|  |
| --- |
| 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» |
|  |
|  |
| Выполнила студентка группы 19ВВ2:  Ходакова М.А.  Принял:  Митрохин М. А  Юрова О.В. |
| Пенза  2020 |

### **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

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

#include <stdio.h>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

queue <int> Q; //очередь

int i, j, N, st, m;

int\*\* graph; // матрица смежности

bool\* visited = new bool[m]; // посещенные вершины = выделение памяти под массив

void BFS(int st) //алгоритм обхода в ширину

{

Q.push(st); //заносим вершину в очередь

visited[st] = true; //помечаем вершину как пройденную

while (!Q.empty())

{

st = Q.front(); //перемещаем в начало очереди

Q.pop(); //выносим из очереди

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

for (j = 0; j < N; j++)

{

if ((graph[st][j] == 1) && (!visited[j]))

{

visited[j] = true; //помечаем вершину, удовлетворяющую условию, как пройденную

Q.push(j); //заносим ее в очередь

}

}

}

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("Ведите размер матрицы ");

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

srand(time(NULL));

graph = new int\* [N]; // выделяем память под граф

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

graph[i] = new int[N];

}

for (i = 0; i < N; ++i)

{

for (j = i; j < N; ++j)

{

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

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

for (j = 0; j < N; j++)

{

printf("%4d ", j + 1); //горизонтальный порядок чисел сверху матрицы

}

printf("\n\n");

for (i = 0; i < N; i++) {

printf(" %d ", i + 1); // вертикальный порядок фиссел слева матрицы

for (j = 0; j < N; j++)

{

printf("%4d ", graph[i][j]);

}

printf("\n\n");

}

printf("\nВведите вершину: ");

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

while (st > N) {

printf("\nТакая вершина не существует!\n");

printf("\nВведи вершину: ");

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

}

for (i = 0; i < N; i++)

{

visited[i] = false; //помечаем все элементы как не пройденный

}

printf("\n Результат обхода матрицы смежности графа в ширину:");

BFS(st - 1);

printf("\n\n");

system("pause");

}

**Результат работы программы:**

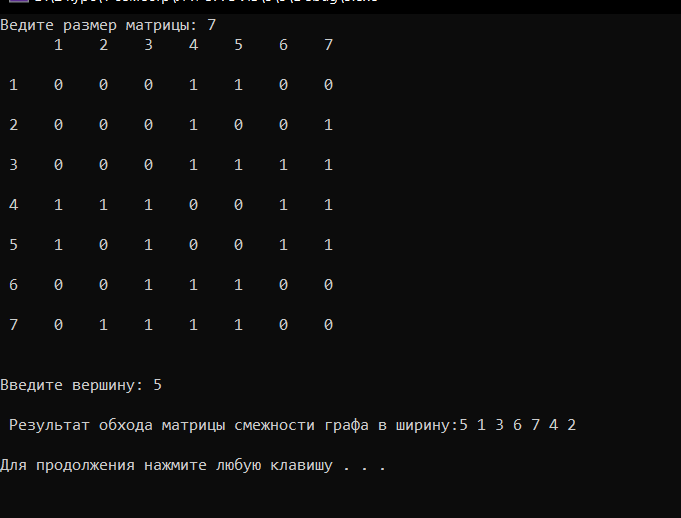


Рис. 1 Задание 1(пункты 1 и 2)

**Вывод:**

В результате работы был реализован обход в ширину с помощью работы с очередью. Идея обхода в ширину заключается в посещении вершин на каждом уровне удаленности.