МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Связность графов

Отчёт по лабораторной работе №5
по дисциплине
«Дискретная математика»
Вариант 8

Выполнил студент гр. ИВТб-1301-05-00 _____/Макаров С.А./
Руководитель преподаватель _____/Пахарева И.В./

Киров 2025

Цель

Цель лабораторной работы: изучение основ теории графов, формирование матрицы связности, разработка приложения на языке Паскаль или СИ согласно заданию.

Задание

Граф задан матрицей инцидентности в файле (вершин >= 4, дуг >= 4). Сформировать матрицу связности. Определить является ли граф несвязным.

Решение

Для решения задач подготовлен ориентированный граф, представленный на рисунке 1.

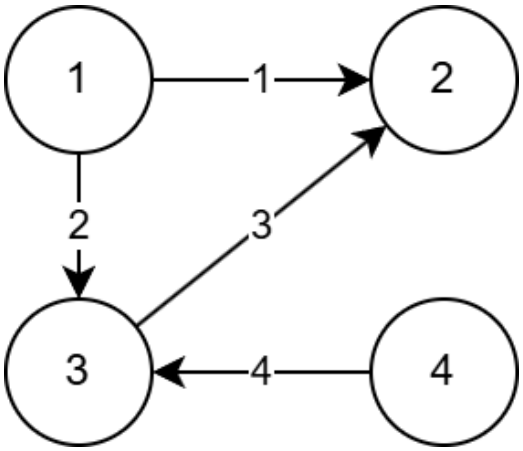

Рисунок 1 – Ориентированный граф и его дополнение

Перед разработкой составлены схемы алгоритмов для решения задач. На рисунке 2 представлены схемы подпрограмм ввода и вывода матрицы.

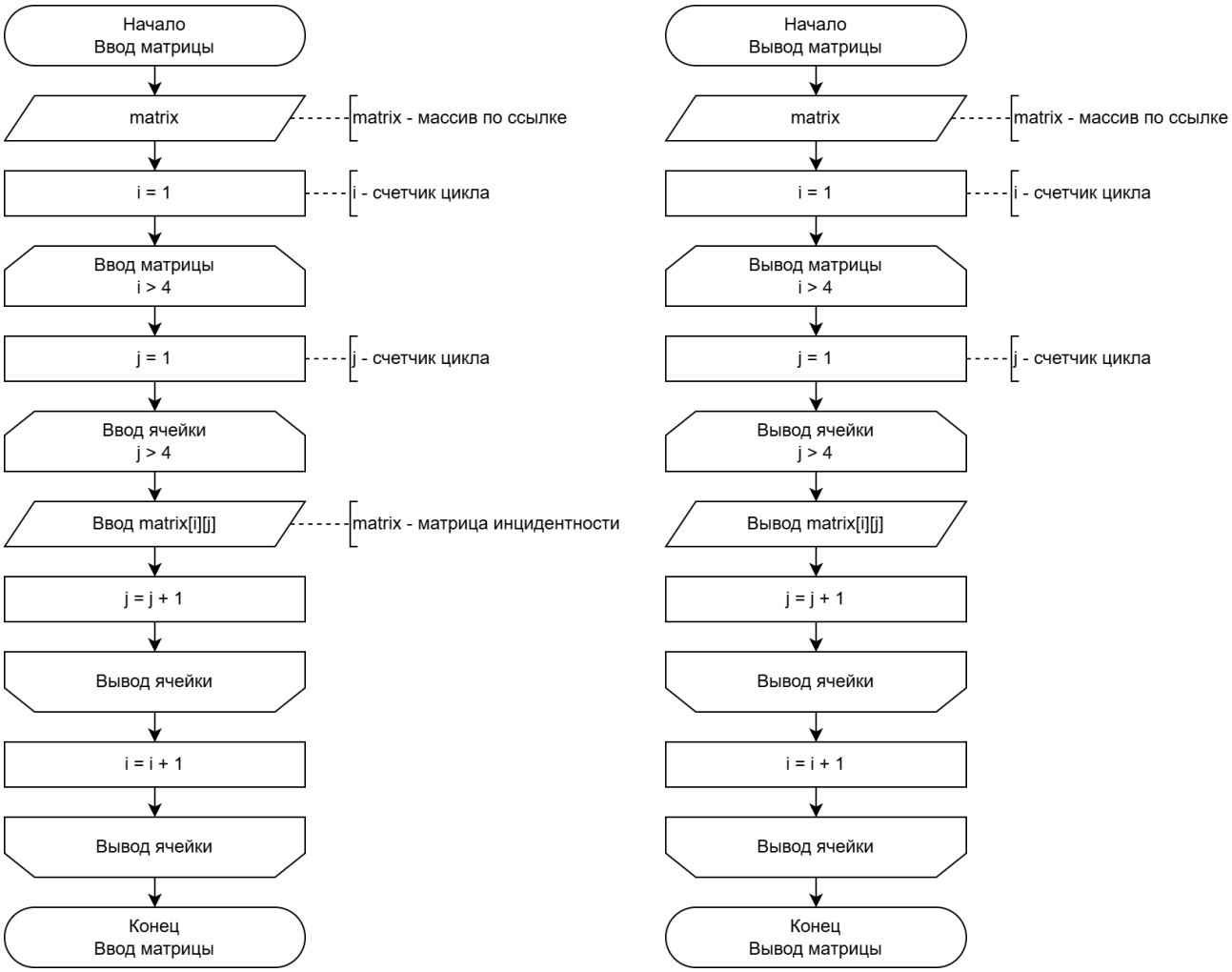

Рисунок 2 – Схемы алгоритмов ввода и вывода матрицы

На рисунке 3 представлена схема подпрограммы перевода матрицы инцидентности в матрицу смежности.

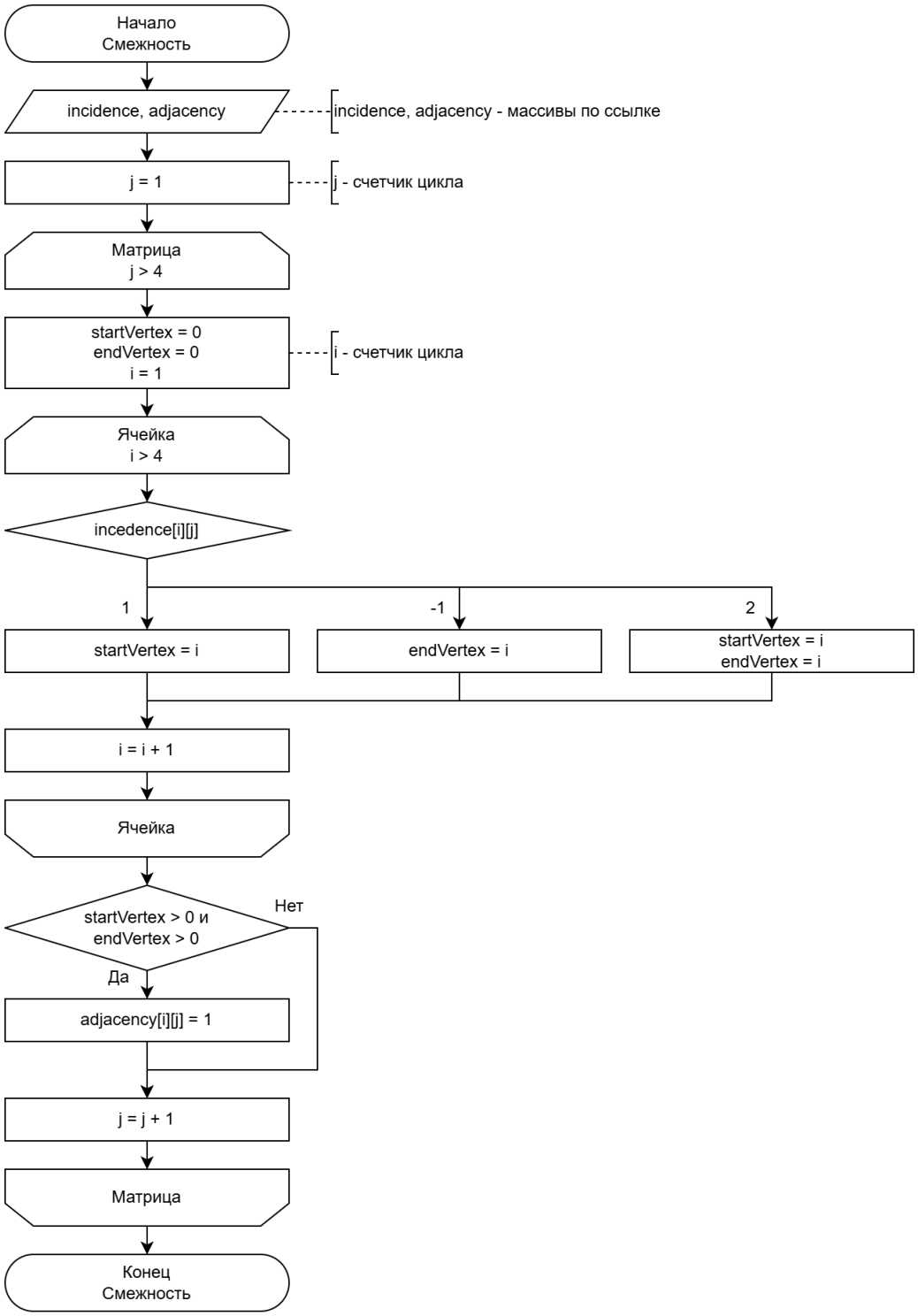

Рисунок 3 – Схема алгоритма перевода в матрицу смежности

На рисунке 4 представлены схемы подпрограмм умножения и сложения матриц.

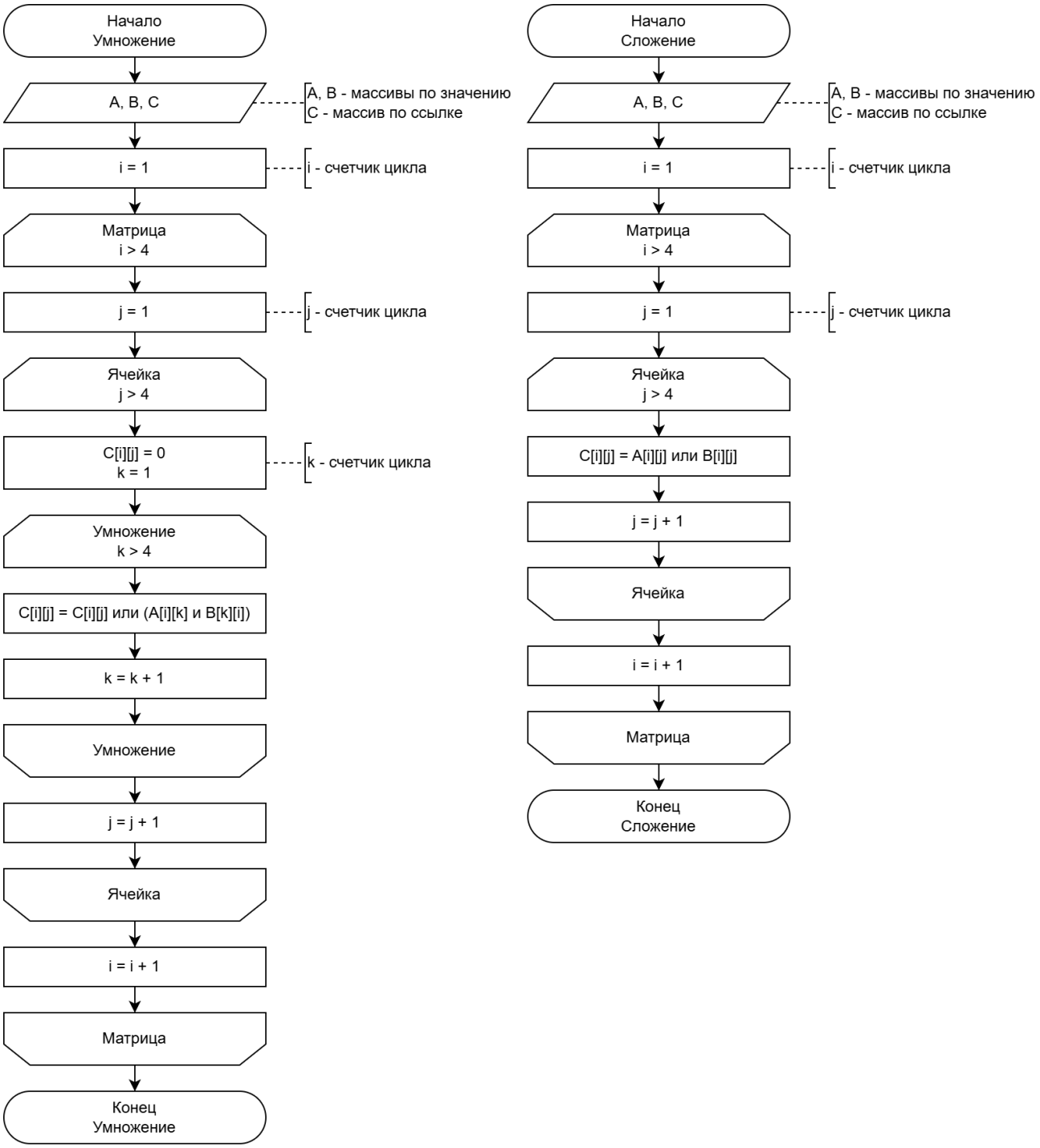

Рисунок 4 – Схемы алгоритмов умножения и сложения матриц

На рисунке 5 представлена схема подпрограммы формирования матрицы достижимости.

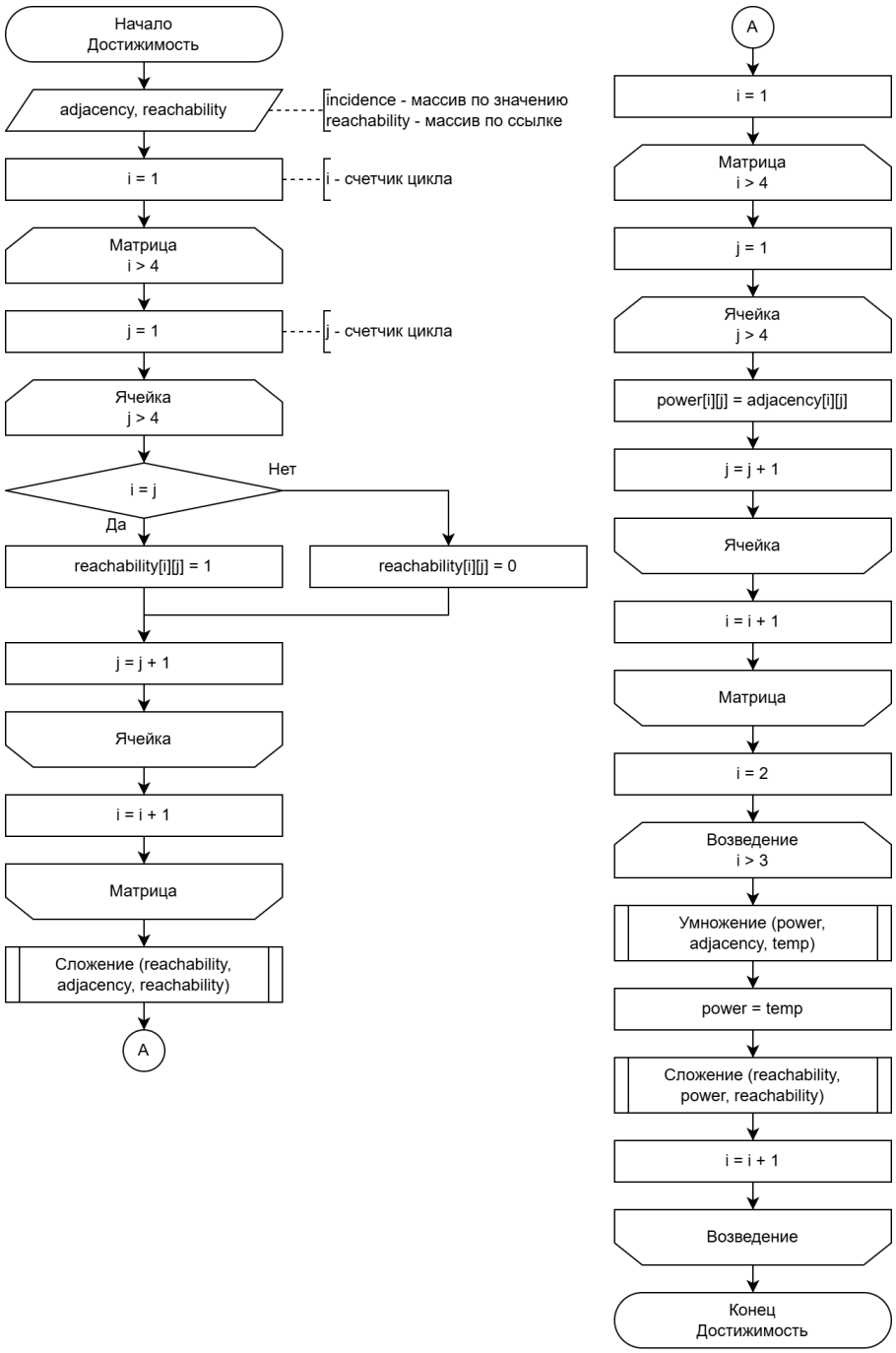

Рисунок 5 – Схема алгоритма формирования матрицы достижимости

На рисунке 6 представлена схемы подпрограмм транспонирования матрицы, формирования матрицы связности и проверки на связность.

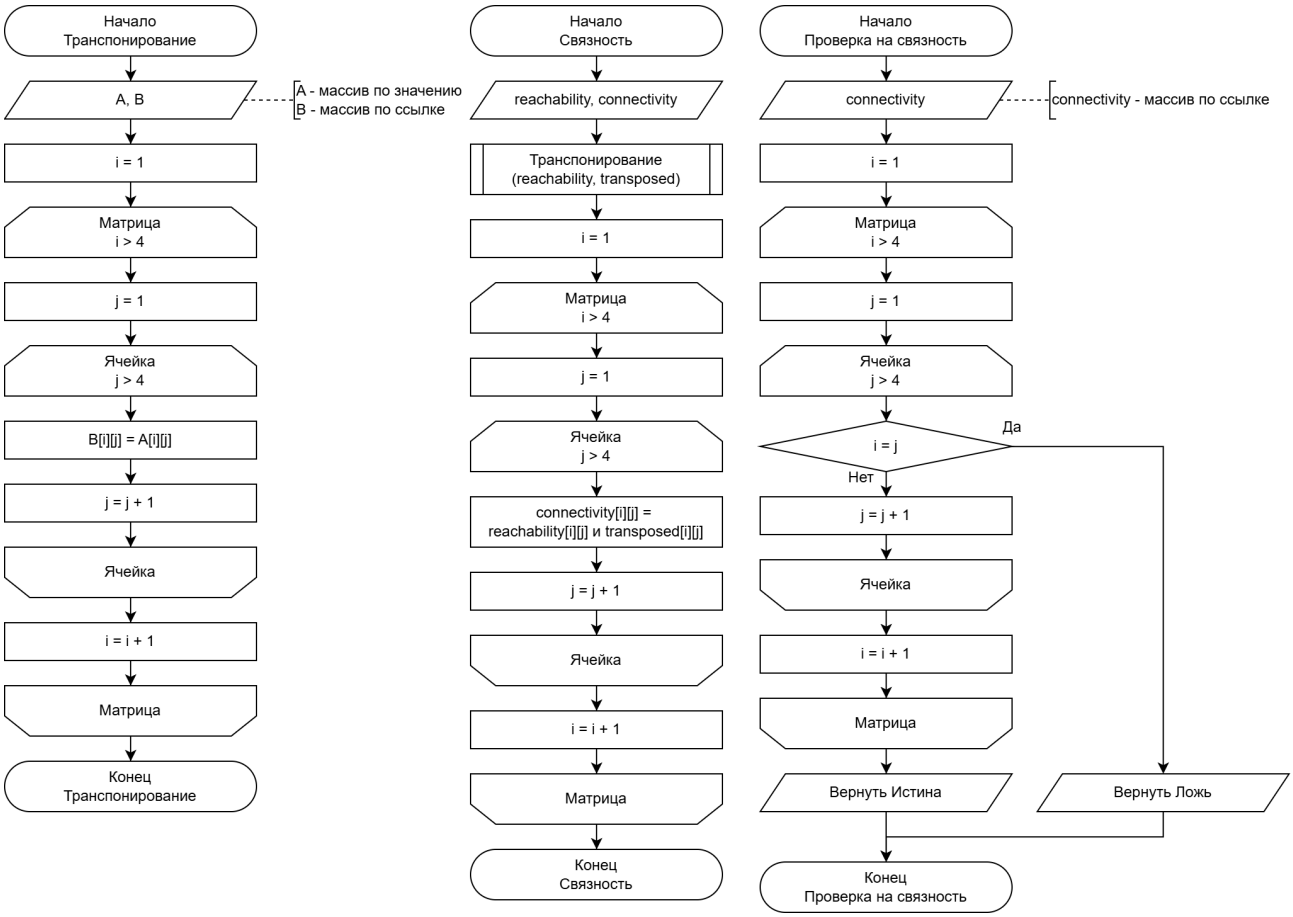

Рисунок 6 – Схема транспонирования матрицы, формирования матрицы связности, проверки на связность

На рисунке 7 представлена схема алгоритма основной программы.

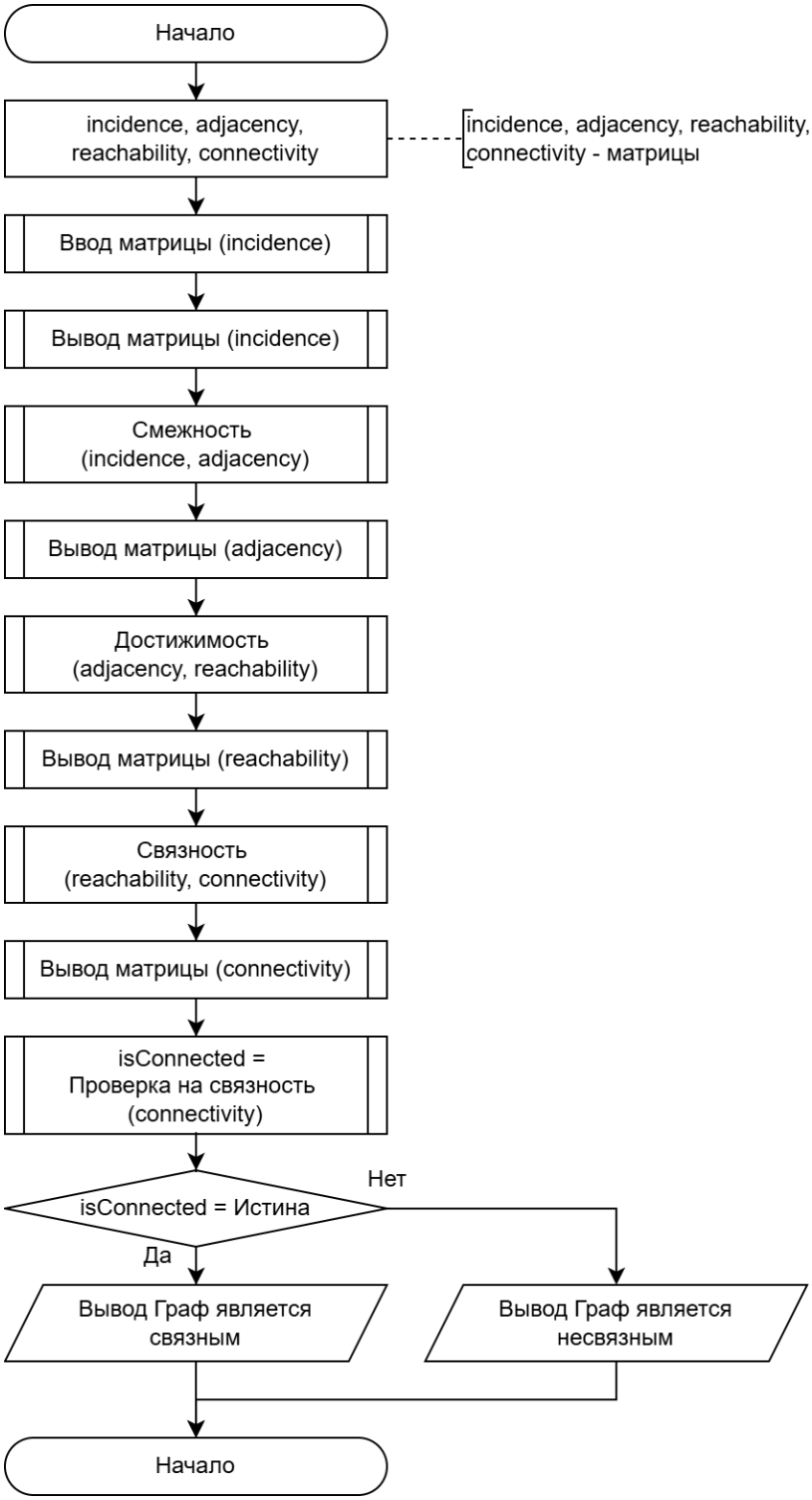

Рисунок 7 – Схема алгоритма основной программы

При разработке реализована программа, исходный код которой представлен ниже.

```
{$codepage UTF8}

const
  MAX_SIZE = 4;

type
  TMatrix = array[1..MAX_SIZE, 1..MAX_SIZE] of integer;

procedure PrintMatrix(var matrix: TMatrix);
var
  i, j: integer;
begin
  Write('  ');
  for i := 1 to MAX_SIZE do
    Write(' ', i, ' ');
  Writeln();

  for i := 1 to MAX_SIZE do
  begin
    Write(i, ' ');
    for j := 1 to MAX_SIZE do
      if matrix[i, j] = -1 then
        Write(matrix[i, j], ' ')
      else
        Write(' ', matrix[i, j], ' ');
    Writeln();
  end;
end;

procedure ReadIncidence(var incidence: TMatrix);
var
  fileInput: text;
  fileLine: string;
  i, j, k: integer;
begin
  Assign(fileInput, 'input.txt');
  Reset(fileInput);

  i := 1;
  while not Eof(fileInput) do
```

```
  begin
    Readln(fileInput, fileLine);

    j := 1;
    for k := 1 to Length(fileLine) do
      if (fileLine[k] <> ' ') and (fileLine[k] <> '-') then
      begin
        if fileLine[k - 1] = '-' then
          incidence[i, j] := -1
        else if fileLine[k] = '1' then
          incidence[i, j] := 1
        else if fileLine[k] = '2' then
          incidence[i, j] := 2;
        j := j + 1;
      end;
    i := i + 1;
  end;

  Close(fileInput);
end;

procedure ConvertToAdjacency(var incidence, adjacency: TMatrix);
var
  startVertex, endVertex, i, j: integer;
begin
  for j := 1 to MAX_SIZE do
  begin
    startVertex := 0;
    endVertex := 0;

    for i := 1 to MAX_SIZE do
      if incidence[i, j] = 1 then
        startVertex := i
      else if incidence[i, j] = -1 then
        endVertex := i
      else if incidence[i, j] = 2 then
      begin
        startVertex := i;
        endVertex := i;
      end;

    if (startVertex > 0) and (endVertex > 0) then
      adjacency[startVertex, endVertex] := 1;
```

```
  end;
end;

procedure MultiplyMatrix(const A, B: TMatrix; var C: TMatrix);
var
  i, j, k: integer;
begin
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
    begin
      C[i, j] := 0;
      for k := 1 to MAX_SIZE do
        C[i, j] := C[i, j] or (A[i, k] and B[k, j]);
    end;
end;

procedure AddMatrix(const A, B: TMatrix; var C: TMatrix);
var
  i, j: integer;
begin
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      C[i, j] := A[i, j] or B[i, j];
end;

procedure ConvertToReachability(const adjacency: TMatrix;
  var reachability: TMatrix);
var
  temp, power: TMatrix;
  i, j: integer;
begin
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      if i = j then reachability[i, j] := 1
      else reachability[i, j] := 0;

  AddMatrix(reachability, adjacency, reachability);

  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      power[i, j] := adjacency[i, j];

  for i := 2 to MAX_SIZE - 1 do
```

```
  begin
    MultiplyMatrix(power, adjacency, temp);
    power := temp;
    AddMatrix(reachability, power, reachability);
  end;
end;

procedure TransposeMatrix(const A: TMatrix; var B: TMatrix);
var
  i, j: integer;
begin
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      B[j, i] := A[i, j];
end;

procedure ConvertToConnectivity(const reachability: TMatrix;
  var connectivity: TMatrix);
var
  transposed: TMatrix;
  i, j: integer;
begin
  TransposeMatrix(reachability, transposed);

  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      connectivity[i, j] := reachability[i, j] and transposed[i, j];
end;

function IsConnected(const connectivity: TMatrix): boolean;
var
  i, j: integer;
begin
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      if connectivity[i, j] = 0 then
        Exit(false);
  IsConnected := true;
end;

var
  incidence, adjacency, reachability, connectivity: TMatrix;
```

```
begin

  ReadIncidence(incidence);
  Writeln(#10, 'Матрица инцидентности', #10);
  PrintMatrix(incidence);

  ConvertToAdjacency(incidence, adjacency);
  Writeln(#10, 'Матрица смежности', #10);
  PrintMatrix(adjacency);

  ConvertToReachability(adjacency, reachability);
  Writeln(#10, 'Матрица достижимости', #10);
  PrintMatrix(reachability);

  ConvertToConnectivity(reachability, connectivity);
  Writeln(#10, 'Матрица связности', #10);
  PrintMatrix(connectivity);

  if (IsConnected(connectivity)) then
    Writeln(#10, 'Граф является связным')
  else
    Writeln(#10, 'Граф является несвязным');

  Readln;

end.
```

Экранная форма программы в виде консольного приложения представлена на рисунке 8.

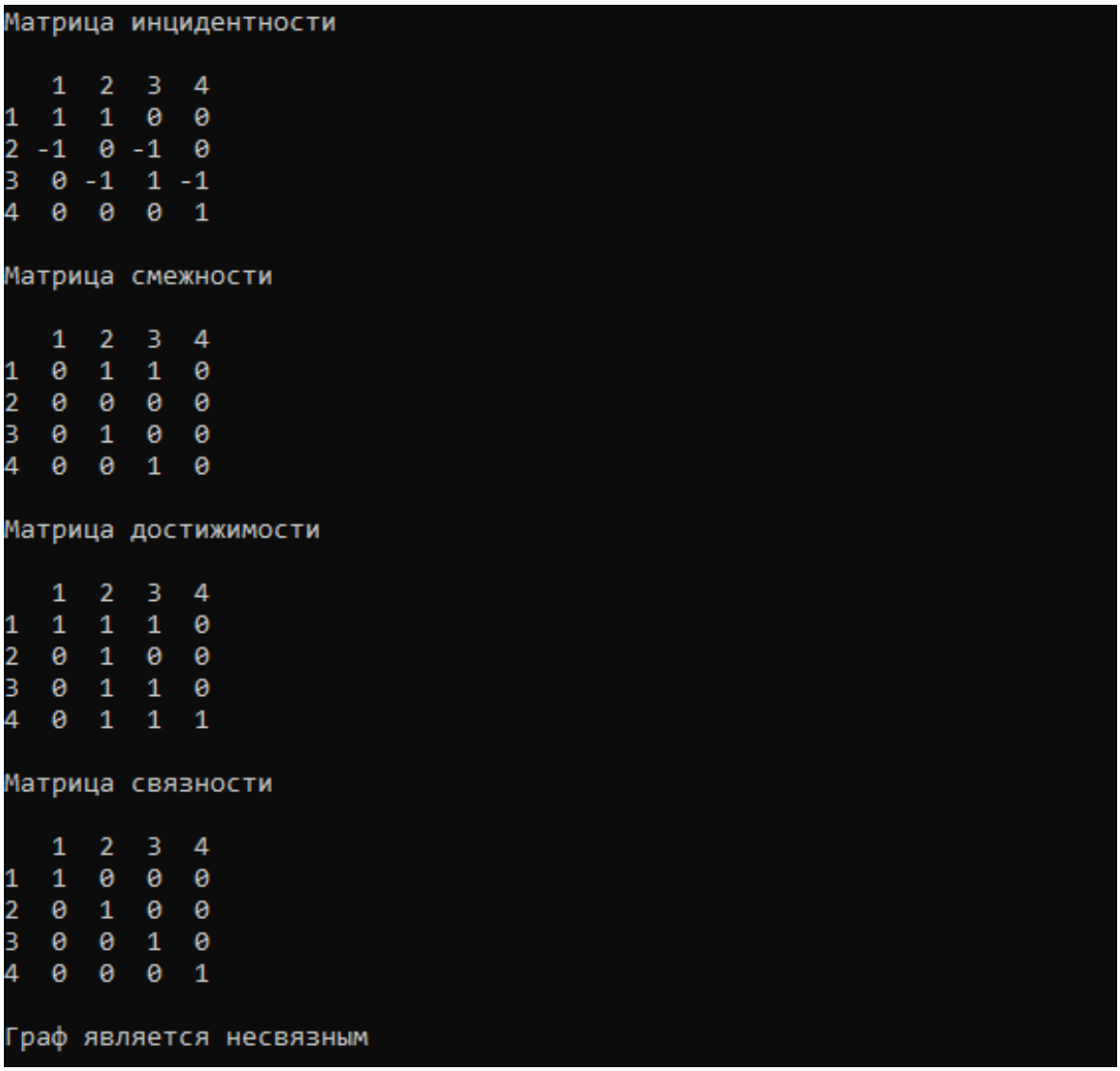

Рисунок 8 – Консольный интерфейс программы

Вывод

В процессе выполнения лабораторной работы, при решении предложенных задач, реализована программа на языке Паскаль, которая формирует матрицу связности, определяет является ли граф несвязным.