Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в инженерных задачах»

### на тему «Обход графа в ширину»

**Выполнили студенты**

**группы 21вв1.1:**

Федулов Е.Д.

Жбанников Д.Н.

**Приняли:**

Юрова О.В.

Акифьев И.В.

Пенза, 2022

**Цель работы:** научиться обходить граф в ширину, находить время выполнения программы

**Задания:**

**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.

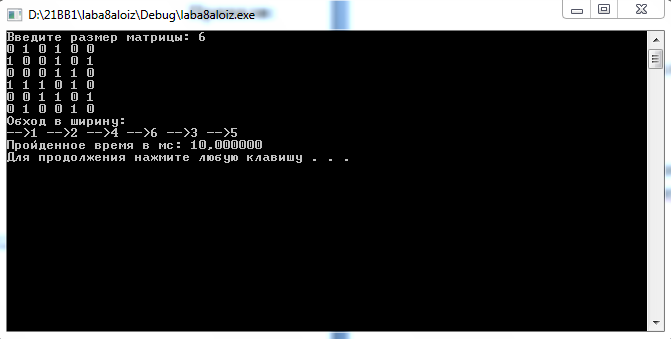
2. Оцените время работы двух реализаций алгоритмов обхода в ширину

(использующего стандартный класс queue и использующего очередь,

реализованную самостоятельно) для графов разных порядков.

**Ход работы**

**Результат программы для основного и дополнительного задания:**

****

**Вывод:** научились обходить граф в ширину,находить время выполнения программы.

**Общий листинг:**

#include <stdlib.h>

#include <stdio.h>

#include <queue>

#include "time.h"

#include <locale.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

queue<int> Queue;

int size;

printf("Введите размер матрицы: ");

scanf("%d", &size);

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

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

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

}

srand(time(NULL));

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

for (int j = 0; j < size; j++) {

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

}

}

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

for (int j = 0; j < size; j++) {

if (i == j) {

G[i][j] = 0;

}

if (G[i][j] = G[j][i]) {

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

}

}

}

for (size\_t i = 0; i < size; ++i) {

for (size\_t j = 0; j < size; ++j)

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

printf("\n");

}

int \*nodes = (int\*)malloc(sizeof(int\*) \* size);

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

nodes[i] = 0;

int start = clock();

Queue.push(0);

printf("Обход в ширину:\n");

while (!Queue.empty())

{

int node = Queue.front();

Queue.pop();

nodes[node] = 2;

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

{

if (G[node][j] == 1 && nodes[j] == 0)

{ // если вершина смежная и не обнаружена

Queue.push(j);

nodes[j] = 1;

}

}

printf("-->%d ", node + 1);

}

int end = clock();

double time = (end-start);

printf("\nПройденное время в мс: %lf \n", time);

system("Pause");

return 0;

}