САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Лабораторная работа № 4: Динамические структуры данных

Студентк гр. 3343 _____ Отмахов Д. В.

Преподаватель _____ Государкин Я. С.

Санкт-Петербург

2024

Цель работы

Изучить принципы работы классов на языке C++, реализовать динамическую структуру стек на базе массива при помощи класса.

Задание

Вариант 1

1) Реализовать класс CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных int.

Объявление класса стека:

```
class CustomStack {
        public:
        // методы push, pop, size, empty, top + конструкторы, деструктор
        private:
        // поля класса, к которым не должно быть доступа извне
        protected:  // в этом блоке должен быть указатель на массив данных
                int* mData;
};
```

Перечень методов класса стека, которые должны быть реализованы:

- void push(int val) - добавляет новый элемент в стек
- void pop() - удаляет из стека последний элемент
- int top() - доступ к верхнему элементу
- size_t size() - возвращает количество элементов в стеке
- bool empty() - проверяет отсутствие элементов в стеке
- extend(int n) - расширяет исходный массив на n ячеек

2) Обеспечить в программе считывание из потока stdin последовательности (не более 100 элементов) из чисел и арифметических операций (+, -, *, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

- Если очередной элемент входной последовательности - число, то положить его в стек,
- Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже),
- Если входная последовательность закончилась, то вывести результат (число в стеке).

Если в процессе вычисления возникает ошибка:

- например вызов метода pop или top при пустом стеке (для операции в стеке не хватает аргументов),
- по завершении работы программы в стеке более одного элемента,

программа должна вывести "error" и завершиться.

Выполнение работы

Описание функций:

- int main(): главная функция программы, возвращает 0 при успешном завершении. Отвечает за ввод данных и вывод полученного значения;
- void push(int val): добавляет новый элемент в стек, если места в стеке недостаточно, изменяет его размер при помощи функции resize();
- void pop(): удаляет из стека последний элемент, при пустом стеке выдает ошибку;
- int top(): доступ к верхнему элементу, при пустом стеке выдает ошибку;
- size_t size(): возвращает количество элементов в стеке;
- bool empty(): проверяет отсутствие элементов в стеке;
- extend(int n): расширяет исходный стек на n ячеек, при отрицательном n, выдает ошибку.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2 + 3 4 - 5 * +	-2	Выходные данные соответствуют ожиданиям.
2.	1 -10 - 2 *	22	Выходные данные соответствуют ожиданиям.
3.	1 7 + 3 X - 5 * +	error	Выходные данные соответствуют ожиданиям.

Выводы

В ходе выполнения работы были изучены принципы работы классов в языке C++, написана программа реализующая стек на базе массива.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.c

```
class CustomStack{
    public:
        CustomStack(){
            mIndex = -1;
            mCapacity = 0;
            mData = new int[mCapacity];
        }

        ~CustomStack(){
            delete[] mData;
        }

        void push(int val){
            if (mIndex + 1 == mCapacity){
                resize(mCapacity + 1);
            }
            mIndex++;
            mData[mIndex] = val;
        }

        void pop(){
            if (empty()){
                cout << "error";
                exit(0);
            }
            mIndex--;
        }

        size_t size(){
            return mIndex + 1;
        }

        bool empty(){
            return mIndex == -1;
        }
```

```
        int top(){
            if (empty()){
                cout << "error";
                exit(0);
            }
            return mData[mIndex];
        }

        void extend(int n){
            if (n <= 0){
                cout << "error";
                exit(0);
            }
            resize(mCapacity + n);
        }
    protected:
        int* mData;
        size_t mIndex;
        size_t mCapacity;

        void resize(size_t newCapacity){
            if (newCapacity == mCapacity){
                return;
            }

            if (newCapacity < mCapacity){
                cout << "error";
                exit(0);
            }

            int* newData = new int[newCapacity];
            copy(mData, mData + mCapacity, newData);
            delete[] mData;
            mCapacity = newCapacity;
            mData = newData;
        }
};
```

```
int main(){
    CustomStack stack;
    string line("");
    string element;
    getline(cin, line);
    stringstream ss(line);
    while (ss >> element)
    {
        if (element == "-" || element == "+" || element == "*" ||
element == "/")
        {
            int b = stack.top();
            stack.pop();
            int a = stack.top();
            stack.pop();
            if (element == "-"){
                stack.push(a - b);
            }
            else if (element == "+"){
                stack.push(a + b);
            }
            else if (element == "*"){
                stack.push(a * b);
            }
            else if (element == "/"){
                stack.push(a / b);
            }
        }
        else{
            int i = stoi(element);
            stack.push(i);
        }
    }

    if (stack.size() == 1)
        cout << stack.top() << endl;
    else
        cout << "error";
```

```
    return 0;
}
```