МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Частное учреждение образования

«Гродненский колледж бизнеса и права»

**Лабораторная работа № 11**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Решение задач, основанных на поиске в ширину и в глубину в графе

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 11**

Тема: Решение задач, основанных на поиске в ширину и в глубину в графе.

Цель:

Образовательная**:**

* Обучить основным алгоритмам обхода графа и научиться решать задачи обхода графа на основе поиска в ширину и поиска в глубину,

Развивающая:

* научить анализировать алгоритмы обхода графа и научить решать задачи обхода графа на основе поиска в ширину и поиска в глубину,
* создать условия для развития способности четко формулировать свои мысли.

Воспитательная:

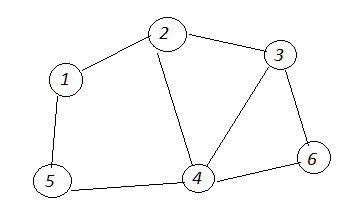
* воспитывать в обучающихся средствами урока уверенность в своих силах;

воспитывать сознательное и серьёзного отношения обучающихся к учебной дисциплине, убеждая их в том, что полученные знания пригодятся им в будущей деятельности.

Задачи: Освоение теоретического материала и выполнение индивидуального задания.

**ЗАДАЧИ**

Условие:



Алгоритм: Предоставлен преподавателю в письменном виде.

Решение:

**uses** crt;

**const**

n = 6;

**type**

MI = **array**[1..n, 1..n] **of** integer;

MB = **array**[1..n] **of** boolean;

**var**

i, j, start: integer;

visited: MB;

**const**

GM: MI =

((0, 1, 0, 0, 1, 0),

(1, 0, 1, 1, 0, 0),

(0, 1, 0, 1, 0, 1),

(0, 1, 1, 0, 1, 1),

(1, 0, 0, 1, 0, 0),

(0, 0, 1, 1, 0, 0));

**procedure** BFS(visited: mb; \_unit: integer);

**var**

queue: **array**[1..n] **of** integer;

count, head, i: integer;

**begin**

**for** i := 1 **to** n **do**

queue[i] := 0;

count := 0;

Head := 0;

count := count + 1;

queue[count] := \_unit;

visited[\_unit] := true;

**while** head < count **do**

**begin**

Head := Head + 1;

\_unit := queue[head];

write(\_unit, ' ');

**for** i := 1 **to** n **do**

**begin**

**if** (GM[\_unit, i] <> 0) **and** (**not** visited[i]) **then**

**begin**

count := count + 1;

queue[count] := i;

visited[i] := true;

**end**;

**end**;

**end**;

**end**;

**begin**

clrscr;

write('Начальная вершина - ');

readln(start);

writeln('Матрица смежности графа: ');

**for** i := 1 **to** n **do**

**begin**

visited[i] := false;

**for** j := 1 **to** n **do**

write(' ', GM[i, j]);

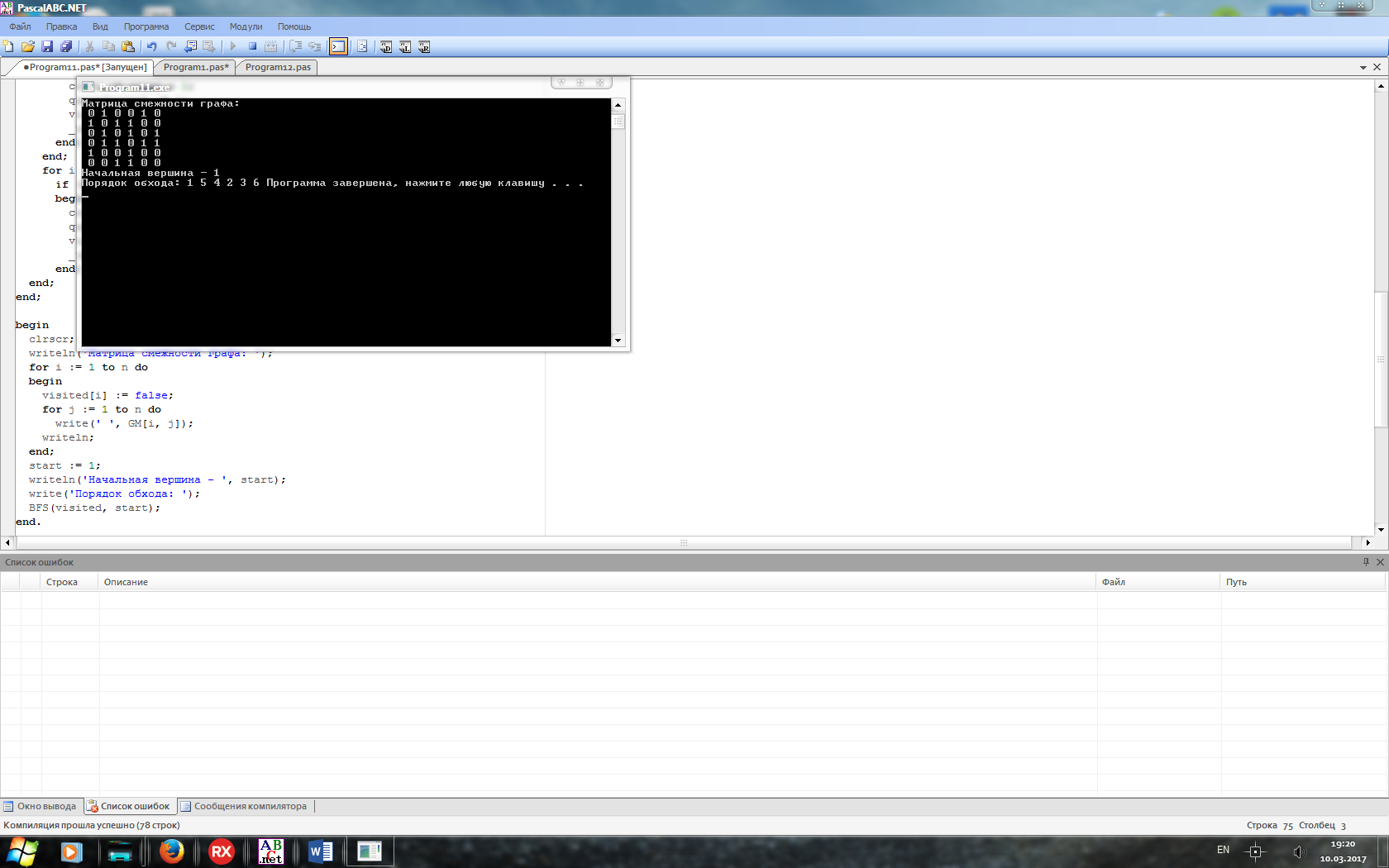
writeln;

**end**;

write('Порядок обхода: ');

BFS(visited, start);

**end**.



**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Как связаны между собой различные способы представления графов?

[**связным**](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), если для любых вершин u {\displaystyle u} , v {\displaystyle v} есть путьv {\displaystyle v} .

**сильно связным** или **ориентированно связным**, если он ориентированный, и из любой вершины в любую другую имеется ориентированный путь.

[**деревом**](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%28%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2%29), если он связный и не содержит нетривиальных циклов.

[**полным**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), если любые его две (различные, если не допускаются петли) вершины соединены ребром.

[**двудольным**](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B3%D1%80%D0%B0%D1%84), если его вершины можно разбить на два непересекающихся подмножества V 1 {\displaystyle V\_{1}} и V 2 {\displaystyle V\_{2}} так, что всякое ребро соединяет вершину из V 1 {\displaystyle V\_{1}} с вершиной из V 2 {\displaystyle V\_{2}} .

**k-дольным**, если его вершины можно разбить на k {\displaystyle k} непересекающихся подмножества V 1 {\displaystyle V\_{1}} , V 2 {\displaystyle V\_{2}} V k {\displaystyle V\_{k}} так, что не будет рёбер, соединяющих вершины одного и того же подмножества.

**полным двудольным**, если каждая вершина одного подмножества соединена ребром с каждой вершиной другого подмножества.

[**планарным**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), если граф можно изобразить диаграммой на плоскости без пересечений рёбер.

**взвешенным**, если каждому ребру графа поставлено в соответствие некоторое число, называемое весом ребра.

[**хордальным**](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%80%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), если граф не содержит индуцированных циклов с длиной больше трёх.

1. Как от вида или представления графа зависит временная сложность алгоритмов поиска в глубину и в ширину?

Древовидные графы обходятся дольше в глубину из-за расположения связей. В остальном обход влияет мало на время.

1. Как при реализации в коде выполняется возвращение из тупиковых вершин при обходе графа?

В цикле задается условие на проверку тупика вершины, если это так, берется следующая вершина цикла.