Lab1 向量计算器实验报告

一、功能简介

1.四则运算部分

- 实现四则表达式运算。包括加、减、乘、除、幂以及含有括号的表达式，适用范围为实数范围。
- 实现积分运算。对输入的表达式进行任意阶的积分运算，系数、指数可为任意正数或者负数。
- 实现微分计算。对输入的表达式进行任意阶的微分运算，系数、指数可为任意正数或者负数。

2.矩阵运算部分

- 实现矩阵加法
- 实现矩阵减法
- 实现矩阵乘法
- 实现矩阵转置
- 实现求行列式的值
- 实现求矩阵的特征值

3.函数运算部分

- 实现定义函数功能
- 实现运行函数功能
- 实现查看历史函数功能
- 实现修改历史函数功能

二、具体实现方式与测试样例

四则运算

1. 四则表达式
    - 实现方式
      通过字符串读入用户输入数据，然后在手写的**vector_2**的基础上进行继承，加入push、pop、top操作，构建手写**stack_2**。创建两个栈，一个为**操作数栈**，一个为**运算符栈**。然后遍历string，对其中的数字与符号进行读取并压栈操作。其中读取数字调用**Get_num**函数的接口，可实现对int与double数据类型的读取操作。然后遇到运算符的时候，与栈顶的运算符进行优先级比较。如果优先级大于栈顶操作符，直接入栈。如果优先级小于或者等于栈顶运算符，则取出栈顶运算符与两个操作数，进行运算后结果压入操作数栈。重复以上操作直到读取的运算符优先级大于栈顶操作符。读取到左括号直接压栈，读取到右括号开始操作运算符栈直到遇到左括号。最后进行顺序操作清空运算符栈与操作数栈并输出结果。
    - 测试样例一
        - 输入：(-2)*8+(((-5)*2^3)*123)-25+56/8
        - 输出：-4954
    - 测试样例二
        - 输入：2*8^2-(-9)*2+((((-4+16)/2)*3+5)+(6-92))-56/(2*4)+6
        - 输出：156

2. 积分运算
   - 实现方式
     首先建立一个结构体**Func**，包括系数，自变量，指数与答案。然后对用户输入的字符串进行分段处理，切割出单个的单项式，然后继续切割为系数、自变量与指数。在这里还是调用Get_num对系数与指数进行提取。随后进行积分操作，根据积分的运算法则处理系数、自变量与指数。最后将系数与指数调用**sprintf_s**函数将浮点数转化为字符串形式，随后将系数、自变量与指数依次放入答案中，将各个单项式的答案依次加起来汇入最终答案，输出即可。

   > 对于任意阶的积分运算，可将上次运算完的结果作为参数再次调用该函数即可，因为在函数之间传递的数据类型为**string**，故十分便捷。

   - 测试样例一
     - 输入：3*x*^*2*+*6*x^3-(-5)*x^2+(-10)*x^(-3)
       1
     - 输出：1.00*x*^*3.00*+*1.50*x^4.00-(-1.67)*x*^*3.00*+*5.00*x^(-2.00)
   - 测试样例二：
     - 输入：200*x^2
       5
     - 输出：0.08*x^7.00

---

3. 微分运算
   - 实现方式
     对用户输入的字符串处理相似于积分运算，分割后进行微分运算，根据微分运算法则分别处理各个单项式，特别注意对**常数**的处理，最后将单项式结果汇总输出即可。

   > 任意阶微分处理方法同积分

   - 测试样例一
     - 输入：(-5)*x*^*(-4)*+*9*x^2-x+3
       1
     - 输出：20.00*x*^*(-5.00)*+*18.00*x^1.00-1.00*x+0
   - 测试样例二
     - 输入：2*x^5
       5
     - 输出：240
   - 测试样例三
     - 输入：2*x^5
       5
     - 输出：0

矩阵运算

1. 矩阵加法
   - 实现方法
     首先用户输入第一个矩阵的行数与列数，再输入具体矩阵。接着输入第二个矩阵的行数与列数，再输入具体矩阵。然后判断是否符合**矩阵加减的规范**。如果符合，对矩阵每个相应项进行加法操作，最后输出；如果不符合，提示格式错误并返回。
   - 测试样例一

- 输入：3 3
  2 2 2
  2 2 2
  2 2 2
  3 3
  4 4 4
  4 4 4
  4 4 4
- 输出：6 6 6
  6 6 6
  6 6 6

- 测试样例二
  - 输入：3 3
    2 2 2
    2 2 2
    2 2 2
    2 2
    4 4
    4 4
  - 输出：警告：检测到格式错误，执行返回操作

---

2. 矩阵减法
   - 实现方法
     基本与加法一致，对矩阵相应项进行减法操作，然后输出
   - 测试样例一
     - 输入：3 3
       9 9 9
       9 9 9
       9 9 9
       3 3
       4 4 4
       4 4 4
       4 4 4
     - 输出：5 5 5
       5 5 5
       5 5 5
   - 测试样例二
     - 输入：3 3
       2 2 2
       2 2 2
       2 2 2
       2 2
       4 4
       4 4
     - 输出：警告：检测到格式错误，执行返回操作

---

3. 矩阵乘法

- 实现方法
  在用户输入两个矩阵之后，会进行规范判断，即第一个矩阵的列数是否等于第二个矩阵的行数。如果相等，则会按照矩阵乘法的规则，计算出结果矩阵的每一项并输出。否则提示**检测到非法输入**，执行返回操作。

```
for(int i=1;i<=hang_1;i++)
  for(int j=1;j<=lie_2;j++)
  {
    for(int k=0;k<lie_2;k++)
      temp_mat_3[i][j]+=temp_mat_1[i][k]*temp_mat_2[k][j];
  }
```

- 测试样例一：
  - 输入：3 2
    2 2 2
    2 2 2
    2 3
    4 4
    4 4
    4 4
  - 输出：16 16 16
    16 16 16
    16 16 16
- 测试样例二：
  - 输入：3 3
    2 2 2
    2 2 2
    2 2 2
    2 2
    4 4
    4 4
  - 输出：警告：检测到格式错误，执行返回操作

---

4. 矩阵转置
   - 实现方法
     在用户输入了一个矩阵之后，会进行将**行列倒置**后存入新的矩阵并输出

```
for(int i=1;i<=hang_1;i++)
 for(int j=1;j<=lie_1;j++)
  {
    cin>>temp_mat_1[i][j];
    temp_mat_2[j][i]=temp_mat_1[i][j];
  }
```

   - 测试样例

- 输入：3 2
  1 2 3
  4 5 6
- 输出：1 3 5
  2 4 6

---

5. 求矩阵行列式的值
   - 实现方法
     调用**Get_deter**函数对第一行进行代数余子式展开，转化成求余子式的值，接着调用**Str_min**函数构造余子式，然后利用递归继续调用**Get_deter**，直到到达**递归基**，即一阶行列式，此时该行列式的值即为本身。然后**回溯**，便可以得到行列式的值。如果该矩阵不符合**行列式规范**，则提示格式错误并执行返回操作。

> 递归调用部分如下

```
double Get_deter(double *before,int hang)
{
  for(int i=0;i<hang;i++)
   {
     Str_min(before,after,hang,i);
     temp_answer+=prefix*before[i]*Get_deter(after,hang-1);
     prefix*=-1;
   }
}
```

   - 测试样例一
     - 输入：3 3
       9 8 7
       4 5 6
       1 3 2
     - 输出：-39
   - 测试样例二
     - 输入：3 2
       2 2 2
       2 2 2
     - 输出：警告:检测到格式错误,执行返回操作

函数运算

1. 定义函数
   - 实现方法
     建立结构体SP_Func,用来存函数的各项信息。包括**函数主体**,如:3*x+1,**函数名称**,如f(x),**变量名称**,如:x,**函数头**,如:f。在用户输入函数字符串后，进行分割操作，将函数的各个部分对应的放入结构体当中，即完成了对一个函数的储存。
   - 测试样例
     - 输入：f(x)=3*x^2+1

- 输出：提示：您已成功定义函数！

---

2. 运行函数
   - 实现方法
     利用string库中的**find**函数，实现**Replace_all**函数，即可将目标字符串中的所有对应字符串更改为所需字符串。在用户输入运行的函数中后，利用**Get_num**获取操作数，并在历史函数中找到对应函数，将函数主体中的变量更换为操作数，调用**四则运算求值**的接口，即可输出答案。

> Replace_all函数的实现

```
void Replace_all(string& str,const string& strfir,const string& strend)
{
  string::size_type pos(0);
  string::size_type symbol(0);
  while(1)
  {
    if((pos=str.find(strfir))==0)
    {
      str.replace(pos,strfir.length(),strend);
    }
    else if((pos=str.find(strfir))!=string::npos&&pos>symbol)
    {
      str.replace(pos,strfir.length(),strend);
      symbol=pos;
    }
    else break;
  }
}
```

   - 测试样例一
     - 输入:f(-2)
     - 输出：13
   - 测试样例二
     - 输入：f(0.5)
     - 输出：1.75

---

3. 查看历史函数
   - 实现方法
     实现了两种查找方式，利用**函数名称查找**与**按照编号查找**。得益于结构体的实现，一个函数的各个信息都被储存了起来。利用名称查找，复杂度为**O(n)**,遍历已储存的结构体，直到找到对应函数。按编号查找复杂度为**O(1)**,相当于循秩查找，直接按下角标提取函数即可。如果无用户输入的函数名或编号超出实际范围，**提示查找失败**，执行重置操作。
   - 测试样例一
     - 输入：f(x)
     - 输出：查找成功！您所查找的函数为:f(x)=3*x^2+1
   - 测试样例二
     - 输入：1
     - 输出：查找成功！您所查找的函数为:f(x)=3*x^2+1

- 测试样例三
  - 输入：g(x)
  - 输出：警告:未找到您的函数!执行重置操作

---

4. 修改历史函数
   - 实现方法
     同查找方式，提供名称查找与编号查找，在找到函数之后，提供**更改函数名称**，**更改变量名称**，**更改函数解析式**,三个选项。三者都是直接对结构体中储存的函数信息进行更改，借助**Repalce_all**函数，可以高效实现对string类型的函数信息的更改。如果更改后的函数解析式与原变量不符，会提示操作不合法，执行返回操作。
   - 测试样例一
     - 输入：g(x)
     - 输出：您已成功修改！修改后的函数为g(x)=3*x^2+1
   - 测试样例二
     - 输入：y
     - 输出：您已成功修改！修改后的函数为f(y)=3*y^2+1
   - 测试样例三
     - 输入：(-5)*x*^*3*+9x^2+6
     - 输出：您已成功修改！修改后的函数为f(x)=(-5)*x*^*3*+9x^2+6
   - 测试样例四
     - 输入：(-5)*y*^*3*+9y^2+6
     - 输出：检测到格式错误,执行返回操作

三、优势与不足

优势

- 最大的特色在于**对用户友好**，具有命令行构建的简洁整齐的画面，还具有详细的操作指导与按键提示，部分操作还配备有输入示例，能极大的便利规范操作的用户，使之获得良好的使用感。
- 在文件中使用了较多的**接口通用，性能优秀的基本函数**，如**Get_num**(获取操作数)，**Replace_all**(批量替换字符串),**Make_qua_ope**(对字符串进行四则运算)等，这些函数前期构建的时候略微有些艰难，但在后期处理各种高阶操作时提供了极大的支持与便利。
- 整体项目文件整洁有序，分为三大部分，函数名称与变量名字以功能为主，便于**理解与移植**。

不足

- 未能实现函数的**嵌套与互调**，日后有精力会继续完善。
- 面对未知输入，无完整错误提示且可能导致**程序崩溃**。
- 存在可能的**bug**

四、总结

1. 收获
   本次Lab耗时2星期14天，又一次磨砺了自己的**基本功能代码与多函数多区域整合能力**，并且对Vector与Stack有了更加深刻的理解，在更高难度的代码撰写上进行了尝试，提高自己的能力。
2. 参考资料

- C++ std::string::find()函数（在字符串中查找内容）
- 四则运算表达式（c++实现）
- 【OpenCV4】计算对称矩阵特征值和特征向量 cv::eigen() 用法详解和代码示例（c++）

3. 项目地址
   本次Lab所有文件已上传至https://github.com/rucerchui/Sophomore/tree/main/Lab1

欢迎您的使用!
