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:

Рауба Арсений

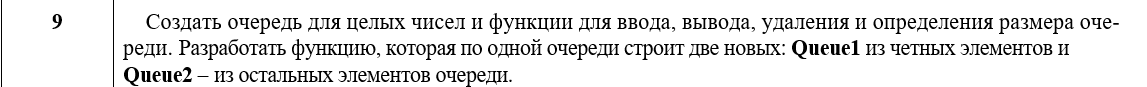
Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

4. Создать проект, демонстрирующий работу с ***очередью***. В соответствии со своим вариантом выполнить задание из таблицы, представленной ниже. Разработать меню и реализовать все операции с очередью через функции. Максимальный размер очереди ввести с клавиатуры.

Вариант №9



#include <iostream>

using namespace std;

struct queue {

int elements;

queue\* next;

};

struct queue1 {

int elements1;

queue1\* next;

};

struct queue2 {

int elements2;

queue2\* next;

};

void input(queue\*\* begin, queue\*\* end, int element);

void input1(queue1\*\* begin1, queue1\*\* end1, int element);

void input2(queue2\*\* begin2, queue2\*\* end2, int element);

void output(queue\* begin);

void output1(queue1\* begin1);

void output2(queue2\* begin2);

void del(queue\*\* begin, queue\*\* end);

void newQueue(queue\*\* t, int element, int size);

int main() {

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

t = new queue;

int element, size;

cout << "Enter size of queue: ";

cin >> size;

cout << "Enter a element: ";

cin >> element;

t->elements = element;

t->next = NULL;

begin = end = t;

for (int i = 1; i < size; i++) {

cout << "Enter a element: ";

cin >> element;

input(&begin, &end, element);

}

cout << "Elements of queue: " << endl;

if (begin == NULL) {

cout << "Queue is empty" << endl;

}

else {

output(begin);

}

newQueue(&t, element, size);

cout << "Delete: " << t->elements;

del(&begin, &end);

}

void input(queue\*\* begin, queue\*\* end, int element) {

queue\* t = new queue;

t->next = NULL;

if (\*begin == NULL) {

\*begin = \*end = t;

}

else {

t->elements = element;

(\*end)->next = t;

\*end = t;

}

}

void input1(queue1\*\* begin1, queue1\*\* end1, int element) {

queue1\* t1 = new queue1;

t1->next = NULL;

t1->elements1 = element;

if (\*begin1 == NULL) {

\*begin1 = \*end1 = t1;

}

else {

(\*end1)->next = t1;

\*end1 = t1;

}

}

void input2(queue2\*\* begin2, queue2\*\* end2, int element) {

queue2\* t2= new queue2;

t2->next = NULL;

t2->elements2 = element;

if (\*begin2 == NULL) {

\*begin2 = \*end2 = t2;

}

else {

(\*end2)->next = t2;

\*end2 = t2;

}

}

void output(queue\* begin) {

queue\* t = begin;

if (t == NULL) {

cout << "Queue is empty" << endl;

return;

}

else {

while (t != NULL) {

cout << t->elements << endl;

t = t->next;

}

}

}

void output1(queue1\* begin1) {

queue1\* t1 = begin1;

if (t1 == NULL) {

cout << "Queue is empty" << endl;

return;

}

else {

while (t1 != NULL) {

cout << t1->elements1 << endl;

t1 = t1->next;

}

}

}

void output2(queue2\* begin2) {

queue2\* t2 = begin2;

if (t2 == NULL) {

cout << "Queue is empty" << endl;

return;

}

else {

while (t2 != NULL) {

cout << t2->elements2 << endl;

t2 = t2->next;

}

}

}

void del(queue\*\* begin, queue\*\* end) {

queue\* t = \*begin;

if (t == NULL) {

cout << "Queue is empty." << endl;

return;

}

else

{

\*begin = (\*begin)->next; //начало очереди переназначается на след. элемент//

delete t; //удаление//

}

if (t == NULL) { //если после удаления начало очереди - NULL//

\*end = NULL; //конец очереди обнуляется//

}

}

void newQueue(queue\*\*t,int element,int size) {

queue1\* begin1 = NULL, \* end1, \* t1;

t1 = new queue1;

queue2\* begin2 = NULL, \* end2, \* t2;

t2 = new queue2;

while (\*t != NULL) {

if ((\*t)->elements % 2 == 0) {

input1(&begin1, &end1, (\*t)->elements);

}

else {

input2(&begin2, &end2,(\*t)->elements);

}

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

}

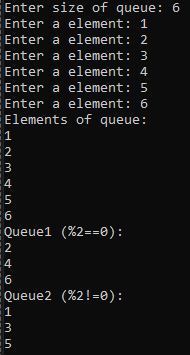
cout << "Queue1 (%2==0):" << endl;

output1( begin1);

cout << "Queue2 (%2!=0):" << endl;

output2(begin2);

}



5. Дополнительное задание.

1. В [приложении 1](#prilogenie1) приведен проект, в котором реализована очередь на основе односвязного *списка* с *приоритетным* включением. На основе данного проекта разработать функции, которые предлагается создать в данном приложении.

#include<iostream>

using namespace std;

struct Item

{

int data;

Item\* next;

};

Item\* head, \* tail;

int main()

{

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

while (choice != 7)

{

cout << "1. Add element." << endl;

cout << "2. Get element from beginning." << endl;

cout << "3. Extract element from beginning." << endl;

cout << "4. Display elements." << endl;

cout << "5. Clear queue." << endl;

cout << "6. Get element from end." << endl;

cout << "7. Exit." << endl;

cin >> choice;

switch (choice)

{

case 1: cout << "Enter element: "; 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;

}

bool isNull(void) //Проверка на пустоту

{

return (head == nullptr);

}

void deletFirst() //Извлечение элемента из начала

{

if (isNull())

cout << "Queue is empty." << endl;

else

{

Item\* p = head;

head = head->next;

delete p;

}

}

void deletlast() //Извлечение элемента из начала

{

if (isNull())

cout << "Queue is empty." << endl;

else

{

Item\* p;

tail->data = NULL;

}

}

void getFromHead() //Получение элемента из начала

{

if (isNull())

cout << "Queue is empty." << endl;

else

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

}

void getFromTail() //Получение элемента из конца

{

if (isNull())

cout << "Queue is empty." << endl;

else

cout << "End = " << 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;

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

{

if (i == 1)

{

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;

i = 1;

}

if (p1 == NULL)

{

v->data = x;

v->next = tail;

p2->next = v;

}

}

}

void printQueue() //Вывод очереди

{

int g;

Item\* p = new Item;

if (isNull())

cout << "Queue is empty." << endl;

else

{

cout << "Queue: ";

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;

}

}

}

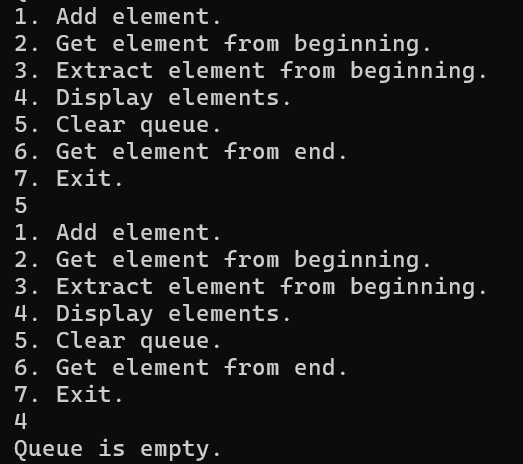
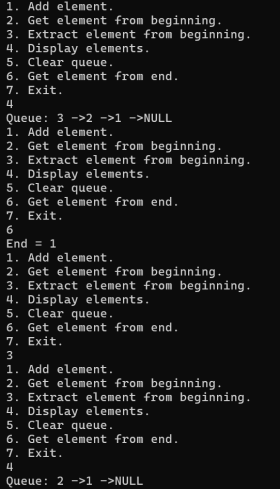
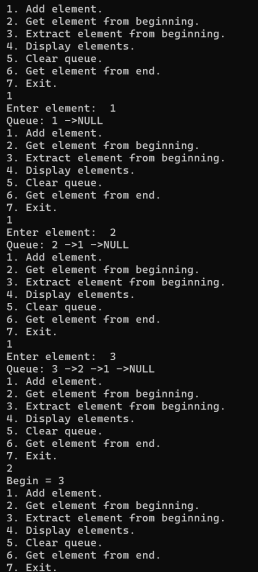
}

void clrQueue() //Очистка очереди

{

while (!isNull()) deletFirst();

}



2. Создать очередь с вещественными числами, и заполнить ее с клавиатуры. Выполнить циклический сдвиг элементов в очереди так, чтобы в ее начале был расположен наибольший элемент.

#include <iostream>

using namespace std;

struct Node {

float Number;

Node\* next;

};

struct Queue {

int size = 0;

Node\* head;

Node\* tail;

};

int main()

{

Queue t;

int choice;

CreateQueue(&t);

for (;;)

{

cout << "Menu" << endl;

cout << "1. Add element." << endl;

cout << "2. Output queue." << endl;

cout << "3. Cyclic shift." << endl;

cout << "4. Exit." << endl;

cin >> choice;

switch (choice)

{

case 1:

AddElem(&t);

break;

case 2:

OutQueue(&t);

break;

case 3:

Shift(&t);

break;

case 4:

return 0;

break;

}

}

}

void CreateQueue(Queue\* Q) {

Q->head = new Node();

Q->head->next = NULL;

Q->tail = Q->head;

Q->size = 0;

}

void AddElem(Queue\* Q) {

float newElem;

cout << "Enter element: ";

cin >> newElem;

Q->tail->next = new Node();

Q->tail = Q->tail->next;

Q->tail->Number = newElem;

Q->tail->next = NULL;

Q->size++;

}

int OutQueue(Queue\* Q) {

if (Q->head == Q->tail) {

cout << "Queue is empty.";

return 0;

}

else {

Node\* Temp = Q->head->next;

while (Temp) {

cout << Temp->Number << endl;

Temp = Temp->next;

}

}

}

int Shift(Queue\* Q) {

if (Q->head == Q->tail) {

cout << "Queue is empty." << endl;

return 0;

}

else {

float max = 0;

Node\* Temp = Q->head->next;

while (Temp) {

if (Temp->Number > max)

max = Temp->Number;

Temp = Temp->next;

}

cout << "Max value: " << max << endl;

cout << "Queue with shift: ";

Temp = Q->head;

Temp->Number = max;

cout << Temp->Number << endl;

while (Temp) {

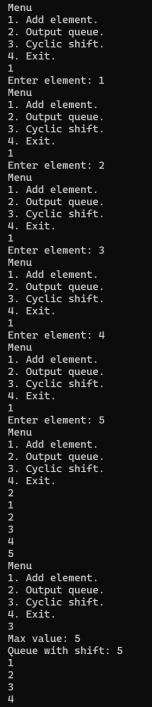
if (Temp->Number != max) cout << Temp->Number << endl;

Temp = Temp->next;

}

}

}



3. Содержимое текстового файла **f**, разделенное на строки, переписать в текстовый файл **g**, перенося при этом в конец каждой строки все входящие в нее цифры (с сохранением исходного взаимного порядка, как среди цифр, так и среди остальных литер строки). Использовать очереди.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node

{

char data;

Node\* next;

};

struct Queue

{

Node\* head;

Node\* tail;

};

int main()

{

Queue Letter;

Queue Number;

CreateQueue(&Letter);

CreateQueue(&Number);

Symbol(&Letter, &Number);

return 0;

}

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("First.txt");

ofstream g("Second.txt");

string line;

char symbol;

getline(f, line);

cout << "Stroke: " << 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 << "Letters: " << endl;

while (line1) {

cout << line1->data;

g << line1->data;

line1 = line1->next;

}

cout << endl;

cout << "Numbers: " << endl;

while (line2) {

cout << line2->data;

g << line2->data;

line2 = line2->next;

}

cout << endl;

f.close();

g.close();

}

