

# 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

<b>1</b>	<b>实现功能及整体框架</b>	<b>3</b>
1.1	实现功能	3
1.2	整体框架	3
1.2.1	函数声明	3
1.2.2	主函数	4
1.2.3	输入处理	4
1.2.4	判断输入合法性	4
1.2.5	基本功能	5
1.2.6	进阶功能	5
<b>2</b>	<b>设计及实现思路</b>	<b>5</b>
2.1	显示个人信息	5
2.2	处理输入	5
2.3	基本功能	7
2.3.1	获取运算符优先级	8
2.3.2	中缀转后缀	8
2.3.3	计算后缀表达式	8
2.4	进阶功能	9
2.4.1	数据分割	10
2.4.2	计算两数相乘	10
2.5	程序鲁棒性	11
2.5.1	基本功能输入异常	11
2.5.2	进阶功能输入异常	13

<b>3</b>	<b>使用说明及实现效果</b>	<b>13</b>
3.1	开始 . . . . .	13
3.2	功能 1: 基本功能 . . . . .	14
3.3	功能 2: 进阶功能 . . . . .	16

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，数字的取值范围为 int 范围内的整形数字，最终结果需在 long long 范围内。

进阶功能是实现两个较大数字的乘法运算，对于较大的数字需要进行输入输出的重定向。

注：两种功能理论上能够计算任意长度的表达式，本程序中 MAX\_SIZE 取值为 1000000，完全满足本次作业需求，更改 MAX\_SIZE 的大小可以实现更大的计算。

### 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 560 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，主要内容如下图所示。

```
10 using namespace std;
11
12 #define MAX_SIZE 1000000
13
14 struct LongLongNum {
15     long long s;
16 };
17
18 //输入
19
20 void Input();
21
22 //基本功能
23
24 extern bool isModuloZero;
25
26 int getPriority(char s);
27
28 bool isLegalInput(string ori_infix);
29
30 char* Infix2Postfix(string ori_infix);
31
32 long long Calculate(char* postfix);
33
34 void showName();
35
36 //进阶功能
37
38 int Str2Int(string num, LongLongNum* arr);
39
40 void MultipleTwoNum(LongLongNum* num1, LongLongNum* num2, int num1_len, int num2_len);
41
42 int getLen(long long num);
43
44 bool isLegalMulInput(string total);
```

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，就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 5 行)、Infix2Postfix() (第 22 行)、Calculate() (第 92 行)、showName() (第 145 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 Str2Int() (第 3 行)、MultipleTwoNum() (第 59 行)、getLen() (第 131 行) 等函数, 分别实现将字符串转化为分割后的数组, 返回首位位置、计算两数相乘、获取每组数字长度的功能。

## 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。

```
1  case '1': {
2      string ori_infix;
3
4      cout << "Basic Func Mode.\n"
5           "Please enter the expression:\n";
6      getline(cin, ori_infix);
7
8      if (isLegalInput(ori_infix)) {
9          //表达式转换并计算
10         long long res = Calculate(Infix2Postfix(ori_infix));
11         if (!isModuloZero) {
12             cout << "The result is:\n" << res << endl;
13         }
14     }
15     break;
16 }
```

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。

由于输入后就删除了其中的空格, 所以遇到 \* 前使用 push\_back() 存入第一个数, 遇到 \* 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。

```
1  bool isNum2 = false;
2
3  for (char i : total) {
4      if (i == '*') {
5          isNum2 = true;
6      }
7      if (!isNum2) {
8          num1.push_back(i);
```

```
9     }
10     else if (i != '*') {
11         num2.push_back(i);
12     }
13 }
```

随后我申请了两块空间，用于存放分割后的数字，利用指针传递数组，最后进行计算。

```
1  LongLongNum* num1_arr = new LongLongNum[MAX_SIZE];
2  LongLongNum* num2_arr = new LongLongNum[MAX_SIZE];
3
4  for (int i = 0; i < MAX_SIZE; ++i) {    //初始化
5      num1_arr[i].s = 0;
6      num2_arr[i].s = 0;
7  }
8  //获取倒序后最高位数字位置
9  int num1_len = Str2Int(num1, num1_arr);
10 int num2_len = Str2Int(num2, num2_arr);
11
12 MultipleTwoNum(num1_arr, num2_arr, num1_len, num2_len);
13
14 delete[] num1_arr;
15 delete[] num2_arr;
```

注：LongLongNum 为 my\_function.h 中定义的结构体，用作存储 long long 类型的“数组”。

```
1  struct LongLongNum {
2      long long s;
3  };
```

## 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：为了统一计算，我将正数改为 0+ 本身，负数改为 0-它的相反数，同时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：

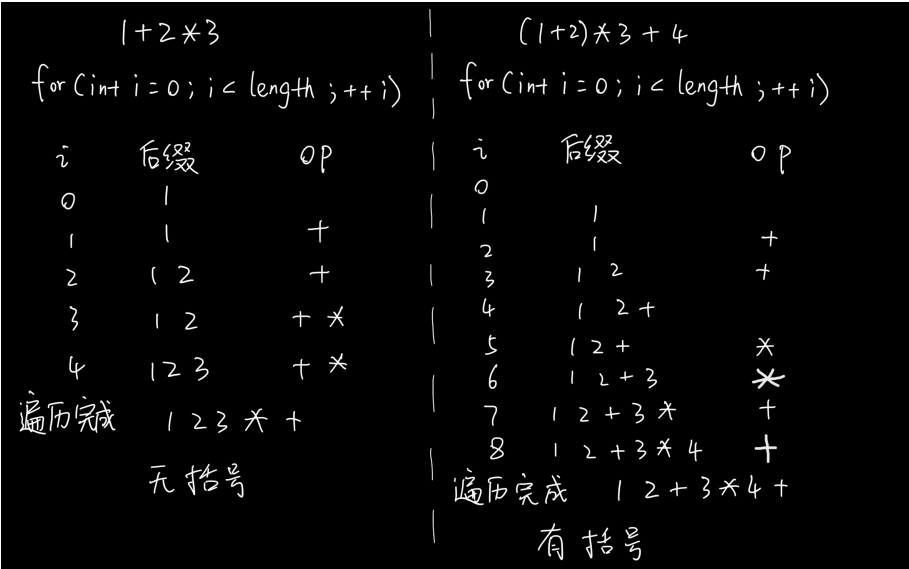


图 3: 中缀转后缀示意图

\* 转化的后缀表达式存储在动态分配的数组 `postfix` 中。

```
char* postfix = new char[MAX_SIZE];
```

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，故我利用 `LongLongNum` 结构体存放 `stack` 变量：

```
LongLongNum* stack = new LongLongNum[MAX_SIZE];。
```



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。

```
1  int j = 0; //模拟控制入栈出栈
2  for (int i = 0; i < strlen(postfix); ++i) {
3      if (postfix[i] >= '0' && postfix[i] <= '9') {
4          num = num * 10 + postfix[i] - '0';
5      } else {
6          if (num != 0 || (i >= 1 && postfix[i - 1] == '0')) {
7              //防止负号前补充的0被忽略
8              stack[j++].s = num;
9              num = 0;
10         }
11         if (j >= 2) {
12             ..... //进行运算
13         }
14     }
15 }
16 if (num != 0) {
17     stack[j].s = num;
18 }
```

## 2.4 进阶功能

进阶功能经历了两次大改，起初我模拟列竖式计算的过程，每次计算两个数位上的数字相乘，但这样计算复杂度太高，算到一万位乘一万位就已经超时了，具体如下：

```
1  for (int i = 0; i < num1.length(); ++i) {
2      for (int j = 0; j < num2.length(); ++j) {
3          result[i + j] += (num1[i] - '0') * (num2[j] - '0');
4      }
5  }
```

后来我发现这种容器每个位置只存储了一位数字，利用率不高。由于 long long 类型范围为 [-9223372036854775808, 9223372036854775807]，能够完整表示 18 位数字，即两个九位数相乘的最大长度，因此我把待计算的数字每九位分为一个小节，一次相乘两个小节的数据，提高计算速度。

注：在本地测试中，更改算法后计算十万位乘十万位需要 0.678000s，相比之前的 22.375000s，极大地提升了运算速度（不同电脑测试时间略有差异）。

2.4.1 数据分割

为了防止相乘后数字进位导致头部溢出，我将数字倒序分割，最后九位放在 arr[0]，头部放在 arr 结尾，以此类推。例如 12123456789987654321 就被分割为了

987654321	123456789	12
arr[0]	arr[1]	arr[2]

主要思路为对于能被分割的部分，相应数位乘十亿、一亿、一千万等组合成一个数字，然后将结果放入数组中，对于不够九位的部分再具体分析。

最后，还要注意头部可能存在 0，譬如输入 00000123456789，就需要去除首位的 0，返回正确的首位位置。

2.4.2 计算两数相乘

首先我申请了一块空间用于存储结果

```
LongLongNum* res = new LongLongNum[MAX_SIZE];
```

主要计算思路依然类似于竖式乘法，只不过竖式乘法每次乘两位数字，这里每次乘两个小节的数字。具体实现依旧是各个数位交叉相乘，乘完一组后直接向上进位，核心代码如下：

```
1  for (int i = 0; i < num1_len; ++i) {
2      carry = i;
3      for (int j = 0; j < num2_len; ++j) {
4          res[carry].s += num1[i].s * num2[j].s; //交叉相乘
5          if (res[carry].s >= 1000000000) {
6              res[carry + 1].s += res[carry].s / 1000000000;
7              res[carry].s %= 1000000000; //进位
8          }
9          ++carry;
```

```
10     }
11 }
```

还需注意的是每组数据是用 long long 而不是 string 保存的，因此如果中间数位不足 9 位，则需要补 0，否则会出现缺 0 的情况。所以我写了一个简单的函数 getLen() 来获取每小节数字长度，该函数如下：

```
1  int getLen(long long num) {
2      int len = 0;
3      while (num > 0) {
4          num /= 10;
5          ++len;
6      }
7      return len;
8  }
```

除去首位后（首位无需补 0），如果 res[i] 的长度小于 9，则需要先输出对应多个 0，然后再输出 res[i]，保证结果准确性。

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，两种功能都会删除表达式中的空格。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，我将按照上表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，因此若第一位是 “\*, %” 中的任何一种，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。

```
1  if (ori_infix[i] < '0' || ori_infix[i] > '9') {  
2      switch (ori_infix[i]) {  
3          case '(':  
4          case ')':  
5          case '+':  
6          case '-':  
7          case '*':  
8          case '%':  
9              break;  
10         default:  
11             cout << "Invalid operator!\n";  
12             return false;  
13         }  
14     }
```

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、除了正负号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，发现任意一种都会输出 “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，如果不是 1 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 “Invalid signs!”

## 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）。

```
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.  
_
```

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。

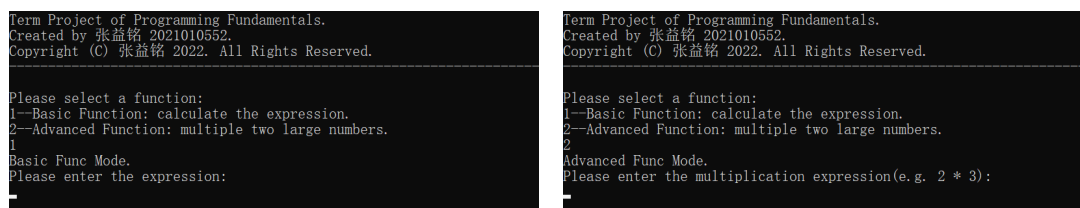


图 5: 两种功能页面

### 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。

```
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.
2
Advanced Func Mode.
Please enter the multiplication expression(e.g. 2 * 3):
123123124132452351345123413521341324*3413241340324123784571932874
The result is:
420248937238745134153510177991568300037542274470401219770285176
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.
2
Advanced Func Mode.
Please enter the multiplication expression(e.g. 2 * 3):
987287351869879785164799818746398289753877129595 * 0192756
782871965987623178560913582374890219709854671345787623897859193051347
098367819759827317825967861201500130135682895467823059125
The result is:
190306333716620700956310486324764955546355301829381110630539387855449
750728741195157491021932685953112035110564797193135553011826516599489
765366890163459804256066719621506972304375
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.
2
Advanced Func Mode.
Please enter the multiplication expression(e.g. 2 * 3):
1237567891263012 * 24878712597812369812580010000032958021
The result is:
30789095887013185680442535253645991378669172066019252
Press any key to continue . . .
```

图 7: 部分输入案例