Translator

1.0

Generated by Doxygen 1.8.8

Mon Apr 13 2015 01:42:27

Contents

Chapter 1

Translator

Translator takes a subset of C programming language and generates intermediate (3 address) code .

Subset of C programming language is defined as below

1. Binary Operators: + , - , $*$, / ,exponentiation operator (denote it as @)
2. Data types: int , unsigned , signed , bool , float
3. Bitwise operators : $|$, & , $\sim$, $\hat{}$(XOR)
4. Logical Operators : $||$, && , !
5. Relational Operators : ==, !=, $<$, $<=$, $>$, $>=$
6. Assignment Operators : = , += , -= , $*=$, /=
7. Unary opeartors : + , -
8. Postfix / Prefix Operators : ++ , –
9. Assignment Statement
10. Expressions : infix expressions
11. Identifiers:
    - Simple identifiers without special characters (starts with alphabet)
12. Control Structures:
    (a) Iterative
    (b) conditional
        i. if
        ii. if-else
        iii. else-if
        iv. switch
    (c) Repetitive
        i. while
    (d) Jump
        i. continue
        ii. break

Note:

1. It follows C operators precedence , syntax rule.
2. There is no scope rule for identifiers
3. There is no function call.
4. Functions shouldn't have argument(s).
5. It reports following error messages
   (a) lval requirement
   (b) case label duplication
   (c) Variable is not defined
   (d) Redeclaration of variable
   (e) Redeclared with different type

Input/output

1. Input
   - C program which follows synatx rule as described above
2. Output
   - 3 address code , Output will be displayed on terminal/cmd and it's also available in file output.txt

To test this program do one of the following

1. If you are having binary file name trans
   (a) In unix like system use **./trans filename** e.g. ./trans test.c
   (b) In Windows use **trans filename** e.g. trans test.c
2. If you are having source code of the program .
   (a) Requirements
      i. GNU make utilities
      ii. GNU flex/LEX
      iii. GNU YACC/Bison
      iv. G++ >=4.8
   (b) Extract source file in one of the folder/directory
   (c) using terminal/cmd enter **make** or **make -f Makefile**
   (d) In unix like system use **./trans filename** e.g. ./trans test.c
   (e) In Windows use **trans filename** e.g. trans test.c

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Attributes Struct Reference

Symbol table attributes.

Public Attributes

- int type
  Variable type.
- int lineno
  Where this identifier was found?

4.1.1 Detailed Description

Symbol table attributes.

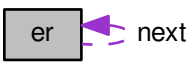
The documentation for this struct was generated from the following file:

- yaccrule.cpp

4.2 er Struct Reference

This records error message(s)

Collaboration diagram for er:



4.2.1 Detailed Description

This records error message(s)

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

- yaccrule.cpp

4.3 Quadruple Struct Reference

Structure which hold generated code.

4.3.1 Detailed Description

Structure which hold generated code.

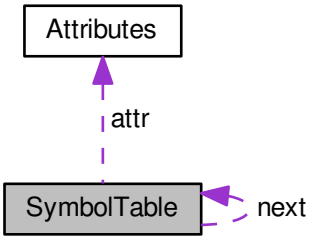
The documentation for this struct was generated from the following file:

- yaccrule.cpp

4.4 SymbolTable Struct Reference

Symbol table node structure.

Collaboration diagram for SymbolTable:



Public Attributes

- char * Identifier
  Identifier name.
- SymbolTable * next
  Next pointer to symbol table If collision occurs then identfier will be added here.
- Attributes * attr
  Symbol table attributes.

4.4.1 Detailed Description

Symbol table node structure.

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

- yaccrule.cpp

Chapter 5

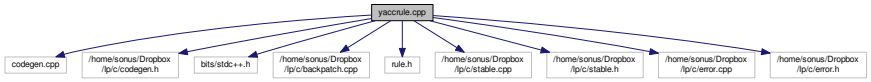
File Documentation

5.1 yaccrule.cpp File Reference

This file includes grammar rule and their semantic action(s).

```
#include "codegen.cpp"
```

Include dependency graph for yaccrule.cpp:



Classes

- struct Quadruple

  Structure which hold generated code.
- struct er

  This records error message(s)
- struct Attributes

  Symbol table attributes.
- struct SymbolTable

  Symbol table node structure.

Functions

- char ∗ newTemp ()

  generate new temporary variable name
- void printCode ()

  print code to the console after completion of parsing and store in file name 'output.txt' for future use
- void genCode (const char ∗result, const char ∗addr1, const char ∗op, const char ∗addr2, const char ∗addr3, int label)

  Generate 3 address code and store in Quadrule table for conditional jump.
- void genCode (const char ∗result, const char ∗addr1, const char ∗op, const char ∗addr2)

  Generate 3 address code and store in Quadrule table for binary expression e.g.
- void genCode (const char ∗result, const char ∗unop, const char ∗addr1)

  Generate 3 address code and store in Quadrule table for unary expression e.g.

- void genCode (const char *result, const char *addr1)

  Generate 3 address code and store in Quadrule table for assignement e.g.
- void genCode (const char *result, int label)

  Generate 3 address code and store in Quadrule table for goto target e.g.
- patchList * makeList (int i)

  Create a new backpatchList.
- patchList * mergeList (patchList *l1, patchList *l2)

  Merge two backpatchList.
- void backpatch (patchList *p, int i)

  Backpatch goto instruction.
- switchLR * makeList (int label, bool type, char *val, int lineno)

  Create backpatchList of switch statement.
- switchLR * mergeSwitchList (switchLR *l1, switchLR *l2)

  Merge two switchList.
- er * newerNode (const char *errmsg, int lineno)

  Create new node to hold error message.
- void AddError (const char *errmsg, int lineno, int8_t ertype)

  Adderror message to Error message linked list.
- void printError ()

  Print error message to console so that user can correct it.
- er * newerNode (const char *errmsg, int lineno)

  Create new node to hold error message.
- int IsDuplicateCaseLabel (YYSTYPE::switchL *l, char *p)

  Check whether any case label is repeated or nor.
- void AddError (const char *errmsg, int lineno, int8_t ertype)

  Adderror message to Error message linked list.
- void printError ()

  Print error message to console so that user can correct it.
- short hashCode (char *s)

  Generate Hashcode for a given string.
- void InsertId (char *, int, int)

  Insert New identifier in Symbol table.
- void InitializeSymbolTable ()

  Initialize Symbol table for future use.
- void DeleteSymbolTable ()

  Delete symbol table.
- bool IsPresent (char *)

  Check whether an identifier is present in Symbol table or not.
- int getLine (char *)

  Get line no where given identifier was found .
- short getType (char *)

  Get variable type of a given identifier.
- void getType (char *errmsg, int type)

  Generate variable type of a given identifier in string format.
- short hashCode (char *str)

  Generate Hashcode for a given string.
- SymbolTable * newTable (char *id, int lineno, int type)

  This function create a new node for symbol table entry.
- void InsertId (char *id, int lineno, int type)

  Insert New identifier in Symbol table.
- void InitializeSymbolTable ()

Initialize Symbol table for future use.

- void DeleteSymbolTable ()

  Delete symbol table.
- bool IsPresent (char *id)

  Check whether an identifier is present in Symbol table or not.
- int getLine (char *id)

  Get line no where given identifier was found .
- short getType (char *id)

  Get variable type of a given identifier.
- void getType (char *errmsg, int type)

  Generate variable type of a given identifier in string format.
- char * newTemp ()

  generate new temporary variable name
- void printCode ()

  print code to the console after completion of parsing and store in file name 'output.txt' for future use
- void genCode (const char *result, const char *addr1, const char *op, const char *addr2)

  Generate 3 address code and store in Quadrule table for binary expression e.g.
- void genCode (const char *result, const char *unop, const char *addr1)

  Generate 3 address code and store in Quadrule table for unary expression e.g.
- void genCode (const char *result, const char *addr1, const char *op, const char *addr2, const char *addr3, int label)

  Generate 3 address code and store in Quadrule table for conditional jump.
- void genCode (const char *result, const char *addr1)

  Generate 3 address code and store in Quadrule table for assignement e.g.
- void genCode (const char *result, int label)

  Generate 3 address code and store in Quadrule table for goto target e.g.

5.1.1 Detailed Description

This file includes grammar rule and their semantic action(s).

Author

sonu kumar , Roll no 127159 , section : A , Course B.tech(3/4)

Version

1.0

5.1.2 Function Documentation

5.1.2.1 char* newTemp ( )

generate new temporary variable name

Parameters

void	None

Returns

newtemp char*

5.1.2.2 void printCode ( )

print code to the console after completion of parsing and store in file name 'output.txt' for future use

Parameters

void	None

Returns

void None

5.1.2.3 void genCode (const char * *result*, const char * *addr1*, const char * *op*, const char * *addr2*, const char * *addr3*, int *label*)

Generate 3 address code and store in Quadrule table for conditional jump.

Parameters

if	char*
address1	char*
relational_operator	char*
address2	char*
goto	char*
jump_instruction_number	int

Returns

void None

5.1.2.4 void genCode (const char * *result*, const char * *addr1*, const char * *op*, const char * *addr2*)

Generate 3 address code and store in Quadrule table for binary expression e.g.

a=t0+b;

Parameters

result	char*
address1	char*
binary_operator	char*
address2	char*

Returns

void None

5.1.2.5 void genCode (const char * *result*, const char * *unop*, const char * *addr1*)

Generate 3 address code and store in Quadrule table for unary expression e.g.

a = -b;

Parameters

result	char*
address1	char*
unary_operator	char*

Returns

void None

5.1.2.6 void genCode (const char * *result*, const char * *addr1*)

Generate 3 address code and store in Quadrule table for assignement e.g.

a = t0;

Parameters

result	char*
address2	char*

Returns

void None

5.1.2.7 void genCode (const char * *result*, int *label*)

Generate 3 address code and store in Quadrule table for goto target e.g.

'goto' -1

Parameters

goto	char*
jump_instruction_number	int

Returns

void None

5.1.2.8 patchList* makeList (int *i*)

Create a new backpatchList.

Parameters

jump_instruction_number	int

Returns

backpatchList patchList*

5.1.2.9 patchList* mergeList (patchList * *l1*, patchList * *l2*)

Merge two backpatchList.

Parameters

backpatchList_1	patchList*
backpatchList_2	patchList*

Returns

backpatchList_3 patchList*

5.1.2.10 void backpatch (patchList * *p*, int *i*)

Backpatch goto instruction.

Parameters

backpatchList	patchList*
target_instruction_number	int

Returns

void None

5.1.2.11 switchLR* makeList (int *label*, bool *type*, char * *val*, int *lineno*)

Create backpatchList of switch statement.

Parameters

instruction_number	int
switch_statement_type	bool
case_value(default:NULL)	char*
line_number	int

Returns

new_switch_list switchL*

5.1.2.12 switchLR* mergeSwitchList (switchLR * *l1*, switchLR * *l2*)

Merge two switchList.

Parameters

SwitchList_1	switchL*
SwitchList_2	switchL*

Returns

SwitchList_3 switchL*

5.1.2.13 er* newerNode (const char * *errmsg,* int *lineno*)

Create new node to hold error message.

Parameters

error_message	char*
line_number	int

Returns

error_node er*

5.1.2.14 void AddError (const char * *errmsg,* int *lineno,* int8_t *ertype*)

Adderror message to Error message linked list.

Parameters

error_message	char*
line_number	int
error_type	ERROR/NOTE/WARNING

If there is no error occur till now then create a new node and assign to error message list Else follow next pointer till end of the list and add there

5.1.2.15 void printError ( )

Print error message to console so that user can correct it.

Parameters

void	None

Returns

void None

5.1.2.16 er* newerNode (const char * *errmsg,* int *lineno*)

Create new node to hold error message.

Parameters

error_message	char*
line_number	int

Returns

error_node er*

5.1.2.17 int IsDuplicateCaseLabel (YYSTYPE::switchL * *l,* char * *p*)

Check whether any case label is repeated or nor.

Parameters

switch_list	switchL∗
value	int

Returns

line_number int

Scan switch list if there is duplicate then return line_number where it was declared first else return -1

5.1.2.18 void AddError (const char ∗ *errmsg,* int *lineno,* int8_t *ertype*)

Adderror message to Error message linked list.

Parameters

error_message	char∗
line_number	int
error_type	ERROR/NOTE/WARNING

If there is no error occur till now then create a new node and assign to error message list Else follow next pointer till end of the list and add there

5.1.2.19 void printError ( )

Print error message to console so that user can correct it.

Parameters

void	None

Returns

void None

5.1.2.20 short hashCode (char ∗ *str*)

Generate Hashcode for a given string.

Parameters

string	char∗

Returns

hashcode short

5.1.2.21 void InsertId (char ∗ *id,* int *lineno,* int *type*)

Insert New identifier in Symbol table.

Parameters

identifier_name	char∗

line_number	int
type	int

Returns

void None

Check whether Symbol table Entry is empty or not ?

1. If it's empty then insert id
2. else follow next pointer

Check whether next pointer is NULL or not ?

1. If it's NULL then insert create a new Symbol table and insert id
2. else follow next pointer of next

5.1.2.22 void InitializeSymbolTable ( )

Initialize Symbol table for future use.

Parameters

void	None

Returns

void None

5.1.2.23 void DeleteSymbolTable ( )

Delete symbol table.

Parameters

void	None

Returns

void None

5.1.2.24 bool IsPresent (char * *id*)

Check whether an identifier is present in Symbol table or not.

Parameters

identifier	char*

Returns

true/false bool

1 . If symbol table is empty then return not found 2 . Check symbol table if necessary then follow next pointer

5.1.2.25 int getLine (char * *id*)

Get line no where given identifier was found .

Parameters

identifier	char*

Returns

line_number int

5.1.2.26 short getType (char * *id*)

Get variable type of a given identifier.

Parameters

identifier	char*

Returns

variable_type int

5.1.2.27 void getType (char * *errmsg*, int *type*)

Generate variable type of a given identifier in string format.

Parameters

errmsg	char*
type	int

Returns

void None

5.1.2.28 short hashCode (char * *str*)

Generate Hashcode for a given string.

Parameters

string	char*

Returns

hashcode short

5.1.2.29 SymbolTable* newTable (char * *id*, int *lineno*, int *type*)

This function create a new node for symbol table entry.

Parameters

identifier_name	char*

line_number	int
type	int

Returns

SymbolTable_node SymbolTable∗

5.1.2.30 void InsertId (char ∗ *id,* int *lineno,* int *type*)

Insert New identifier in Symbol table.

Parameters

identifier_name	char∗
line_number	int
type	int

Returns

void None

Check whether Symbol table Entry is empty or not ?

1. If it's empty then insert id
2. else follow next pointer

Check whether next pointer is NULL or not ?

1. If it's NULL then insert create a new Symbol table and insert id
2. else follow next pointer of next

5.1.2.31 void InitializeSymbolTable ( )

Initialize Symbol table for future use.

Parameters

void	None

Returns

void None

5.1.2.32 void DeleteSymbolTable ( )

Delete symbol table.

Parameters

void	None

Returns

void None

5.1.2.33 bool IsPresent (char * *id*)

Check whether an identifier is present in Symbol table or not.

Parameters

identifier	char*

Returns

true/false bool

1 . If symbol table is empty then return not found 2 . Check symbol table if necessary then follow next pointer

5.1.2.34 int getLine (char * *id*)

Get line no where given identifier was found .

Parameters

identifier	char*

Returns

line_number int

5.1.2.35 short getType (char * *id*)

Get variable type of a given identifier.

Parameters

identifier	char*

Returns

variable_type int

5.1.2.36 void getType (char * *errmsg*, int *type*)

Generate variable type of a given identifier in string format.

Parameters

errmsg	char*
type	int

Returns

void None

5.1.2.37 char* newTemp ( )

generate new temporary variable name

Parameters

void	None

Returns

newtemp char*

5.1.2.38 void printCode ( )

print code to the console after completion of parsing and store in file name 'output.txt' for future use

Parameters

void	None

Returns

void None

5.1.2.39 void genCode (const char * *result,* const char * *addr1,* const char * *op,* const char * *addr2*)

Generate 3 address code and store in Quadrule table for binary expression e.g.

a=t0+b;

Parameters

result	char*
address1	char*
binary_operator	char*
address2	char*

Returns

void None

5.1.2.40 void genCode (const char * *result,* const char * *unop,* const char * *addr1*)

Generate 3 address code and store in Quadrule table for unary expression e.g.

a = -b;

Parameters

result	char*
address1	char*
unary_operator	char*

Returns

void None

5.1.2.41 void genCode (const char * *result,* const char * *addr1,* const char * *op,* const char * *addr2,* const char * *addr3,* int *label*)

Generate 3 address code and store in Quadrule table for conditional jump.

Parameters

if	char*
address1	char*
relational_operator	char*
address2	char*
goto	char*
jump_instruction_number	int

Returns

void None

5.1.2.42 void genCode (const char * *result*, const char * *addr1*)

Generate 3 address code and store in Quadrule table for assignement e.g.

a = t0;

Parameters

result	char*
address2	char*

Returns

void None

5.1.2.43 void genCode (const char * *result*, int *label*)

Generate 3 address code and store in Quadrule table for goto target e.g.

'goto' -1

Parameters

goto	char*
jump_instruction_number	int

Returns

void None

Index