


计算器小程序设计

注意：一下文本为旧版本操作方法，新版本函数不再使用 $()$ 而改用 $\{ \}$

例如：使用函数时，输入 $\sin\{3.14*2\}$ ，而非上个版本的 $\sin(3.14*2)$

使用方法

1. 基本加减乘除四则运算

 计算器 ×

输入表达式：

1+2*3-4/2


计算结果：

5.000000

计算

退出

2. 支持多位整数、小数的计算

 计算器 ×

输入表达式：

23.6-3.6+1.2*2+2.1/0.7


计算结果：

25.400000

计算

退出

3. 支持负数运算，支持在数前带有+号

 计算器 ×

输入表达式：


-4.4+52.9

计算结果：

48.500000

计算

退出

 计算器 ×


输入表达式:

计算结果:

计算

退出

4. 支持括号内的数优先计算

 计算器 ×


输入表达式:

计算结果:

计算

退出

5. 支持乘方、百分号的计算

 计算器 ×


输入表达式:

计算结果:

计算

退出

6. 支持函数计算

 计算器 ×

输入表达式:

计算结果:

计算

退出

7. 支持报错功能

- 格式错误报错

计算器

输入表达式:

计算结果:

计算

退出

计算器

输入表达式:

计算结果:

计算

退出

计算器

输入表达式:

计算结果:

计算

退出

计算器

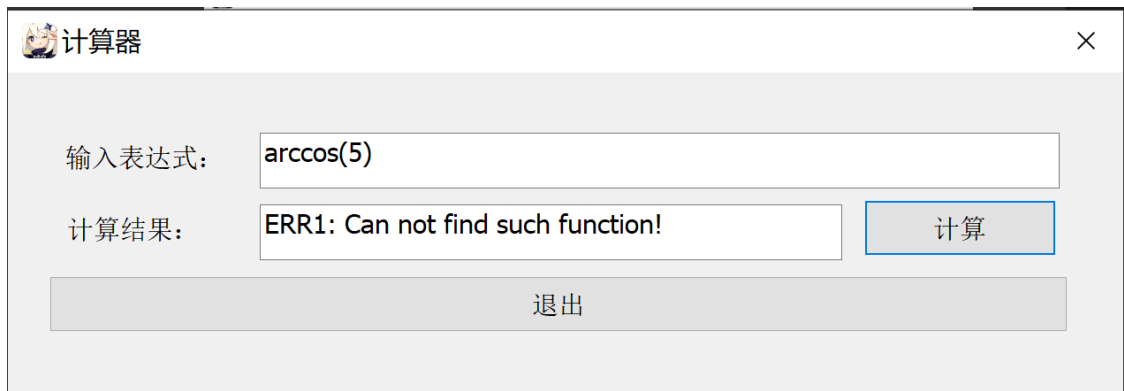
输入表达式:

计算结果:

计算

退出

- 无法识别的函数报错



设计思路

关于四则运算

本部分课本已经明确，此处不再赘述

关于小数与多位整数处理

```
if (*temp >= '0' && *temp <= '9')    //读入数字
{
    double cache;                    //建立一个缓存
    int cnt = 0;                      //统计数的位数
    cache = multy_number_in(&temp, &cnt);
    if (*temp == '.')                //是小数
    {
        temp++;
        cnt = 0;                    //重新计数
        cache += multy_number_in(&temp, &cnt) * pow(0.1, cnt);
        OVS.ins_sq_stack(cache*flag);    //入栈
    }
    else OVS.ins_sq_stack(cache*flag);
    flag = 1;                        //flag用于负号判断
}

/*
* 功能：从str循环读入字符直到遇见非数字字符结束，将读入到的数按多位数形式返回，并返回多位数的位数。
* 参数：**temp：指向当前读入指针的指针。*cnt：指向计数器cnt的指针。
* 返回：从str读入的多位数。
*/
double multy_number_in(char** temp, int* cnt)
{
    double number;    //要返回的数
    number = **temp - '0';
    (*cnt)++;
    while (1)
    {
        (*temp)++;
        if (**temp >= '0' && **temp <= '9') //下一位是数字
```

```

    {
        number = number * 10 + (**temp - '0');
        (*cnt)++;
    }
    else break;    //返回后指针指向非数字字符
}
return number;
}

```

多位整数读取原理为通过指针的递进一直读到非数字的项。而多位小数可以分为整数与小数部分，两个部分均可按照多位整数读取，无非小数部分=pow(0.1,小数位数)

关于符号的处理

符号处理较为繁琐，这里需要使用重重判断将负号与减号，正号与加号区分开来。

```

if (*temp == '-' && (OPS.read_sq_stack() == ';' || OPS.read_sq_stack() == '(') && (*
(temp - 1) < '0' || *(temp - 1) > '9') && *(temp - 1) != ')' && *(temp - 1) != '%')
//为负号
{
    flag = -1;
    temp++;
}
else if (*temp == '+' && (OPS.read_sq_stack() == ';' || OPS.read_sq_stack() == '(')
&& *(temp - 1) < '0' || *(temp - 1) > '9') && *(temp - 1) != ')' && *(temp - 1) !=
'%') //为正号
{
    temp++;
}

```

关于乘方与百分号的计算

这两个处理较为简单

乘方可以理解为优先级最高的双目运算符

百分号则为单目运算符

关于函数计算

这里函数采用<math.h>中的函数。读入相应的函数名（字符串），与函数库中已有函数进行对比，返回相应的函数代号，进而判断该执行哪个函数

在文件func_config中可以自行添加需要的函数，加入其他函数也只是工作量的问题，故本作品只存入3个最常用的函数，使用者可以根据需要添加<math.h>中自己需要的函数

```

else if (*temp >= 'a' && *temp <= 'z')    //可能是函数
{

```

```

func_buffer_ptr = func_buffer;           //指针指向buffer首部
while (1)
{
    if (*temp == '(') break;
    if (*temp == ';') goto ERR0;
    *func_buffer_ptr = *temp;
    temp++;
    func_buffer_ptr++;
}
*func_buffer_ptr = '\0';                 //变为字符串
func_numb = func_config_func_find(func_buffer); //查找函数
if (func_numb == -1)
{
    *sta = "ERR1: Can not find such function!";
    break;
}
}

const char* func_list[] = { "sin", "cos", "tan" }; //可用的函数列表

/*
 * 函数查找函数
 */
int func_config_func_find(char* str)
{
    int i;
    int flag = 0;
    for (i = 0; i < 3; i++)
    {
        if (strcmp(func_list[i], str) == 0)
        {
            flag = 1;
            break;
        }
    }
    if (flag) return i;
    else return -1;
}

/*
 * 函数使用函数
 */
double func_config_func_use(int func_numb, double x)
{
    switch (func_numb)
    {
        case 0: return sin(x);
        case 1: return cos(x);
        case 2: return tan(x);
        default: return -1;
    }
}

```

关于错误警告

函数错误警告：在上边函数代码中给出

格式错误警告：

```
else
{
    ERR0:    //用作上边代码goto的标签
    *sta = "ERR0: Invalid format!";
    break;
}
```

MFC设计

点击计算按钮时操作

```
void Cabcdlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(true);
    CString inputCstr;
    CString outputCstr;

    char *sta = new char[100];           //字符串，用于存储错误报告
    GetDlgItem(IDC_EDIT_EQ)->GetWindowTextW(inputCstr);           //读入数据
    USES_CONVERSION;
    CString end(";");                     //在末尾自动添加';'
    inputCstr += end;
    char* sInput = w2A(inputCstr);
    double output = calculator(sInput,&sta);
    if (sta[0] == '\\0')                   //如果成功计算，返回double
    {
        outputCstr.Format(_T("%f"), output);
    }
    else                                   //出错，返回错误警告
    {
        outputCstr = A2W(sta);
    }

    GetDlgItem(IDC_EDIT_EQ2)->SetWindowTextW(outputCstr);

    UpdateData(false);
}
```

git的使用

本次作业代码量较大，开发过程较为复杂，因此使用git进行版本管理

```
Administrator@PC-202303041131 MINGW64 /e/my/code/cpp/中缀表达式 (master)
$ git log
commit 41a4a48df77a55c0b8f3d9a6d4fcf9566612ccb4 (HEAD -> master)
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Mon Mar 18 16:22:21 2024 +0800

    v3

commit c50171a58ffc8de4e1a66867ecf6b296a0267224
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Sat Mar 16 01:55:04 2024 +0800

    v3_修复部分bug，增添函数运算

commit dd5072bae0c94887cf41656c87e3bf5d3a1041b6
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Fri Mar 15 19:17:15 2024 +0800

    v2_修复部分bug并增添负数运算

commit d4be069f687dbf9ba1ec2f3ee6a1022532e29a3b
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Fri Mar 15 17:36:01 2024 +0800

    v1_增添计算多位数与小数的功能
:...skipping...
commit 41a4a48df77a55c0b8f3d9a6d4fcf9566612ccb4 (HEAD -> master)
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Mon Mar 18 16:22:21 2024 +0800

    v3

commit c50171a58ffc8de4e1a66867ecf6b296a0267224
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Sat Mar 16 01:55:04 2024 +0800

    v3_修复部分bug，增添函数运算

commit dd5072bae0c94887cf41656c87e3bf5d3a1041b6
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Fri Mar 15 19:17:15 2024 +0800

    v2_修复部分bug并增添负数运算

commit d4be069f687dbf9ba1ec2f3ee6a1022532e29a3b
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Fri Mar 15 17:36:01 2024 +0800

    v1_增添计算多位数与小数的功能

commit 71b740d59eefee97c8750407cc2445097d564a13
Author: dongzai1952 <dongzai1952@outlook.com>
Date: Fri Mar 15 16:04:55 2024 +0800
```