Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Информационные динамические структуры. Стеки»

Выполнил студент ИВТ-22-2б:

Захаров Дмитрий Сергеевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

Записи в линейном списке содержат ключевое поле типа int. Сформировать очередь. Удалить из нее К элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом

**Алгоритм решения**

1. Создать структуру узла
2. Создать функцию создания очереди
3. Создать функцию добавления элемента в очередь
4. Создать функцию удаления элемента из очереди
5. Создать функцию печати очереди
6. Создать функцию для записи очереди в файл
7. Создать функцию для уничтожения очереди
8. Создать главную функцию main

**Код программы**

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

int key;

Node\* next;

};

Node\* createQueue() {

return NULL;

}

void enqueue(Node\*\* queue, int key) {

Node\* newNode = new Node;

newNode->key = key;

newNode->next = NULL;

if (\*queue == NULL) {

\*queue = newNode;

return;

}

Node\* temp = \*queue;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

void dequeue(Node\*\* queue) {

if (\*queue == NULL) {

return;

}

Node\* temp = \*queue;

\*queue = temp->next;

delete temp;

}

void printQueue(Node\* queue) {

if (queue == NULL) {

cout << "Queue is empty." << endl;

return;

}

Node\* temp = queue;

while (temp != NULL) {

cout << temp->key << " ";

temp = temp->next;

}

cout << endl;

}

void writeToFile(Node\* queue, string fileName) {

ofstream file;

file.open(fileName);

if (file.is\_open()) {

Node\* temp = queue;

while (temp != NULL) {

file << temp->key << " ";

temp = temp->next;

}

file.close();

cout << "Queue written to file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

void destroyQueue(Node\*\* queue) {

while (\*queue != NULL) {

dequeue(queue);

}

cout << "Queue destroyed successfully." << endl;

}

void restoreStack(Node\*\* stack, Node\*\* queue) {

while (\*queue != NULL) {

int key = (\*queue)->key;

dequeue(queue);

Node\* newNode = new Node;

newNode->key = key;

newNode->next = \*stack;

\*stack = newNode;

}

cout << "Stack restored from queue successfully." << endl;

}

int main() {

Node\* queue = createQueue();

int k;

int val;

cout << "Enter the quantity of elements: ";

cin >> k;

for (int i = 1; i <= k; i++) {

cout << i << ". ";

cin >> val;

enqueue(&queue, val);

}

cout << "Initial queue: ";

printQueue(queue);

int keyToAdd = 40;

enqueue(&queue, keyToAdd);

cout << "Queue after adding key " << keyToAdd << ": ";

printQueue(queue);

int numToRemove = 2;

for (int i = 0; i < numToRemove; i++) {

dequeue(&queue);

}

cout << "Queue after removing " << numToRemove << " elements: ";

printQueue(queue);

writeToFile(queue, "queue.txt");

destroyQueue(&queue);

Node\* stack = NULL;

restoreStack(&stack, &queue);

return 0;

}