山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

学号：202200130048	姓名： 陈静雯	班级： 6
实验题目：栈		
实验学时：2	实验日期： 11.2	

实验目的：
掌握栈结构的定义与实现；
掌握栈结构的使用。

软件开发工具：
Vscode

1. 实验内容
创建栈类，采用数组描述；计算数学表达式的值。 输入数学表达式，输出表达式的计算结果。数学表达式由单个数字和运算符“+”、“-”、“”、“/”、“(”、“）”构成，例如 2+3(4+5)－6/4。

2. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）
定义两个栈，一个保存符号，一个保存数字，从第一位开始扫描表达式，数字的存入数字栈，符号的在存入符号栈之前，与栈顶比较，如果新的运算符优先级高，压入栈，如果低，就把栈顶的运算符弹出，数字栈中弹出两个数字，进行运算，直到比栈顶优先级高为止。如果运算符是 ），就弹出栈的运算符，进行计算，直到把（ 弹出为止。

3. 测试结果（测试输入，测试输出）

```
=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
3
1+6/1*7+2*1*4+9/1+2*0*9+9+7/(9*5)-1*6-0*8-7-9*2+6-(0-5-2*8-7-9*5*(6-5*5*2*6-2-7-5+6*7+6*9-1*0*0+3*0+2/1-6/6+5))
-9197.84
0-4-1/6*(1-(6/7)-4+6+2+6*1)-1*7+2-8*2+0-(4+6-6*1+(3-8*6/4-6-5)*6/4/8+7-1*4/9*5)-0/6+1-0-2+7-2+6*4-3*6+2/8+6+1*6*2
-3.47
5-3*9+5/1*5-9+1*8-6-8-4*1+5-2+9/3*2-2/5/(2-6)*2/7-9*0-2+4/6*6*7*8-8-8*6+8*9*(3+0*1/5/2*7*8+0-8*8-5+8/5*2-0)
-4362.57
```

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）
代码没有考虑负数和多位数的情况，负数的话要判断前一位是符号还是数字，如果是符号，则该“-”为负号，否则为减号。如果是多位数，就判断前一位是不是数字，如果是，就要前一位数字*10+本位数字，直到扫描到符号位为止。

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
#include <iostream>
#include <iomanip>
using namespace std;

template<class T>
class mystack{   //数组描述
public:
```

```
    mystack(int n=10);
    bool empty();
    int size();
    T top();
    void pop();
    void push(T& thelement);
    ~mystack();
private:
    T* element;
    int stacktop;
    int stacksize;
};

template<class T>
mystack<T>::mystack(int n){
    stacksize=n;
    element=new T [n];
    stacktop=-1;
}

template<class T>
bool mystack<T>::empty(){
    return stacktop==-1;
}

template <class T>
int mystack<T>::size(){
    return stacktop+1;
}

template <class T>
T mystack<T>::top(){
    if(stacktop!=-1){
        return element[stacktop];
    }
    else{
        return 0.0;
    }
}

template <class T>
void mystack<T>::pop(){
    if(stacktop!=-1){
        element[stacktop--]=0;
    }
}
```

```
template <class T>
void mystack<T>::push(T& thelement){
    if(stacktop+1==stacksize){                    //若空间不够,重新进行动态分配
        stacksize*=2;
        T* temp = new T [stacksize];
        for(int i=0;i<stacksize;i++){
            temp[i]=element[i];
        }
        element=temp;
    }
    element[++stacktop]=thelement;
}

template<class T>
mystack<T>::~mystack(){
    stacksize=0;
    stacktop=-1;
    T* p=element;
    delete [] p;
    element=NULL;
}

bool check(char top,char y){     //判断是否压栈
    if(y=='('||top=='(') return true;
    if(y==')') return false;
    if((top=='+' || top=='-') && (y=='*' || y=='/')){
        return true;
    }
    return false;
}

template<class T, class M>
void tosum(mystack<T>&a, mystack<M>& b){     //进行二元运算
    double x = a.top();
    a.pop();
    double y = a.top();
    a.pop();
    char sign = b.top();
    b.pop();
    double sum=0;
    if(sign=='+') sum=x+y;
    if(sign=='-') sum=y-x;
    if(sign=='*') sum=x*y;
    if(sign=='/') sum=y/x;
    a.push(sum);
```

```
}

void calculate(){
    mystack<double> a;  //数字
    mystack<char> b;  //运算符
    char p;
    getchar();
    while(cin.peek()!='\n'){
        cin>>p;
        if(p<='9' && p>='0'){
            double num = p-'0';
            a.push(num);
        }
        else{
            if(b.empty() || check(b.top(),p)){
//如果运算符栈空或者新的运算符等级高，就压栈
                b.push(p);
            }
            else{
                if(p==')'){
                    while(b.top()!='('){    //计算到栈顶是(，即括号运算结束
                        tosum(a,b);
                    }
                    b.pop();
                }
                else{
                    while(!b.empty() && !check(b.top(),p)){
 //一直把比新运算符等级高或同级的运算都算完或者栈空才结束运算
                        tosum(a,b);
                    }
                    b.push(p);
                }
            }
        }
    }
    while(!b.empty()){
        tosum(a,b);
    }
    double sum = a.top();
    cout<<fixed<<setprecision(2)<<sum<<endl;
}

int main(){
    int n;
    cin>>n;
    for(int i=0;i<n;i++){
```

```
        calculate();
    }
}
```