# 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. 9303 |  | Жорже М.А. |
| Преподаватель |  | Чайка К. В. |

Санкт-Петербург 2020

**Цель работы.**

Изучать принципы работы структуры данных стек, реализовать его при помощи инструментария языка С++.

**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

1. 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.

**ПРИЛОЖЕНИЕ**

**А ИСХОДНЫЙ КОД ПРОГРАММЫ**

#include <iostream>

using namespace std;

class CustomStack

{

private:

int sizes;

int sizeStack;

protected:

int \*mData;

public:

CustomStack()

{

sizes = 0;

sizeStack = 100;

mData = new int[sizeStack];

}

~CustomStack()

{

delete[] mData;

}

bool empty()

{

if (sizes == 0)

{

return 1;

}

return 0;

}

size\_t size()

{

return sizes;

}

void pop()

{

if (this->empty())

{

this->~CustomStack();

cout << "error" << endl;

exit(0);

}

sizes--;

}

void push(int val)

{

mData[sizes] = val;

sizes++;

}

int top()

{

if (this->empty())

{

this->~CustomStack();

cout << "error" << endl;

exit(0);

}

return mData[sizes - 1];

}

void out()

{

if (this->size() != 1)

{

this->~CustomStack();

cout << "error" << endl;

exit(0);

}

cout << mData[0];

}

int Operator(char st)

{

int b = this->top();

this->pop();

int a = this->top();

this->pop();

switch (st)

{

case '\*':

return (a \* b);

break;

case '/':

return (a / b);

break;

case '+':

return (a + b);

break;

case '-':

return (a - b);

break;

}

}

};

int check(char sym)

{

if ((sym == '+') || (sym == '-') || (sym == '/') || (sym == '\*'))

return 1;

return -1;

}

char \*Initial()

{

char \*str = new char[10];

char st;

int i = 0;

while (((st = getchar()) != EOF) && (st != ' ') && (st != '\n'))

{

str[i] = st;

i++;

}

if (i == 0)

{

return NULL;

}

str[i] = '\0';

return str;

}

int main()

{

CustomStack Stack;

char \*str;

while ((str = Initial()) != NULL)

{

if ((check(\*str) > 0 && (str[1] == '\0')))

{

Stack.push(Stack.Operator(\*str));

}

else

{

Stack.push(atoi(str));

}

delete[] str;

}

Stack.out();

return 0;

}