

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. 1304

Андреев В.В.

Преподаватель

Чайка К. В.

Санкт-Петербург

2022

Цель работы.

Научиться работать с динамическими структурами в контексте ооп и ознакомиться с основами языка C++.

Задание.

Вариант 1.

*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.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 -10 - 2 *	22	Ответ верный.
2.	1 2 + 3 4 - 5 * +	-2	Ответ верный.

Выводы.

Были изучены принципы создания динамических структур и освоены основы работы с ними на языке C++.

По итогу написана программа, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе массива.

В процессе работы над программой были использованы классы, методы и поля классов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <cstring>
#include <cstdlib>

using namespace std;

class CustomStack
{
public:
    CustomStack()
    {
        extend(10);
    }

    ~CustomStack()
    {
        free(mData);
        mData = nullptr;
    }

public:
    void push(int val)
    {
        if( Size + 1 >= Capacity )
        {
            Capacity *= 2;

            extend(Capacity);
        }

        mData[Size] = val;
        ++Size;
    }

    void pop()
    {
        if( Size > 0 )
        {
            --Size;
        }
        else
        {
            cout << "error";
            exit(0);
        }
    }
}
```

```

int top()
{
    if( Size > 0 )
    {
        return mData[Size - 1];
    }
    else
    {
        cout << "error";
        exit(0);
    }
}

size_t size()
{
    return Size;
}

bool empty()
{
    return Size == 0;
}

void extend(int n)
{
    Capasity += n;

    int* NewData = (int*)malloc(sizeof(int) * Capasity);

    if( mData != nullptr )
    {
        memcpy(NewData, mData, Size * sizeof(int));
        free(mData);
    }

    mData = NewData;
}

private:

    size_t Size = 0;
    size_t Capasity = 0;

protected:

    int* mData = nullptr;

};

```

```

int main()
{
    CustomStack LStack;

    string Str;
    getline(cin, Str, '\n');

    int LNumber = 0;
    int LNumberSign = 1;
    bool IsNewNumber = false;

    for( int i = 0; i < Str.length(); ++i )
    {
        if( Str[i] == '*' )
        {
            int RValue = LStack.top();
            LStack.pop();
            int LValue = LStack.top();
            LStack.pop();

            LStack.push(LValue * RValue);
            continue;
        }
        if( Str[i] == '/' )
        {
            int RValue = LStack.top();
            LStack.pop();
            int LValue = LStack.top();
            LStack.pop();

            LStack.push(LValue / RValue);
            continue;
        }
        if( Str[i] == '+' )
        {
            int RValue = LStack.top();
            LStack.pop();
            int LValue = LStack.top();
            LStack.pop();

            LStack.push(LValue + RValue);
            continue;
        }
        if( Str[i] == '-' )
        {
            if( i < Str.length() - 1 && isdigit(Str[i + 1]) )
            {
                LNumberSign = -1;
            }
            else
            {
                int RValue = LStack.top();
                LStack.pop();
                int LValue = LStack.top();
                LStack.pop();
            }
        }
    }
}

```

```

        LStack.push(LValue - RValue);
    }

    continue;
}

if( Str[i] == ' ' )
{
    if( IsNewNumber )
    {
        LStack.push(LNumber * LNumberSign);
        LNumber = 0;
        LNumberSign = 1;
        IsNewNumber = false;
    }

    continue;
}

IsNewNumber = true;
LNumber = LNumber * 10 + (int)(Str[i] - '0');
}
if( IsNewNumber )
{
    LStack.push(LNumber * LNumberSign);
}

if( LStack.size() != 1 )
{
    cout<<"error";
    exit(0);
}
cout << LStack.top();

return 0;
}

```