Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №19**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Основные алгоритмы работы со стеками

Выполнил:

Студент группы РИС-20-1б

Ремянников Александр Владимирович

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Получить опыт создания основных алгоритмов стека.

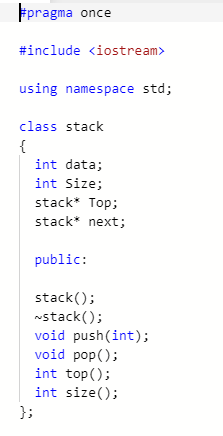
**Постановка задачи**

Реализовать пользовательский класс stack и методы для работы с ним.

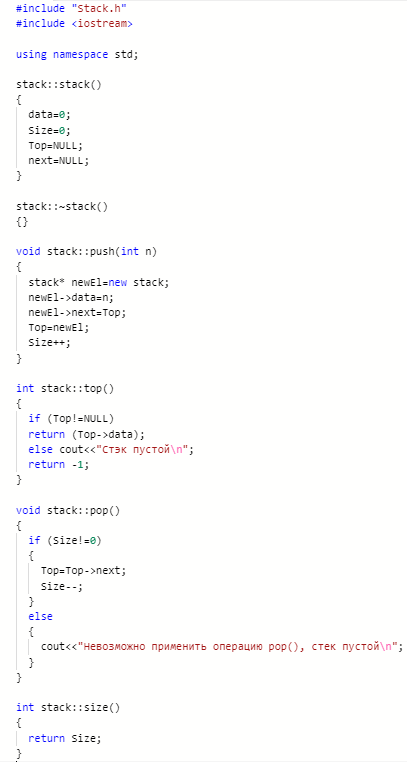
**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать пользовательский класс Stack.



1.2. Описать методы для работы с классом Stack.



**2.** В ходе работы были использованы следующие типы данных:

2.1. Для хранения элементов стека и размера стека использовался тип int.



**3.** Для решения задачи данные были представлены в следующем виде:

3.1. Стек был реализован с использованием пользовательского класса, одним из атрибутов которого является указать на элемент класса stack, что позволяет последовательно перемещаться по стеку.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции вывода использовался оператор cout.



**5.** Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код для демонстрации работы методов класса Stack.

****

**Решение**

**Main.cpp**

#include <iostream>

#include "Stack.h"

using namespace std;

int main()

{

stack st;

cout<<st.top()<<" size="<<st.size()<<'\n';

st.push (12);

cout<<st.top()<<" size="<<st.size()<<'\n';

st.push (32);

cout<<st.top()<<" size="<<st.size()<<'\n';

st.push (41);

cout<<st.top()<<" size="<<st.size()<<'\n';

st.push (56);

cout<<st.top()<<" size="<<st.size()<<'\n';

st.pop();

cout<<st.top()<<" size="<<st.size()<<'\n';

st.pop();

cout<<st.top()<<" size="<<st.size()<<'\n';

st.pop();

cout<<st.top()<<" size="<<st.size()<<'\n';

st.pop();

cout<<st.top()<<" size="<<st.size()<<'\n';

st.pop();

}

**Stack.h**

#pragma once

#include <iostream>

using namespace std;

class stack

{

int data;

int Size;

stack\* Top;

stack\* next;

public:

stack();

~stack();

void push(int);

void pop();

int top();

int size();

};

**Stack.cpp**

#include "Stack.h"

#include <iostream>

using namespace std;

stack::stack()

{

data=0;

Size=0;

Top=NULL;

next=NULL;

}

stack::~stack()

{}

void stack::push(int n)

{

stack\* newEl=new stack;

newEl->data=n;

newEl->next=Top;

Top=newEl;

Size++;

}

int stack::top()

{

if (Top!=NULL)

return (Top->data);

else cout<<"Стэк пустой\n";

return -1;

}

void stack::pop()

{

if (Size!=0)

{

Top=Top->next;

Size--;

}

else

{

cout<<"Невозможно применить операцию pop(), стек пустой\n";

}

}

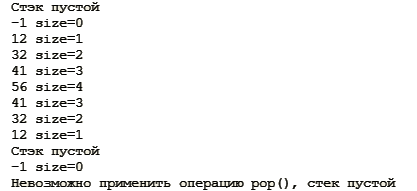
int stack::size()

{

return Size;

}

**Скриншоты результатов работы программы**

****

**GitHub**

<https://github.com/sugarrrfqs/Lab19/tree/main>