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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Динамические структуры данных

Студент гр. 3341 _____ Мокров И.О.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучить концепцию динамических структур данных в языке C++, понять принципы их реализации, применение и управление памятью. Освоить методы работы с основными динамическими структурами, такими как списки, стеки, очереди и деревья, и научиться применять их для решения различных задач. Развить навыки написания, отладки и тестирования кода, используя динамические структуры данных.

Задание

Стековая машина.

Требуется написать программу, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе массива.

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.

Примечания:

1. Указатель на массив должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.
3. Предполагается, что пространство имен std уже доступно.
4. Использование ключевого слова using также не требуется.

Пример:

Исходная последовательность: 1 -10 - 2 *

Результат: 22

Выполнение работы

Описание класса CustomStack:

- Класс CustomStack реализует базовые операции стека: push, pop, top, size, empty, и extend.
- Стек основан на массиве, который динамически расширяется при необходимости.
- Метод push добавляет элемент в стек. Если стек достиг максимальной емкости, он автоматически расширяется с использованием метода extend.
- Метод pop удаляет верхний элемент стека. Если стек пуст, метод вызывает исключение std::runtime_error.
- Метод top возвращает верхний элемент стека. Если стек пуст, вызывает исключение std::runtime_error.
- Метод size возвращает количество элементов в стеке.
- Метод empty проверяет, пуст ли стек.
- Метод extend увеличивает емкость массива, добавляя к нему новые ячейки.

Описание основной программы:

- Программа считывает строку входных данных из стандартного ввода, представляющую последовательность чисел и арифметических операций, разделенных пробелами.
- Входная последовательность интерпретируется, и если элемент — число, оно добавляется в стек с использованием метода push.
- Если элемент — арифметический оператор, программа извлекает два верхних элемента из стека, применяет операцию и возвращает результат в стек.
- Если возникает ошибка (например, операция pop на пустом стеке), программа выводит "error".
- После обработки всей последовательности программа проверяет, что в стеке остался

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	OK
2.	3 4 + 2 * 7 -	7	OK

Выводы

В процессе выполнения данной лабораторной работы по изучению динамических структур данных на языке C++, была освоена и закреплена теория работы со стеком и практические навыки его реализации.

Приложение А

Исходный код программы

Название файла: main.c

```
#include <iostream>
#include <cstdlib>
#include <sstream>

class CustomStack {
public:
     CustomStack(size_t capacity = 10) : mCapacity(capacity), mSize(0),
mData(new int[capacity]) {}
    ~CustomStack() { delete[] mData; }

    void push(int val) {
        if (mSize >= mCapacity) {
            extend(10);
        }
        mData[mSize++] = val;
    }

    void pop() {
        if (mSize > 0) {
            --mSize;
        } else {
            throw std::runtime_error("Stack underflow");
        }
    }

    int top() const {
        if (mSize > 0) {
            return mData[mSize - 1];
        } else {
            throw std::runtime_error("Stack is empty");
        }
    }

    size_t size() const {
        return mSize;
    }

    bool empty() const {
        return mSize == 0;
    }

    void extend(size_t n) {
        size_t newCapacity = mCapacity + n;
        int* newData = new int[newCapacity];
        std::copy(mData, mData + mSize, newData);
        delete[] mData;
        mData = newData;
        mCapacity = newCapacity;
    }

private:
    size_t mCapacity;
```

```
    size_t mSize;
protected:
    int* mData;
};

int main() {
    CustomStack stack;

    std::string input;
    std::getline(std::cin, input);

    std::istringstream iss(input);
    std::string token;
    bool error = false;

    while (iss >> token) {
        if (std::isdigit(token[0]) || (token[0] == '-' && token.size() >
1)) {
            stack.push(std::atoi(token.c_str()));
        } else {
            if (stack.size() < 2) {
                error = true;
                break;
            }

            int b = stack.top();
            stack.pop();
            int a = stack.top();
            stack.pop();

            if (token == "+") {
                stack.push(a + b);
            } else if (token == "-") {
                stack.push(a - b);
            } else if (token == "*") {
                stack.push(a * b);
            } else if (token == "/") {
                if (b == 0) {
                    error = true;
                    break;
                }
                stack.push(a / b);
            } else {
                error = true;
                break;
            }
        }
    }

    if (error || stack.size() != 1) {
        std::cout << "error" << std::endl;
    } else {
        std::cout << stack.top() << std::endl;
    }

    return 0;
}
```