МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № \_\_

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил\_:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Брест – 2022

***Задание.***

1. Построить матрицу смежности и инцидентности для заданного графа. Изобразить граф.
2. Используя поиск в глубину и поиск в ширину написать программу, определяющую число компонент связности графа. Методы представляются в виде функций.
3. Варианты заданий указаны в таблице 1.
4. В таблице граф задан списком ребер.

**Ход работы**

|  |  |
| --- | --- |
| Матрица смежности: | Матрица инцидентности: |

Код функций:

|  |
| --- |
| void Graph::DFS(int\*& arr, int& size, int start) {  int s\_size = 0;  int\* stack = new int[s\_size];  bool\* used = new bool[vertices];  for (int i = 0; i < vertices; i++)  used[i] = 0;  push\_elem(stack, s\_size, start - 1);  used[start - 1] = 1;  while (s\_size > 0) {  int v = pop\_front\_elem(stack, s\_size);  push\_elem(arr, size, v);  for (int i = 0; i < vertices; i++)  if (matrix[v][i] && !used[i]) {  push\_elem(stack, s\_size, i);  used[i] = 1;  break;  }  }  delete[] stack;  delete[] used;  }  void Graph::BFS(int\* &arr, int& size, int start) {  int q\_size = 0;  int\* queue = new int[q\_size];  bool\* used = new bool[vertices];  for (int i = 0; i < vertices; i++)  used[i] = 0;  push\_elem(queue, q\_size, start - 1);  used[start - 1] = 1;  while (q\_size > 0) {  int v = pop\_back\_elem(queue, q\_size);  push\_elem(arr, size, v);  for (int i = 0; i < vertices; i++)  if (matrix[v][i] && !used[i]) {  push\_elem(queue, q\_size, i);  used[i] = 1;  }  }  delete[] queue;  delete[] used;  }  int Graph::CALC\_COMPONENTS\_DFS() {  int count = 0;  bool\* used = new bool[vertices];  for (int i = 0; i < vertices; i++)  used[i] = 0;  for (int i = 0; i < vertices; i++) {  if (used[i])  continue;  int size = 0;  int\* arr = new int[size];  DFS(arr, size, i + 1);  for (int j = 0; j < size; j++)  used[arr[j]] = 1;  delete[] arr;  count++;  }  delete[] used;  return count;  }  int Graph::CALC\_COMPONENTS\_BFS() {  int count = 0;  bool\* used = new bool[vertices];  for (int i = 0; i < vertices; i++)  used[i] = 0;  for (int i = 0; i < vertices; i++) {  if (used[i])  continue;  int size = 0;  int\* arr = new int[size];  BFS(arr, size, i + 1);  for (int j = 0; j < size; j++)  used[arr[j]] = 1;  delete[] arr;  count++;  }  delete[] used;  return count;  } |

Main и вывод:

|  |
| --- |
| int main()  {  string list = "(1,2),(1,3),(1,4),(2,3),(2,4),(3,4),(5,7),(6,8)";  Graph G(8, 8, list);  G.PRINT\_MATRIX();  G.PRINT\_DFS(1);  G.PRINT\_BFS(1);  G.PRINT\_DFS(5);  G.PRINT\_BFS(5);  G.PRINT\_DFS(6);  G.PRINT\_BFS(6);  G.PRINT\_AMOUNT\_COMPONENTS\_DFS();  G.PRINT\_AMOUNT\_COMPONENTS\_BFS();  } |

