**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**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3344 |  | Коняева М.В. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы изучить и использовать базовые механизмы языка С++, необходимые для реализации стека и очереди.

## 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*.

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

Был реализован класс *CustomStack*. Весь функционал данного класса (поля, методы) был описан в задании и реализован в соответствии с ним.

В функции *main()*вызывается конструктор по умолчанию. Далее идет считывание и работа со стеком в цикле *while()*, который работает, пока на вход поддаются данные. Если считанные данные – это число, то переписываем их в переменную типа *int* и добавляем в стек. Если считанные данные – это арифметическая операция, то идет проверка размера стека: если в стеке меньше 2 чисел, значит программа не сможет произвести операцию и, выведя ошибку, завершит работу.

Если стек имеет корректный размер (больше 2 чисел), то записываем в переменные *num2* и *num1* верхние числа стека с помощью метода t*op()* и удаляем их с помощью метода *pop()*. Производим арифметическую операцию над числами, и записываем результат в стек с помощью метода *push()*.

После завершения обработки входных данных проверяем, что в стеке осталось одно число. Если данное условие не выполняется, то программа выводит ошибку и завершает работу. Выводим результат (число в стеке).

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 1 2 + 3 4 - 5 \* + | -2 | Данные обработаны корректно. |
|  | 2 3 4 + + + | error | Данные обработаны корректно. |
|  | 3 4 5 + + | 12 | Данные обработаны корректно. |

## Выводы

Были изучена работа с базовыми механизмами языка C++. Была создана программа, в которой реализован стек на основе динамического массива.

# Приложение А Исходный код программы

Название файла: lb4.c

class CustomStack {

public:

CustomStack(){

mData = new int[10];

len = 0;

memory = 10;

}

~CustomStack() {

delete[] mData;

}

void push(int val){

if(len == memory){

extend(1);

}

mData[len++] = val;

}

void pop(){

len--;

}

int top(){

return mData[len-1];

}

size\_t size(){

return len;

}

bool empty(){

return len == 0;

}

void extend(int n){

memory += n;

int \*tmp\_arr = new int[memory];

for (int i = 0; i < len; ++i) {

tmp\_arr[i] = mData[i];

}

delete[] mData;

mData = tmp\_arr;

}

protected:

int\* mData;

private:

size\_t len;

size\_t memory;

};

int main() {

CustomStack stack;

char input[10];

int num1, num2;

while (cin >> input) {

if (isdigit(input[0]) || isdigit(input[1])) {

int num = strtol(input, NULL, 10);

stack.push(num);

} else {

if (stack.size() < 2) {

cout << "error" << endl;

return 1;

}

num2 = stack.top();

stack.pop();

num1 = stack.top();

stack.pop();

switch (input[0]) {

case '+':

stack.push(num1 + num2);

break;

case '-':

stack.push(num1 - num2);

break;

case '\*':

stack.push(num1 \* num2);

break;

case '/':

if (num2 != 0) {

stack.push(num1 / num2);

} else {

cout << "error" << endl;

return 1;

}

break;

default:

cout << "error" << endl;

return 1;

}

}

}

if (stack.size() != 1) {

cout << "error" << endl;

return 1;

}

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

return 0;

}