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

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

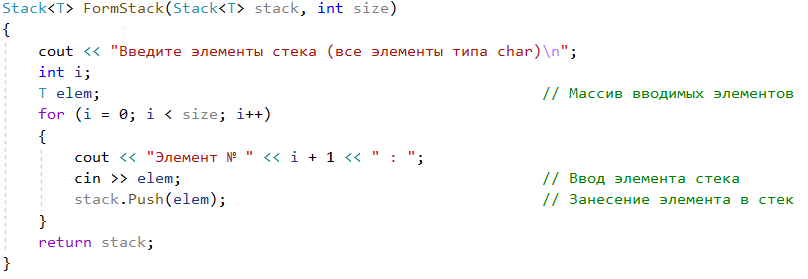
Получить практические навыки работы со стеками.

**Постановка задачи**

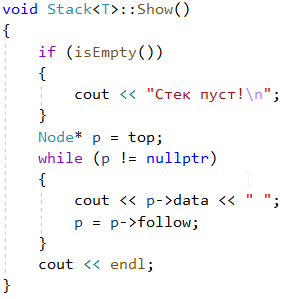
Тип информационного поля char\*. Добавить в стек элемент после элемента с заданным информационным полем. Реализация стека через класс.

**Анализ задачи**

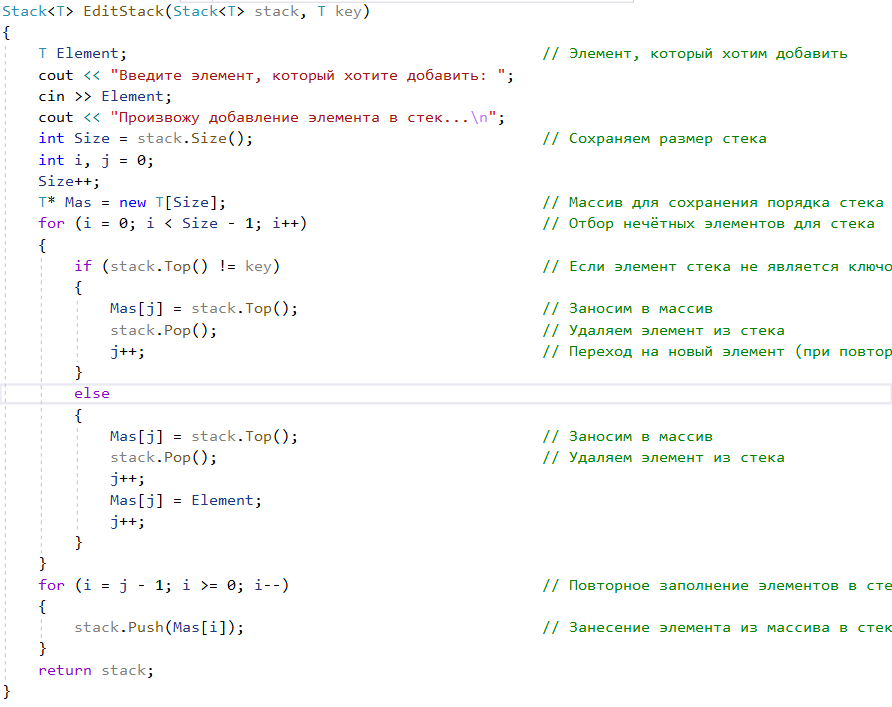
1. Для решения задачи необходимо:
   1. Разработать функцию для формирования стека через класс.



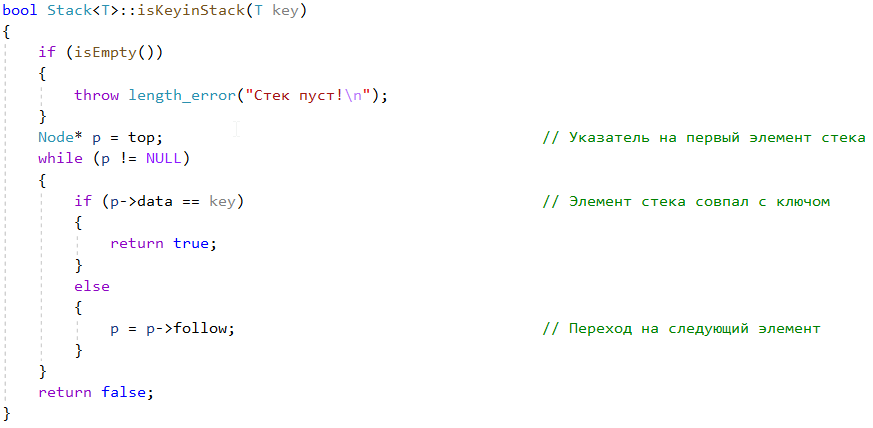
* 1. Разработать функцию для вывода информационного поля стека на экран через консоль.



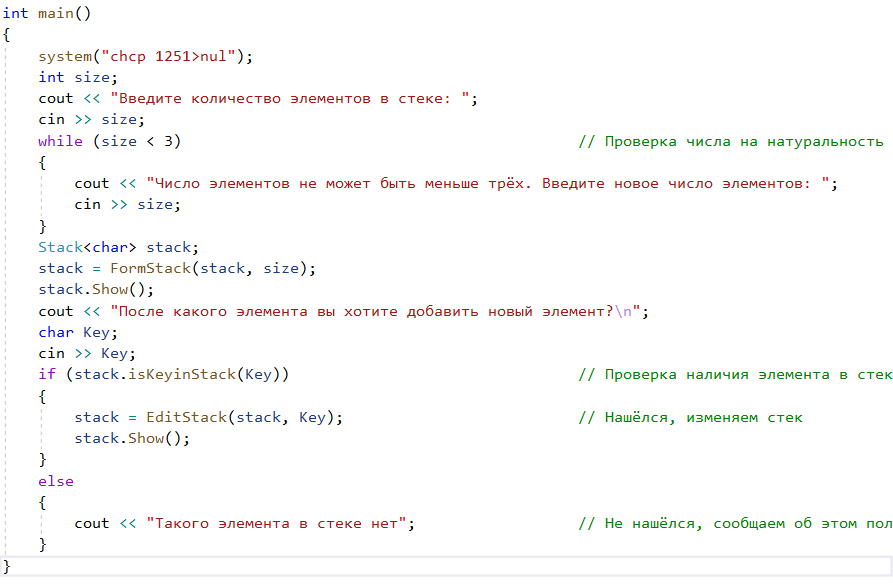
* 1. Разработать функцию для добавления нового элемента после определённого в стек.



* 1. Разработать функцию для нахождения ключа в стеке.



* 1. Реализовать применение этих функций в программе.



1. В ходе работы были использованы типы данных:
   1. Для функции FormStack использовались следующие аргументы:
   2. Тип Stack<T>: стек.
   3. Тип int: размер стека.

Сама функция имеет тип Stack<T> и возвращает стек.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_sJcPioHLnO.png

* 1. Для конструктора без параметров класса Stack не используются аргументы.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_XyCdq26hr6.png

* 1. Для метода Push используются следующие аргументы:

Сам метод имеет тип void, поскольку при работе метода ничего не нужно возвращать.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_VghrU5Xeg1.png

* 1. Для метода isEmpty не используются аргументы.

Сам метод имеет тип bool и возвращает истину, если стек пуст.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_40laCC37ey.png

* 1. Для метода Top не используются аргументы.

Сам метод имеет тип const T& возвращает вершину стека.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_nsM86WPT5q.png

* 1. Для метода Pop не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода ничего не нужно возвращать.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_y46ZUUJU7f.png

* 1. Для метода Size не используются аргументы.

Сам метод имеет тип int и возвращает размер стека.

* 1. Для метода Show не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода ничего не нужно возвращать.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_1ehifMzVe4.png

* 1. Для метода EditStack использовались следующие аргументы:
  2. Тип Stack<T>: стек.
  3. Тип T: ключ поиска.

Сам метод имеет тип Stack<T> и возвращает стек.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_4h6oQRrtTh.png

* 1. Для метода isKeyinStack использовались следующие аргументы:
  2. Тип T: ключ поиска.

Сам метод имеет тип bool и возвращает значение истина, если ключ нашелся в списке, или ложь, если ключа в списке нет.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_z5SGrilpf5.png

1. Для операций ввода и вывода данных использовались следующие операторы и функции:
   1. Для вывода информации на консоль использовался оператор cout.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_Fpw1ef0xpt.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_JMK5EqYzIp.png

* 1. Для ввода данных через консоль использовался оператор cin.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_MaXQm3TaYp.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_d5CIhDC2W5.png

1. Поставленные задачи решены следующими действиями:
   1. Все действия со стеком реализованы через написанные вручную дружественные функции и методы класса.

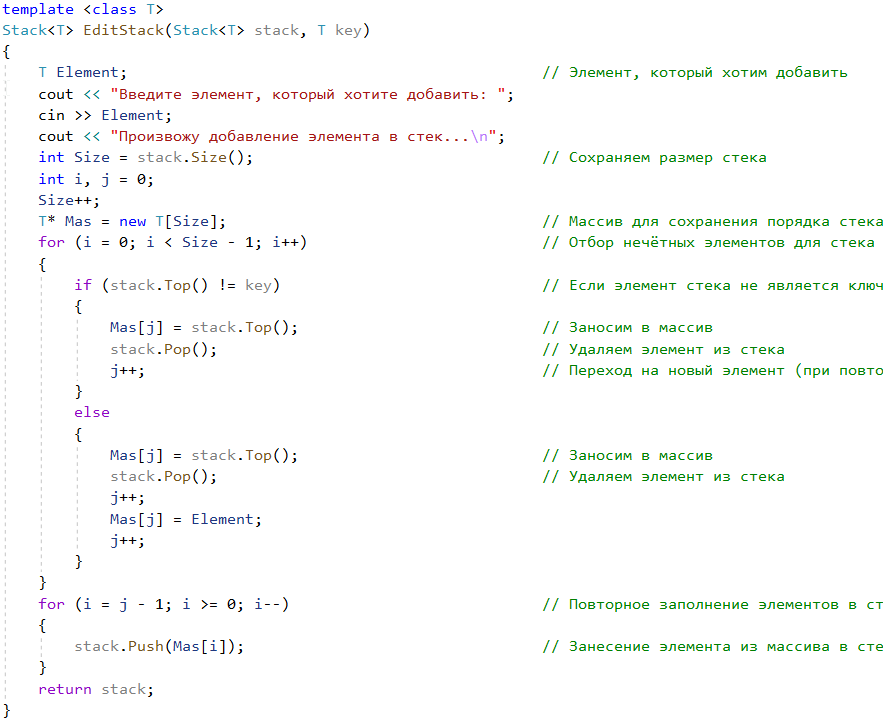
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_sJcPioHLnO.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_4h6oQRrtTh.png

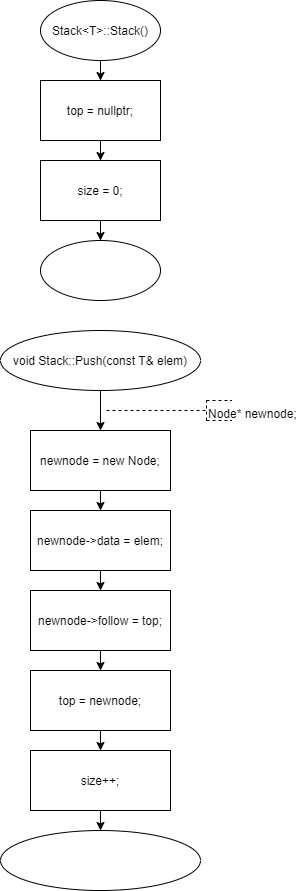
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_z5SGrilpf5.png

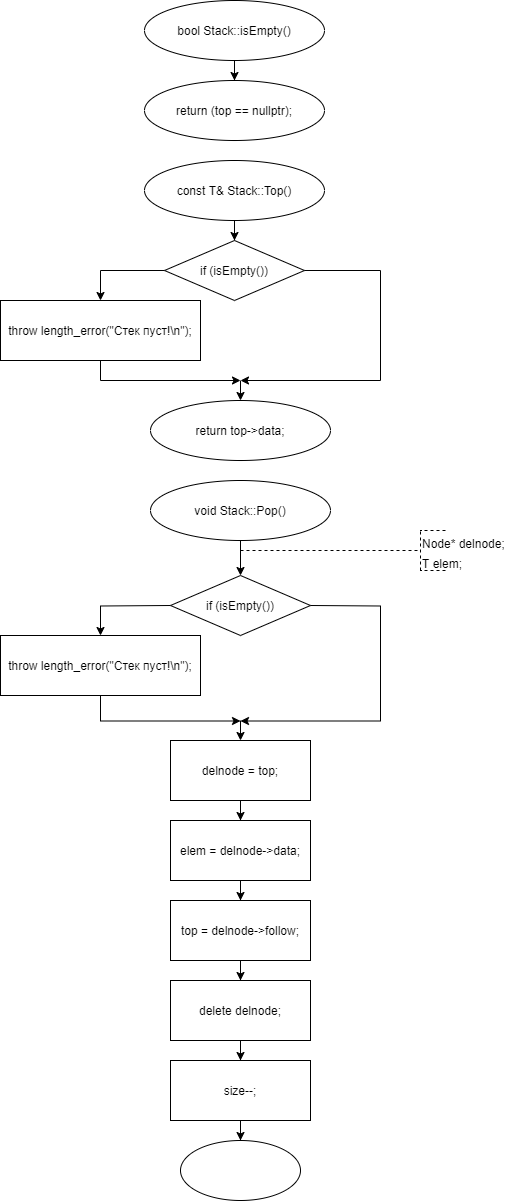
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_1ehifMzVe4.png

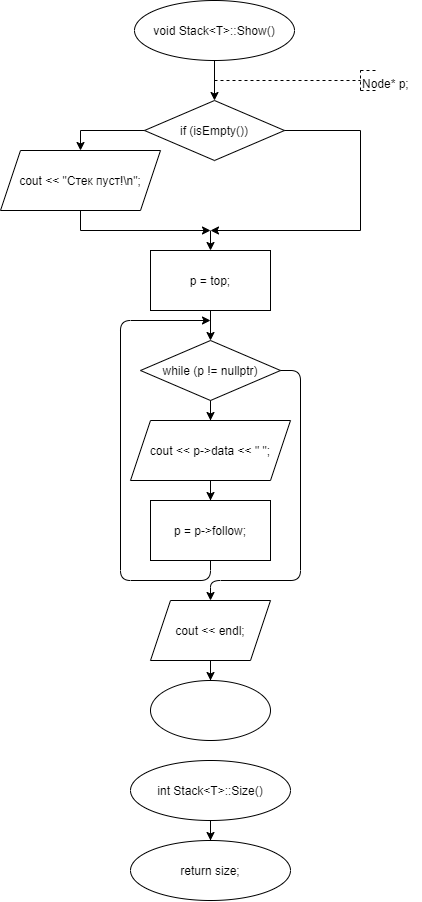
* 1. Добавление элемента после определённого элемента в стеке реализовано с помощью цикла for и вспомогательных функций поиска ключа во множестве.

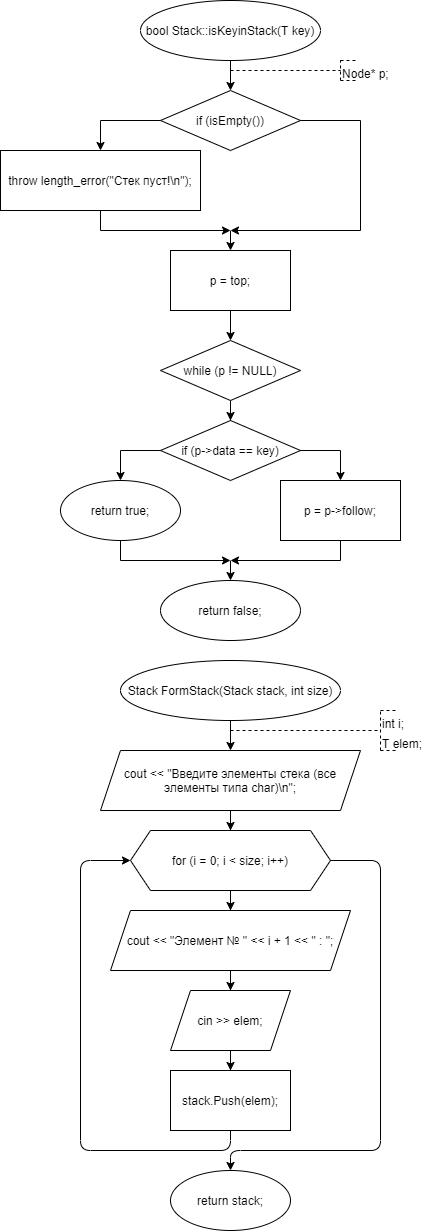


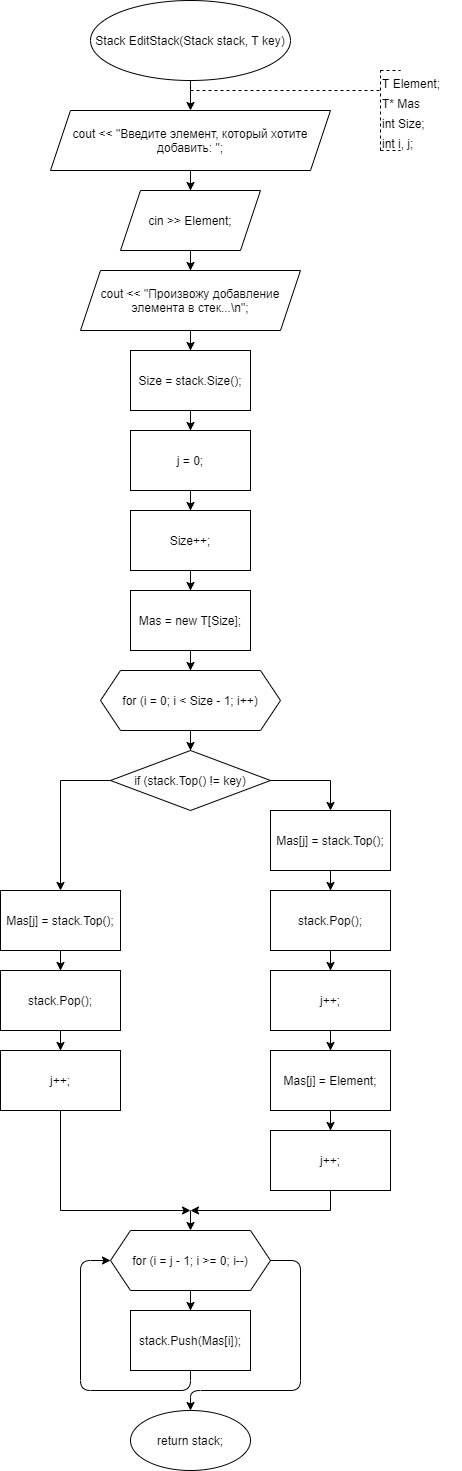
**Блок-схема программы**

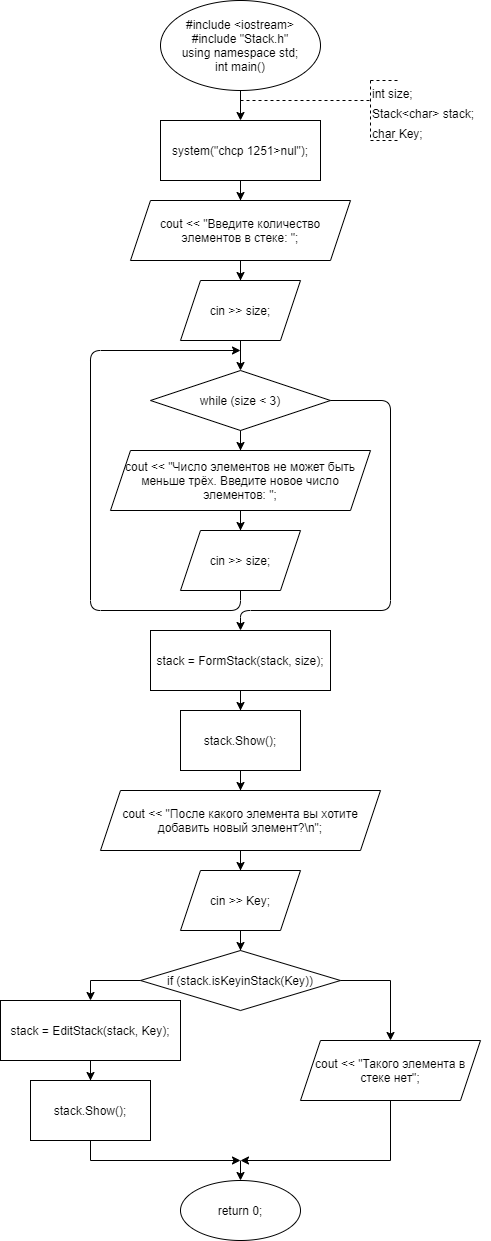


****

****

****

****

****

**Реализация задачи на языке С++**

**Заголовочный файл Stack.h**

#pragma once

#include <iostream>

#include <stdexcept>

using namespace std;

template <class T>

class Stack

{

struct Node

{

T data;

Node\* follow;

};

Node\* top;

int size;

public:

Stack();

void Push(const T& elem);

bool isEmpty();

const T& Top();

void Pop();

void Show();

int Size();

bool isKeyinStack(T key);

template <class T> friend Stack<T> FormStack(Stack<T> stack, int size);

template <class T> friend Stack<T> EditStack(Stack<T> stack, T key);

};

template <class T>

Stack<T>::Stack()

{

top = nullptr;

size = 0;

}

template <class T>

void Stack<T>::Push(const T& elem)

{

Node\* newnode = new Node;

newnode->data = elem;

newnode->follow = top;

top = newnode;

size++;

}

template <class T>

bool Stack<T>::isEmpty()

{

return (top == nullptr);

}

template <class T>

const T& Stack<T>::Top()

{

if (isEmpty())

{

throw length\_error("Стек пуст!\n");

}

return top->data;

}

template <class T>

void Stack<T>::Pop()

{

if (isEmpty())

{

throw length\_error("Стек пуст!\n");

}

Node\* delnode = top;

T elem = delnode->data;

top = delnode->follow;

delete delnode;

size--;

}

template <class T>

void Stack<T>::Show()

{

if (isEmpty())

{

cout << "Стек пуст!\n";

}

Node\* p = top;

while (p != nullptr)

{

cout << p->data << " ";

p = p->follow;

}

cout << endl;

}

template <class T>

int Stack<T>::Size()

{

return size;

}

template <class T>

bool Stack<T>::isKeyinStack(T key)

{

if (isEmpty())

{

throw length\_error("Стек пуст!\n");

}

Node\* p = top;

while (p != NULL)

{

if (p->data == key)

{

return true;

}

else

{

p = p->follow;

}

}

return false;

}

template <class T>

Stack<T> FormStack(Stack<T> stack, int size)

{

cout << "Введите элементы стека (все элементы типа char)\n";

int i;

T elem;

for (i = 0; i < size; i++)

{

cout << "Элемент № " << i + 1 << " : ";

cin >> elem;

stack.Push(elem);

}

return stack;

}

template <class T>

Stack<T> EditStack(Stack<T> stack, T key)

{

T Element;

cout << "Введите элемент, который хотите добавить: ";

cin >> Element;

cout << "Произвожу добавление элемента в стек...\n";

int Size = stack.Size();

int i, j = 0;

Size++;

T\* Mas = new T[Size];

for (i = 0; i < Size - 1; i++)

{

if (stack.Top() != key)

{

Mas[j] = stack.Top();

stack.Pop();

j++;

}

else

{

Mas[j] = stack.Top();

stack.Pop();

j++;

Mas[j] = Element;

j++;

}

}

for (i = j - 1; i >= 0; i--)

{

stack.Push(Mas[i]);

}

return stack;

}

**Файл Лабораторная работа №19.cpp**

#include <iostream>

#include "Stack.h"

using namespace std;

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в стеке: ";

cin >> size;

while (size < 3)

{

cout << "Число элементов не может быть меньше трёх. Введите новое число элементов: ";

cin >> size;

}

Stack<char> stack;

stack = FormStack(stack, size);

stack.Show();

cout << "После какого элемента вы хотите добавить новый элемент?\n";

char Key;

cin >> Key;

if (stack.isKeyinStack(Key))

{

stack = EditStack(stack, Key);

stack.Show();

}

else

{

cout << "Такого элемента в стеке нет";

}

}

**Скриншоты результатов работы программы**

