Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №5

Тема: «Работа с очередью»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

**I. Ответы на контрольные вопросы**

**1. Что представляет собой очередь?**

Очередь — это структура данных, которая построена по принципу LILO (last in — last out: последним пришел — последним вышел)

**2. Какие известны виды очередей?**

Очереди делятся на линейные и кольцевые (циклические). По количеству позиций записи и считывания – на простые и приоритетные.

**3. На основе каких структур данных могут организовываться очереди?**

Очереди могут организовываться на основе массива и списка.

**4. Какой характер имеет операция считывания для очередей?**

Считывание происходит последовательно.

**5. Какими свойствами обладают очереди?**

Добавление нового элемента происходит в конец очереди, а удаление происходит из начала очереди.

**6. Какие операции над элементами характерны для очередей?**

* Инициализация очереди;
* Проверка наличия элементов в очереди;
* Добавление нового элемента в конец очереди;
* Удаление элемента из начала очереди;
* Вывод элементов очереди;
* Очистка очереди.

**7. Перечислите основные отличия очереди от стека.**

Добавление и удаление происходят по разным принципам: в стеках (FILO – first in last out), а в очередях (FIFO – first in first out).

**8. К каким позициям очереди возможен доступ при записи и чтении информации?**

Считывать можно любой элемент, а запись можно производить только в конец очереди.

**9. Типы очереди.**

* Простая очередь
* Круговая Очередь
* Очередь приоритетов

**10. Чем отличается приоритетная очередь от простоя?**

В приоритетной очереди элемент может иметь более высокий приоритет и добавляется не в самый конец, а за тем элементом, который имеет схожий приоритет, а в случае отсутствия такого элемента добавление происходит в начало очереди.

**11. В чем особенность циклической очереди?**

В циклической очереди за последним её элементом следует первый её элемент, тем самым образуя «кольцо».

**12. В чем преимущества циклической очереди?**

У циклической очереди есть преимущество перед очередью, сделанной на основе массива, так как у массива ограниченное количество чисел, а у циклической очереди нет конца.

**13. Для решения каких задач применяются очереди?**

* Для диспетчеризации задач операционной системы;
* Для буферизации ввода/вывода;
* Для моделирования процессов.

**II. Выполнить упражнение**

**1. Постановка задачи**

1. Разработать алгоритм, блок схему и программный код, реализующий все операции допустимые к очереди-

* инициализация очереди;
* добавление элемента в очередь;
* удаление элемента из очереди;
* просмотр элементов очереди;
* уничтожение очереди;
* поиска элемента по ключу;
* поиск позиции по заданному значению;
* нахождения количества элементов в очереди,
* очистка очереди.

2. Разработать пользовательский интерфейс программы.

3. Проверить корректность ввода данных.

**2. Описание входных и выходных данных**

**Исходные данные:**

**Queue\* \_last = NULL;**

**Queue\* \_first = NULL;**

**void push(Queue\*& \_Last, Queue\*& \_First);** - функция инициализации и добавления;

**void output(Queue\* \_First);** - функция вывода;

**void pop(Queue\*& \_Last, Queue\*& \_First);** - функция удаления;

**void seachByIndex(Queue\* \_First);** - функция поиска по индексу;

**void searchByValue(Queue\* \_First);** - функция поиска по значению;

**int count(Queue\* \_First);** - функция вывода кол-во элементов в очереди;

**void delQ(Queue\*& \_First);** - функция уничтожения очереди;

**void clearQ(Queue\*& \_Last, Queue\*& \_First);** - функция очищения очереди;

int main()



void push(Queue\*& \_Last, Queue\*& \_First)



void output(Queue\* \_First)



void pop(Queue\*& \_Last, Queue\*& \_First)



void seachByIndex(Queue\* \_First)



void searchByValue(Queue\* \_First)



int count(Queue\* \_First)



void delQ(Queue\*& \_First)



void clearQ(Queue\*& \_Last, Queue\*& \_First)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct Queue {

int \_item;

Queue\* \_next;

};

int x = 0, i = 1;

int \_count = 0;

bool id = false;

void push(Queue\*& \_Last, Queue\*& \_First);

void output(Queue\* \_First);

void pop(Queue\*& \_Last, Queue\*& \_First);

void seachByIndex(Queue\* \_First);

void searchByValue(Queue\* \_First);

int count(Queue\* \_First);

void delQ(Queue\*& \_First);

void clearQ(Queue\*& \_Last, Queue\*& \_First);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Queue\* \_last = NULL;

Queue\* \_first = NULL;

bool f = true;

int choice;

push(\_last, \_first);

id = true;

while (f) {

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Добавление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Вывод элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Удаление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Поиска элемента по индексу |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 5.Поиск позиции по значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 6.Количество элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 7.Уничтожения очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 8.Очистка очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 9.Выход |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1: push(\_last, \_first); id = true; break;

case 2: output(\_first); break;

case 3: pop(\_last, \_first); break;

case 4: seachByIndex(\_first); break;

case 5: searchByValue(\_first); break;

case 6: cout << "Кол - во элементов очереди = " << count(\_first) << endl; break;

case 7: delQ(\_first); \_first = 0; break;

case 8: clearQ(\_last, \_first); break;

case 9: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

delete \_last;

delete \_first;

return 0;

}

void push(Queue\*& \_Last, Queue\*& \_First) {

if (id == false) {

\_count++;

x = 0;

Queue\* q = new Queue;

cout << "Инициализируйте первый элемент:";

cin >> x;

system("cls");

q->\_item = x;

q->\_next = \_Last;

\_First = q;

\_Last = \_First;

cout << "Первый элемент инициализован" << endl;

}

else {

\_count++;

x = 0;

Queue\* q = new Queue;

cout << "Введите новый элемент:";

cin >> x;

q->\_item = x;

q->\_next = NULL;

\_Last->\_next = q;

\_Last = q;

cout << "Элемент " << x << " добавлен" << endl;

}

}

void output(Queue\* \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего показывать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

q = \_First;

while (q != NULL) {

x = q->\_item;

cout << "Queue[" << i << "] = " << x << endl;

i++;

q = q->\_next;

}

}

}

void pop(Queue\*& \_Last, Queue\*& \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего удалять" << endl;

}

else {

\_count--;

Queue\* q = new Queue;

q = \_First;

\_First = \_First->\_next;

if (q->\_next == NULL) {

\_Last = NULL;

}

cout << "Элемент " << q->\_item << " удален" << endl;

delete q;

}

if (\_count == 0)

{

id = false;

}

}

void seachByIndex(Queue\* \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

q = \_First;

cout << "Введите индекс искоемого элемента:";

cin >> x;

if (x <= 0 || x > \_count) {

cout << "Такого индекса нету!!!" << endl;

}

else {

while (q != NULL) {

if (x == i) {

cout << "Queue[" << i << "] = " << q->\_item << endl;

}

q = q->\_next;

i++;

}

}

}

}

void searchByValue(Queue\* \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

bool k = false;

q = \_First;

cout << "Введите искоемый элемент:";

cin >> x;

while (q != NULL) {

if (q->\_item == x) {

k = true;

cout << "Queue[" << i << "] = " << x << endl;

}

q = q->\_next;

i++;

}

if (k == false) {

cout << "искомый элемент не найден!" << endl;

}

}

}

int count(Queue\* \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

return 0;

}

else {

i = 0;

Queue\* q = \_First;

while (q != NULL) {

q = q->\_next;

i++;

}

return i;

}

}

void delQ(Queue\*& \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего уничтожать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_count = 0;

id = false;

cout << "Очередь уничтожена" << endl;

}

}

void clearQ(Queue\*& \_Last, Queue\*& \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего очищать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_Last = NULL;

\_count = 0;

id = false;

cout << "Очередь очищена" << endl;

}

}

**1.Постановка задачи**

Разработать алгоритм, блок схему и программу создания кольцевой очереди, предусмотреть выполнение основных операций над структурой данных.

**2.Описание входных и выходных данных**

**Исходные данные:**

**void round(Queue\*& \_Last, Queue\*& \_First); -** функция зацикливания очереди;

void round(Queue\*& \_Last, Queue\*& \_First)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct Queue {

int \_item;

Queue\* \_next;

};

int x = 0, i = 1;

int \_count = 0;

bool id = false, ifround = false;

void push(Queue\*& \_Last, Queue\*& \_First);

void output(Queue\* \_First);

void pop(Queue\*& \_Last, Queue\*& \_First);

void seachByIndex(Queue\* \_First);

void searchByValue(Queue\* \_First);

int count(Queue\* \_First);

void round(Queue\*& \_Last, Queue\*& \_First);

void delQ(Queue\*& \_Last, Queue\*& \_First);

void clearQ(Queue\*& \_Last, Queue\*& \_First);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Queue\* \_last = NULL;

Queue\* \_first = NULL;

bool f = true;

int choice;

push(\_last, \_first);

id = true;

while (f) {

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Добавление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Вывод элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Удаление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Поиска элемента по индексу |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 5.Поиск позиции по значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 6.Количество элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 7.Закольцевать очередь |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 8.Уничтожения очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 9.Очистка очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 10.Выход |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1: push(\_last, \_first); id = true; break;

case 2: output(\_first); break;

case 3: pop(\_last, \_first); break;

case 4: seachByIndex(\_first); break;

case 5: searchByValue(\_first); break;

case 6: count(\_first); break;

case 7: round(\_first, \_last); break;

case 8: delQ(\_last, \_first); break;

case 9: clearQ(\_last, \_first); break;

case 10: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

delete \_last;

delete \_first;

return 0;

}

void push(Queue\*& \_Last, Queue\*& \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя добавлять" << endl;

}

else

{

if (id == false) {

\_count++;

x = 0;

Queue\* q = new Queue;

cout << "Инициализируйте первый элемент:";

cin >> x;

system("cls");

q->\_item = x;

q->\_next = \_Last;

\_First = q;

\_Last = \_First;

cout << "Первый элемент инициализован" << endl;

}

else {

\_count++;

x = 0;

Queue\* q = new Queue;

cout << "Введите новый элемент:";

cin >> x;

q->\_item = x;

q->\_next = NULL;

\_Last->\_next = q;

\_Last = q;

cout << "Элемент " << x << " добавлен" << endl;

}

}

}

void output(Queue\* \_First) {

if (ifround == false) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего показывать" << endl;

}

else {

x = 0;

i = 1;

int k = 0;

Queue\* q = new Queue;

q = \_First;

while (q != NULL) {

x = q->\_item;

cout << "Queue[" << i << "] = " << x << endl;

i++;

k++;

q = q->\_next;

}

}

}

else {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего показывать" << endl;

}

else {

x = 0;

i = 1;

int k = 0;

Queue\* q = new Queue;

q = \_First;

while (q != NULL) {

x = q->\_item;

cout << "Queue[" << i << "] = " << x << endl;

i++;

k++;

q = q->\_next;

if (i == \_count + 1)

{

i = 1;

cout << endl;

}

if (k == (\_count \* 5))

{

break;

}

}

}

}

}

void pop(Queue\*& \_Last, Queue\*& \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя удалять" << endl;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего удалять" << endl;

}

else {

\_count--;

Queue\* q = new Queue;

q = \_First;

\_First = \_First->\_next;

if (q->\_next == NULL) {

\_Last = NULL;

}

cout << "Элемент " << q->\_item << " удален" << endl;

delete q;

}

if (\_count == 0)

{

id = false;

}

}

}

void seachByIndex(Queue\* \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя искать по индексу" << endl;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

q = \_First;

cout << "Введите индекс искоемого элемента:";

cin >> x;

if (x <= 0 || x > \_count) {

cout << "Такого индекса нету!!!" << endl;

}

else {

while (q != NULL) {

if (x == i) {

cout << "Queue[" << i << "] = " << q->\_item << endl;

}

q = q->\_next;

i++;

}

}

}

}

}

void searchByValue(Queue\* \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя искать по значению" << endl;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

bool k = false;

q = \_First;

cout << "Введите искоемый элемент:";

cin >> x;

while (q != NULL) {

if (q->\_item == x) {

k = true;

cout << "Queue[" << i << "] = " << x << endl;

}

q = q->\_next;

i++;

}

if (k == false) {

cout << "искомый элемент не найден!" << endl;

}

}

}

}

int count(Queue\* \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя узнать кол - во очереди" << endl;

return 0;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

return 0;

}

else {

i = 0;

Queue\* q = \_First;

while (q != NULL) {

q = q->\_next;

i++;

}

cout << "Кол - во элементов очереди = " << i << endl;

return i;

}

}

}

void round(Queue\*& \_Last, Queue\*& \_First) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

if (ifround == false) {

\_Last->\_next = \_First;

ifround = true;

cout << "Last->next: " << \_Last->\_next << endl;

cout << "First: " << \_First << endl;

}

else {

\_Last->\_next = NULL;

ifround = false;

}

if (ifround == true) {

cout << "Очередь закольцована" << endl;

}

else

{

cout << "Очередь откальцована" << endl;

}

}

void delQ(Queue\*& \_Last, Queue\*& \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя уничтожить" << endl;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего уничтожать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_Last = NULL;

\_count = 0;

id = false;

cout << "Очередь уничтожена" << endl;

}

}

}

void clearQ(Queue\*& \_Last, Queue\*& \_First) {

if (ifround == true)

{

cout << "У вас закольцована очередь!!!" << endl << "Нельзя очистить" << endl;

}

else

{

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl << "Нечего очищать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_Last = NULL;

\_count = 0;

id = false;

cout << "Очередь очищена" << endl;

}

}

}

**1.Постановка задачи**

Разработать алгоритм, блок схему и программу сцепления (конкатенации) двух очередей. Предусмотреть выполнение основных операций над конкатенированной очередью.

**2.Описание входных и выходных данных**

**Исходные данные:**

**void concotination(Queue\*& \_Last1, Queue\*& \_First1, Queue\*& \_Last2, Queue\*& \_First2, int choice, bool& id); -** функция конкатенации очередей;

**void concotination(Queue\*& \_Last1, Queue\*& \_First1, Queue\*& \_Last2, Queue\*& \_First2, int choice, bool& id)**



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct Queue {

int \_item;

Queue\* \_next;

};

int x = 0, i = 1;

void push(Queue\*& \_Last, Queue\*& \_First, bool& id);

void output(Queue\* \_First);

void pop(Queue\*& \_Last, Queue\*& \_First, bool& id);

void seachByIndex(Queue\* \_First);

void searchByValue(Queue\* \_First);

int count(Queue\* \_First);

void concotination(Queue\*& \_Last1, Queue\*& \_First1, Queue\*& \_Last2, Queue\*& \_First2, int choice, bool& id);

void delQ(Queue\*& \_Last, Queue\*& \_First, bool& id);

void clearQ(Queue\*& \_Last, Queue\*& \_First, bool& id);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Queue\* \_last1 = NULL;

Queue\* \_first1 = NULL;

Queue\* \_last2 = NULL;

Queue\* \_first2 = NULL;

bool f1 = true, f2 = true;

bool id1 = false, id2 = false;

int choice;

push(\_last1, \_first1, id1);

push(\_last2, \_first2, id2);

id1 = true;

id2 = true;

while (f1) {

f2 = true;

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1. Меню очереди № 1 |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2. Меню очереди № 2 |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Конкатенация очередей |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Выход |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1:

while (f2) {

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню Очереди № 1: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Добавление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Вывод элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Удаление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Поиска элемента по индексу |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 5.Поиск позиции по значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 6.Количество элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 7.Уничтожения очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 8.Очистка очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 9.Назад |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1: push(\_last1, \_first1, id1); id1 = true; break;

case 2: output(\_first1); break;

case 3: pop(\_last1, \_first1, id1); break;

case 4: seachByIndex(\_first1); break;

case 5: searchByValue(\_first1); break;

case 6: count(\_first1); break;

case 7: delQ(\_last1, \_first1, id1); break;

case 8: clearQ(\_last1, \_first1, id1); break;

case 9: f2 = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}; break;

case 2:

while (f2) {

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню Очереди № 2: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Добавление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Вывод элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Удаление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Поиска элемента по индексу |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 5.Поиск позиции по значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 6.Количество элементов |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 7.Уничтожения очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 8.Очистка очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 9.Назад |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1: push(\_last2, \_first2, id2); id2 = true; break;

case 2: output(\_first2); break;

case 3: pop(\_last2, \_first2, id2); break;

case 4: seachByIndex(\_first2); break;

case 5: searchByValue(\_first2); break;

case 6: count(\_first2); break;

case 7: delQ(\_last2, \_first2, id2); break;

case 8: clearQ(\_last2, \_first2, id2); break;

case 9: f2 = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}; break;

case 3:

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1. Первая очередь + Второая очередь |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2. Вторая очередь + Первая очередь |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Назад |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

}

if (choice == 1) {

concotination(\_last1, \_first1, \_last2, \_first2, choice, id2);

}

else {

concotination(\_last1, \_first1, \_last2, \_first2, choice, id1);

}; break;

case 4: f1 = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

delete \_last1;

delete \_first1;

delete \_last2;

delete \_first2;

return 0;

}

void push(Queue\*& \_Last, Queue\*& \_First, bool& id) {

if (id == false) {

x = 0;

Queue\* q = new Queue;

cout << "Инициализируйте первый элемент:";

cin >> x;

system("cls");

q->\_item = x;

q->\_next = \_Last;

\_First = q;

\_Last = \_First;

cout << "Первый элемент инициализован" << endl;

}

else {

x = 0;

Queue\* q = new Queue;

cout << "Введите новый элемент:";

cin >> x;

q->\_item = x;

q->\_next = NULL;

\_Last->\_next = q;

\_Last = q;

cout << "Элемент " << x << " добавлен" << endl;

}

}

void output(Queue\* \_First) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего показывать" << endl;

}

else {

x = 0;

i = 1;

int k = 0;

Queue\* q = new Queue;

q = \_First;

while (q != NULL) {

x = q->\_item;

cout << "Queue[" << i << "] = " << x << endl;

i++;

k++;

q = q->\_next;

}

}

}

void pop(Queue\*& \_Last, Queue\*& \_First, bool& id) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего удалять" << endl;

}

else {

Queue\* q = new Queue;

q = \_First;

\_First = \_First->\_next;

if (q->\_next == NULL) {

\_Last = NULL;

}

cout << "Элемент " << q->\_item << " удален" << endl;

delete q;

}

if (\_First == 0)

{

id = false;

}

}

void seachByIndex(Queue\* \_First) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

q = \_First;

cout << "Введите индекс искоемого элемента:";

cin >> x;

if (x <= 0) {

cout << "Такого индекса нету!!!" << endl;

}

else {

while (q != NULL) {

if (x == i) {

cout << "Queue[" << i << "] = " << q->\_item << endl;

}

q = q->\_next;

i++;

}

}

}

}

void searchByValue(Queue\* \_First) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего искать" << endl;

}

else {

x = 0;

i = 1;

Queue\* q = new Queue;

bool k = false;

q = \_First;

cout << "Введите искоемый элемент:";

cin >> x;

while (q != NULL) {

if (q->\_item == x) {

k = true;

cout << "Queue[" << i << "] = " << x << endl;

}

q = q->\_next;

i++;

}

if (k == false) {

cout << "искомый элемент не найден!" << endl;

}

}

}

int count(Queue\* \_First) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl;

return 0;

}

else {

i = 0;

Queue\* q = \_First;

while (q != NULL) {

q = q->\_next;

i++;

}

cout << "Кол - во элементов очереди = " << i << endl;

return i;

}

}

void concotination(Queue\*& \_Last1, Queue\*& \_First1, Queue\*& \_Last2, Queue\*& \_First2, int choice, bool& id)

{

if (\_First1 <= 0) {

cout << "Очередь № 1 пуст!!!" << endl;

return;

}

if (\_First2 <= 0) {

cout << "Очередь № 2 пуст!!!" << endl;

return;

}

if (choice == 1)

{

\_Last1->\_next = \_First2;

\_Last1 = \_Last2;

\_Last2 = NULL;

\_First2 = NULL;

id = false;

}

else if (choice == 2)

{

\_Last2->\_next = \_First1;

\_Last2 = \_Last1;

\_Last1 = NULL;

\_First1 = NULL;

id = false;

}

else

{

return;

}

}

void delQ(Queue\*& \_Last, Queue\*& \_First, bool& id) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего уничтожать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_Last = NULL;

id = false;

cout << "Очередь уничтожена" << endl;

}

}

void clearQ(Queue\*& \_Last, Queue\*& \_First, bool& id) {

if (\_First <= 0) {

cout << "Очередь пуст!!!" << endl << "Нечего очищать" << endl;

}

else {

Queue\* q = new Queue;

while (\_First != 0) {

q = \_First;

\_First = \_First->\_next;

delete q;

}

\_Last = NULL;

id = false;

cout << "Очередь очищена" << endl;

}

}

**1.Постановка задачи**

Разработать алгоритм, блок схему и программный код для очереди с приоритетом. Предусмотреть выполнение основных операций над структурой данных.

**2. Описание входных и выходных данных**

**void inition(P\_Queue\*& \_Q); -** функция для инициализации элементов;

**void push(P\_Queue\*& \_Q, int value, int p); -** функции для добавления элементов;

**void output(P\_Queue\*& \_Q); -** функция для вывода;

**void pop(P\_Queue\*& \_Q); -** функция для удаления элементов;

int main()

void inition(P\_Queue\*& \_Q)



void push(P\_Queue\*& \_Q, int value, int p)



void output(P\_Queue\*& \_Q)



void pop(P\_Queue\*& \_Q)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct P\_Queue{

int value;

int pos;

P\_Queue\* \_next;

};

void inition(P\_Queue\*& \_Q);

void push(P\_Queue\*& \_Q, int value, int p);

void output(P\_Queue\*& \_Q);

void pop(P\_Queue\*& \_Q);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

P\_Queue\* Q = new P\_Queue;

bool f = true;

int choice, x, i;

cout << "Инициализируйте первый элемент:";

inition(Q);

Q->\_next = 0;

while (f) {

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Добавление элемента из очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Просмотр элементов очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Удаление элемента из очереди |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Выход |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

if (cin.fail())

{

cout << "Введен некоректный символ!!!" << endl;

break;

}

else {

switch (choice) {

case 1:

cout << "Введите число которое хотите добавить:";

cin >> x;

cout << "Введите приоритет числа позицию:";

cin >> i;

push(Q, x, i);

break;

case 2: output(Q); break;

case 3: pop(Q); break;

case 4: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

return 0;

}

void inition(P\_Queue\*& \_Q) {

P\_Queue\* Q = new P\_Queue;

int x; cin >> x;

Q->\_next = \_Q;

Q->value = x;

Q->pos = 1;

\_Q = Q;

}

void push(P\_Queue\*& \_Q, int value, int p) {

P\_Queue\* q, \* t;

q = NULL;

t = new P\_Queue;

t->value = value;

t->pos = p;

if (\_Q == NULL || p < \_Q->pos) {

t->\_next = \_Q;

\_Q = t;

} else {

q = \_Q;

while (q->\_next != NULL && q->\_next->pos <= p) {

q = q->\_next;

}

t->\_next = q->\_next;

q->\_next = t;

}

}

void output(P\_Queue\*& \_Q) {

P\_Queue\* Q;

if (\_Q == NULL) {

cout << "Очередь пуста!\n";

} else {

Q = \_Q;

cout << "\t------------------------------------" << endl;

cout << "\t| Элементы\t:\tПриоритеты |\n";

cout << "\t------------------------------------" << endl;

while (Q != NULL) {

cout << "\t|" << Q->value << "\t\t:\t" << Q->pos << "\t |" << endl;

cout << "\t------------------------------------" << endl;

Q = Q->\_next;

}

}

}

void pop(P\_Queue\*& \_Q) {

P\_Queue\* Q;

if (\_Q == NULL) {

cout << "Очередь пуста!\n";

} else {

Q = \_Q;

cout << "Удален элемент: " << Q->value << endl;

\_Q = \_Q->\_next;

free(Q);

}

}