Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Алгоритмы и структуры данных»

Отчет к лабораторной работе № 1

«Абстрактный тип данных на примере реализации очереди в последовательной и связной памяти»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Рыженков Д.В.

Орёл, 2024г

Листинг файла main.cpp:

#include <iostream>

#include <string>

#include "QueueList.h"

#include <vector>

using namespace std;

class Detail {

public:

void inputDetail() {

string inCode = getValidString();

for (int i = 0; i < 4; i++)

{

this->code[i] = inCode[i];

}

this->code[4] = '\0';

this->time = getValidNumber();

}

char code[5] = "----";

int time = 0;

private:

int getValidNumber() {

int number;

while (true) {

cout << "Введите время: ";

cin >> number;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите корректное целое число!" << endl;

}

else {

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

break;

}

}

return number;

}

string getValidString() {

string input;

while (true) {

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введите код детали: ";

getline(cin, input);

if (input.length() >= 4) {

return input.substr(0, 4);

}

else {

cout << "Ошибка: строка должна содержать ровно 4 символа!" << endl;

}

}

}

};

int main()

{

setlocale(LC\_ALL, "ru");

Queue<Detail> queue;

string menu =

"\n1. Постановка на обработку\n"

"2. Временной переход\n"

"3. Снять деталь\n"

"4. Посмотреть очередь\n"

"0. Выход\n"

;

while (true) {

cout << menu;

string cmd;

cin >> cmd;

if (cmd == "0") {

return 0;

}

else if (cmd == "1") {

Detail det;

det.inputDetail();

if (queue.insert(det)) {

cout << "Успешно!" << endl;

}

else {

cout << "Очередь переполнена" << endl;

}

}

else if (cmd == "2") {

Detail\* det = queue.peek();

if (det != nullptr) {

det->time--;

if (det->time <= 0) {

cout << "Деталь с кодом " << det->code << " закончила обработку" << endl;

queue.dequeue();

}

}

}

else if (cmd == "3") {

Detail\* det = queue.peek();

if (det != nullptr) {

cout << "Деталь с кодом " << det->code << " преждевременно снята" << endl;

queue.dequeue();

}

else {

cout << "Очередь пуста" << endl;

}

}

else if (cmd == "4") {

vector<Detail> arr;

queue.getQueue(arr);

if (arr.size() == 0) {

cout << "Очередь пуста" << endl;

}

for (int i = 0; i < arr.size(); i++)

{

cout << ">> Code: " << arr[i].code << "; Time Remaining: " << arr[i].time << endl;

}

}

else {

cout << "Неизвестная команда" << endl;

}

}

}

Листинг файла Queue.h:

#pragma once

#include <vector>

template <typename ElType>

class Queue {

public:

    bool insert(ElType el) {

        if (isFull()) {

            return false;

        }

        elements[tail] = el;

        tail = (tail + 1) % maxElements;

        return true;

    }

    ElType\* peek() {

        if (isEmpty()) {

            return nullptr;

        }

        return &elements[head];

    }

    bool dequeue() {

        if (isEmpty()) {

            return false;

        }

        head = (head + 1) % maxElements;

        return true;

    }

    bool isFull() {

        return (tail + 1) % maxElements == head;

    }

    bool isEmpty() {

        return head == tail;

    }

    void getQueue(std::vector<ElType> &arr) {

        for (int i = head; i != tail; i = (i + 1) % maxElements)

        {

            arr.push\_back(elements[i]);

        }

    }

private:

    ElType elements[6];

    int maxElements = 6;

    int head = 0, tail = 0;

};

Листинг файла QueueList.h:

#pragma once

#include <vector>

template <typename T>

struct Node {

    Node(T el) {

        this->data = el;

    }

    T data;

    Node<T>\* next = nullptr;

};

template <typename ElType>

class Queue {

public:

    bool insert(ElType el) {

        Node<ElType>\* newNode = new Node<ElType>(el);

        if (tail != nullptr) {

            tail->next = newNode;

        }

        else {

            head = newNode;

        }

        tail = newNode;

        return true;

    }

    ElType\* peek() {

        if (isEmpty()) {

            return nullptr;

        }

        return &head->data;

    }

    bool dequeue() {

        if (isEmpty()) {

            return false;

        }

        Node<ElType>\* oldHead = head;

        head = head->next;

        if (head == nullptr) {

            tail = nullptr;

        }

        delete oldHead;

        return true;

    }

    bool isEmpty() {

        return head == nullptr;

    }

    void getQueue(std::vector<ElType> &arr) {

        Node<ElType>\* tmp = head;

        while (tmp != nullptr)

        {

            arr.push\_back(tmp->data);

            tmp = tmp->next;

        }

    }

private:

    Node<ElType>\* head = nullptr;

    Node<ElType>\* tail = nullptr;

};