|  |
| --- |
| Министерство образования Российской Федерации  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 19ВВ3:  Трубаненко А.Г.  Принял:  Митрохин М. А. |
| Пенза  2020 |

**Цель работы**

Реализовать алгоритм обхода графа в ширину.

**Листинг программы**

**Source.cpp**

#include <stdio.h>

#include <time.h>

#include <windows.h>

#include <conio.h>

#include <queue>

#include <iostream>

#include "Matrix\_Graph.h"

#pragma warning (disable : 4996)

int first\_false(bool\* arr, int size) {

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

if (!arr[i]) { return i; };

}

return -1;

}

void BFS(graph\* g,int v, bool\* visited) {

std::queue <int> qu;

qu.push(v);

visited[v] = 1;

while (!qu.empty()) {

v = qu.front();

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

qu.pop();

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

if ((g->matrix[v][i] == 1) && (visited[i] == 0)) {

qu.push(i);

visited[i] = 1;

}

}

}

}

void component(graph\* g) {

bool\* visited = (bool\*)calloc(g->size, 1);

int next\_vertex = 0;

while (next\_vertex != -1) {

BFS(g, next\_vertex, visited);

printf("\n");

next\_vertex = first\_false(visited, g->size);

}

free(visited);

}

void main()

{

int ver;

int size;

printf("\nEnter the probability from 0 to 100: ");

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

printf("Eneter size of graph:");

scanf("%d", &size);

graph\* g = init\_graph(size);

srand(time(NULL));

generate\_matrix(g,ver);

print\_graph(g);

printf("\nComponents:\n");

component(g);

}

**Matrix\_graph.cpp**

#include "Matrix\_Graph.h"

int\*\* matrix(int size)

{

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

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

{

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

}

return matrix;

}

void generate\_matrix(Graph\* graph, int ver)

{

int tmp;

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

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

if (j != i) {

tmp = rand() % 100;

if (tmp < ver) {

graph->matrix[i][j] = 1;

}

else {

graph->matrix[i][j] = 0;

}

graph->matrix[j][i] = graph->matrix[i][j];

}

else if (i == j) graph->matrix[i][j] = 0;

}

}

}

Graph\* init\_graph(int size)

{

Graph\* graph = (Graph\*)malloc(sizeof(Graph));

graph->matrix = matrix(size);

graph->size = size;

return graph;

};

void free\_graph(Graph\* graph)

{

for (int i = 0; i < graph->size; i++)

{

free(graph->matrix[i]);

}

free(graph->matrix);

free(graph);

}

void print\_graph(Graph\* graph)

{

printf(" ");

for (int k = 1; k <= graph->size; k++) printf(" %d", k);

printf("\n -----------\n");

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

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

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

printf("%d ", graph->matrix[i][j]);

}

printf("\n");

}

}

**Matrix\_graph.h**

#pragma once

#include "stdlib.h"

#include "stdio.h"

typedef struct Graph

{

int\*\* matrix;

int size;

}graph;

int\*\* matrix(int size);

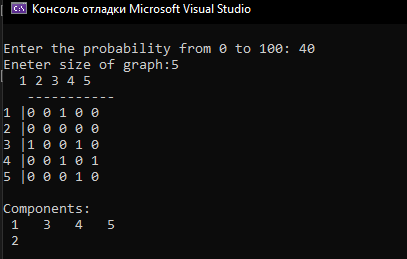
void generate\_matrix(Graph\* graph, int ver);

Graph\* init\_graph(int size);

void free\_graph(Graph\* graph);

void print\_graph(Graph\* graph);

### **Результат работы программы:**



**Вывод:** Реализовал алгоритм обхода в ширину графа, представленного матрицей смежности.