Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №5

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

## на тему «Обход графа в ширину»

Выполнил студент группы 19ВВ4:

Земсков М.А.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** освоить алгоритм обхода графа в ширину на матрицах.

**Общие сведения:**

Обход графа в ширину – еще один распространенный способ обхода

графов.

Основная идея такого обхода состоит в том, чтобы посещать вершины

по уровням удаленности от исходной вершины. Удалённость в данном

случае понимается как количество ребер, по которым необходимо прейти до

достижения вершины. Так, алгоритм обхода в ширину продолжает осматривать уровень за

уровнем, пока не пройдет все доступные вершины.

Чтобы не заходить повторно в уже пройденные вершины, они

помечаются, как и в алгоритме обхода в глубину.

Для того, чтобы проход осуществлялся по уровням необходимо

хранить информацию о требуемом порядке посещения вершин. Вершины,

которые являются ближайшими соседями исходной вершины (из которой

начат обход) должны быть посещены раньше, чем соседи соседей и т.д.

Такой порядок позволяет задать структура данных «очередь». Просматривая

строку матрицы смежности (или список смежности) для текущей вершины

мы помещаем всех её ещё не посещенных соседей в очередь. На следующей

итерации текущей вершиной становится та, которая стоит в очереди первой и

уже её не посещенные соседи будут помещены в очередь. Но место в очереди

они займут после тех вершин, которые были помещены туда на предыдущих

итерациях.

**Практическая часть**

1. Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Выведите матрицу на экран.

2. Для сгенерированного графа осуществите процедуру обхода в ширину,

реализованную в соответствии с приведенным выше описанием. При

реализации алгоритма в качестве очереди используйте класс queue из

стандартной библиотеки С++.

**Листинг.**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#include <queue>

using namespace std;

int\*\* G;

int\* vis;

int ver;

void BFS(int s, int rows) {

int v;

queue <int> q;

q.push(s);

vis[s] = 1;

while (!q.empty()) {

v = q.front();

printf(" %d", v);

q.pop();

for (int i = 0; i < rows; i++) {

if (G[v][i] == 1 && vis[i] == 0) {

vis[i] = 1;

q.push(i);

}

}

}

}

int main()

{

int rows;

int n = 0;

int k = 0;

system("cls");

printf("rows:");

scanf("%d", &rows);

srand(time(NULL));

G = (int\*\*)malloc(rows \* sizeof(int\*));

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

{

G[i] = (int\*)malloc(rows \* sizeof(int));

}

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

{

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

{

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

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

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

{

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

{

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

}

printf("\n");

}

printf("\n");

vis = (int\*)malloc(rows \* sizeof(int));

for (int i = 0; i < rows; i++) {

vis[i] = 0;

}

printf("\n");

printf("Vershina:");

scanf("%d", &ver);

printf("\n");

BFS(ver, rows);

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

free(G[i]);

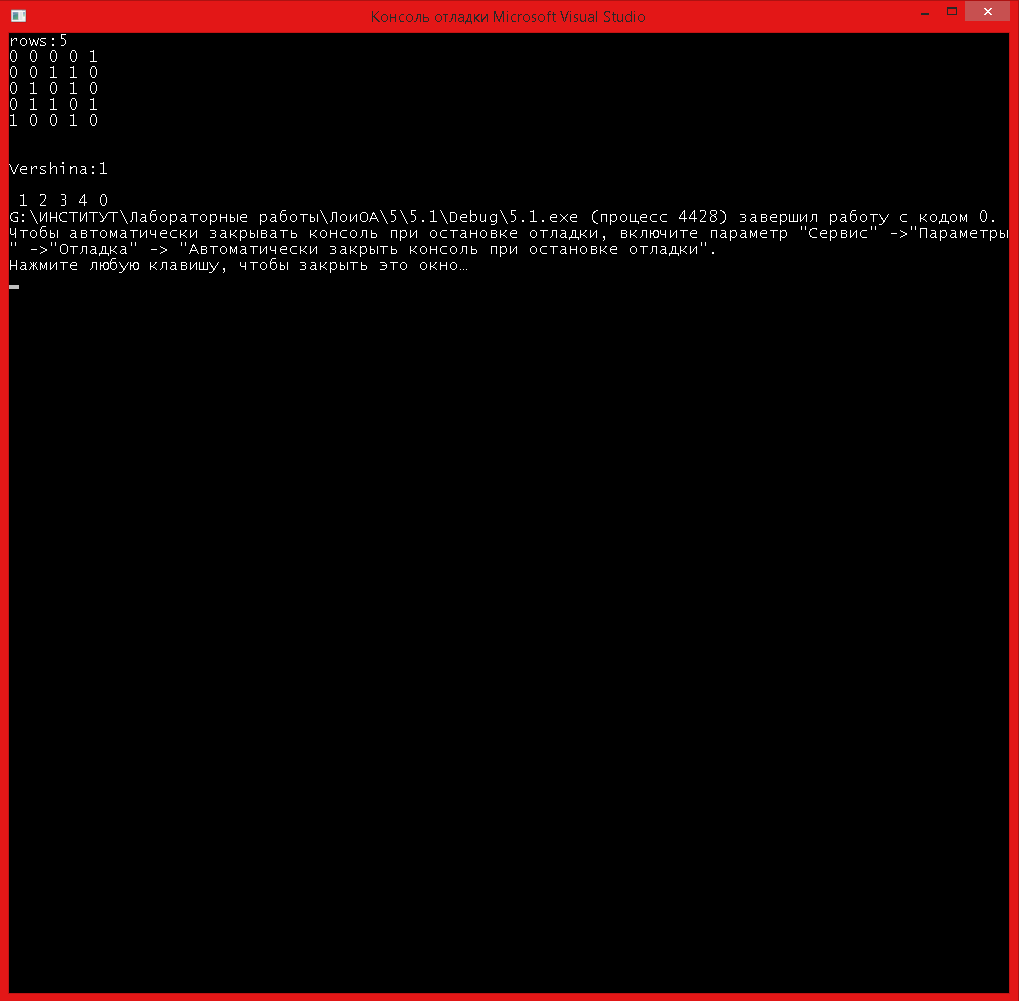
free(G);

getchar();

return 0;

}

## Результат работы.



**Вывод:** освоили алгоритм обхода графа в ширину на матрицах .