Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

по дисциплине «Логика и основы алгоритмизации в инженерных задачах»

по лабораторной работе №3

«Динамические списки»

Выполнили:

студенты группы 22ВВС1

Эрмедов А.Э.

Казаров И.И.

Приняли:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Название**

Оценка времени выполнения программ

**Практическая часть**

Задание:

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализуйте структуру данных *Очередь*.
3. \* На основе приведенного кода реализуйте структуру данных *Стек*.

**Листинг loaviz31.cpp**

#include <iostream>

#include <string>

#include <locale.h>

#include <Windows.h>

using namespace std;

struct Node {

string data;

int priority;

Node\* next;

};

class PriorityQueue {

private:

Node\* head;

public:

PriorityQueue() {

head = nullptr;

}

// Метод для добавления элемента в приоритетную очередь

void enqueue(const string& data, int priority) {

Node\* newNode = new Node; // Создаем новый узел

newNode->data = data; // Присваиваем ему значение данных

newNode->priority = priority; // Присваиваем ему значение приоритета

if (head == nullptr || priority > head->priority) {

newNode->next = head; // Если очередь пуста или приоритет нового элемента выше,

head = newNode; // то новый элемент становится головой очереди

}

else {

Node\* current = head;

while (current->next != nullptr && priority <= current->next->priority) {

current = current->next;

}

newNode->next = current->next; // Вставляем новый элемент в середину очереди

current->next = newNode;

}

}

// Метод для удаления элемента из приоритетной очереди и возвращения его значения

string dequeue() {

if (head == nullptr) {

cout << "Очередь пуста" << endl; // Если очередь пуста, выводим сообщение об этом

return ""; // и возвращаем пустую строку

}

string data = head->data; // Сохраняем значение данных первого элемента

Node\* temp = head;

head = head->next; // Перемещаем указатель на следующий элемент

delete temp; // Удаляем первый элемент

return data; // Возвращаем сохраненное значение данных первого элемента

}

// Метод для вывода содержимого очереди на экран

void printQueue() {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " "; // Выводим значение данных текущего элемента

current = current->next; // Перемещаем указатель на следующий элемент

}

cout << endl;

}

};

int main() {

setlocale(LC\_ALL, "ru\_RU.UTF-8");

PriorityQueue pq;

string data;

int priority;

char choice;

while (true) {

cout << "Введите данные: ";

cin >> data;

cout << "Введите приоритет: ";

cin >> priority;

pq.enqueue(data, priority);

cout << "Вы хотите добавить элемент? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

break;

}

}

cout << "Содержимое приоритетной очереди: ";

pq.printQueue();

while (true) {

cout << "Хотите исключить элемент? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

string dequeuedData = pq.dequeue();

cout << "Исключенный элемент: " << dequeuedData << endl;

}

else {

break;

}

cout << "Хотите добавить элемент? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

break;

}

cout << "Введите данные: ";

cin >> data;

cout << "Введите приоритет: ";

cin >> priority;

pq.enqueue(data, priority);

}

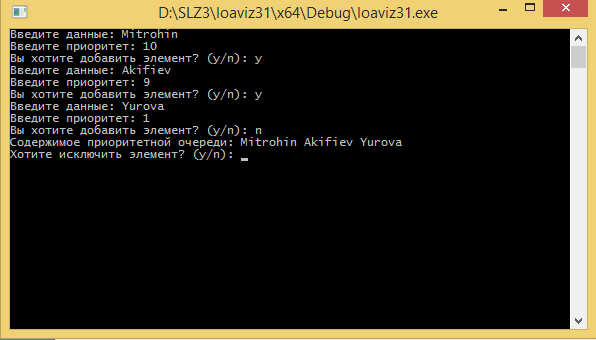
cout << "Содержимое приоритетной очереди после операций: ";

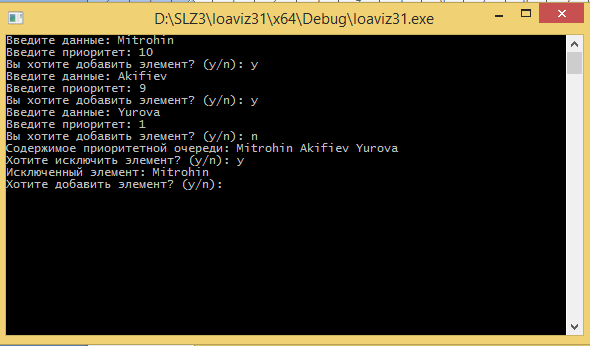
pq.printQueue();

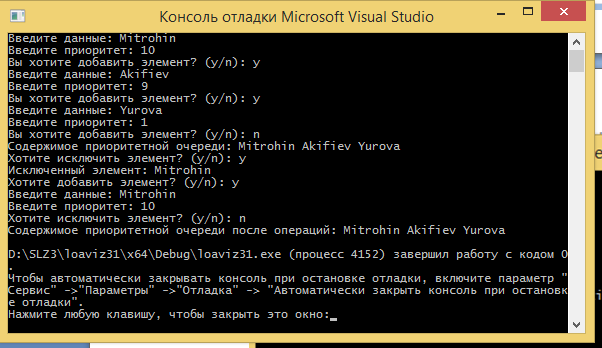
return 0;

}

**Результат**

****

****

****

**Листинг loaviz31ex1.cpp**

#include <iostream>

#include <string>

#include <locale.h>

using namespace std;

struct Node {

string data;

Node\* next;

};

class Queue {

private:

Node\* head;

Node\* tail;

public:

Queue() {

head = nullptr;

tail = nullptr;

}

void enqueue(const string& data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

}

string dequeue() {

if (head == nullptr) {

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

return "";

}

string data = head->data;

Node\* temp = head;

if (head == tail) {

head = nullptr;

tail = nullptr;

}

else {

head = head->next;

}

delete temp;

return data;

}

void printQueue() {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

};

int main() {

setlocale(LC\_ALL, "ru\_RU.UTF-8");

Queue queue;

string data;

char choice;

while (true) {

cout << "Enter data: ";

cin >> data;

queue.enqueue(data);

cout << "You want add element? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

break;

}

}

cout << "Queue content: ";

queue.printQueue();

while (true) {

cout << "You want delete element? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

string dequeuedData = queue.dequeue();

cout << "Deleted element: " << dequeuedData << endl;

}

else {

break;

}

cout << "You want add element? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

break;

}

cout << "Enter data: ";

cin >> data;

queue.enqueue(data);

}

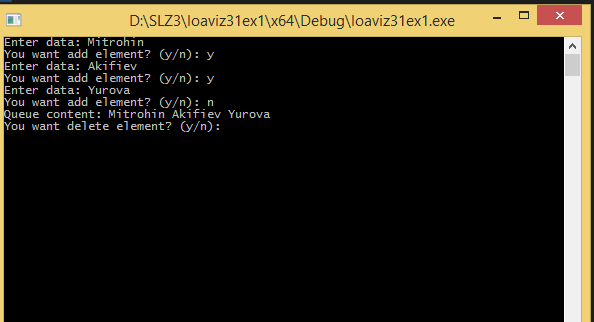
cout << "Queue content after operations: ";

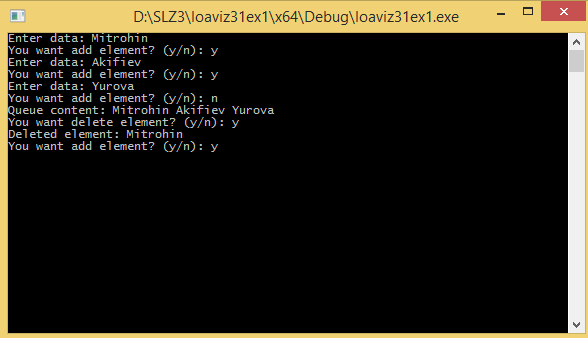
queue.printQueue();

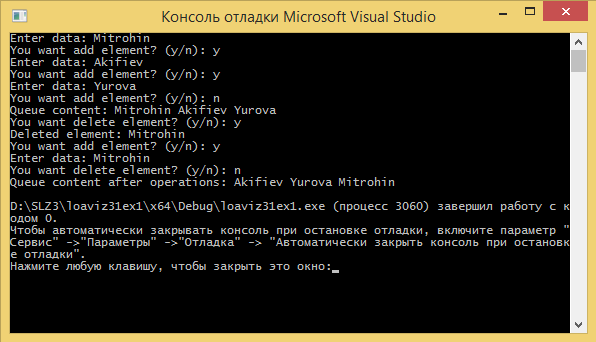
return 0;

}

**Результат**

****





**Листинг loaviz31ex2.cpp**

#include <iostream>

#include <string>

#include <locale.h>

using namespace std;

struct Node {

string data;

Node\* next;

};

class Stack {

private:

Node\* top;

public:

Stack() {

top = nullptr;

}

void push(const string& data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = top;

top = newNode;

}

string pop() {

if (top == nullptr) {

cout << "Stack is empty" << endl;

return "";

}

string data = top->data;

Node\* temp = top;

top = top->next;

delete temp;

return data;

}

bool remove(const string& data) {

if (top == nullptr) {

cout << "Stack is empty" << endl;

return false;

}

if (top->data == data) {

Node\* temp = top;

top = top->next;

delete temp;

return true;

}

Node\* current = top;

while (current->next != nullptr && current->next->data != data) {

current = current->next;

}

if (current->next != nullptr) {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

return true;

}

return false;

}

string peek() {

if (top == nullptr) {

cout << "Stack is empty" << endl;

return "";

}

return top->data;

}

bool isEmpty() {

return top == nullptr;

}

};

int main() {

setlocale(LC\_ALL, "ru\_RU.UTF-8");

Stack stack;

string data;

char choice;

while (true) {

cout << "Enter data: ";

cin >> data;

stack.push(data);

cout << "You want add element? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N') {

break;

}

}

cout << "Choose action:\n";

cout << "1. Delete top element of stack \n";

cout << "2. Delete specific element\n";

cout << "Your choice: ";

int action;

cin >> action;

if (action == 1) {

stack.pop();

cout << "Top element of stack deleted" << endl;

}

else if (action == 2) {

cout << "Enter element what you want to delete: ";

cin >> data;

bool removed = stack.remove(data);

if (removed) {

cout << "Element succesfully deleted from stack" << endl;

}

else {

cout << "Element not found in stack" << endl;

}

}

else {

cout << "Incorrect action choice" << endl;

}

cout << "Stack content: ";

while (!stack.isEmpty()) {

cout << stack.pop() << " ";

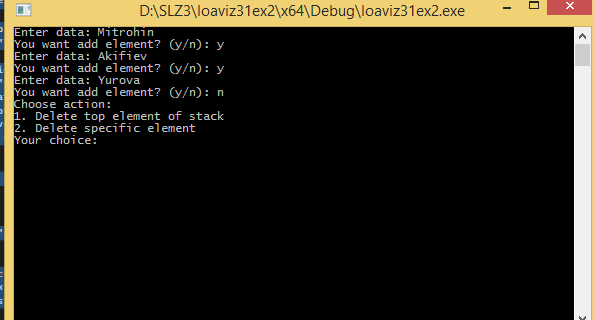
}

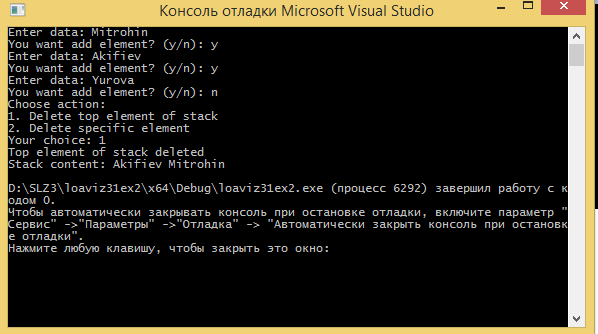
cout << endl;

return 0;

}

**Результат**

****

****

**Вывод**

Задание:

1. Реализовали приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализовали структуру данных *Очередь*.
3. \* На основе приведенного кода реализовали структуру данных *Стек*.