

三、语义分析 (8. 符号表)

魏恒峰

hfwei@nju.edu.cn

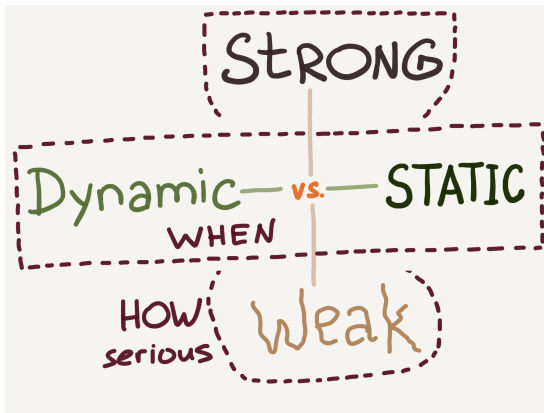
2024 年 04 月 12 日



Semantic Analysis Explained

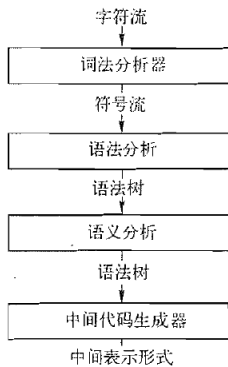


类型检查 (Type Checking)

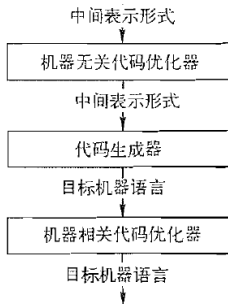


符号 (Symbols) 检查

```
5   int one = 1;  
6   int three = one + two;  
7   int five = len("Hello");  
8  
9   int two = one(one);  
10  int one = 1;
```



符号表



符号: 变量名、函数名、类型名、标签名、...

Definition (符号表 (Symbol Table))

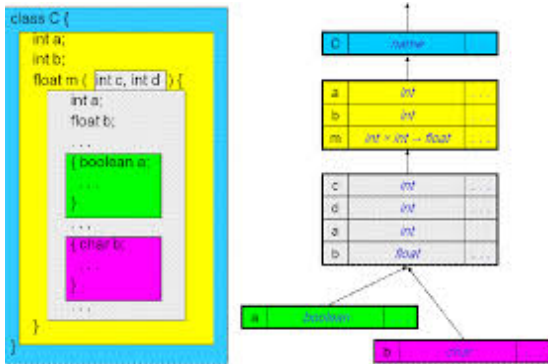
符号表是用于保存**各种信息**的**数据结构**。

Name	Type	Size	Dimension	Line of Declaration	Line of Usage	Address	...
<i>count</i>	int	4	0
<i>str</i>	char[]	5	1

“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
```

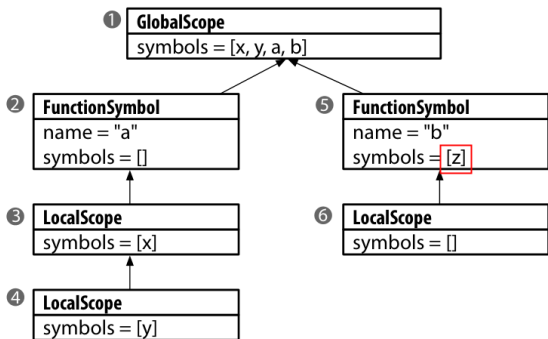
“通用程序设计语言” (GPL) 通常需要**嵌套作用域**




```

1 int x;
  int y;
2 void a()
3 {
    int x;
    x = 1;
    y = 2;
4     { int y = x; }
5 }
6 void b(int z)
7 { }

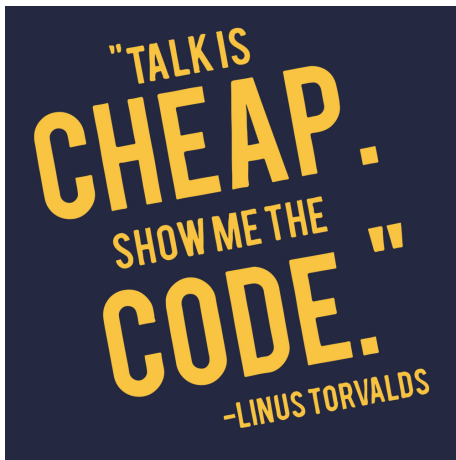
```



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  
}
```

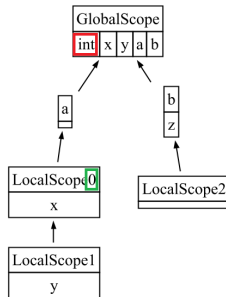
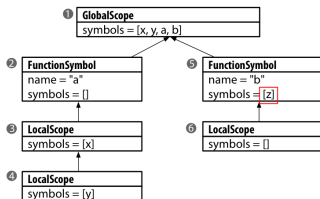
全局作用域、函数/方法作用域、局部作用域

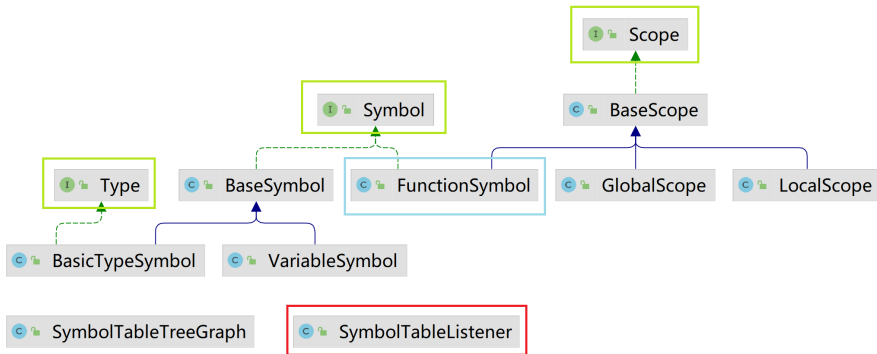


```

1 int x;
  int y;
2 void a()
3 {
    int x;
    x = 1;
    y = 2;
4     { int y = x; }
5 }
6 void b(int z)
7 { }

```
















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

 SymbolTableListener

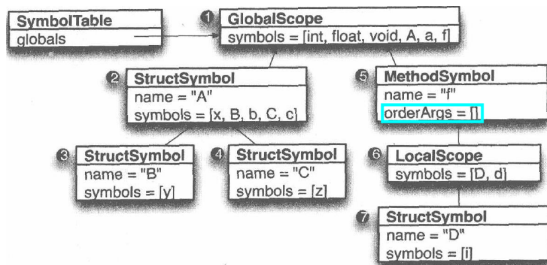
 SymbolTableListener		
	 currentScope	Scope
	 globalScope	GlobalScope
	 graph	SymbolTableTreeGraph
	 localScopeCounter	int

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

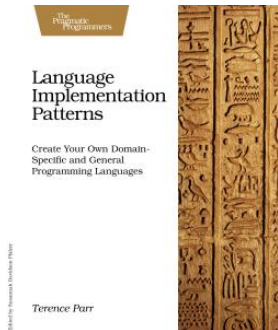
struct/class: 类型作用域

```
1
2 struct A {
3     int x;
4     struct B { int y; };
5     B b;
6     struct C {int z; };
7     C c;
8 };
9 A a;
10
11 void f()
12 {
13     struct D {
14         int i;
15     };
16     D d;
17     d.i = a.b.y;
18 }
```

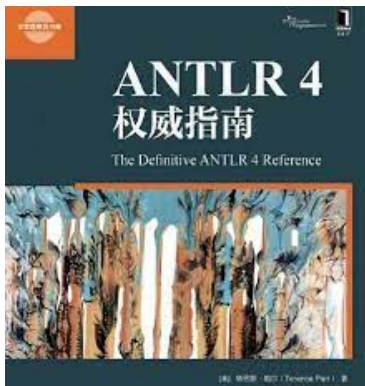


d.i *a.b.y*

第 6 章: 记录并识别程序中的符号



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