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»

 Выполнила:

Студентка1 курса 6 группы

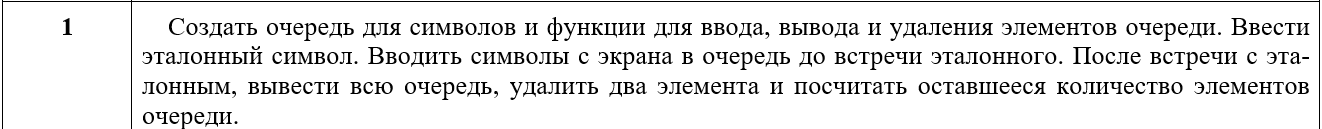
Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 1

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.



#include <iostream>

using namespace std;

// Определение структуры для очереди

struct Queue {

int front, rear, size;

char\* arr;

// Конструктор для инициализации очереди

Queue(int capacity) {

size = capacity;

front = rear = -1;

arr = new char[size];

}

// Проверка, пуста ли очередь

bool is\_empty() {

return front == -1;

}

// Проверка, полна ли очередь

bool is\_full() {

return (rear + 1) % size == front;

}

// Добавление элемента в очередь

void enqueue(char item) {

if (is\_full()) {

cout << "Очередь полна. Невозможно добавить элемент." << endl;

return;

}

rear = (rear + 1) % size;

arr[rear] = item;

if (front == -1) {

front = rear;

}

}

// Удаление элемента из очереди

void dequeue() {

if (is\_empty()) {

cout << "Очередь пуста. Невозможно удалить элемент." << endl;

return;

}

if (front == rear) {

front = rear = -1;

}

else {

front = (front + 1) % size;

}

}

// Вывод содержимого очереди

void display() {

if (is\_empty()) {

cout << "Очередь пуста." << endl;

return;

}

cout << "Содержимое очереди:" << endl;

int i = front;

while (i != rear) {

cout << arr[i] << " "; // Вывод элементов очереди

i = (i + 1) % size;

}

cout << arr[rear] << endl; // Вывод последнего элемента

}

};

int main() {

setlocale(LC\_ALL, "Russian");

int max\_size;

cout << "Введите максимальный размер очереди: ";

cin >> max\_size;

Queue charQueue(max\_size);

char reference\_char;

cout << "Введите эталонный символ: ";

cin >> reference\_char;

cout << "Введите символы (остановитесь, когда встретите эталонный символ): ";

while (true) {

char input\_char;

cin >> input\_char;

if (input\_char == reference\_char) {

break;

}

charQueue.enqueue(input\_char); // Добавление символа в очередь

}

charQueue.display(); // Вывод содержимого очереди

charQueue.dequeue(); // Удаление первого элемента

charQueue.dequeue(); // Удаление второго элемента

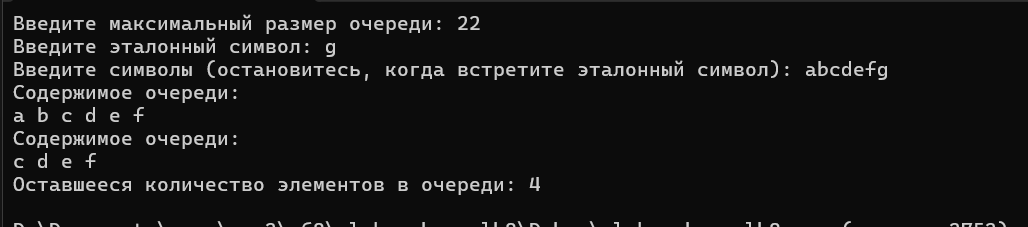
charQueue.display(); // Вывод содержимого очереди

int remaining\_elements = (charQueue.rear - charQueue.front + charQueue.size) % charQueue.size + 1;

cout << "Оставшееся количество элементов в очереди: " << remaining\_elements << endl;

return 0;

}



Дополнительные задания

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;

bool isNull(void) //Проверка на пустоту

{

return (head == NULL);

}

void deletFirst() //Извлечение элемента из начала

{

if (isNull())

cout << "Очередь пуста" << endl;

else

{

Item\* p = head;

head = head->next;

delete p;

}

}

void getFromHead() //Получение элемента из начала

{

if (isNull())

cout << "Очередь пуста" << endl;

else

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

}

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

{

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

p->data = x;

p->next = NULL;

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 != NULL) //пока очередь не закончится

{

if (i == 1)

{

if (x <= p1->data) //число меньше, чем в очереди

{

v->data = x;

v->next = p1;

p2->next = v;

return;

}

p2 = p2->next; // следующее число

}

else

{

if (x <= p1->data)

{

v->data = x;

v->next = p1;

head = v;

return;

}

}

p1 = p1->next;

i = 1;

}

if (p1 == NULL)

{

tail->next = p;

tail = p;

}

}

}

void printQueue() //Вывод очереди

{

Item\* p = new Item;

if (isNull())

cout << "Очередь пуста" << endl;

else

{

cout << "Очередь = ";

p = head;

while (!isNull())

{

if (p != NULL)

{

cout << p->data << " "; cout << "->";

p = p->next;

}

else

{

cout << "NULL" << endl;

return;

}

}

}

}

void clrQueue() //Очистка очереди

{

while (!isNull()) deletFirst();

}

int main()

{

setlocale(LC\_CTYPE, "Russian");

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

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 << "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;

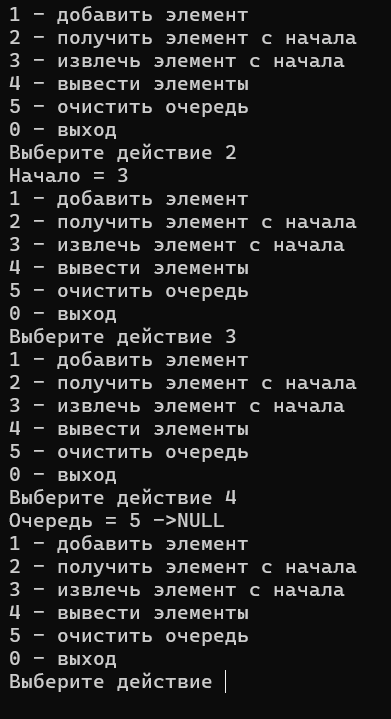
case 4: printQueue(); break;

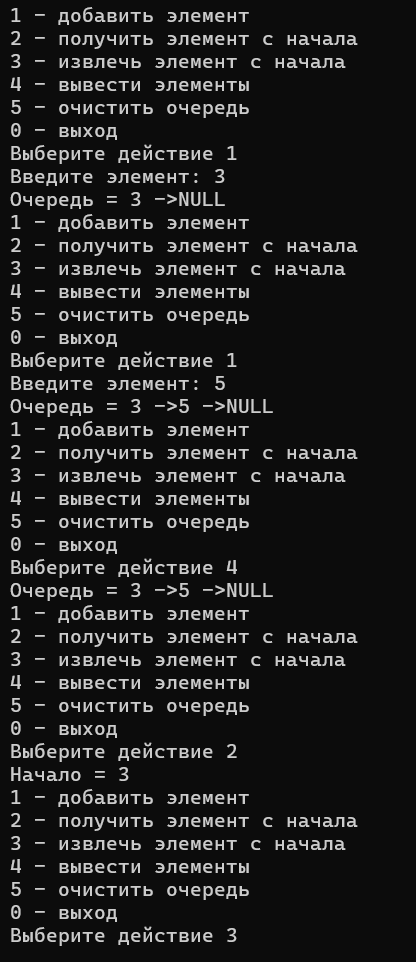
case 5: clrQueue(); break;

}

}

return 0;

}



Задание 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>//библиотеки

using namespace std;

struct QUEUE {//очередь

float value;

QUEUE\* next;

};

//прототипы функций

void input\_elements(QUEUE\*\* begin, QUEUE\*\* end, float element);

void print\_elements(QUEUE\* begin);

void sort\_queue(QUEUE\*\* begin);

int main()//главная функция

{

setlocale(LC\_ALL, "Russian");

QUEUE\* begin = nullptr;

QUEUE\* end, \* current = new QUEUE;

int queue\_size;

cout << "Введите размер очереди: ";

cin >> queue\_size;

float element;

cout << "Введите первый элемент очереди: ";

cin >> element;

current->value = element;

current->next = nullptr;

begin = end = current;

char choise;

do {

cout << "1-Вставить элементы" << endl;

cout << "2-Печать элементов" << endl;

cout << "3-Сортировка" << endl;

cout << "4-Очистить консоль" << endl;

cout << "5-Выход" << endl;

cin >> choise;

//проверка ввода

while (!(choise >= '1' && choise <= '5')) {

cout << "Неверный ввод! Попробуйте снова: ";

cin >> choise;

}

int choise\_sw = choise - '0';//преобразование char в int

switch (choise\_sw) {

case 1:

for (int i = 0; i < queue\_size - 1; i++) {

cout << "Введите элемент: ";

cin >> element;

input\_elements(&begin, &end, element);

}

break;

case 2:print\_elements(begin); break;

case 3:sort\_queue(&begin); break;

case 4:system("cls"); break;

}

} while (choise != '5');

}

void input\_elements(QUEUE\*\* begin, QUEUE\*\* end, float element)//функция для ввода элементов

{

QUEUE\* temp = new QUEUE;

temp->next = nullptr;

if (\*begin == nullptr) {

\*begin = \*end = temp;

}

else {

temp->value = element;

(\*end)->next = temp;

\*end = temp;

}

}

void print\_elements(QUEUE\* begin)//функция для печати всей очереди

{

QUEUE\* current = begin;//указатель, указывающий на начало очереди

if (current == nullptr) {//если очередь пуста

cout << "Очередь пуста" << endl;

return;

}

while (current != nullptr) {

cout << current->value << ' ';

current = current->next;//перемещение указателя

}

cout << endl;

}

void sort\_queue(QUEUE\*\* begin)//функция для сортировки очереди

{

if (\*begin == nullptr) {

cout << "Нечего сортировать. Эта очередь пуста" << endl;

return;

}

QUEUE\* i, \* j;//указатели для взаимодействия с оригинальной очередью

float temp;

for (i = \*begin; i != nullptr; i = i->next) {//сортировка пузырьком

for (j = \*begin; j != nullptr; j = j->next) {

if (i->value > j->value) {

temp = i->value;//меняем местами

i->value = j->value;

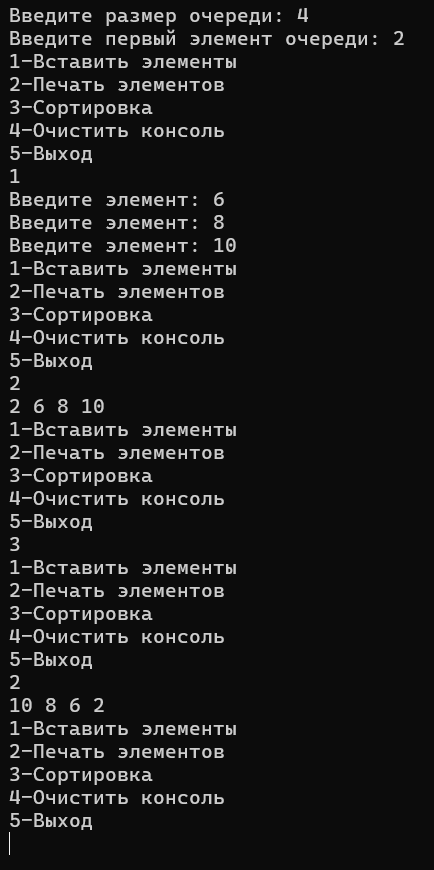
j->value = temp;

}

}

}

}



Задание 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>

#include <queue>

using namespace std;

void processFile(const string& inputFileName, const string& outputFileName) {

    ifstream inputFile(inputFileName);

    ofstream outputFile(outputFileName);

    if (!inputFile.is\_open() || !outputFile.is\_open()) {

        cerr << "Ошибка открытия файлов." << endl;

        return;

    }

    string line;

    while (getline(inputFile, line)) {

        queue<char> charQueue;

        queue<char> digitQueue;

        for (char c : line) {

            if (isdigit(c)) {

                digitQueue.push(c);

            }

            else {

                charQueue.push(c);

            }

        }

        while (!charQueue.empty()) {

            outputFile << charQueue.front();

            charQueue.pop();

        }

        while (!digitQueue.empty()) {

            outputFile << digitQueue.front();

            digitQueue.pop();

        }

        outputFile << endl;

    }

    inputFile.close();

    outputFile.close();

}

int main() {

    setlocale(LC\_ALL, "ru");

    processFile("f.txt", "g.txt");

    return 0;

}

f.txt

A screenshot of a computer

Description automatically generated

g.txt

A screenshot of a computer

Description automatically generated