Министерство образования Российской Федерации   
Пензенский государственный университет   
Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной №6   
На тему: «Обход графа в ширину»  
По дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

Выполнили студенты группы 20ВВ2:

Борунов Д.А

Сурков М.С

Приняли:

Митрохин М.А. Юрова О.В.

**Пенза 2020г**

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**3.**\* Реализуйте процедуру поиска расстояний для графа, представленного списками смежности.

**Задание 2\***

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.
2. Реализуйте процедуру поиска расстояний на основе обхода в глубину для графа, представленного списками смежности.
3. Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.

**Методы решения заданий**

**Задание 1.1 и 1.2.**

Создать матрицу смежности графа (двумерный динамический массив). Описать функцию:

void BSFD(int st, int\*\* b, int num\_of\_elem, int \*vis)

int st **–**текущая и стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

int \*vis – указатель на массив посещенных вершин

int num\_of\_elem – количество элементов в сгенерированной матрице 11

1. Перед вызовом функции **BSFD** заполнить массив не посещенных вершин -1(-1 индикатор того, что вершина не посещена, так как 0 может быть расстоянием).
2. В теле функции реализовать алгоритм обхода графа в ширину на основе очереди Q с небольшими изменениями. В теле цикла, проверяющего наличие смежной вершины не посещенной по отношению к текущей, изменить условие на if(b[t][i]==1 && vis[i]==-1) и помечать вершину посещенной путем инкрементирования значения текущей вершины st в массиве посещенных вершин, тем самым определяя ее расстояние от стартовой вершины visited[i] = visited[st] + 1;
3. Вывести массив посещенных вершин в консоль

**Задание 2.1**

**Алгоритм поиска расстояния на основе обхода в глубину (не рекурсивный метод)**

Описать функцию: void DSFD(int st, int\*\* b, int num\_of\_elem, int\* visited)

void BSFD(int st, int\*\* b, int num\_of\_elem, int \*vis)

int st **–**текущая и стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

int \*vis – указатель на массив посещенных вершин

int num\_of\_elem – количество элементов в сгенерированной матрице

Тело данной функции повторяет тело функции из **BSFD** за исключением того, что вместо очереди, используется стек.

**Алгоритм поиска расстояния на основе обхода в глубину (рекурсивный метод)**

Описать функцию: void DSFD\_Recr(int st, int\*\* b, int num\_of\_elem, int\* visited) {

int i;

printf("%d ", st+1);

for (i = 0; i < num\_of\_elem; i++)

{

if (b[st][i]==1 and visited[i]==-1)

{

visited[i] = visited[st] + 1;

DSFD\_Recr(i, b, num\_of\_elem, visited);

}

}

}

int st **–**текущая и стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

int \* visited – указатель на массив посещенных вершин

int num\_of\_elem – количество элементов в сгенерированной матрице

1. Перед вызовом функции **BSFD\_Recr** заполнить массив не посещенных вершин -1(-1 индикатор того, что вершина не посещена, так как 0 может быть расстоянием).
2. В теле функции реализовать алгоритм обхода графа в глубину (рекурсивный метод). В теле цикла, проверяющего наличие смежной не посещенной вершины по отношению к текущей применить условие (b[st][i]==1 and visited[i]==-1) после которого следует помечать вершину посещенной путем инкрементирования значения текущей вершины st в массиве посещенных вершин, тем самым определяя ее расстояние от стартовой вершины visited[i] = visited[st] + 1;
3. Вывести массив посещенных вершин в консоль

**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <cstdlib>

#include <windows.h>

#include <conio.h>

#include <ctime>

#include <cmath>

#include <stack>

#include <queue>

using namespace std;

int max\_size = 7, i, j, start;

int\* DIST = (int\*)malloc(max\_size \* sizeof(int));

void BSFD(int st, int\*\* b, int num\_of\_elem, int \*vis) {

queue<int> Q;

int t;

Q.push(st);

vis[st] = 0;

while (!Q.empty())

{

t = Q.front();

Q.pop();

printf("%d ", t + 1);

for (int i = 0; i < num\_of\_elem; i++)

{

if(b[t][i]==1 && vis[i]==-1)

{

vis[i] = vis[t] + 1;

Q.push(i);

}

}

}

}

void DSFD(int st, int\*\* b, int num\_of\_elem, int\* visited) {

stack <int> Q;

int t;

for (i = 0; i < max\_size; i++)

{

visited[i] = -1;

}

Q.push(st);

visited[st] = 0;

while (!Q.empty())

{

t = Q.top();

Q.pop();

printf("%d ", t + 1);

for (int i = 0; i < num\_of\_elem; i++)

{

if (b[t][i] == 1 && visited[i] == -1)

{

Q.push(i);

visited[i] = visited[t] + 1;

}

}

}

}

void DSFD\_Recr(int st, int\*\* b, int num\_of\_elem, int\* visited) {

int i;

printf("%d ", st+1);

for (i = 0; i < num\_of\_elem; i++)

{

if (b[st][i]==1 and visited[i]==-1)

{

visited[i] = visited[st] + 1;

DSFD\_Recr(i, b, num\_of\_elem, visited);

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int\*\* a = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

a[i] = new int[max\_size];

}

for (i = 0; i < max\_size; i++)

{

for (j = i + 1; j < max\_size; j++)

{

a[i][j] = rand() % 2;

a[j][i] = a[i][j];

}

}

for (i = 0; i < max\_size; i++)

{

DIST[i] = -1;

for (j = 0; j < max\_size; j++)

{

if (i == j)

a[i][j] = 0;

printf("%2d", a[i][j]);

}

printf("\n");

}

printf("\nПоиск расстояний обходомм в ширину\ncтартовая вершина - ");

scanf\_s("%d", &start);

BSFD(start - 1, a, max\_size, DIST);

printf("\nРасстояние из первой:\n");

for (i = 0; i < max\_size; i++) {

printf("до %d = %d\n", i+1, DIST[i]);

DIST[i] = -1;

}

printf("\nПоиск расстояний обходомм в глубину\ncтартовая вершина - ");

scanf\_s("%d", &start);

DSFD(start - 1, a, max\_size, DIST);

printf("\nРасстояние из первой:\n");

for (i = 0; i < max\_size; i++) {

printf("до %d = %d\n", i + 1, DIST[i]);

DIST[i] = -1;

}

printf("\nПоиск расстояний обходомм в глубину рекурсией\ncтартовая вершина - ");

scanf\_s("%d", &start);

DIST[start-1]=0;

DSFD\_Recr(start - 1, a, max\_size, DIST);

printf("\nРасстояние из первой:\n");

for (i = 0; i < max\_size; i++) {

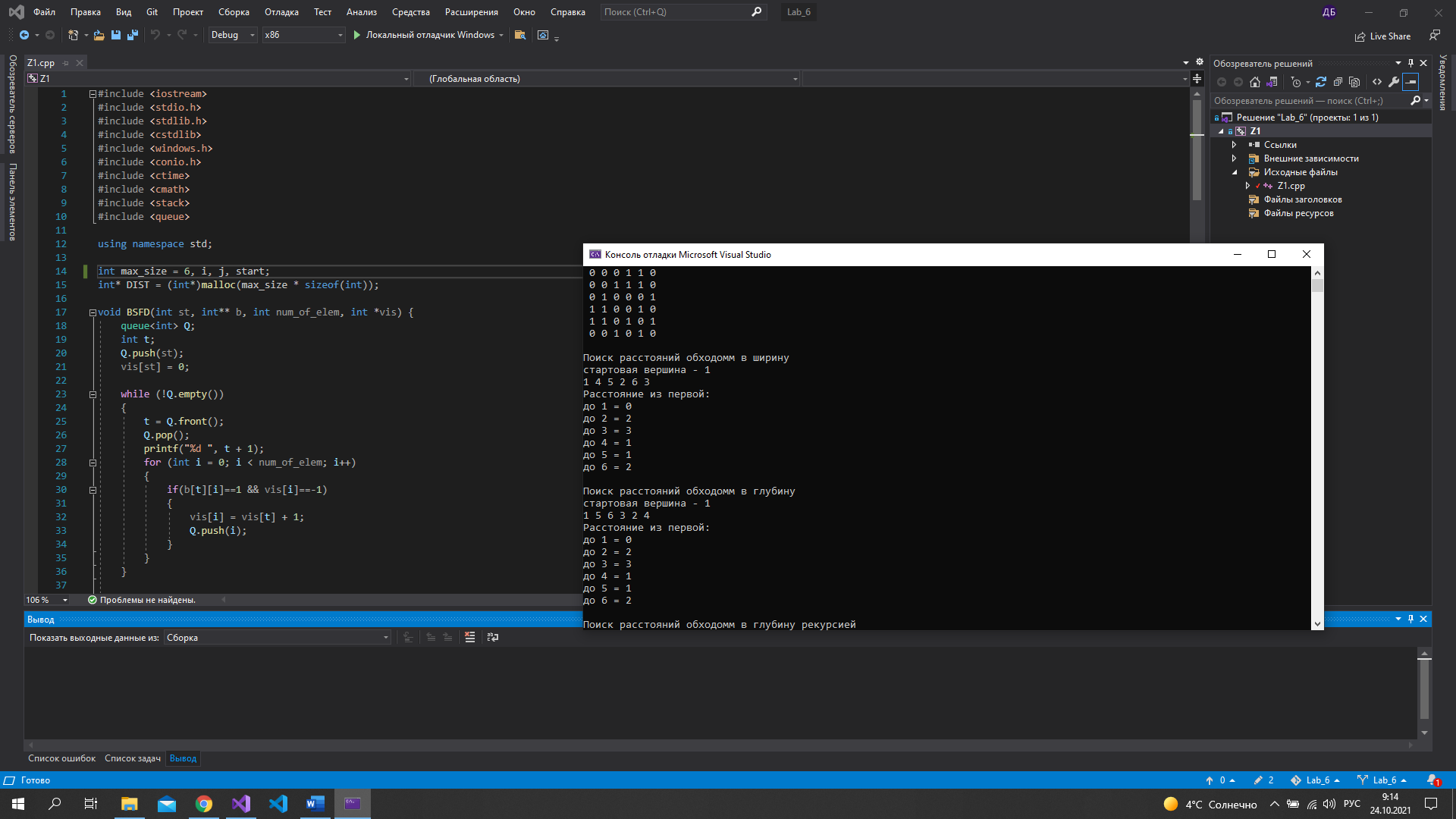
printf("до %d = %d\n", i + 1, DIST[i]);

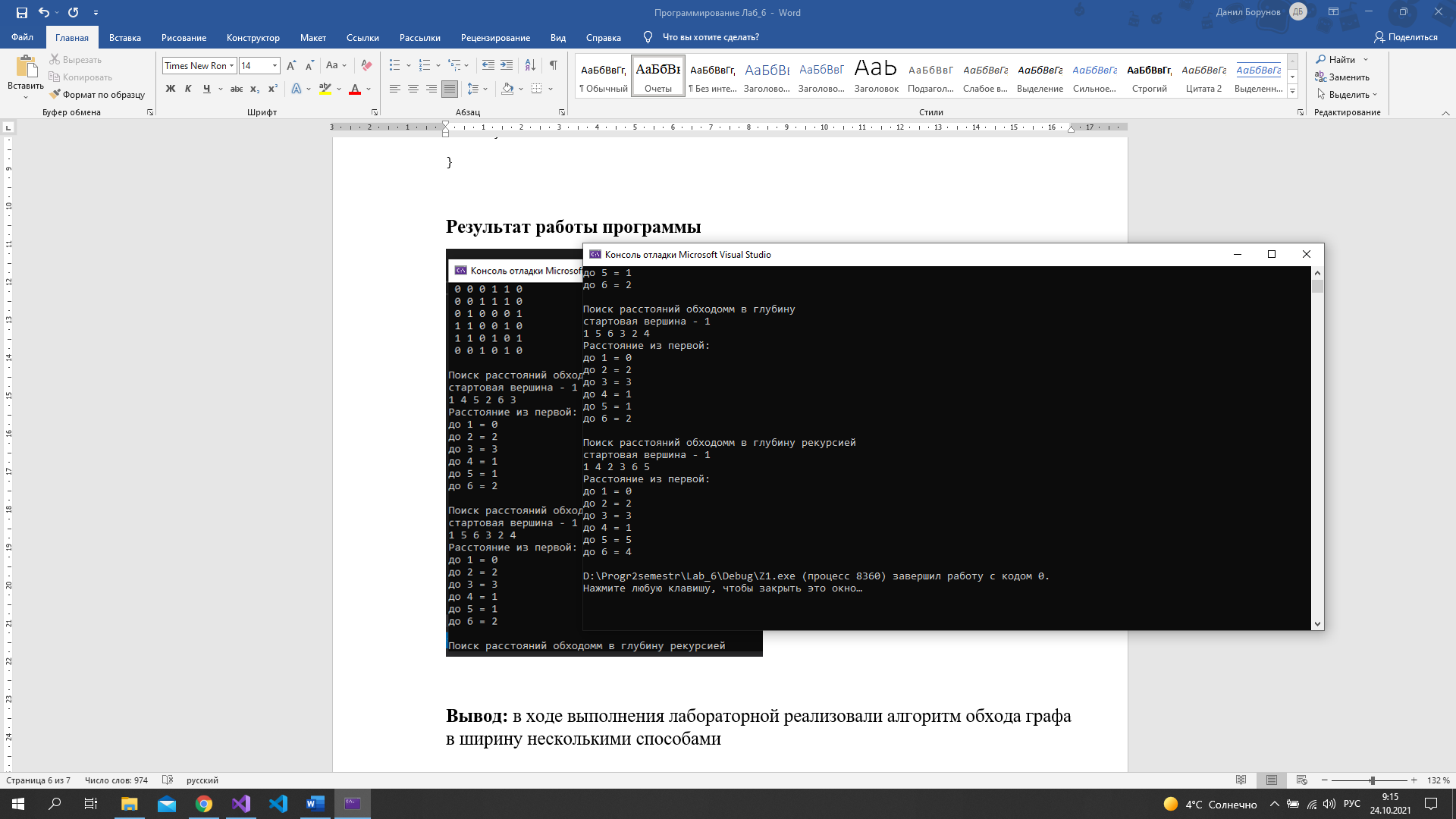
DIST[i] = -1;

}

}

**Результат работы программы**





**Вывод:** в ходе выполнения лабораторной реализовали алгоритм поиска расстояний в графе несколькими способами