

ЦИКЛ С РАЗВЕТВЛЕНИЕМ

1. Использование массивов

Массив представляет собой переменную, для хранения значения которой отводится не одна (как для обычной переменной), а несколько ячеек памяти. Каждая ячейка отводится под отдельный элемент массива. Таким образом, массив является структурированной переменной в отличие от скалярной простой переменной. Все элементы массива имеют один и тот же тип. При необходимости выполнить некоторое действие над всеми элементами массива можно сослаться на массив как целое по его имени. Возможны также ссылки на его отдельные элементы. Каждый отдельный элемент массива определяется именем массива и значениями индексов. Например, ссылки $a(7)$ или $a1(2, 9)$ означают, что:

a – это имя одномерного массива (вектора) с одним индексом, элемент массива имеет значение индекса равное 7;

$a1$ – это имя двумерного массива (матрицы), элемент массива принадлежит 2 строке и 9 столбцу.

Как и простые переменные, массивы описываются с помощью инструкций `Dim`, `Static`, `Private` или `Public`. Разница в объявлении между скалярными переменными (т.е. не массивами) и массивами состоит в том, что для массива надо указывать его размер (количество ячеек, отводимых под массив). Массив с заданным размером называется фиксированным. Массив, объявление размера которого отложено, называется динамическим.

1.1. Объявление массива фиксированного размера

При объявлении фиксированного массива, кроме его имени и типа элементов, необходимо для каждого индекса указать его верхнюю границу, нижняя граница всегда равна 0.

В следующей строке программы массив фиксированного размера описывается как массив типа `Integer`, имеющий 11 строк и 11 столбцов:

```
Dim a1(10, 10) As Integer
```

Первый аргумент внутри скобок обозначает наибольший номер строки, а второй – наибольший номер столбца.

Как и при описании переменных, если тип при описании массива не задается, то массив будет иметь тип `Object`.

1.2. Описание динамического массива

Если массив описан как динамический, можно изменять его размер во время работы программы. Для описания динамического

массива используются инструкции `Static`, `Dim`, `Private` или `Public` с парой скобок, внутри которых помещаются запятые для обозначения размерности, если она больше 1:

```
Dim a2(), b(,) As Single
```

При выполнении программы (но только внутри подпрограмм и функций) можно переобъявлять динамический массив с помощью инструкции `ReDim`. Пусть, например, на уровне проекта или на уровне формы был объявлен динамический массив `a3`:

```
Dim a3(,) As Single
```

Тогда среди инструкций некоторой подпрограммы или функции можно записать:

```
ReDim a3(5, 10)
```

Далее этот массив может быть переопределен:

```
ReDim a3(7, 15)
```

С помощью инструкции `ReDim` можно изменять размер массива (число элементов), верхние границы индексов. Инструкцию `ReDim` можно применять для изменения динамического массива столько раз, сколько потребуется. Однако при каждом её применении данные, содержащиеся в массиве, теряются.

Инструкция `ReDim Preserve` может увеличить размер массива, сохраняя при этом его содержимое. В следующем примере показывается, как можно увеличить размер массива `a4` на 10 элементов без уничтожения текущих значений элементов массива. Пусть был объявлен динамический массив:

```
Dim a4() As Integer
```

Затем в программе установлен размер этого массива:

```
ReDim a4(n)
```

Далее в программе может быть, например, записано:

```
ReDim Preserve a4(n+10)
```

Использование зарезервированных слов `ReDim Preserve` вместе с динамическим массивом позволяет изменить только верхнюю границу и только последней размерности массива.

2. Логические операции

Логические операции применяются к данным и выражениям логического типа (`Boolean`). Мы рассмотрим три логические операции из имеющихся в VB.NET: `And` («И»), `Or` («ИЛИ») и `Not` (отрицание).

Операция `Not` имеет следующий синтаксис:

Not Операнд

Операнд, имеющий логический тип, – это отношение, переменная или функция логического типа или результат логической операции. Результат логического отрицания имеет значение противоположное значению ее операнда, что показывает таблица:

| Значение операнда | Значение операции Not Операнд |
|-------------------|-------------------------------|
| True | False |
| False | True |

Операция And имеет два операнда:

Операнд 1 **And** Операнд 2

Результат операции «И» определяет таблица:

| Значение операнда 1 | Значение операнда 2 | Значение операции Операнд 1 And Операнд 2 |
|---------------------|---------------------|---|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

Операция And имеет значение True только тогда, когда оба операнда имеют значение True.

Операция Or также имеет два операнда:

Операнд 1 **Or** Операнд 2

Результат операции «ИЛИ» дан в таблице:

| Значение операнда 1 | Значение операнда 2 | Значение операции Операнд 1 Or Операнд 2 |
|---------------------|---------------------|--|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

Операция Or имеет значение True, если хотя бы один операнд имеет значение True.

Логические операции имеют более низкий приоритет, чем арифметические операции.

Таблица приоритетов:

| Приоритет | Операция |
|-----------|-------------------------|
| 1-7 | Арифметические операции |
| 8 | <, >, <=, >=, =, <> |
| 9 | Not |
| 10 | And |
| 11 | Or |

В качестве примера запишем логическое выражение, имеющее значение True, когда выполняется двойное неравенство $a < x < b$.

Это выражение следует записать так: $a < x$ And $x < b$

3. Программирование разветвлений

Наиболее часто для программирования разветвлений используется инструкция `If...End If` (блок-схема реализуемого алгоритма на рис. 1).

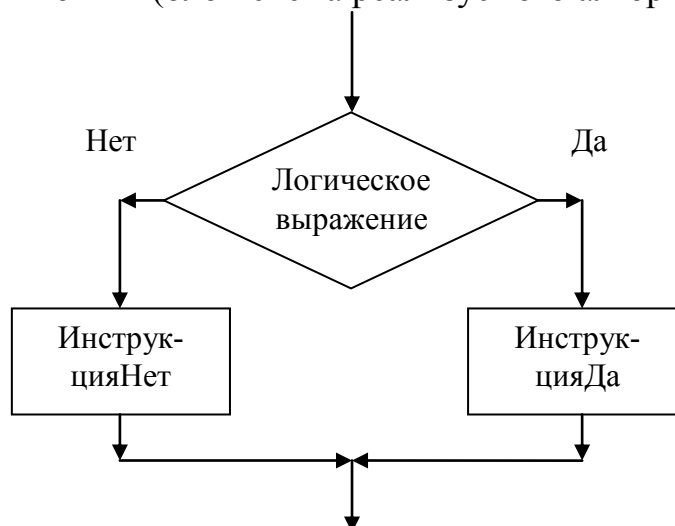


Рис. 1. Разветвление

Эта инструкция может иметь однострочный или блочный синтаксис.

3.1. Однострочный синтаксис

If ЛВ **Then** ИнструкцияДа [**Else** ИнструкцияНет]

Здесь If (если), Then (то) и Else (иначе) - зарезервированные слова. ЛВ – логическое выражение, которое может принимать значение либо True, либо False.

Функционирование этой инструкции происходит относительно просто. Если ЛВ равно True, то выполняется ИнструкцияДа, а ИнструкцияНет пропускается. Если же ЛВ равно False, то выполняется ИнструкцияНет, если таковая имеется, а ИнструкцияДа пропускается.

Например:

```
If a = 7 Then Beep
```

```
If x < 9 Then MsgBox("False!") Else MsgBox("True!")
```

Первая строка задает подачу звукового сигнала, если переменная a равна 7. Во второй строке в окне функции MsgBox выводится текст False!, если значение переменной x меньше 9. В противном случае выводится текст True!.

Существенно, что после слова Then, а также после слова Else может находиться несколько отделенных друг от друга двоеточием простых инструкций (в них не могут входить многострочные инструкции, например инструкция цикла). В этом случае такая последовательность инструкций является

одной группой, которая либо выполняется, либо пропускается в зависимости от значения ЛВ.

Пример:

```
Dim a As Boolean = False, b As Integer
If a Then b = b + 1 : b = b + 1 : b = b + 1
Console.WriteLine(b)
```

В этом примере переменная `b` получит значение 0. Если бы переменная `a` была равна `True`, то переменная `b` получила бы значение 3.

3.2. Блочный синтаксис

Блочная структура с `ElseIf` позволяет анализировать значения нескольких логических выражений.

```
If ЛВ Then
    [ИнструкцииДа]
[ElseIf ЛВi Then
    Инструкцииi]
[Else]
    [ИнструкцииНет]
End If
```

Если в зависимости от значения ЛВ необходимо выполнить не одну, а несколько инструкций (возможно не простых), следует использовать блочный синтаксис. Строка с `ElseIf` может появляться несколько раз, но только до строки `Else`.

Если значение ЛВ равно `True`, то выполняются ИнструкцииДа, а все остальные пропускаются.

Если же значение ЛВ равно `False`, то ИнструкцииДа пропускаются и вычисляется значение логического выражения ЛВ_i очередной *i*-ой строки с `ElseIf`, расположенной ниже. Если значение ЛВ_i равно `True`, то выполняются Инструкции_i, а все остальные пропускаются. Если же значение ЛВ_i равно `False`, то Инструкции_i пропускаются и вычисляется значение логического выражения ЛВ_i очередной строки с `ElseIf`, расположенной ниже.

Если же ни одно из ЛВ_i не имело значения `True` то выполняются ИнструкцииНет.

Пример 1

```
If a = 7 Then
    Beep
End If
```

В этом примере сигнал прозвучит, если только значение переменной `a` равно 7.

Пример 2

```
If Name = "Иванов" Then
    MsgBox("Ваша карточка удерживается!")
```

```
Else
    MsgBox("Получите деньги, пожалуйста!")
End If
```

В этом примере будет выведено Ваша карточка удерживается!, если переменная Name имеет значение Иванов. В противном случае будет выведено Получите деньги, пожалуйста!.

Пример 3

```
If Обращение = 1 Then
    MsgBox("Здравствуйте, сударь")
ElseIf Обращение = 2 Then
    MsgBox("Здравствуйте, сударыня")
ElseIf Обращение = 3 Then
    MsgBox("Здравствуйте, сударыни и судари")
Else
    MsgBox("Здравствуйте")
End If
```

Результат работы этого кода поясняет таблица:

| Значение переменной Обращение | Результат вывода |
|-------------------------------|---------------------------------|
| 1 | Здравствуйте, сударь |
| 2 | Здравствуйте, сударыня |
| 3 | Здравствуйте, сударыни и судари |
| Любое другое | Здравствуйте |

4. Задание

Разработайте проект для решения задачи, сформулированной в соответствующем варианте задания. Предусмотрите вывод исходных данных для контроля правильности выполнения ввода. Подготовьте тестовые наборы исходных данных и рассчитайте для них результаты работы программы. Тестовых наборов данных может требоваться больше одного. Вместе они должны доказать правильность работы всех ветвей алгоритма.

Отчет по выполненной работе должен включать: титульный лист (см. приложение 5), условие задания вместе с условием варианта, описание применяемых данных, укрупненную блок-схему алгоритма, программный код, тесты и расчетные результаты их выполнения.

Варианты задания.

1. Найти сумму и число тех элементов заданного вектора X_1, X_2, \dots, X_n которые больше заданной величины P , но меньше другой заданной величины T ($P < T$).

2. Подсчитать по отдельности суммы $C1$ и $C2$ и количества $M1$ и $M2$ отрицательных и положительных элементов заданного вектора X_1, X_2, \dots, X_n .

3. Найти произведение, а также количество тех элементов заданного вектора X_1, X_2, \dots, X_n , которые положительны и для которых в то же время выполняется неравенство $\sin(X_i) > 0,5$.

4. Найти сумму и общее количество тех элементов заданного вектора X_1, X_2, \dots, X_n , абсолютная величина которых отличается от 1 не более чем на заданную величину H .

5. Для заданного вектора X_1, X_2, \dots, X_n найти среднее арифметическое CX положительных элементов, имеющих четные номера.

6. При заданных элементах вектора X_1, X_2, \dots, X_n найти по отдельности суммы $C1, C2$ и количества $M1, M2$ элементов, значения которых соответственно больше 1 или меньше -1 .

7. Для заданных целочисленных векторов X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n , проверяя на равенство элементы пар $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$, подсчитать число случаев равенства пары элементов, а также среднее арифметическое элементов X_1, X_2, \dots, X_n .

8. Вычислить сумму и число тех элементов заданного вектора X_1, X_2, \dots, X_n , значения которых меньше 10 или находятся в пределах от 20 до 30 (включая указанные границы).

9. Для заданной величины A и заданных значениях элементов векторов X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n , определить число произведений $X_i * Y_i$, удовлетворяющих условию $X_i * Y_i \leq A$, и сумму таких произведений.

10. Найти среднее арифметическое тех элементов заданного вектора X_1, X_2, \dots, X_n , значения которых не превышают X_1 , включая и сам элемент X_1 ; найти также среднее арифметическое всех элементов данного вектора.

11. Найти $CX * CY$, где CX и CY - средние арифметические положительных элементов заданных векторов X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n , соответственно.

12. Найти сумму и число тех элементов заданного вектора X_1, X_2, \dots, X_n , которые больше элемента с тем же номером из другого заданного вектора Y_1, Y_2, \dots, Y_n , а также положительны.

13. При заданных абсциссах X_1, X_2, \dots, X_n и ординатах Y_1, Y_2, \dots, Y_n n точек на плоскости XOY определить, у какого числа этих точек положительна как абсцисса, так и ордината, а также найти среднюю ординату всех прочих точек из числа заданных.

14. При заданных значениях переменных A и B подсчитать, сколько кругов с заданными радиусами R_1, R_2, \dots, R_n имеют большую площадь, чем прямоугольник со сторонами A и B .

15. При заданных абсциссах X_1, X_2, \dots, X_n и ординатах Y_1, Y_2, \dots, Y_n n точек на плоскости XOY подсчитать количество точек, ординаты которых больше абсциссы, и сумму расстояний от начала координат для всех заданных точек.

16. При заданных значениях $A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_n$ и C_1, C_2, \dots, C_n для каждой из n троек вида (A_i, B_i, C_i) проверить, может ли быть построен треугольник со сторонами A_i, B_i, C_i , при этом подсчитать число треугольников и сумму их периметров.

17. При заданных абсциссах X_1, X_2, \dots, X_n и ординатах Y_1, Y_2, \dots, Y_n n точек на плоскости XOY подсчитать, сколько из них находится в пределах круга заданного радиуса R с центром в начале координат, а также среднее арифметическое расстояние от начала координат до всех заданных точек.

18. При заданных значениях X_t, Y_t , абсциссах X_1, X_2, \dots, X_n и ординатах Y_1, Y_2, \dots, Y_n n точек плоскости XOY определить, в каком числе случаев расстояние между одной из этих точек и точкой с координатами X_t, Y_t превышает заданную величину B , и найти средние координаты для заданной совокупности точек, исключая точку (X_t, Y_t) .

19. Найти среднее арифметическое не равных нулю элементов заданного вектора X_1, X_2, \dots, X_n и подсчитать число элементов с неотрицательными значениями (включая и элементы, равные нулю).

20. Изменить значения всех положительных элементов заданного вектора X_1, X_2, \dots, X_n делением каждого из них на его номер в массиве и подсчитать число отрицательных элементов данного вектора.

21. При заданных значениях X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n заменить значение каждого неположительного элемента вектора X_1, X_2, \dots, X_n абсолютной величиной соответствующего (по номеру) элемента вектора Y_1, Y_2, \dots, Y_n и подсчитать количество замен.

22. При заданных значениях X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n получить вектор T_1, T_2, \dots, T_n , элементы которого получают значения по правилу: $T_i = X_i$, если $X_i > Y_i$, иначе $T_i = Y_i$. Подсчитать, сколько элементов T_i получило значения X_i .

23. При заданных значениях X_1, X_2, \dots, X_n найти вектор элементов Y_1, Y_2, \dots, Y_n по правилу:

$$Y_k = 1 - \sin(X_k), \text{ если } X_k > 0;$$

$$Y_k = 1 - \cos(X_k), \text{ если } X_k \leq 0.$$

При этом подсчитать число неположительных элементов X_k .

24. В заданном векторе X_1, X_2, \dots, X_n заменить значения отрицательных элементов их абсолютными величинами, при этом подсчитать число элементов, равных нулю.

25. Подсчитать, сколько среди заданных элементов вектора X_1, X_2, \dots, X_n имеют отрицательные значения, и изменить значение каждого положительного элемента делением его на значение первого элемента.

26. Найти вектор элементов Y_1, Y_2, \dots, Y_n на основе заданного целочисленного вектора X_1, X_2, \dots, X_n , используя правило:

$$Y_k = 0, \text{ если } X_k \leq 0;$$

$$Y_k = X_k, \text{ если } X_k > 0.$$

При этом подсчитать число элементов X_i , равных нулю.

27. В заданном векторе X_1, X_2, \dots, X_n изменить значения всех положительных элементов, умножив их значения на 5, а отрицательные элементы уменьшить вдвое, при этом подсчитать количество элементов, абсолютная величина которых не превышает 1.

28. При заданных значениях X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n заменить значения тех элементов вектора X_1, X_2, \dots, X_n , для которых выполняется условие $|X_i - Y_i| \leq 10^{-2}$, значениями элементов Y_i , а также подсчитать число произведенных замен.

29. При заданных значениях X_1, X_2, \dots, X_n и Y_1, Y_2, \dots, Y_n заменить значение каждого элемента среди Y_1, Y_2, \dots, Y_n новым значением, определяемым по правилу:

$$Y_k = X_k - Y_k, \text{ если } X_k \geq 0;$$

$$Y_k = Y_k - X_k, \text{ если } X_k < 0.$$

30. При заданных значениях X_1, X_2, \dots, X_n ; Y_1, Y_2, \dots, Y_n и Z_1, Z_2, \dots, Z_n получить новые значения этих элементов, последовательно рассматривая тройки (X_i, Y_i, Z_i) : X_i следует задать наименьшее из этих трех значений, Z_i - наибольшее, а Y_i - оставшееся значение данной тройки.

5. Пример разработки проекта

5.1. Условие

При заданных значениях элементов векторов a_1, a_2, \dots, a_m , b_1, b_2, \dots, b_m и некоторого порога h получить значения c_1, c_2, \dots, c_m , последовательно рассматривая тройки (h, a_i, b_i) . Элементам вектора c_i следует задать значение по правилу:

- a_i , если a_i наибольшее из этих трех значений;
- b_i , если b_i наибольшее из этих трех значений;
- h при любых других соотношениях значений элементов в тройке (h, a_i, b_i) .

Также требуется определить по отдельности, сколько элементов вектора a и вектора b стали значениями элементов вектора c .

5.2. Выбор данных

Исходные данные:

m – количество элементов векторов, переменная целого типа;

h – порог, переменная ординарной точности с дробной частью;

a, b – одномерные векторы, переменные ординарной точности с дробной частью.

Результаты:

c – искомый одномерный вектор, ординарной точности с дробной частью;

k_a – количество элементов вектора a , вошедших в вектор c , переменная целого типа;

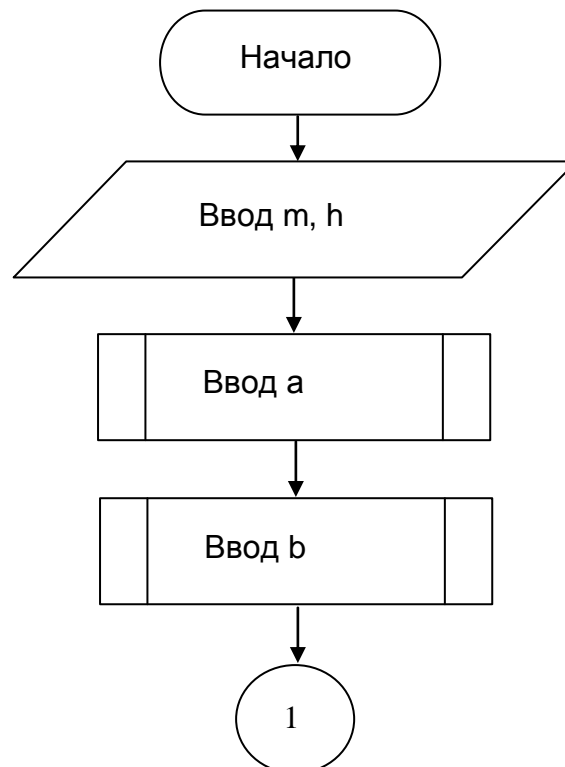
k_b – количество элементов вектора b , вошедших в вектор c , переменная целого типа.

Промежуточные:

i – параметр цикла, переменная целого типа.

5.3. Разработка алгоритма

На рис. 2 приведена укрупненная блок-схема алгоритма, на которой укрупненными блоками обозначены алгоритмы ввода значений векторов a и b (с номера 1 по номер m), а также алгоритм вывода значений элементов вектора c (с номера 1 по номер m).



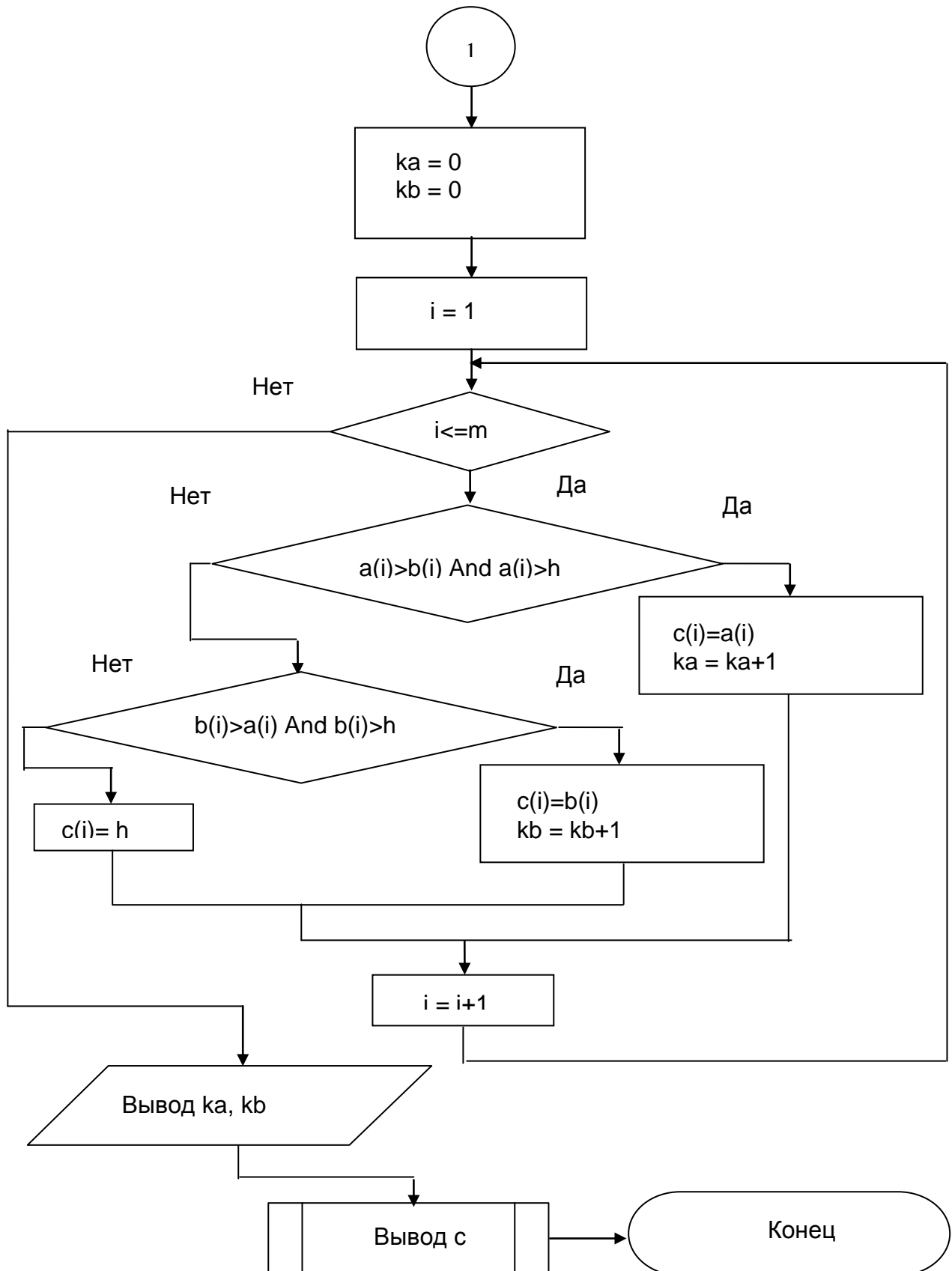


Рис. 2. Укрупненная блок – схема алгоритма

В тех случаях, когда блок-схема не размещается на одной странице, применяют соединители линий, вынужденно разрываемых при переходе на другую страницу. Соответствие соединителей можно устанавливать с помощью номеров.

С точки зрения структуры алгоритм включает 3 базовые структуры следования.


В первой из них осуществляется ввод значений переменных m и h , а также обнуление переменных – счетчиков, ka и kb .

Вторая структура следования содержит цикл, тело которого в свою очередь содержит разветвление, одна из ветвей которого также является разветвлением.

И, наконец, третья структура следования – это вывод результатов.

5.4. Выполнение примера на компьютере

1. Создайте новый проект с именем Цикл_с_разветвлением, следуя приложению 1.

2. Если окно конструктора форм не открыто, то откройте его щелчком на кнопке  View Designer (просмотреть конструктор), расположенной на панели инструментов окна обозревателя решений. Если же эта кнопка на панели инструментов отсутствует, то предварительно щелкните в окне обозревателя решений на компоненте проекта Form1.vb.

3. Для разработки интерфейса следуйте приложению 2.

4. Двойным щелчком на кнопке btnПуск вставьте в программный код заготовку подпрограммы btnПуск_Click – обработчика события, заключающегося в щелчке на кнопке btnПуск. Это событие будет для проекта командой выполнить требуемые вычисления.

5. Двойным щелчком на надписи Label1 создайте заготовку подпрограммы Label1_Click.

6. Введите код подпрограмм btnПуск_Click и Label1_Click.

7. Копируйте в проект из приложения 3 код подпрограмм InputVector и OutputVector. В итоге код проекта должен соответствовать листингу 1.

Листинг 1. Код проекта

```
Public Class Form1
    Private Sub btnПуск_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnПуск.Click
1:      Dim m, i, ka, kb As Integer
          Dim h, a(), b(), c() As Single
          txtЖурнал.Clear()
          m = InputBox("m = ?")
```

```

5:      txtЖурнал.AppendText("m = " & m & vbCrLf)
      h = InputBox("h = ?")
      txtЖурнал.AppendText("h = " & h & vbCrLf)
      ReDim a(m), b(m), c(m)
      InputVector(a, "a")
10:     txtЖурнал.AppendText("Вектор a" & vbCrLf)
      OutputVector(a, txtЖурнал)
      InputVector(b, "b")
      txtЖурнал.AppendText("Вектор b" & vbCrLf)
      OutputVector(b, txtЖурнал)
15:     For i = 1 To m
          If a(i) > b(i) And a(i) > h Then
              c(i) = a(i)
              ka = ka + 1
          ElseIf b(i) > a(i) And b(i) > h Then
20:              c(i) = b(i)
              kb = kb + 1
          Else
              c(i) = h
          End If
25:     Next
      txtЖурнал.AppendText("ka = " & ka & vbTab)
      txtЖурнал.AppendText("kb = " & kb & vbCrLf)
      txtЖурнал.AppendText("Вектор c" & vbCrLf)
      OutputVector(c, txtЖурнал)
End Sub

Private Sub Label1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Label1.Click
    End
End Sub

Private Sub InputVector(ByRef x() As Single, Optional _
ByVal ArrayName As String = "элемент")
    Dim i As Integer
    For i = 1 To x.GetUpperBound(0)
        x(i) = InputBox(ArrayName & "(" & i & ") = ?")
    Next
End Sub

Private Sub OutputVector(ByRef x() As Single, ByVal _
txtBx As TextBox)
    Dim i As Integer
    For i = 1 To x.GetUpperBound(0)
        txtBx.AppendText(" " & x(i))
    Next
    txtBx.AppendText(vbCrLf)
End Sub
End Class

```

8. Поясним назначение инструкций этого кода. Начнем с подпрограммы `btnПуск_Click`. Эта подпрограмма выполняется при каждом нажатии на кнопку, на которой написано `Вычислить`.

В строке 1 тела подпрограммы объявлены простые переменные. В строке 2 объявлены три динамических массива. Инструкция 3 предусматривает очистку текстового поля `txtЖурнал`. Инструкции 4 – 7 обеспечивают ввод значений и контрольный вывод переменных `m` и `h`. Инструкция 8 завершает объявление трех одномерных динамических массивов и обеспечивает их размещение в памяти.

Строка 9 кода обращается к подпрограмме `InputVector`, которая обеспечивает ввод значений всех элементов вектора `a`, начиная с значения `a(1)`. Первый аргумент этой подпрограммы – имя вектора, который нужно ввести. Второй необязательный аргумент этой подпрограммы – строка символов, являющаяся именем вводимого массива. Действие этого обращения к подпрограмме следует понимать как выполнение тела подпрограммы `InputVector`, в котором имя `x` везде заменено именем `a`.

Инструкция 10 выводит в текстовом поле `txtЖурнал` строку Вектор `a`. Строка 11 обращается к подпрограмме `OutputVector`, которая обеспечивает вывод в текстовом поле `txtЖурнал` значений всех элементов вектора `a`, начиная с значения `a(1)`.

Строки 12 – 14 кода обеспечивают ввод и контрольный вывод значений всех элементов вектора `b`, начиная с `b(1)`.

Строки с 15 по 25 являются циклом, в котором обеспечивается повторное выполнение разветвления (строки 16 – 24).

Строки 26 – 29 кода обеспечивают вывод в текстовое поле `txtЖурнал` результатов вычислений.

9. Подпрограмма `InputVector` обеспечивает ввод значений всех элементов вектора `x`, начиная с `x(1)` по последний. Вторым аргументом этой функции типа строки символов не является обязательным. Он передает функции `InputBox` имя вводимого вектора, что позволяет сделать операцию ввода нагляднее. Номер последнего элемента массива `x` возвращает метод массива `GetUpperBound(0)`. Его аргумент, равный 0, это индекс первой размерности массива. Размерности массива нумеруются, начиная с нуля (0, 1, 2, ...).

10. Подпрограмма `OutputVector` обеспечивает вывод значений всех элементов вектора `x`, начиная с `x(1)` по последний, в заданное значением аргумента `txtBx` текстовое поле. Номер последнего элемента массива возвращает метод массива `GetUpperBound(0)`.

11. Запустите проект. Проверьте его работу при значениях исходных данных: $m = 5$, $h = 2$, $a = (3,5; 1; 3; 1; 2)$, $b = (2; 4,3; 3; 1; 2)$. Если при разработке проекта не были допущены ошибки, то результат выполнения программы должен соответствовать рис. 3.

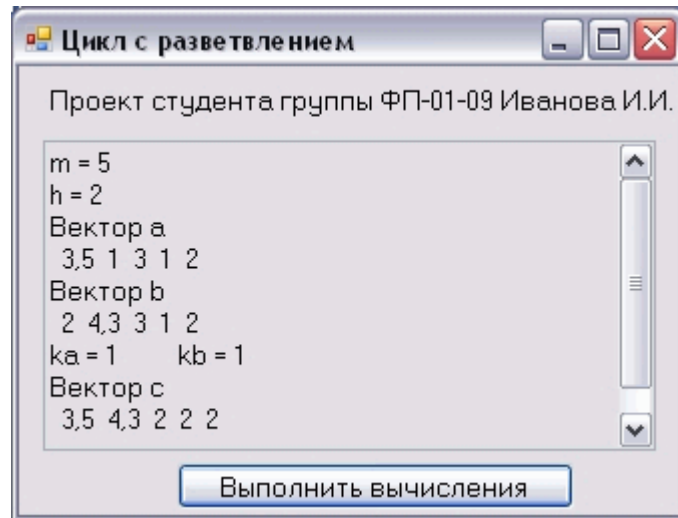


Рис. 3. Результат выполнения проекта

12. Продемонстрируйте работу проекта преподавателю.

6. Выполнение индивидуального задания

1. Замените код подпрограммы `btnПуск` кодом, составленным Вами для решения заданного Вам индивидуального варианта задания.

2. Сохраните проект.

3. Выполните отладку и тестирование проекта. Устраните обнаруженные ошибки.

4. Попробуйте ответить на вопросы для контроля.

5. Покажите полученный результат и отчет по выполненной работе преподавателю.

6. Копируйте рабочую папку либо на сетевой диск *o* в папку с шифром Вашей учебной группы, либо на собственную флэш-панель.

7. Удалите на диске *d* свою рабочую папку.

7. Вопросы для контроля

1. В чем отличие массива от переменной?
2. Могут ли элементы массива иметь разный тип?
3. Что означает инструкция `Dim g(5) As String`?
4. Что означает инструкция `Dim g() As String`?
5. Что означает инструкция `Dim g(,) As String`?
6. Чем отличается фиксированный массив от динамического массива?

7. В какой последовательности выполняются операции в выражении $a+b/c*d-e^2$?

8. Имеется квадрат со стороной h и круг диаметра d ($d < h$). Центры обеих фигур расположены в начале координат. Запишите логическое выражение, которое принимает значение `True` только в том случае, когда точка с заданными координатами (x, y) принадлежит квадрату, но не принадлежит кругу.

9. Что означает инструкция `ReDim g(n)`?

10. Для чего применяются обращение к методу `GetUpperBound(0)`?

11. Что определяют значения свойств надписи: `Text`, `TextAlign`?

12. Что определяют значения свойств кнопки: `Name`, `Text`, `Font`?

13. Что определяют значения свойств текстового поля: `Name`, `Text`, `Font`, `Multiline`, `ReadOnly`, `Anchor`, `SkrollBars`?