《编译原理》第三次书面作业

截止日期：2024 年 11 月 19 日

若发现问题，或有任何想法（改进题目、调整任务量等等），请及时联系助教

Q1. 以下是某简单语言的一段代码。其语法基本与 C 语言一致，但支持嵌套的函数声明，且允许全局语句（类似 *Python*）。

```
1  int a0, b0, a2;
2  void foo() {
3      int a1, b1;
4      void bar() {
5          int a2;
6          a2 = a1 + b1;
7          if (a0 != b0) baz();
8      }
9      a1 = a0 - b0;
10     b1 = a0 + b0;
11     if (a1 < b1) bar();
12 }
13 void baz() {
14     int a3;
15     a3 = a0 * b0;
16     if (a2 != a3) foo();
17 }
18 a0 = 1;
19 b0 = 2;
20 a2 = a0 - b0;
21 baz();
```

1. 假设该语言编译器采用一个全局的单符号表栈结构。试指出：在分析至语句 *11* 和 *16* 时，当前开作用域分别有几个？各包含哪些符号？在分析至语句 *16* 时，所访问的 *a2* 是在哪行语句声明的？

2. 假设该语言编译器采用多符号表的组织和管理方式，即每个静态作用域均对应一个符号表；且该语言编译器采用多遍扫描机制，在静态语义检查之前每个作用域中的所有表项均已生成。试指出：在分析至语句 *11* 和 *16* 时，当前开作用域分别有几个？各包含哪些符号？

Q2. 现有文法 *G[S]* 的翻译模式：

$$S \to A \ \{\ \textit{print(A.x)}\ \}$$
$$A \to A_1 aB \ \{\ \textit{A.x := A1.x + B.x}\ \}$$
$$A \to B \ \{\ \textit{A.x := B.x}\ \}$$
$$B \to bA \ \{\ \textit{B.x := A.x + 1}\ \}$$
$$B \to c \ \{\ \textit{B.x = 0}\ \}$$

1. 对于输入串 $cabc$，最终会打印出什么？请画出带标注的语法分析树。

2. 该文法含有左递归，无法应用 $LL(1)$ 分析。请消除文法中的左递归（新引入的非终结符用 R 表示），并给出消除后的翻译模式。

3. 针对消除左递归后的翻译模式构造递归下降分析程序，下面给出了相关全局变量和函数的声明，其中 *matchToken(char)* 的定义与课件一致。请写出 *parseR(int)* 的定义。

```
static char lookahead;
void parseS();
int parseA();
int parseB();
int parseR(int R_i);
void matchToken(char expected);
```

Q3. 现有文法 $G[S]$ 的翻译模式（注意 ; 和 , 都是该语言的终结符）：

$S \to D$ { *print(D.width)* }

$D \to D_1;T$ { *L.type := T.type; L.offset := D1.width; L.width := T.width* } L
{ *D.width := D1.width + L.num * T.width* }

$D \to T$ { *L.type := T.type; L.offset := 0; L.width := T.width* } L
{ *D.width := L.num * T.width* }

$T \to \underline{int}$ { *T.type := 'int'; T.width := 4* }

$T \to \underline{real}$ { *T.type := 'real'; T.width := 8* }

$L \to$ { *L1.type := L.type; L1.offset := L.offset; L1.width := L.width* } $L_1, \underline{id}$
{ *foo(id.name, L.type, L.offset + L1.num * L.width); L.num := L1.num + 1* }

$L \to \underline{id}$ { *foo(id.name, L.type, L.offset); L.num := 1* }

1. 变换翻译模式，使嵌在产生式中间的语义动作集中仅含复写规则，且在自底向上的语法分析过程中，文法符号的所有继承属性均可以通过归约前已出现在分析栈中的确定的综合属性进行访问。

2. 在自底向上分析过程中，语法栈为 *v*，栈顶位置为 *top*，可以用 *v[top+i].width* 访问 *top+i* 位置的非终结符的 *width* 属性。写出在按每个产生式归约时语义计算的一个代码片断。不用考虑对 *top* 的维护。

3. 对于输入串 $\underline{int}\ \underline{id}, \underline{id}$，最终会打印出什么？（虽然可以靠语义猜出来，但建议还是用 LR 分析过程完整算一遍）