Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка программно-информационных систем (РИС)

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Информационные динамические структуры»

Вариант 18

Выполнил

Студент группы РИС-22-1б

Кирпичников И. И.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

**Введение**

**Цель работы**: Знакомство с динамическими информационными структурами на примере стеков и очередей.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. провести анализ предметной области

2. реализовать разработку программы

3. провести тестирование программы

**Анализ предметной области**

**Постановка задачи**

Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать стек/очередь. Удалить элемент с заданным ключом. Добавить К элементов в начало списка.

Разработать следующие функции:

1.Создание стека/очереди.

2.Добавление элемента в стек/очередь (в соответствии со своим вариантом).

3.Удаление элемента из стека/очереди (в соответствии со своим вариантом).

4.Печать стека/очереди.

5.Запись стека/очереди в файл.

6.Уничтожение стека/очереди.

7.Восстановление стека/очереди из файла.

**Реализация разработки программы**

Блок - схема программы

Stack.h

#pragma once

struct Node

{

int data;

Node\* next = nullptr;

};

class Stack

{

private:

int size;

Node\* head;

public:

Stack();

Stack(int, int);

Stack(const Stack&);

~Stack();

void push(int);

void pop();

void print();

void erase(int);

void insert(int, int);

void operator delete[](void\* ptr);

};

Stack.cpp

#include <iostream>

#include "Stack.h"

void Stack::operator delete[](void\* ptr)

{

while (((Stack\*)ptr)->head != nullptr)

{

Node\* temp = ((Stack\*)ptr)->head;

((Stack\*)ptr)->head = ((Stack\*)ptr)->head->next;

delete temp;

}

delete (Stack\*)ptr;

}

Stack::Stack()

{

this->head = nullptr;

this->size = 0;

}

Stack::Stack(int n, int k)

{

this->head = nullptr;

this->size = 0;

for (int i = 0; i < n; i++)

{

this->push(k);

}

}

Stack::Stack(const Stack& st)

{

this->head = nullptr;

this->size = 0;

Stack\* temp\_stack = new Stack;

Node\* temp\_node = st.head;

while (temp\_node != nullptr)

{

temp\_stack->push(temp\_node->data);

temp\_node = temp\_node->next;

}

temp\_node = temp\_stack->head;

while (temp\_node != nullptr)

{

this->push(temp\_node->data);

temp\_node = temp\_node->next;

}

temp\_stack->operator delete[](temp\_stack);

}

Stack::~Stack()

{

while (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

}

size = 0;

}

void Stack::push(int k)

{

Node\* new\_node = new Node;

new\_node->data = k;

new\_node->next = this->head;

this->head = new\_node;

this->size++;

}

void Stack::pop()

{

if (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

size--;

}

}

void print\_nodes(Node\* node)

{

if (node == nullptr)

{

return;

}

else

{

print\_nodes(node->next);

std::cout << node->data << ' ';

}

}

void Stack::print()

{

if (head == nullptr)

{

std::cout << "Стек пуст\n";

return;

}

else

{

print\_nodes(head);

}

}

void Stack::erase(int k)

{

Stack\* temp\_stack = new Stack;

while ((this->head->next != nullptr) && (this->head->data != k))

{

temp\_stack->push(this->head->data);

this->pop();

}

if (this->head->data == k)

{

this->pop();

while (temp\_stack->head != nullptr)

{

this->push(temp\_stack->head->data);

temp\_stack->pop();

}

}

else

{

while (temp\_stack->head != nullptr)

{

this->push(temp\_stack->head->data);

temp\_stack->pop();

}

}

temp\_stack->operator delete[](temp\_stack);

}

void Stack::insert(int n, int k)

{

Stack\* temp\_stack = new Stack;

for (int i = this->size; i > n; i--)

{

temp\_stack->push(this->head->data);

this->pop();

}

this->push(k);

while (temp\_stack->head != nullptr)

{

this->push(temp\_stack->head->data);

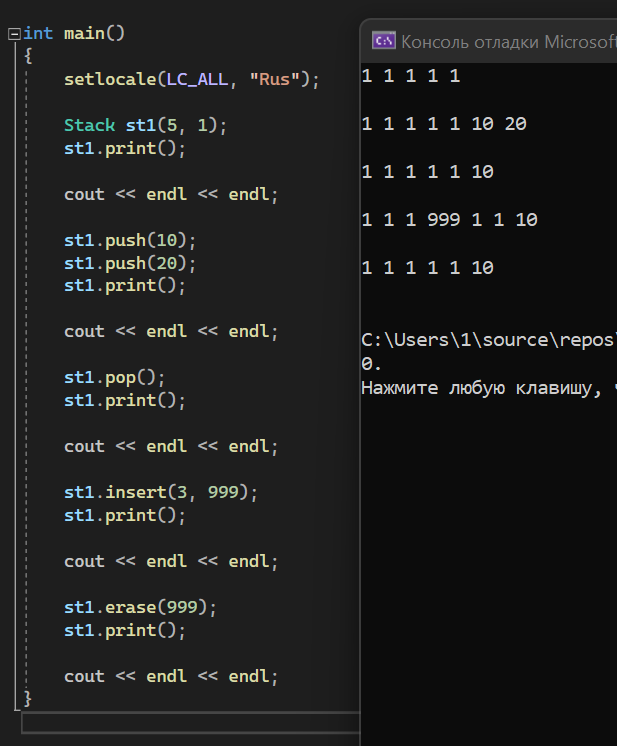
temp\_stack->pop();

}

temp\_stack->operator delete[](temp\_stack);

}

**Тестирование программы**



**Заключение**

Была разработана программа для знакомства с динамическими информационными структурами на примере стеков и очередей.