计算机程序设计基础大作业
实验报告

机械 108　　张益铭　　2021010552

2022 年 5 月 24 日

目录

p.s. 点击目录进行跳转:)

1 实现功能及整体框架

1.1 实现功能

本次大作业选题为大数计算器，实现了**基本功能**和**进阶功能**，这两个功能在一个程序中实现。

其中，基本功能能够对用户输入的加、减、乘以及求余数（$+, -, *, \%$）表达式进行计算，取值范围为 int 范围内的整形数字，最终返回一个在 long long 范围内的结果。

进阶功能是实现两个较大数字的乘法运算，理论上可以计算数字位数在 int 范围内的两数相乘，实际可能需要进行输入输出重定向。

1.2 整体框架

此次大作业代码模块化程度较高，主要分为输入处理部分、判断输入表达式合法性、基本功能的实现以及进阶功能的实现。总共有一个 *.h 头文件和五个 *.cpp 源文件，总代码量为 400 行左右，使用 GBK 格式编码，注释、命名、分块较为合理。

1.2.1 函数声明

本次大作业使用的函数均在 my_function.h 中进行声明，其中包含相关库的调用以及部分变量的声明，具体内容如下图所示。

```
#ifndef TERMPROJECT_MY_FUNCTION_H
#define TERMPROJECT_MY_FUNCTION_H

#include <iostream>
#include <string>
#include <cstring>
#include <vector>
#include <deque>

#define MAX_SIZE 10000

using namespace std;

//输入

void InputData();

//基本功能

extern bool isModuloZero;                                   //判断是否对0取模

int getPriority(char s);                                    //获取运算符优先级

bool isLegalInput(string ori_infix);                        //判断输入是否合法

void Infix2Postfix(string ori_infix);                       //中缀表达式转后缀表达式

long long Calculate();                                      //计算后缀表达式

void showName();                                            //显示信息

//进阶功能

void MultipleTwoNum(string num1, string num2);              //模拟竖式计算，每位相乘最后进位

bool isLegalMulInput(string total);                         //判断乘法输入是否合法

#endif
```

图 1: my_function.h

1.2.2 主函数

主函数 main.cpp 内容较简单，内含注释表明个人信息。main 函数调用 showName() 和 Input() 函数，具体功能见后续介绍。

```
1  //
2  //Term Project of Programming Fundamentals.
3  //Created by 张益铭 2021010552 on 4/13/2022.
4  //Copyright (C) 张益铭 2022. All Rights Reserved.
5  //Encoding with GBK.
6  //
7
8  #include "my_function.h"
9
10 int main() {
11     showName();
12
13     Input();
14
15     system("pause");
16     return 0;
17 }
```

1.2.3 输入处理

输入处理工作主要由 Input.cpp 实现。

Input.cpp 里的 Input() 函数（第 3 行）会对输入数据进行处理。用户需要先选择功能类型，1 代表 Basic Function（基本功能），2 代表 Advanced Function（进阶功能）。随后程序会提示用户输入相应表达式，若表达式合法，将会调用相应的计算功能函数，否则会提示“Invalid function type!”。

1.2.4 判断输入合法性

判断输入合法性主要由 Judgement.cpp 实现。

Judgement.cpp 中包含两个 bool 类型函数：isLegalInput() 函数（第 5 行）和 isLegalMulInput() 函数（第 91 行），分别判断基本功能和进阶功能的表达式合法性。若表达式合法，将会返回 true 并进行计算，否则会给出具体出错提示，并返回 false。

1.2.5 基本功能

基本功能主要由 BasicFunc.cpp 实现。

包括 getPriority()（第 11 行），Infix2Postfix()（第 28 行），Calculate()（第 84 行）、showName()（第 133 行）等函数，分别实现获取运算符优先级、将输入的中缀表达式转化为后缀表达式、计算后缀表达式以及显示本人信息的功能。

1.2.6 进阶功能

进阶功能主要由 AdvancedFunc.cpp 实现。

包含 MultipleTwoNum()（第 3 行）函数，负责计算两个大数相乘，主要通过模拟竖式计算实现。

2 设计及实现思路

2.1 显示个人信息

运行该程序后，先调用 showName() 函数，显示个人信息，该函数在 BasicFunc.cpp 中定义，较简单故不再赘述。

```
133 void showName() {
134     cout << "Term Project of Programming Fundamentals.\n"
135         "Created by 张益铭 2021010552.\n"
136         "Copyright (C) 张益铭 2022. All Rights Reserved.\n"
137         "--------------------------------------------------------------------\n\n";
```

图 2: showName()

其次调用 Input()，处理输入。

2.2 处理输入

功能选择的合法的输入为 1 和 2，对应基本和进阶功能。为区分不合法输入，本人使用 string 类变量 whichFunc 来记录用户选择，考虑到可能的错误如下：

a) 输入字符串长度不为 1

b) 字符串长度为 1，但不是 1 或 2

对于 a，直接输出”Invalid function type!”，程序结束；对于 b，使用 switch 来扫描 whichFunc[0]，default 项为输出”Invalid function type!”。

对于上述 switch 函数，case '1' 为基本功能。若表达式合法，即通过 isLegalInput() 的检验，再使用 Infix2Postfix() 将输入的中缀表达式转化为后缀表达式进行计算，若运算中没有出现对 0 取余的情况，则输出结果 res。

```
case '1': {
    string ori_infix;

    cout << "Basic Func Mode.\n"
        "Please enter the expression:\n";
    getline(cin, ori_infix);

    if (isLegalInput(ori_infix)) {
        Infix2Postfix(ori_infix);
        long long res = Calculate();
        if (!isModuloZero) {
            cout << "The result is:\n" << res << endl;
        }
    }
    break;
}
```

case '2' 项对应进阶功能。考虑到 long long 也装不下待计算数字，故定义 string 类变量 num1 和 num2 对应第一和第二个乘数。表达式通过 isLegalMulInput() 的检验后，会被分割为两个乘数。

遇到空格或 * 前使用.push_back() 存入第一个数，遇到空格和 * 后开始存入第二个数。关于是否到第二个数，我定义了 bool 类型变量 isNum2 来判断，部分代码如下。

```
bool isNum2 = false;

for (char i : total) {
        if (i == '*' || i == ' ') {
                isNum2 = true;
        }
        if (!isNum2) {
                num1.push_back(i);
        }
```

```
10          else if (i != '*' && i != ' ') {
11                  num2.push_back(i);
12          }
13  }
```

2.3 基本功能

关于表达式合法性检测，我会在后文解释，现在我们假定输入均为合法输入。

考虑到中缀表达式并不利于计算机处理，并且受到 PA08 中 3039 后缀表达式作业的启发，我先将中缀表达式转化为后缀，再模仿栈的运行方式进行计算。

2.3.1 获取运算符优先级

由 getPriority() 完成。由于遇到'(' 时入栈，直到遇到')' 时将栈内运算符全部弹出，故定义'(' 优先级为 1，'+'、'-' 为 2，'*'、'%' 为 3，')' 最高，为 4，并默认数字优先级为 0。

2.3.2 中缀转后缀

由 Infix2Postfix() 完成。预处理：为处理负数，我将其改为 0 - 正数，即在符号前加入一个 0，这样就能进行负数的输入，并且为了方便转化以及避免不必要的麻烦，我在转化前遍历中缀表达式清除了其中的空格。

转化的主要思路是遍历中缀，遇到数字则存入后缀表达式，遇到运算符则存入 vector 容器 op 中，模拟栈的运行。若遇到的运算符优先级小于上一个，则上一个运算符出栈，存入后缀，该运算符入栈，遍历完成后将栈内元素清空。伪代码示意图如下：

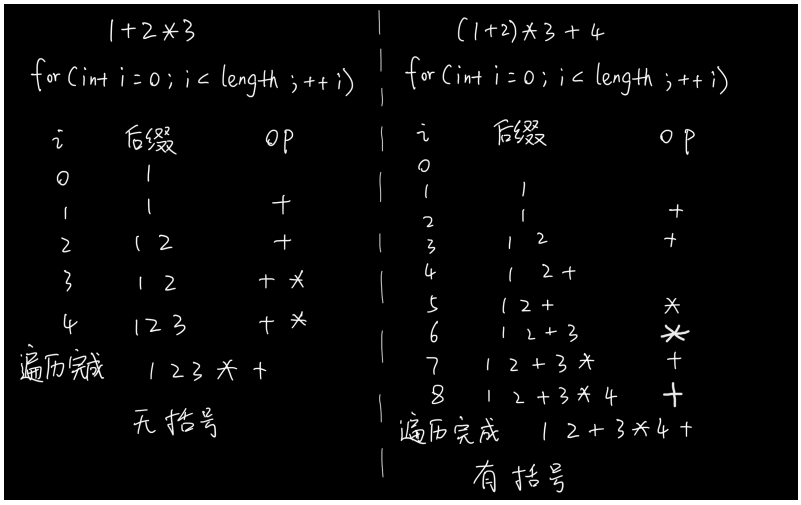


图 3: 中缀转后缀示意图

*转化的后缀表达式存储在 *BasicFunc.cpp* 中的 *char* 型全局数组 *postfix* 中。

2.3.3 计算后缀表达式

由 Calculate() 完成。由于计算的结果在 long long 范围，int 数组并不能满足，故我定义了一个结构体 Stack，用来作为“long long 数组”存储数据，并使用 vector 容器创建 stack 变量 `vector<Stack> stack(MAX_SIZE);`。

```
struct Stack {
        long long s;
};
```

思路是遇到数则压入栈中，遇到运算符则计算栈顶的前两个数字，栈顶元素出栈。为防止对 0 取余，我定义了一个全局 bool 变量 isModuloZero，检测到% 时判断栈顶元素是否为 0，如果为 0 则将 isModuloZero 赋值为 true，并输出“Modulo 0!”，反之正常计算。

考虑到用户可能只输入一个数字，按照上述方法则不能输出该数字，所以在遍历完成时需检测 num 是否为 0，若不是则将 num 作为结果输出。

```
int j = 0;  //模拟控制入栈出栈
for (int i = 0; i < strlen(postfix); ++i) {
    if (postfix[i] >= '0' && postfix[i] <= '9') {
        num = num * 10 + postfix[i] - '0';
    }
    else {
        if (num != 0 || (i >= 1 && postfix[i - 1] == '0')) {
            //防止负号前补充的0被忽略
            stack[j++].s = num;
            num = 0;
        }
        if (j >= 2) {
            ......  //进行运算
        }
    }
}
if (num != 0) {
    stack[j].s = num;
}
```

2.4 进阶功能

进阶功能的实现较为简单，可以通过模拟竖式计算来完成。为了便于队首及队尾元素的增删，我使用了 deque 容器。

2.4.1 各个数位交叉相乘

我先将每位元素相乘到对应数位，最后统一处理进位。

```
for (int i = 0; i < num1.length(); ++i) {
    for (int j = 0; j < num2.length(); ++j) {
        result[i + j] += (num1[i] - '0') * (num2[j] - '0');
    }
}
```

2.4.2 处理进位

关于进位问题，主要思路为遍历未处理进位的 result，定义一个 int 型 carry 变量，存储上一位进上来的数，随后 carry 更新为 result[i] 除以 10，即在十位之上的数字，最后将每一位数先加上 carry 再对 10 取余，至此就完成了进位问题。

但是这种方法有一种不足之处，即无法对最高位进行进位，这也是我使用 deque 容器而不是 vector 的原因，因为 vector 只能处理末尾，不能对首位进行增删操作。

2.4.3 修补最高位

对最高位进行进位也较为简单，大致思路为使用 while 判断 carry 是否为 0，若是则保留 carry 的个位为最高位，将十位以上的数继续向上进位，直到 carry 为 0 为止。

```
while (carry != 0) {
        int temp = carry % 10;
        result.push_front(temp);
        carry /= 10;
}
```

同时为了防止最高位出现 0，即会出现 010*2=020 的情况，所以当结果不为 0 时会使用 while 删除最高位的 0，避免输出出错（此处省略代码）。

p.s. 最开始提到“理论上可以计算数字位数在 int 范围内的两数相乘”，其中的 int 范围限制可以通过改变循环变量 i 的类型，例如改为 long long 类型来实现更多数位的计算。

2.5 程序鲁棒性

~~众所周知，写代码 5 分钟，debug 两小时~~，为了避免各种可能的异常输入导致程序崩溃，我想到了如下了可能出现的异常。

可能的输入异常		
	基本功能	进阶功能
1	是否输入	是否输入
2	第一位不是数字	乘数个数
3	英文/中文符号	乘号个数
4	输入不是数字	是否是乘号
5	缺少/多余/错误运算符	\
6	多/少括号	\
7	对 0 取余	\

注：为了避免不必要的麻烦，在两种检测功能最开始都会遍历输入，删除其中的空格，之后遇到任何一种错误都会给出提示并返回 false，只有通过全部检验才回返回 true，进行后续计算。

2.5.1 基本功能输入异常

基本功能需考虑的异常相对较多，我将按照上述顺序依次介绍相关处理思路。

1. 对于是否输入，只需检测长度是否为 0，若为 0，立即显示“Please enter the expression!”，程序结束。

2. 一般情况下，表达式中第一位都是数字或者括号，只有输入的第一数为负数时才会出现运算符，因此若第一位是“+，*，%”中的一种，都会提示“Operator missing operand!”。但是只输入一个符号也是不行的，因此如果表达式长度为 1 并且第一位是符号也会报错。

3. 由于计算表达式中含有括号，部分用户可能输入中文输入法下的括号，造成输入异常。经测试知，‘（’和‘(’对应的 ASCII 码并不相同，中文每个汉字由两个及以上字节组成，具体表现为其 ASCII 码小于 0，因此在遍历过程中遇到某位的 ASCII 码小于 0 就会提示“Please enter expressions in English!”

4. 当输入不是数字时，则必须是合法运算符中的一种，所以在遍历中使用了 switch 函数判断输入不是数字的情况。

```
if (ori_infix[i] < '0' || ori_infix[i] > '9') {
    switch (ori_infix[i]) {
                case '(':
                case ')':
                case '+':
                case '-':
                case '*':
                case '%':
                        break;
                default:
                        cout << "Invalid operator!\n";
                        return false;
        }
}
```

5. 对于输入运算符错误，可能的情况有：连续多个加减乘取余符号、除了负数之外的左括号连接运算符（不包括连续多个左括号）、运算符连接右括号（不包括连续多个右括号）等问题，发现一种都会输出“Operator missing operand!”

6. 多/少括号的判断则比较简单，只需统计左右两种括号的个数，二者不相等就会输出“Parenthesis DO NOT match!”

7. 关于判断是否对 0 取余，已经在 Calculate() 中进行了介绍，不再赘述。

2.5.2 进阶功能输入异常

进阶功能输入异常相对较少，主要如下。

1. 对于是否是输入，处理方法同基本功能。

2. 若第一位是'*'，说明未输入第一个乘数，会提示“Please enter the first multiplier!”，如果表达式最后一位是'*'，说明未输入第二个乘数，提示“Please enter the second multiplier!”

3. 统计乘号个数，如果不唯一则提示“You can ONLY multiply two numbers!”

4. 由于进阶功能的输入只有数字和乘号，所以对于数字之外的输入，如果是乘号，则乘号个数 +1，否则输出“You can ONLY enter multiplication signs or numbers!”

3 使用说明及实现效果

注：相关输入要求见 readme.txt

3.1 开始

首先，程序会提示用户选择功能类型（1 代表 Basic Function，2 代表 Advanced Function）.

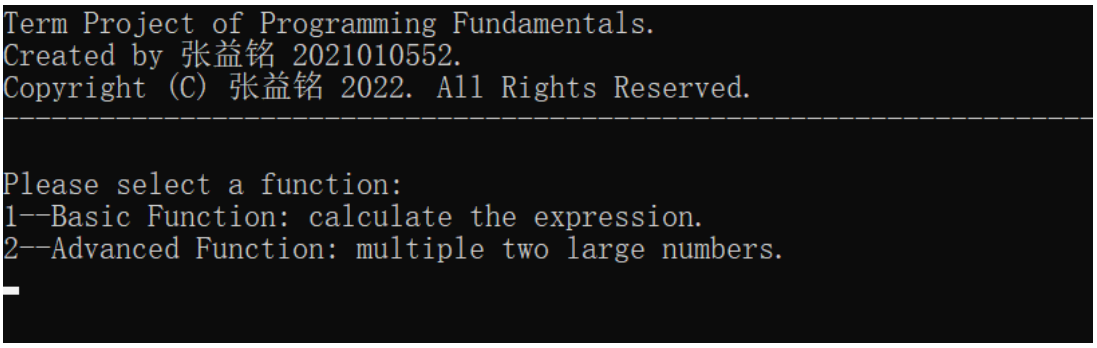


图 4: 选择功能

选择完毕后程序会显示当前的功能，并提示输入表达式。

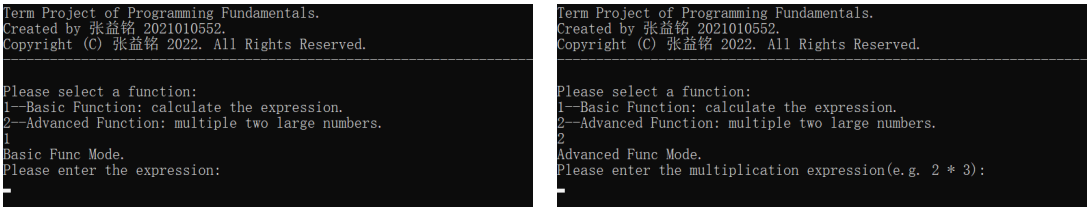


图 5: 两种功能页面

3.2 功能 1：基本功能

用户在基本功能里可以输入加、减、乘、取余表达式，能够进行括号运算，数字内部不可以有空格，两个数之间可以有任意数量的空格，输入完成按下 enter 后，会显示“The result is:”并输出结果，部分输入结果如图。

```
Term Project of Programming Fundamentals.
Created by 张益铭 2021010552.
Copyright (C) 张益铭 2022. All Rights Reserved.
--------------------------------------------------------------------------------

Please select a function:
1--Basic Function: calculate the expression.
2--Advanced Function: multiple two large numbers.
1
Basic Func Mode.
Please enter the expression:
1+2*(-3%4)+5*6
The result is:
25
Press any key to continue . . .
```

```
Term Project of Programming Fundamentals.
Created by 张益铭 2021010552.
Copyright (C) 张益铭 2022. All Rights Reserved.
--------------------------------------------------------------------------------

Please select a function:
1--Basic Function: calculate the expression.
2--Advanced Function: multiple two large numbers.
1
Basic Func Mode.
Please enter the expression:
1324 * 234 + 2345715     %  2134-23145 * 123
The result is:
-2536570
Press any key to continue . . .
```

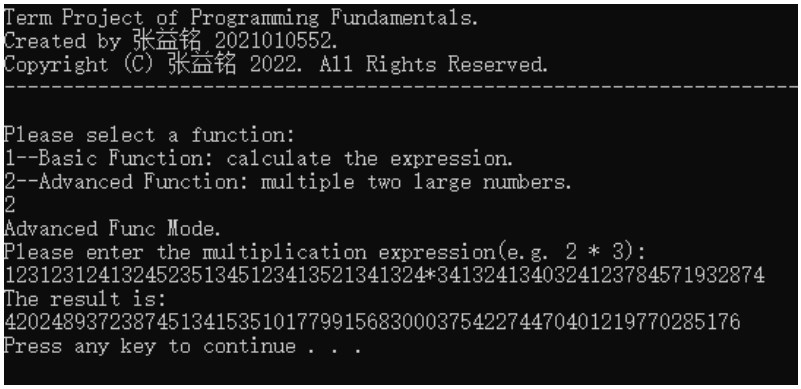
```
Term Project of Programming Fundamentals.
Created by 张益铭 2021010552.
Copyright (C) 张益铭 2022. All Rights Reserved.
--------------------------------------------------------------------------------

Please select a function:
1--Basic Function: calculate the expression.
2--Advanced Function: multiple two large numbers.
1
Basic Func Mode.
Please enter the expression:
21374 - (143+(327685 * 124 -12)%123 + (3244 * 21)) % 124 + 1442
The result is:
22748
Press any key to continue . . .
```

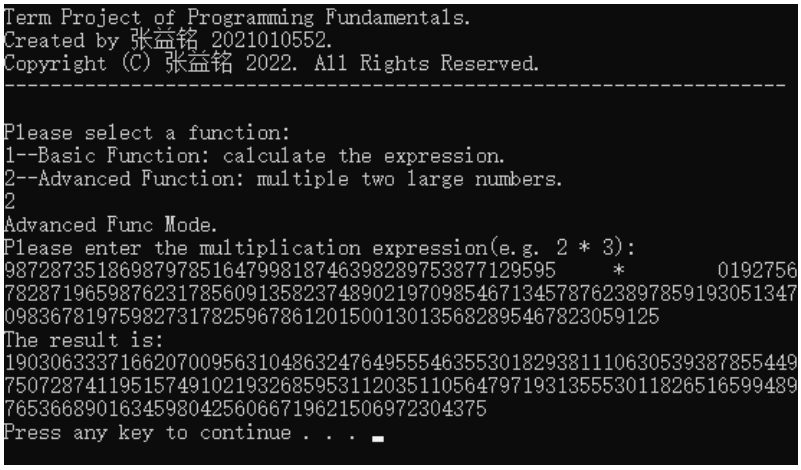
```
Term Project of Programming Fundamentals.
Created by 张益铭 2021010552.
Copyright (C) 张益铭 2022. All Rights Reserved.
--------------------------------------------------------------------------------

Please select a function:
1--Basic Function: calculate the expression.
2--Advanced Function: multiple two large numbers.
1
Basic Func Mode.
Please enter the expression:
(1 + (-2)) * 5% 2 - 4
The result is:
-5
Press any key to continue . . .
```

图 6: 部分输入案例

3.3 功能 2：进阶功能

进阶功能用户可计算两个大数的乘法，其中数字的位数应该在 int 范围内（远远满足作业 100000 位的要求），且两个数应均为非负数，输入完成按下 enter 后，会显示“The result is:”并输出结果，部分输入结果如图。





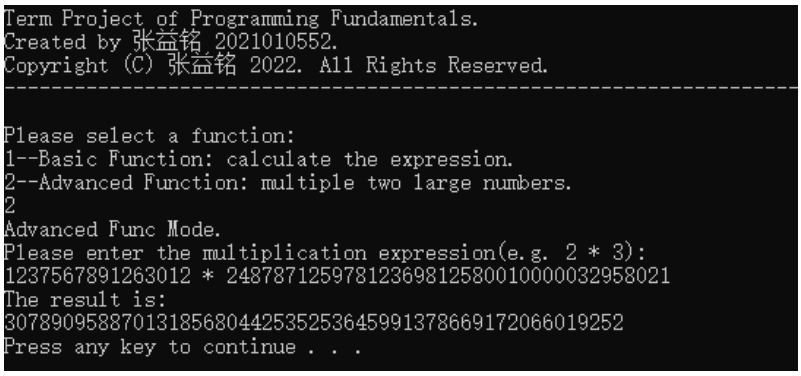


图 7: 部分输入案例