МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

**ОТЧЁТ**

**«ЛАБОРАТОРНАЯ №11.2: СТЕК»**

Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-2б

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

[1. Постановка задачи 3](#_Toc98242087)

[2. Анализ задачи 4](#_Toc98242088)

[3. Исходный код 5](#_Toc98242089)

[4. Анализ результатов 8](#_Toc98242090)

# Постановка задачи

Продемонстрировать работу стека как структуры данных.

# 

# Анализ задачи

Стек можно реализовать на основе односвязного списка с ограниченным функционалом. Необходимо реализовать методы добавления узла в начало стека и удаления узла из начала стека для организации принципа LIFO (last in, first out).

# 

# Исходный код

#include <iostream>

template <typename T>

class Stack {

public:

Stack();

~Stack();

T top() const;

size\_t size() const;

void push(const T& data);

void pop();

void clear();

private:

struct Node {

Node(T data = T(), Node\* pointer\_to\_next\_node = nullptr)

: data(data), pointer\_to\_next\_node(pointer\_to\_next\_node) {}

Node(const Node& copy)

: data(copy.data), pointer\_to\_next\_node(copy.pointer\_to\_next\_node) {}

Node& operator=(const Node& right) {

if (this != &right) {

data = right.data;

pointer\_to\_next\_node = right.pointer\_to\_next\_node;

}

return \*this;

}

T data;

Node\* pointer\_to\_next\_node;

};

void pushFirstNode(Node\* node);

void pushFrontNode(Node\* node);

void popFirstNode();

void popFrontNode();

size\_t stack\_size;

Node\* head\_node;

};

template<typename T>

Stack<T>::Stack()

: stack\_size(size\_t{0}), head\_node(nullptr) {}

template<typename T>

Stack<T>::~Stack()

{

clear();

}

template<typename T>

T Stack<T>::top() const

{

return head\_node->data;

}

template<typename T>

size\_t Stack<T>::size() const

{

return stack\_size;

}

template<typename T>

void Stack<T>::push(const T& data)

{

Node\* new\_node = new Node(data);

stack\_size ? pushFrontNode(new\_node) : pushFirstNode(new\_node);

++stack\_size;

}

template<typename T>

void Stack<T>::pop()

{

if (stack\_size == size\_t{0}) return;

Node\* remove\_node = head\_node;

(stack\_size == size\_t{1}) ? popFirstNode() : popFrontNode();

delete remove\_node;

--stack\_size;

}

template<typename T>

void Stack<T>::clear()

{

if (stack\_size == size\_t{0}) return;

Node\* remove;

Node\* next\_node = head\_node;

while (stack\_size) {

remove = next\_node;

next\_node = next\_node->pointer\_to\_next\_node;

delete remove;

--stack\_size;

}

head\_node = nullptr;

}

template<typename T>

void Stack<T>::pushFirstNode(Node\* node)

{

head\_node = node;

}

template<typename T>

void Stack<T>::pushFrontNode(Node\* node)

{

node->pointer\_to\_next\_node = head\_node;

head\_node = node;

}

template<typename T>

void Stack<T>::popFirstNode()

{

head\_node = nullptr;

}

template<typename T>

void Stack<T>::popFrontNode()

{

head\_node = head\_node->pointer\_to\_next\_node;

}

int main()

{

std::setlocale(LC\_ALL, "Russian");

Stack<int> stack;

for (int i = 1; i != 6; ++i) {

stack.push(i); std::cout << stack.top() << ' ';

} std::cout << '\n';

while (stack.size()) {

if (stack.size()) std::cout << stack.top() << ' ';

stack.pop();

}

}

# Анализ результатов

Результаты работы программы (рис. 1).

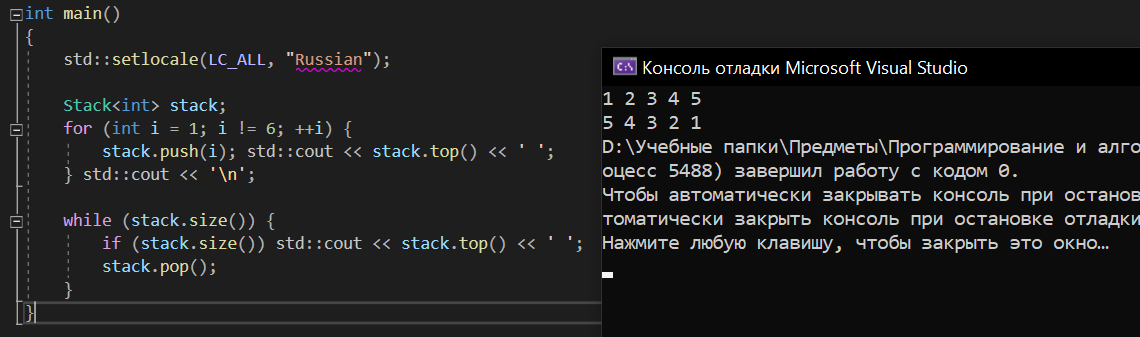


Рисунок 1 — Результаты