

5.80.1 Detailed Description

template<typename SourceType, typename EventArgType>class GoFPatterns::Event< SourceType, EventArgType >

Definition at line 25 of file Event.h.

5.80.2 Member Typedef Documentation

5.80.2.1 template<typename SourceType, typename EventArgType> typedef void(∗ GoFPatterns::Event< SourceType, EventArgType >::EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context) [protected]

Event handler function pointer definition source - Subject - the object which fired the event.

eventArg - The event argument context - Context information, which a subscriber needs to get with an event notification Usually, this can be a pointer to the subscriber object itself.

Definition at line 35 of file Event.h.

5.80.2.2 template<typename SourceType, typename EventArgType> typedef void(∗ GoFPatterns::Event< SourceType, EventArgType >::EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context) [protected]

Event handler function pointer definition source - Subject - the object which fired the event.

eventArg - The event argument context - Context information, which a subscriber needs to get with an event notification Usually, this can be a pointer to the subscriber object itself.

Definition at line 36 of file TAC_Scanner.ll.

5.80.2.3 template<typename SourceType, typename EventArgType> typedef void(∗ GoFPatterns::Event< SourceType, EventArgType >::EventHandler)(SourceType ∗source, EventArgType eventArg, void ∗context) [protected]

Event handler function pointer definition source - Subject - the object which fired the event.

eventArg - The event argument context - Context information, which a subscriber needs to get with an event notification Usually, this can be a pointer to the subscriber object itself.

Definition at line 36 of file TAC_Parser.yy.

The documentation for this class was generated from the following file:

- mips/Event.h

5.81 Function Struct Reference

Public Attributes

- string name

  The name of the function.
- bool hasReturn

  Indicates if the function has a return value;.
- list< pair< string, int > > variables

  The variables in the function for which stack space must be allocated (The name and offset of the variable).
- list< Parameter ∗ > parameters

  The parameters of the function.
- int stackSpace

  The number of bytes required on the stack to hold all of the variables of the function.

5.81.1 Detailed Description

Definition at line 25 of file FunctionTable.h.

The documentation for this struct was generated from the following file:

- mips/FunctionTable.h

5.82 FunctionTable Class Reference

Public Member Functions

- int GetStackSpace (string name)

  Returns the stack sace required for a function with the given name.
- void AddFunction (string fname, bool hasReturn=false)

  Adds a function with the given name to the table.
- Function ∗ GetFunction (string name)

  Gets a reference to a function with the given name.
- void AddVariable (string fname, string vname, int size)

  Adds a variable to the list of variables for the given function and calculates its offset.
- void AddParameter (string fname, string pname, bool byValue)

  Adds a parameter to the given function.
- int GetVarOffset (string fname, string vname)

Gets the offset of a given variable within a given function.

- int GetStackSpace (string name)

  Returns the stack sace required for a function with the given name.
- void AddFunction (string fname, bool hasReturn=false)

  Adds a function with the given name to the table.
- Function $*$ GetFunction (string name)

  Gets a reference to a function with the given name.
- void AddVariable (string fname, string vname, int size)

  Adds a variable to the list of variables for the given function and calculates its offset.
- void AddParameter (string fname, string pname, bool byValue)

  Adds a parameter to the given function.
- int GetVarOffset (string fname, string vname)

  Gets the offset of a given variable within a given function.
- int GetStackSpace (string name)

  Returns the stack sace required for a function with the given name.
- void AddFunction (string fname, bool hasReturn=false)

  Adds a function with the given name to the table.
- Function $*$ GetFunction (string name)

  Gets a reference to a function with the given name.
- void AddVariable (string fname, string vname, int size)

  Adds a variable to the list of variables for the given function and calculates its offset.
- void AddParameter (string fname, string pname, bool byValue)

  Adds a parameter to the given function.
- int GetVarOffset (string fname, string vname)

  Gets the offset of a given variable within a given function.

Private Attributes

- map$<$ string, Function $*$ $>$ **functions**

5.82.1 Detailed Description

Definition at line 54 of file FunctionTable.h.

5.82.2 Member Function Documentation

5.82.2.1 void FunctionTable::AddFunction (string *fname*, bool *hasReturn* = `false`)

Adds a function with the given name to the table.

Parameters

fname	The name of the function to add
hasReturn	Indicates if the function has a return value

Definition at line 17 of file FunctionTable.cpp.

5.82.2.2 void FunctionTable::AddFunction (string *fname*, bool *hasReturn* = `false`)

Adds a function with the given name to the table.

Parameters

fname	The name of the function to add
hasReturn	Indicates if the function has a return value

5.82.2.3 void FunctionTable::AddFunction (string *fname*, bool *hasReturn* = `false`)

Adds a function with the given name to the table.

Parameters

fname	The name of the function to add
hasReturn	Indicates if the function has a return value

5.82.2.4 void FunctionTable::AddParameter (string *fname*, string *pname*, bool *byValue*)

Adds a parameter to the given function.

Parameters

fname	The function name
pname	The parameter name
byValue	Indicates if the parameter is passed by value or reference

Definition at line 54 of file FunctionTable.cpp.

5.82.2.5 void FunctionTable::AddParameter (string *fname*, string *pname*, bool *byValue*)

Adds a parameter to the given function.

Parameters

fname	The function name
pname	The parameter name
byValue	Indicates if the parameter is passed by value or reference

5.82.2.6 void FunctionTable::AddParameter (string *fname*, string *pname*, bool *byValue*)

Adds a parameter to the given function.

Parameters

fname	The function name
pname	The parameter name
byValue	Indicates if the parameter is passed by value or reference

5.82.2.7 void FunctionTable::AddVariable (string *fname*, string *vname*, int *size*)

Adds a variable to the list of variables for the given function and calculates its offset.

Parameters

fname	The function name
vname	The variable name

size	The size of the variable in bytes

Definition at line 43 of file FunctionTable.cpp.

5.82.2.8 void FunctionTable::AddVariable (string *fname*, string *vname*, int *size*)

Adds a variable to the list of variables for the given function and calculates its offset.

Parameters

fname	The function name
vname	The variable name
size	The size of the variable in bytes

5.82.2.9 void FunctionTable::AddVariable (string *fname*, string *vname*, int *size*)

Adds a variable to the list of variables for the given function and calculates its offset.

Parameters

fname	The function name
vname	The variable name
size	The size of the variable in bytes

5.82.2.10 Function ∗ FunctionTable::GetFunction (string *name*)

Gets a reference to a function with the given name.

Parameters

name	The name of the function

Returns

A reference to the function

Definition at line 28 of file FunctionTable.cpp.

5.82.2.11 Function∗ FunctionTable::GetFunction (string *name*)

Gets a reference to a function with the given name.

Parameters

name	The name of the function

Returns

A reference to the function

5.82.2.12 Function∗ FunctionTable::GetFunction (string *name*)

Gets a reference to a function with the given name.

Parameters

name	The name of the function

Returns

A reference to the function

5.82.2.13 int FunctionTable::GetStackSpace (string *name*)

Returns the stack sace required for a function with the given name.

Returns -1 if there is no function with the given name.

Parameters

name	The name of the function to lookup

Returns

The size of the stack required for the function.

Definition at line 3 of file FunctionTable.cpp.

5.82.2.14 int FunctionTable::GetStackSpace (string *name*)

Returns the stack sace required for a function with the given name.

Returns -1 if there is no function with the given name.

Parameters

name	The name of the function to lookup

Returns

The size of the stack required for the function.

5.82.2.15 int FunctionTable::GetStackSpace (string *name*)

Returns the stack sace required for a function with the given name.

Returns -1 if there is no function with the given name.

Parameters

name	The name of the function to lookup

Returns

The size of the stack required for the function.

5.82.2.16 int FunctionTable::GetVarOffset (string *fname*, string *vname*)

Gets the offset of a given variable within a given function.

Parameters

fname	The function name
vname	The variable name

Returns

The variable's offset

Definition at line 69 of file FunctionTable.cpp.

5.82.2.17 int FunctionTable::GetVarOffset (string *fname*, string *vname*)

Gets the offset of a given variable within a given function.

Parameters

fname	The function name
vname	The variable name

Returns

The variable's offset

5.82.2.18 int FunctionTable::GetVarOffset (string *fname*, string *vname*)

Gets the offset of a given variable within a given function.

Parameters

fname	The function name
vname	The variable name

Returns

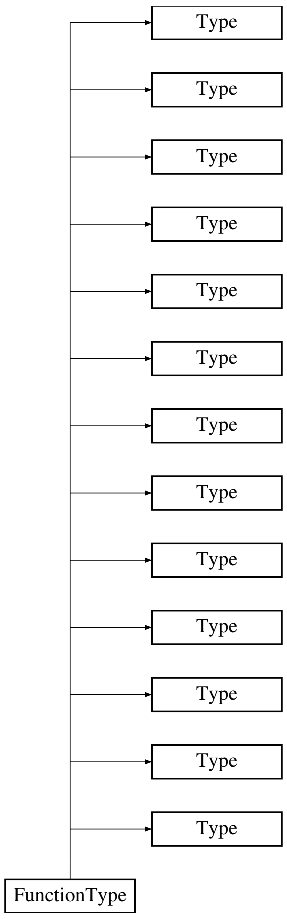
The variable's offset

The documentation for this class was generated from the following files:

- mips/FunctionTable.h
- mips/FunctionTable.cpp

5.83 FunctionType Class Reference

Inheritance diagram for FunctionType:


Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
FunctionType

Public Types

- enum **DerivedType** {
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()

- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **FunctionType** (string n)
- void **AddParam** (Type ∗t)
- void **SetReturnType** (Type ∗t)
- int **GetParamCount** ()
- Type ∗ **GetReturnType** ()
- bool **CheckType** (FunctionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- vector< Type * > **params**
- Type * **returnType**
- string **name**
- int **size**

5.83.1 Detailed Description

Definition at line 170 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.84 InputLine Struct Reference

A structure for buffering lines input code.

```
#include <CCompiler.h>
```

Public Member Functions

- InputLine (int l, string s)

  Defualt constructor.
- InputLine (int l, string s)

  Defualt constructor.
- InputLine (int l, string s)

  Defualt constructor.

Public Attributes

- int line

  The line number.
- string text

  The line of code.

5.84.1 Detailed Description

A structure for buffering lines input code.

Definition at line 29 of file CCompiler.h.

The documentation for this struct was generated from the following file:

- CCompiler.h

5.85 AVLTree< DataItem >::Node Struct Reference

A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.

Public Attributes

- DataItem **data**
- Node ∗ **children** [CHILD_SIZE]
- int **balanceFactor**

5.85.1 Detailed Description

template<class DataItem>struct AVLTree< DataItem >::Node

A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.

Definition at line 41 of file AvlTree.h.

The documentation for this struct was generated from the following files:

- AvlTree.h
- CParser.yy
- CScanner.ll

5.86 Parameter Struct Reference

Public Attributes

- string **name**
- bool **byValue**
- int **size**
- bool **byRegister**
- string **regName**
- int **offset**

5.86.1 Detailed Description

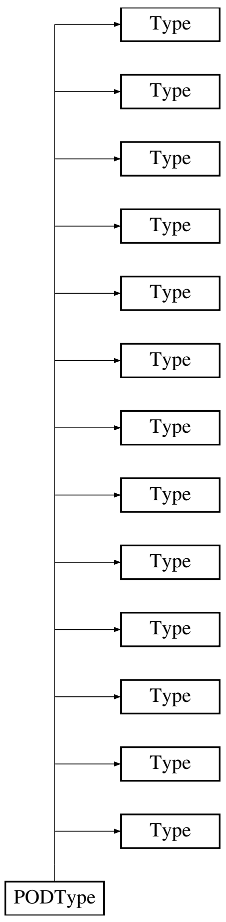
Definition at line 11 of file FunctionTable.h.

The documentation for this struct was generated from the following file:

- mips/FunctionTable.h

5.87 PODType Class Reference

Inheritance diagram for PODType:


Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
PODType

Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)

- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PODType** (string n, int s)
- bool **isSigned** ()
- void **SetSigned** (bool isSigned)
- bool **CheckType** (PODType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()

- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- bool **is_signed**
- string **name**
- int **size**

5.87.1 Detailed Description

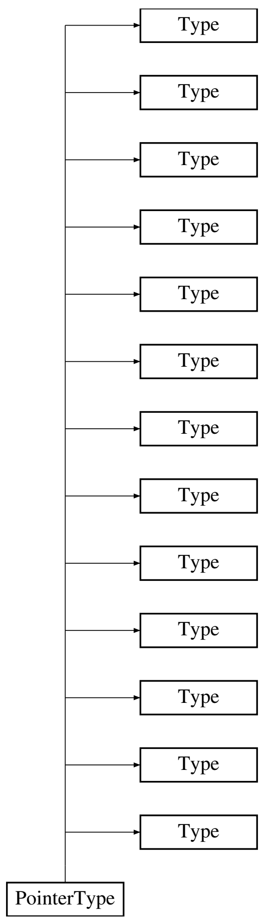
Definition at line 89 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.88 PointerType Class Reference

Inheritance diagram for PointerType:



Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()

- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **PointerType** (Type ∗base, string n, int d)
- **PointerType** (Type ∗base, bool baseIsPtr, string n)
- Type ∗ **GetBase** ()
- void **SetBaseType** (Type ∗base)
- bool **CheckType** (PointerType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()

- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- Type ∗ **baseType**
- int **ptrDepth**
- string **name**
- int **size**

5.88.1 Detailed Description

Definition at line 185 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.89 RegAllocTable Class Reference

Public Member Functions

- **RegAllocTable** (int maxRegisters, string regPrefix)
- void PrintRegisters ()

    This function is for the purpose of debugging the register allocation algorithms.
- std::string GetRegister (std::string name, bool &isNew)

    Handles the allocation of registers including spilling when required.
- void FreeRegister (std::string name)

    Handles the deallocation of a register assigned to a temporary.
- int GetSpillSize ()

    Returns the size of the global .space directive that needs to be allocated for the spill registers.
- string Lookup (string name)

    This function lookups up the register location of the given name.
- string LookupOwner (string reg)

    This function lookups up the owner a of real register.
- void SetFstream (fstream *fs)

    Used to set the pointer to the output stream which must be created by the owning class (i.e.
- void SetVerbose (bool flag)

    Turns verbose comments on or off.
- **RegAllocTable** (int maxRegisters, string regPrefix)
- void PrintRegisters ()

    This function is for the purpose of debugging the register allocation algorithms.
- std::string GetRegister (std::string name, bool &isNew)

    Handles the allocation of registers including spilling when required.
- void FreeRegister (std::string name)

    Handles the deallocation of a register assigned to a temporary.
- int GetSpillSize ()

    Returns the size of the global .space directive that needs to be allocated for the spill registers.
- string Lookup (string name)

    This function lookups up the register location of the given name.
- string LookupOwner (string reg)

    This function lookups up the owner a of real register.
- void SetFstream (fstream *fs)

    Used to set the pointer to the output stream which must be created by the owning class (i.e.
- void SetVerbose (bool flag)

    Turns verbose comments on or off.
- **RegAllocTable** (int maxRegisters, string regPrefix)
- void PrintRegisters ()

    This function is for the purpose of debugging the register allocation algorithms.
- std::string GetRegister (std::string name, bool &isNew)

Handles the allocation of registers including spilling when required.

- void FreeRegister (std::string name)

  Handles the deallocation of a register assigned to a temporary.

- int GetSpillSize ()

  Returns the size of the global .space directive that needs to be allocated for the spill registers.

- string Lookup (string name)

  This function lookups up the register location of the given name.

- string LookupOwner (string reg)

  This function lookups up the owner a of real register.

- void SetFstream (fstream ∗fs)

  Used to set the pointer to the output stream which must be created by the owning class (i.e.

- void SetVerbose (bool flag)

  Turns verbose comments on or off.

- **RegAllocTable** (int maxRegisters, string regPrefix)
- void PrintRegisters ()

  This function is for the purpose of debugging the register allocation algorithms.

- std::string GetRegister (std::string name, bool &isNew)

  Handles the allocation of registers including spilling when required.

- void FreeRegister (std::string name)

  Handles the deallocation of a register assigned to a temporary.

- int GetSpillSize ()

  Returns the size of the global .space directive that needs to be allocated for the spill registers.

- string Lookup (string name)

  This function lookups up the register location of the given name.

- string LookupOwner (string reg)

  This function lookups up the owner a of real register.

- void SetFstream (fstream ∗fs)

  Used to set the pointer to the output stream which must be created by the owning class (i.e.

- void SetVerbose (bool flag)

  Turns verbose comments on or off.

- **RegAllocTable** (int maxRegisters, string regPrefix)
- void PrintRegisters ()

  This function is for the purpose of debugging the register allocation algorithms.

- std::string GetRegister (std::string name, bool &isNew)

  Handles the allocation of registers including spilling when required.

- void FreeRegister (std::string name)

  Handles the deallocation of a register assigned to a temporary.

- int GetSpillSize ()

  Returns the size of the global .space directive that needs to be allocated for the spill registers.

- string Lookup (string name)

  This function lookups up the register location of the given name.

- string LookupOwner (string reg)

  This function lookups up the owner a of real register.

- void SetFstream (fstream ∗fs)

  Used to set the pointer to the output stream which must be created by the owning class (i.e.

- void SetVerbose (bool flag)

  Turns verbose comments on or off.

Public Attributes

- Event< RegAllocTable, void ∗ > UpdatedEvent

  This class implements the observable pattern so that the Address Table can update the register locations it stores whenever possible changes to the register locations have occured.

Private Member Functions

- int GetSpillIndex ()

  Finds the index of the next open spill register.
- int GetOpenRegIndex (bool &found)

  Finds the index of the next open register.
- void incrementLifes ()

  Increments the lifespan of all registers (real and spill) which are owned.
- int GetHighestLifeIndex ()

  Gets the index of the register with the highest lifespan.
- void RegToSpill (int rindex, int sindex)

  Outputs the MIPS code to move the contents of a real register to a spill register.
- void SpillToReg (int sindex, int rindex)

  Outputs the MIPS code to move the contents of a spill register into a real register.
- void SwapRegToSpill (int rindex, int sindex)

  Outputs the MIPS code to swap a spill register with a real register.
- int GetSpillIndex ()

  Finds the index of the next open spill register.
- int GetOpenRegIndex (bool &found)

  Finds the index of the next open register.
- void incrementLifes ()

  Increments the lifespan of all registers (real and spill) which are owned.
- int GetHighestLifeIndex ()

  Gets the index of the register with the highest lifespan.
- void RegToSpill (int rindex, int sindex)

  Outputs the MIPS code to move the contents of a real register to a spill register.
- void SpillToReg (int sindex, int rindex)

  Outputs the MIPS code to move the contents of a spill register into a real register.
- void SwapRegToSpill (int rindex, int sindex)

  Outputs the MIPS code to swap a spill register with a real register.
- int GetSpillIndex ()

  Finds the index of the next open spill register.
- int GetOpenRegIndex (bool &found)

  Finds the index of the next open register.
- void incrementLifes ()

  Increments the lifespan of all registers (real and spill) which are owned.
- int GetHighestLifeIndex ()

  Gets the index of the register with the highest lifespan.
- void RegToSpill (int rindex, int sindex)

  Outputs the MIPS code to move the contents of a real register to a spill register.
- void SpillToReg (int sindex, int rindex)

  Outputs the MIPS code to move the contents of a spill register into a real register.
- void SwapRegToSpill (int rindex, int sindex)

  Outputs the MIPS code to swap a spill register with a real register.
- int GetSpillIndex ()

Finds the index of the next open spill register.

- int GetOpenRegIndex (bool &found)

  Finds the index of the next open register.
- void incrementLifes ()

  Increments the lifespan of all registers (real and spill) which are owned.
- int GetHighestLifeIndex ()

  Gets the index of the register with the highest lifespan.
- void RegToSpill (int rindex, int sindex)

  Outputs the MIPS code to move the contents of a real register to a spill register.
- void SpillToReg (int sindex, int rindex)

  Outputs the MIPS code to move the contents of a spill register into a real register.
- void SwapRegToSpill (int rindex, int sindex)

  Outputs the MIPS code to swap a spill register with a real register.
- int GetSpillIndex ()

  Finds the index of the next open spill register.
- int GetOpenRegIndex (bool &found)

  Finds the index of the next open register.
- void incrementLifes ()

  Increments the lifespan of all registers (real and spill) which are owned.
- int GetHighestLifeIndex ()

  Gets the index of the register with the highest lifespan.
- void RegToSpill (int rindex, int sindex)

  Outputs the MIPS code to move the contents of a real register to a spill register.
- void SpillToReg (int sindex, int rindex)

  Outputs the MIPS code to move the contents of a spill register into a real register.
- void SwapRegToSpill (int rindex, int sindex)

  Outputs the MIPS code to swap a spill register with a real register.

Private Attributes

- int **size**
- string prefix

  Indicates the number of available registers that can be used.
- int width

  Indicates the column width in the output MIPS.
- int numSpills

  Indicates the number of spill registers allocated by default in the data section.
- fstream ∗ fout

  A pointer to the MIPS file.
- Register ∗ registers

  Collection of real registers.
- Register ∗ spills

  Collection of spill registers declared in the global data block.
- bool verbose

  Indicates if verbose comments should be output in the MIPS.

5.89.1 Detailed Description

Definition at line 60 of file RegAllocTable.h.

5.89.2 Member Function Documentation

5.89.2.1 void RegAllocTable::FreeRegister (std::string *name*)

Handles the deallocation of a register assigned to a temporary.

Note: this function DOES NOT save anything to memory! This function only marks a register as available, allowing that register to be allocated again in the future.

Parameters

name	The name of the owner of a register which must be freed

Definition at line 116 of file RegAllocTable.cpp.

5.89.2.2 void RegAllocTable::FreeRegister (std::string *name*)

Handles the deallocation of a register assigned to a temporary.

Note: this function DOES NOT save anything to memory! This function only marks a register as available, allowing that register to be allocated again in the future.

Parameters

name	The name of the owner of a register which must be freed

5.89.2.3 void RegAllocTable::FreeRegister (std::string *name*)

Handles the deallocation of a register assigned to a temporary.

Note: this function DOES NOT save anything to memory! This function only marks a register as available, allowing that register to be allocated again in the future.

Parameters

name	The name of the owner of a register which must be freed

5.89.2.4 void RegAllocTable::FreeRegister (std::string *name*)

Handles the deallocation of a register assigned to a temporary.

Note: this function DOES NOT save anything to memory! This function only marks a register as available, allowing that register to be allocated again in the future.

Parameters

name	The name of the owner of a register which must be freed

5.89.2.5 void RegAllocTable::FreeRegister (std::string *name*)

Handles the deallocation of a register assigned to a temporary.

Note: this function DOES NOT save anything to memory! This function only marks a register as available, allowing that register to be allocated again in the future.

Parameters

name	The name of the owner of a register which must be freed

5.89.2.6 int RegAllocTable::GetHighestLifeIndex ( ) [private]

Gets the index of the register with the highest lifespan.

Returns

The index of the register with the highest lifespan.

5.89.2.7 int RegAllocTable::GetHighestLifeIndex ( ) [private]

Gets the index of the register with the highest lifespan.

Returns

The index of the register with the highest lifespan.

5.89.2.8 int RegAllocTable::GetHighestLifeIndex ( ) [private]

Gets the index of the register with the highest lifespan.

Returns

The index of the register with the highest lifespan.

Definition at line 262 of file RegAllocTable.cpp.

5.89.2.9 int RegAllocTable::GetHighestLifeIndex ( ) [private]

Gets the index of the register with the highest lifespan.

Returns

The index of the register with the highest lifespan.

5.89.2.10 int RegAllocTable::GetHighestLifeIndex ( ) [private]

Gets the index of the register with the highest lifespan.

Returns

The index of the register with the highest lifespan.

5.89.2.11 int RegAllocTable::GetOpenRegIndex (bool & *found*) [private]

Finds the index of the next open register.

Parameters

found	Indicates if an open register was found.

Returns

The index of the open register. Invalid if found == false

Definition at line 246 of file RegAllocTable.cpp.

5.89.2.12 int RegAllocTable::GetOpenRegIndex (bool & *found*) [private]

Finds the index of the next open register.

Parameters

found	Indicates if an open register was found.

Returns

The index of the open register. Invalid if found == false

5.89.2.13 int RegAllocTable::GetOpenRegIndex (bool & *found*) [private]

Finds the index of the next open register.

Parameters

found	Indicates if an open register was found.

Returns

The index of the open register. Invalid if found == false

5.89.2.14 int RegAllocTable::GetOpenRegIndex (bool & *found*) [private]

Finds the index of the next open register.

Parameters

found	Indicates if an open register was found.

Returns

The index of the open register. Invalid if found == false

5.89.2.15 int RegAllocTable::GetOpenRegIndex (bool & *found*) [private]

Finds the index of the next open register.

Parameters

found	Indicates if an open register was found.

Returns

The index of the open register. Invalid if found == false

5.89.2.16 std::string RegAllocTable::GetRegister (std::string *name*, bool & *isNew*)

Handles the allocation of registers including spilling when required.

This function first checks if the name is already in a register. If the name is already assigned to a register, then isNew is set to false and the name of the register is returned. If the name has not already been assigned to a register, then the function attempts to find an open register. If one is present, then the name is assigned to it, isNew is set to true, and the name of the register is returned. Otherwise, spilling must occur. This function uses a Last Recently Allocated algorithm to decide which register should be spilled to memory and allocated to the new name.

Parameters

name	The name of the temporary register in the 3AC which needs to be mapped to a real register.
isNew	An output ref parameter indicating if the temp register was already mapped to this real register, or if the mapping is new.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.89.2.17 std::string RegAllocTable::GetRegister (std::string *name*, bool & *isNew*)

Handles the allocation of registers including spilling when required.

This function first checks if the name is already in a register. If the name is already assigned to a register, then isNew is set to false and the name of the register is returned. If the name has not already been assigned to a register, then the function attempts to find an open register. If one is present, then the name is assigned to it, isNew is set to true, and the name of the register is returned. Otherwise, spilling must occur. This function uses a Last Recently Allocated algorithm to decide which register should be spilled to memory and allocated to the new name.

Parameters

name	The name of the temporary register in the 3AC which needs to be mapped to a real register.
isNew	An output ref parameter indicating if the temp register was already mapped to this real register, or if the mapping is new.

Returns

The name of the real register to which the temp register has been mapped/assigned.

Definition at line 61 of file RegAllocTable.cpp.

5.89.2.18 std::string RegAllocTable::GetRegister (std::string *name*, bool & *isNew*)

Handles the allocation of registers including spilling when required.

This function first checks if the name is already in a register. If the name is already assigned to a register, then isNew is set to false and the name of the register is returned. If the name has not already been assigned to a register, then the function attempts to find an open register. If one is present, then the name is assigned to it, isNew is set to true, and the name of the register is returned. Otherwise, spilling must occur. This function uses a Last Recently Allocated algorithm to decide which register should be spilled to memory and allocated to the new name.

Parameters

name	The name of the temporary register in the 3AC which needs to be mapped to a real register.
isNew	An output ref parameter indicating if the temp register was already mapped to this real register, or if the mapping is new.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.89.2.19 std::string RegAllocTable::GetRegister (std::string *name*, bool & *isNew*)

Handles the allocation of registers including spilling when required.

This function first checks if the name is already in a register. If the name is already assigned to a register, then isNew is set to false and the name of the register is returned. If the name has not already been assigned to a register, then the function attempts to find an open register. If one is present, then the name is assigned to it, isNew is set to true, and the name of the register is returned. Otherwise, spilling must occur. This function uses a Last Recently Allocated algorithm to decide which register should be spilled to memory and allocated to the new name.

Parameters

name	The name of the temporary register in the 3AC which needs to be mapped to a real register.
isNew	An output ref parameter indicating if the temp register was already mapped to this real register, or if the mapping is new.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.89.2.20 std::string RegAllocTable::GetRegister (std::string *name*, bool & *isNew*)

Handles the allocation of registers including spilling when required.

This function first checks if the name is already in a register. If the name is already assigned to a register, then isNew is set to false and the name of the register is returned. If the name has not already been assigned to a register, then the function attempts to find an open register. If one is present, then the name is assigned to it, isNew is set to true, and the name of the register is returned. Otherwise, spilling must occur. This function uses a Last Recently Allocated algorithm to decide which register should be spilled to memory and allocated to the new name.

Parameters

name	The name of the temporary register in the 3AC which needs to be mapped to a real register.
isNew	An output ref parameter indicating if the temp register was already mapped to this real register, or if the mapping is new.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.89.2.21 string RegAllocTable::Lookup (string *name*)

This function lookups up the register location of the given name.

Parameters

name	The name of a variable or temp register to lookup

Returns

The register which is owned by the given name, or an empty string if that name doesn't own a register

5.89.2.22 string RegAllocTable::Lookup (string *name*)

This function lookups up the register location of the given name.

Parameters

name	The name of a variable or temp register to lookup

Returns

The register which is owned by the given name, or an empty string if that name doesn't own a register

Definition at line 201 of file RegAllocTable.cpp.

5.89.2.23 string RegAllocTable::Lookup (string *name*)

This function lookups up the register location of the given name.

Parameters

name	The name of a variable or temp register to lookup

Returns

The register which is owned by the given name, or an empty string if that name doesn't own a register

5.89.2.24 string RegAllocTable::Lookup (string *name*)

This function lookups up the register location of the given name.

Parameters

name	The name of a variable or temp register to lookup

Returns

The register which is owned by the given name, or an empty string if that name doesn't own a register

5.89.2.25 string RegAllocTable::Lookup (string *name*)

This function lookups up the register location of the given name.

Parameters

name	The name of a variable or temp register to lookup

Returns

The register which is owned by the given name, or an empty string if that name doesn't own a register

5.89.2.26 string RegAllocTable::LookupOwner (string *reg*)

This function lookups up the owner a of real register.

Parameters

reg	The name of the register to lookup

Returns

The owner of the register, or an empty string if that register doesn't have an owner

5.89.2.27 string RegAllocTable::LookupOwner (string *reg*)

This function lookups up the owner a of real register.

Parameters

reg	The name of the register to lookup

Returns

The owner of the register, or an empty string if that register doesn't have an owner

5.89.2.28 string RegAllocTable::LookupOwner (string *reg*)

This function lookups up the owner a of real register.

Parameters

reg	The name of the register to lookup

Returns

The owner of the register, or an empty string if that register doesn't have an owner

Definition at line 218 of file RegAllocTable.cpp.

5.89.2.29 string RegAllocTable::LookupOwner (string *reg*)

This function lookups up the owner a of real register.

Parameters

reg	The name of the register to lookup

Returns

The owner of the register, or an empty string if that register doesn't have an owner

5.89.2.30 string RegAllocTable::LookupOwner (string *reg*)

This function lookups up the owner a of real register.

Parameters

reg	The name of the register to lookup

Returns

The owner of the register, or an empty string if that register doesn't have an owner

5.89.2.31 void RegAllocTable::PrintRegisters ( )

This function is for the purpose of debugging the register allocation algorithms.

It simply prints out a listing of each of the owned registers.

Definition at line 37 of file RegAllocTable.cpp.

5.89.2.32 void RegAllocTable::PrintRegisters ( )

This function is for the purpose of debugging the register allocation algorithms.

It simply prints out a listing of each of the owned registers.

5.89.2.33 void RegAllocTable::PrintRegisters ( )

This function is for the purpose of debugging the register allocation algorithms.

It simply prints out a listing of each of the owned registers.

5.89.2.34 void RegAllocTable::PrintRegisters ( )

This function is for the purpose of debugging the register allocation algorithms.

It simply prints out a listing of each of the owned registers.

5.89.2.35 void RegAllocTable::PrintRegisters ( )

This function is for the purpose of debugging the register allocation algorithms.

It simply prints out a listing of each of the owned registers.

5.89.2.36 void RegAllocTable::SetFstream (fstream ∗ *fs*)

Used to set the pointer to the output stream which must be created by the owning class (i.e. tac2mips).

5.89.2.37 void RegAllocTable::SetFstream (fstream ∗ *fs*)

Used to set the pointer to the output stream which must be created by the owning class (i.e. tac2mips).

5.89.2.38 void RegAllocTable::SetFstream (fstream ∗ *fs*)

Used to set the pointer to the output stream which must be created by the owning class (i.e. tac2mips).

Definition at line 227 of file RegAllocTable.cpp.

5.89.2.39 void RegAllocTable::SetFstream (fstream ∗ *fs*)

Used to set the pointer to the output stream which must be created by the owning class (i.e. tac2mips).

5.89.2.40 void RegAllocTable::SetFstream (fstream ∗ *fs*)

Used to set the pointer to the output stream which must be created by the owning class (i.e. tac2mips).

5.89.3 Member Data Documentation

5.89.3.1 string RegAllocTable::prefix [private]

Indicates the number of available registers that can be used.

Indicates how registers are named (i.e. $t for MIPS)

Definition at line 149 of file RegAllocTable.h.

The documentation for this class was generated from the following files:

- mips/RegAllocTable.h
- mips/TAC_Parser.yy
- mips/TAC_Scanner.ll
- mips/RegAllocTable.cpp

5.90 Register Struct Reference

Public Member Functions

- **Register** (const Register &rhs)
- **Register** (const Register &rhs)
- **Register** (const Register &rhs)
- **Register** (const Register &rhs)
- **Register** (const Register &rhs)

Public Attributes

- int lifespan

  Indicates how long the register has been owned for.
- bool isOwned

  Indicates if the register is owned.
- string owner

  Name of the temporary or variable which is loaded in the register.
- string name

  Name used by the assembly language to reference the register.
- int spillOffset

  Indicates the offset of this register into the spill register memory segment.

5.90.1 Detailed Description

Definition at line 17 of file RegAllocTable.h.

5.90.2 Member Data Documentation

5.90.2.1 string Register::name

Name used by the assembly language to reference the register.

(i.e. "$t0")

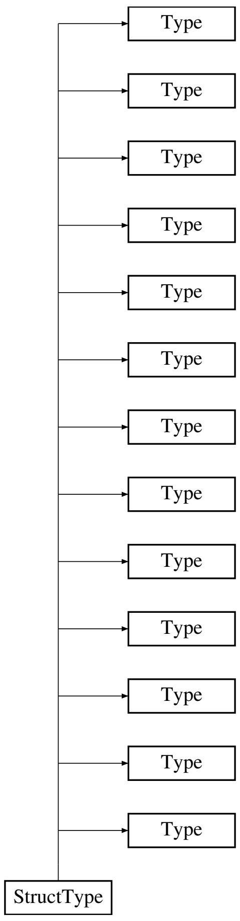
Definition at line 36 of file RegAllocTable.h.

The documentation for this struct was generated from the following files:

- mips/RegAllocTable.h
- mips/TAC_Parser.yy
- mips/TAC_Scanner.ll

5.91 StructType Class Reference

Inheritance diagram for StructType:



Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)

- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **StructType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (StructType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()

- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- vector< string > **memberNames**
- vector< Type ∗ > **memberTypes**
- string **name**
- int **size**

5.91.1 Detailed Description

Definition at line 144 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.92 GoFPatterns::Event< SourceType, EventArgType >::SubscriberRecord Class Reference

This inner class, for each EventHandler, stores the associated context information - pointer.

`#include <Event.h>`

Private Member Functions

- **SubscriberRecord** (EventHandler handler, void ∗context=0)
- **SubscriberRecord** (EventHandler handler, void ∗context=0)
- **SubscriberRecord** (EventHandler handler, void ∗context=0)

Private Attributes

- EventHandler **handlerProc**
- void ∗ **handlerContext**

Friends

- class **Event**

5.92.1 Detailed Description

template<typename SourceType, typename EventArgType>class GoFPatterns::Event< SourceType, EventArgType >::-SubscriberRecord

This inner class, for each EventHandler, stores the associated context information - pointer.

This context pointer can be used to pass additional informations from the point of subscription to the event handler function. The best use of context pointer is to use the "this" pointer of subscriber itself.

Definition at line 44 of file Event.h.

The documentation for this class was generated from the following file:

- mips/Event.h

5.93 SymbolInfo Struct Reference

Public Member Functions

- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)

- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()
- **SymbolInfo** (const SymbolInfo &sym)
- int **operator<** (SymbolInfo inf)
- int **operator==** (SymbolInfo inf)
- int **operator>** (SymbolInfo inf)
- string **GetKey** ()

Public Attributes

- string **symbol_name**
- Type ∗ **symbolType**
- int **type_qualifier**
- string **function_name**
- int **offset**
- bool **isEnumConst**
- bool **struct_union_name**
- bool **isStrunctOrUnionItem**
- int **typeTableIndex**
- int **storage_class**
- int **flags**

Friends

- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)
- ostream & **operator**$<<$ (ostream &outStream, const SymbolInfo &inf)

5.93.1 Detailed Description

Definition at line 22 of file CParser.yy.

The documentation for this struct was generated from the following file:

- SymTab.h

5.94 SymTab Class Reference

Public Member Functions

- **SymTab** (CCompiler $*$ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo $*$ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list$<$ SymbolInfo $>$ **GetGlobals** ()
- list$<$ SymbolInfo $>$ **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler $*$ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo $*$ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list$<$ SymbolInfo $>$ **GetGlobals** ()
- list$<$ SymbolInfo $>$ **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)

- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)

- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)
- **SymTab** (CCompiler ∗ref)
- void **EnterScope** ()
- void **LeaveScope** ()
- void **map_function_vars** (const string &funcName)
- void **insert_symbol** (SymbolInfo symbolInfo)
- void **insert_symbol** (SymbolInfo symbolInfo, int level)
- bool **find_symbol** (SymbolInfo symbolInfo, int &level)
- SymbolInfo ∗ **fetch_symbol** (SymbolInfo symbolInfo, int level)
- void **dump_table** ()
- void **dump_table** (int level)
- list< SymbolInfo > **GetGlobals** ()
- list< SymbolInfo > **GetLocals** (string &funcName)
- bool **IsGlobal** (string &symName)
- int **GetFuncOffset** (string &funcName)

Private Member Functions

- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)
- void **error** (string msg)
- void **warning** (string msg)

Private Attributes

- int **currentLevel**
- vector< AVLTree< SymbolInfo > > **symTable**
- map< int, int > **offsetMap**
- CCompiler ∗ **driver**
- list< string > **funcNames**
- map< string, AVLTree
  < SymbolInfo > > **funcSymMap**
- map< string, int > **funOffMap**

5.94.1 Detailed Description

Definition at line 82 of file CParser.yy.

The documentation for this class was generated from the following files:

- SymTab.h
- SymTab.cpp

5.95 tac2mips Class Reference

Public Member Functions

- tac2mips (std::string filename)

  Standard constructor.
- ∼tac2mips ()

  Standard destructor.
- void Run ()

  Runs the conversion from 3AC to MIPS.

- void scan_begin ()

  Initializes the scanning process.
- void scan_end ()

  Closes the different output streams used in the scanner.
- void error (const std::string &msg)

  Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.
- std::string GetRegister (std::string tempName)

  Handles the allocation of registers including spilling when required.
- void FreeRegister (std::string name)

  Frees a register after when it is no longer live.
- bool LabelExists (std::string name)

  Checks if a label exists.
- void AddLabel (std::string name)

  Adds a label to the collection of labels.
- void OutputPreamble ()

  Outputs the beginning of the MIPS file.
- void Comment (std::string txt, bool VerboseOnly)

  Outputs a comment into the MIPS file.
- void WS (int lines=1)

  Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.
- void toMIPS (std::string psuedoop)

  Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1)

  Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1, std::string op2)

  Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1, std::string op2, std::string op3)

  Adds a line of code to the MIPS file and formats it correctly.
- void Macro (std::string name, std::string params)

  Outputs a macro with the given name and parameters.
- void Macro (std::string name)

  Outputs a macro with the given name and no parameters.
- void Label (std::string txt)

  Adds a label to the MIPS file and formats it correctly.
- void SetVerbose (bool flag)

  Turns verbose comments on or off.
- tac2mips (std::string filename)

  Standard constructor.
- ∼tac2mips ()

  Standard destructor.
- void Run ()

  Runs the conversion from 3AC to MIPS.
- void scan_begin ()

  Initializes the scanning process.
- void scan_end ()

  Closes the different output streams used in the scanner.
- void error (const std::string &msg)

  Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.
- std::string GetRegister (std::string tempName)

Handles the allocation of registers including spilling when required.
- void FreeRegister (std::string name)

    Frees a register after when it is no longer live.
- bool LabelExists (std::string name)

    Checks if a label exists.
- void AddLabel (std::string name)

    Adds a label to the collection of labels.
- void OutputPreamble ()

    Outputs the beginning of the MIPS file.
- void Comment (std::string txt, bool VerboseOnly)

    Outputs a comment into the MIPS file.
- void WS (int lines=1)

    Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.
- void toMIPS (std::string psuedoop)

    Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1)

    Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1, std::string op2)

    Adds a line of code to the MIPS file and formats it correctly.
- void toMIPS (std::string opcode, std::string op1, std::string op2, std::string op3)

    Adds a line of code to the MIPS file and formats it correctly.
- void Macro (std::string name, std::string params)

    Outputs a macro with the given name and parameters.
- void Macro (std::string name)

    Outputs a macro with the given name and no parameters.
- void Label (std::string txt)

    Adds a label to the MIPS file and formats it correctly.
- void SetVerbose (bool flag)

    Turns verbose comments on or off.
- tac2mips (std::string filename)

    Standard constructor.
- ∼tac2mips ()

    Standard destructor.
- void Run ()

    Runs the conversion from 3AC to MIPS.
- void scan_begin ()

    Initializes the scanning process.
- void scan_end ()

    Closes the different output streams used in the scanner.
- void error (const std::string &msg)

    Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.
- std::string GetRegister (std::string tempName)

    Handles the allocation of registers including spilling when required.
- void FreeRegister (std::string name)

    Frees a register after when it is no longer live.
- bool LabelExists (std::string name)

    Checks if a label exists.
- void AddLabel (std::string name)

    Adds a label to the collection of labels.
- void OutputPreamble ()

Outputs the beginning of the MIPS file.

- void Comment (std::string txt, bool VerboseOnly)

  Outputs a comment into the MIPS file.

- void WS (int lines=1)

  Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.

- void toMIPS (std::string psuedoop)

  Adds a line of code to the MIPS file and formats it correctly.

- void toMIPS (std::string opcode, std::string op1)

  Adds a line of code to the MIPS file and formats it correctly.

- void toMIPS (std::string opcode, std::string op1, std::string op2)

  Adds a line of code to the MIPS file and formats it correctly.

- void toMIPS (std::string opcode, std::string op1, std::string op2, std::string op3)

  Adds a line of code to the MIPS file and formats it correctly.

- void Macro (std::string name, std::string params)

  Outputs a macro with the given name and parameters.

- void Macro (std::string name)

  Outputs a macro with the given name and no parameters.

- void Label (std::string txt)

  Adds a label to the MIPS file and formats it correctly.

- void SetVerbose (bool flag)

  Turns verbose comments on or off.

Public Attributes

- std::string fname

  Standard assembly code output stream filename.

- RegAllocTable regtab

  The register allocation table.

- AddressTable addtab

  The address table.

- FunctionTable funtab

  A table for tracking functions and the associated information for each function such as the required stack space for the variables in the function and the parameters to the function.

Private Attributes

- std::string outfname

  The name of the output MIPS file.

- std::fstream fout

  The MIPS output file stream.

- set< string > labels

  A collection of all of the labels in the MIPS output.

- bool verbose

  Indicates if verbose comments should be output in the MIPS.

5.95.1 Detailed Description

Examples:

test1.c, test2.c, test3.c, test4.c, test5.c, test6.c, and test7.c.

Definition at line 26 of file tac2mips.h.

5.95.2 Constructor & Destructor Documentation

5.95.2.1 tac2mips::tac2mips (std::string *filename*)

Standard constructor.

Parameters

filename	The name of the input 3AC file

5.95.2.2 tac2mips::tac2mips (std::string *filename*)

Standard constructor.

Parameters

filename	The name of the input 3AC file

5.95.2.3 tac2mips::tac2mips (std::string *filename*)

Standard constructor.

Parameters

filename	The name of the input 3AC file

5.95.3 Member Function Documentation

5.95.3.1 void tac2mips::Comment (std::string *txt,* bool *VerboseOnly*)

Outputs a comment into the MIPS file.

Parameters

txt	The comment text to output.
VerboseOnly	Indicates if this comment should only be printed if the verbose flag is true

Definition at line 101 of file tac2mips.cpp.

5.95.3.2 void tac2mips::Comment (std::string *txt,* bool *VerboseOnly*)

Outputs a comment into the MIPS file.

Parameters

txt	The comment text to output.
VerboseOnly	Indicates if this comment should only be printed if the verbose flag is true

5.95.3.3 void tac2mips::Comment (std::string *txt,* bool *VerboseOnly*)

Outputs a comment into the MIPS file.

Parameters

txt	The comment text to output.
VerboseOnly	Indicates if this comment should only be printed if the verbose flag is true

5.95.3.4 void tac2mips::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

5.95.3.5 void tac2mips::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

Definition at line 39 of file tac2mips.cpp.

5.95.3.6 void tac2mips::error (const std::string & *msg*)

Prints an error to the standard compiler output stream and terminates the program with with an EXIT_FAILURE return value.

Parameters

msg	The error message

5.95.3.7 void tac2mips::FreeRegister (std::string *name*)

Frees a register after when it is no longer live.

Parameters

name	The name of the owner of a register which needs to be freed

5.95.3.8 void tac2mips::FreeRegister (std::string *name*)

Frees a register after when it is no longer live.

Parameters

name	The name of the owner of a register which needs to be freed

Definition at line 57 of file tac2mips.cpp.

5.95.3.9 void tac2mips::FreeRegister (std::string *name*)

Frees a register after when it is no longer live.

Parameters

name	The name of the owner of a register which needs to be freed

5.95.3.10 std::string tac2mips::GetRegister (std::string *tempName*)

Handles the allocation of registers including spilling when required.

Parameters

tempName	The name of the temporary register in the 3AC which needs to be mapped to a real register.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.95.3.11 std::string tac2mips::GetRegister (std::string *tempName*)

Handles the allocation of registers including spilling when required.

Parameters

tempName	The name of the temporary register in the 3AC which needs to be mapped to a real register.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.95.3.12 std::string tac2mips::GetRegister (std::string *tempName*)

Handles the allocation of registers including spilling when required.

Parameters

tempName	The name of the temporary register in the 3AC which needs to be mapped to a real register.

Returns

The name of the real register to which the temp register has been mapped/assigned.

5.95.3.13 void tac2mips::Label (std::string *txt*)

Adds a label to the MIPS file and formats it correctly.

Parameters

txt	A MIPS label to output, i.e. "spills"

Definition at line 147 of file tac2mips.cpp.

5.95.3.14 void tac2mips::Label (std::string *txt*)

Adds a label to the MIPS file and formats it correctly.

Parameters

txt	A MIPS label to output, i.e. "spills"

5.95.3.15 void tac2mips::Label (std::string *txt*)

Adds a label to the MIPS file and formats it correctly.

Parameters

txt	A MIPS label to output, i.e. "spills"

5.95.3.16 void tac2mips::Macro (std::string *name,* std::string *params*)

Outputs a macro with the given name and parameters.

Parameters

name	The name of the macro to output
params	The parameters for the macro

5.95.3.17 void tac2mips::Macro (std::string *name,* std::string *params*)

Outputs a macro with the given name and parameters.

Parameters

name	The name of the macro to output
params	The parameters for the macro

5.95.3.18 void tac2mips::Macro (std::string *name,* std::string *params*)

Outputs a macro with the given name and parameters.

Parameters

name	The name of the macro to output
params	The parameters for the macro

5.95.3.19 void tac2mips::Macro (std::string *name*)

Outputs a macro with the given name and no parameters.

Parameters

name	The name of the macro to output

5.95.3.20 void tac2mips::Macro (std::string *name*)

Outputs a macro with the given name and no parameters.

Parameters

name	The name of the macro to output

5.95.3.21 void tac2mips::Macro (std::string *name*)

Outputs a macro with the given name and no parameters.

Parameters

name	The name of the macro to output

5.95.3.22 void tac2mips::OutputPreamble ( )

Outputs the beginning of the MIPS file.

This includes items like an include directive to include the macros file which is used for simplifying the MIPS output. This also output comments referring to the source 3AC file.

Definition at line 67 of file tac2mips.cpp.

5.95.3.23 void tac2mips::OutputPreamble ( )

Outputs the beginning of the MIPS file.

This includes items like an include directive to include the macros file which is used for simplifying the MIPS output. This also output comments referring to the source 3AC file.

5.95.3.24 void tac2mips::OutputPreamble ( )

Outputs the beginning of the MIPS file.

This includes items like an include directive to include the macros file which is used for simplifying the MIPS output. This also output comments referring to the source 3AC file.

5.95.3.25 void tac2mips::scan_begin ( )

Initializes the scanning process.

This function sets up the scanner input streams.

5.95.3.26 void tac2mips::scan_begin ( )

Initializes the scanning process.

This function sets up the scanner input streams.

5.95.3.27 void tac2mips::scan_begin ( )

Initializes the scanning process.

This function sets up the scanner input streams.

5.95.3.28 void tac2mips::toMIPS (std::string *psuedoop*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

psuedoop	A MIPS psuedo-opcode command, i.e. ".globl"

Definition at line 117 of file tac2mips.cpp.

5.95.3.29 void tac2mips::toMIPS (std::string *psuedoop*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

psuedoop	A MIPS psuedo-opcode command, i.e. ".globl"

5.95.3.30 void tac2mips::toMIPS (std::string *psuedoop*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

psuedoop	A MIPS psuedo-opcode command, i.e. ".globl"

5.95.3.31 void tac2mips::toMIPS (std::string *opcode,* std::string *op1*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "jr"
op1	The operand, i.e. "$t31"

5.95.3.32 void tac2mips::toMIPS (std::string *opcode,* std::string *op1*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "jr"
op1	The operand, i.e. "$t31"

5.95.3.33 void tac2mips::toMIPS (std::string *opcode,* std::string *op1*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "jr"
op1	The operand, i.e. "$t31"

5.95.3.34 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "sw"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "0($sp)"

5.95.3.35 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "sw"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "0($sp)"

5.95.3.36 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "sw"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "0($sp)"

5.95.3.37 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2,* std::string *op3*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "add"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "$t1"
op3	The operand, i.e. "$t2"

5.95.3.38 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2,* std::string *op3*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "add"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "$t1"
op3	The operand, i.e. "$t2"

5.95.3.39 void tac2mips::toMIPS (std::string *opcode,* std::string *op1,* std::string *op2,* std::string *op3*)

Adds a line of code to the MIPS file and formats it correctly.

Parameters

opcode	A MIPS command, i.e. "add"
op1	The operand, i.e. "$t0"
op2	The operand, i.e. "$t1"
op3	The operand, i.e. "$t2"

5.95.3.40 void tac2mips::WS (int *lines* = `1`)

Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.

Parameters

lines	The number of blank lines to output

5.95.3.41 void tac2mips::WS (int *lines* = `1`)

Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.

Parameters

lines	The number of blank lines to output

Definition at line 109 of file tac2mips.cpp.

5.95.3.42 void tac2mips::WS (int *lines* = `1`)

Outputs the specified number of blank lines to the MIPS file in order make the MIPS easier to read.

Parameters

lines	The number of blank lines to output

The documentation for this class was generated from the following files:

- mips/tac2mips.h
- mips/tac2mips.cpp

5.96 TAC_Generator Class Reference

A class for generating three address code.

`#include <TAC_Generator.h>`

Public Types

- enum ThreeOpInstructions {
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC**,
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC** }

  Enumeration of 3 operand instructions.

- enum TwoOpInstructions {
NEG, NOT, ASSIGN, ADDR,
GLOBAL, STRING, IMMEDIATE_I, IMMEDIATE_F,
MOV, NEG, NOT, ASSIGN,
ADDR, GLOBAL, STRING, IMMEDIATE_I,
IMMEDIATE_F, MOV, NEG, NOT,
ASSIGN, ADDR, GLOBAL, STRING,
IMMEDIATE_I, IMMEDIATE_F, MOV, NEG,
NOT, ASSIGN, ADDR, GLOBAL,
STRING, IMMEDIATE_I, IMMEDIATE_F, MOV }

  Enum of 2 operand instructions.

- enum OneOpInstructions {
LABEL, BR, ARGS, REFOUT,
VALOUT, CALL, PROCENTRY, COMMENT,
BEGINFRAME, LABEL, BR, ARGS,
REFOUT, VALOUT, CALL, PROCENTRY,
COMMENT, BEGINFRAME, LABEL, BR,
ARGS, REFOUT, VALOUT, CALL,
PROCENTRY, COMMENT, BEGINFRAME, LABEL,
BR, ARGS, REFOUT, VALOUT,
CALL, PROCENTRY, COMMENT, BEGINFRAME }

  Enum of 1 operand instructions.

- enum NoOpInstructions {
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME }

Enum of instructions without operands.

- enum ThreeOpInstructions {
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC**,
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC** }

  Enumeration of 3 operand instructions.

- enum TwoOpInstructions {
NEG, NOT, ASSIGN, ADDR,
GLOBAL, STRING, IMMEDIATE_I, IMMEDIATE_F,
MOV, NEG, NOT, ASSIGN,
ADDR, GLOBAL, STRING, IMMEDIATE_I,
IMMEDIATE_F, MOV, NEG, NOT,
ASSIGN, ADDR, GLOBAL, STRING,
IMMEDIATE_I, IMMEDIATE_F, MOV, NEG,
NOT, ASSIGN, ADDR, GLOBAL,
STRING, IMMEDIATE_I, IMMEDIATE_F, MOV }

  Enum of 2 operand instructions.

- enum OneOpInstructions {
LABEL, BR, ARGS, REFOUT,
VALOUT, CALL, PROCENTRY, COMMENT,
BEGINFRAME, LABEL, BR, ARGS,
REFOUT, VALOUT, CALL, PROCENTRY,
COMMENT, BEGINFRAME, LABEL, BR,
ARGS, REFOUT, VALOUT, CALL,
PROCENTRY, COMMENT, BEGINFRAME, LABEL,
BR, ARGS, REFOUT, VALOUT,
CALL, PROCENTRY, COMMENT, BEGINFRAME }

  Enum of 1 operand instructions.

- enum NoOpInstructions {
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME }

  Enum of instructions without operands.

- enum ThreeOpInstructions {
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC**,
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC** }

  Enumeration of 3 operand instructions.

- enum TwoOpInstructions {
NEG, NOT, ASSIGN, ADDR,
GLOBAL, STRING, IMMEDIATE_I, IMMEDIATE_F,
MOV, NEG, NOT, ASSIGN,
ADDR, GLOBAL, STRING, IMMEDIATE_I,
IMMEDIATE_F, MOV, NEG, NOT,
ASSIGN, ADDR, GLOBAL, STRING,
IMMEDIATE_I, IMMEDIATE_F, MOV, NEG,
NOT, ASSIGN, ADDR, GLOBAL,
STRING, IMMEDIATE_I, IMMEDIATE_F, MOV }

  Enum of 2 operand instructions.

- enum OneOpInstructions {
LABEL, BR, ARGS, REFOUT,
VALOUT, CALL, PROCENTRY, COMMENT,
BEGINFRAME, LABEL, BR, ARGS,
REFOUT, VALOUT, CALL, PROCENTRY,
COMMENT, BEGINFRAME, LABEL, BR,
ARGS, REFOUT, VALOUT, CALL,
PROCENTRY, COMMENT, BEGINFRAME, LABEL,
BR, ARGS, REFOUT, VALOUT,
CALL, PROCENTRY, COMMENT, BEGINFRAME }

  Enum of 1 operand instructions.

- enum NoOpInstructions {
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME }

  Enum of instructions without operands.

- enum ThreeOpInstructions {
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC**,
ADD, SUB, MULT, DIV,
REM, SHIFTL, SHIFTR, AND,
XOR, OR, LOR, LAND,
EQ, GT, LT, GE,
LE, NE, BREQ, BRGT,
BRLT, BRGE, BRLE, BRNE,
BOUND, **ALLOC**, ADD, SUB,
MULT, DIV, **REM**, SHIFTL,
SHIFTR, AND, XOR, OR,
LOR, LAND, EQ, GT,
LT, GE, LE, NE,
BREQ, BRGT, BRLT, BRGE,
BRLE, BRNE, BOUND, **ALLOC** }

  Enumeration of 3 operand instructions.

- enum TwoOpInstructions {
NEG, NOT, ASSIGN, ADDR,
GLOBAL, STRING, IMMEDIATE_I, IMMEDIATE_F,
MOV, NEG, NOT, ASSIGN,
ADDR, GLOBAL, STRING, IMMEDIATE_I,
IMMEDIATE_F, MOV, NEG, NOT,
ASSIGN, ADDR, GLOBAL, STRING,
IMMEDIATE_I, IMMEDIATE_F, MOV, NEG,
NOT, ASSIGN, ADDR, GLOBAL,
STRING, IMMEDIATE_I, IMMEDIATE_F, MOV }

  Enum of 2 operand instructions.

- enum OneOpInstructions {
LABEL, BR, ARGS, REFOUT,
VALOUT, CALL, PROCENTRY, COMMENT,
BEGINFRAME, LABEL, BR, ARGS,
REFOUT, VALOUT, CALL, PROCENTRY,
COMMENT, BEGINFRAME, LABEL, BR,
ARGS, REFOUT, VALOUT, CALL,
PROCENTRY, COMMENT, BEGINFRAME, LABEL,
BR, ARGS, REFOUT, VALOUT,
CALL, PROCENTRY, COMMENT, BEGINFRAME }

  Enum of 1 operand instructions.

- enum NoOpInstructions {
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME,
HALT, ENDPROC, RETURN, ENDFRAME }

  Enum of instructions without operands.

Public Member Functions

- TAC_Generator (const string &filename)

  The paramaterized constructor.
- TAC_Generator ()

  The default constructor.
- ∼TAC_Generator ()

  The destructor.
- void toTAC (ThreeOpInstructions t, void ∗op1, void ∗op2, void ∗op3, string c="")

  Generate a 3AC string.
- void toTAC (TwoOpInstructions t, void ∗op1, void ∗op2, string c="")

  Generate a 3AC string.
- void toTAC (OneOpInstructions t, void ∗op1, string c="")

  Generate a 3AC string.
- void toTAC (NoOpInstructions t, string c="")

  Generate a 3AC string.
- void Fetch (string varName, SymTab ∗symbolTable, string targetTemp)

  Sets the symbol which should appear at the end of all comments.
- void **SetCommentStart** (string commentStart)
- void SetCommentEnd (string commentEnd)

  Sets the symbol which should appear at the beginning of all comments.
- void WriteComment (string comment)

  Writes a comment string to the 3AC output.
- void Blank ()

  Puts a blank line in the 3AC output.
- void SetFile (const string &filename)

  Sets the name of the file in which the output 3AC should be saved.
- void SetColumnWidth (int w)

  Sets the fixed column width for outputting 3AC statements.
- void SetFormatFlags (ios_base::fmtflags ff)

  Sets the ios_base format flags used when generating formatted 3AC strings.
- void SetBlankBeforeComments (bool flag)

  Sets the blankBeforeComments flag.
- TAC_Generator (const string &filename)

  The paramaterized constructor.
- TAC_Generator ()

  The default constructor.
- ∼TAC_Generator ()

  The destructor.
- void toTAC (ThreeOpInstructions t, void ∗op1, void ∗op2, void ∗op3, string c="")

  Generate a 3AC string.
- void toTAC (TwoOpInstructions t, void ∗op1, void ∗op2, string c="")

  Generate a 3AC string.
- void toTAC (OneOpInstructions t, void ∗op1, string c="")

  Generate a 3AC string.
- void toTAC (NoOpInstructions t, string c="")

  Generate a 3AC string.
- void Fetch (string varName, SymTab ∗symbolTable, string targetTemp)

  Sets the symbol which should appear at the end of all comments.
- void **SetCommentStart** (string commentStart)
- void SetCommentEnd (string commentEnd)

Sets the symbol which should appear at the beginning of all comments.

- void WriteComment (string comment)

  Writes a comment string to the 3AC output.
- void Blank ()

  Puts a blank line in the 3AC output.
- void SetFile (const string &filename)

  Sets the name of the file in which the output 3AC should be saved.
- void SetColumnWidth (int w)

  Sets the fixed column width for outputting 3AC statements.
- void SetFormatFlags (ios_base::fmtflags ff)

  Sets the ios_base format flags used when generating formatted 3AC strings.
- void SetBlankBeforeComments (bool flag)

  Sets the blankBeforeComments flag.
- TAC_Generator (const string &filename)

  The paramaterized constructor.
- TAC_Generator ()

  The default constructor.
- ∼TAC_Generator ()

  The destructor.
- void toTAC (ThreeOpInstructions t, void ∗op1, void ∗op2, void ∗op3, string c="")

  Generate a 3AC string.
- void toTAC (TwoOpInstructions t, void ∗op1, void ∗op2, string c="")

  Generate a 3AC string.
- void toTAC (OneOpInstructions t, void ∗op1, string c="")

  Generate a 3AC string.
- void toTAC (NoOpInstructions t, string c="")

  Generate a 3AC string.
- void Fetch (string varName, SymTab ∗symbolTable, string targetTemp)

  Sets the symbol which should appear at the end of all comments.
- void **SetCommentStart** (string commentStart)
- void SetCommentEnd (string commentEnd)

  Sets the symbol which should appear at the beginning of all comments.
- void WriteComment (string comment)

  Writes a comment string to the 3AC output.
- void Blank ()

  Puts a blank line in the 3AC output.
- void SetFile (const string &filename)

  Sets the name of the file in which the output 3AC should be saved.
- void SetColumnWidth (int w)

  Sets the fixed column width for outputting 3AC statements.
- void SetFormatFlags (ios_base::fmtflags ff)

  Sets the ios_base format flags used when generating formatted 3AC strings.
- void SetBlankBeforeComments (bool flag)

  Sets the blankBeforeComments flag.
- TAC_Generator (const string &filename)

  The paramaterized constructor.
- TAC_Generator ()

  The default constructor.
- ∼TAC_Generator ()

  The destructor.
- void toTAC (ThreeOpInstructions t, void ∗op1, void ∗op2, void ∗op3, string c="")

Generate a 3AC string.

- void toTAC (TwoOpInstructions t, void ∗op1, void ∗op2, string c="")

  Generate a 3AC string.
- void toTAC (OneOpInstructions t, void ∗op1, string c="")

  Generate a 3AC string.
- void toTAC (NoOpInstructions t, string c="")

  Generate a 3AC string.
- void Fetch (string varName, SymTab ∗symbolTable, string targetTemp)

  Sets the symbol which should appear at the end of all comments.
- void **SetCommentStart** (string commentStart)
- void SetCommentEnd (string commentEnd)

  Sets the symbol which should appear at the beginning of all comments.
- void WriteComment (string comment)

  Writes a comment string to the 3AC output.
- void Blank ()

  Puts a blank line in the 3AC output.
- void SetFile (const string &filename)

  Sets the name of the file in which the output 3AC should be saved.
- void SetColumnWidth (int w)

  Sets the fixed column width for outputting 3AC statements.
- void SetFormatFlags (ios_base::fmtflags ff)

  Sets the ios_base format flags used when generating formatted 3AC strings.
- void SetBlankBeforeComments (bool flag)

  Sets the blankBeforeComments flag.

Static Public Member Functions

- static string GetLabelName ()

  Generates a unique label string.
- static string GetIVarName ()

  Generates a unique string for integer temps.
- static string GetFVarName ()

  Generates a unique string for floating-point temps.
- static string GetLabelName ()

  Generates a unique label string.
- static string GetIVarName ()

  Generates a unique string for integer temps.
- static string GetFVarName ()

  Generates a unique string for floating-point temps.
- static string GetLabelName ()

  Generates a unique label string.
- static string GetIVarName ()

  Generates a unique string for integer temps.
- static string GetFVarName ()

  Generates a unique string for floating-point temps.
- static string GetLabelName ()

  Generates a unique label string.
- static string GetIVarName ()

  Generates a unique string for integer temps.
- static string GetFVarName ()

  Generates a unique string for floating-point temps.

Private Member Functions

- void Emit (string CodeToEmit)

  This function saves the string passed in to a STL list for later output.
- void Emit (string CodeToEmit)

  This function saves the string passed in to a STL list for later output.
- void Emit (string CodeToEmit)

  This function saves the string passed in to a STL list for later output.
- void Emit (string CodeToEmit)

  This function saves the string passed in to a STL list for later output.

Private Attributes

- list< string > buffer

  A buffer for the generated 3AC.
- ofstream fout

  Output stream.
- string commentStart

  String to be placed at the beginning of every comment.
- string commentEnd

  String to be placed at the end of every comment.
- bool blankBeforeComment

  Flag for placing blank lines before comments.
- int width

  Fixed column width of the output 3AC.
- ios_base::fmtflags flags

  Format flags.
- string **CurrentLabel**

Static Private Attributes

- static int lCount = 0

  Current label counter for generating unique labels.
- static int iCount = 0

  Current integer counter for generating unique integer labels.
- static int fCount = 0

  Current float counter for generating unique float labels.

5.96.1 Detailed Description

A class for generating three address code.

The TAC_Generator class is responsible for generating well-formatted three address code (3AC or TAC). The generator stores all generated 3AC in a STL list of strings during runtime, and outputs the 3AC to a file when the destructor is called. This allows for the 3AC to be manipulated prior to output (i.e. putting all function decls at the top of the 3AC).

Definition at line 24 of file CParser.yy.

5.96.2 Member Enumeration Documentation

5.96.2.1 enum TAC_Generator::NoOpInstructions

Enum of instructions without operands.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

Definition at line 107 of file CParser.yy.

5.96.2.2 enum TAC_Generator::NoOpInstructions

Enum of instructions without operands.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

Definition at line 107 of file CParser.yy.

5.96.2.3 enum TAC_Generator::NoOpInstructions

Enum of instructions without operands.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

Definition at line 107 of file TAC_Generator.h.

5.96.2.4 enum TAC_Generator::NoOpInstructions

Enum of instructions without operands.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

HALT Immediately halt execution.

ENDPROC Mark the end of a procedure.

RETURN Return control to the caller.

ENDFRAME Marks the end of a stack frame.

Definition at line 107 of file CScanner.ll.

5.96.2.5 enum TAC_Generator::OneOpInstructions

Enum of 1 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

Definition at line 88 of file CParser.yy.

5.96.2.6 enum TAC_Generator::OneOpInstructions

Enum of 1 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

Definition at line 88 of file CScanner.ll.

5.96.2.7 enum TAC_Generator::OneOpInstructions

Enum of 1 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

Definition at line 88 of file CParser.yy.

5.96.2.8 enum TAC_Generator::OneOpInstructions

Enum of 1 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

LABEL Generate a label.

BR Branch to a label.

ARGS Specify the number of arguments being sent to the next call.

REFOUT Pass op1 by reference.

VALOUT Pass op1 by value.

CALL Call the procedure named op1.

PROCENTRY Marks the beginning of a procedure.

COMMENT Output op1 as a comment.

BEGINFRAME Marks the beginning of a new stack frame, and passes in the size of the memory required on the stack.

Definition at line 88 of file TAC_Generator.h.

5.96.2.9 enum **TAC_Generator::ThreeOpInstructions**

Enumeration of 3 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void $*$ parameters to the toTAC functions are.

Enumerator

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>=$ op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

Definition at line 33 of file CParser.yy.

5.96.2.10 enum TAC_Generator::ThreeOpInstructions

Enumeration of 3 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>$= op2, or 0 otherwise.

LE Set op3 to 1 is op1 $<$= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 $<$ op2) goto op3.

BRGE If(op1 $>$= op2) goto op3.

BRLE If(op1 $<$= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>$= op2, or 0 otherwise.

LE Set op3 to 1 is op1 $<$= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

Definition at line 33 of file CScanner.ll.

5.96.2.11 enum TAC_Generator::ThreeOpInstructions

Enumeration of 3 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>$= op2, or 0 otherwise.

LE Set op3 to 1 is op1 $<$= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 $<$ op2) goto op3.

BRGE If(op1 $>$= op2) goto op3.

BRLE If(op1 $<$= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>$= op2, or 0 otherwise.

LE Set op3 to 1 is op1 $<$= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 $<$ op2) goto op3.

BRGE If(op1 $>$= op2) goto op3.

BRLE If(op1 $<$= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

Definition at line 33 of file CParser.yy.

5.96.2.12 enum **TAC_Generator::ThreeOpInstructions**

Enumeration of 3 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 $<$ op2) goto op3.

BRGE If(op1 $>$= op2) goto op3.

BRLE If(op1 $<$= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 $<$ op2, or 0 otherwise.

GE Set op3 to 1 is op1 $>$= op2, or 0 otherwise.

LE Set op3 to 1 is op1 $<$= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 $>$ op2) goto op3.

BRLT If(op1 $<$ op2) goto op3.

BRGE If(op1 $>$= op2) goto op3.

BRLE If(op1 $<$= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL $<$ Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 $>$ op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

ADD Add the value of two temps.

SUB Subtract the value of two temps.

MULT Multiply the value of two temps.

DIV Divide the value of two temps.

SHIFTL < Reminder of the value of two temps left shift the value of two temps

SHIFTR right shift the value of two temps

AND biwise and the value of two temps

XOR biwise xor the value of two temps

OR biwise OR the value of two temps

LOR logical OR the value of two temps

LAND logical OR the value of two temps

EQ Set op3 to 1 is op1 == op2, or 0 otherwise.

GT Set op3 to 1 is op1 > op2, or 0 otherwise.

LT Set op3 to 1 is op1 < op2, or 0 otherwise.

GE Set op3 to 1 is op1 >= op2, or 0 otherwise.

LE Set op3 to 1 is op1 <= op2, or 0 otherwise.

NE Set op3 to 1 is op1 != op2, or 0 otherwise.

BREQ If(op1 == op2) goto op3.

BRGT If(op1 > op2) goto op3.

BRLT If(op1 < op2) goto op3.

BRGE If(op1 >= op2) goto op3.

BRLE If(op1 <= op2) goto op3.

BRNE If(op1 != op2) goto op3.

BOUND Checks the bounds of an array access.

Definition at line 33 of file TAC_Generator.h.

5.96.2.13 enum TAC_Generator::TwoOpInstructions

Enum of 2 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

Definition at line 69 of file CParser.yy.

5.96.2.14 enum TAC_Generator::TwoOpInstructions

Enum of 2 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

Definition at line 69 of file CParser.yy.

5.96.2.15 enum TAC_Generator::TwoOpInstructions

Enum of 2 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

Definition at line 69 of file CScanner.ll.

5.96.2.16 enum TAC_Generator::TwoOpInstructions

Enum of 2 operand instructions.

These enum values serve as flags to the toTAC functions in order to indicate which 3AC statement should be generated, and what the types of the void ∗ parameters to the toTAC functions are.

Enumerator

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

NEG op2 = -(op1)

NOT Set op2 to 1 if op1 == 0, or 0 otherwise.

ASSIGN Assign the value of op1 to op2.

ADDR Assign the address of op1 to op2.

GLOBAL Declare op1 as a global of size op2.

STRING Associate string op1 with label op2.

IMMEDIATE_I Load an integer into a register.

IMMEDIATE_F Load a float into a register.

MOV does op1 = op2

Definition at line 69 of file TAC_Generator.h.

5.96.3 Constructor & Destructor Documentation

5.96.3.1 TAC_Generator::TAC_Generator (const string & *filename*)

The paramaterized constructor.

This constructor opens the 3AC file with the given filename.

Parameters

filename	The name of the file in which to output 3AC

Definition at line 8 of file TAC_Generator.cpp.

5.96.3.2 TAC_Generator::TAC_Generator ( )

The default constructor.

This constructor does not open an output file. If this constructor is used, then the function SetFile must be called.

See Also

SetFile()

Definition at line 22 of file TAC_Generator.cpp.

5.96.3.3 TAC_Generator::∼TAC_Generator ( )

The destructor.

This destructor is responsible for outputting the 3AC from the list of strings to the output file and then closing the output file.

Definition at line 36 of file TAC_Generator.cpp.

5.96.3.4 TAC_Generator::TAC_Generator (const string & *filename*)

The paramaterized constructor.

This constructor opens the 3AC file with the given filename.

Parameters

filename	The name of the file in which to output 3AC

5.96.3.5 TAC_Generator::TAC_Generator ( )

The default constructor.

This constructor does not open an output file. If this constructor is used, then the function SetFile must be called.

See Also

SetFile()

5.96.3.6 TAC_Generator::∼TAC_Generator ( )

The destructor.

This destructor is responsible for outputting the 3AC from the list of strings to the output file and then closing the output file.

5.96.3.7 TAC_Generator::TAC_Generator (const string & *filename*)

The paramaterized constructor.

This constructor opens the 3AC file with the given filename.

Parameters

filename	The name of the file in which to output 3AC

5.96.3.8 TAC_Generator::TAC_Generator ( )

The default constructor.

This constructor does not open an output file. If this constructor is used, then the function SetFile must be called.

See Also

SetFile()

5.96.3.9 TAC_Generator::∼TAC_Generator ( )

The destructor.

This destructor is responsible for outputting the 3AC from the list of strings to the output file and then closing the output file.

5.96.3.10 TAC_Generator::TAC_Generator (const string & *filename*)

The paramaterized constructor.

This constructor opens the 3AC file with the given filename.

Parameters

filename	The name of the file in which to output 3AC

5.96.3.11 TAC_Generator::TAC_Generator ( )

The default constructor.

This constructor does not open an output file. If this constructor is used, then the function SetFile must be called.

See Also

SetFile()

5.96.3.12 TAC_Generator::∼TAC_Generator ( )

The destructor.

This destructor is responsible for outputting the 3AC from the list of strings to the output file and then closing the output file.

5.96.4 Member Function Documentation

5.96.4.1 void TAC_Generator::Emit (string *CodeToEmit*) `[private]`

This function saves the string passed in to a STL list for later output.

NOTE: The 3AC Generator "emits" code to a list first, and then after all code has been emitted, it is pushed to a file. This is done so as to allow for changes to be made to the 3AC before it is finalized (ie: moving all function decls to the top of the code)

5.96.4.2 void TAC_Generator::Emit (string *CodeToEmit*) [private]

This function saves the string passed in to a STL list for later output.

NOTE: The 3AC Generator "emits" code to a list first, and then after all code has been emitted, it is pushed to a file. This is done so as to allow for changes to be made to the 3AC before it is finalized (ie: moving all function decls to the top of the code)

5.96.4.3 void TAC_Generator::Emit (string *CodeToEmit*) [private]

This function saves the string passed in to a STL list for later output.

NOTE: The 3AC Generator "emits" code to a list first, and then after all code has been emitted, it is pushed to a file. This is done so as to allow for changes to be made to the 3AC before it is finalized (ie: moving all function decls to the top of the code)

Definition at line 684 of file TAC_Generator.cpp.

5.96.4.4 void TAC_Generator::Emit (string *CodeToEmit*) [private]

This function saves the string passed in to a STL list for later output.

NOTE: The 3AC Generator "emits" code to a list first, and then after all code has been emitted, it is pushed to a file. This is done so as to allow for changes to be made to the 3AC before it is finalized (ie: moving all function decls to the top of the code)

5.96.4.5 void TAC_Generator::Fetch (string *varName,* SymTab ∗ *symbolTable,* string *targetTemp*)

Sets the symbol which should appear at the end of all comments.

Parameters

commentStart	String to be place at the beginning of every comment

```
TODO Add code to fetch the symbole for the symbol table
```

Assume all variables are GLOBAL for the time being

Definition at line 758 of file TAC_Generator.cpp.

5.96.4.6 void TAC_Generator::Fetch (string *varName,* SymTab ∗ *symbolTable,* string *targetTemp*)

Sets the symbol which should appear at the end of all comments.

Parameters

commentStart	String to be place at the beginning of every comment

5.96.4.7 void TAC_Generator::Fetch (string *varName,* SymTab ∗ *symbolTable,* string *targetTemp*)

Sets the symbol which should appear at the end of all comments.

Parameters

commentStart	String to be place at the beginning of every comment

5.96.4.8 void TAC_Generator::Fetch (string *varName,* SymTab ∗ *symbolTable,* string *targetTemp*)

Sets the symbol which should appear at the end of all comments.

Parameters

commentStart	String to be place at the beginning of every comment

5.96.4.9 void TAC_Generator::SetBlankBeforeComments (bool *flag*)

Sets the blankBeforeComments flag.

If true, a blank line will be output in the final 3AC before each comment.

Parameters

flag	True if there should be an empty line before each comment

5.96.4.10 void TAC_Generator::SetBlankBeforeComments (bool *flag*)

Sets the blankBeforeComments flag.

If true, a blank line will be output in the final 3AC before each comment.

Parameters

flag	True if there should be an empty line before each comment

5.96.4.11 void TAC_Generator::SetBlankBeforeComments (bool *flag*)

Sets the blankBeforeComments flag.

If true, a blank line will be output in the final 3AC before each comment.

Parameters

flag	True if there should be an empty line before each comment

5.96.4.12 void TAC_Generator::SetBlankBeforeComments (bool *flag*)

Sets the blankBeforeComments flag.

If true, a blank line will be output in the final 3AC before each comment.

Parameters

flag	True if there should be an empty line before each comment

Definition at line 708 of file TAC_Generator.cpp.

5.96.4.13 void TAC_Generator::SetColumnWidth (int *w*)

Sets the fixed column width for outputting 3AC statements.

Parameters

w	Integer indicating the width of the columns to print the 3AC in.

5.96.4.14 void TAC_Generator::SetColumnWidth (int *w*)

Sets the fixed column width for outputting 3AC statements.

Parameters

w	Integer indicating the width of the columns to print the 3AC in.

5.96.4.15 void TAC_Generator::SetColumnWidth (int *w*)

Sets the fixed column width for outputting 3AC statements.

Parameters

w	Integer indicating the width of the columns to print the 3AC in.

Definition at line 698 of file TAC_Generator.cpp.

5.96.4.16 void TAC_Generator::SetColumnWidth (int *w*)

Sets the fixed column width for outputting 3AC statements.

Parameters

w	Integer indicating the width of the columns to print the 3AC in.

5.96.4.17 void TAC_Generator::SetCommentEnd (string *commentEnd*)

Sets the symbol which should appear at the beginning of all comments.

Parameters

commentEnd	String to be placed at the end of every comment

5.96.4.18 void TAC_Generator::SetCommentEnd (string *commentEnd*)

Sets the symbol which should appear at the beginning of all comments.

Parameters

commentEnd	String to be placed at the end of every comment

5.96.4.19 void TAC_Generator::SetCommentEnd (string *commentEnd*)

Sets the symbol which should appear at the beginning of all comments.

Parameters

commentEnd	String to be placed at the end of every comment

Definition at line 664 of file TAC_Generator.cpp.

5.96.4.20 void TAC_Generator::SetCommentEnd (string *commentEnd*)

Sets the symbol which should appear at the beginning of all comments.

Parameters

commentEnd	String to be placed at the end of every comment

5.96.4.21 void TAC_Generator::SetFile (const string & *filename*)

Sets the name of the file in which the output 3AC should be saved.

Parameters

filename	The name of the file in which to output 3AC

5.96.4.22 void TAC_Generator::SetFile (const string & *filename*)

Sets the name of the file in which the output 3AC should be saved.

Parameters

filename	The name of the file in which to output 3AC

Definition at line 689 of file TAC_Generator.cpp.

5.96.4.23 void TAC_Generator::SetFile (const string & *filename*)

Sets the name of the file in which the output 3AC should be saved.

Parameters

filename	The name of the file in which to output 3AC

5.96.4.24 void TAC_Generator::SetFile (const string & *filename*)

Sets the name of the file in which the output 3AC should be saved.

Parameters

filename	The name of the file in which to output 3AC

5.96.4.25 void TAC_Generator::SetFormatFlags (ios_base::fmtflags *ff*)

Sets the ios_base format flags used when generating formatted 3AC strings.

Parameters

ff	Format flags (i.e. left, right, etc.)

5.96.4.26 void TAC_Generator::SetFormatFlags (ios_base::fmtflags *ff*)

Sets the ios_base format flags used when generating formatted 3AC strings.

Parameters

ff	Format flags (i.e. left, right, etc.)

5.96.4.27 void TAC_Generator::SetFormatFlags (ios_base::fmtflags *ff*)

Sets the ios_base format flags used when generating formatted 3AC strings.

Parameters

ff	Format flags (i.e. left, right, etc.)

Definition at line 703 of file TAC_Generator.cpp.

5.96.4.28 void TAC_Generator::SetFormatFlags (ios_base::fmtflags *ff*)

Sets the ios_base format flags used when generating formatted 3AC strings.

Parameters

ff	Format flags (i.e. left, right, etc.)

5.96.4.29 void TAC_Generator::toTAC (ThreeOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, void ∗ *op3*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
op3	A pointer to the third operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

Definition at line 44 of file TAC_Generator.cpp.

5.96.4.30 void TAC_Generator::toTAC (ThreeOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, void ∗ *op3*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
op3	A pointer to the third operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.31 void TAC_Generator::toTAC (ThreeOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, void ∗ *op3*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
op3	A pointer to the third operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.32 void TAC_Generator::toTAC (ThreeOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, void ∗ *op3*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
op3	A pointer to the third operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.33 void TAC_Generator::toTAC (TwoOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.34 void TAC_Generator::toTAC (TwoOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.35 void TAC_Generator::toTAC (TwoOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.36 void TAC_Generator::toTAC (TwoOpInstructions *t*, void ∗ *op1*, void ∗ *op2*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operands based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
op2	A pointer to the second operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

Definition at line 403 of file TAC_Generator.cpp.

5.96.4.37 void TAC_Generator::toTAC (OneOpInstructions *t*, void ∗ *op1*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operand based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.38 void TAC_Generator::toTAC (OneOpInstructions *t*, void ∗ *op1*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operand based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.39 void TAC_Generator::toTAC (OneOpInstructions *t*, void ∗ *op1*, string *c* = " ")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operand based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

Definition at line 531 of file TAC_Generator.cpp.

5.96.4.40 void TAC_Generator::toTAC (OneOpInstructions *t*, void ∗ *op1*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement. NOTE: this function determines the type of the operand based on the flag passed as the first parameter. If the incorrect flag is passed, then the program will cast the address to the wrong type, so be careful.

Parameters

t	Flag indicating the type of 3AC statement to generate
op1	A pointer to the first operand (cast as a void∗)
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.41 void TAC_Generator::toTAC (NoOpInstructions *t*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement.

Parameters

t	Flag indicating the type of 3AC statement to generate
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.42 void TAC_Generator::toTAC (NoOpInstructions *t*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement.

Parameters

t	Flag indicating the type of 3AC statement to generate
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.43 void TAC_Generator::toTAC (NoOpInstructions *t*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement.

Parameters

t	Flag indicating the type of 3AC statement to generate
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

5.96.4.44 void TAC_Generator::toTAC (NoOpInstructions *t*, string *c* = "")

Generate a 3AC string.

The toTAC overloads take in a flag to indicate the type of three address code statement and a series of parameters required by the particular statement, in order to generate a formatted 3AC statement.

Parameters

t	Flag indicating the type of 3AC statement to generate
c	An optional comment to prepend to the 3AC statement (useful for outputting the original input code as comments to the 3AC file)

Definition at line 627 of file TAC_Generator.cpp.

5.96.4.45 void TAC_Generator::WriteComment (string *comment*)

Writes a comment string to the 3AC output.

Parameters

comment	String to output as a comment

Definition at line 669 of file TAC_Generator.cpp.

5.96.4.46 void TAC_Generator::WriteComment (string *comment*)

Writes a comment string to the 3AC output.

Parameters

comment	String to output as a comment

5.96.4.47 void TAC_Generator::WriteComment (string *comment*)

Writes a comment string to the 3AC output.

Parameters

comment	String to output as a comment

5.96.4.48 void TAC_Generator::WriteComment (string *comment*)

Writes a comment string to the 3AC output.

Parameters

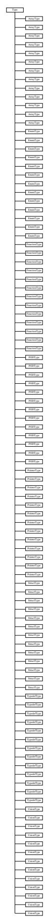
comment	String to output as a comment

The documentation for this class was generated from the following files:

- TAC_Generator.h
- TAC_Generator.cpp

5.97 Type Class Reference

Inheritance diagram for Type:



Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)

- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **Type** (string n, int s)
- **Type** (Type &t)
- string **GetName** ()
- int **GetSize** ()
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- string **name**
- int **size**

5.97.1 Detailed Description

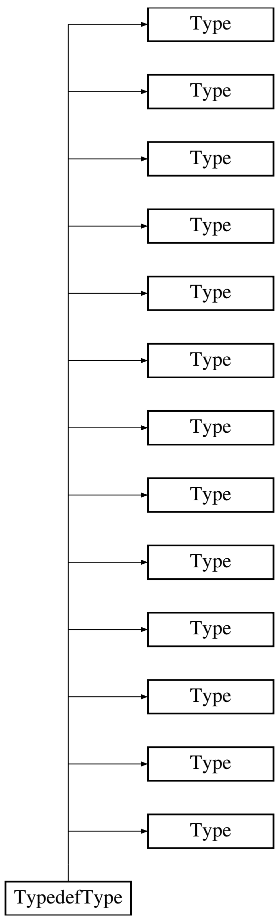
Definition at line 60 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.98 TypedefType Class Reference

Inheritance diagram for TypedefType:


Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
Type
TypedefType

Public Types

- enum **DerivedType** {
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

Public Member Functions

- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()

- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **TypedefType** (Type ∗actual, string tdname)
- Type ∗ **GetActual** ()
- string **GetTypedefName** ()
- bool **CheckType** (TypedefType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()

- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- Type ∗ **actualType**
- string **typedefName**
- string **name**
- int **size**

5.98.1 Detailed Description

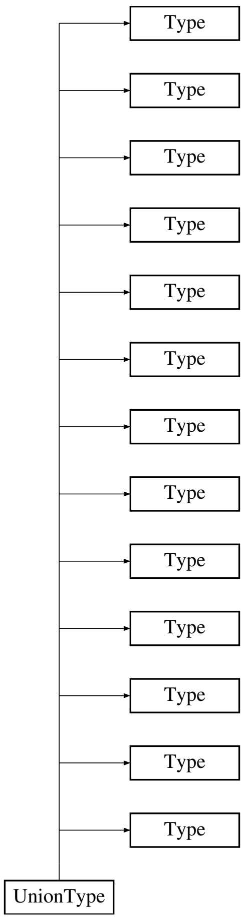
Definition at line 101 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.99 UnionType Class Reference

Inheritance diagram for UnionType:



Public Types

- enum **DerivedType** {
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
  ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
  FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
  TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
  UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
  PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
  STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
  BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
  ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
  POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE** }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

- enum **DerivedType** {

BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE, **BASE**, **PODTYPE**, **TYPEDEFTYPE**,
ENUMTYPE, **ARRAYTYPE**, **STRUCTTYPE**, **UNIONTYPE**,
FUNCTIONTYPE, **POINTERTYPE**, **BASE**, **PODTYPE**,
TYPEDEFTYPE, **ENUMTYPE**, **ARRAYTYPE**, **STRUCTTYPE**,
UNIONTYPE, **FUNCTIONTYPE**, **POINTERTYPE**, **BASE**,
PODTYPE, **TYPEDEFTYPE**, **ENUMTYPE**, **ARRAYTYPE**,
STRUCTTYPE, **UNIONTYPE**, **FUNCTIONTYPE**, **POINTERTYPE**,
BASE, **PODTYPE**, **TYPEDEFTYPE**, **ENUMTYPE**,
ARRAYTYPE, **STRUCTTYPE**, **UNIONTYPE**, **FUNCTIONTYPE**,
POINTERTYPE }

Public Member Functions

- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)

- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- **UnionType** (string n)
- void **AddMember** (string s, Type ∗t)
- bool **MemberExists** (string s)
- bool **CheckType** (UnionType ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- string **GetName** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()
- int **GetSize** ()

- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- void **SetName** (string n)
- bool **CheckType** (Type ∗rhs, bool &isConvertable, CONVERSIONTYPE &t)

Static Public Member Functions

- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)
- static Type ∗ **GetResultingType** (CONVERSIONTYPE ct, bool castUp)

Public Attributes

- enum Type::DerivedType **t**

Protected Attributes

- vector< string > **memberNames**
- vector< Type ∗ > **memberTypes**
- string **name**
- int **size**

5.99.1 Detailed Description

Definition at line 157 of file CParser.yy.

The documentation for this class was generated from the following files:

- Type.h
- Type.cpp

5.100 Visualizer Class Reference

A class for visualizing the generation of the AST.

`#include <Visualizer.h>`

Public Member Functions

- Visualizer ()

  Default constructor.
- Visualizer (string fname)

  Parameterized constructor.
- ∼Visualizer ()

  Destructor.
- void begin ()

  Creates the opening part of the GraphViz file.
- void end ()

  Creates the closing part of the GraphViz file.
- void addNode (int uid, string label)

  Adds a node to the graph with a unique id and a label.
- void addNode (int parentid, int childid, string parent_label)

  Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.
- void addDummyNode (int parentid, string label)

  Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.
- void addEdge (int parent, int child)

  Adds an edge from a parent node to a child node.
- Visualizer ()

  Default constructor.
- Visualizer (string fname)

  Parameterized constructor.
- ∼Visualizer ()

  Destructor.
- void begin ()

  Creates the opening part of the GraphViz file.
- void end ()

  Creates the closing part of the GraphViz file.
- void addNode (int uid, string label)

  Adds a node to the graph with a unique id and a label.
- void addNode (int parentid, int childid, string parent_label)

  Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.
- void addDummyNode (int parentid, string label)

  Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.
- void addEdge (int parent, int child)

  Adds an edge from a parent node to a child node.
- Visualizer ()

  Default constructor.
- Visualizer (string fname)

  Parameterized constructor.
- ∼Visualizer ()

  Destructor.
- void begin ()

Creates the opening part of the GraphViz file.
- void end ()

  Creates the closing part of the GraphViz file.
- void addNode (int uid, string label)

  Adds a node to the graph with a unique id and a label.
- void addNode (int parentid, int childid, string parent_label)

  Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.
- void addDummyNode (int parentid, string label)

  Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.
- void addEdge (int parent, int child)

  Adds an edge from a parent node to a child node.
- Visualizer ()

  Default constructor.
- Visualizer (string fname)

  Parameterized constructor.
- ∼Visualizer ()

  Destructor.
- void begin ()

  Creates the opening part of the GraphViz file.
- void end ()

  Creates the closing part of the GraphViz file.
- void addNode (int uid, string label)

  Adds a node to the graph with a unique id and a label.
- void addNode (int parentid, int childid, string parent_label)

  Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.
- void addDummyNode (int parentid, string label)

  Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.
- void addEdge (int parent, int child)

  Adds an edge from a parent node to a child node.

Static Public Member Functions

- static int GetNextUID ()

  Gets a unique id to be used in adding nodes to the AST.
- static int GetNextUID ()

  Gets a unique id to be used in adding nodes to the AST.
- static int GetNextUID ()

  Gets a unique id to be used in adding nodes to the AST.
- static int GetNextUID ()

  Gets a unique id to be used in adding nodes to the AST.

Private Attributes

- fstream file

  The filename of the output GraphViz file.
- string gname

  The name of the graph in the GraphViz file.

Static Private Attributes

- static int nextUID = 0

  The next unique id value.

5.100.1 Detailed Description

A class for visualizing the generation of the AST.

The Visualizer class provides a method for generating a GraphViz output file which can be converted to a graphic representing the AST.

Definition at line 17 of file CParser.yy.

5.100.2 Constructor & Destructor Documentation

5.100.2.1 Visualizer::Visualizer ( )

Default constructor.

This version of the constructor defaults the output filename to "vis.dot". It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

Definition at line 5 of file Visualizer.cpp.

5.100.2.2 Visualizer::Visualizer (string *fname*)

Parameterized constructor.

This version of the constructor takes the output filename as a parameter. It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

Parameters

fname	The output GraphVis file filename

Definition at line 20 of file Visualizer.cpp.

5.100.2.3 Visualizer::∼Visualizer ( )

Destructor.

Completes the GraphViz file and closes the filestream.

Definition at line 35 of file Visualizer.cpp.

5.100.2.4 Visualizer::Visualizer ( )

Default constructor.

This version of the constructor defaults the output filename to "vis.dot". It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

5.100.2.5 Visualizer::Visualizer (string *fname*)

Parameterized constructor.

This version of the constructor takes the output filename as a parameter. It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

Parameters

fname	The output GraphVis file filename

5.100.2.6 Visualizer::∼Visualizer ( )

Destructor.

Completes the GraphViz file and closes the filestream.

5.100.2.7 Visualizer::Visualizer ( )

Default constructor.

This version of the constructor defaults the output filename to "vis.dot". It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

5.100.2.8 Visualizer::Visualizer (string *fname*)

Parameterized constructor.

This version of the constructor takes the output filename as a parameter. It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

Parameters

fname	The output GraphVis file filename

5.100.2.9 Visualizer::∼Visualizer ( )

Destructor.

Completes the GraphViz file and closes the filestream.

5.100.2.10 Visualizer::Visualizer ( )

Default constructor.

This version of the constructor defaults the output filename to "vis.dot". It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

5.100.2.11 Visualizer::Visualizer (string *fname*)

Parameterized constructor.

This version of the constructor takes the output filename as a parameter. It attempts to open the file, and exits with EXIT_FAILURE if the file cannot be opened.

Parameters

fname	The output GraphVis file filename

5.100.2.12 Visualizer::∼Visualizer ( )

Destructor.

Completes the GraphViz file and closes the filestream.

5.100.3 Member Function Documentation

5.100.3.1 void Visualizer::addDummyNode (int *parentid,* string *label*)

Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.

Parameters

uid	The unique id of the parent node off of which the dummy node should be attached
label	The label to go inside the dummy node

5.100.3.2 void Visualizer::addDummyNode (int *parentid,* string *label*)

Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.

Parameters

uid	The unique id of the parent node off of which the dummy node should be attached
label	The label to go inside the dummy node

5.100.3.3 void Visualizer::addDummyNode (int *parentid,* string *label*)

Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.

Parameters

uid	The unique id of the parent node off of which the dummy node should be attached
label	The label to go inside the dummy node

Definition at line 63 of file Visualizer.cpp.

5.100.3.4 void Visualizer::addDummyNode (int *parentid,* string *label*)

Adds a node to the graph which is only for visualizing extra info rather than visualizing an actual node in the AST.

Parameters

uid	The unique id of the parent node off of which the dummy node should be attached
label	The label to go inside the dummy node

5.100.3.5 void Visualizer::addEdge (int *parent,* int *child*)

Adds an edge from a parent node to a child node.

Parameters

parent	The unique id of the parent
child	The unique id of the child

5.100.3.6 void Visualizer::addEdge (int *parent,* int *child*)

Adds an edge from a parent node to a child node.

Parameters

parent	The unique id of the parent
child	The unique id of the child

5.100.3.7 void Visualizer::addEdge (int *parent,* int *child*)

Adds an edge from a parent node to a child node.

Parameters

parent	The unique id of the parent
child	The unique id of the child

5.100.3.8 void Visualizer::addEdge (int *parent,* int *child*)

Adds an edge from a parent node to a child node.

Parameters

parent	The unique id of the parent
child	The unique id of the child

Definition at line 71 of file Visualizer.cpp.

5.100.3.9 void Visualizer::addNode (int *uid,* string *label*)

Adds a node to the graph with a unique id and a label.

Parameters

uid	A unique id
label	The label to go inside the node in the graph

5.100.3.10 void Visualizer::addNode (int *uid,* string *label*)

Adds a node to the graph with a unique id and a label.

Parameters

uid	A unique id
label	The label to go inside the node in the graph

Definition at line 51 of file Visualizer.cpp.

5.100.3.11 void Visualizer::addNode (int *uid,* string *label*)

Adds a node to the graph with a unique id and a label.

Parameters

uid	A unique id
label	The label to go inside the node in the graph

5.100.3.12 void Visualizer::addNode (int *uid,* string *label*)

Adds a node to the graph with a unique id and a label.

Parameters

uid	A unique id
label	The label to go inside the node in the graph

5.100.3.13 void Visualizer::addNode (int *parentid,* int *childid,* string *parent_label*)

Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.

Parameters

parentid	A unique id for the new node
childid	The unique id of the child node to create an edge to
parent_label	The label to go inside the node in the graph

5.100.3.14 void Visualizer::addNode (int *parentid,* int *childid,* string *parent_label*)

Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.

Parameters

parentid	A unique id for the new node
childid	The unique id of the child node to create an edge to
parent_label	The label to go inside the node in the graph

5.100.3.15 void Visualizer::addNode (int *parentid,* int *childid,* string *parent_label*)

Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.

Parameters

parentid	A unique id for the new node
childid	The unique id of the child node to create an edge to
parent_label	The label to go inside the node in the graph

Definition at line 56 of file Visualizer.cpp.

5.100.3.16 void Visualizer::addNode (int *parentid,* int *childid,* string *parent_label*)

Adds a node to the graph with a unique id and a label, then creates an edge from the new node to a child node.

Parameters

parentid	A unique id for the new node
childid	The unique id of the child node to create an edge to

parent_label	The label to go inside the node in the graph

5.100.3.17 static int Visualizer::GetNextUID ( ) [inline], [static]

Gets a unique id to be used in adding nodes to the AST.

Returns

A unique integer valued id

Definition at line 92 of file Visualizer.h.

5.100.3.18 static int Visualizer::GetNextUID ( ) [inline], [static]

Gets a unique id to be used in adding nodes to the AST.

Returns

A unique integer valued id

Definition at line 93 of file CParser.yy.

5.100.3.19 static int Visualizer::GetNextUID ( ) [inline], [static]

Gets a unique id to be used in adding nodes to the AST.

Returns

A unique integer valued id

Definition at line 93 of file CScanner.ll.

5.100.3.20 static int Visualizer::GetNextUID ( ) [inline], [static]

Gets a unique id to be used in adding nodes to the AST.

Returns

A unique integer valued id

Definition at line 93 of file CParser.yy.

The documentation for this class was generated from the following files:

- Visualizer.h
- Visualizer.cpp

Chapter 6

File Documentation

6.1 CParser.yy File Reference

NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATED DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE BISON FILES!

```
#include <string>
#include <sstream>
#include <iostream>
#include "SymTab.h"
#include "Ast.h"
#include "CCompiler.h"
```

Classes

- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType

- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class Visualizer

    A class for visualizing the generation of the AST.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- class AVLTree< DataItem >

    An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

    A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class TAC_Generator

    A class for generating three address code.
- class AST

    Abstract syntax tree node type.
- class AstTypeName
- class AstString
- class AstConstant
- class AstUnaryOp
- class AstID
- class AstPrimaryExpr
- class AstArgExprList
- class AstPostfixExpr
- class AstUnaryExpr
- class AstCastExpr
- class AstMultExpr
- class AstAddExpr
- class AstShiftExpr
- class AstRelExpr
- class AstEqExpr

- class AstAndExpr
- class AstXORExpr
- class AstORExpr
- class AstLogicAndExpr
- class AstLogicOrExpr
- class AstConditionalExpr
- class AstConstantExpr
- class AstAssignOp
- class AstAssignExpr
- class AstExpression
- class AstReturn
- class AstContinue
- class AstBreak
- class AstGoto
- class AstJump
- class AstDoWhile
- class AstWhile
- class AstFor
- class AstIteration
- class AstSwitch
- class AstIfElse
- class AstSelection
- class AstCompoundStmt
- class AstExprStmt
- class AstLabeledStmt
- class AstStatement
- class AstStructUniSpeci
- class AstStructDeclatorList
- class AstStructDeclarator
- class AstPointer
- class AstParamList
- class AstParamDec
- class AstInitList
- class AstIDList
- class AstDeclarationList
- class AstTypeSpeci
- class AstTypeQualList
- class AstTrans
- class AstStructDeclList
- class AstStatementList
- class AstDirectDecl
- class AstDecSpeci
- class AstDeclList
- class AstDecl
- class AstAbstractDecl
- class AstTypeParamList
- class AstDirectAbsDecl
- class AstInitDeclList
- class AstInitDeclarator
- class AstInitializer
- class AstEnumerator
- class AstEnumList
- class EnumSpecifier
- class AstSpeciQualList
- class AstStructDecl

- class AstDeclarator
- class AstFuncDef
- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class Visualizer

  A class for visualizing the generation of the AST.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType

- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class TAC_Generator

    A class for generating three address code.
- class AST

    Abstract syntax tree node type.
- class AstTypeName
- class AstString
- class AstConstant
- class AstUnaryOp
- class AstID
- class AstPrimaryExpr
- class AstArgExprList
- class AstPostfixExpr
- class AstUnaryExpr
- class AstCastExpr
- class AstMultExpr
- class AstAddExpr
- class AstShiftExpr
- class AstRelExpr
- class AstEqExpr
- class AstAndExpr
- class AstXORExpr
- class AstORExpr
- class AstLogicAndExpr
- class AstLogicOrExpr
- class AstConditionalExpr
- class AstConstantExpr
- class AstAssignOp
- class AstAssignExpr
- class AstExpression
- class AstReturn
- class AstContinue
- class AstBreak
- class AstGoto
- class AstJump
- class AstDoWhile
- class AstWhile
- class AstFor
- class AstIteration
- class AstSwitch
- class AstIfElse
- class AstSelection
- class AstCompoundStmt
- class AstExprStmt
- class AstLabeledStmt
- class AstStatement
- class AstStructUniSpeci
- class AstStructDeclatorList

- class AstStructDeclarator
- class AstPointer
- class AstParamList
- class AstParamDec
- class AstInitList
- class AstIDList
- class AstDeclarationList
- class AstTypeSpeci
- class AstTypeQualList
- class AstTrans
- class AstStructDeclList
- class AstStatementList
- class AstDirectDecl
- class AstDecSpeci
- class AstDeclList
- class AstDecl
- class AstAbstractDecl
- class AstTypeParamList
- class AstDirectAbsDecl
- class AstInitDeclList
- class AstInitDeclarator
- class AstInitializer
- class AstEnumerator
- class AstEnumList
- class EnumSpecifier
- class AstSpeciQualList
- class AstStructDecl
- class AstDeclarator
- class AstFuncDef
- struct InputLine

  A structure for buffering lines input code.
- class CCompiler

  A minimalist C programming language compiler class.

Enumerations

- enum **CONVERSIONTYPE** {
  NONE, **NONE**, **NONE**, **NONE**,
  NONE, **NONE**, **NONE**, **NONE**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
  INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
  INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
  SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
  SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
  SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
  SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
  LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
  LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
  LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
  LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
  FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
  FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
  FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
  FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
  FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
  FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
  DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
  DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
  DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
  DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
  DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
  DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
  NONE, **NONE**, **NONE**, **NONE**,
  NONE, **NONE**, **NONE**, **NONE**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
  INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
  INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
  SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
  SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
  SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
  SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
  LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
  LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
  LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
  LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
  FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
  FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
  FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
  FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
  FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
  FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
  DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
  DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
  DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
  DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
  DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
  DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
  NONE, **NONE**, **NONE**, **NONE**,
  NONE, **NONE**, **NONE**, **NONE**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
  INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,

FLOAT, **DOUBLE**, **CHAR** }

- enum

  Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,
STATIC, STATIC, STATIC, STATIC,
STATIC, STATIC, STATIC, STATIC,
EXTERN, EXTERN, EXTERN, EXTERN,
EXTERN, EXTERN, EXTERN, EXTERN,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
TYPEDEF, TYPEDEF, TYPEDEF, TYPEDEF,
AUTO, AUTO, AUTO, AUTO,
AUTO, AUTO, AUTO, AUTO,
REGISTER, REGISTER, REGISTER, REGISTER,
REGISTER, REGISTER, REGISTER, REGISTER,

TYPEDEF }

- enum **TypeQualifier** {
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

- enum

Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,

FLOAT, **DOUBLE**, **CHAR** }

- enum

  Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

```
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
```

FLOAT, **DOUBLE**, **CHAR** }

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **CONST**, **CONST**, **CONST**,
  VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
  CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,

FLOAT, **DOUBLE**, **CHAR** }

- enum

Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

Functions

- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- void **GenGlobals** (SymTab &symTab, TAC_Generator &tacGen)
- void **VisVist** (int total,...)
- void **VisAddIntNode** (AST ∗parentAst, int number)
- void **VisAddStringNode** (AST ∗parentASt, string &node)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- void **GenGlobals** (SymTab &symTab, TAC_Generator &tacGen)
- **if** (driver.currentSymbol->symbolType==NULL) driver.currentSymbol-> symbolType=new PODType("VOID", INT_SIZE)
- newPod **SetSigned** (true)
- driver **printRed** ("type_specifier -> SIGNED")
- newPod **SetSigned** (false)
- driver **printRed** ("type_specifier -> UNSIGNED")
- **if** (driver.structUnionMode) driver.structUnionTypes.push_back(∗inf)
- driver **allocateSymbol** ()
- **if** (inf->symbolType->GetName()=="POINTER"||inf->symbolType->GetName()=="ARRAY")
- while (endItem!=driver.structUnionTypes.begin()&&(endItem) ->symbolType->GetName()!="STRUCT")
- **while** (count)
- **for** (int count=0;count< driver.structVarCount;count++) itemFixStart--
- **while** (itemFixStart!=driver.structUnionTypes.end())
- driver **printRed** ("struct_declarator_list -> struct_declarator")
- driver **printRed** ("struct_declarator_list -> struct_declarator_list COMMA struct_declarator")
- driver **printRed** ("struct_declarator -> declarator")
- driver **printRed** ("struct_declarator -> COLON constant_expression")
- driver **printRed** ("struct_declarator -> declarator COLON constant_expression")
- driver **printRed** ("enum_specifier -> ENUM LBRACE enumerator_list RBRACE")
- **while** (enumItems!=driver.enumConsts.end())
- driver enumConsts **clear** ()
- driver **printRed** ("enum_specifier -> ENUM identifier LBRACE enumerator_list RBRACE")
- **if** ($$->needsCast) driver.error(yyloc = (AST∗) new AstLogicOrExpr((AstLogicOrExpr∗)$1, (AstLogicAndExpr∗)$3)
- driver **error** (yylloc,"Cast expressions are not currently supported by this compiler!")
- driver **printRed** ("identifier -> IDENTIFIER")
- **if** (driver.SymbolTable.find_symbol(s, level))

Variables

- require define
  parser_class_name CParser code **requires**
- parse **param**

- union {
    int **ival**
    double **dval**
    std::string ∗ **sval**
    SymbolInfo ∗ **sym**
    AST ∗ **ast**
  };

- **printer** { driver.ydbFile << "Value: " << $$

- < sym > **destructor** { if(!$$) delete $$

- < sval >< sym > **code**

- token END EOF token< sym >

IDENTIFIER token< ival >
INTEGER_CONSTANT token< dval >
FLOATING_CONSTANT token< sval >
CHARACTER_CONSTANT token< sym >
ENUMERATION_CONSTANT token
< sval > STRING_LITERAL token
< sym > TYPEDEF_NAME token
SIZEOF token PTR_OP token
INC_OP DEC_OP token LEFT_OP
RIGHT_OP token LE_OP GE_OP
EQ_OP NE_OP token AND_OP OR_OP
token MUL_ASSIGN DIV_ASSIGN
MOD_ASSIGN ADD_ASSIGN
SUB_ASSIGN token LEFT_ASSIGN
RIGHT_ASSIGN AND_ASSIGN
XOR_ASSIGN OR_ASSIGN token
STAR BIN_OR BIN_XOR BIN_AND
token LT_OP GT_OP PLUS MINUS
DIV MOD TILDE BANG DOT WHAT
token LBRACE RBRACE LBRAK
RBRAK LPAREN RPAREN EQ COMMA
COLON SEMI token TYPEDEF
EXTERN STATIC AUTO REGISTER
token CHAR SHORT INT LONG
SIGNED UNSIGNED FLOAT DOUBLE
CONST VOLATILE VOID token
STRUCT UNION ENUM ELIPSIS
token CASE DEFAULT IF ELSE
SWITCH WHILE DO FOR GOTO
CONTINUE BREAK RETURN type
< ast > identifier string
constant
argument_expression_list
primary_expression type< ast >
postfix_expression
unary_operator
unary_expression type< ast >
multiplicative_expression
additive_expression
shift_expression type< ast >
relational_expression
equality_expression
and_expression type< ast >
exclusive_or_expression
inclusive_or_expression
logical_and_expression
logical_or_expression type
< ast > constant_expression
conditional_expression
assignment_operator
assignment_expression type
< ast > expression
cast_expression type_name type
< ast > jump_statement
iteration_statement statement
selection_statement
statement_list
compound_statement
declaration_list type< ast >
expression_statement
labeled_statement
direct_abstract_declarator
parameter_type_list
abstract_declarator type< ast >

struct_declarator **translation_unit**
- driver **source_ast** = $$
- translation_unit **external_declaration**
- enter_scope **__pad0__**
- ss<< "Entering new scope: line - "
  << @$.begin.line<< ", col - "
  << @$.begin.column;driver.printDebug(ss.str());driver.enterScope();};leave_scope:{std::stringstream
  ss;ss<< "Leaving scope: line - "
  << @$.begin.line<< ", col - "
  << @$.begin.column;driver.printDebug(ss.str());driver.leaveScope();};insert_mode:{std::stringstream
  ss;ss<< "Starting insert mode:
  line - "<< @$.begin.line
  << ", col - "
  << @$.begin.column;driver.printDebug(ss.str());driver.set_insert_mode(true);};lookup_mode:{std::stringstream
  ss;ss<< "Starting lookup mode:
  line - "<< @$.begin.line
  << ", col - "
  << @$.begin.column;driver.printDebug(ss.str());driver.set_insert_mode(false);};external_declaration-
  :function_definition{driver.printRed("exterfunction_definitionnal_declaration -> **function_declaration**
- **declaration**
- function_definition **__pad1__** = (AST ∗)$1
- declarator set_function_name
  declaration_list **compound_statement**
- set_function_name **__pad2__** = (AST ∗) new AstFuncDef ((AstDeclarator ∗) $2 , (AstCompound ∗) $5 ,(AstDeclList ∗)$4, (AstDecSpeci∗)$1)
- cout<<"Current function name = "
  << driver.currentFunctionName
  << endl;};declaration:declaration_specifiers
  SEMI{driver.printDebug("Declaration
  .... ");driver.printRed("declaration ->
  declaration_specifiers **SEMI**
- SymbolInfo ∗ **inf** = driver.currentSymbol
- declaration_list **__pad3__** = (AST ∗) new AstDecl ((AstDecSpeci ∗)$1 , (AstInitSpecList ∗)$2)
- declaration_specifiers **__pad4__** = (AST ∗) new AstDeclList((AstDeclList ∗)$1 ,(AstDecl ∗) $2)
- storage_class_specifier **declaration_specifiers**
- **type_specifier**
- **type_qualifier**
- storage_class_specifier **__pad5__** = (AST ∗) new AstDecSpeci("",driver.currentTypeQual ,(AstDecSpeci ∗)$2,NULL)
- driver currentSymbol **storage_class** = AUTO
- **REGISTER**
- driver **currentStorageType** = "REGISTER"
- **STATIC**
- **EXTERN**
- **TYPEDEF**
- type_specifier **__pad6__**
- **CHAR**
- **SHORT**
- **INT**
- **LONG**
- **FLOAT**
- **DOUBLE**
- SIGNED

  else ///std::cout<< driver.currentSymbol->symbolType->GetName();
- driver currentSymbol **symbolType** = newPod
- **UNSIGNED**

- **struct_or_union_specifier**
- **enum_specifier**
- **TYPEDEF_NAME**
- type_qualifier **__pad7__**
- driver **currentTypeQual** = "CONST"
- **VOLATILE**
- struct_or_union_specifier **__pad8__**
- struct_or_union LBRACE
  struct_declaration_list **RBRACE**
- struct_or_union **identifier**
- struct_or_union **__pad9__** = (AST ∗)new AstStructUniSpeci("STRUCT" , (AstID ∗) $2 , NULL)
- driver **structUnionMode**
- **UNION**
- struct_declaration_list **__pad10__**
- struct_declaration_list **struct_declaration**
- init_declarator_list **__pad11__** = (AST ∗)new AstStructDeclList((AstStructDecl ∗) $2 , (AstStructDeclList ∗) $1)
- init_declarator_list COMMA
  new_symbol_declaration **init_declarator**
- new_symbol_declaration **__pad12__** = (AST ∗)new AstInitDeclList((AstInitDeclarator ∗)$4 , (AstInitDeclList ∗) $1)
- **else**
- new_enum_constant **__pad13__**
- reset_current_symbol **__pad14__**
- new_struct_union_decl **__pad15__**
- struct_union_decl_end **__pad16__**
- list< SymbolInfo >::iterator **endItem** = driver.structUnionTypes.end()
- list< SymbolInfo >::iterator **structPos** = endItem
- SymbolInfo **currentStruct**
- StructType ∗ **currentStructType**
- int **count** = 0
- init_declarator **__pad17__**
- declarator EQ **initializer**
- struct_declaration **__pad18__** = (AST ∗) new AstInitDeclarator((AstDeclarator ∗)$1 , (AstInitializer ∗)$3)
- set_member_type **__pad19__** = (AST ∗) new AstStructDecl ((AstSpeciQualList∗)$1 , (AstStructDeclList ∗)$3)
- fix_struct_member_types **__pad20__**
- Type ∗ **innerType**
- driver **structVarCount** = 0
- specifier_qualifier_list **__pad21__**
- type_specifier **specifier_qualifier_list**
- struct_declarator_list **__pad22__** = (AST ∗)new AstSpeciQualList (NULL , driver.currentTypeQual , (AstSpeciQualList∗)$2)
- struct_declarator_list **NULL**
- struct_declarator_list COMMA **struct_declarator**
- struct_declarator **__pad23__**
- COLON **constant_expression**
- driver structUnionTypes push_back **driver**
- enum_specifier **__pad24__**
- enum_specifier AstEnumList
  ∗list< string >::iterator **enumItems** = driver.enumConsts.begin()
- new_enum_type **__pad25__**
- driver **enumType** = new EnumType("ENUM", 0)
- driver enumSym **symbol_name** = driver.currentSymbol->symbol_name
- enumerator_list **__pad26__**

- enumerator_list COMMA **enumerator**
- enumerator **__pad27__** = (AST ∗) new AstEnumList ((AstEnumerator ∗) $3 , (AstEnumList ∗)$1)
- declarator **__pad28__** = (AST ∗) new AstEnumerator ((AstID ∗)$1 , (AstExpression ∗)$4)
- pointer **direct_declarator**
- direct_declarator **__pad29__** = (AST ∗) new AstDeclarator((AstPointer ∗)$1 , (AstDirectDecl ∗) $2)
- LPAREN declarator **RPAREN**
- direct_declarator LBRAK **RBRAK**
- pointer **__pad30__** = (AST ∗) new AstDirectDecl(NULL , (AstDirectDecl ∗) $1 , NULL , NULL , NULL, (AstIDList ∗)$3,7)
- STAR **type_qualifier_list**
- STAR **pointer**
- type_qualifier_list **__pad31__** = (AST ∗) new AstPointer((AstPointer ∗)$3 , (AstTypeQualList ∗)$2)
- parameter_type_list **__pad32__** = (AST ∗)new AstTypeQualList(driver.currentTypeQual,(AstTypeQualList ∗) $1)
- parameter_list COMMA **ELIPSIS**
- parameter_list **__pad33__** = (AST ∗) new AstTypeParamList (2 , (AstParamList ∗)$1)
- parameter_list COMMA **parameter_declaration**
- parameter_declaration **__pad34__** = (AST ∗) new AstParamList ((AstParamDec ∗)$3 , (AstParamList ∗)$3)
- declaration_specifiers **abstract_declarator**
- identifier_list **__pad35__** = (AST ∗) new AstParamDec((AstDecSpeci ∗)$1 , NULL , (AstAbstractDecl ∗) $2)
- initializer **__pad36__** = (AST ∗) new AstIDList ((AstID ∗)$3 , (AstIDList ∗) $1)
- initializer_list **__pad37__** = (AST ∗) new AstInitializer (NULL , (AstInitList ∗)$2 , 3)
- type_name **__pad38__** = (AST ∗) new AstInitList((AstInitializer ∗)$3 , (AstInitList ∗)$1)
- abstract_declarator **__pad39__** = (AST ∗) new AstTypeName((AstSpeciQualList ∗)$1 , (AstAbstractDecl ∗)$2)
- **direct_abstract_declarator**
- direct_abstract_declarator **__pad40__** = (AST ∗) new AstAbstractDecl((AstPointer ∗)$1 , (AstDirectAbsDecl ∗)$2)
- statement **__pad41__** = (AST ∗) new AstDirectAbsDecl (9, NULL , NULL , (AstDirectAbsDecl∗) $1 ,(AstTypeParamList ∗)$3)
- **expression_statement**
- **selection_statement**
- **iteration_statement**
- **jump_statement**
- labeled_statement **__pad42__** = (AST∗) new AstStatement((AstJump∗)$1)
- CASE constant_expression COLON **statement**
- expression_statement **__pad43__** = (AST∗) new AstLabeledStmt((AstStatement∗)$3)
- compound_statement **__pad44__** = (AST∗) new AstExprStmt((AstExpression∗)$1)
- statement_list **__pad45__** = (AST∗) new AstCompoundStmt((AstDeclarationList∗)$5, (AstStatementList∗)$7)
- selection_statement **__pad46__** = (AST∗) new AstStatementList((AstStatementList∗)$1, (AstStatement∗)$2)
- iteration_statement **__pad47__** = (AST∗) new AstSelection(new AstSwitch((AstExpression∗)$3, (AstStatement∗)$5))
- jump_statement **__pad48__** = (AST∗) new AstIteration(new AstFor((AstExpression∗)$3, (AstExpression∗)$5, (AstExpression∗)$7, (AstStatement∗)$9))
- expression **__pad49__** = (AST∗) new AstJump(new AstReturn(), (AstExpression∗)$2)
- expression COMMA **assignment_expression**
- assignment_expression **__pad50__** = (AST∗) new AstExpression((AstExpression∗)$1, (AstAssignExpr∗)$3)
- assignment_operator **__pad51__** = (AST∗) new AstAssignExpr((AstUnaryExpr∗)$1, (AstAssignOp∗)$2, (AstAssignExpr∗)$3)
- **MUL_ASSIGN**
- **DIV_ASSIGN**
- **MOD_ASSIGN**
- **ADD_ASSIGN**
- **SUB_ASSIGN**
- **LEFT_ASSIGN**

- **RIGHT_ASSIGN**
- **AND_ASSIGN**
- **XOR_ASSIGN**
- **OR_ASSIGN**
- conditional_expression **__pad52__** = (AST∗) new AstAssignOp(AstAssignOp::OR_ASSIGN)
- logical_or_expression WHAT
  expression COLON **conditional_expression**
- These types must **match**
- constant_expression **__pad53__**
- logical_or_expression **__pad54__** = (AST∗) new AstConstantExpr((AstConditionalExpr∗)$1)
- logical_or_expression OR_OP **logical_and_expression**
- These types cannot be implicitly **converted**
- logical_and_expression **__pad55__**
- logical_and_expression AND_OP **inclusive_or_expression**
- inclusive_or_expression **__pad56__**
- inclusive_or_expression BIN_OR **exclusive_or_expression**
- exclusive_or_expression **__pad57__**
- exclusive_or_expression BIN_XOR **and_expression**
- and_expression **__pad58__**
- and_expression BIN_AND **equality_expression**
- equality_expression **__pad59__**
- equality_expression EQ_OP **relational_expression**
- relational_expression **__pad60__**
- relational_expression LT_OP **shift_expression**
- shift_expression **__pad61__**
- shift_expression LEFT_OP **additive_expression**
- additive_expression **__pad62__**
- additive_expression PLUS **multiplicative_expression**
- multiplicative_expression **__pad63__**
- multiplicative_expression STAR **cast_expression**
- cast_expression **__pad64__**
- unary_expression **__pad65__** = (AST∗) new AstCastExpr((AstTypeName∗)$2, (AstCastExpr∗)$4)
- INC_OP **unary_expression**
- unary_operator **__pad66__** = (AST∗) new AstUnaryExpr((AstTypeName∗)$3)
- **STAR**
- **PLUS**
- **MINUS**
- **TILDE**
- **BANG**
- postfix_expression **__pad67__** = (AST∗) new AstUnaryOp(AstUnaryOp::BANG)
- postfix_expression **INC_OP**
- postfix_expression **DEC_OP**
- primary_expression **__pad68__** = (AST∗) new AstPostfixExpr((AstPostfixExpr∗)$1, AstPostfixExpr::DEC_O-P)
- **constant**
- **string**
- argument_expression_list **__pad69__** = (AST∗) new AstPrimaryExpr((AstExpression∗)$2)
- constant **__pad70__** = (AST∗) new AstArgExprList((AstArgExprList∗)$1, (AstAssignExpr∗)$3)
- **CHARACTER_CONSTANT**
- **FLOATING_CONSTANT**
- **ENUMERATION_CONSTANT**
- string **__pad71__**
- identifier **__pad72__** = (AST∗) new AstString(∗$1, new Type("STRING_LITERAL", 0))
- int **level**
- SymbolInfo **s**

6.1.1 Detailed Description

NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATED DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE BISON FILES!

Definition in file CParser.yy.

6.1.2 Function Documentation

6.1.2.1 while (endItem! = driver.structUnionTypes.begin() && (endItem)->symbolType->GetName() !=)

currentStructType->AddMember(endItem->symbol_name,new PODType("INT",INT_SIZE));

Definition at line 543 of file CParser.yy.

6.1.3 Variable Documentation

6.1.3.1 string __pad71__

Initial value:

```
= (AST*) new AstConstant(
                    ((EnumType*)$1->symbolType)->GetConstVal($1->symbol_name),
                    $1->symbol_name,
                    $1->symbolType
                 )
```

Definition at line 1787 of file CParser.yy.

6.1.3.2 specifier_qualifier_list abstract_declarator

Initial value:

```
{
            driver.printRed("parameter_declaration -> declaration_specifiers abstract_declarator")
```

Definition at line 956 of file CParser.yy.

6.1.3.3 ADD_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> ADD_ASSIGN")
```

Definition at line 1336 of file CParser.yy.

6.1.3.4 shift_expression RIGHT_OP additive_expression

Initial value:

```
{
            driver.printRed("shift_expression -> shift_expression LEFT_OP additive_expression")
```

Definition at line 1536 of file CParser.yy.

6.1.3.5 AND_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> AND_ASSIGN")
```

Definition at line 1356 of file CParser.yy.

6.1.3.6 exclusive_or_expression BIN_XOR and_expression

Initial value:

```
{
            driver.printRed("exclusive_or_expression -> exclusive_or_expression BIN_XOR and_expression")
```

Definition at line 1448 of file CParser.yy.

6.1.3.7 argument_expression_list COMMA assignment_expression

Initial value:

```
{
            driver.printRed("expression -> expression COMMA assignment_expression")
```

Definition at line 1295 of file CParser.yy.

6.1.3.8 BANG

Initial value:

```
{
            driver.printRed("unary_operator -> BANG")
```

Definition at line 1678 of file CParser.yy.

6.1.3.9 unary_operator cast_expression

Initial value:

```
{
            driver.printRed("multiplicative_expression -> multiplicative_expression STAR cast_expression")
```

Definition at line 1580 of file CParser.yy.

6.1.3.10 CHAR

Initial value:

```
{
            driver.printRed("type_specifier -> CHAR")
```

Definition at line 321 of file CParser.yy.

6.1.3.11 CHARACTER_CONSTANT

Initial value:

```
{
            driver.printRed("constant -> CHARACTER_CONSTANT")
```

Definition at line 1775 of file CParser.yy.

6.1.3.12 declaration_specifiers declarator set_function_name declaration_list compound_statement

Initial value:

```
{
            driver.printRed("function_definition -> declarator declaration_list compound_statement")
```

Definition at line 170 of file CParser.yy.

6.1.3.13 logical_or_expression WHAT expression COLON conditional_expression

Initial value:

```
{
            driver.printRed("conditional_expresion -> logical_or_expression WHAT expression COLON
       conditional_expression")
```

Definition at line 1379 of file CParser.yy.

6.1.3.14 constant

Initial value:

```
{
            driver.printRed("primary_expression -> constant")
```

Definition at line 1736 of file CParser.yy.

6.1.3.15 identifier new_enum_constant EQ constant_expression

Initial value:

```
{
            $$ = (AST *) new AstStructDeclarator( NULL , (
      AstExpression *)$2)
```

Definition at line 707 of file CParser.yy.

6.1.3.16 postfix_expression DEC_OP

Initial value:

```
{
            driver.printRed("postfix_expression -> postfix_expression DEC_OP")
```

Definition at line 1723 of file CParser.yy.

6.1.3.17 declaration_list declaration

Initial value:

```
{
            driver.printRed("external_declaration -> declaration")
```

Definition at line 156 of file CParser.yy.

6.1.3.18 type_qualifier declaration_specifiers

Initial value:

```
{
            driver.printRed("declaration_specifiers -> storage_class_specifier declaration_specifiers")
```

Definition at line 247 of file CParser.yy.

6.1.3.19 pointer direct_abstract_declarator

Initial value:

```
{
            driver.printRed("abstract_declarator -> direct_abstract_declarator")
```

Definition at line 1027 of file CParser.yy.

6.1.3.20 pointer direct_declarator

Initial value:

```
{
            driver.printRed("declarator -> pointer direct_declarator")
```

Definition at line 822 of file CParser.yy.

6.1.3.21 DIV_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> DIV_ASSIGN")
```

Definition at line 1326 of file CParser.yy.

6.1.3.22 DOUBLE

Initial value:

```
{
            driver.printRed("type_specifier -> DOUBLE")
```

Definition at line 358 of file CParser.yy.

6.1.3.23 parameter_list COMMA ELIPSIS

Initial value:

```
{
            driver.printRed("parameter_type_list -> parameter_list COMMA ELIPSIS")
```

Definition at line 925 of file CParser.yy.

6.1.3.24 else

Initial value:

```
{
              if ( inf->storage_class == TYPEDEF)
              {
                   inf->symbolType = new TypedefType(inf->symbolType,inf->symbol_name);

              }

              driver.SymbolTable.insert_symbol(*inf)
```

Definition at line 483 of file CParser.yy.

6.1.3.25 enum_specifier

Initial value:

```
{
            driver.printRed("type_specifier -> enum_specifier")
```

Definition at line 392 of file CParser.yy.

6.1.3.26 ENUMERATION_CONSTANT

Initial value:

```
{
            driver.printRed("constant -> ENUMERATION_CONSTANT")
```

Definition at line 1785 of file CParser.yy.

6.1.3.27 enumerator_list COMMA enumerator

Initial value:

```
{
            driver.printRed("enumerator_list -> enumerator_list COMMA enumerator")
```

Definition at line 795 of file CParser.yy.

6.1.3.28 and_expression BIN_AND equality_expression

Initial value:

```
{
            driver.printRed("and_expression -> and_expression BIN_AND equality_expression")
```

Definition at line 1463 of file CParser.yy.

6.1.3.29 inclusive_or_expression BIN_OR exclusive_or_expression

Initial value:

```
{
            driver.printRed("inclusive_or_expression -> inclusive_or_expression BIN_OR
       exclusive_or_expression")
```

Definition at line 1433 of file CParser.yy.

6.1.3.30 expression_statement

Initial value:

```
{
            driver.printRed("statement -> expression_statement")
```

Definition at line 1100 of file CParser.yy.

6.1.3.31 EXTERN

Initial value:

```
{
            driver.printRed("storage_class_specifier -> EXTERN")
```

Definition at line 294 of file CParser.yy.

6.1.3.32 translation_unit external_declaration

Initial value:

```
{
            driver.printRed("translation_unit -> translation_unit external_declaration")
```

Definition at line 96 of file CParser.yy.

6.1.3.33 FLOAT

Initial value:

```
{
            driver.printRed("type_specifier -> FLOAT")
```

Definition at line 350 of file CParser.yy.

6.1.3.34 FLOATING_CONSTANT

Initial value:

```
{
            driver.printRed("constant -> FLOATING_CONSTANT")
```

Definition at line 1780 of file CParser.yy.

6.1.3.35 postfix_expression PTR_OP identifier

Initial value:

```
{
            driver.printRed("struct_or_union_specifier -> struct_or_union identifier")
```

Definition at line 431 of file CParser.yy.

6.1.3.36 postfix_expression INC_OP

Initial value:

```
{
            driver.printRed("postfix_expression -> INC_OP")
```

Definition at line 1718 of file CParser.yy.

6.1.3.37 logical_and_expression AND_OP inclusive_or_expression

Initial value:

```
{
            driver.printRed("logical_and_expression -> logical_and_expression AND_OP
       inclusive_or_expression")
```

Definition at line 1418 of file CParser.yy.

6.1.3.38 init_declarator_list COMMA new_symbol_declaration init_declarator

Initial value:

```
{
            driver.printRed("init_declarator_list COMMA init_declarator")
```

Definition at line 471 of file CParser.yy.

6.1.3.39 initializer_list COMMA initializer

Initial value:

```
{
            driver.printRed("init_declarator -> declarator EQ initializer")
```

Definition at line 598 of file CParser.yy.

6.1.3.40 INT

Initial value:

```
{
            driver.printRed("type_specifier -> INT")
```

Definition at line 335 of file CParser.yy.

6.1.3.41 iteration_statement

Initial value:

```
{
            driver.printRed("statement -> iteration_statement")
```

Definition at line 1111 of file CParser.yy.

6.1.3.42 jump_statement

Initial value:

```
{
            driver.printRed("statement -> jump_statement")
```

Definition at line 1116 of file CParser.yy.

6.1.3.43 LEFT_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> LEFT_ASSIGN")
```

Definition at line 1346 of file CParser.yy.

6.1.3.44 logical_or_expression OR_OP logical_and_expression

Initial value:

```
{
            driver.printRed("logical_or_expression -> logical_or_epression OR_OP logical_and_expression")
```

Definition at line 1403 of file CParser.yy.

6.1.3.45 LONG

Initial value:

```
{
            driver.printRed("type_specifier -> LONG")
```

Definition at line 343 of file CParser.yy.

6.1.3.46 MINUS

Initial value:

```
{
            driver.printRed("unary_operator -> MINUS")
```

Definition at line 1668 of file CParser.yy.

6.1.3.47 MOD_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> MOD_ASSIGN")
```

Definition at line 1331 of file CParser.yy.

6.1.3.48 MUL_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> MUL_ASSIGN")
```

Definition at line 1321 of file CParser.yy.

6.1.3.49 additive_expression MINUS multiplicative_expression

Initial value:

```
{
            driver.printRed("additive_expression -> additive_expression PLUS multiplicative_expression")
```

Definition at line 1558 of file CParser.yy.

6.1.3.50 OR_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> OR_ASSIGN")
```

Definition at line 1366 of file CParser.yy.

6.1.3.51 parse param

Initial value:

```
{ CCompiler& driver }
%lex-param   { CCompiler& driver }

%initial-action
{

    @$.begin.filename = @$.end.filename = &driver.fname;
}
```

Definition at line 15 of file CParser.yy.

6.1.3.52 parameter_list COMMA parameter_declaration

Initial value:

```
{
            driver.printRed("parameter_list -> parameter_list COMMA parameter_declaration")
```

Definition at line 938 of file CParser.yy.

6.1.3.53 PLUS

Initial value:

```
{
            driver.printRed("unary_operator -> PLUS")
```

Definition at line 1663 of file CParser.yy.

6.1.3.54 STAR type_qualifier_list pointer

Initial value:

```
{
            driver.printRed("pointer -> STAR pointer")
```

Definition at line 889 of file CParser.yy.

6.1.3.55 LBRACE reset_current_symbol enter_scope insert_mode declaration_list lookup_mode statement_list leave_scope RBRACE

Initial value:

```
{
            driver.printRed("struct_or_union_specifier -> struct_or_union LBRACE struct_declaration_list
       RBRACE")
```

Definition at line 426 of file CParser.yy.

6.1.3.56 postfix_expression LBRAK expression RBRAK

Initial value:

```
{
            driver.printRed("direct_declarator -> LBRAK RBRAK")
```

Definition at line 843 of file CParser.yy.

6.1.3.57 REGISTER

Initial value:

```
{
            driver.printRed("storage_class_specifier -> REGISTER")
```

Definition at line 280 of file CParser.yy.

6.1.3.58 equality_expression NE_OP relational_expression

Initial value:

```
{
            driver.printRed("equality_expression -> equality_expression EQ_OP relational_expression")
```

Definition at line 1478 of file CParser.yy.

6.1.3.59 RIGHT_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> RIGHT_ASSIGN")
```

Definition at line 1351 of file CParser.yy.

6.1.3.60 LPAREN expression RPAREN

Initial value:

```
{
            driver.printRed("direct_declarator -> LPAREN declarator RPAREN")
```

Definition at line 838 of file CParser.yy.

6.1.3.61 selection_statement

Initial value:

```
{
            driver.printRed("statement -> selection_statement")
```

Definition at line 1106 of file CParser.yy.

6.1.3.62 RETURN expression SEMI

Initial value:

```
{
            driver.printRed("declaration -> declaration_specifiers init_declarator_list SEMI")
```

Definition at line 197 of file CParser.yy.

6.1.3.63 relational_expression GE_OP shift_expression

Initial value:

```
{
            driver.printRed("relational_expression -> relational_expression LT_OP shift_expression")
```

Definition at line 1500 of file CParser.yy.

6.1.3.64 SHORT

Initial value:

```
{
            driver.printRed("type_specifier -> SHORT")
```

Definition at line 328 of file CParser.yy.

6.1.3.65 SIGNED

Initial value:

```
{
            PODType *newPod = new PODType("INT",INT_SIZE)
```

else ///std::cout<< driver.currentSymbol->symbolType->GetName();

Definition at line 368 of file CParser.yy.

6.1.3.66 type_qualifier specifier_qualifier_list

Initial value:

```
{
            driver.printRed("specifier_qualifier_list -> type_specifier specifier_qualifier_list")
```

Definition at line 667 of file CParser.yy.

6.1.3.67 STAR

Initial value:

```
{
            driver.printRed("unary_operator -> STAR")
```

Definition at line 1658 of file CParser.yy.

6.1.3.68 FOR LPAREN expression SEMI expression SEMI expression RPAREN statement

Initial value:

```
{
            driver.printRed("labeled_statement -> CASE constant_expression COLON statement")
```

Definition at line 1129 of file CParser.yy.

6.1.3.69 STATIC

Initial value:

```
{
            driver.printRed("storage_class_specifier -> STATIC")
```

Definition at line 287 of file CParser.yy.

6.1.3.70 string

Initial value:

```
{
            driver.printRed("primary_expression -> string")
```

Definition at line 1741 of file CParser.yy.

6.1.3.71 struct_declaration_list struct_declaration

Initial value:

```
{
            driver.printRed("struct_declaration_list -> struct_declaration_list struct_declaration")
```

Definition at line 458 of file CParser.yy.

6.1.3.72 struct_declarator_list COMMA struct_declarator

Initial value:

```
{
            $$ = (AST *)  new AstStructDeclatorList ( (
      AstStructDeclarator *)$3 , ( AstStructDeclList *) $1 )
```

Definition at line 689 of file CParser.yy.

6.1.3.73 struct_or_union_specifier

Initial value:

```
{
            driver.printRed("type_specifier -> struct_or_union_specifier")
```

Definition at line 387 of file CParser.yy.

6.1.3.74 SUB_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> SUB_ASSIGN")
```

Definition at line 1341 of file CParser.yy.

6.1.3.75 TILDE

Initial value:

```
{
            driver.printRed("unary_operator -> TILDE")
```

Definition at line 1673 of file CParser.yy.

6.1.3.76 type_qualifier_list type_qualifier

Initial value:

```
{
            driver.printRed("declaration_specifiers -> type_qualifier")
```

Definition at line 262 of file CParser.yy.

6.1.3.77 STAR type_qualifier_list

Initial value:

```
{
            driver.printRed("pointer -> STAR type_qualifier_list")
```

Definition at line 881 of file CParser.yy.

6.1.3.78 type_specifier

Initial value:

```
{
            driver.printRed("declaration_specifiers -> type_specifier")
```

Definition at line 252 of file CParser.yy.

6.1.3.79 TYPEDEF

Initial value:

```
{
            driver.printRed("storage_class_specifier -> TYPEDEF")
```

Definition at line 301 of file CParser.yy.

6.1.3.80 TYPEDEF_NAME

Initial value:

```
{
            driver.printRed("type_specifier -> TYPEDEF_NAME")
```

Definition at line 396 of file CParser.yy.

6.1.3.81 SIZEOF unary_expression

Initial value:

```
{
            driver.printRed("unary_expression -> INC_OP unary_expression")
```

Definition at line 1625 of file CParser.yy.

6.1.3.82 UNION

Initial value:

```
{
            driver.printRed("struct_or_union -> UNION")
```

Definition at line 445 of file CParser.yy.

6.1.3.83 UNSIGNED

Initial value:

```
{
            PODType *newPod = new PODType("INT",INT_SIZE)
```

Definition at line 378 of file CParser.yy.

6.1.3.84 VOLATILE

Initial value:

```
{
            driver.printRed("type_qualifier -> VOLATILE")
```

Definition at line 409 of file CParser.yy.

6.1.3.85 XOR_ASSIGN

Initial value:

```
{
            driver.printRed("assignment_operator -> XOR_ASSIGN")
```

Definition at line 1361 of file CParser.yy.

6.2 CScanner.ll File Reference

NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATED DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE FLEX FILES!

```
#include <cstdlib>
#include <limits>
#include <string>
#include <stdexcept>
#include <cerrno>
#include <iostream>
#include "CParser.hpp"
#include "CCompiler.h"
```

Classes

- class AVLTree< DataItem >

  An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

  A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType

- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class AVLTree< DataItem >

    An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

    A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class Visualizer

    A class for visualizing the generation of the AST.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- class AVLTree< DataItem >

    An implementation of a balanced binary tree called an AVL tree.
- struct AVLTree< DataItem >::Node

    A node which composes the DataItem template class with pointers to its children nodes in the AVL tree and the balance factor at the current node.
- class Type
- class PODType
- class TypedefType
- class EnumType
- class ArrayType
- class StructType
- class UnionType
- class FunctionType
- class PointerType
- struct SymbolInfo
- class SymTab
- class TAC_Generator

    A class for generating three address code.
- class AST

    Abstract syntax tree node type.
- class AstTypeName

- class AstString
- class AstConstant
- class AstUnaryOp
- class AstID
- class AstPrimaryExpr
- class AstArgExprList
- class AstPostfixExpr
- class AstUnaryExpr
- class AstCastExpr
- class AstMultExpr
- class AstAddExpr
- class AstShiftExpr
- class AstRelExpr
- class AstEqExpr
- class AstAndExpr
- class AstXORExpr
- class AstORExpr
- class AstLogicAndExpr
- class AstLogicOrExpr
- class AstConditionalExpr
- class AstConstantExpr
- class AstAssignOp
- class AstAssignExpr
- class AstExpression
- class AstReturn
- class AstContinue
- class AstBreak
- class AstGoto
- class AstJump
- class AstDoWhile
- class AstWhile
- class AstFor
- class AstIteration
- class AstSwitch
- class AstIfElse
- class AstSelection
- class AstCompoundStmt
- class AstExprStmt
- class AstLabeledStmt
- class AstStatement
- class AstStructUniSpeci
- class AstStructDeclatorList
- class AstStructDeclarator
- class AstPointer
- class AstParamList
- class AstParamDec
- class AstInitList
- class AstIDList
- class AstDeclarationList
- class AstTypeSpeci
- class AstTypeQualList
- class AstTrans
- class AstStructDeclList
- class AstStatementList
- class AstDirectDecl

- class AstDecSpeci
- class AstDeclList
- class AstDecl
- class AstAbstractDecl
- class AstTypeParamList
- class AstDirectAbsDecl
- class AstInitDeclList
- class AstInitDeclarator
- class AstInitializer
- class AstEnumerator
- class AstEnumList
- class EnumSpecifier
- class AstSpeciQualList
- class AstStructDecl
- class AstDeclarator
- class AstFuncDef
- struct InputLine

  A structure for buffering lines input code.
- class CCompiler

  A minimalist C programming language compiler class.

Macros

- #define **yywrap**() 1
- #define **yyterminate**() return token::END

Enumerations

- enum **CONVERSIONTYPE** {
NONE, **NONE**, **NONE**, **NONE**,
NONE, **NONE**, **NONE**, **NONE**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
NONE, **NONE**, **NONE**, **NONE**,
NONE, **NONE**, **NONE**, **NONE**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
INT2DBL, **INT2DBL**, **INT2DBL**, **INT2DBL**,
SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
SHT2FLT, **SHT2FLT**, **SHT2FLT**, **SHT2FLT**,
SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
SHT2DBL, **SHT2DBL**, **SHT2DBL**, **SHT2DBL**,
LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
LNG2FLT, **LNG2FLT**, **LNG2FLT**, **LNG2FLT**,
LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
LNG2DBL, **LNG2DBL**, **LNG2DBL**, **LNG2DBL**,
FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
FLT2INT, **FLT2INT**, **FLT2INT**, **FLT2INT**,
FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
FLT2SHT, **FLT2SHT**, **FLT2SHT**, **FLT2SHT**,
FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
FLT2LNG, **FLT2LNG**, **FLT2LNG**, **FLT2LNG**,
DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
DBL2INT, **DBL2INT**, **DBL2INT**, **DBL2INT**,
DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
DBL2SHT, **DBL2SHT**, **DBL2SHT**, **DBL2SHT**,
DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
DBL2LNG, **DBL2LNG**, **DBL2LNG**, **DBL2LNG**,
NONE, **NONE**, **NONE**, **NONE**,
NONE, **NONE**, **NONE**, **NONE**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
CHARFLAG, **CHARFLAG**, **CHARFLAG**, **CHARFLAG**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,
INT2FLT, **INT2FLT**, **INT2FLT**, **INT2FLT**,

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

```
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
FLOAT, FLOAT, FLOAT, FLOAT,
FLOAT, FLOAT, FLOAT, FLOAT,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
DOUBLE, DOUBLE, DOUBLE, DOUBLE,
CHAR, CHAR, CHAR, CHAR,
CHAR, CHAR, CHAR, CHAR,
VOID, VOID, VOID, VOID,
VOID, VOID, VOID, VOID,
INT, INT, INT, INT,
INT, INT, INT, INT,
SHORT, SHORT, SHORT, SHORT,
SHORT, SHORT, SHORT, SHORT,
LONG, LONG, LONG, LONG,
LONG, LONG, LONG, LONG,
```

FLOAT, **DOUBLE**, **CHAR** }

- enum

  Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,

FLOAT, **DOUBLE**, **CHAR** }

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

- enum

Enumeration for indicating the direction of the rotations on the AVL tree.

- enum **CONVERSIONTYPE** {

```
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2FLT, SHT2FLT, SHT2FLT, SHT2FLT,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
SHT2DBL, SHT2DBL, SHT2DBL, SHT2DBL,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2FLT, LNG2FLT, LNG2FLT, LNG2FLT,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
LNG2DBL, LNG2DBL, LNG2DBL, LNG2DBL,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2INT, FLT2INT, FLT2INT, FLT2INT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2SHT, FLT2SHT, FLT2SHT, FLT2SHT,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
FLT2LNG, FLT2LNG, FLT2LNG, FLT2LNG,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2INT, DBL2INT, DBL2INT, DBL2INT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2SHT, DBL2SHT, DBL2SHT, DBL2SHT,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
DBL2LNG, DBL2LNG, DBL2LNG, DBL2LNG,
NONE, NONE, NONE, NONE,
NONE, NONE, NONE, NONE,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
CHARFLAG, CHARFLAG, CHARFLAG, CHARFLAG,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2FLT, INT2FLT, INT2FLT, INT2FLT,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
INT2DBL, INT2DBL, INT2DBL, INT2DBL,
```

DBL2SHT, **DBL2LNG** }

- enum **StorageSpecifiers** {

AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
STATIC, **STATIC**, **STATIC**, **STATIC**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
EXTERN, **EXTERN**, **EXTERN**, **EXTERN**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
TYPEDEF, **TYPEDEF**, **TYPEDEF**, **TYPEDEF**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
AUTO, **AUTO**, **AUTO**, **AUTO**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
REGISTER, **REGISTER**, **REGISTER**, **REGISTER**,
STATIC, **STATIC**, **STATIC**, **STATIC**,

TYPEDEF }

- enum **TypeQualifier** {
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **CONST**, **CONST**, **CONST**,
VOLATILE, **VOLATILE**, **VOLATILE**, **VOLATILE**,
CONST, **VOLATILE** }

- enum **BaseTypes** {

VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
DOUBLE, **DOUBLE**, **DOUBLE**, **DOUBLE**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
CHAR, **CHAR**, **CHAR**, **CHAR**,
VOID, **VOID**, **VOID**, **VOID**,
VOID, **VOID**, **VOID**, **VOID**,
INT, **INT**, **INT**, **INT**,
INT, **INT**, **INT**, **INT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
SHORT, **SHORT**, **SHORT**, **SHORT**,
LONG, **LONG**, **LONG**, **LONG**,
LONG, **LONG**, **LONG**, **LONG**,
FLOAT, **FLOAT**, **FLOAT**, **FLOAT**,

FLOAT, **DOUBLE**, **CHAR** }

Functions

- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- Type ∗ **GetInnerType** (Type ∗arrayOrPointer)
- void **GenGlobals** (SymTab &symTab, TAC_Generator &tacGen)
- void **VisVist** (int total,...)
- void **VisAddIntNode** (AST ∗parentAst, int number)
- void **VisAddStringNode** (AST ∗parentASt, string &node)
- ws[\t\v\f] letter[a-zA-Z]
  digit[0-9] hexdig[0-9a-fA-F] octdig[0-7] **id** ({letter}|_)(

6.2.1 Detailed Description

NOTE: USE THE LINK BELOW TO VIEW THE SOURCE FOR THIS FILE, THE GENERATED DOCUMENTATION IS NOT VALID SINCE DOXYGEN CANNOT PROPERLY PARSE FLEX FILES!

Definition in file CScanner.ll.

6.3 main.cpp File Reference

This file serves as an entry point to the compiler.

```
#include <iostream>
#include "CCompiler.h"
```

Functions

- void **usage** (char ∗∗argv)
- int **main** (int argc, char ∗∗argv)

6.3.1 Detailed Description

This file serves as an entry point to the compiler. It simply creates a compiler object, parses the command-line arguments and starts the compiler.

Definition in file main.cpp.

6.4 Platform.h File Reference

This file simply lists the size in bytes of the base data types.

Macros

- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8

- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4

- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1
- #define **CHAR_SIZE** 1
- #define **INT_SIZE** 4
- #define **LONG_SIZE** 8
- #define **FLOAT_SIZE** 4
- #define **DOUBLE_SIZE** 8
- #define **POINTER_SIZE** 4
- #define **SHORT_SIZE** 1

6.4.1 Detailed Description

This file simply lists the size in bytes of the base data types.

Definition in file Platform.h.

Chapter 7

Example Documentation

7.1 test1.c

7.1.1 About This Example:

The most basic example. This example demonstrates the basic capabilities of the compiler to generate valid MI-PS assembler code for variable declarations, simple assignments, and basic arithmetic. In addition, this example demonstrates the ability of the register allocation algorithm to handle the need for utelizing spill registers.

7.1.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test1.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test1.tac
cd ..
./tac2mips tests/test1.tac
```

This will generate the output file test1.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.1.3 The Example Code:

```
int main()
{
    int i;
    int j;
    int k;

    i = 1;
    j = 2;

    k = i + j; //Expected value of k: 3

    k = j - i; //Expected value of k: 1

    k = i * j; //Expected value of k: 2

    k = j / i; //Expected value of k: 2

    k = (1 + 2) + (3 + 4) + (5 + 6) + (7 + 8) + (9 + 10); //Expected value of k: 55
    return 0;
}
```

7.2 test2.c

7.2.1 About This Example:

This test is designed to test the proper allocation of arrays and calculation of array indices and access operations.

7.2.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test2.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test2.tac
cd ..
./tac2mips tests/test2.tac
```

This will generate the output file test2.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.2.3 The Example Code:

```
int main()
{
    int a[3];

    a[0] = 3;
    a[1] = 2;
    a[2] = 1;

    return 0;
}
```

7.3 test3.c

7.3.1 About This Example:

This test is designed to test the use of if-else statements.

7.3.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test3.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test3.tac
cd ..
./tac2mips tests/test3.tac
```

This will generate the output file test3.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.3.3 The Example Code:

```
int main()
{
    int a;

    a = 1;

    if(a == 2)
    {
        a = 0;
    }
    else
    {
        a = 3;
    }

    //Expected value of a: 3

    return 0;
}
```

7.4 test4.c

7.4.1 About This Example:

This test is designed to test all of the C loops including the for, do-while, and while loops. In addition, this file tests the incrementor and decrementor.

7.4.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test4.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test4.tac
cd ..
./tac2mips tests/test4.tac
```

This will generate the output file test4.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.4.3 The Example Code:

```
 int main()
 {
    int i;
    int a;

    //This will increment a by 2 from 1 to 11
    a = 1;

    for(i = 0; i < 10; i++)
    {
        a = a + 2;
    }

    //This will increment a by 2 from 1 to 11
    a = 1;

    while(a < 10)
    {
        a = a + 2;
    }

    //This will increment a by 2 from 1 to 11
    a = 1;

    do
    {
        a = a + 2;
    } while (a < 10);

    return 0;
}
```

7.5 test5.c

7.5.1 About This Example:

This test is designed to test scoping of variables and calculated array indices (i.e. line 35). This bubble sort algorithm will be used in the next test which tests function calls.

7.5.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test5.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test5.tac
cd ..
./tac2mips tests/test5.tac
```

This will generate the output file test5.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.5.3 The Example Code:

```
//The bubble sort algorithm
int main()
{
    int arr[3];
    bool swapped;
    int i;
    int tmp;

    //Initialize the array to sort
    arr[0] = 3;
    arr[1] = 2;
    arr[2] = 1;

    //Perform the sort
    do
    {
        swapped = false;

        for(i = 1; i < 3; ++i)
        {
            if(a[i - 1] > a[i])
            {
                tmp = a[i - 1];
                a[i - 1] = a[i];
                a[i] = tmp;
                swapped = true;
            }
        }

    } while (swapped);

    //Expected output of arr: [1, 2, 3]

    return 0;
}
```

7.6 test6.c

7.6.1 About This Example:

This test is designed to test function calls. This is the primary goal of the compiler project. This is a very simple bubble sort algorithm which uses a call to a swap function to swap two integers in a small array. The input array is [3, 2, 1] and the output array should be [1, 2, 3].

7.6.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test6.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test6.tac
cd ..
./tac2mips tests/test6.tac
```

This will generate the output file test6.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.6.3 The Example Code:

```
//Swap takes two integers as reference parameters and swaps their values
void swap(int &a, int &b)
{
   int tmp;

   tmp = a;
   a = b;
   b = tmp;
}

//The bubble sort algorithm
int main()
{
   int arr[3];
   bool swapped;
   int i;

   //Initialize the array to sort
   arr[0] = 3;
   arr[1] = 2;
```

```
    arr[2] = 1;

    //Perform the sort
    do
    {
        swapped = false;

        for(i = 1; i < 3; ++i)
        {
            if(a[i - 1] > a[i])
            {
                swap(a[i - 1], a[i]);
                swapped = true;
            }
        }

    } while (swapped);

    //Expected output of arr: [1, 2, 3]

    return 0;
 }
```

7.7 test7.c

7.7.1 About This Example:

This is the final test. This test evaluates the ability of the compiler to handle 2 dimensional arrays by running a simple matrix multiplication algorithm.

7.7.2 Running This Example:

This example can be tested by running the following commands in a bash terminal:

```
./ccomp tests/test7.c
cp TAC.out mips/tests/
cd mips/tests/
mv TAC.out test7.tac
cd ..
./tac2mips tests/test7.tac
```

This will generate the output file test7.asm, which can be assembled and run in the MARS MIPS simulator. Please note, assembly code in this project was created with the MARS simulator in mind. The code is therefore MARS compliant, but may not be SPIM compliant. In addition, the generated code relies upon a MARS asm macro file located in the mips/tests/ directory, and the assembly file should therefore only be run in MARS from the directory it is originally generated in. If you wish to move the asm file to a different folder, you must also move the macros.asm file.

7.7.3 The Example Code:

```
int main()
{
    int i, j, k;
    int a[2][2];
    int b[2][2];
    int c[2][2];

    //a is the identity matrix
    a[0][0] = 1;
    a[0][1] = 0;
    a[1][0] = 0;
    a[1][1] = 1;

    //b is the matrix [[1,2],[3,4]]
    b[0][0] = 1;
    b[0][1] = 2;
    b[1][0] = 3;
    b[1][1] = 4;
    //Matrix Multiplication: c = a * b (c should have the same values as b above)
    for(i = 0; i < 2; i++)
    {
        for(j = 0; j < 2; j++)
        {
            for(k = 0; k < 2; k++)
            {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    //Expected output of c: | 1, 2 |
    //                      | 3, 4 |

    return 0;
}
```