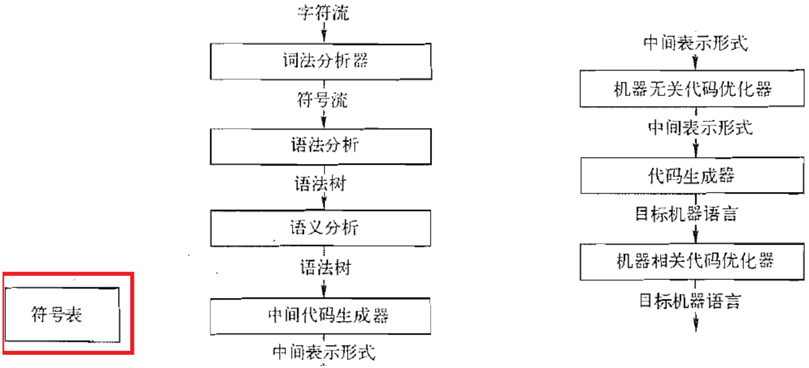
三、语义分析 (1. 符号表)

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))

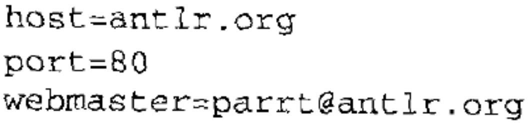
符号表是用于保存**各种信息**的**数据结构**。

Definition (符号表 (Symbol Table))

符号表是用于保存**各种信息**的**数据结构**。

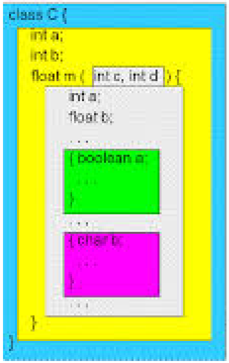
Name	**Type**	**Size**	**Dimension**	**Line of Declaration**	**Line of Usage**	**Address**	$\cdots$
count	int	4	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$
str	char[]	5	1	$\cdots$	$\cdots$	$\cdots$	$\cdots$

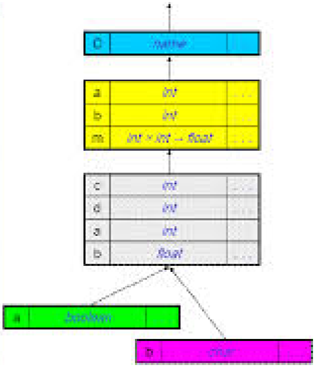
“领域特定语言” (DSL) 通常只有**单作用域** (全局作用域)

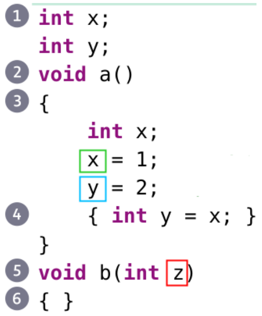


```
host=antlr.org
port=80
webmaster=parrt@antlr.org
```

“通用程序设计语言” (GPL) 通常需要**嵌套作用域**

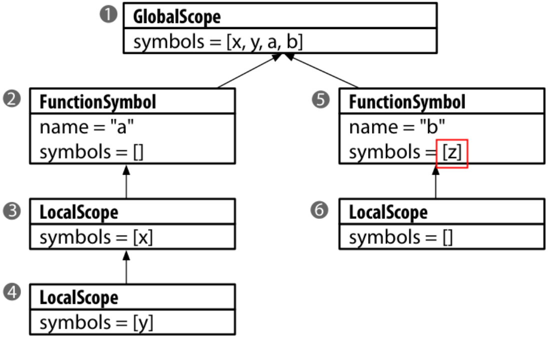




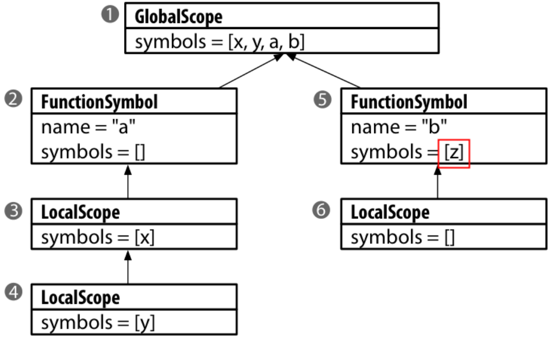


```
① int x;
  int y;
② void a()
③ {
      int x;
      x = 1;
      y = 2;
④     { int y = x; }
  }
⑤ void b(int z)
⑥ { }
```

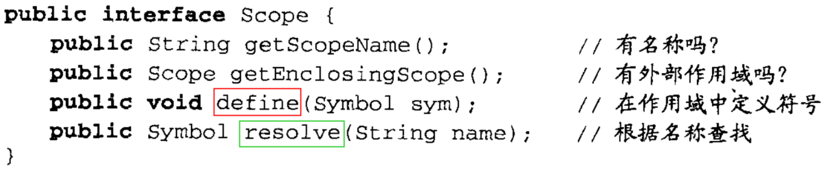
```
❶ int x;
  int y;
❷ void a()
❸ {
      int x;
     x = 1;
     y = 2;
❹     { int y = x; }
  }
❺ void b(int z)
❻ { }
```



```
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 { }
```

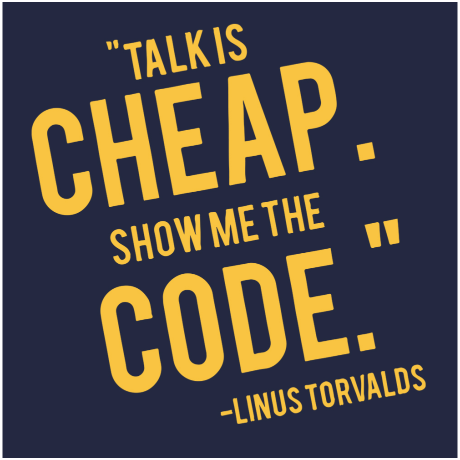


We take a **WRONG** assumption here about `FunctionSymbol`'s scope.

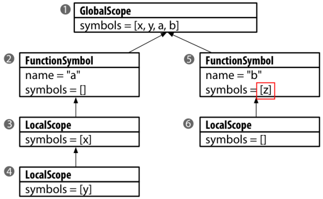


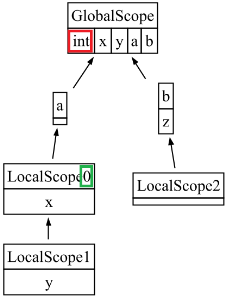
```
public interface Scope {
    public String getScopeName();                // 有名称吗?
    public Scope getEnclosingScope();            // 有外部作用域吗?
    public void define(Symbol sym);              // 在作用域中定义符号
    public Symbol resolve(String name);          // 根据名称查找
}
```

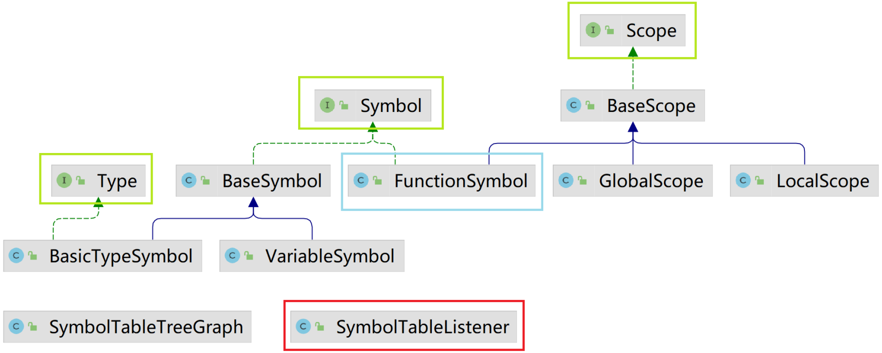
全局作用域、函数/方法作用域、局部作用域

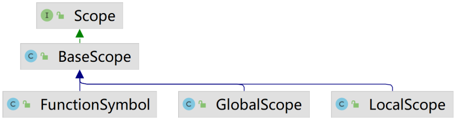

"TALK IS
CHEAP.
SHOW ME THE
CODE."
-LINUS TORVALDS

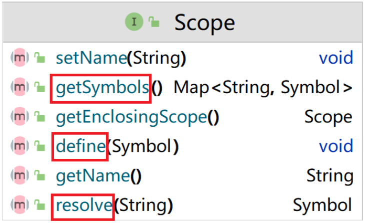
```
1 int x;
  int y;
2 void a()
3 {
      int x;
      x = 1;
      y = 2;
4     { int y = x; }
  }
5 void b(int z)
6 { }
```





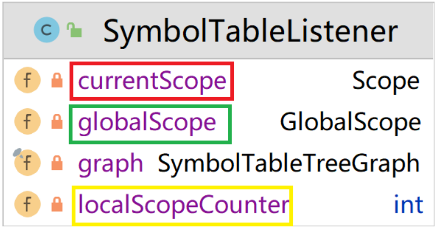

Scope
Symbol
BaseScope
Type
BaseSymbol
FunctionSymbol
GlobalScope
LocalScope
BasicTypeSymbol
VariableSymbol
SymbolTableTreeGraph
SymbolTableListener





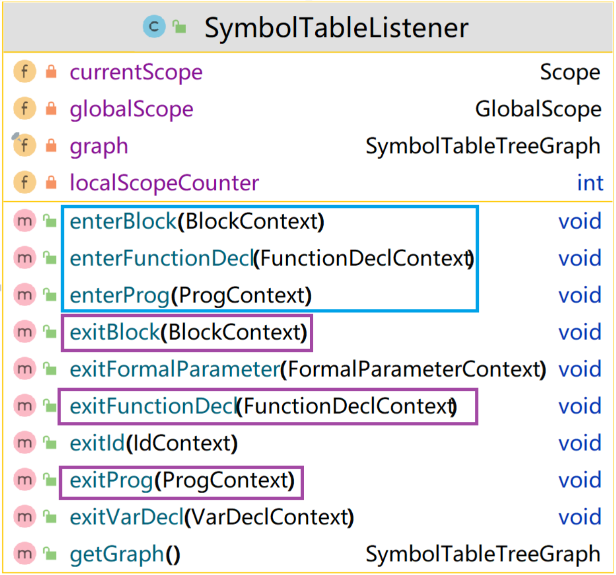
Scope	
setName(String)	void
getSymbols()	Map<String, Symbol>
getEnclosingScope()	Scope
define(Symbol)	void
getName()	String
resolve(String)	Symbol

SymbolTableListener



SymbolTableListener

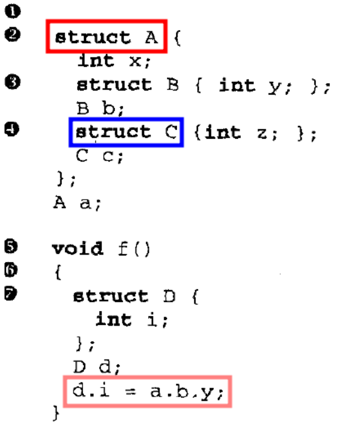
currentScope	Scope
globalScope	GlobalScope
graph	SymbolTableTreeGraph
localScopeCounter	int



SymbolTableListener

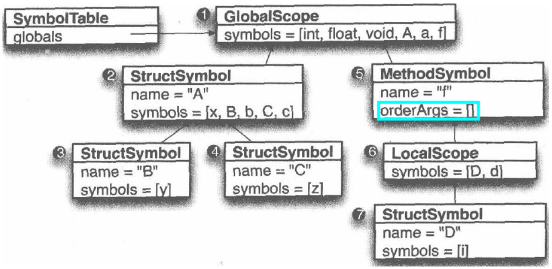
currentScope	Scope
globalScope	GlobalScope
graph	SymbolTableTreeGraph
localScopeCounter	int
enterBlock(BlockContext)	void
enterFunctionDecl(FunctionDeclContext)	void
enterProg(ProgContext)	void
exitBlock(BlockContext)	void
exitFormalParameter(FormalParameterContext)	void
exitFunctionDecl(FunctionDeclContext)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

struct/class: 类型作用域



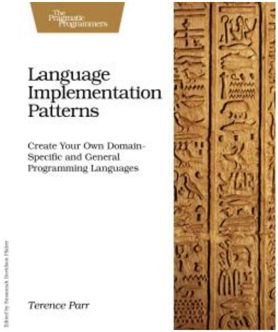
```
❶
❷ struct A {
      int x;
❸     struct B { int y; };
      B b;
❹     struct C {int z; };
      C c;
  };
  A a;

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

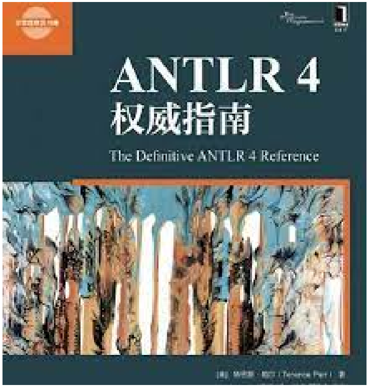


$d.i$ $a.b.y$

第 6 章: 记录并识别程序中的符号


The Pragmatic Programmers
Language Implementation Patterns
Create Your Own Domain-Specific and General Programming Languages
Terence Parr

第 7 章: 管理数据聚集的符号表

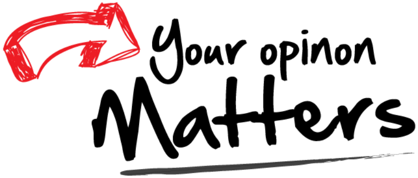


第 8.4 节: 验证程序中符号的使用

symtab @ antlr by parrt

symtab @ cs652 by parrt

Thank
You!



Office 926

hfwei@nju.edu.cn