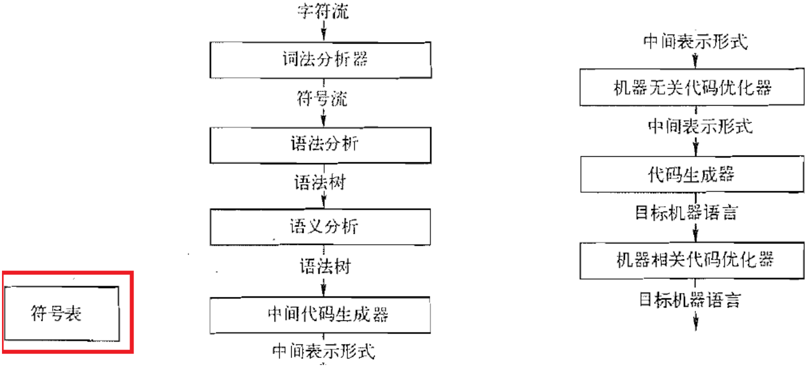
符号表

魏恒峰

hfwei@nju.edu.cn

2023 年 04 月 12 日


字符流
词法分析器
符号流
语法分析
语法树
语义分析
语法树
中间代码生成器
中间表示形式
符号表
中间表示形式
机器无关代码优化器
中间表示形式
代码生成器
目标机器语言
机器相关代码优化器
目标机器语言

Definition (符号表 (Symbol Table))

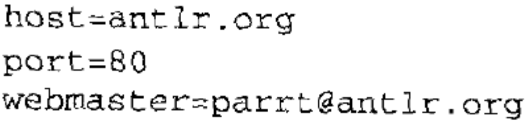
符号表是用于保存**各种信息**的**数据结构**。

Definition (符号表 (Symbol Table))

符号表是用于保存**各种信息**的**数据结构**。

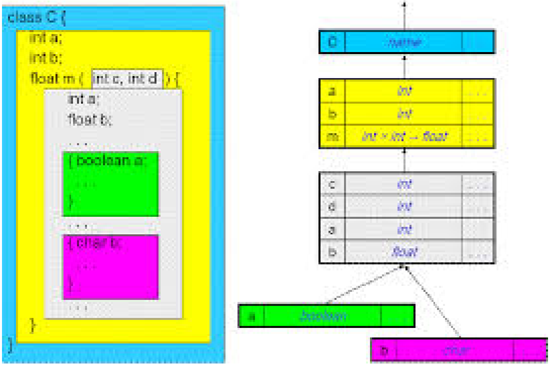
Name	**Type**	**Size**	**Dimension**	**Line of Declaration**	**Line of Usage**	**Address**	$\cdots$
count	int	4	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$
str	char	5	1	$\cdots$	$\cdots$	$\cdots$	$\cdots$

“领域特定语言” (DSL) 通常只有**单作用域** (全局作用域)



```
host=antlr.org
port=80
webmaster=parrt@antlr.org
```

“通用程序设计语言” (GPL) 通常需要**嵌套作用域**



```
class C {
  int a;
  int b;
  float m ( int c, int d ) {
    int a;
    float b;
    . . .
    { boolean a;
      . . .
    }
    . . .
    { char b;
      . . .
    }
    . . .
  }
}
```

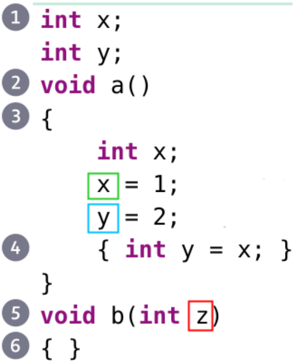
C	*name*	

a	*int*	. . .
b	*int*	. . .
m	*int × int → float*	. . .

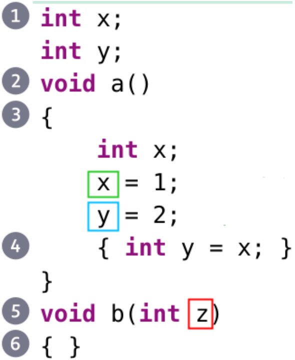
c	*int*	. . .
d	*int*	. . .
a	*int*	
b	*float*	. . .

a	*boolean*	

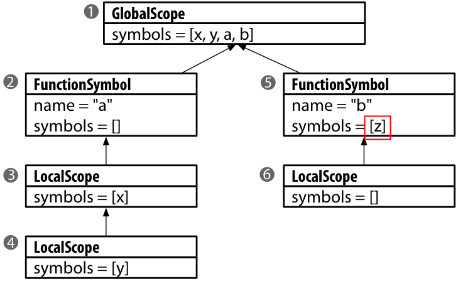
b	*char*	. . .



```
① int x;
  int y;
② void a()
③ {
      int x;
      x = 1;
      y = 2;
④     { int y = x; }
  }
⑤ void b(int z)
⑥ { }
```



```
① int x;
  int y;
② void a()
③ {
      int x;
      x = 1;
      y = 2;
④     { int y = x; }
  }
⑤ void b(int z)
⑥ { }
```



❶ GlobalScope
symbols = [x, y, a, b]

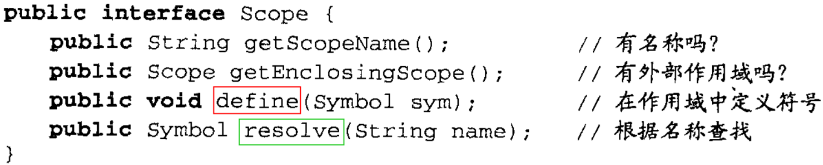
❷ FunctionSymbol
name = "a"
symbols = []

❸ LocalScope
symbols = [x]

❹ LocalScope
symbols = [y]

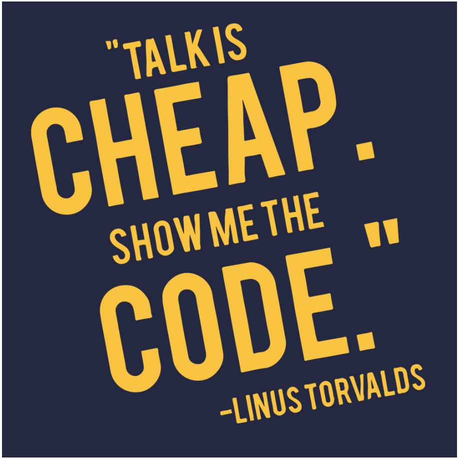
❺ FunctionSymbol
name = "b"
symbols = [z]

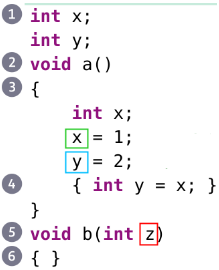
❻ LocalScope
symbols = []



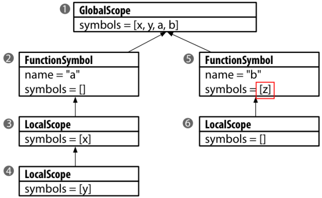
```
public interface Scope {
    public String getScopeName();                  // 有名称吗?
    public Scope getEnclosingScope();              // 有外部作用域吗?
    public void define(Symbol sym);                // 在作用域中定义符号
    public Symbol resolve(String name);            // 根据名称查找
}
```

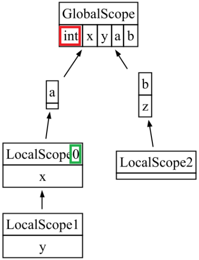
全局作用域、函数/方法作用域、局部作用域

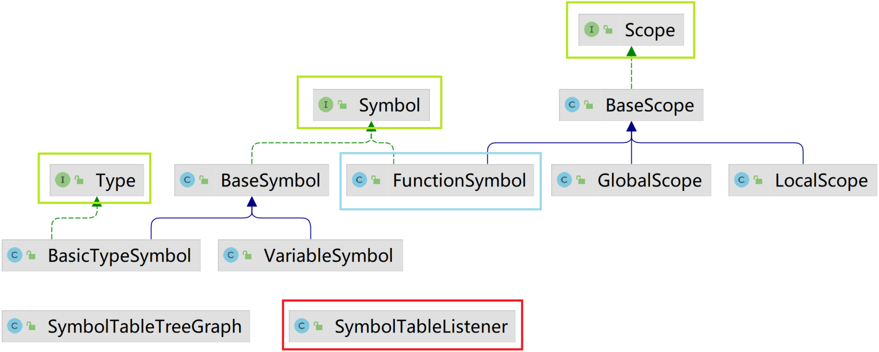

"TALK IS
CHEAP.
SHOW ME THE
CODE."
-LINUS TORVALDS

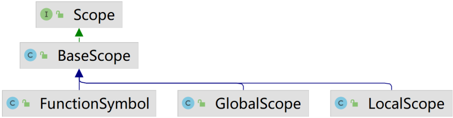


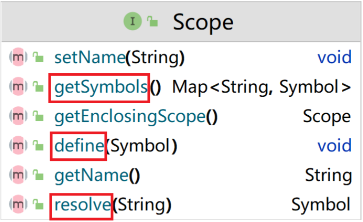
```
① int x;
  int y;
② void a()
③ {
      int x;
      x = 1;
      y = 2;
④     { int y = x; }
  }
⑤ void b(int z)
⑥ { }
```






Scope
Symbol
BaseScope
Type
BaseSymbol
FunctionSymbol
GlobalScope
LocalScope
BasicTypeSymbol
VariableSymbol
SymbolTableTreeGraph
SymbolTableListener



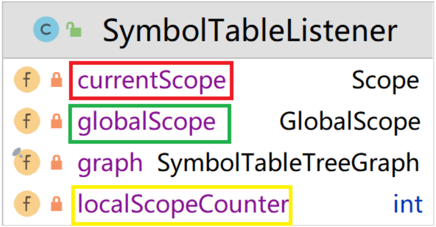


Scope

setName(String)	void
getSymbols()	Map<String, Symbol>
getEnclosingScope()	Scope
define(Symbol)	void
getName()	String
resolve(String)	Symbol

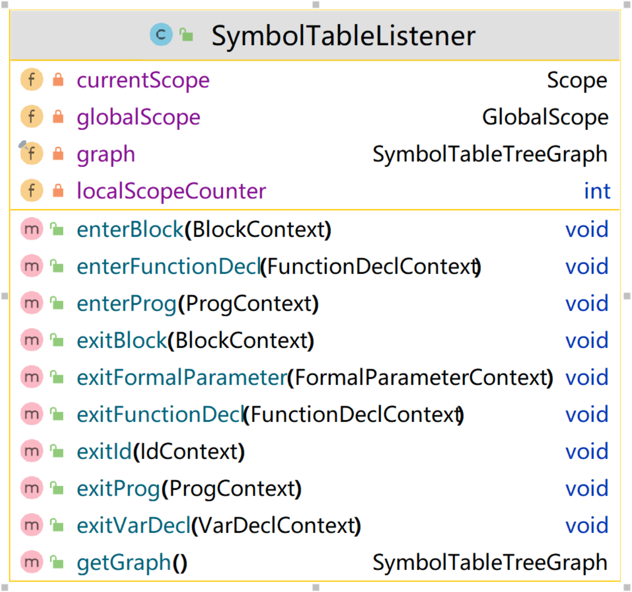
SymbolTableListener

SymbolTableListener



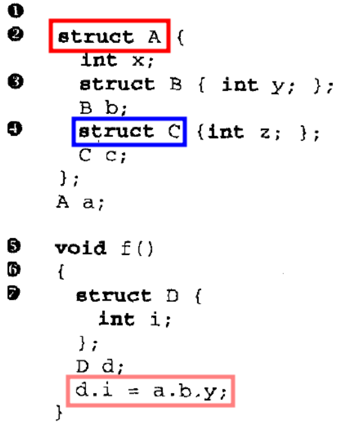
SymbolTableListener

currentScope	Scope
globalScope	GlobalScope
graph	SymbolTableTreeGraph
localScopeCounter	int



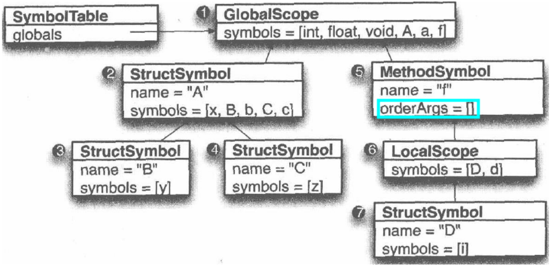
c SymbolTableListener	
f currentScope	Scope
f globalScope	GlobalScope
f graph	SymbolTableTreeGraph
f localScopeCounter	int
m enterBlock(BlockContext)	void
m enterFunctionDecl(FunctionDeclContext)	void
m enterProg(ProgContext)	void
m exitBlock(BlockContext)	void
m exitFormalParameter(FormalParameterContext)	void
m exitFunctionDecl(FunctionDeclContext)	void
m exitId(IdContext)	void
m exitProg(ProgContext)	void
m exitVarDecl(VarDeclContext)	void
m getGraph()	SymbolTableTreeGraph

struct: 类型作用域



```
❶
❷ struct A {
      int x;
❸     struct B { int y; };
      B b;
❹     struct C {int z; };
      C c;
  };
  A a;

❺ void f()
❻ {
❼     struct D {
          int i;
      };
      D d;
      d.i = a.b.y;
  }
```



SymbolTable
globals

❶ GlobalScope
symbols = [int, float, void, A, a, f]

❷ StructSymbol
name = "A"
symbols = [x, B, b, C, c]

❺ MethodSymbol
name = "f"
orderArgs = []

❸ StructSymbol
name = "B"
symbols = [y]

❹ StructSymbol
name = "C"
symbols = [z]

❻ LocalScope
symbols = [D, d]

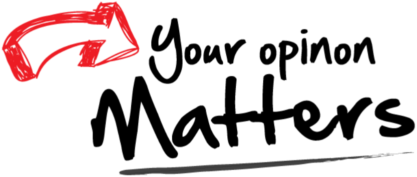
❼ StructSymbol
name = "D"
symbols = [i]

$d.i \qquad a.b.y$

symtab @ antlr by parrt

symtab @ cs652 by parrt

Thank You!



Office 926

hfwei@nju.edu.cn