成绩：_____

西安建筑科技大学

设计说明书（报告）

题　　目	基于MFC的科学计算器
课程名称	面向对象与可视化程序设计
学生姓名	
学　　号	2006030113
院（系）	信息与控制工程学院
专　　业	通信工程
时　　间	2023年　02月　26日

目录

基于 MFC 的科学计算器

1 需求分析

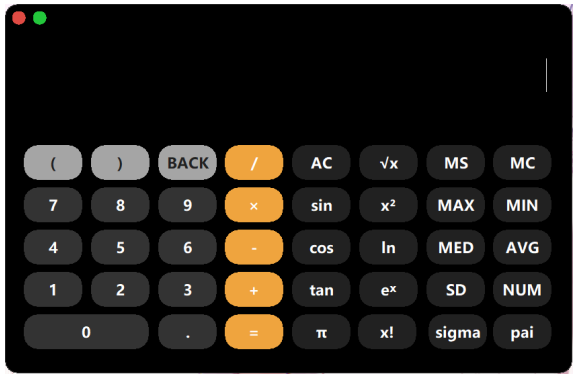
1．实现用户可交互的界面
2．实现整数和小数加法，减法，乘法，除法的运算
3．能计算任意表达式的值，包括加减乘除混合运算，带括号的混合运算，带三角函数的等科学计算函数的混合运算
4．实现包括正弦，余弦，正切，开方，乘方，指数，对数计算
5．实现整数的阶乘计算
6．实现删除和清空的功能
7．实现数据存储和统计功能，统计计算包括最大值，最小值，中值，平均值，标准差，累加，累乘

2 总体设计

由于计算器界面比较简单这里采用 Microsoft Visual Studio 2022 开发，其中选择 MFC 应用程序、基于对话框的开发。

2.1 用户界面设计

在对原版 MFC 界面进行重构的过程中，我们发现原版 MFC 界面的设计较为简陋，缺乏美感，且与现代应用程序的设计风格不符。因此，在重构过程中，我们继承了 CButton 类并重新定义了绘制按钮的函数，使得按钮看起来更加圆润，不同类型的按钮采用不同的颜色，从而增强了用户体验。此外，我们还取消了调用系统菜单栏，并采用按钮的方式来实现关闭和最小化窗口，这不仅使得界面更加美观，而且更加符合现代应用程序的设计风格。最后，我们定义了整体的窗口绘制函数，使得窗口由原来的白色直角变为黑色圆角，进一步提高了整体的美观性和用户体验。总体用户界面如下



2.2 按钮含义：

- **AC**：清零
- **BACK**：删除上一次输入

- MS：添加计算结果到统计运算中
- MC：删除当前统计运算的全部数据
- MAX：求最大值
- MIN：求最小值
- MED：求中位数
- AVG：求平均数
- SD：求方差
- NUM：当前存储数据的个数
- Sigma：求和
- Pai：求累乘

2.3 算法设计

完整的计算器涉及的算法较为复杂，涉及到高精度浮点型计算，括号匹配（双栈），函数名称识别等。在阶乘运算中，还需要使用多线程的方法来提高大数阶乘的计算。而高精度除法运算还需要使用牛顿迭代法进行计算。这里尝试了重新定义运算符实现高精度四则运算。以下截取了部分代码。

```
BigNum operator/(const BigNum& b) const {
    int len1 = len - dot, len2 = b.len - b.dot;
    int dot1 = dot, dot2 = b.dot;
    if (len2 == 0 || (len2 == 1 && b.s[dot2] == '0')) {
        // 除数为0或1的情况
        throw runtime_error("Division by zero");
    }
    if (len1 < len2) {
        return BigNum();
    }
    BigNum c;
    c.len = len1 - len2 + 1;
    c.dot = dot1 - dot2;
    for (int i = len1 - 1; i >= dot1; i--) {
        c = c * 10;
        c.s[0] = s[i];
    }
    BigNum d;
    d.len = len1 - len2 + 1;
    d.dot = 0;
    for (int i = len1 - len2; i >= 0; i--) {
        while (c >= b) {
            int k = 0;
            while (c >= (b * (k + 1))) {
                k++;
            }
            d.s[i] = k;
```

```
            c = c - b * k;
        }

    }
    d.clean();
    return d;
}
```

但是由于还存在少许漏洞，算法没有达到预期效果。于是采用 C++调用 Python 的方法，使用 Python math 库的 eval 解决计算问题。

还涉及到统计相关的计算，这里使用容器保存计算结果，并设计了相应的函数，一起封装在 DoubleContainer()类中。

2.4 用户交互算法设计

这里使用 string 类型保存内容，在用户按下普通按键时，往字符串中添加一个字符。删除时，识别最后一个字符类型，进行删除，保证能只删除一个字符和输入错误的完整函数字符。每次字符内容有所改变就直接更新现实，这样能方便维护用户交互界面的正常显示。

3 功能详细设计

3.1 用户界面设计

3.1.1 对话框设计

这里使用 OnEraseBkgnd 消息重新绘制对话框，使得对话框变为圆角和黑色。这里需要在对话框属性中设置边框为 None。具体实现的函数如下

```
BOOL CCalculatorDlg::OnEraseBkgnd(CDC* pDC)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    CRect rect;
    GetClientRect(rect);
    CRgn myrgn1;
    myrgn1.CreateRoundRectRgn(0, 0, rect.Width(), rect.Height(), 35
, 35);
    CBrush frameBrush, bgbrush;
    bgbrush.CreateSolidBrush(RGB(0, 0, 0));
    frameBrush.CreateSolidBrush(RGB(51, 51, 51));
    pDC->FillRgn(&myrgn1, &bgbrush);
    pDC->FrameRgn(&myrgn1, &frameBrush, 1, 1);
    return true;
    return CDialogEx::OnEraseBkgnd(pDC);
}
```

3.1.2 文本框设计

如果对话框变为黑色，使用默认白色对话框不美观。通过 OnCtlColor 消息，重设文本框的颜色和文字。这里有个特殊的点，就是我们设置了文本框为只读类型，这样方便输入的统一管理。而只读文本框默认标志为 CTLCOLOR_STATIC 如果使用 CTLCOLOR_EDIT 会导致无法更改出想要的结果。具体实现如下

```
HBRUSH CCalculatorDlg::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
    HBRUSH hbr = CDialogEx::OnCtlColor(pDC, pWnd, nCtlColor);
    switch (nCtlColor)
    {
    case CTLCOLOR_EDIT:
    case CTLCOLOR_STATIC:
        switch (pWnd->GetDlgCtrlID())
        {
        case IDC_EDIT1:
            pDC->SetBkColor(m_color);
            pDC->SetTextColor(m_textcolor);
            pDC->SetTextAlign(TA_RIGHT & TA_BOTTOM & TA_UPDATECP);

            hbr = (HBRUSH)m_brush;
            break;
        default:
            hbr = CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
            break;
        }
        break;
    }
    return hbr;
}
```

除此之外，还需要重新定义文本框中文字格式。这里使用 CFont 类实现，详细请见附录。

3.1.3 按钮设计

为了实现美观的按钮，这里定义了 CRoundButton 类，继承 CBuntton 类。其中包含两个函数 CRoundButton::PreSubclassWindow()用来定义按钮的有效大小，CRoundButton::DrawItem 函数用来定义按钮显示的颜色形状字体等，这里通过私有变量 Kind 来识别不同的按钮，以此来配置不同颜色。具体代码如下：

```
#include "pch.h"
#include "CRoundButton.h"

void CRoundButton::PreSubclassWindow()
{
```

```
6.      // TODO:  在此添加专用代码和/或调用基类
7.      ModifyStyle(0, BS_OWNERDRAW);
8.      CButton::PreSubclassWindow();
9.  }
10. void CRoundButton::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
11. {
12.
13.     // TODO:  添加您的代码以绘制指定项
14.     CRect rct = lpDrawItemStruct->rcItem;
15.
16.     CRgn coverrgn;
17.     coverrgn.CreateRectRgn(0, 0, rct.Width(), rct.Height());
18.     CDC* pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
19.     int nSaveDC = pDC->SaveDC();
20.     CBrush backBrush;
21.     backBrush.CreateSolidBrush(RGB(0, 0, 0));
22.     pDC->FillRgn(&coverrgn, &backBrush);
23.
24.     CRgn keyrgn;
25.     if (kind==39||kind==40) {
26.         keyrgn.CreateEllipticRgn(rct.CenterPoint().x - 11, rct.Cent
    erPoint().y - 11, rct.CenterPoint().x + 11, rct.CenterPoint().y + 1
    1);
27.     }
28.     else {
29.         keyrgn.CreateRoundRectRgn(rct.TopLeft().x, rct.TopLeft().y,
     rct.BottomRight().x, rct.BottomRight().y, min(rct.Width(), rct.Hei
    ght()) / 2 + 15, min(rct.Width(), rct.Height()) / 2 + 15);
30.     }
31.     CBrush keyBrush;
32.     if (lpDrawItemStruct->itemState & ODS_SELECTED)
33.     {
34.         if (kind >=14&&kind<=18) {
35.             keyBrush.CreateSolidBrush(RGB(254, 254, 254));
36.             pDC->SetTextColor(RGB(239, 164, 62));
37.         }
38.         else if (kind >= 0 && kind <= 10 ) {
39.             keyBrush.CreateSolidBrush(RGB(254, 254, 254));
40.             pDC->SetTextColor(RGB(51, 51, 51));
41.         }
42.         else if (kind == 39) {
43.             keyBrush.CreateSolidBrush(RGB(150, 150, 150));
44.         }
45.         else if (kind == 40) {
```

```
46.             keyBrush.CreateSolidBrush(RGB(150, 150, 150));
47.         }
48.         else if (kind <= 38 && kind >= 19) {
49.             keyBrush.CreateSolidBrush(RGB(254, 254, 254));
50.             pDC->SetTextColor(RGB(33, 33, 33));
51.         }
52.         else if (kind <= 13 && kind >= 11) {
53.             keyBrush.CreateSolidBrush(RGB(33, 33, 33));
54.             pDC->SetTextColor(RGB(165, 165, 165));
55.         }
56.         else {
57.             keyBrush.CreateSolidBrush(RGB(0, 0, 0));
58.             pDC->SetTextColor(RGB(0, 0, 0));
59.         }
60.     }
61.     else {
62.         if (kind >= 14 && kind <= 18) {
63.             keyBrush.CreateSolidBrush(RGB(239, 164, 62));
64.             pDC->SetTextColor(RGB(254, 254, 254));
65.         }
66.         else if (kind >= 0 && kind <= 10) {
67.             keyBrush.CreateSolidBrush(RGB(51, 51, 51));
68.             pDC->SetTextColor(RGB(254, 254, 254));
69.         }
70.         else if (kind == 39) {
71.             keyBrush.CreateSolidBrush(RGB(224, 75, 68));
72.             pDC->SetTextColor(RGB(254, 254, 254));
73.         }
74.         else if (kind == 40) {
75.             keyBrush.CreateSolidBrush(RGB(35, 199, 60));
76.             pDC->SetTextColor(RGB(254, 254, 254));
77.         }
78.         else if (kind <= 38 && kind >= 19) {
79.             keyBrush.CreateSolidBrush(RGB(33, 33, 33));
80.             pDC->SetTextColor(RGB(254, 254, 254));
81.         }
82.         else if (kind <= 13 && kind >= 11) {
83.             keyBrush.CreateSolidBrush(RGB(165, 165, 165));
84.             pDC->SetTextColor(RGB(33, 33, 33));
85.         }
86.         else {
87.             keyBrush.CreateSolidBrush(RGB(0, 0, 0));
88.             pDC->SetTextColor(RGB(0, 0, 0));
89.         }
```

```
90.
91.     }
92.     pDC->FillRgn(&keyrgn, &keyBrush);
93.
94.     pDC->SetBkMode(TRANSPARENT);
95.
96.     CString strText{};
97.     CFont font;
98.     //font.CreatePointFont(180, L"幼圆");//设置控件文字大小与字体
99.     font.CreateFont(
100.            30,                        // nHeight
101.            0,                         // nWidth
102.            0,                         // nEscapement
103.            0,                         // nOrientation
104.            FW_BOLD,                  // nWeight
105.            FALSE,                     // bItalic
106.            FALSE,                     // bUnderline
107.            0,                         // cStrikeOut
108.            ANSI_CHARSET,              // nCharSet
109.            OUT_DEFAULT_PRECIS,        // nOutPrecision
110.            CLIP_DEFAULT_PRECIS,       // nClipPrecision
111.            DEFAULT_QUALITY,           // nQuality
112.            DEFAULT_PITCH | FF_SWISS, // nPitchAndFamily
113.            _T("微软雅黑"));             // lpszFacename
114.        pDC->SelectObject(&font);
115.        GetWindowText(strText);
116.        pDC->DrawText(strText, rct, DT_CENTER | DT_VCENTER | DT_SING
    LELINE);
117.        pDC->RestoreDC(nSaveDC);
118.    }
119.    CRoundButton::CRoundButton(const char a)
120.    {
121.        kind = a;
122.    }
123.    CRoundButton::~CRoundButton()
124.    {
125.
126.    }
```

3.2 算法设计

这里使用 Python 实现计算器算法，需要在 C++中引用 Python.h 首先需要在解决方案资源管理器中的属性中添加以下部分

- 配置属性/VC++目录/包含目录，添加<python 安装地址>/include

- 配置属性/VC++目录/库目录，添加<python 安装地址>/libs
- 链接器/输入/附加依赖项，添加<python 安装地址>/libs/python39_d.lib

除此之外还需要在系统环境变量中添加 python39.dll 的地址。

以上完成了对 Python 的引用，在使用 python 实还需要对 python 环境初始化，由于在当前环境下使用 conda 安装了多个 python，这里还指定了被使用 python 的地址。具体代码在

CCalculatorDlg::OnInitDialog()，及对话框初始化中。如下：

```
Py_SetPythonHome((wchar_t*)L"C:\\Users\\20826\\.conda\\envs\\cpluslus");
Py_Initialize();
PyRun_SimpleString("import sys");
PyRun_SimpleString("sys.path.append(r'C:\\Users\\20826\\Documents\\GitHub\\Calculator')");
pModule = PyImport_ImportModule("src");
pFunc = PyObject_GetAttrString(pModule, "Calculate");
```

这里具体使用在，用户按下等于号时。这里将保存输入内容的字符串由 string 类型转换为 char*类型，再转化为 python 的 str 数据类型，传入前面已经初始化好的函数。得到的结果也需要经过上面两部的反向变换。之后将结果更新到文本框中。具体实现如下。

```
void CCalculatorDlg::OnBnClickedButton20()
{
    char* p = (char*)nowstr.data();
    pParams = Py_BuildValue("(s)",p);
    pResult = PyObject_CallObject(pFunc, pParams);
    char *res= NULL;
    PyArg_Parse(pResult, "s", &res);
    nowstr =res;
    CString str(nowstr.c_str());
    outEDIT.SetWindowText(str);
}
```

在 python 中，需要对输入类容进行处理，将函数名称替换为该函数要求使用的名称，同时如果出现阶乘符号则需要将阶乘符合转换为对应函数。当然如果输入不合法需要使用 catch 捕获异常并返回异常的标记。这里实现的代码如下

```
import math

def Calculate(expression):
    try:
        substring = 'sin'
        newstring = 'math.sin'
        expression = expression.replace(substring, newstring)
        substring = 'cos'
        newstring = 'math.cos'
        expression = expression.replace(substring, newstring)
        substring = 'tan'
```

```
        newstring = 'math.tan'
        expression = expression.replace(substring, newstring)
        substring = 'pi'
        newstring = 'math.pi'
        expression = expression.replace(substring, newstring)
        substring = '#2'
        newstring = '**0.5'
        expression = expression.replace(substring, newstring)
        substring = '^2'
        newstring = '**2'
        expression = expression.replace(substring, newstring)
        substring = 'ln'
        newstring = 'math.log'
        expression = expression.replace(substring, newstring)
        substring = 'e^'
        newstring = 'math.exp'
        expression = expression.replace(substring, newstring)
        expression=process_factorial(expression)
        #print(expression)
        substring = '!'
        newstring = 'math.factorial('
        expression = expression.replace(substring, newstring)

        result = eval(expression)
        return  str(result)

    except:
        return "Expression Error"

def process_factorial(expr):
    if '!' in expr:
        if len(expr) > 1 and expr[-1] == '!':
            i = len(expr) - 2
            while i >= 0 and expr[i].isdigit():
                i -= 1
            if expr[i] == '!':
                new_expr = '!' + expr[0:-1]+')'
                return new_expr
        raise ValueError("Invalid expression: " + expr)
    else:
        return expr
```

3.3 用户交互设计

这里采用一个字符串保存输入，当用户按下按键时，在字符串末尾添加对应内容，并更新。实现如下。

```
void CCalculatorDlg::OnBnClickedButton24()
{
    nowstr += "cos(";
    CString str(nowstr.c_str());
    outEDIT.SetWindowText(str);
}
```

在按下删除键时，需对最后一个字符进行判断，从而决定删除多少个字符。这里定义了一个函数用来处理当前字符，并返回处理后的字符如下。

```
std::string CCalculatorDlg::processString(const std::string& str) {

    std::string result = str;
    std::string specials[] = { "sin(", "cos(", "tan(", "pi", "#2", "^2", "ln(", "e^(","Expression Error" };
    for (const std::string& s : specials) {
        if (result.size() >= s.size() && result.substr(result.size() - s.size()) == s) {
            result.erase(result.size() - s.size(), s.size());
            return result;
        }
    }
    result.pop_back();
    return result;
}
```

3.4 统计算法实现

统计部分单独定义了一个类来实现。在这个类中，包含了私有变量一个容器。需要通过 add 和 clean 的方法来操作该容器中的内容。除此之外，还有各种计算。这些计算由 ChatGPT 辅助完成。以下不做详细分析，类中包含类容如下：

```
#ifndef DOUBLE_CONTAINER_H
#define DOUBLE_CONTAINER_H

#include <vector>

class DoubleContainer {
public:
    DoubleContainer(); // 构造函数
    void add(double num); // 添加元素
    double get_max(); // 返回最大值
    double get_min(); // 返回最小值
    double get_median(); // 返回中位数
    double get_average(); // 返回平均数
```

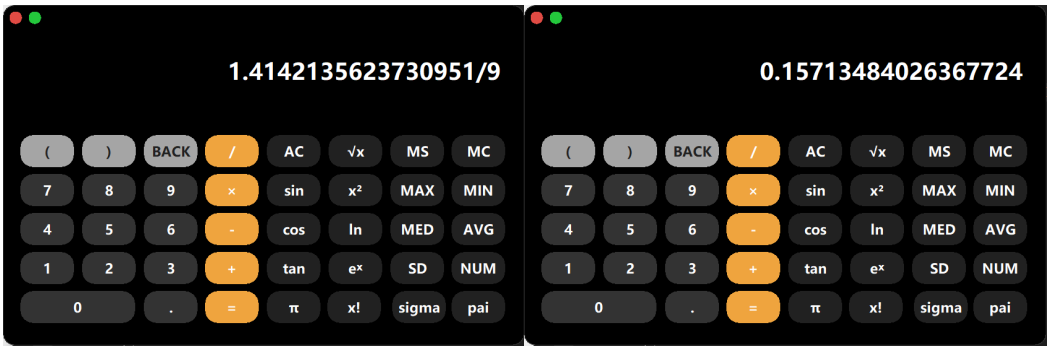
```
    double get_stddev(); // 返回标准差
    double get_sum(); // 返回累加值
    double get_product(); // 返回累乘值
    int get_size(); // 返回容器内元素个数
    void DoubleContainer::clear();
private:
    std::vector<double> nums; // 存储元素
};

#endif
```

4 实验测试结果

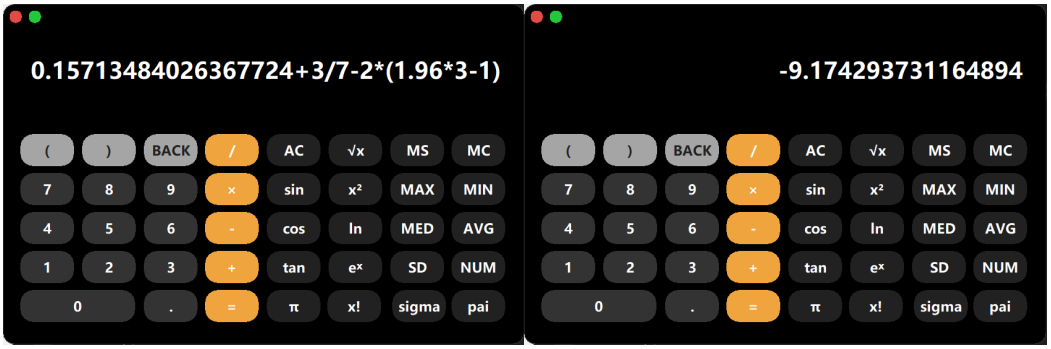
4.1 基本运算测试结果

以下左图为输入表达式右图为计算后的结果，可见计算正确，功能基本实现。



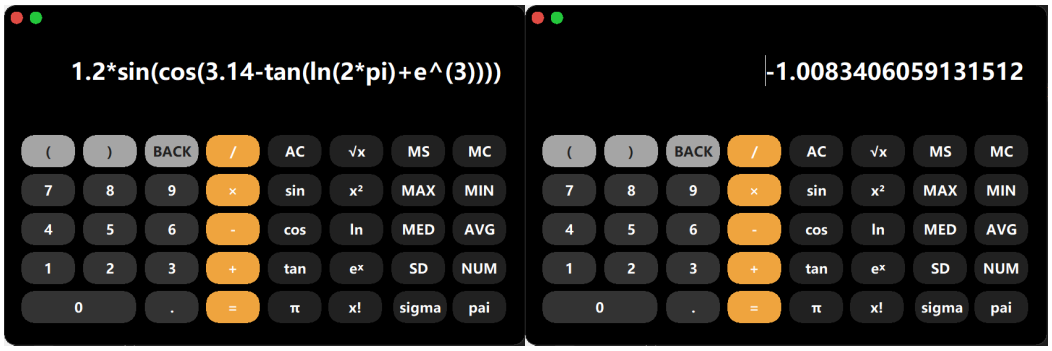
4.2 混合运算测试结果

以下左图为输入表达式右图为计算后的结果，可见计算正确，功能基本实现。



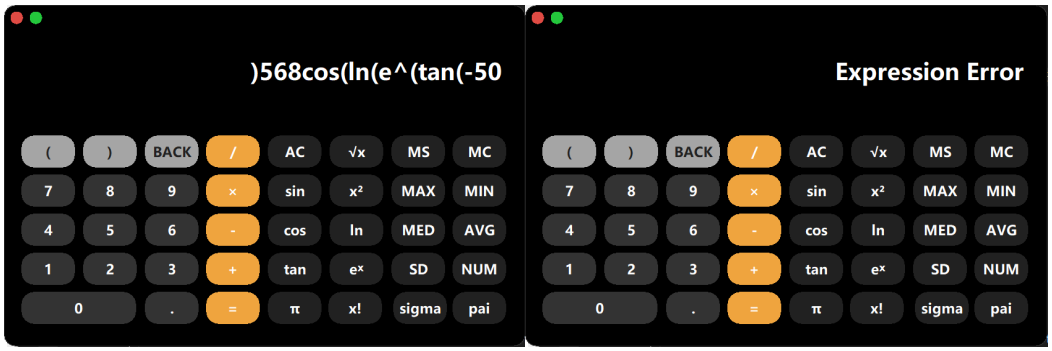
4.3 函数运算测试结果

以下左图为输入表达式右图为计算后的结果，可见计算正确，功能基本实现。



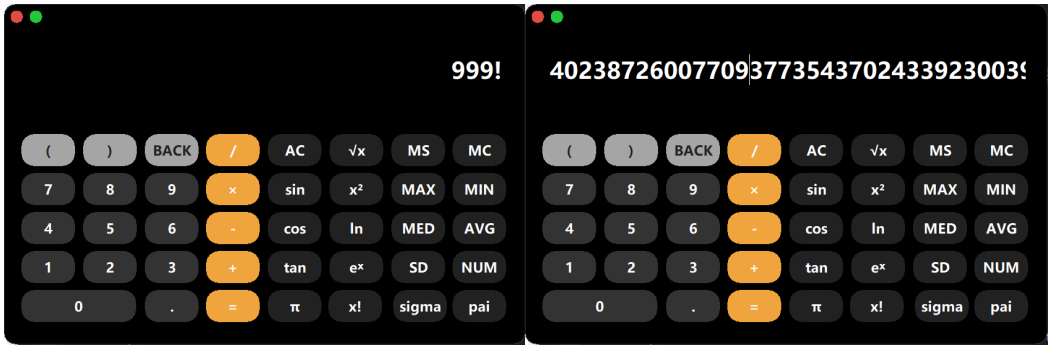
4.4 错误表达式测试运算结果

以下左图为输入表达式右图为计算后的结果，对错误表达式也能捕获而不中断程序。



4.5 阶乘运算结果

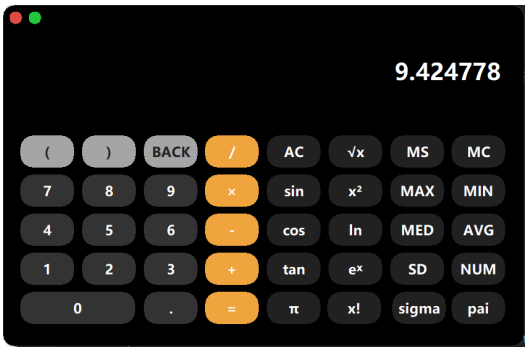
以下左图为输入表达式右图为计算后的结果，阶乘计算也没有卡顿，数据也没有溢出而出现乱码。



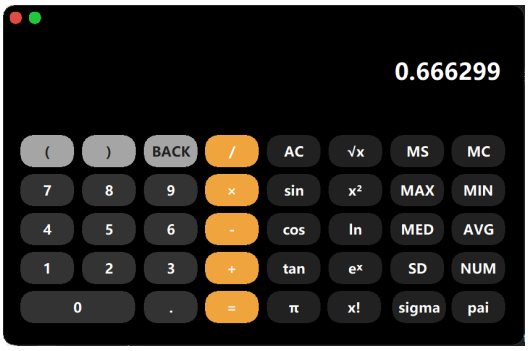
4.6 统计运算测试结果

以下为输入的四个数据在这里只允许使用数字输入，如果输入为表达式则会给出警告。

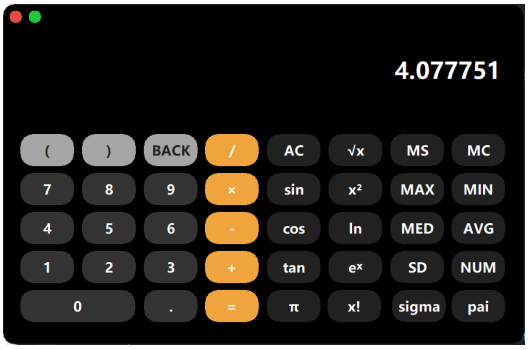
4.6.1 最大值



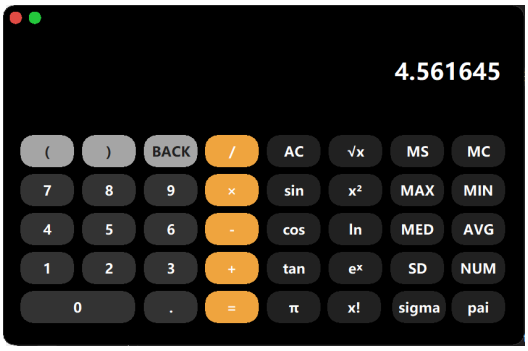
4.6.2 最小值



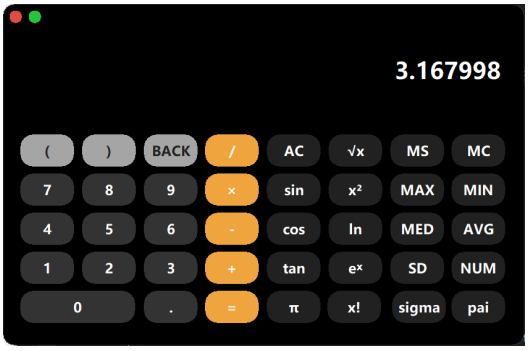
4.6.3 中位数



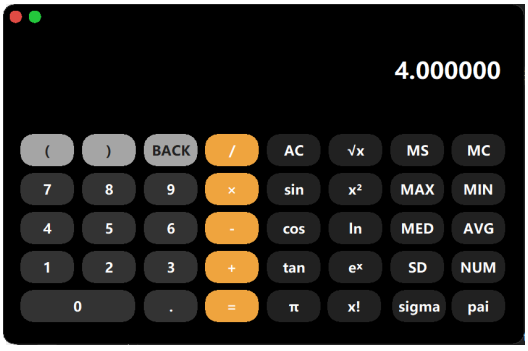
4.6.4 平均值



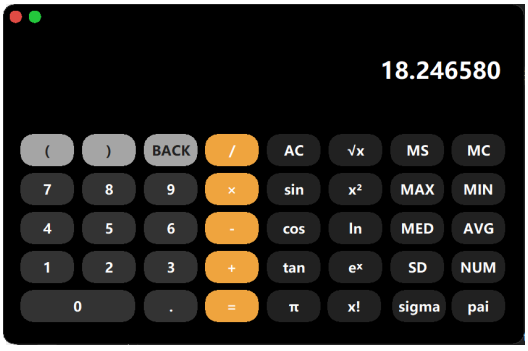
4.6.5 方差



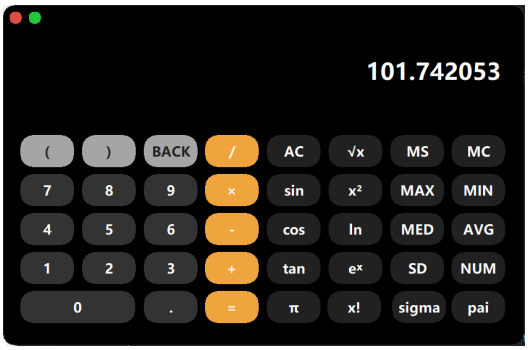
4.6.6 个数



4.6.7 求和



4.6.8 求积



5 总结

本次课程设计的主要目的是实现一个科学计算器，通过编程的方式深入了解面向对象编程思想，并在算法设计和用户交互方面得到提高。

在算法设计方面，本设计采用 Python 语言实现了高精度计算和矩阵运算，通过学习并应用高效算法，对算法设计有了更深入的理解和实践。同时，也注意到在算法实现中，时间复杂度和空间复杂度是需要平衡的，需要在实际问题中进行权衡和取舍。

在用户交互方面，通过使用 MFC 开发界面，对 MFC 的理解和使用也得到了提高。重构界面的过程中，考虑了用户的使用习惯和界面设计规范，使得界面更加美观和易用。

通过本次课程设计，我学会了如何使用面向对象编程思想设计程序，如何运用高效算法解决实际问题，以及如何使用 MFC 进行界面设计。这些都是我作为一名程序员所必须具备的技能，对我的职业发展具有重要意义。

6 附录

所有代码已经，配置规则以上传到 GitHub，详情请见 KayatoDQY/Calculator (github.com)

7 参考文献

[1] Stroustrup, B. (2014). Programming: Principles and practice using C++. Addison-Wesley Professional.

[2] Eckel, B. (2000). Thinking in C++: Introduction to standard C++, Volume 1 (2nd ed.). Prentice Hall.

[3] Meyer, B. (2008). Effective C++: 55 specific ways to improve your programs and designs (3rd ed.). Addison-Wesley Professional.

[4] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

[5] McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction (2nd ed.). Microsoft Press.

[6] Kruglinski, D. (1998). Inside Visual C++: Microsoft Foundation Class Reference (2nd ed.). Microsoft Press.

[7] Prosise, J. (1999). Programming Windows with MFC (2nd ed.). Microsoft Press.
[8] Rector, B., & Schildt, H. (1999). MFC Programming from the Ground Up. Osborne/McGraw-Hill.
[9] Tiberiu, C., & Dogaru, C. (2014). Object-Oriented Programming with C++. In Computer Science & Information Technology (pp. 259-269). Springer International Publishing.
[10] Böcker, M., & Vossen, G. (2007). Teaching C++ using a modern approach based on design patterns. In International Conference on Software Engineering (pp. 305-314). IEEE.