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 _____ Ягудин Д.Р.

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

Описание основной программы:

- Программа считывает строку входных данных из стандартного ввода.
- Входная последовательность разбивается на подстроки, и если элемент — число, оно добавляется в стек с использованием метода push, иначе применяет оператор к двум последним элементам в стеке.
- Если возникает ошибка, программа выводит "error".
- После обработки всей последовательности программа проверяет, что в стеке осталось последнее значение и выводит его.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	123000 50 48 12 34 + - * /	1230	OK
2.	12 10 - 25 *	50	OK

Выводы

В ходе выполнения данного задания по изучению динамических структур данных на языке C++, была закреплена теория работы со стеком и практические навыки его реализации.

Приложение А

Исходный код программы

Название файла: main.c++

```
class CustomStack {

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

    void push(int item) {
        if (mSize >= mCapacity) {
            extend(10);
        }

        mData[mSize++] = item;
    }

    void pop() {
        if (mSize > 0) {
            --mSize;
        }else{
            throw runtime_error("not enough stack size\n");
        }
    }

    int top() const {
        if (mSize > 0) {
            return mData[mSize - 1];
        }else{
            throw runtime_error("empty stack\n");
        }
    }

    int size() const {
        return mSize;
    }

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

    void extend(int n) {
        int newCapacity = mCapacity + n;

        int* newData = new int[newCapacity];

        copy(mData, mData + mSize, newData);
        delete[] mData;

        mData = newData;
        mCapacity = newCapacity;
    }

private:
```

```
    int mCapacity;
    int mSize;

protected:
    int* mData;

};

int main() {
     string str;
     getline(cin, str);

     istringstream command(str);
     string tok;

     CustomStack stack;
     int com = 0;
     int flag = 0;

     while (command >> tok){
          if ((tok[0] == '-' && tok.size() > 1) || isdigit(tok[0])){
               stack.push(atoi(tok.c_str()));
          }else{
               if (tok == "+") com = 1;
               if (tok == "-") com = 2;
               if (tok == "*") com = 3;
               if (tok == "/") com = 4;

               if (stack.size() < 2){
                    cout << "error\n";
                    return 0;
               }

               int s_2 = stack.top();
               stack.pop();
               int s_1 = stack.top();
               stack.pop();

               switch (com){
                    case 1:{
                         stack.push(s_1 + s_2);
                         break;
                    }

                    case 2:{
                         stack.push(s_1 - s_2);
                         break;
                    }

                    case 3:{
                         stack.push(s_1 * s_2);
                         break;
                    }

                    case 4:{
                         if (s_2 == 0){
                              cout << "error\n";
                              return 0;
```

```
                        }
                        stack.push(s_1 / s_2);
                        break;
                    }
                }
            }
        }

        if (stack.size() != 1){
            cout << "error\n";
            return 0;
        }

        cout << stack.top() << '\n';

    return 0;
}
```