专题5\_编译器前端实现

13281132 李若森 计科1301

# 实验目的

词法分析的输入和输出，二元式的表示，二元式作为语法分析程序的输入，语法分析程序的输出，理解编译程序的整体概念，理解编译器前端的概念。

# 实验内容

## 实验项目

将词法分析程序（专题1）和语法分析程序（在专题2、3、4中选择一个）形成一个程序，即程序的输入为符号串源程序（算术表达式），输出为判断结果（该符号串是否为正确的算术表达式）。

## 设计说明

终结符号 i 为用户定义的简单变量，即标识符的定义。

## 设计要求

1. 词法分析器的输入为符号串（算术表达式），词法分析器的输出为二元式形式的中间文件
2. 语法分析器读取二元式形式的中间文件并判断符号串是否为正确的算术表达式；
3. 设计两个测试用例（尽可能完备），并给出程序执行结果。

## 任务分析

重点解决编译程序总体结构理解问题；词法分析的输出为语法分析的输入。

# 实现过程

本实验将专题一和专题二粘合在一起构成编译器的前端。词法分析器和语法分析器分别拷到本专题下lexer和parse文件夹下，每个文件夹下有各自的makefile。最终在test文件夹下通过C语言程序调用两个文件夹的函数完成功能衔接，通过makefile将所有程序最终粘合成一个程序。

## 衔接程序：

1. #include "main.h"
2. int main( int argc, char \*\*argv ) {
3. bool flag;
4. flag = lexer( argv[1] );
5. if( flag ) flag = parse();
6. if( flag ) puts( "Program OK!" );
7. else puts( "Syntax Error!" );
8. return 0;
9. }

## 总makefile：

1. compiler: subdir main.o
2. g++ -std=c++11 -O2 -o $@ ../lexer/\*.o ../parse/\*.o main.o
3. main.o: main.cpp main.h
4. g++ -std=c++11 -O2 -c $<
5. subdir:
6. $(MAKE) -C ../lexer
7. $(MAKE) -C ../parse
8. .PHONY : clean
9. clean:
10. $(MAKE) -C ../lexer clean
11. $(MAKE) -C ../parse clean
12. del \*.o

## 注意事项

在将各工程拷贝至相应目录后需要将各自的main函数改为对外接口，并且新生成一个头文件将该该接口提供给外部程序。因为词法分析器和语法分析器的编程标准是相同的，所以在头文件的定义上有许多相同之处，因此在总编译时会出现重定义现象。所以需要将两个工程中相同的部分提取出来作为公共头文件。

所有函数注释详见附录中代码。

# 程序验证

## 实验截图：

|  |  |
| --- | --- |
| 正确例子： |  |
|  |  |
| 错误例子： |  |
|  |  |

\* 实验样例见附录。

# 实验心得

通过此次试验，我完整的完成了一个编译器的前端部分。词法分析器和语法分析器是分开的两个部分，但是通过中间文件的粘连可以使他们很顺畅的结合在一起。由于递归下降文法可以准确的定位错误的原因，所以我选择了递归下降分析法作为语法分析器的算法。

在整合两个程序时出现了一些不曾想到的问题。由于两个程序的代码标准是一样的，所以在整合编译的时候出现了许多的重复定义的错误，还有函数接口未对外公开导致衔接程序无法正常工作，分项目中的makefile中生成可执行文件部分在拷贝过来后未删除导致程序生成冗余文件，中间文件路径设置错误导致无法找到等一系列问题。当这一切细节都解决后，一个完整的编译器前端就完成了。

此专题虽然是将两个以前的专题衔接就可以完成的，但是我依旧做了半天的时间，我重新调整代码规范化了注释修复各类细节bug，使得我对词法分析器和递归下降文法有了更为深刻的理解，对递归下降文法FIRST集合FOLLOW集以及报错方式等有了更为深刻的认识。这同时也锻炼了我工程能力，使我更加熟练的掌握了makefile的编写技巧，更熟练的掌握了此C/C++跨平台开发技术。

目前该项目已上传至我的Github仓库中。

# 附录：

1. include文件夹下为公共头文件及同一定义文件
2. lexer文件夹下为词法分析器
3. parse文件夹下为词法分析器
4. test文件夹下为衔接程序和测试文件
5. test1.txt和test2.txt分别为正确样例和错误样例，tmp.lexer为二元式中间文件
6. compiler.exe为最终生成程序