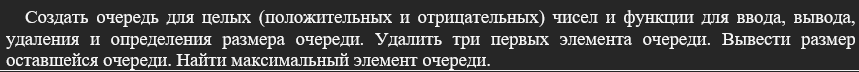
Лабораторная работа 8

По дисциплине «Основы алгоритмизации и программирования»

На тему «Полустатические структуры данных: очереди»

***Задание#1***



*Macros.h*

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <cstring>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

*MyQueue.h*

#pragma once

#define MYQUEUE1\_EQE 0x0000 // возврат в случае пустоты очереди

struct Queue

{

int Head; // голова очереди

int Tail; // хвост очереди

int Size; // размер очереди (макс. колич.+1)

void\*\* Data; // хранилище данных очереди

Queue(int size) // физический размер очереди

{

Head = Tail = 0;

Data = new void\* [Size = size + 1];

}

bool isFull() const; // очередь заполнена?

bool isEmpty() const; // очередь пуста?

};

Queue createQueue(int n); // n – макс. количество

Queue createQueue(const Queue& pq); // создать очередь по образцу

bool enQueue(Queue& q, void\* x); // добавить x

void\* delQueue(Queue& q); // удалить элемент

void\* peekQueue(const Queue& q); // получить первый элемент

int clearQueue(Queue& q); // очистить очередь

void releaseQueue(Queue& q); // освободить ресурсы #pragma once

*MyQueue.cpp*

#include "MyQueue.h"

#include "Macros.h"

Queue createQueue(int n) // Выделить ресурс для очереди

{

ret \*(new Queue(n));

}

Queue createQueue(const Queue& pq) // Создать очередь

{

Queue\* rc = new Queue(pq.Size - 1);

rc->Head = pq.Head;

rc->Tail = pq.Tail;

for (int i = 0; i < pq.Size; i++)

rc->Data[i] = pq.Data[i];

ret \*rc;

}

bool Queue::isFull() const // Очередь заполнена?

{

ret (Head % Size == (Tail + 1) % Size);

}

bool Queue::isEmpty() const // Очередь пуста?

{

ret (Head % Size == Tail % Size);

}

bool enQueue(Queue& q, void\* x) // Добавить элемент x

{

bool rc = true;

if (rc = !q.isFull())

{

q.Data[q.Tail] = x;

q.Tail = (q.Tail + 1) % q.Size;

}

else

rc = false;

ret rc;

}

void\* delQueue(Queue& q) // Удалить элемент

{

void\* rc = (void\*)MYQUEUE1\_EQE;

if (!q.isEmpty())

{

rc = q.Data[q.Head];

q.Head = (q.Head + 1) % q.Size;

}

else

rc = nullptr;

ret rc;

}

void\* peekQueue(const Queue& q) // Получить первый элемент очереди

{

void\* rc = (void\*)MYQUEUE1\_EQE;

if (!q.isEmpty())

rc = q.Data[q.Head];

else

rc = nullptr;

ret rc;

}

int clearQueue(Queue& q) // Очистить очередь

{

int rc = (q.Tail - q.Head) >= 0 ? q.Tail - q.Head : (q.Size - q.Head + q.Tail + 1);

q.Tail = q.Head = 0;

ret rc; // количество элементов до очистки

}

void releaseQueue(Queue& q) // Освободить ресурсы очереди

{

delete[] q.Data;

q.Size = 1;

q.Head = q.Tail = 0;

}

*Main.cpp*

#include "Macros.h"

#include "MyQueue.h"

using namespace std;

struct Que

{

int a;

char b;

};

void printQueue(Queue& s)

{

while (!s.isEmpty())

{

vi ((Que\*)peekQueue(s))->a << " " << ((Que\*)peekQueue(s))->b; nl

delQueue(s);

}

}

Queue removeFirstThree(Queue& s)

{

for (int i = 0; i < 3; ++i)

delQueue(s);

ret s;

}

int main()

{

SCOCP1251

SCCP1251

Queue q1 = createQueue(4);

Que a1 = { 1, '1' }, a2 = { 2, '2' }, a3 = { 3, '3' };

enQueue(q1, &a1);

enQueue(q1, &a2);

enQueue(q1, &a3);

Que\* a4 = new Que;

a4->a = 4;

a4->b = '4';

enQueue(q1, a4);

vi "Размер очереди до удаления: " << q1.Size - 1; nl

q1 = removeFirstThree(q1);

vi "Размер оставшейся очереди: " << q1.Size - 1; nl

int maxElement = INT\_MIN;

while (!q1.isEmpty())

{

Que\* element = (Que\*)delQueue(q1);

maxElement = max(maxElement, element->a);

}

vi "Максимальный элемент очереди: " << maxElement; nl

pause

ret 0;

}

Очередь 1 2 3 4

