Пензенский Государственный университет

Кафедра «Вычислительной техники»

Отчёт

По лабораторной работе №6

По дисциплине «Л и ОА в ИЗ»

На тему: «Поиск расстояний в графе»

***Выполнил студент группы 19ВВ1:***

Гусев В

**Приняли:**

Митрохин М.А.

Юрова О.В.

**Цель работы:** Изучить алгоритм поиска расстояний в графе.

**Задание:**

1. Сгенерировать (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Вывести матрицу на экран.
2. Для сгенерированного графа осуществить процедуру поиска расстояний. При  реализации алгоритма в качестве очереди использовать класс **queue** из стандартной библиотеки С++.

**Ход работы**

Поиск расстояний – довольно распространенная задача анализа графов.

Для поиска расстояний можно использовать процедуры обхода графа. Для этого при каждом переходе в новую вершину необходимо запоминать, сколько шагов до нее мы сделали. При этом вектор, который хранил информацию о посещении вершин становится вектором расстояний. Довольно просто модернизировать для поиска расстояний в графе алгоритм обхода в ширину, т.к. этот алгоритм проходит вершины по уровням удаленности, то для не ориентированного графа для вершин каждого следующего уровня глубины расстояние от исходной вершины увеличивается на 1. Удалённость в данном случае понимается как количество ребер, по которым необходимо прейти до достижения вершины.

**Листинг**

// laba5.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "iostream"

#include "conio.h"

#include "locale.h"

#include <queue>

using namespace std;

int \*dist; // указатель на вектор

int \*\*matr;//указатель для массива указателей

void BFS(int s , int n)

{

queue <int> Q;

Q.push(s);

dist[s] = 0;

printf("\Порядок обхода:");

while(!Q.empty())

{

s = Q.front();

Q.pop();

printf("%d", s);

for(int i = 0;i < n; i++)

{

if((matr[s][i] == 1) && dist[i] == -1)

{

Q.push(i);

dist[i] = dist[s] + 1;

}

}

}

printf("\nРасстояние: ");

for(int i = 0; i < n; i++)

{

if(dist[i] != -1)

{

printf("%d", dist[i]);

}

dist[i] = -1;

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

system("chcp 1251");

system("cls");

int n,i,j, Nach;

printf("Введите размер матрицы:");

scanf("%d", &n);

dist = (int\*)malloc(n \* sizeof(int));

matr = (int\*\*)malloc(n \* sizeof(int\*));

//выделение памяти под массив указателей

for(i=0; i<n; i++){

matr[i] = (int\*)malloc(n \* sizeof(int));

}

//выделение памяти для массива значений

for( i = 0; i < n; i++)

{

for( j = 0; j < n; j++)

{

if(i == j)

{

matr[i][j] = 0;

}

if (i < j)

{

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

matr[j][i] = matr[i][j];

}

}

}

for(i = 0; i < n; i++)

{

dist[i] = -1;

}

printf( "Result: ");

for(i = 0; i < n; i++)

{

printf("\n");

for (j = 0; j < n; j++)

printf("%d\t",matr[i][j]);

}

printf( " \n ");

printf( " \n ");

for(int i = 0; i < n; i++)

{

printf("%d\t", dist[i]);

}

printf( " \n ");

for(int i = 0; i < n; i++)

{

BFS(i, n);

printf( " \n ");

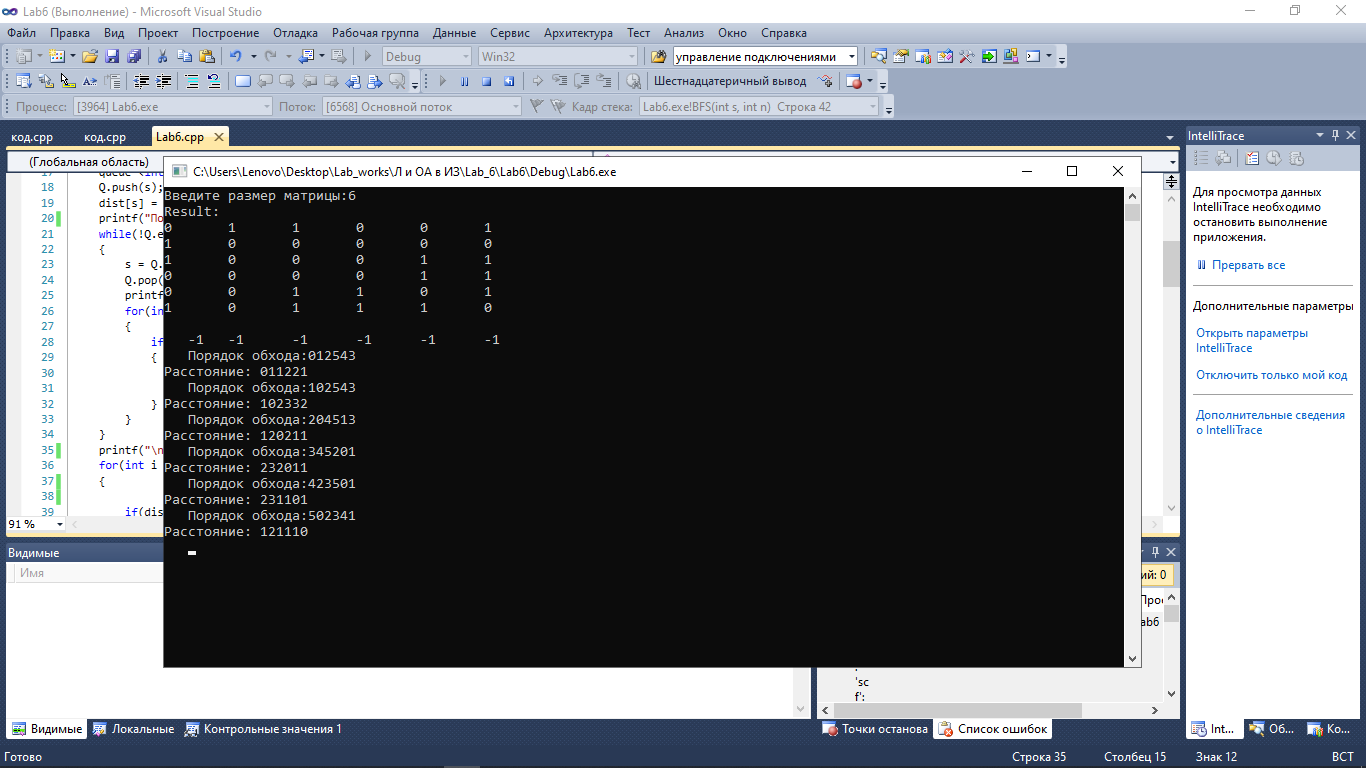
}

getch();

}

**Результат работы программы**

Результат работы алгоритма на матрице 6х6



**Вывод:** В ходе данной работы был изучен алгоритм поиска расстояний в графе. Так же данный метод был реализован для матрицы , размер которой пользователь может вводить с экрана.