



|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | 17130130312 | **姓名** | 曹晨瑶 |
| **班级** | **1713013** | **任课教师** | **王献青** |
| **实验名称** | 模块编程 | | |
| **实验学期** | **2018 – 2019 学年第2学期** | | |
| **实验日期** | 2019年5月18日 | **实验地点** | **G334** |
| **报告成绩** |  |  |  |

西安电子科技大学软件学院

# 实验目的

本次实验通过实现一个计算器程序，以熟悉 C++为程序员提供的模块与分别编译相关机制，学会模块化程序设计范型的基本策略，能够灵活运用相应机制，提高编程水平。

# 实验环境

操作系统： Windows10

开发工具： Code::Blocks

# 实验内容

## 桌面计算器

结合第8、9章、习题 8.5.3 的要求，将第6章的桌面计算器按照以下要求改造实现：

1. 采用多个头文件、多个源文件的方式实现完整程序，使得源代码的物理结构和逻辑结构保持一致；

2. 输入中每遇到一个分号或回车(‘\n’)就认为一个完整表达式结束，并将该完整表达式记作一行。在处理过程中累计行数；

3. 对于输入中存在错误时，除了提示错误现象外，还需提示相应的出错行号。

4. 必须支持命令行参数，用以指明从何处读取数据。

- 命令行参数中的第2项开始，指定0~多个输入文件之路径。输入文件就是普通的文本文件，其中预先输入了若干表达式，内容和来自标准输入的完全相同。

- 若命令行参数未指明输入文件，则程序从标准输入设备(cin)读取表达式；

- 若命令行指明了1~N个输入文件，则程序依次从这些文件读入表达式。

# 数据结构与算法设计

## 桌面计算器

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件main.cpp ，定义了 main()函数。

模块2：词法分析器，包括以下两个文件：

lexer.hpp：声明该模块的接口，包括供其他模块使用的函数、变量的非定义声明，类型定义等

lexer.cpp：该模块的实现，包括函数定义，变量定义等。

模块3：语法分析器，包括以下两个文件：

parser.hpp：声明该模块的接口，包括函数、变量的非定义声明，类型定义等。

parser.cpp：该模块的实现，包括函数定义，变量定义等。

模块4：错误处理器，包括以下两个文件：

error.hpp：声明该模块的接口，包括函数、变量的非定义声明，类型定义等。

error.cpp：该模块的实现，包括函数定义，变量定义等。

模块5：符号表，包括以下两个文件：

sym.hpp：声明该模块的接口，包括函数、变量的非定义声明，类型定义等。

sym.cpp：该模块的实现，包括函数定义，变量定义等。

**关键数据结构设计：**

无

**算法1.1 int main()**

作 用：主控函数，也实现对题目所需其他内容的测试。

参 数：int argc,char\*argv[]

返回值：总是返回0。

计算过程：

1. 调用算法1.3，若文件打开，返回1；若未打开，输出错误并返回0
2. 符号表登记预定义的符号pi，e
3. 调用算法1.2，若输入未结束就继续分析下一个表达式；表达式为空，继续分析下一个表达式，表达式为遇到错误，丢弃错误表达式的剩余部分
4. 调用算法算法1.8赋值给value
5. 表达式分析结束后，对curr\_tok进行判断；若有错误，调用算法1.9
6. 输入结束，删除指向标准输入设备或文件的指针input

**算法1.2 Lexer::Token\_value Lexer::get\_token()**

作 用：读取字符，判断类型

参 数：无

返回值：返回当前字符的类型

计算过程：

1. 判断ch。若其最后一行没有以换行或为分号，则返回PRINT；

当前文件结束后，未成功打开下一个文件，则返回END，表示输入结束；

否则，ch为空格且跳过该次循环。

1. 判断当前字符。

若为0，则当前字符种类为END；

若为换行符或分好，则行数加1且返回PRINT；

若为加减乘除、括号或赋值符，则返回其各自的token种类；

若为数字，则赋于number\_value并返回NUMBER；

否则，若为该字符为字母，则判断该字母后的字符是否为字母或数字。

若仍为字母或数字，则记录下来直至其不为字母或数字，并将记录的字符串视为一个整体，以备在table中查找其值。

若为无效字符，则报错并返回ERR。

**算法1.3 int Lexer::parseCommandLine(****int argc, char\* argv[])**

作 用：读取命令行参数，读取文件

参 数：int argc, char\* argv[]

返回值：1或者0

计算过程：

1. 判断命令行参数的个数，若为1，输入数据，返回0；否则，进行如下操作。
2. 将参数列表中所有的文件名放入容器fileNames中，然后接着打开第1个输入文件。
3. 若成功打开，则去除第一个文件名返回1；若打开失败，返回0.

**算法1.4 int Lexer::switch\_input()**

作 用：当前文件结束，打开下一个文件夹

参 数：无

返回值：1或者0

计算过程：

1. 判断fileNames容器是否为空，若空返回1
2. 否则关闭上个文件，接着打开下个文件；然后判断下个文件是否成功打开。
3. 若成功打开，则去除该文件名并返回1；若打开失败，则返回非0值并报错。

**算法1.5**  **void Lexer::skip()**

作 用：若发现有错误，丢弃表达式剩余部分

参 数：无

返回值：无返回值

计算过程：

当表达式输入未结束时，则获取当前字符。判断该字符的种类，若不为PRINT，则继续获取下一个字符，直至其为PRINT，即为分号或换行符时，该错误表达式结束。

**算法1.6**  **double Parser::prim(bool get)()**

作 用：处理初等项

参 数：bool get

返回值：返回双精度度浮点型

计算过程：

1. 如果get为真，调用算法1.2
2. 对curr\_tok进行判断，若为数字，调用算法1.2，返回number\_value;若为符号，

**算法1.7 double Parser::term(bool get)**

作 用：进行乘除运算

参 数：bool get

返回值：返回双精度浮点型

计算过程：

1. 调用算法1.6赋值给left
2. 进入循环，对curr\_tok进行判断，分为乘、除、错误处理三种情况进行分析

**算法1.8 double Parser::expr(bool get)**

作 用：进行加减运算

参 数：bool get

返回值：返回双精度浮点型

计算过程：

1. 调用算法1.7赋值给left
2. 进入循坏,对curr\_tok进行判断，分为加、减、错误处理三种情况进行分析

**算法1.9 double Error::error (const char\*s)**

作 用：进行错误处理

参 数：const char\*s

返回值：双精度浮点型

计算过程：

1. 错误总数加1
2. 输出错误
3. 返回11

# 测试用例与测试结果

## 桌面计算器

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 测试结果 |
| 1 | 3+5 | Line：1 result is 8 |
| 2 | 2\*4 | Line：2result is 8 |
| 3 | 3/0 | Line：3 has error:divide by 0 |
| 4 | 1@@3 | Line: 4 has error :ERR |

# 实验总结

通过本次实验，我了解了模块化编程。在此次实验之前，对于使用命令行参数还不是十分的了解，不能够熟练使用，尤其是对如何打开命令行参数中的文件十分陌生，通过查资料和阅读老师的实验解析逐步摸索出了使用方法。虽然在之前的实验中有提到关于map的知识，但是我并没有使用，直到这次实验才了解了它的机制，学会了它的使用。同时，在实验过程中遇到了许多问题，但是通过和同学的交流都已经解决了，在学习过程中，我们需要和别人交流，这样才能有所提高。在学习面向对象程序设计过程中，我还需要不断地通过编程进行练习，熟能生巧，才能学好这一门课程。