Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

(национальный исследовательский университет)

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)_____

О т ч е т

по лабораторной работе № 7 (11 вариант)

Название лабораторной работы: Подпрограммы. Средства отладки Delphi

Дисциплина: Основы программирования

Студент гр. ИУ6-13Б _____26.09.2022_ ___В.К. Залыгин__

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _А.М. Минитаева_

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Цель работы

Изучить принципы работы подпрограмм (функций и процедур), а также принципы работы средств отладки в среде Lazarus. Сравнить функции и процедуры в ЯП Pascal.

Задание

Решить задачу, используя процедуру или функцию. Выбор обосновать. На примере полученной программы продемонстрировать умение:

1) назначать точку останова;
2) выполнить программу по шагам с заходом в процедуры и без захода;
3) определять значения переменных на конкретном шаге.

Даны три матрицы разных порядков. Найти сумму их наименьших элементов (считая, что в каждой матрице такой элемент единственный).

Проект программы

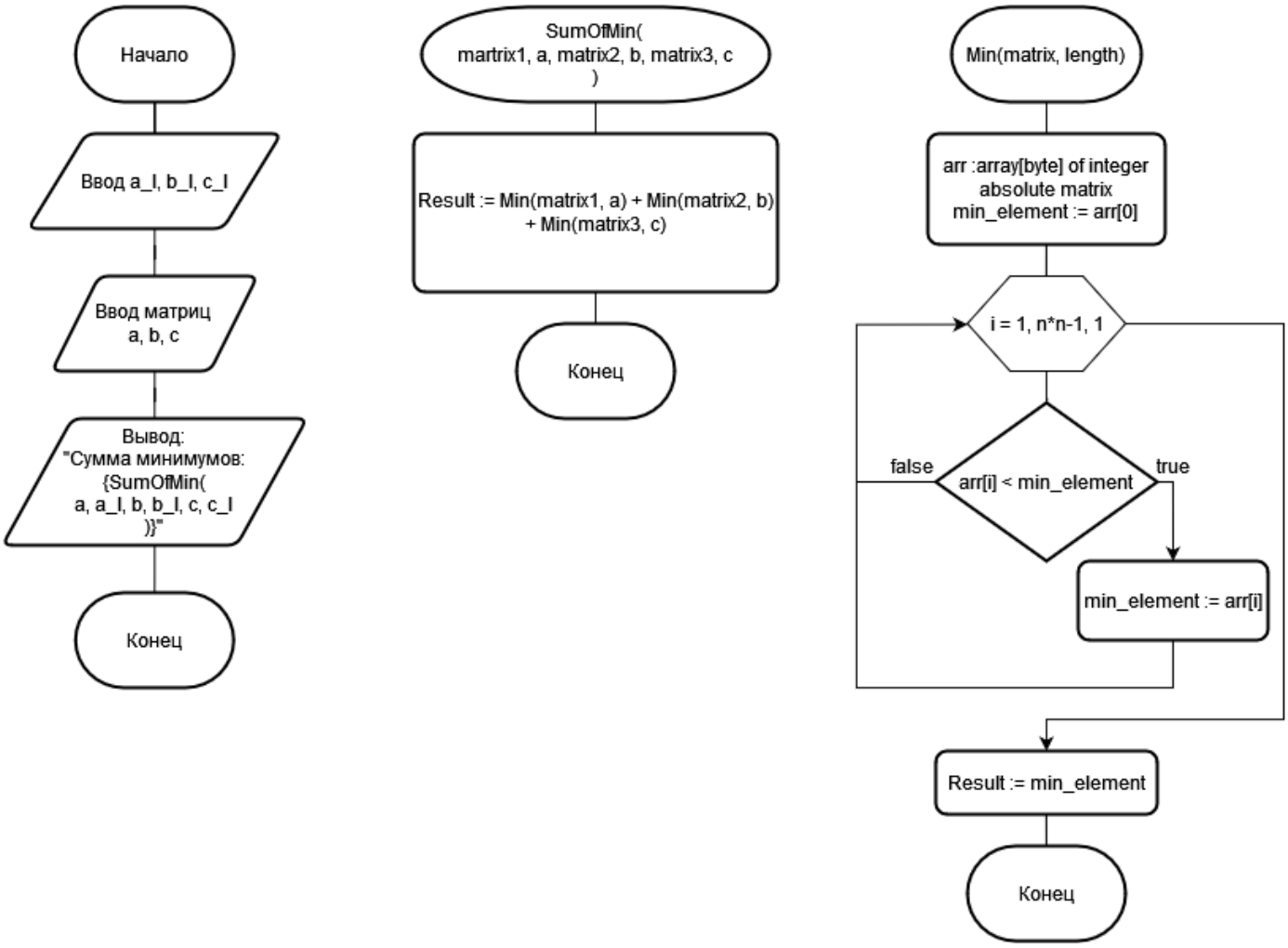

Рисунок 1 - схема алгоритма

Текст программы

```
program lab7;

// Функция нахождения минимального элемента матрицы с произвольным размером.
// matr - матрица, n - порядок матрицы
// Возвращает целое число - минимальный элемент матрицы.
function Min(const matr; n :integer) :integer;
var min_element, i :integer;
    arr :array[byte] of integer absolute matr;
begin
  min_element := arr[0];
  for i := 0 to n*n-1 do
    if arr[i] < min_element then
      min_element := arr[i];
  Result := min_element;
end;

// Функция получения суммы минимумов 3 матриц с произвольным размером.
// matrix1, matrix2, matrix3 - матрицы 1, 2, 3 соответственно.
// a, b, c - длины матриц 1, 2, 3 соответственно.
// Возращает целое число - сумму минимумов.
function SumOfMin(const matrix1; a :integer;
                  const matrix2; b :integer;
                  const matrix3; c :integer) :integer;
begin
  Result := Min(matrix1, a) + Min(matrix2, b) + Min(matrix3, c);
end;

// Процедура для задания матрицы с произвольным размером.
// matr - матрица, n - размер матрицы
procedure InputMatrix(var matr; n :integer);
var i, j :integer;
    arr :array[byte] of integer absolute matr;
begin
  for i := 0 to n-1 do
    for j := 0 to n-1 do
      Read(arr[i + j*n]);
  ReadLn;
end;
```

Рисунок 2 - код программы часть 1

```
type matrix = array[byte, byte] of integer;
// a, b, c - матрицы 1, 2, 3 соответственно.
var a, b, c :matrix;
// a_1, b_1, c_1 - порядки матриц 1, 2, 3 соответственно.
    a_1, b_1, c_1 :integer;
begin
  Write('Введите порядки матриц: >');
  ReadLn(a_1, b_1, c_1);
  WriteLn('Введите матрицу 1:');
  InputMatrix(a, a_1);
  WriteLn('Введите матрицу 2:');
  InputMatrix(b, b_1);
  WriteLn('Введите матрицу 3:');
  InputMatrix(c, c_1);

  WriteLn('Сумма минимумов: ', SumOfMin(a, a_1, b, b_1, c, c_1));
  ReadLn;
end.
```

Рисунок 3 - код программы часть 2

Демонстрация средств отладки Lazarus

Назначим несколько точек останова:

```
40  type matrix = array[byte, byte] of integer;
    // a, b, c - матрицы 1, 2, 3 соответственно.
    var a, b, c :matrix;
    // a_1, b_1, c_1 - порядки матриц 1, 2, 3 соответственно.
        a_1, b_1, c_1 :integer;
45  begin
      Write('Введите порядки матриц: >');
      ReadLn(a_1, b_1, c_1);
      WriteLn('Введите матрицу 1:');
      InputMatrix(a, a_1);
50    WriteLn('Введите матрицу 2:');
      InputMatrix(b, b_1);
      WriteLn('Введите матрицу 3:');
      InputMatrix(c, c_1);

55    WriteLn('Сумма минимумов: ', SumOfMin(a, a_1, b, b_1, c, c_1));
      ReadLn;
57  end.
```

Рисунок 4 - точки останова на вызовах процедуры InputMatrix

```
    // Функция нахождения минимального элемента матрицы с проивольным размером.
    // matr - матрица, n - порядок матрицы
5   // Возвращает целое число - минимальный элемент матрицы.
    function Min(const matr; n :integer) :integer;
    var min_element, i :integer;
        arr :array[byte] of integer absolute matr;
    begin
10    min_element := arr[0];
      for i := 0 to n*n-1 do
        if arr[i] < min_element then
          min_element := arr[i];
      Result := min_element;
15  end;
```

Рисунок 5 - точка остановы в функции Min

Для захода в процедуру нужно воспользоваться кнопкой «Шаг со входом»:

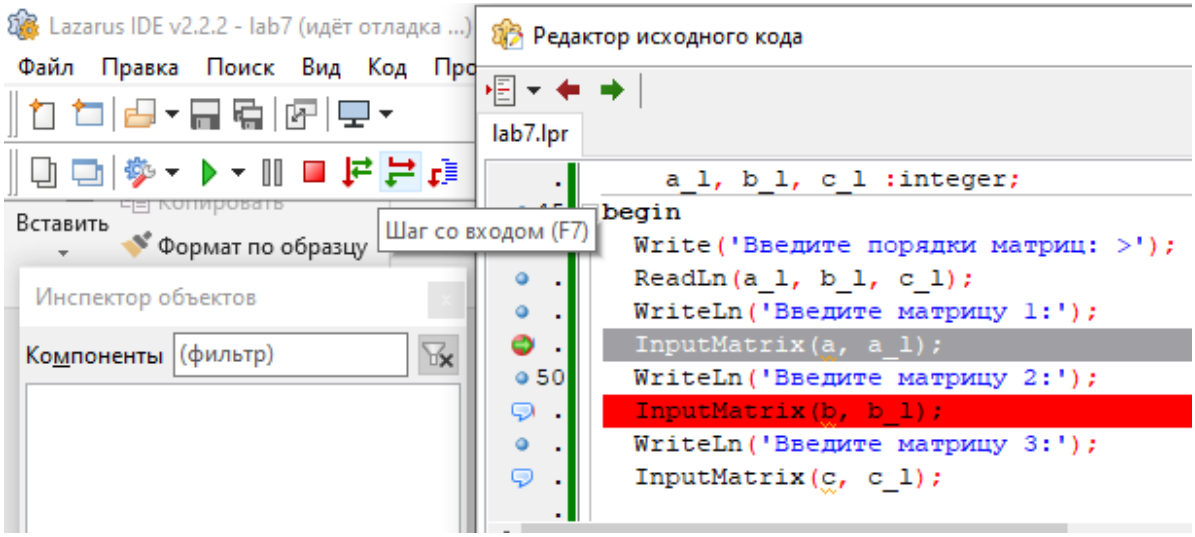

Рисунок 6 - поток управления блокируется отладчиком при попадании на точку останова

При нажатии на кнопку «Шаг со входом» отладчик «проваливается» в вызываемую процедуру:

```
// Процедура для задания матрицы с произвольным размером.
// matr - матрица, n - размер матрицы
procedure InputMatrix(var matr; n :integer);
var i, j :integer;
    arr :array[byte] of integer absolute matr;
begin
  for i := 0 to n-1 do
    for j := 0 to n-1 do
      Read(arr[i + j*n]);
  ReadLn;
end;
```

Рисунок 7 - пооперационное выполнение процедуры InputMatrix

Для немедленного выполнения процедуры нужно воспользоваться кнопкой «Шаг в обход»:

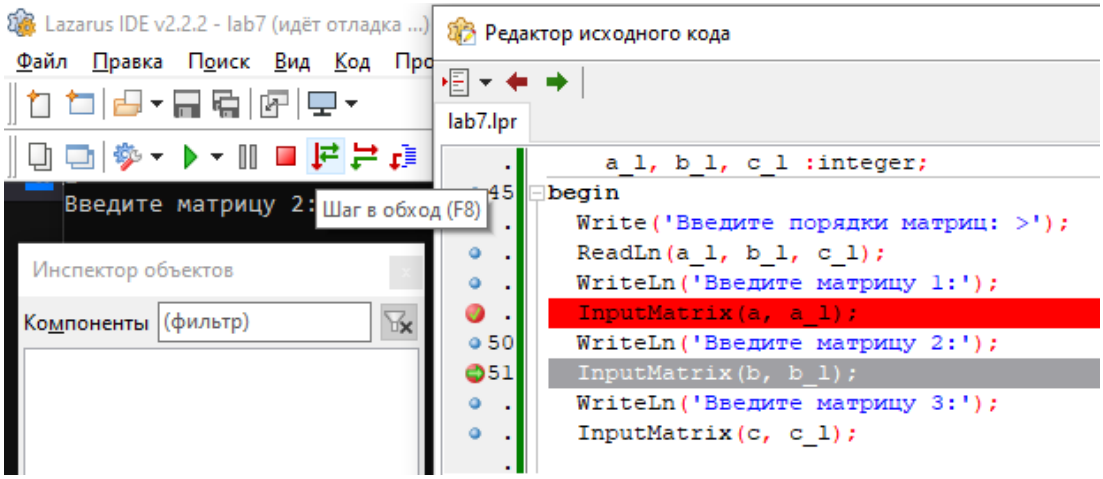

Рисунок 8 - для пропуска пооперационного выполнения можно использовать "Шаг в обход"

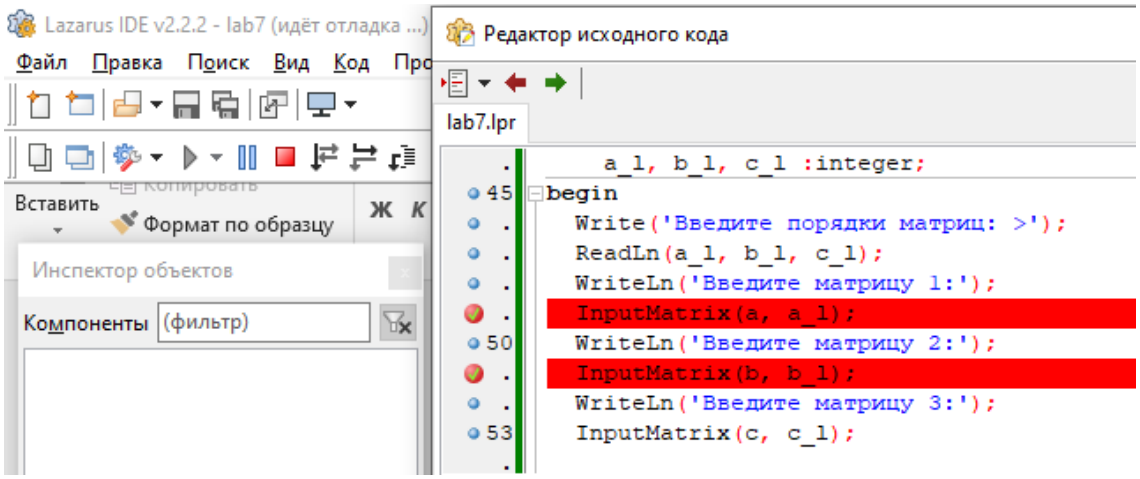

Рисунок 9 - отладчик разблокировал поток управления, давая процедуре выполниться без участия человека. На скрине процедура ещё не успела завершить работу, т.к. ожидает пользовательский ввод

Для просмотра значений переменных нужно навести курсор на название желаемой переменной или обратиться к окну «Локальные переменные»:

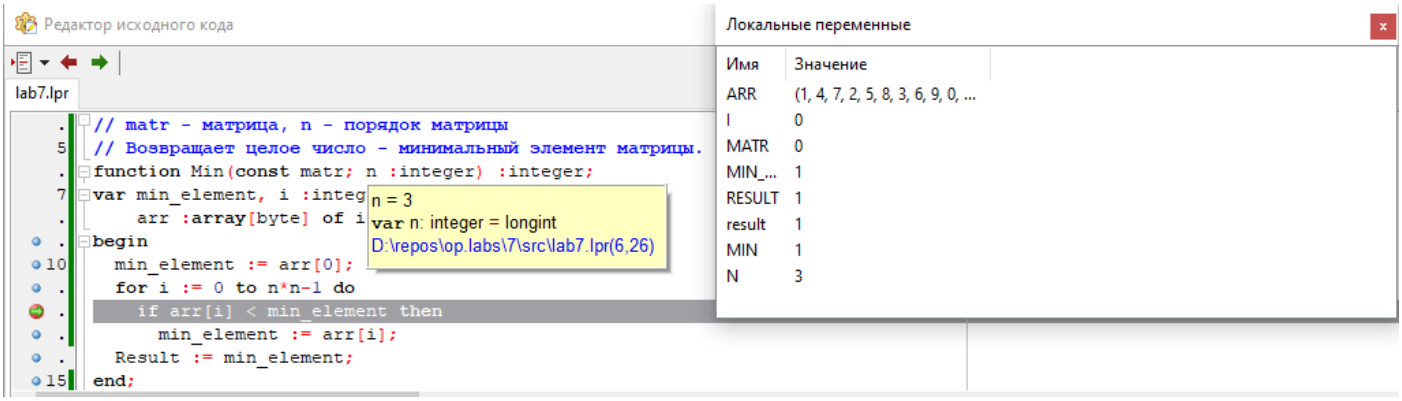

Рисунок 10 - просмотр значений локальных переменных во время исполнения

Тестовые данные

```
Введите порядки матриц (до 255):> 1 2 3
Введите матрицу 1:
100
Введите матрицу 2:
40 30
20 10
Введите матрицу 3:
9 8 7
6 5 4
3 2 1
Сумма минимумов: 111
```

Рисунок 11 - пример работы программы

Вывод

В результате были изучены принципы работы функций и процедур, принципы отладки в Lazarus, установлены различия между функциями и процедурами. На мой взгляд, для решения данной задачи предпочтительнее будет использование функций, чем процедур, т.к. функции могут возвращать значения, что удобнее для написания кода.