Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах.»

на тему: «ОБХОД ГРАФА В ГЛУБИНУ.»

Выполнили**:**

студенты группы 21ВВ4

Федоренко Вероника

Роганов Данила

Проверили:

Юрова О.В,

Акифьев И.В.

Пенза 2022

**Цель работы**

Научиться реализовывать алгоритм поиска в глубину.

**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 обхода в глубину, реализованную в соответствии с приведенным выше описанием.

Листинг

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

bool checkIsVertertexAdjacent(int\*\* sourceMatrix, int firstVertexToDelite, int secondVertexToDelite) {

if (sourceMatrix[firstVertexToDelite][secondVertexToDelite] == 1) {

return true;

}

return false;

}

void allocateMatrix(int\*\*\* sourceMatrix, int columnCount, int rowsCount) {

\*sourceMatrix = (int\*\*)malloc(columnCount \* sizeof(int\*));

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

{

(\*sourceMatrix)[i] = (int\*)calloc(rowsCount, sizeof(int));

}

}

void fillMatrixRandomElements(int\*\* sourseMatrix, int columnCount, int rowsCount) {

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

{

for (int j = 0; j < columnCount; j++)

{

if (i == j) {

sourseMatrix[i][j] = 0;

}

else

{

sourseMatrix[i][j] = rand() & 1;

sourseMatrix[j][i] = sourseMatrix[i][j];

}

//sourseMatrix[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

}

}

}

void printMatrix(int\*\* sourseMatrix, int columnCount, int rowsCount) {

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

{

printf("\n");

for (int j = 0; j < columnCount; j++)

{

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

}

}

}

void depthFirstTraversal(int\*\* sourceMatrix, int matrixSize, int\* visitedVertex, int vertex) {

visitedVertex[vertex] = 1;

printf("%d", vertex);

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

{

//в текущей вершине есть связь И вершина не посещена

if (sourceMatrix[vertex][i] == 1 && visitedVertex[i] == 0) {

depthFirstTraversal(sourceMatrix, matrixSize, visitedVertex, i);

}

}

}

void main() {

srand(time(NULL));

setlocale(LC\_ALL, "Rus");

int\*\* sourceMatrix = NULL;

int matrixSize = 0;

int\* visitedVertexs = NULL;

printf("Введите количество первой матрицы смежности = ");

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

visitedVertexs = (int\*)calloc(matrixSize, sizeof(int));

allocateMatrix(&sourceMatrix, matrixSize, matrixSize);

fillMatrixRandomElements(sourceMatrix, matrixSize, matrixSize);

printf("\nИсходная матрица смежности\n");

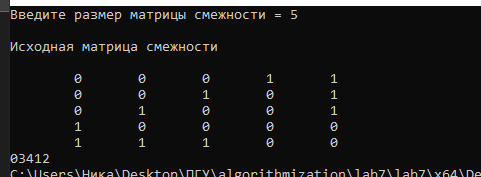
printMatrix(sourceMatrix, matrixSize, matrixSize);

printf("\n");

depthFirstTraversal(sourceMatrix, matrixSize, visitedVertexs, 0);

}

Результат



Вывод: в ходе выполнения лабораторной работы были разработаны программу, реализующую алгоритм обхода графа в глубину.