Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 8

По дисциплине «Основы алгоритмизации и программирования»

На тему « **Полустатические структуры данных: очереди**»

Выполнил:

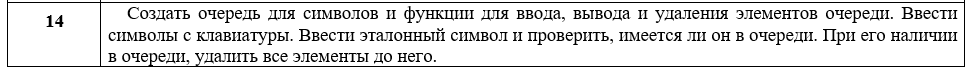
Федорович Вадим

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

Основа:



#include<iostream>

using namespace std;

struct Number

{

char symbol;

Number\* next;

};

bool flag = false;

void create(Number\*\* begin, Number\*\* end, char p); //формирование элементов очереди

void view(Number\* begin); //вывод элементов очереди

Number\* found\_element(Number\* begin); //определение минимального элемента

void DeltoElement(Number\*\* begin, Number\*\* p); //удаление до минимального элемента

int main()

{

setlocale(0, "");

Number\* begin = NULL, \* end, \* t;

t = new Number;

char p;int size;

cout << "\nEnter size queue="; cin >> size;

cout << "Enter number= "; cin >> p;

t->symbol = p; //первый элемент

t->next = NULL;

begin = end = t;

for (int i = 1; i < size; i++) //создание очереди

{

cout << "Enter number= "; cin >> p;

create(&begin, &end, p);

}

cout << "\nelements of queue: \n";

if (begin == NULL) //вывод на экран

cout << "No elements" << endl;

else

view(begin);

t = found\_element(begin); //определение минимального

if (flag == true) {

cout << "Символ найден" << endl;

DeltoElement(&begin, &t);

}

cout << "\nnew Queue:\n";

view(begin);

return 0;

}

void create(Number\*\* begin, Number\*\* end, char p) //Формирование элементов очереди

{

Number\* t = new Number;

t->next = NULL;

if (\*begin == NULL)

\*begin = \*end = t;

else

{

t->symbol = p;

(\*end)->next = t;

\*end = t;

}

}

void view(Number\* begin) //Вывод элементов очереди

{

Number\* t = begin;

if (t == NULL)

{

cout << "Number is empty\n";

return;

}

else

while (t != NULL)

{

cout << t->symbol << endl;

t = t->next;

}

}

Number\* found\_element(Number\* begin) //Определение минимального элемента

{

Number\* t = begin, \* try\_t = nullptr;

char enter\_symbol = ' ';

cout << "Введите желаемый символ: ";

cin >> enter\_symbol;

if (t == NULL)

{

cout << "Number is empty\n"; return 0;

}

else

{

while (t != NULL)

{

if (t->symbol == enter\_symbol)

{

try\_t = t;

flag = true;

break;

}

t = t->next;

}

}

return try\_t;

}

void DeltoElement(Number\*\* begin, Number\*\* p) //Удаление до минимального элемента

{

Number\* t;

t = new Number;

while (\*begin != \*p)

{

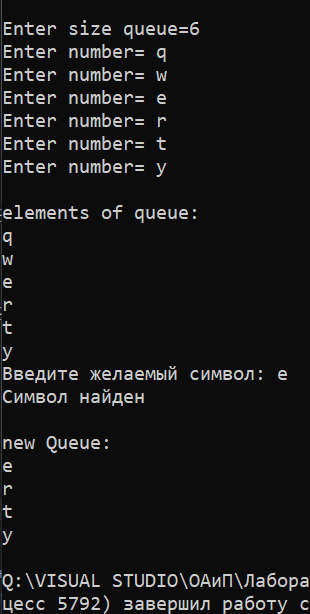
t = \*begin;

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

delete t;

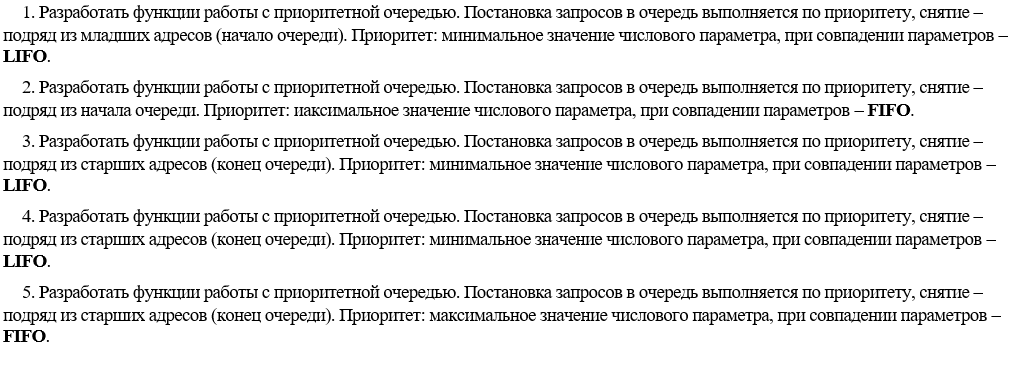
}

}



Допы:





#include<iostream>

using namespace std;

struct Item

{

int data;

Item\* next;

};

Item\* head, \* tail;

bool isNull(void) //Проверка на пустоту

{

return (head == nullptr);

}

void deletFirst() //Извлечение элемента из начала

{

if (isNull())

cout << "Очередь пуста" << endl;

else

{

Item\* p = head;

head = head->next;

delete p;

}

}

void deletlast() //Извлечение элемента из начала

{

if (isNull())

cout << "Очередь пуста" << endl;

else

{

Item\* p;

tail->data = NULL;

}

}

void getFromHead() //Получение элемента из начала

{

if (isNull())

cout << "Очередь пуста" << endl;

else

cout << "Начало = " << head->data << endl;

}

void getFromTail() //Получение элемента из конца

{

if (isNull())

cout << "Очередь пуста" << endl;

else

cout << "Конец = " << tail->data << endl;

}

void insertToQueue(int x) //Добавление элемента в очередь

{

Item\* p = new Item; //новый указатель

p->data = x;

p->next = nullptr;

Item\* v = new Item; //указатель для нового числа

Item\* p1 = new Item;

Item\* p2 = new Item;

int i = 0; //флажок

if (isNull())

head = tail = p;

else

{

p2 = head; p1 = head; // !!!Чтобы реализовать ввод LiFo необходимо чтобы указатель p2 был позади, FiFo наобарот

while (p1 != nullptr) //пока очередь не закончится

{

if (i == 1) //ТАКЖЕ НЕОБХОДИМЫ ДЛЯ РЕАЛИЗАЦИИ LiFo

{

if (i == 1)

{

if (x > p1->data) //число больше, чем в очереди

{

v->data = x;

v->next = p1;

p2->next = v;

return;

}

if (x == p1->data)

{

v->data = x;

v->next = p1->next;

p1->next = v;

return;

}

p2 = p2->next;

}

if (x > p1->data)

{

v->data = x;

v->next = p1;

head = v;

return;

}

if (p1->next == nullptr)

{

v->data = x;

v->next = nullptr;

tail->next = v;

tail = v;

return;

}

i = 1;

p1 = p1->next; // следующее число

}

else // else необходимое для реализации вставки LiFo

{

if (x >= p1->data)

{

v->data = x;

v->next = p2;

head = v;

return;

}

}

p1 = p1->next; //ТАКЖЕ НЕОБХОДИМЫ ДЛЯ РЕАЛИЗАЦИИ LiFo

i = 1;

}

if (p1 == NULL) // Выражение ДЛЯ РЕАЛИЗЦИИ LiFo

{

v->data = x;

v->next = tail;

p2->next = v;

}

}

}

void printQueue() //Вывод очереди

{

int g;

Item\* p = new Item;

if (isNull())

cout << "Очередь пуста" << endl;

else

{

cout << "Очередь = ";

p = head;

while (!isNull())

{

if (p != nullptr)

{

g = p->data;

if (g == 0) {

}

else { cout << p->data << " "; cout << "->"; }

p = p->next;

}

else

{

cout << "NULL" << endl;

return;

}

}

}

}

void clrQueue() //Очистка очереди

{

while (!isNull()) deletFirst();

}

int main()

{

setlocale(0,"");

int i = 1, choice = 1, z; head = nullptr; tail = nullptr;

while (choice != 0)

{

cout << "1 - добавить элемент" << endl;

cout << "2 - получить элемент с начала" << endl;

cout << "3 - извлечь элемент с начала" << endl;

cout << "4 - вывести элементы" << endl;

cout << "5 - очистить очередь" << endl;

cout << "6 - получить элемент с конца" << endl;

cout << "0 - выход" << endl;

cout << "Выберите действие "; cin >> choice;

switch (choice)

{

case 1: cout << "Введите элемент: "; cin >> z;

insertToQueue(z); printQueue(); break;

case 2: getFromHead(); break;

case 3: deletFirst(); break;

case 4: printQueue(); break;

case 5: clrQueue(); break;

case 6: getFromTail(); break;

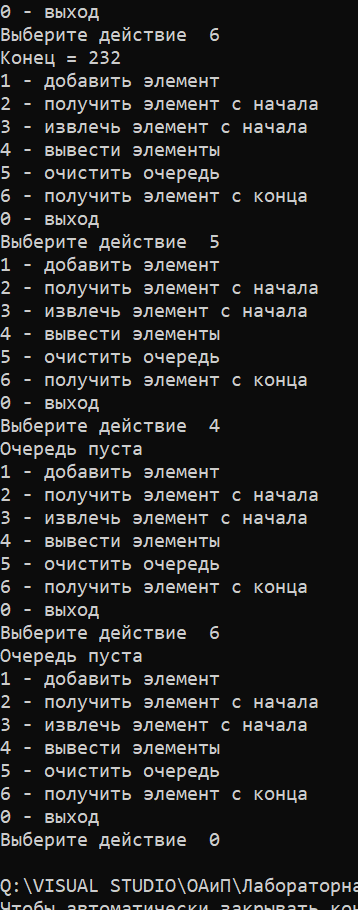
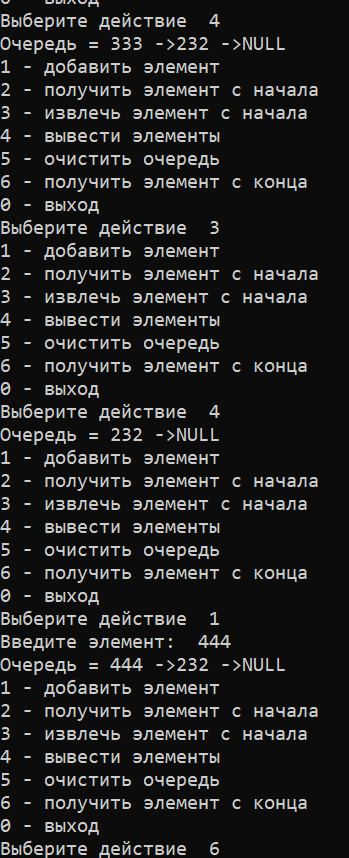
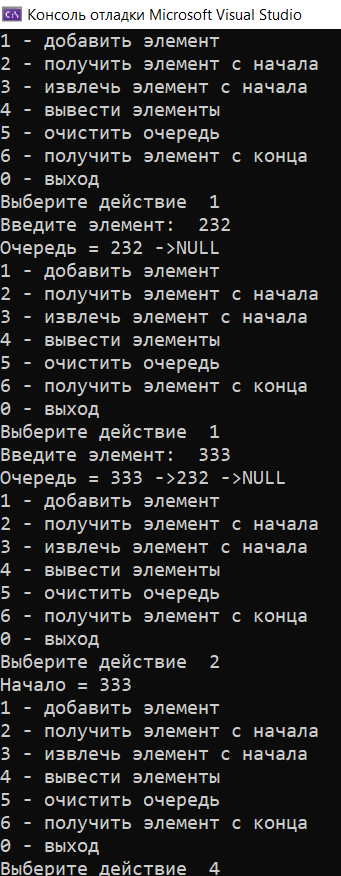
case 7: deletlast(); break;

}

}

return 0;

}





#include<iostream>

using namespace std;

struct Number

{

int symbol;

Number\* next;

};

bool flag = false;

void create(Number\*\* begin, Number\*\* end, int p); //формирование элементов очереди

void view(Number\* begin); //вывод элементов очереди

Number\* found\_element(Number\* begin); //определение минимального элемента

void DeltoElement(Number\*\* begin, Number\*\* p, Number\*\* end); //удаление до минимального элемента

int main()

{

setlocale(0, "");

Number\* begin = NULL, \* end, \* t;

t = new Number;

int p; int size;

cout << "Enter size queue = "; cin >> size;

cout << "Enter number = "; cin >> p;

t->symbol = p; //первый элемент

t->next = NULL;

begin = end = t;

for (int i = 1; i < size; i++) //создание очереди

{

cout << "Enter number = "; cin >> p;

create(&begin, &end, p);

}

cout << "\nelements of queue: \n";

if (begin == NULL) //вывод на экран

cout << "No elements" << endl;

else

view(begin);

t = found\_element(begin); //определение максимального

DeltoElement(&begin, &t, &end);

cout << "\nnew Queue:\n";

view(begin);

return 0;

}

void create(Number\*\* begin, Number\*\* end, int p) //Формирование элементов очереди

{

Number\* t = new Number;

t->next = NULL;

if (\*begin == NULL)

\*begin = \*end = t;

else

{

t->symbol = p;

(\*end)->next = t;

\*end = t;

}

}

void view(Number\* begin) //Вывод элементов очереди

{

Number\* t = begin;

if (t == NULL)

{

cout << "Number is empty\n";

return;

}

else

while (t != NULL)

{

cout << t->symbol << endl;

t = t->next;

}

}

Number\* found\_element(Number\* begin) //Определение минимального элемента

{

Number\* t = begin, \* max = nullptr;

int max\_digit;

if (t == NULL)

{

cout << "Number is empty\n"; return 0;

}

else

{

max\_digit = t->symbol;

while (t != NULL)

{

if (t->symbol > max\_digit)

{

max = t;

max\_digit = t->symbol;

}

t = t->next;

}

}

return max;

}

void DeltoElement(Number\*\* begin, Number\*\* p, Number\*\* end) //Сдвиг до макимального элемента

{

while (\*begin != \*p) {

Number\* t = new Number;

t->next = NULL;

t->symbol = (\*begin)->symbol;

(\*end)->next = t;// добавление в конец

\*end = t;

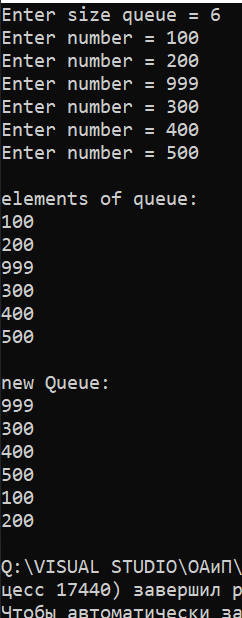
t = \*begin;//удаление

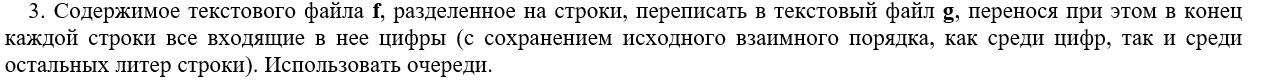
\*begin = (\*begin)->next;

delete t;

}

}





#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node

{

char data;

Node\* next;

};

struct Queue

{

Node\* head;

Node\* tail;

};

void CreateQueue(Queue\* q)

{

q->head = NULL;

q->tail = NULL;

}

void Enqueue(Queue\* q, char data)

{

Node\* newNode = new Node();

newNode->data = data;

newNode->next = NULL;

if (q->tail == NULL)

{

q->head = newNode;

q->tail = newNode;

}

else

{

q->tail->next = newNode;

q->tail = newNode;

}

}

void Symbol(Queue\* Letter, Queue\* Number)

{

ifstream f("Q:/F.txt");

ofstream g("Q:/G.txt");

string line;

char symbol;

getline(f, line);

cout << "Исходная строка: " << line << endl;

for (int i = 0; i < line.length(); i++) {

symbol = line[i];

if ((symbol >= 97 && symbol <= 122) || (symbol >= 65 && symbol <= 90)) {

Enqueue(Letter, symbol);

}

if (symbol >= 48 && symbol <= 57) {

Enqueue(Number, symbol);

}

}

Node\* line1 = Letter->head;

Node\* line2 = Number->head;

cout << "Буквы: " << endl;

while (line1) {

cout << line1->data;

g << line1->data;

line1 = line1->next;

}

cout << endl;

cout << "Цифры: " << endl;

while (line2) {

cout << line2->data;

g << line2->data;

line2 = line2->next;

}

cout << endl;

f.close();

g.close();

}

int main()

{

setlocale(LC\_ALL, "RU");

Queue Letter;

Queue Number;

CreateQueue(&Letter);

CreateQueue(&Number);

Symbol(&Letter, &Number);

return 0;

}

