

## Процедуры

Процедуры представляют собой часть программного кода, снабженную именем. Записав код процедуры один раз (это называется объявлением процедуры), можно заставлять его работать (это называется вызовом процедуры или обращением к процедуре) сколько угодно раз. Процедуры различают:

- событийные (обработчики событий), которые выполняются, когда с объектом происходит соответствующее событие;
- общие, которые выполняются при обращении к ним с использованием их имени;
- процедуры – свойства, которые применяются, если объект нужно снабдить новым свойством.

Также различают процедуры:

- подпрограммы;
- функции.

Код формы и модуля состоит из кода процедур, а также кода объявлений, которые не входят в состав процедур. Выполняемый код (в отличие от объявлений) может быть только внутри процедур.

### 1. Выполнение общих процедур

Чтобы код общей функции или подпрограммы был выполнен, она должна быть вызвана.

Вызов подпрограммы является отдельной инструкцией и осуществляется обращением к ее имени в коде. Сразу после имени процедуры в скобках должны следовать значения аргументов процедуры, если таковые предусмотрены:

**ИмяПодпрограммы** ([ЗначениеАргумента1, ЗначениеАргумента2, \_  
... , ЗначениеАргументаN])

Имя подпрограммы не обладает типом, не может принимать каких либо значений и служит только для идентификации подпрограммы.

Например, вызов подпрограммы с именем Сумма, которая вычисляет значение sum суммы n элементов вектора, может выглядеть так:

Сумма (n, a, sum)

Здесь аргумент a –это имя вектора, сумма элементов которого должна быть вычислена. Обратите внимание на то, что список аргументов подпрограммы заключен в скобки.

В отличие от подпрограммы функция возвращает значение результата в точку вызова и не обязательно является отдельной инструкцией. В остальном же вызов функции подобен вызову подпрограммы. Чтобы функция могла вернуть значение, ее вызов должен осуществляться из выражения, в котором она играет роль операнда этого выражения. Обращение к функции имеет следующий синтаксис:

**ИмяФункции** ([ЗначениеАргумента1, ЗначениеАргумента2, ... , \_  
ЗначениеАргументаN])

Имя функции служит не только для идентификации, но также обладает типом и принимает значение, возвращаемое в выражение в точку ее вызова.

Например, обращение к функции `Сумма1`, которая вычисляет значение суммы  $n$  элементов вектора, происходит дважды в инструкции:

$$b = a * \text{Сумма1}(n, x) + \text{Сумма1}(m, y)$$

## 2. Объявление общих процедур

Для объявления общей подпрограммы используют следующий синтаксис:

```
[Private/Public/Static] Sub ИмяПроцедуры _
                                ([СписокАргументов])
```

[Инструкции]

[Exit Sub]

[Инструкции]

**End Sub**

Синтаксис объявления общей функции выглядит несколько иначе:

## [Private/Public/Static] Function

ИмяФункции ( [СписокАргументов] ) **As** ИмяТипа

[Инструкции]

[ИмяФункции = Выражение]

[Return Выражение]

## [Exit Function]

[Инструкции]

[ИмяФункции = Выражение]

## End Function

## Пояснения синтаксиса

Здесь жирным шрифтом выделены слова. Команды **Exit Sub** или **Exit Function** осуществляют досрочный выход из подпрограммы или соответственно функции.

Перед словами и может стоять одно из слов Private, Public, Static.

Слово `Private` (подразумевается по умолчанию) означает, что имя процедуры действует во всех процедурах формы или модуля, в которой эта процедура объявлена.

Слово `Public` можно применять только при объявлении процедуры в модуле. В этом случае имя процедуры действует во всех процедурах всех форм и модулей проекта.

Слово `Static` означает, что локальные переменные, объявленные в процедуре, сохраняют свои значения после выхода из неё.

Каждый аргумент списка аргументов имеет следующий совсем не простой синтаксис:

```
[Optional]ByVal / ByRef[ParamArray]
```

ИмяАргумента [ ( ) ] [**As** ИмяТипа] [= ЗначениеПоУмолчанию]

В этой многовариантной конструкции обязательными является только зарезервированные слова `ByVal` или `ByRef` и имя аргумента, остальные слова могут быть опущены. Слово `Optional` означает, что аргумент не является обязательным. Для такого аргумента должно быть задано

ЗначениеПоУмолчанию. После необязательного аргумента не может следовать обязательный аргумент, а также массив аргументов (ParamArray).

Ограничимся рассмотрением упрощенного синтаксиса аргумента:

**ByVal** / **ByRef** ИмяАргумента [ ( ) ] [**As** ИмяТипа]

**ByVal**. Аргумент, которому предшествует это слово, будет передаваться в процедуру по значению (передается его копия). Если аргументом окажется внешняя относительно процедуры переменная, то никакие манипуляции с этим аргументом в теле процедуры не изменят значения этой внешней переменной. Такой способ применяется для входных данных. При обращении к процедуре соответствующий аргумент может быть выражением.

**ByRef**. Передача аргумента будет производиться по ссылке (передается физический адрес аргумента). Значение внешней переменной, переданной процедуре в качестве этого аргумента, может быть изменено инструкциями процедуры. Этот режим передачи аргументов по ссылке используется для выходных параметров. При обращении к процедуре или функции соответствующий аргумент может быть только переменной.

В следующем примере функция Celsius пересчитывает градусы Фаренгейта в градусы Цельсия. Когда функция вызывается подпрограммой Main, переменная, содержащая значение аргумента, передается функции. Результат вычислений возвращается вызывающей подпрограмме и выводится в окно сообщения.

```
Sub Main()
    temp = InputBox( _
        "Введите температуру в градусах Фаренгейта")
    MsgBox "Температура равна " & Celsius(temp) _
        & " градусов Цельсия"
End Sub
```

```
Function Celsius(ГрадФар)
    Celsius = (ГрадФар - 32) * 5 / 9
End Function
```

Для того чтобы процедура была доступна во всех модулях и формах проекта, она должна быть объявлена в модуле. Процедура, объявленная в форме, доступна процедурам только этой формы.

В список аргументов подпрограммы рекомендуется включать все входные и все выходные для этой подпрограммы данные.

В список аргументов функции рекомендуется включать все входные для этой функции данные. Ее результат (если он единственный) возвращается в вызывающую программу через имя функции.

Процедура задает правило обработки объектов. Значения аргументов при обращении к процедуре указывают, к каким объектам (или их копиям) действующим в вызывающей программе будет применено это правило. При выполнении тела процедуры во всех её инструкциях аргументы заменяются

соответствующими значениями аргументов, заданными при обращении в списке значений аргументов.

### 3. Задание

Разработайте проект, применяя процедуры для решения задачи с заданным вариантом условия, а также для организации ввода и вывода массивов.

Отчет по выполненной работе должен включать титульный лист, условие задания вместе с условием варианта, описание применяемых данных, блок-схему алгоритма, программу, тесты и предполагаемые расчетные результаты их выполнения.

#### Варианты задания

1. Составьте подпрограмму, изменяющую исходный вектор путем деления его положительных элементов на свои индексы и считающую число таких замен. Используя эту подпрограмму, определите, в каком из двух заданных векторов  $A$  или  $B$  будет больше измененных элементов.

2. Составьте функцию, принимающую значение `True`, если все элементы вектора упорядочены по убыванию, и `False` – в противном случае. Применяя эту функцию к двум заданным векторам  $TA$  и  $TB$ , выведите вектор, если его элементы не упорядочены в порядке убывания их значений. В противном случае выведите сообщение: « $TA$  упорядочен» или « $TB$  упорядочен».

3. Составьте функцию, принимающую значение `True`, если все элементы вектора имеют значения больше некоторой заданной величины, или значение `False` – в противном случае. Применяя эту функцию для каждого из двух заданных векторов  $CT$  и  $DT$ , измените значение каждого элемента вектора на обратное, если окажется, что все элементы вектора имеют исходное значение больше  $N$ . В противном случае выведите сообщение: «Условие для  $CT$  не выполнено», или «Условие для  $DT$  не выполнено».

4. Составьте подпрограмму, заменяющую все элементы вектора, меньшие заданной величины на значение этой величины и считающую число таких замен. Используя эту подпрограмму, измените каждый из двух векторов  $P$  и  $C$  и выведите тот вектор, в котором оказалось большее число замен. Если же эти числа замен будут равны, выведите число замен.

5. Составьте функцию, принимающую значение `True`, если в двух векторах нет равных элементов, и значение `False` – в противном случае. Используя эту функцию для трех векторов  $R$ ,  $S$  и  $T$ , в зависимости от результатов проверок выведите сообщение: «Равных элементов нет» или «Равные элементы есть».

6. Составьте функцию, принимающую значение `True`, если в первом из двух заданных векторов количество отрицательных элементов окажется больше, чем во втором, и значение `False` – в противном случае. Применив эту функцию к двум заданным векторам  $A$  и  $B$ , измените на противоположный знак значения элементов того вектора, у которого оказалось больше отрицательных элементов.

Если количество отрицательных элементов одинаково, то выведите соответствующее сообщение.

7. Составьте функцию, определяющую значение индекса элемента вектора, равного заданной величине  $K$  и расположенного ближе к началу массива, если он не единственный. Используя эту функцию, выведите тот из двух заданных векторов  $A$  и  $B$ , в котором элемент, равный  $K$ , находится ближе к началу массива. При одинаковом положении такого элемента от начала выведите сообщение «Позиция одинакова». Если же хотя бы в одном из двух заданных векторов нет элемента, равного  $K$ , выведите сообщение: «Равный  $K$  элемент не обнаружен».

8. Составьте функцию, определяющую значение индекса элемента вектора, значение которого отрицательно и расположенного ближе к началу массива, если он не единственный. Используя эту функцию, рассчитайте среднее арифметическое значение элементов того из двух заданных векторов  $C$  и  $T$ , в котором отрицательный элемент расположен ближе к началу вектора. В случае равенства индексов таких элементов рассчитайте среднее арифметическое для каждого из этих двух векторов. Если же хотя бы у одного из двух заданных векторов нет отрицательных элементов, то выведите сообщение: «Нет отрицательных элементов».

9. Составьте функцию для определения количества нулевых элементов в векторе. Используя эту функцию, установите, в каком из этих двух заданных векторов  $IC$  и  $IE$  количество нулевых элементов меньше, и выведите сумму элементов этого массива. В случае равенства количества нулевых элементов, выведите лишь это значение.

10. Составьте подпрограмму для определения количества отрицательных элементов в векторе. Используя эту подпрограмму, установите, в каком из двух заданных векторов  $P$  и  $B$  больше количество отрицательных элементов. Для этого вектора получите среднее значение элементов. В случае равенства количества отрицательных элементов этих векторов выведите «Одинаковое число отрицательных элементов».

11. Составьте функцию, принимающую значение `True`, если количество положительных элементов исходного вектора больше количества его отрицательных элементов, и значение `False` – в противном случае. Для каждого из двух заданных векторов  $A$  и  $B$  получите сумму положительных элементов, если положительных элементов в векторе больше, чем отрицательных, иначе – определите сумму отрицательных элементов.

12. Составьте функцию, принимающую значение `True`, если количество отрицательных элементов вектора больше количества его нулевых элементов, и значение `False` – в противном случае. Применяя эту функцию, выполните следующие вычисления для каждого из двух заданных векторов  $A$  и  $B$ . Если отрицательных элементов массиве больше, чем нулевых, вычислите произведение отрицательных элементов, иначе – определите сумму индексов нулевых элементов.

13. Составьте подпрограмму, определяющую среднее арифметическое значение тех элементов заданного одномерного массива, квадрат значений которых больше некоторой заданной величины. Если среднее арифметическое значение элементов вектора  $A$ , квадрат которых больше  $N$ , меньше среднего арифметического значения аналогичных элементов вектора  $B$ , то выведите сообщение: «Среднее  $A$  меньше нормы». В противном случае выведите сообщение: «Среднее  $A$  в норме». Если же хотя бы в одном из векторов  $A$  или  $B$  не окажется элемента, значение которого больше  $N$ , то выведите сообщение: «Нет запаса».

14. Составьте подпрограмму, вычисляющую среднее арифметическое значение тех элементов вектора, модуль значения которых меньше заданной величины. Используя эту подпрограмму, выведите те элементы заданных векторов  $A$  и  $B$ , значение которых больше соответствующего найденного среднего арифметического. Если среднее не существует, выведите соответствующее сообщение.

15. Составьте подпрограмму, заменяющую все отрицательные элементы исходной матрицы их модулями и подсчитывающую число таких замен. Примените эту подпрограмму для заданных матриц  $T1$  и  $T2$ , выведите общее число выполненных замен, причем в случае совпадения числа замен с общим числом элементов в матрице, предусмотрите вывод соответствующего поясняющего текста.

16. Составьте подпрограмму для замены всех отрицательных элементов вектора их модулями и подсчета числа таких замен. Примените эту подпрограмму к каждому из двух заданных векторов  $A$  и  $B$ . Если окажется, что число замен в векторе  $A$  больше 5, выведите этот измененный вектор, иначе – выведите вектор  $B$ .

17. Составьте функцию, принимающую значение True, если элементы главной диагонали квадратной матрицы расположены в ней в порядке возрастания их значений, и False – в противном случае. Если все элементы главной диагонали любой из заданных матриц  $A$  и  $B$  расположены по возрастанию, увеличьте элементы каждой строки этой матрицы на соответствующий элемент главной диагонали, в противном случае выведите сообщение "Условие нарушено".

18. Составьте функцию, принимающую значение True, если все элементы главной диагонали квадратной матрицы равны между собой, и False – в противном случае. Заданы матрицы  $A$  и  $B$ . Для каждой из них следует проделать следующее. Если элементы главной диагонали матрицы равны между собой, увеличьте каждый элемент матрицы на значение элемента главной диагонали. В противном случае выведите сообщение "Элементы не совпадают".

19. Составьте функцию для определения минимального элемента матрицы. Применяя эту функцию, определите для каждой из заданных матриц  $A$  и  $B$  значение минимального элемента и, если оно положительно, увеличьте каждый элемент главной диагонали на модуль этого значения. Иначе выведите сообщение: «Минимальный  $\leq 0$ ».

20. Составьте функцию для определения максимального элемента матрицы. Используя эту функцию, определите для каждой из заданных матриц  $A$  и  $B$  значение максимального элемента и, если оно больше заданной величины  $N$ , извлеките квадратный корень из значения каждого элемента первой строки матрицы. В противном случае выведите сообщение: "Значение не предельно".

21. Составьте функцию для вычисления среднего арифметического значения элементов главной диагонали квадратной матрицы. Используя эту функцию для двух заданных матриц  $A$  и  $B$ , выведите первую строку матрицы, если среднее арифметическое значение элементов ее главной диагонали положительно. В противном случае выведите сообщение «Условие не выполнено».

22. Составьте подпрограмму для нахождения индексов максимального элемента квадратной матрицы. Используя эту подпрограмму для каждой из двух квадратных матриц  $A$  и  $B$ , получите скалярное произведение строки матрицы на ее столбец, на пересечении которых находится максимальный элемент. Под скалярным произведением  $k$  – ой строки на  $m$  – ый столбец квадратной матрицы  $A$ , имеющей  $n$  строк и  $n$  столбцов, понимается сумма попарных произведений  $a_{k1} * a_{1m} + a_{k2} * a_{2m} + \dots + a_{kn} * a_{nm}$ .

23. Составьте подпрограмму для нахождения индексов минимального элемента квадратной матрицы. Примените эту подпрограмму для каждой из двух квадратных матриц  $A$  и  $B$  и получите для каждой матрицы вектор, элементы которого равны попарным суммам элементов строки и столбца, на пересечении которых находится минимальный элемент. Если минимальный элемент матрицы  $A$  находится на пересечении  $k$  – ой строки и  $m$  – го столбца, то  $i$  – ый элемент искомого вектора равен  $a_{ki} + a_{im}$  ( $i = 1, 2, \dots, n$ ).

24. Составьте функцию для определения индекса максимального по модулю элемента вектора. Если максимальные по модулю элементы двух заданных векторов  $R$  и  $T$  имеют равные индексы, выведите вектор  $R$ , иначе – вектор  $T$ .

25. Составьте подпрограмму, заменяющую все меньшие заданной величины элементы вектора на ноль и определяющую число таких замен. Примените эту подпрограмму для двух заданных векторов  $P$  и  $C$  и выведите тот вектор, в котором число замен оказалось большим.

26. Составьте функцию для определения минимального по модулю элемента вектора. Если минимальные по модулю элементы двух заданных векторов  $X$  и  $T$  отличаются менее чем на  $B$ , просуммируйте все положительные элементы обоих векторов. В противном случае выведите значения найденных минимальных элементов.

27. Составьте процедуру для определения максимального по модулю элемента вектора. Если максимальные по модулю элементы двух заданных векторов  $MT$  и  $MP$  имеют равные значения, вычислите среднее арифметическое значение отрицательных элементов для каждого вектора. В противном случае выведите значения найденных максимальных элементов.

28. Составьте подпрограмму, формирующую вектор из сумм элементов в столбцах заданной матрицы. Используя эту подпрограмму для двух матриц  $A$  и  $B$ , подсчитайте общее количество положительных сумм элементов в столбцах заданных матриц.

29. Составьте подпрограмму, формирующую вектор из произведений элементов в столбцах заданной матрицы. Используя эту подпрограмму для двух матриц  $A$  и  $B$ , получите количество отрицательных произведений элементов в столбцах каждой из заданных матриц.

30. Составьте функцию для определения среднего арифметического значения элементов матрицы. Используя эту функцию, рассчитайте средний балл в сессию для каждой из двух групп студентов, сдававших  $k$  экзаменов. Если окажется, что средний балл группы меньше 4, то подсчитайте для этой группы количество неудовлетворительных оценок.

## 4. Пример разработки проекта

### 4.1. Условие

Составьте процедуру, определяющую среднее арифметическое значение тех элементов матрицы, значения которых меньше заданной величины *Порог*. Если среднее арифметическое значение элементов матрицы  $a$ , которые меньше *Порог*, меньше среднего арифметического значения аналогичных элементов матрицы  $b$ , то измените обе матрицы, уменьшив их каждый элемент на среднее матрицы  $a$ . В противном случае измените обе матрицы, уменьшив их каждый элемент на среднее матрицы  $b$ . Если же хотя бы в одной из матриц  $a$  или  $b$  не окажется элемента, значение которого меньше *Порог*, то выведите сообщение: «Превышение».

### 4.2. Выделение подзадач

Наметьте те части задачи, которые выполняются не один раз и их оформление в виде процедуры будет способствовать сокращению объема программы. Такие части задачи называют подзадачами. Разбиение задачи на подзадачи имеет значение не только с точки зрения сокращения объема программного кода, но и приводит к структурированию задачи, позволяет ее решать по частям и облегчает ее понимание. При программировании подзадачи рекомендуется применить функцию, если в результате решения подзадачи находится один результат. Если же результатов несколько, то рекомендуется применить подпрограмму.

Применительно к условию, приведенному в параграфе 4.1, можно выделить следующие подзадачи.

1. Вычисление среднего арифметического значения тех элементов матрицы, значения которых меньше заданной величины. На первый взгляд, здесь один результат – значение среднего арифметического. Однако не следует забывать, что среднее может не существовать, если нет ни одного элемента матрицы, меньше заданной величины. Следовательно, в результате решения этой подзадачи должны быть получены значения двух переменных: одна – среднее значение, а вторая – сигнал того, чем закончилось решение подзадачи: нормально



– среднее получено, или не нормально – среднее не существует. Для программирования этой подзадачи применим подпрограмму:

```
Средн(ByVal X(,) As Single, ByVal h As Single, _  
ByRef Среднее As Single, ByRef Сигнал As Boolean).
```

Первый аргумент – это матрица, которую нужно обработать.

Второй аргумент – это пороговая величина, с которой мы должны сравнивать каждый элемент матрицы.

Третий аргумент – возвращаемое среднее значение.

Четвертый аргумент (последний в списке аргументов) – это переменная, которая будет сигнализировать, чем закончилось обращение к подпрограмме. Если среднее существует, она будет иметь значение True, в противном случае – False.

2. Вычитание из всех элементов матрицы. Применим процедуру `Вычитание(ByRef x(,) As Single, ByVal v As Single)`.

Первый аргумент – это изменяемая матрица.

Второй аргумент – вычитаемая величина.

3. Ввод матрицы. Применим подпрограмму

```
InputMatrix(ByRef x(,) As Single, Optional _  
ByVal ArrayName As String = "элемент").
```

Эта подпрограмма применялась в примере предыдущего задания и относительно нее там уже были сделаны необходимые пояснения.

4. Вывод матрицы. Применим процедуру `OutputMatrix Вывод_матрицы(ByRef x(,) As Single)`.

Эта подпрограмма также применялась в примере предыдущего задания и относительно нее там тоже были сделаны необходимые пояснения.

Алгоритм подзадачи, указанной выше в пункте 2, достаточно прост, поэтому можно обойтись без составления его блок-схемы. Он представляют собой кратный цикл.

#### *4.3. Разработка алгоритма подзадачи вычисления среднего значения элементов матрицы*

##### *Выбор данных*

Входные данные:

x – матрица, двумерный массив элементов с дробной частью одинарной точности;

h – величина, с которой сравниваются элементы матрицы, переменная с дробной частью одинарной точности.

Выходные данные:

Среднее – среднее арифметическое элементов матрицы, значение которых меньше величины Порог;

Сигнал – переменная логического типа, принимающая значение True, если вычисление среднего закончилось получением его значения, или False, если среднего не существует.

Промежуточные данные:

$m$  – количество строк матрицы, переменная целого типа;

$n$  – количество столбцов матрицы, переменная целого типа;

$i, j$  – параметры циклов, переменные целого типа;

Сумма – сумма элементов матрицы, значения которых меньше заданной величины, переменная с дробной частью обычной точности;

Количество – количество элементов матрицы, значения которых меньше заданной величины, переменная целого типа.

#### *Блок-схема алгоритма подзадачи 1*

Блок-схема алгоритма приведена на рис. 1. Обратите внимание на то, что отсутствуют операторы ввода входных данных. Значения входных данных передаются процедуре из вызывающей программы через список аргументов. Вывод выходных данных в процедурах, как правило, не предусмотрен. Они также передаются в вызывающую программу через список аргументов.

#### *4.4. Разработка главного алгоритма проекта*

##### *Выбор данных*

Исходные данные:

$a$  и  $b$  – заданные матрицы, двумерные массивы с дробной частью одинарной точности.

$ma$  и  $na$  количество соответственно строк и столбцов матрицы  $a$ , переменные целого типа.

$mb$  и  $nb$  количество соответственно строк и столбцов матрицы  $b$ , переменные целого типа.

Порог – величина, с которой сравниваются значения элементов матриц  $a$  и  $b$ .

Результаты:

$sa$  – среднее значение элементов матрицы  $a$ , значение которых меньше величины Порог.

$sb$  – среднее значение элементов матрицы  $b$ , значение которых меньше величины Порог.


Промежуточные данные:

СигА, СигВ – признаки успешности завершения вычислений среднего значения элементов матриц  $a$  и  $b$  соответственно, переменные логического типа.

Блок-схема главного алгоритма изображена на рис. 2.

#### *4.5. Выполнение примера на компьютере*

1. Создайте новый проект с именем Процедуры, 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.

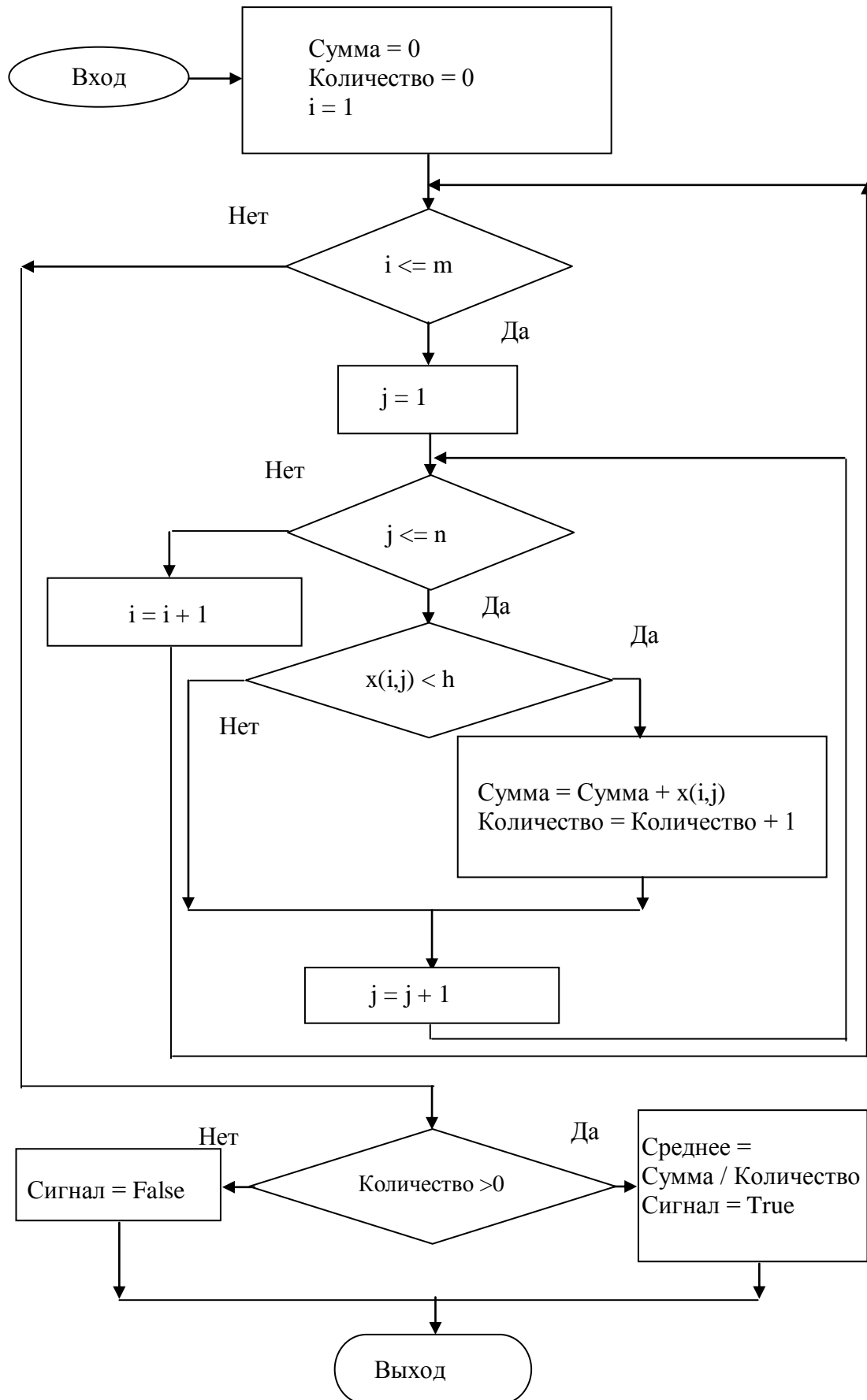
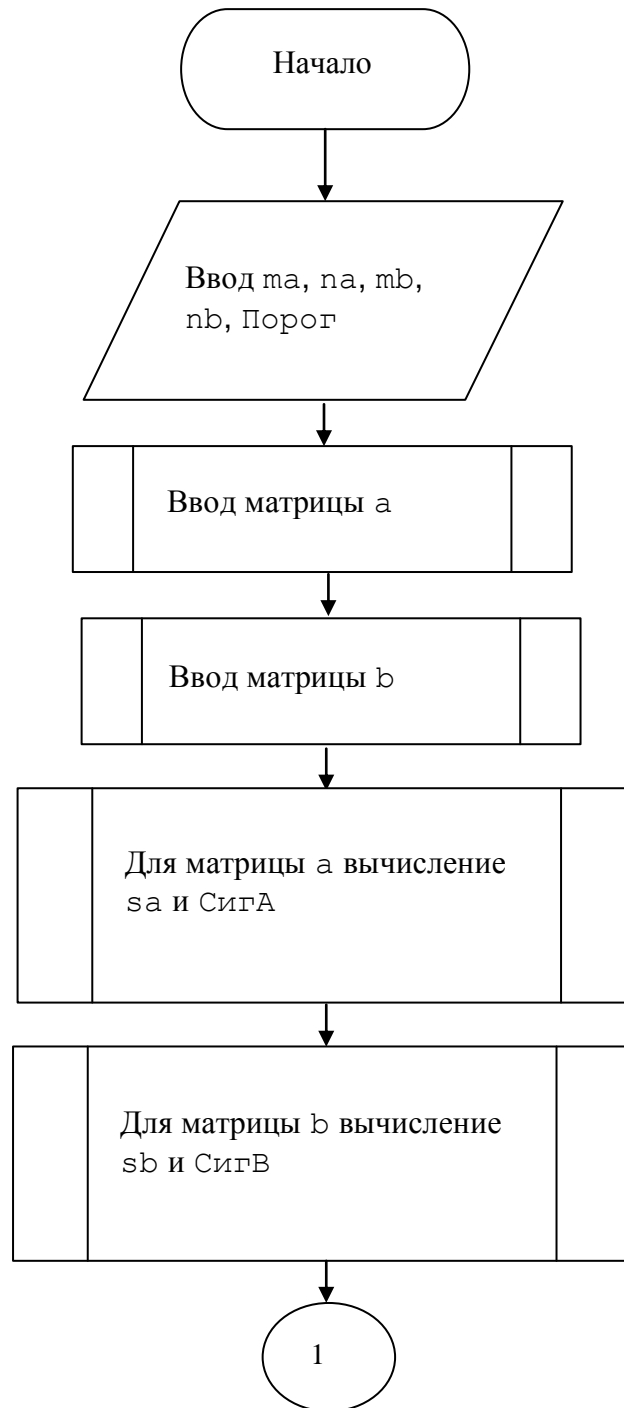


Рис. 1. Блок-схема алгоритма подзадачи 1

3. Для разработки типового 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`.



**Рис. 2 (начало)**

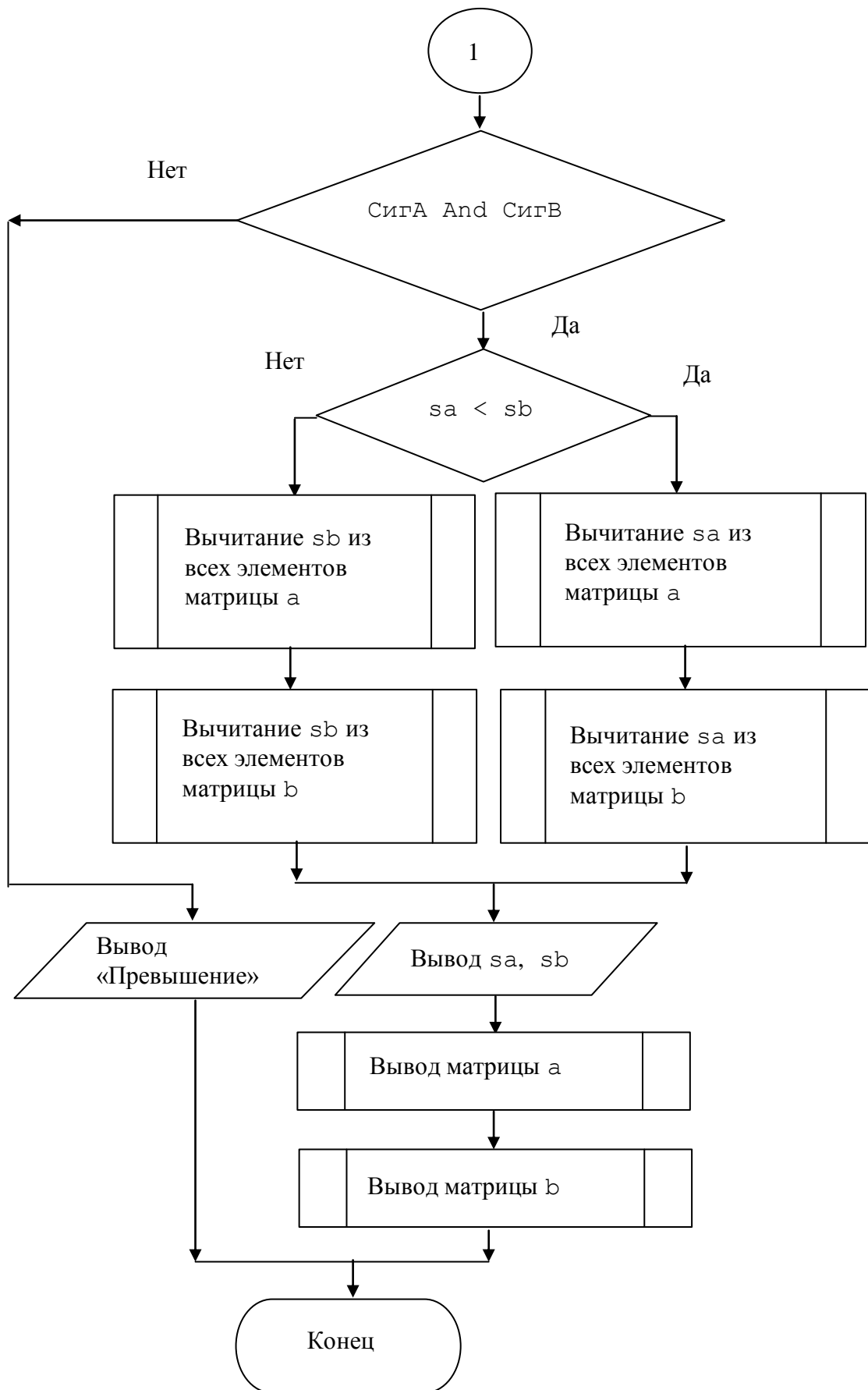


Рис. 2. Блок - схема главного алгоритма (окончание)

6. Введите код подпрограмм btnПуск\_Click и Label1\_Click.
7. Введите код подпрограммы Средн.
8. Введите код подпрограммы Вычитание.
9. Копируйте в проект из приложения 3 код подпрограмм InputMatrix и OutputMatrix.
10. В итоге код проекта должен соответствовать листингу 1, который включает событийную подпрограмму btnПуск\_Click, подпрограмму Label1\_Click, а также подпрограмму Средн, подпрограмму Вычитание, подпрограмму InputMatrix для ввода матрицы и подпрограмму OutputMatrix, выполняющую вывод матрицы.

Для контроля правильности введенных значений исходных данных в программе предусмотрен их вывод.

#### 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 ma, na, mb, nb As Integer
            Dim a(,) As Single, b(,) As Single
            Dim Порог, sa, sb As Single
            Dim СигА, СигВ As Boolean
5:         txtЖурнал.Clear()
            ma = InputBox _
                ("Задайте количество строк матрицы a")
            na = InputBox _
                ("Задайте количество столбцов матрицы a")
            mb = InputBox _
                ("Задайте количество строк матрицы b")
            nb = InputBox _
                ("Задайте количество столбцов матрицы b")
10:        Порог = InputBox("Задайте Порог")
            txtЖурнал.AppendText("ma = " & ma & _
                vbTab & "na = " & na & vbCrLf)
            txtЖурнал.AppendText("mb = " & mb & _
                vbTab & "nb = " & nb & vbCrLf)
            txtЖурнал.AppendText("Порог = " & Порог & vbCrLf)
            ReDim a(ma, na)
15:        ReDim b(mb, nb)
            InputMatrix(a, "a")
            txtЖурнал.AppendText("Матрица a" & vbCrLf)
            OutputMatrix(a, txtЖурнал)
            InputMatrix(b, "b")
20:        txtЖурнал.AppendText("Матрица b" & vbCrLf)
            OutputMatrix(b, txtЖурнал)
            Средн(a, Порог, sa, СигА)
            Средн(b, Порог, sb, СигВ)
            If СигА And СигВ Then
25:                If sa < sb Then
                    Вычитание(a, sa)
                    Вычитание(b, sa)
                End If
            End If
        End Sub
End Class
```

```

Else
    Вычитание(a, sb)
30:    Вычитание(b, sb)
End If
txtЖурнал.AppendText("sa = " & sa & _
vbTab & "sb = " & sb & vbCrLf)
txtЖурнал.AppendText("Измененная матрица a" _
& vbCrLf)
OutputMatrix(a, txtЖурнал)
35:    txtЖурнал.AppendText("Измененная матрица b" _
& vbCrLf)
OutputMatrix(b, txtЖурнал)
Else
    txtЖурнал.AppendText("Превышение" & vbCrLf)
End If

End Sub

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

Private Sub Средн(ByVal x(,) As Single, ByVal h As _
Single, ByRef Среднее As Single, ByRef Сигнал As Boolean)
1:    Dim i, j As Integer
    Dim Сумма As Single, Количество As Integer
    For i = 1 To x.GetUpperBound(0)
        For j = 1 To x.GetUpperBound(1)
5:            If x(i, j) < h Then
                Сумма = Сумма + x(i, j)
                Количество = Количество + 1
            End If
        Next
10:    Next
    If Количество > 0 Then
        Среднее = Сумма / Количество
        Сигнал = True
    Else
15:        Сигнал = False
    End If
End Sub

Private Sub Вычитание(ByRef x(,) As Single, _
ByVal v As Single)
1:    Dim i As Integer, j As Integer
    For i = 1 To x.GetUpperBound(0)
        For j = 1 To x.GetUpperBound(1)
5:            x(i, j) = x(i, j) - v
        Next
    Next
End Sub

```

```

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

Private Sub OutputMatrix(ByVal x(,) As Single, _
ByVal y As TextBox)
    Dim i, j As Integer
    For i = 1 To x.GetUpperBound(0)
        For j = 1 To x.GetUpperBound(1)
            y.AppendText(x(i, j) & vbTab)
        Next
        y.AppendText(vbCrLf)
    Next
End Sub
End Class

```

В событийной подпрограмме btnПуск\_Click в инструкциях 22 и 23 выполняется вычисление средних значений элементов матриц a и b, значение которых меньше величины Порог. Затем в инструкциях 24 – 40 проверяется существование этих двух средних значений, а также в зависимости от итога этой проверки вычисляются и выводятся требуемые результаты.

В подпрограмме Средн в строках 3 – 10 вычисляется сумма и количество элементов матрицы, значение которых меньше h. В строках 11 – 16 выполняется проверка существования среднего значения, а также в зависимости от результатов этой проверки выполняется вычисление значения среднего значения.

В подпрограмме Вычитание из всех элементов заданной матрицы вычитается заданная величина.

Подпрограмма InputMatrix обеспечивает ввод значений всех элементов матрицы, указанной первым аргументом, кроме элементов, находящихся в нулевой строке и в нулевом столбце. Второй аргумент этой подпрограммы является не обязательным. Он является строкой (именем вводимого массива) и предназначен для применения в качестве подсказки в окне функции InputBox. Применяемый в этой подпрограмме метод массива GetUpperBound возвращает максимальное значение индекса строки (при обращении x.GetUpperBound(0)). Применяемое в этой подпрограмме обращение к методу массива GetUpperBound(1) возвращает максимальное значение индекса столбца (при обращении x.GetUpperBound(0)).

Подпрограмма OutMatrix обеспечивает вывод значений всех элементов матрицы, указанной первым аргументом кроме элементов, находящихся в



нулевой строке и в нулевом столбце. Второй аргумент этой подпрограммы является именем текстового окна, предназначенного для этого вывода.

### 11. Сохраните проект.

Для тестирования проекта необходимо применить, по крайней мере, три теста, чтобы проверить работу всех трех ветвей главного алгоритма.

Тест № 1 . Порождает ситуацию: выражение СигА And СигВ равно True и  $sa < sb$ .

$ma = 2; na = 2; mb = 2; nb = 3; \text{Порог} = 5.$

$$a = \begin{Bmatrix} 1 & 3 \\ 6 & 2 \end{Bmatrix} \quad b = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{Bmatrix}$$

Тест № 2. Порождает ситуацию: выражение СигА And СигВ равно True и  $sa \geq sb$ .

$ma = 2; na = 2; mb = 2; nb = 3; \text{Порог} = 5;$

$$a = \begin{Bmatrix} 4 & 3 \\ 6 & 1 \end{Bmatrix} \quad b = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{Bmatrix}$$

Тест № 3. Порождает ситуацию: выражение СигА And СигВ равно False.

$ma = 2; na = 2; mb = 2; nb = 3; \text{Порог} = 5;$

$$a = \begin{Bmatrix} 6 & 7 \\ 8 & 9 \end{Bmatrix} \quad b = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{Bmatrix}$$

12. Выполните тестирование проекта. Для этого запустите проект и выполните счет для теста № 1. Результат работы проекта должен соответствовать рис. 3 (слева).

$ma = 2$ $na = 2$ $mb = 2$ $nb = 3$ Порог = 5 Матрица a 1        3 6        2 Матрица b 1        2        3 4        5        6 $sa = 2$ $sb = 2,5$ Измененная матрица a -1        1 4        0 Измененная матрица b -1        0        1 2        3        4	$mb = 2$ $nb = 3$ Порог = 5 Матрица a 4        3 6        1 Матрица b 1        2        3 4        5        6 $sa = 2,666667$ $sb = 2,5$ Измененная матрица a 1,5        0,5 3,5        -1,5 Измененная матрица b -1,5        -0,5        0,5 1,5        2,5        3,5	$ma = 2$ $na = 2$ $mb = 2$ $nb = 3$ Порог = 5 Матрица a 6        7 8        9 Матрица b 1        2        3 4        5        6 Превышение
--	---	---

Рис. 3. Результаты счета для теста 1 (слева), и теста 2 (посередине), и теста 3 (справа)

13. Повторите тестирование проекта с тестом № 2. Результат работы проекта должен соответствовать рис. 3 (посередине).

14. Повторите тестирование проекта с тестом № 3. Результат работы проекта должен соответствовать рис. 3 (справа).

15. Продемонстрируйте работу проекта примера преподавателю.

### **5. Выполнение индивидуального задания**

16. Замените программный код проекта соответствующим кодом, заранее разработанным Вами для заданного варианта задания.

17. Сохраните проект.

18. Выполните отладку и тестирование проекта.

19. Попробуйте ответить на вопросы для контроля.

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

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

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

### **6. Вопросы для контроля**

1. Какие виды процедур определены в VB?

2. Как вызвать подпрограмму?

3. Как вызвать функцию?

4. Поясните синтаксис объявления подпрограммы.

5. Поясните синтаксис объявления функции.

6. Что означает передача аргумента процедуры по значению? Чем может быть такой аргумент при обращении к процедуре?

7. Что означает передача аргумента процедуры по ссылке? Чем может быть такой аргумент при обращении к процедуре?

8. Где должна быть объявлена процедура или функция, чтобы к ней можно было обращаться из любой процедуры и функции формы?

9. Где и как должна быть объявлена процедура или функция, чтобы к ней можно было обращаться из любой процедуры и функции любой формы проекта?

10. Ознакомьтесь с фрагментом объявления процедуры *s*, находящимся в форме *Form1*:

```
Private Sub s (ByRef a As Single, ByVal b _
    As Integer, ByVal c As String, ByVal d As Boolean)
    .
    .
    .
End Sub
```

Будет ли правильным обращение к этой процедуре, находящееся в следующем фрагменте процедуры *cmd\_Click*, объявление которой принадлежит той же форме *Form1*:

```
Private Sub cmd_Click()
Dim e As Single, f As Integer, g As String, _
h As Boolean
```

```
·
·
·
·
·
·
```

```
s(e, g, f, h)
```

```
End Sub
```

11. Ознакомьтесь с фрагментом объявления процедуры *s*, находящимся в форме Form1:

```
Private Sub s (ByRef a As Single, ByVal b _
As Integer, ByVal c As String, ByVal d As Boolean)
```

```
·
·
·
```

```
End Sub
```

Будет ли правильным обращение к этой процедуре, находящееся в следующем фрагменте процедуры *cmd\_Click*, объявление которой принадлежит той же форме Form1:

```
Private Sub cmd_Click()
Dim e As Single, f As Integer, g As String, h As
Boolean
```

```
·
·
·
·
·
·
```

```
s(e, g, f)
```

```
End Sub
```

12. Ознакомьтесь с фрагментом объявления процедуры *s*, находящимся в форме Form1:

```
Private Sub s (ByVal a() As Single, ByRef b As
Integer)
```

```
·
·
·
```

```
End Sub
```

Будет ли правильным обращение к этой процедуре, находящееся в следующем фрагменте процедуры *cmd\_Click*, объявление которой принадлежит той же форме Form1:

```

Private Sub cmd_Click()
Dim e() As Single, f As Integer
f = InputBox("f = ?")
ReDim e(f)

.
.
.

s(f, e)

.
.
.

End Sub

```

13. Какой способ передачи аргументов должен быть применен для выходных аргументов процедуры, вычисленные значения которых должны быть переданы в вызывающую процедуру?

- a) ByVal
- b) ByRef