

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ПРАКТИЧЕСКАЯ РАБОТА № 1**

на тему «Простые структуры данных: массив, список, двунаправленный список, стек, очередь, дек, динамический массив»

Выполнила обучающаяся гр. ВКБ32

Куринский Кирилл Игоревич

Проверил

Савельев Василий Александрович

Ростов-на-Дону

2022

**Лабораторная работа №1**

**«Алгоритмы нахождения кратчайших путей в графе»**

Реализовать нахождение кратчайшего пути в графе между двумя заданными вершинами. Внутренняя структура графа не задается, но рекомендуется ориентироваться на решения эффективные и по памяти, и по времение реализации алгоритма. Также в задание входит создание вспомогательных библиотек, обеспечивающих ввод/вывод графа.

**Задание:**

Создать библиотеку, обеспечивающую ввод и вывод графа в бинарнойй форме. Использовать для сохранения значений 32-разрабные целые поля в сетевом порядке байтов. Написать тесты и отладить библиотеку. Измерить скорость ввода/вывода. Оценить ускорение ввода/вывода при переходе к бинарному формату.

#include <iostream>  
#include <queue> // очередь  
#include <stack> // стек  
using namespace std;  
struct Edge {  
  int begin;  
  int end;  
};  
int main()  
{  
  system("chcp 1251");  
  system("cls");  
  queue<int> Queue;  
  stack<Edge> Edges;  
  int req;  
  Edge e;  
  int mas[7][7] = { { 0, 1, 1, 0, 0, 0, 1 },  
  { 1, 0, 1, 1, 0, 0, 0 },  
  { 1, 1, 0, 0, 0, 0, 0 },  
  { 0, 1, 0, 0, 1, 0, 0 },  
  { 0, 0, 0, 1, 0, 1, 0 },  
  { 0, 0, 0, 0, 1, 0, 1 },  
  { 1, 0, 0, 0, 0, 1, 0 } };  
  int nodes[7]; // вершины графа  
  for (int i = 0; i < 7; i++) // исходно все вершины равны 0  
    nodes[i] = 0;  
  cout << "N = "; cin >> req; req--;  
  Queue.push(0); // помещаем в очередь первую вершину  
  while (!Queue.empty())  
  {  
    int node = Queue.front(); // извлекаем вершину  
    Queue.pop();  
    nodes[node] = 2; // отмечаем ее как посещенную  
    for (int j = 0; j < 7; j++)  
    {  
      if (mas[node][j] == 1 && nodes[j] == 0)  
      { // если вершина смежная и не обнаружена  
        Queue.push(j); // добавляем ее в очередь  
        nodes[j] = 1; // отмечаем вершину как обнаруженную  
        e.begin = node; e.end = j;  
        Edges.push(e);  
        if (node == req) break;  
      }  
    }  
    cout << node + 1 << endl; // выводим номер вершины  
  }  
  cout << "Путь до вершины " << req + 1 << endl;  
  cout << req + 1;  
  while (!Edges.empty()) {  
    e = Edges.top();  
    Edges.pop();  
    if (e.end == req) {  
      req = e.begin;  
      cout << " <- " << req + 1;  
    }  
  }  
  cin.get(); cin.get();  
  return 0;  
}