**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. 3342 |  | Лучкин М.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью данной лабораторной работы является изучение работы с динамическими структурами данных и их создание. Также одна из целей – изучение основ работы с языком C++. Требуется написать программу, моделирующую работу стека на базе **массива.**

## Задание

Вариант 1.

**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.

## Выполнение работы

Реализован класс CustomStack, который имеет следующие методы: push, pop, size, empty, top, extend. Он имеет приватное поле, содержащее его размер. В защищенном поле mData находятся данные стека. Реализован main() в котором считываются и выполняются пользовательские команды. Есть проверка на пустоту массива, при вызове pop и top.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 1 2 + 3 4 - 5 \* + | -2 |

## Выводы

Была разработана программа на языке C++, которая создаёт динамическую структуру данных – стек на базе массива. Реализованы методы для работы с созданной структурой и считывание пользовательских команд и их выполнение.

# Приложение А Исходный код программы

Название файла: main.cpp

void printError(){

cout << "error";

exit(0);

}

class CustomStack {

public:

CustomStack(){

mData = new int[100];

mSize = 0;

}

void push(int val){

mData[mSize++] = val;

}

void pop(){

if (mSize == 0){

printError();

}

mSize--;

}

int top(){

if (mSize == 0){

printError();

}

return mData[mSize-1];

}

int get\_elem(){

int elem = top();

pop();

return elem;

}

size\_t size(){

return mSize;

}

bool empty(){

return mSize == 0;

}

void extend(int n){

mSize += n;

int\* newData = new int [mSize];

for (size\_t i = 0; i < mSize; ++i)

newData[i] = mData[i];

delete [] mData;

mData = newData;

}

~CustomStack(){

delete [] mData;

mSize = 0;

}

protected:

int\* mData;

private:

size\_t mSize;

};

bool isNumber(string& str) {

if (str.empty()) {

return false;

}

size\_t i = 0;

if (str[i] == '-') {

++i;

}

bool hasDigit = false;

for (; i < str.size(); ++i) {

if (isdigit(str[i])) {

hasDigit = true;

} else {

return false;

}

}

return hasDigit;

}

int main() {

CustomStack stack;

string s;

getline(cin, s);

stringstream ss(s);

string text;

int x, y;

while (getline(ss, text, ' ')) {

if (isNumber(text)) {

stack.push(stoi(text));

continue;

}

y = stack.get\_elem();

x = stack.get\_elem();

char token = \*text.data();

if (token == '+') {

stack.push(x + y);

} else if (token == '-') {

stack.push(x - y);

} else if (token == '\*') {

stack.push(x \* y);

} else if (token == '/') {

stack.push(x / y);

}

}

if (stack.size() != 1)

printError();

cout << stack.top() << endl;

return 0;

}