Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

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

на тему «обход графа в ширину»

**Выполнил:**

студент группы 20ВВ3

Шадрин Данила

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Обход графа в ширину.

**Цель работы** – изучение алгоритма обхода матрицы смежности в ширину с помощью библиотеки queue.

# 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 ширину, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**3.**\* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

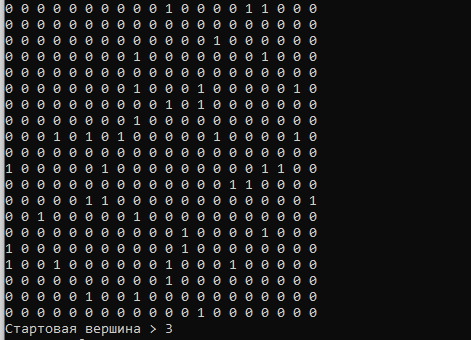
**Задание 2\***

1. Для матричной формы представления графов реализуйте алгоритм обхода в ширину с использованием очереди, построенной на основе структуры данных «список», самостоятельно созданной в лабораторной работе № 3.
2. Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс **queue** и использующего очередь, реализованную самостоятельно) для графов разных порядков.

# Результат работы программы

**Задание 1.1.**

Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

**Задание 1.2. Результат обхода в ширину.**



**Вывод:** яизучил алгоритм обхода матрицы смежности в ширину с помощью библиотеки queue.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <vector>

#include <queue>

using namespace std;

void BFS(int n, bool\* Arr2, int i, int\*\* Arr)

{

queue<int> que;

que.push(i);

Arr2[i] = true;

while (!que.empty())

{

i = que.front();

que.pop();

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

for (int j = 1; j < n; j++)

{

if (Arr[i][j] == 1 && Arr2[j] == false)

{

que.push(j);

Arr2[j] = true;

}

}

}

}

void Create\_matrix() {

int start, n, t = clock();

cout << "Задание 1" << endl;

cout << "Количество вершин > "; cin >> n;

bool\* visited = new bool[n];

int\*\* matrix;

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

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

//matrix[i] = new int[n];

visited[i] = false;

}

int s = 1;

srand(time(NULL));

for (int i = 0; i < n; i++) // создание матрицы

{

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

matrix[i][i] = 0;

for (int j = s; j < n; j++)

{

if (rand() % 100 > 10)

{

matrix[i][j] = 0;

}

else {

matrix[i][j] = 1;

}

}

s++;

}

s = 1;

for (int i = 0; i < n; i++) // неориентированный граф

{

for (int j = s; j < n; j++)

matrix[j][i] = matrix[i][j];

s++;

}

cout << "Matrix"; // вывод матрицы на экран

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

{

cout << endl;

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

{

cout << matrix[i][j] << " ";

}

}

cout << endl;

cout << "Стартовая вершина > "; cin >> start;

cout << "Порядок обхода: ";

BFS(n, visited, start -1, matrix);

delete[]visited;

cout << "\nТакты:" << ((float)t / CLOCKS\_PER\_SEC) << "\n";

system("pause");

}

void main() {

system("chcp 1251"); system("cls");

Create\_matrix();

}