ГУО «БГУИР»

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Отчет по

Лабораторной работе №3

Динамическая структура ОЧЕРЕДЬ

Подготовил:

Студент гр.222401

Рабинчук И.М.

Проверила:

Семижон Е.А.

Минск 2023

Цель: изучить возможности работы со списками, организованными в виде очереди.

Написать программу по созданию, добавлению (в начало, в конец), просмотру (с начала, с конца) и решению поставленной в лаб. работе № 2 задачи для двунаправленных линейных списков.

Вариант: 9

Код:

#include <iostream>

#include <conio.h>

using namespace std;

struct Queue

{

int info;

Queue \*next;

Queue \*prev;

}\*begin, \*end, \*t;

void Create(Queue \*\*begin, Queue \*\*end, int i)

{

t = new Queue;

t->info = i;

t->next = t->prev = NULL;

\*begin = \*end = t;

}

void Add(bool flag, Queue \*\*begin, Queue \*\*end, int i)

{

t = new Queue;

t->info = i;

if (flag == true) //Добавление в начало

{

t->prev = NULL;

t->next = \*begin;

(\*begin)->prev = t;

\*begin = t;

}

else

{

t->next = NULL; //Добавление в конец

t->prev = \*end;

(\*end)->next = t;

\*end = t;

}

}

void View(bool flag, Queue \*begin)

{

t = begin;

while (t != NULL)

{

cout << t->info <<" ";

if (flag == true)

t=t->next;

else

t = t->prev;

}

cout << endl;

}

void Delete\_all(Queue \*\*begin)

{

Queue \*t;

if (\*begin = NULL)

cout << "Очередь пуста" << endl;

while(\*begin != NULL)

{

t = \*begin;

\*begin = (\*begin)->next;

delete t;

}

}

Queue \*Max(Queue \*begin)

{

Queue \*max = begin;

while (begin)

{

if (begin->info > max->info)

max = begin;

if (begin->info == max->info)

max = begin; //???

begin = begin->next;

}

return max;

}

Queue \*Min(Queue \*begin)

{

Queue\* min = begin;

while (begin)

{

if (begin->info < min->info)

min = begin;

if (begin->info == min->info)

min = begin; //???

begin = begin->next;

}

return min;

}

Queue\* Work(Queue \*begin)

{

if (begin == NULL)

return begin;

Queue\* max = Max(begin);

Queue\* min = Min(begin);

if (max == min)

return begin;

if (min == max)

return begin;

if (max->next == min)

return begin;

if (min->next == max)

return begin;

int del\_numbs = 0;

Queue\* current=begin;

Queue\* folowwing = current->next;

while (folowwing != NULL)

{

if (folowwing == max)

{

Queue \*tmp\_1 =folowwing->next;

while (tmp\_1 != min)

{

Queue\* tmp\_2 = tmp\_1;

tmp\_1 = tmp\_1->next;

delete tmp\_2;

}

folowwing->next= min;

return begin;

}

if (folowwing == min)

{

Queue \*tmp\_1 = folowwing->next;

while (tmp\_1 != max)

{

Queue\* tmp\_2 = tmp\_1;

tmp\_1 = tmp\_1->next;

delete tmp\_2;

}

folowwing->next = max;

return begin;

}

current = folowwing;

folowwing = folowwing->next;

}

return begin;

}

Queue\* Work\_2(Queue \*begin, Queue \*end)

{

if (begin == NULL)

return begin;

Queue\* max = Max(begin);

Queue\* min = Min(begin);

if (max == min)

return begin;

if (min == max)

return begin;

if (max->next == min)

return begin;

if (min->next == max)

return begin;

Queue\* current = begin;

Queue\* folowwing = current->next;

Queue\* current\_end = end;

Queue\* folowwing\_end = current\_end->prev;

while (folowwing!=NULL)

{

if (folowwing == max)

{

while (folowwing\_end != min)

{

folowwing\_end = folowwing\_end->prev;

}

folowwing\_end = min;

folowwing->next = folowwing\_end;

folowwing\_end->prev = folowwing;

return begin;

}

if (folowwing == min)

{

while (folowwing\_end != max)

folowwing\_end = folowwing\_end->prev;

folowwing\_end = max;

folowwing->next = folowwing\_end;

folowwing\_end->prev = folowwing;

return begin;

}

current = folowwing;

folowwing = folowwing->next;

}

return begin;

}

int main()

{

setlocale(LC\_ALL, "rus");

Queue \*begin = NULL;

Queue \*end = NULL;

int n, i;

bool flag = false;

bool menu = true;

bool flag\_menu = false;

char sw;

while (menu == true)

{

cout << "\t\t\t\t\t\t\t Меню \n" << endl;

cout << "1)Создание первого элемента" << endl;

cout << "2)Добавление элементов в очередь" << endl;

cout << "3)Вывод очереди" << endl;

cout << "4)Очистка очереди" << endl;

cout << "5)Выполнение работы первым способом" << endl;

cout << "6)Выполнение работы вторым способом" << endl;

cout << "7)Завершить работу программы" << endl;

cout << endl;

do

{

sw = \_getch();

switch (sw)

{

case '1':

{

cout << "Введите первый элемент очереди: ";

while (!(cin >> i))

{

cout << "Введено некорректное значение. Введите значение заново." << endl;

cout << endl;

cin.clear();

cin.ignore();

cout << "Число i: " << endl;

}

cout << endl;

Create(&begin, &end, i);

menu = true;

flag\_menu = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '2':

{

cout << "1)Добавление с начала" << endl;

cout << "2)Добавление с конца" << endl;

sw = \_getch();

do {

switch (sw)

{

case '1':

{

cout << "Добавление с начала: " << endl;

flag = true;

cout << "Введите кол-во добавляемы элементов: ";

cin >> n;

cout << endl;

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

{

cin >> i;

Add(flag, &begin, &end, i);

}

flag\_menu = true;

break;

}

case '2':

{

cout << "Добавление с конца: " << endl;

flag = false;

cout << "Введите кол-во добавляемы элементов: ";

cin >> n;

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

{

cin >> i;

Add(flag, &begin, &end, i);

}

flag\_menu = true;

break;

}

default:

{

menu = true;

flag\_menu = false;

break;

}

}

} while (flag\_menu == false);

menu = true;

flag\_menu = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '3':

{

if (begin == NULL)

cout << "Очередь пуста" << endl;

cout << "1)Вывод с начала" << endl;

cout << "2)Вывод с конца" << endl;

sw = \_getch();

do {

switch (sw)

{

case '1':

{

cout << "Вывод с начала:" << endl;

flag = true;

t = begin;

View(flag, t);

flag\_menu = true;

break;

}

case '2':

{

cout << "Вывод с конца:" << endl;

flag = false;

t = end;

View(flag, t);

flag\_menu = true;

break;

}

default:

{

flag\_menu = false;

menu = true;

break;

}

}

} while (flag\_menu == false);

menu = true;

flag\_menu = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '4':

{

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

{

Delete\_all(&begin);

}

menu = true;

flag = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '5':

{

begin=Work(begin);

menu = true;

flag\_menu = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '6':

{

begin = Work\_2(begin, end);

menu = true;

flag\_menu = true;

cout << endl;

cout << "Выберите новую операцию:" << endl;

cout << endl;

break;

}

case '7':

{

flag\_menu = true;

menu = false;

break;

}

default:

{

flag\_menu = false;

menu = true;

break;

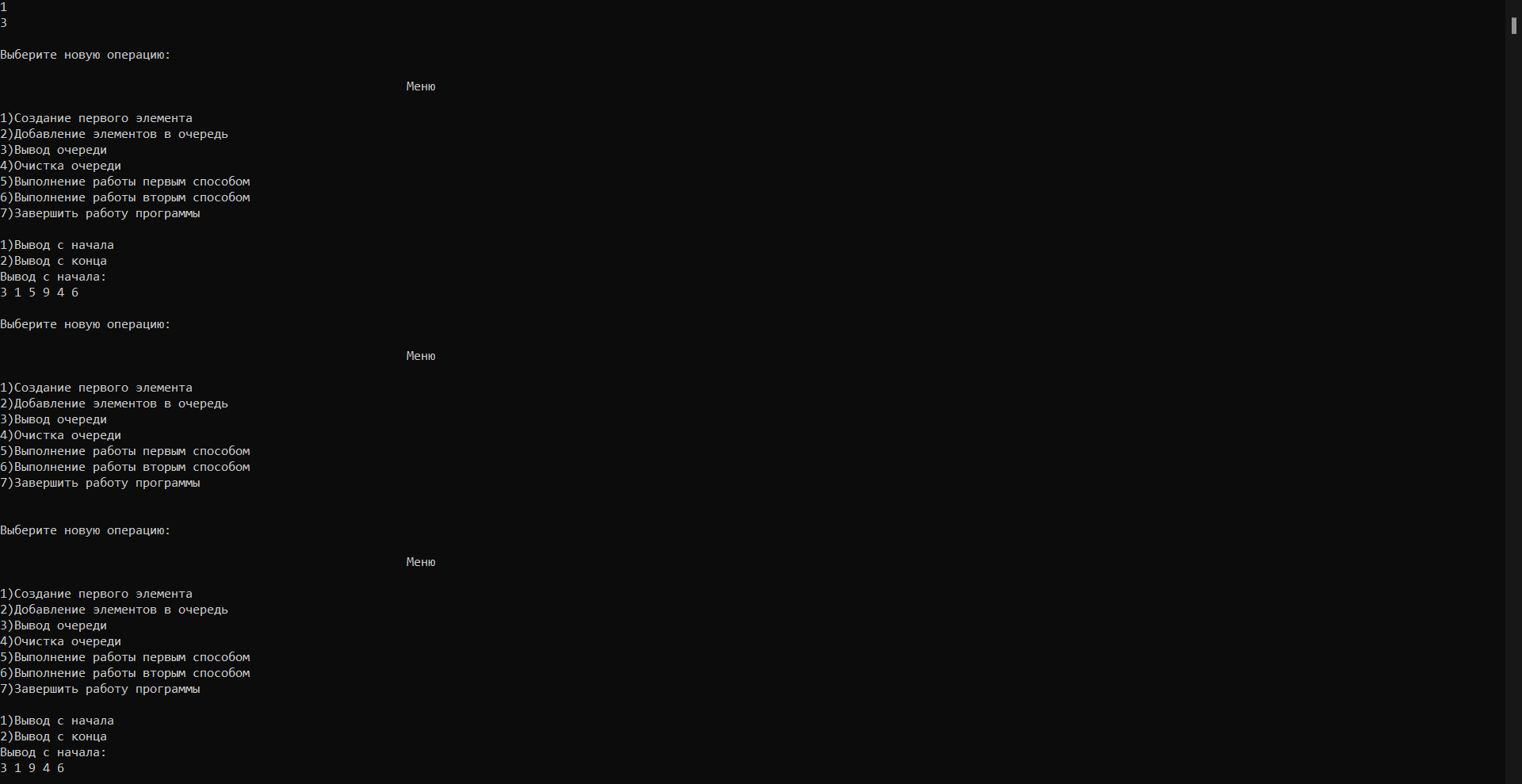
}

}

} while (flag\_menu == false);

}

}



Вывод: изучил возможности работы со списками, организованными в виде очереди.