编译原理课程实验报告

实验 3：语义分析

姓名	王少博	院系	计算机科学与技术学院	学号	181110315
任课教师	韩希先		指导教师	韩希先	
实验地点	研究院中 507		实验时间	2020.11.19	
实验课表现	出勤、表现得分		实验报告得分		实验总分	
	操作结果得分					

一、实验目的

要求：需分析本次实验的基本目的，并综述你是如何实现这些目的的？

基本目的：

1. 加深对语法制导翻译原理的理解。
2. 掌握将语法分析所识别的语法成分变换为中间代码的语义翻译方法。
3. 掌握语法制导翻译的基本功能，并将其实现。
4. 能够自行给出各语句的语义动作，对基于语法指导翻译的知识进行语义分析，并仔细分析语义分析结果的输出形式，尽可能处理符号表中的信息。

如何实现这些目的：

基于词法和语法分析的基础，将每个语义规则放到相应产生式的末尾得到一个后缀式语法制导翻译模式，而后对产生式归约时执行语义动作

二、实验内容

(1) 语义动作

表 1 语义动作

表达式	语义动作
P→F	-
F→T'ID(A)S	-
F→T'ID()S	-
T'→void	Offset=0;
T'→T	Offset=0; Enter("return",T.type,offset); Offset+=T.width;
A→D	-
A→D,A	-
D→T ID	Enter(ID.name,T.type,offset); Offset += T.width;
T→char	T.type="char"; T.width=1;
T→int	T.type="int"; T.width=4;
T→float	T.type="real"; T.width = 8;

S→{L}	S.nextlist = L.nextlist;
L→S	L.nextlist=S.nextlist;
L→MS	Backpatch(M.nextlist,M.quad); L.nextlist=S.nextlist;
S→D;	S.nextlist=null;

S→ID=E;	int p=lookup(ID.name);
	if(p!=null) gencode(p,":=",E.addr);
	else error;

S→if(BMS EL S	Backpatch(B.truelist,M.quad) Backpatch(B.falselist,EL.quad); S.nextlist = merge(S1.nextlist,merge(El.nextlist,S2.nextlist));
S→if(BMS	Backpatch(B.truelist,M.quad); S.nextlist=merge(B.falselist,S1.nextlist)
S→W(BMS	Backpatch(S1.nextlist,W.quad); Backpatch(B.truelist,M.quad); S.nextlist = B.falselist; Gencode("goto",W.quad);
E→E+E	E.addr = newtemp(E.type,E.width); Gencode(E.addr,":=",E1.addr,"+",E2.addr);
E→E*E	E.addr=newtemp(E.type,E.width); Gencode(E.addr,":=",E1.addr,"*",E2.addr);
E→-E	E.addr = newtemp(E.type,E.width); Gencode(E.addr,":=","minus",E1.addr);
E→(E)	E.addr = E1.addr;

E→ID	int p=lookup(ID.name);
	if(p!=null)E.addr=p;
	else error;

E→C	E.type=C.type; E.width=C.width; E.addr=newtemp(C.type,C.width); Gencode(E.addr,":=","",C.val,"");
C→INT	C.type="int"; C.width=4; C.val=INT.val;
C→FLOAT	C.type="real"; C.width=8; C.val=REAL.val;
C→CHAR	C.type="char"; C.width=1; C.val=CHAR.val;
B→B OR B	Backpatch(B1.falselist,OR.quad); B.truelist=merge(B1.truelist,B2.truelist); B.falselist=B2.falselist;

B→B AND B	Backpatch(B1.truelist,AND.quad); B.falselist=merge(B1.falselist,B2.falselist); B.truelist=B2.truelist;
B→!B	B.truelist=B1.falselist; B.falselist=B1.truelist;
B→(B)	B.truelist = B1.truelist; B.falselist = B1.falselist;
B→ERE	B.truelist=makelist(nextquad); B.falselist=makelist(nextquad+1); Gencode(“if”,E1.addr,R.val,E2.addr,”goto-”); Gencode(“goto-”);
B→E	B.truelist=makelist(nextquad); B.falselist=makelist(nextquad+1); Gencode(“if”,E.addr,”!=0”,”goto-”); Gencode(“goto-”);
R→==	R.val=”==”
R→>	R.val=”>”
R→<	R.val=”<”
R→>=	R.val=”>=”
R→<=	R.val=”<=”
R→!=	R.val=”!=”
M→)	M.quad=nextquad;
W→while	W.quad=nextquad;
EL→else	EL.nextlist=makelist(nextquad); Gencode(“goto-”); EL.quad=nextquad;
AND→&&	AND.quad=nextquad;
OR→||	OR.quad=nextquad;

(2)数据结构及其物理实现

数据结构：

主要的数据结构为语法分析阶段生成的语法分析树，每个结点保存着一个 token 二元组。具体实现借助 python 的 list 实现。

物理实现：

该语法分析树在语法分析阶段的 LR 分析过程中生成，具体算法为在查询 action 表的结果为归约时，将要归约的表达式左部的字符作为父结点，表达式右部的字符作为子结点，这样自底向上逐步生成一棵语法分析树。

表 2 数据结构物理实现

名称	类型	注释
table	list	符号表
Argument 类	type, name 都是字符串 offset, width 都是整数	变量类，以四元式的形式保存
offset	整数	符号表偏移量

sentences	list	生成的语句序列
nextquad	整数	下一条指令行编号
Value 类	整数和 dict 形式存储	语法变量类，用于保存语法变量的种别码以及相关属性

注：python 中没有类型的定义，所以表中仅仅是描述实现时候的定义。

(2) 语义属性的分析、设计和实现

表 3 语义属性的分析、设计和实现

变量/终结符	语义属性	类型	描述
T	type width	字符串 整数	变量定义的类型 变量所占的空间（子节）
C	type width val	字符串 整数 字符串	变量定义的类型 变量所占的空间（子节） 数值
E	type width	字符串 整数	变量定义的类型 变量所占的空间（子节）
R	val	字符串	数值
S	nextlist	list	需要回填的结束代码下标序列
L	nextlist	list	需要回填的结束代码下标序列
M	nextlist quad	list 整数	需要回填的结束代码下标序列 待回填下标
B	truelist falselist	list list	需要回填的 true 跳转代码下标序列 需要回填的 false 跳转代码下标序列

(3) 符号表的相关处理

表 4 符号表

操作名	功能	参数	返回值
Enter	添加变量	name, type, width	-
NewTemp	新建临时变量	type, width	offset
Lookup	给定名称查找	name	type, width, offset

(5) 错误处理

- 如果出现了符号表查不到的变量：程序终止并报错
- 如果出现了变量值读取异常：程序终止并报错。
- 输入的文法不符合规定导致语义分析无法正常结束：程序终止并报错。
- 此外，括号不完整，缺少赋值符号等问题也会引发程序中断

三、实验结果

测试样例 1

(1) 测试样例：程序源代码如下

```
int main(){
    float x;
    x = 3.2;
    char y;
    y = 'y';
    c = 32;
    if (x >= 0 && x < 45){
        x = 60;
    }
    else{
        x = 100;
    }
    while(x > 60){
        c = c + 1;
    }
}
```

图 1 测试样例

(2) 词法分析结果

```
(int,-)
(ID,main)
((,-)
(),-)
({,-)
(float,-)
(ID,x)
(;,-)
(ID,x)
(=,-)
(FLOAT,3.2)
(;,-)
(char,-)
(ID,y)
(;,-)
(ID,y)
(=,-)
(CHAR,y)
(;,-)
(int,-)
(ID,c)
(;,-)
(ID,c)
(=,-)
(INT,60)
(;,-)
(if,-)
((,-)
(ID,x)
(>=,-)
(INT,0)
(&&,-)
(ID,x)
(<,-)
(INT,45)
(),-)
({,-)
(ID,x)
(=,-)
(INT,60)
(;,-)
(},-)
(else,-)
({,-)
(ID,x)
(=,-)
(INT,100)
(=,-)
(INT,100)
(;,-)
(},-)
(while,-)
((,-)
(ID,x)
(>,-)
(INT,60)
(),-)
({,-)
(ID,c)
(=,-)
(ID,c)
(+,-)
(INT,1)
(;,-)
(},-)
(},-)
```

图 2 词法分析结果

(3) 语义分析结果：

```
100:    12 := '3.2'
101:    4 := 12
102:    21 := 'y'
103:    20 := 21
104:    26 := '32'
105:    22 := 26
106:    30 := '0'
107:    if 4 >= 30 goto 109
108:    goto 115
109:    34 := '45'
110:    if 4 < 34 goto 112
111:    goto 115
112:    38 := '60'
113:    4 := 38
114:    goto 117
115:    42 := '100'
116:    4 := 42
117:    46 := '60'
118:    if 4 > 46 goto 120
119:    goto 124
120:    50 := '1'
121:    54 := 22 + 50
122:    22 := 54
123:    goto 117
124:    end.
```

图 3 语义分析结果

(4) 符号表

	Name	Type	Width	Offset
1	return	int	4	0
2	x	float	8	4
3		float	8	12
4	y	char	1	20
5		char	1	21
6	c	int	4	22
7		int	4	26
8		int	4	30
9		int	4	34
10		int	4	38
11		int	4	42
12		int	4	46
13		int	4	50
14		int	4	54
15				

图 4 符号表

(5) 错误处理

在程序中，如果出现了符号表查不到的变量或者读取异常都会出现报错。此外，括号不匹配，缺少赋值符号等诸多问题也会引发程序中断。每次出现错误的时候，会输出错误的信息，并且终止程序。以下面的例子为例。

测试样例 2

```
1    int main(){
2        char b;
3        b = 0.34;
4    }
```

图 5 测试样例 2

这里因为 char 类型的 b 被赋值了 0.34，类型不匹配，此时会打印相关的错误提示。

```
7    语义分析错误：b = 0.34类型错误
```

图 6 错误提示

四、实验中遇到的问题总结

（一） 实验过程中遇到的问题总结：

1. 符号表的查找效率比较低耗时比较长：python 作为解释性语言效率比较低，同时算法设计时查表会出现很多次无用的查询。
   解决方案：使用哈希表进行存储，提高效率。
2. 回填问题：增加一行作为结束行，然后以该下标回填。
3. 语法变量属性的保存和更新问题：定义一个语法变量类，里面嵌套一个字典类型用来保存该语法变量的属性值。

（二） 思考题的思考与分析

思考题 1：你编写的程序能否和语义动作完全无关，即无论什么样的语义动作，都不需要修改你编写的程序，说明要达到完全无关需满足什么样的条件？

我的程序尚且不能和语义动作完全无关。如果要达到完全无关，可以在设计上进行重构，加强接口与实现的责任分离以实现与语义动作无关。若要达到无论什么样的语义动作，就得需要多方面判断，将各语义动作加入程序，且至少应满足一个十分紧密完善的统一的语义描述规则。

思考题 2：如果你采用的自顶向下的语法分析，你是如何处理综合属性的，如果你采用的是自底向上的语法分析，你是如何处理继承属性的？

综合属性是从子节点的属性值计算出来的。修改文法和语义动作，采用只含综合属性的语法制导定义进行自底向上的语法分析。

思考题 3：你产生的结果（四元组）还需要经过什么样的处理后就可以等价于汇编程序了？

还需要进行中间代码优化后，再经过依赖目标机器的目标代码生成器生成汇编程序。对于指定架构的硬件，需要根据其指令集合和硬件特性等进行重构。经过优化后，再经过依赖目标机器的目标代码生成器生成汇编程序。

五、实验体会

语法分析主要是分析程序的逻辑结构，判断程序是否有语法错误，决定程序的执行过程。而语义分析是在语法分析的基础之上，根据语义规则，对符号表中的数值进行操作，并生成中间代码。通过本次实验，我对老师上课所讲的语法制导翻译机制以及语义分析、语法分析、中间代码生成之间的关系有了一个更深的理解，并且通过尝试，也熟悉了回填的这个过程，学会了使用布尔表达式和分支循环语句的回填机制。也算是真正跳出书本，将上课所学的知识付诸实践。同时自己的代码能力得到了锻炼。这是编译原理最后一个实验，从词法分析器到语法分析器再到语义分析器。

纸上得来终觉浅，绝知此事要躬行。

指导教师评语：

日期：