编译原理实验：Lab2

指导老师：徐辉

2021 年秋季学期

1 实验介绍

Lab2 的目标是在 Lab1 的基础上，实现一个简单的语法分析器，能够根据指定的文法，将词法分析器输出的单词流进行语法分析，得到并输出一个抽象语法树。

1.1 语法分析

语法分析（Syntacticanalysis，也叫 Parsing）是根据某种给定的形式文法对由单词序列（如英语单词序列）构成的输入文本进行分析并确定其语法结构的一种过程。一般有两种方式，一是自顶向下分析，二是自底向上分析。本次实验提供的框架是偏向于自顶向下的分析方法。

在本实验中，首先定义了用于存放语法分析树的诸多 class，且在 class 中已经定义好了输出函数。在实现过程中，这些 class 已经足够存储本实验目标要求生成的语法分析树的所有信息，一般来说不需要再进行额外的增添修改，当然如果有意愿自由发挥的话，也是完全没有问题的。

接着可以看到代码中的 Top-level 部分，这部分为最上层的分析入口，也就是顶层函数。因为框架为自顶向下分析，所以一开始就要从这一部分进入整个语法分析过程。框架在 $MainLoop()$ 中定义了可能的两类顶层，一个是外部函数 (带有 $extern$ 关键字)，还有一个是用户定义函数 (普通的函数)。相应的处理函数进入主体 parse 部分。同时，在主体部分中，还定义好了运算符号的优先级，储存在 BinopPrecedence 中 (为了简单起见只定义了四个)。

主体 parse 部分是这次 lab 需要补充完成的部分，根据给定的文法，正确地去分析输入流。在框架中已经有了部分实现以供参考。

1.2 语法产生式

本实验完整的语法产生式如表 1所示：

lab2 语法产生式的设计是基于 c 语言来进行的，因此函数定义的格式与之一致。出于一开始的简单性考虑，此次的语法定义是一个极简化的形式，函数的 body 部分只允许一个 statement，而且该 statement 就是直接作为返回值，没有赋值语句、返回语句、循环之类的结构。比如如下代码：

```
int foo(int a, int b){
    a+b
}
```

在这个代码示例中，整个 program 就只有这一个 foo 函数，也就是只有一个 $< function >$，这个 $< function >$ 又可以拆解成 $< prototype >$ 以及 $< body >$ 部分。其中 $< prototype >$ 对应的就

$< program >$	::=	$< gdecl >^* < function >^*$
$< gdecl >$	::=	$extern < prototype >;$
$< function >$	::=	$< prototype >< body >$
$< prototype >$	::=	$< type >< ident > (< paramlist >)$
$< paramlist >$	::=	$\epsilon \mid < type >< ident > [, < type >< ident >]^*$
$< body >$	::=	$< stmt >$
$< stmt >$	::=	$< exp >$
$< exp >$	::=	$(< exp >) \mid < const > \mid < ident > \mid$
		$< exp >< binop >< exp > \mid < callee >$
$< callee >$	::=	$< ident > (\epsilon \mid < exp > [, < exp >]^*)$
$< ident >$	::=	$[A-Z\_a-z][0-9A-Z\_a-z]^*$
$< const >$	::=	$< intconst > \mid < doubleconst >$
$< binop >$	::=	$+ \mid - \mid * \mid <$
$< intconst >$	::=	$[0-9][0-9]^*$
$< doubleconst >$	::=	$< intconst > . < intconst >$
$< type >$	::=	$int \mid double$

表 1: 语法产生式

是函数定义部分，而 $< body >$ 是函数的主体部分。在主体部分中 a 跟 b 都可以归约到 $< ident >$ 中，而两个 $< ident >$ 又归约成两个 $< exp >$，进而由 $< exp >::=< exp >< binop >< exp >$ 归约到 $< exp >$ 再到 $< stmt >$ 和 $< body >$。

值得注意的是，本次 lab 的文法中是包含左递归的，需要自行设计消除解决的办法，并在实验报告中加以阐述。

2 实验测试

lab2 实验的测试用例为压缩包中的 *input_example.data* 以及 *output_example.data*, 在配置好环境后，在当前目录的命令行下输入如下指令，即可输出测试结果。

```
clang++ lab2.cpp -o lab2
./lab2 input_file_name
```

其中，*input_file_name* 就是所要进行词法分析的目标文件。测试的输入输出框架都已提供好，不需要修改，在没有做任何代码填充的情况下，用提供的 input.data 文件作为输入 (里面定义了一个 extern 函数，这部分已经部分实现)，可以得到一个正确的输出。

3 实验提交

实验完成之后，将实验代码在截止日期之前上传至 elearning，助教将根据代码完成质量以及测试用例通过情况进行打分。

提交时需要提交补充完整可编译运行的代码文件以及一份实验报告，实验报告简洁为主，主要描述一下实现思路和解决左递归的办法 (可以用简单的语法产生式扩展来进行描述)。将代码和

实验报告 (实验报告也以学号命名) 压缩到以学号命名的压缩包中 (zip 格式)，提交压缩包文件。代码的文件名保持原样，为 lab2.c。