## Тема 4.7

## Программирование алгоритмов формирования и обработки одномерных массивов

### 4.7.1. Структурированные данные

Часто приходится обрабатывать не одиночные данные, а совокупность данных одного типа. Например, задача ***табулирования функции***, которая состоит в получении последовательности значений заданной функции при нескольких значениях аргумента. Для промежуточного хранения каждого значения полученных данных требуется объявить собственную переменную с уникальным именем.

Обращение к каждой переменной последовательности по имени превращается в длинную вереницу однотипных операций с каждой переменной. Программный код становится плохо обозримым. Для размещения такой программе требуется много памяти.

Для устранения указанных проблем в алгоритмических языках используются структурированные данные. Самыми простыми структурированными данными являются ***массивы данных***.

***Массив*** – это совокупность однотипных переменных (***элементов массива***). Имя у всех переменных одно и то же, а для доступа к конкретному элементу массива используется дополнительный идентификатор – его порядковый номер (индекс), который начинается с 0.

Кроме массивов в программировании для построения эффективных алгоритмов могут использоваться и другие стандартные структуры данных, такие структуры данных, как ***стеки, очереди, связанные списки*** и другие.

Наряду со стандартными структурами данных, могут использоваться структуры данных, определяемые пользователем. Эти структуры данных определяются средствами объектно-ориентированного программирования с помощью ***классов***.

### 4.7.2. Средства описания и работы с одномерными массивами данных

***Массив*** – последовательность переменных одинакового типа, объединенных общим ***именем***. Например: одномерный массив **а(9)** состоит из 10 элементов с общим именем **а: a(0), a(1), a(2), a(3),..., a(9),** упорядоченных по индексу i**,** который принимает значения от 0 до 9:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a(i) | **5** | **4** | **1** | **0** | **22** | **6** | **183** | **5** | **1** | **9** |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Массив в программе VBобъявляется точно так же, как объявляются простые переменные. Если массив объявлен ***локальным***, его можно использовать только в той процедуре, в которой он объявлен. Если массив объявлен как ***глобальный***, он может быть использован в любом месте программы.

При объявлении массива оператор объявления должен включать следующую информацию:

* ***имя массива*** – имя (идентификатор), которое используется для представления массива в программе;
* ***тип данных*** – тип данных, который имеют элементы массива;
* ***размерность (ранг)*** – количество измерений объявляемого массива (т.е. количество индексов при объявлении; одномерные массивы имеют одно измерение);
* ***количество элементов*** – количество элементов, которые будут содержаться в массиве.

Рассмотрим примеры некоторых описаний массивов:



В этих примерах объявлены следующие массивы:

* одномерный массив **d**, состоящий из 31 элемента типа **Integer** с индексами от 0 до 30;
* одномерный массив **a,** состоящий из 11 элементов типа **Double** с индексами от 0 до 10;
* двумерный массив **b**, состоящий из 14х11=151 элемента типа **Single** с индексами по строкам от 0 до 13 и по столбцам от 0 до 10.

***Обратите внимание, что значением нижней границы массива в*** *VB* ***может быть только 0***.

Таким образом, массив состоит из элементов, которые могут быть доступны при помощи ***индексов***. При обращении к элементам массива ***индексы*** записываются вслед за именем в круглых скобках и могут представлять собой любое допустимое целочислен­ное выражение. Например, **d(24), a(2\*i+1).**

***Обратите внимание, что количество индексов указывает на размерность массива.*** Так, в приведенном выше примере размерность массива **a(10)** равна единице. Массив **b(2,3)** имеет размерность 2.

В отличие от ***размерности***, ***размер массива*** – это количество элементов в массиве. В нашем примере размер массива, **а(10)** равен 11.

Перед использованием массива в программе его необходимо объявить с помощью оператора **Dim**, а элементам массива присвоить конкретные значения. Оператор **Dim *выделяет место в памяти*** компьютера для размещения элементов массива, обнуляет элементы числовых массивов или заполняет элементы строковых массивов пустыми строками ('''').

Как и для простых типов, данных, при объявлении массивов, которые являются структурированными типами данных, различают два способа распределения памяти: ***статическое*** – на этапе компиляции до выполнения программы, и ***динамическое*** – в ходе выполнения программы. По умолчанию массив, границы которого заданы константными выражениями, считается статическим. Память для размещения такого массива выделяется на этапе компиляции и сохраняется за ним на весь период выполнения.

Заполнить элементы массива конкретными значениями можно с помощью ***ввода*** значений элементов массива, с помощью оператора ***присваивания*** или с помощью ***инициализации*** элементов массива.

***Инициализация*** элементов массива – это поэлементное присваивание значения в операторе объявления массива. В этом случае размер массива не указывается в круглых скобках после имени массива, а определяется неявно размером списка значений. Список значений начинается с элемента с индексом 0 и заключается в фигурные ***скобки***, например:

|  |
| --- |
| **Dim Город ( ) As String = {"Рязань" , "Тула" , "Калуга"}** |

Следует отметить, что независимо от конкретной задачи, алгоритмы формирования и обработки массивов обычно строятся с использованием регулярных циклических структур:

|  |
| --- |
| **For** i = 0 **То** *КоличествоЭлементовМассива* – 1  *ИмяМассива*(i) = *выражение* или *Переменная* = *ИмяМассива*(i)  **Next i** |

Чтобы облегчить работу с массивами в процедурах, для определения верхней границы массива используется встроенная функция  
 **Bound**(*ИмяМассива*)**.**

Эта функция возвращает (определяет) номер последнего элемента массива и позволяет обрабатывать массивы в процедурах, не передавая в них в качестве параметра количество элементов массива. Например,

|  |
| --- |
| **For i = 0 То UBound(***ИмяМассива***)**  *ИмяМассива***(i) =** *выражение***или***Переменная* = *ИмяМассива***(i)**  **Next i** |

Кроме того, для определения верхней границы одномерного массива можно использовать метод **GetUpperBound().** Поскольку массив одномерный, то в скобках следует указывать значение 0. Например:

|  |
| --- |
| **For i = 0 To a.GetUpperBound(0)**  **sum = sum + a(i)**  **Next i** |

Если имя массива, является формальным параметром процедуры, то после имени массива необходимо поместить пустые круглые скобки:

|  |
| --- |
| **ByVal** *ИмяМассива*() **As** *Тип* **или ByRef** *ИмяМассива*() **As** *Тип* |

Кроме того, известно, что ключевое слово **ByVal** указывает передачу аргумента-массива по значению, а ключевое слово **ByRef** указывает, что аргумент-массив передается по ссылке. Заметим, что если ключевые слова **ByVal** или **ByRef** опущены, то аргумент-массив переда­ется по ссылке.

Таким образом, при описании формальных параметров любой процедуры ***после*** *ИмяМассива* ***не­обходимо всегда включать пустые круглые скобки, так как они указывают, что этот параметр являет­ся одномерным массивом.***

|  |
| --- |
| **Sub Show1(ByRef Lines() As Single, ByVal NLines As Integer)**  **…**  **End Sub**  **Function Sort(ByRef List() As String) NLines As Integer**  **…**  **End Sub** |

Обращение к этим процедурам может, например, быть следующим:

|  |
| --- |
| **Show1(Lines, 5)**  **N1 = Sort(List)** |

***Обратите внимание на то, что после имени массива, который является фактическим параметром, скобки отсутствуют***.

Как известно, передача аргументов по значению (с помощью ключевого слова **ByVal)** приводит к тому, что VB передает копию данных процедуре. Поэтому не следует передавать массивы по значению, если в этом нет особой необходимости.

**Пример 4.7.2-1. Написать процедуры ввода/вывода, которые могут использоваться в алгоритмах формирования и отображения одномерных массивов.**

Процедуры ввода и вывода для одномерных массивов представлены на   
рис. 4.7.2-1–4.7.2-3.

|  |
| --- |
| *'Процедура ввода элементов массива типа Single с клавиатуры*  **Sub vvodSngMac15(ByRef a( ) As Single, ByVal L As ListBox)**  **Dim i As Integer**  **For i = 0 To UBound(a)**  **a(i) = CSng(Val(InputBox("Введите" & i & "-й элемент"))**  **Next i**  **End Sub** |

*Рис. 4.7.2-1. Процедура ввода элементов массива* **Single** *с клавиатуры*

**Примера 4.7.2-1**

|  |
| --- |
| *'Процедура формирования массива случайным образом на интервале [2;4]*  **Sub vvodSngRnd16(ByRef a( ) As Single)**  **Dim i As Integer**  **For i = 0 To UBound(a)**  **a(i) = 2 + 2 \* Rnd( )**  **Next i**  **End Sub** |

*Рис. 4.7.2-2. Процедура формирования массива случайным образом*

**Примера 4.7.2-1**

|  |
| --- |
| *'Процедура форматного вывода массива типа Single в ListBox*  **Sub vivodSngMac17(ByRef a( ) As Single, ByVal L As ListBox)**  **Dim i As Integer**  **Dim m As String = ""**  **For i = 0 To UBound(a)**  **m = m + Format(a(i), "0.000") + Space(4)**  **Next i**  **If m ="" Then m = "массив пуст"**  **L.Items.Add(m)**  **End Sub** |

*Рис. 4.7.2-3. Процедура форматного вывода массива* **Single** *в* **ListBox**

**Примера 4.7.2-1**

### 4.7.3. Динамические массивы

Различают два способа распределения памяти: ***статическо*е** – на этапе компиляции до выполнения программы, и ***динамическое*** – в ходе выполнения программы. По умолчанию массив, границы которого заданы константными выражениями, считается статическим. Память для размещения такого массива выделяется на этапе компиляции программы и сохраняется за ним на весь период выполнения программы. Например:

|  |
| --- |
| **Dim Sigma(5) As Integer, m(3) As Single** |

Однако размеры массивов не всегда известны заранее, часто они определяются в ходе выполнения программы. Например, при табулировании значений функции количество хранимых значений в одномерном массиве определяется шагом дискретизации и диапазоном табулируемых значений функции, которые могут задаваться пользователем или вычисляться в процессе выполнения программы. Объявлять размерности массивов такими большими, что это будет заведомо достаточно для всех случаев, не всегда возможно и нерационально.

Одно из решений проблемы – выделять память под массив не на этапе компиляции – статически, а после определения его размера – динамически. В качестве размера массива может быть использована переменная, значение которой вычисляется или вводится перед объявлением массива:

|  |
| --- |
| *РазмерМассива = Выражение или РазмерМассива* **= Cint(TextBox1.Text)**  **Dim** *ИмяМассива(РазмерМассива)* **As** *Тип* |

Другое решение проблемы – разделить в программе объявление массива и определение его размера – выделение памяти под него.

При объявлении массива размер не указывается:

|  |
| --- |
| **Dim** *ИмяМассива( )* **As** *Тип* |

Значение размерности определяется позже (вычисляется или вводится) непосредственно перед его использованием, и тогда для выделения памяти уже объявленному массиву с указанием конкретной размерности массива используется оператор **ReDim** или **ReDim Preserve**:

|  |
| --- |
| **ReDim** *ИмяМассива (РазмерМассива),*  **ReDim Preserve** *ИмяМассива (РазмерМассива)* |

Таким образом, динамические массивы имеют весьма полезное свойство – их размеры могут изменяться в процессе выполнения программ. При этом оператор **ReDim** изменяет размер массива и очищает его (обнуляет его элементы), а оператор **ReDim Preserve** изменяет размер массива и сохраняет значения существующих элементов.

В следующем примере каждый раз при добавлении нового элемента к массиву происходит увеличение размера массива на единицу:

|  |
| --- |
| **n = n + 1**  **ReDim Preserve Mas(n)**  **Mas(n) = n + 4** |

Таким образом, для создания ***динамического массива*** его необходимо предварительно объявить, не указывая количество элементов массива:

|  |
| --- |
| **Dim Мас( ) As String** *'объявление динамического массива* |

Затем, в момент, когда необходимо распределить под него память, используется оператор **ReDim**:

|  |
| --- |
| **ReDim Мас(9)**  *' или*  **ReDim Preserve Мас(9)** |

Второй вариант используется для изменения размера массива и для сохранения содержимого.

На рис. 4.7.4-7, рис. 4.7.4-9 и рис. 4.7.4-10 приведены примеры программных кодов, использующие динамические массивы.

### 4.7.4. Базовые алгоритмы обработки одномерных массивов

**и примеры их программирования**

Типичными задачами работы с одномерными массивами являются ***определение факта наличия в них заданного элемента и отбор элементов, удовлетворяющих определённым условиям.*** В обоих случаях используется циклическое сравнение элементов массива с заданным образцом. Для определения факта наличия заданного образца в массиве достаточно единственного совпадения, после чего дальнейший просмотр прекращается. Если условие отбора может выполняться для нескольких элементов массива, то необходим просмотр всего массива до конца. Для досрочного прекращения просмотра элементов массива в цикле используется оператор   
**Exit For**.

В массивах больших размеров для сокращения времени просмотра применяется упорядоченное хранение элементов. Для числовых массивов используется упорядочение элементов по значению, для текстовых – по алфавиту (точнее, по значению кода символов). Упорядочение обычно производиться путём сортировки элементов неупорядоченного массива.

Из всего многообразия существующих задач, связанных с обработкой одномерных массивов, можно выделить следующие базовые алгоритмы:

* Нахождение суммы значений элементов массива при заданных условиях (Пример 4.7.4-1).
* Нахождение произведения значений элементов массива при заданных условиях (Пример 4.7.4-2).
* Нахождение максимального и минимального значений элементов массива и их индексов (Пример 4.7.4-3).
* Формирование нового массива из элементов одного или нескольких массивов в соответствии с заданными критериями (Примеры 4.7.4-3 – 4.7.4-4).
* Формирование массива в соответствии с определенными правилами (Пример 4.7.4-7).
* Вставка элемента по заданному критерию в массив   
  (Пример 4.7.4-5).
* Удаление элемента по заданному критерию из массива   
  (Пример 4.7.4-6).
* Обмен значений элементов массива по заданному критерию   
  (Пример 4.7.4-8).
* Сортировка (упорядочивание) элементов массива (Пример 4.7.4-9).

Пример 4.7.4-1. Разработать процедуру, которая вычисляет произведение положительных и сумму отрицательных элементов массива x().

Схема алгоритма и программный код процедуры приведены на   
рис. 4.7.3-1.

Напомним, что вычисление суммы и произведения обычно осуществляется по рекуррентным формул: **s0 = 0; si+1 = si + xi+1; p0 = 1;   
pi+1 = pi \* xi+1.**

|  |  |
| --- | --- |
|  | **Sub Pr741(ByRef x() As Single, \_**  **ByRef s As Single, ByRef p As Single)**  **Dim i As Integer**  **s = 0 : p = 1**  **For i = 0 To UBound(x)**  **If x(i) > 0 Then**  **p = p \* x(i)**  **Else**  **s = s + x(i)**  **End If**  **Next**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim s,p, a( ) As Single**  **vvodSngMac15(а)**  **vivodSngMac17(а, ListBox1)**  **Pr741(а, s, p)**  **vivod3(s, TextBox1) : Vivod3(p, TexBox2)**  **End Sub** |

*Рис. 4.7.4-1. Схема алгоритма и программный код процедуры* **Pr741()**

**Примера 4.7.4-1**

**Пример 4.7.4-2. Разработать процедуру, в которой вычисляется произведение ненулевых элементов вещественного массив a().**

Схема алгоритма и программный код приведены на рис. 4.7.4-2.

|  |  |
| --- | --- |
|  | **Function Pr742(ByRef a( ) As Single) As Single**  **Dim i As Integer, z Fs Single**  **z = 1**  **For i = 0 To Ubound(a)**  **If a(i) <> 0 Then z = z \* a(i)**  **Next i**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim z As Single**  **vvodSngMac15(a) : vivodSngMac17(a, ListBox1)**  **z = Pr742(a) : vivodSngMac17 (z, TextBox1)**  **End Sub** |

*Рис. 4.7.4-2. Схема алгоритма и программный код процедуры* **Pr742()**

**Примера 4.7.4-2**

**Пример 4.7.4-3**. Разработать процедуру-Function, которая находит максимальное значения элементов массива t().

Схема алгоритма и программный код приведены на рис. 4.7.4-3.

|  |  |
| --- | --- |
|  | **Sub Pr743(ByRef t() As Double) As Double**  **Dim i, n, j, k As Integer**  **Dim xmax As Single**  **n = UBound(t)**  **xmax = x(0)**  **For i = 1 To n**  **If x(i) > xmax Then**  **xmax = x(i)**  **End If**  **Next i**  **Return xmax**  **End Sub** |

*Рис. 4.7.4-3. Схема алгоритма и программный код процедуры* **Pr743()**

**Примера 4.7.4-3**

**Пример 4.7.4-4. Разработать процедуру-Function, которая находит индекс минимального значения элементов массива t().**

|  |  |
| --- | --- |
|  | **Sub Pr744(ByRef t() Double) As Integer** **Dim i, n, j, k As Integer**  **Dim xmin As Double**  **n = UBound(t)**  **xmin = x(0) : k = 0**  **`For i = 1 To n**  **If x(i) < xmin Then**  **xmax = x(i)**  **k = i**  **End If**  **Next i**  **Return к**  **End Sub** |

*Рис. 4.7.4-4. Схема алгоритма и программный код процедуры* **Pr744()**

**Примера 4.7.4-4**

**Пример 4.7.4-5. Разработать процедуру, которая в заданном массиве c() переставляет элементы с целыми значениями в начало массива.**

|  |  |
| --- | --- |
|  | **Sub Pr745(ByRef с() As Single)**  **Dim temp As Single, i, k As Integer**  **For i = 0 To UBound(c)**  **If c(i) - Fix(c(i)) = 0 Then**  **temp = c(k) : c(k) = c(i)**  **c(i) = temp : k = k + 1**  **End If**  **Next i**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim с(9) As Single**  **vvodSngMac15(с): vivodSngMac17(с,ListBox1)**  **Pr7475(с) : vivodSngMac17(с,ListBox2)**  **End Sub** |

*Рис. 4.7.4-5. Схема алгоритма и программный код процедуры* **Pr745()**

**Примера 4.7.4-5**

Для того чтобы переставить целые элементы в начало массива, в переменной **k** будем хранить номер элемента, в который переписывается очередное целое значение. Чтобы определить, является ли очередной элемент массива целым числом, проводится сравнение разности значения целой части очередного элемента и значения очередного элемента массива **c(i)** с нулем.

Целая часть значения **c(i)** выделяется с помощью функции **Fix()**. Если очередной элемент массива **c(i)** содержит целое значение, то производится обмен значений двух элементов массива **c(k)** и **c(i)** c помощью **temp**.

Схема алгоритма и программа приведены на рис. 4.7.4-5.

**Пример 4.7.4-6. Разработать процедуру-Sub, в которой необходимо сформировать массив c(), по следующему правилу:**

******

Схема алгоритма и программный код приведены на рис. 4.7.4-6.

|  |  |
| --- | --- |
|  | **Sub Pr746(ByRef c( )As Single)**  **Dim i, k As Integer**  **For i = 0 To UBound( c )**  **If i < 5 Then**  **c(i) = (i^3 - 4)/(i + 1)**  **Else**  **c(i) = (i^2 - 36) / i**  **End If**  **Next i**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim с(9) As Single**  **Pr746(с)**  **vivodSngMac17(с, ListBox1)**  **End Sub** |

*Рис. 4.7.4-6. Схема алгоритма и программный код процедуры* **Pr746()**

**Примера 4.7.4-6**

**Пример 4.7.4-7**. **Разработать процедуру-Sub, в которой необходимо сформировать массив y(), переписав в него положительные элементы исходного массива x().**

|  |  |
| --- | --- |
|  | **Sub Pr747(ByRef x() As Single, \_**  **ByRef y() As Single)**  **Dim i, n As Integer**  **For i = 0 To UBound(x)**  **If x(i) > 0 Then**  **ReDim Preserve y(n)**  **y(n) = x(i) : n = n + 1**  **End If**  **Next i**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim x( ), y( ) As Single**  **vvodSngMac15(x)**  **vivodSngMac173(x, ListBox1)**  **Pr747(x, y)**  **vivodSngMac17(y, ListBox2)**  **End Sub** |

*Рис. 4.7.4-7. Схема алгоритма и программный код процедуры* **Pr747()**

**Пример 4.7.4-7**

Для решения поставленной задачи необходимо проверить знак у всех элементов массива **х()**. При этом текущий индекс **n** формируемого массива **y()** меняется независимо от индекса **i** исходного массива **х()**. Индекс **n** увеличивается на единицу только при появлении положительного элемента **х()**.

Таким образом, после проверки всех элементов массива **х()** в переменной **n** будет содержаться число положительных элементов исходного массива.

Схема алгоритма и программный код приведены на рис. 4.7.4-7.

**Пример 4.7.4-8.** **Разработать процедуру-Sub, в которой необходимо из двух исходных массивов p() и r() с одинаковым числом элементов получить массив v() путем последовательного попарного переписывания в него элементов массивов p() и r().**

В данной задаче для формирования массива **v()** используется переменная k, которая представляет собой номер очередного элемента массива **v()**. В цикле одновременно заполняются два элемента массива **v()**: в элемент с номером k переписывается **i**-й элемент из массива **p()**, а в элемент с номером **(k+1)** переписывается **i**-й элемент из массива **r()**.

На рис. 4.7.4-8 приведены алгоритм и процедура и решения задачи.

|  |  |
| --- | --- |
|  | **Sub Pr748(ByRef p() As Single, \_**  **ByRef r() As Single,**  **ByRef v() As Single)**  **Dim i, k As Integer**  **For i = 0 To UBound(p)**  **v(k) = p(i)**  **v(k + 1) = r(i)**  **k = k + 2**  **Next**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim p(), r(), v() As Single**  **vvodSngMac15(p)**  **vivodSngMac17(p, ListBox1)**  **vvodSngMac15(r)**  **vivodSngMac17(r, ListBox2)**  **Pr748(p, r, v)**  **vivodSngMac17(v, ListBox3)**  **End Sub** |

*Рис. 4.7.4-8. Схема алгоритма и программный код процедуры* **Pr748()**

**Пример 4.7.4-8**

**Пример 4.7.4-9.** **Разработать программный код процедуры-Sub, который из массива вещественных чисел x(n) удаляет все отрицательные элементы и подсчитывает их количество k.**

Удаление всех отрицательных элементов массива реализуется в процедуре **Pr748()** по так называемому алгоритму «сжатия». Метод заключается в поиске удаляемого отрицательного элемента, фиксации его номера, а затем в последовательной перезаписи всех последующих элементов массива так, чтобы значение следующего **i+1** элемента записывалось на место предыдущего, и так до конца массива.

|  |  |
| --- | --- |
|  | **Sub Pr749(ByRef x() As Single, ByRef k As Integer)**  **Dim i, n, j As Integer**  **k = 0**  **i = 0**  **n = UBound(x)**  **Do**  **If x(i) < 0 Then**  **k = k + 1 *'сжатие массива***  **For j = i To n - 1**  **x(j) = x(j + 1)**  **Next**  **n = n - 1**  **ReDim Preserve x(n)**  **Else**  **i = i + 1**  **Loop Until i > n**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim n, k As Intege**  **n = CInt(InputBox("Введите" & \_**  **"количество элементов "))**  **Dim x(n - 1) As Double**  **vvodSngRnd16(x)**  **vivodSngMac17(x, ListBox1)**  **Pr749(x, k)**  **vivodSngMac17(x, ListBox2)**  **TextBox1.Text = CStr(k)**  **End Sub** |

*Рис. 4.7.4-9. Схема алгоритма и программный код процедуры Pr749()*

**Пример 4.7.4-9**

Из последовательности исчезает удаляемый элемент, однако последний элемент повторяется дважды, поэтому после выхода из внутреннего цикла по перезаписи элементов длина массива (число **n**) должна быть уменьшена на единицу. Для изменения размерности массива используется оператор **ReDim Preserve**.уменьшена на единицу. Для изменения размерности массива используется оператор **ReDim Preserve**.

Так как отрицательные элементы могут идти подряд, то **i+1** элемент, который перешел на место **i**-го, тоже может быть меньше нуля. Поэтому необходимо снова проверить текущий **i**-й элемент (бывший **i+1**) и, возможно, тоже удалить его «сжатием». Переход к следующему элементу по параметру **i** (т.е. увеличение **i** на 1) происходит, только если проверяемый **i**-й элемент оказался неотрицательным, поэтому алгоритм «сжатия» может

быть реализован с помощью внешнего итеративного (не регулярного) цикла.

Ввод исходного одномерного массива случайными числами из диапазона **[-10;5]** осуществляет процедура **vvod()**, а вывод массива на форму в элемент управления **ListBox** осуществляет процедура **vivod()**, которая по вычисляемому методом **Length** количеству элементов массива проводит проверку, не является ли выводимый массив пустым.

На рис. 4.7.4-9 приведены алгоритм и программа решения задачи.

**Пример 4.7.4-10. Разработать процедуру-Sub, действие которой заключается в удалении из массива x() одинаковых элементов.**

|  |  |
| --- | --- |
|  | **Sub Pr7410(ByRef x() As Double)**  **Dim i, n, j, k As Integer**  **i = 0**  **n = UBound(x)**  **Do While i < n**  **j = i + 1**  **Do**  **If x(i) = x(j) Then**  **For k = j To n - 1**  **x(k) = x(k + 1)**  **Next**  **n = n - 1**  **Else**  **j = j + 1**  **If**  **Loop While j <= n**  **i = i + 1**  **Loop**  **ReDim Preserve x(n)**  **End Sub**  **Private Sub Button1\_Click(…)**  **Dim n As Integer**  **n = CInt(InputBox("введите кол." \_**  **&"элементов исходного массива"))**  **Dim x(n - 1) As Double**  **vvod(x, n) : vivod(x, ListBox1)**  **Pr7410(x) : vivod(x, ListBox2)**  **End Sub** |

*Рис. 4.7.4-10. Схема алгоритма и программный код процедуры* **Pr7410()**

**Пример 4.7.4-10**

Решения задачи заключается в последовательном сравнении каждого элемента исходного массива со всеми остальными. В цикле происходит формирование массива **x(j)**. Если элемент массива **x(i)** равен элементу **x(j)**, то **j**-й элемент удаляется, а длина массива уменьшается на единицу.

Алгоритм «сжатия» для получения массива с уникальными элементами и программа, реализующая данный алгоритм, приведены на рис. 4.7.4-10.

**Пример 4.7.4-11. Разработать процедуры-Sub, в которых осуществляется сортировка по убыванию значений элементов массива.**

Сделаем несколько общих замечаний по поводу сортировки элементов массива.

Во-первых, стоит отметить, что под упорядочиванием (сортировкой) массива подразумевается процесс перестановки элементов массива с целью размещения элементов массива в определенном порядке. Наиболее часто применяются следующие способы упорядочивания массивов.

1. Упорядочивание по возрастанию. Каждый следующий элемент в массиве должен быть больше предыдущего.
2. Упорядочивание по убыванию. Каждый следующий элемент в массиве должен быть меньше предыдущего.
3. Упорядочивание по неубыванию. Каждый последующий элемент в массиве должен быть больше или равен предыдущему.
4. Упорядочивание по невозрастанию. Каждый последующий элемент в массиве должен быть меньше или равен предыдущему.

Во-вторых, необходимо отметить, что главным показателем качества алгоритма сортировки считается его быстродействие, реальные программы работают с большими объемами данных, сохраняемыми в массивах. Повышение быстродействия, под которым понимается уменьшение количества просмотров массива от начала до конца (проходов) с целью определения упорядоченности, может быть достигнуто за счет применения различных методов сортировки.

Одним из простейших методов сортировки массива является *сортировка* *прямого выбора*, который нельзя отнести к высокоэффективным, так как в этом методе используется фиксированное число проходов, а оно не зависит от уровня первоначальной упорядоченности массива.

Можно отметить также, что при разработке программ, связанных с сортировкой элементов массивов, при выводе результирующего массива можно использовать дополнительный цикл, который позволяет пользователю увидеть количество проходов, необходимых для окончательной сортировки массива.

Рассмотрим «сортировку выбором» (метод «прямого выбора»).

Алгоритм и программа сортировки массива по убыванию методом «***прямого выбора***» приведены на рис. 4.7.4-11.

|  |  |
| --- | --- |
|  | **Sub Pr7411(ByRef x() As Single)**  **Dim i, n, j, k As Integer**  **Dim xmax As Single**  **n = UBound(x)**  *'Внешний цикл сортировки*  **For i = 0 To n - 1**  **xmax = x(i)**  **k = i**  *'Поиск xmax и k*  **For j = i + 1 To n**  **If x(j) > xmax Then**  **xmax = x(j)**  **k = j**  **End If**  **Next j**  **x(k) = x(i)**  **x(i) = xmax**  **Next i**  **End Sub** |

*Рис. 4.7.4-11. Схема алгоритма и программный код процедуры* **Pr7411()**

**Пример 4.7.4-11**

Суть этого метода сортировки состоит в следующем.

Сортировка элементов массива по убыванию производится с помощью вложенных циклов. Сначала осуществляется поиск наибольшего элемента массива и его индекса среди всех элементов, начиная с 1-го. Найденный максимальный элемент меняется с первым элементом. Затем вновь осуществляется поиск наибольшего элемента массива и его индекса, но уже со второго элемента, и найденный максимальный элемент обменивается со 2-м элементом, и т.д. Таким образом, число найденных максимумов (поисков) равно **n**. При этом во внешнем цикле, начиная с первого и до предпоследнего элемента массива, сначала очередной элемент принимается за максимальный, а затем, после выполнения внутреннего цикла, обеспечи­вается заполнение этого очередного элемента массива наибольшим «среди оставшихся элементов».

Внут­ренний цикл осуществляет перебор и сравнение последующих эле­ментов, начиная с **(i+1) -**го до последнего элемента массива. В результате выполнения внутреннего цикла, в переменной **xmax** фиксируется значение наибольшего элемента, а в переменной **k** - его номер. Далее, во внешнем цикле выполняется перестановка найденного максимального эле­мента на место очередного **i**-го элемента массива.

Рассмотрим сортировку **элементов массива** *модифицированным методом «пузырька» (прямого обмена)***.**

Алгоритм и программа упорядочения массива по модифицированному методу «***пузырька***» приведены на рис. 4.7.4-12.

|  |  |
| --- | --- |
|  | **Sub Pr7412(ByRef x( ) As Single)**  **Dim i, n, j As Integer**  **Dim xx As Single**  **n = UBound(x)**  **For i = 0 To n-1**  **For j = i + 1 To n**  **If x(i) > x(j) Then**  **xx = x(i)**  **x(i) = x(j)**  **x(j) = xx**  **End If**  **Next j**  **Next i**  **End Sub** |

*Рис. 4.7.4-12. Схема алгоритма и программный код процедуры* **Pr7412()**

**Пример 4.7.4-11**

Суть метода заключается в последовательном сравнении первого элемента массива со вторым, третьим и т.д. Если значения в паре расположены не в порядке возрастания (т.е. по убыванию), то их меняют местами. После первого просмотра элемент массива с наименьшим значением становится первым. Далее все элементы, начиная с третьего, сравниваются со вторым, в результате чего элемент со значением, следующим за минимальным элементом, становится на второе место.

Продолжая таким же образом, вплоть до предпоследнего элемента, сортируют весь массив.

Обратите внимание, что в схемах алгоритмов обмен значениями обозначается символом «**↔**», а в примере обмен реализован через дополнительную переменную **xx**.

### 4.7.5. Элементы управления для работы со списками

Для работы со списками используются элементы управления **ListBox** и **ComboBox*.*** Элемент управления **ListBox** представляет собой список значений, расположенных в одну или несколько колонок, из которых пользователь может выбрать одно из предложенных значений. Значения в списке могут размещаться в одну или несколько колонок в зависимости от значения свойства MultiColumn***.*** Если элементы списка расположены в нескольких колонках, с помощью свойства ColumnWidth можно изменить заданную по умолчанию ширину колонок. В том случае если элементы списка не помещаются в выделенную для них в форме область, появляются полосы прокрутки. Чтобы полосы прокрутки всегда отображались, необходимо присвоить значение *True* свойству ScrollAlwaysVisible.

Элемент управления **ComboBox**, кроме того, имеет дополнительное текстовое поле, в котором пользователь может ввести значение, позволяющее ему осуществить выбор значения из списка. Списки типа **ComboBox** называют раскрывающимися или полями со списком. Раскрывающимися их называют потому, что для выбора значения из списка необходимо список открыть, нажав кнопку со стрелкой.

Формирование списков может быть статическим (средствами визуального программирования) или динамическим (в процессе выполнения программы). Для статического заполнения используется свойство-коллекция Items. Программное добавление в конец списка производится методом   
**Add( )** этой коллекции.

|  |
| --- |
| *ИмяСписка***.Items.Add(***Выражение***)** |

Для добавления строки в заданное место в списке используется метод **Insert():**

|  |
| --- |
| *ИмяСписка***.Items.Insert(***ИндексВСписке***,** *Выражение***)** |

Индекс в ***списке*** начинается с 0. Количество элементов в списке можно получить через свойство Countколлекции **Items**:

|  |
| --- |
| **Dim** *КоличествоЭлементов* **As Integer** = *ИмяСписка*.**Items.Count** |

Доступ к элементу списка может производиться по индексу:

|  |
| --- |
| *ВыбранныйЭлемент = ИмяСписка.***Items***(ИндексЭлемента)* |

Индекс элемента может быть определен по его значению:

|  |
| --- |
| *ИндексЭлемента =* **ComboBox1.FindString***(ЗначениеЭлемента)* |

Удаление элемента в списке с указанным значением производится методом **Remove(),** например:

|  |
| --- |
| **ListBox1.Items.Remove***(ЗначениеЭлемента)* |

Удаление элемента в списке с указанным индексом производится ме тодом **RemoveAt( )**:

|  |
| --- |
| **ListBox1.Items RemoveAt(***ИндексЭлемента***)** |

Удаление всех элементов списка производится методом **Clear()**. Например:

|  |
| --- |
| **ListBox1.Items.Clear( )** |

Значение элемента, выбранного пользователем, указано в свойстве Text:

|  |
| --- |
| **MsgBox(**"Пользователь выбрал **"& ListBox1.Text)** |

Индекс элемента, выбранного пользователем, указан в свойстве SelectedIndex:

|  |
| --- |
| **MsgBox("Выбран" & ListBox1.SelectedIndex& "-й элемент Cписка")** |

В комбинированном (раскрывающемся) списке дополнительно программируется работа с полем ввода. Например, добавление введенного пользователем значения в конец списка:

|  |
| --- |
| **ComboBox1.Items.Add(ComboBox1.Text)** |

Пример добавления введенного элемента в указанную позицию списка:

|  |
| --- |
| **ComboBox1.Items.Insert**(*ИндексЭлемента*, ComboBox1.Text) |

Список может быть отсортирован по возрастанию значений кодов:

|  |
| --- |
| **ComboBox1.Sorted** = **True** |

### 4.7.6. Тестовые задания

1. **Массив - это**
2. совокупность данных одного типа, объединенных общим именем
3. совокупность данных одного типа
4. набор индексированных данных
5. набор разных данных
6. набор однотипных файлов на диске
7. **Индексом массива может быть**
8. выражение любого типа
9. любое целочисленное выражение
10. только целочисленные переменные
11. переменные любого типа
12. **На размерность массива указывает**
13. значения индексов
14. суммарное количество индексов
15. количество индексов
16. сумма значений индексов
17. **Оператор Dim**
18. резервирует область памяти для элементов массива
19. резервирует имя для элементов массива
20. выстраивает элементы массива в линейку
21. подсчитывает количество элементов массива
22. **Сортировка массива – это**
23. упорядочивание элементов массива либо по возрастанию, либо по убыванию
24. перезапись элементов массива в обратном порядке
25. удаление нулевых элементов массива
26. в списке нет правильного ответа
27. **Определите правильный результат выполнения следующего фрагмента програмы**

|  |
| --- |
| **Dim a (5) As Single**  **For i = 0 To 5**  **a(i) = i**  **Next i**  **TextBox1.Text = CStr(a(i))** |

1. 5
2. сообщение об ошибке
3. 6
4. 0
5. **После выполнения фрагмента программы**

|  |
| --- |
| **Dim a (5) Single**  **For i = 1 To 7**  **TextBox1.Text = TextBox1.Text & CStr(a(i)) & " "**  **Next i** |

массив будет напечатан

1. в столбец
2. в строку
3. со сдвиг по заданному критерию
4. нет верного ответа
5. **Фрагмент программы**

|  |
| --- |
| **For i = k To n - 1**  **a(i) = a(i+1)**  **Next i** |

выполняет

1. сжатие массива «сдвигом влево»
2. сжатие массива «сдвигом вправо»
3. удаление первых **K** элементов массива
4. удаление последних **N-K** элементов массива
5. **Что выполняет следующий фрагмент программы**

|  |
| --- |
| **For i = 0 To n-1**  **For j = i + 1 To n**  **If a(i) < a(j) Then r = a(i) : a(i) = a(j) : a(j) = r**  **Next j**  **Next i** |

1. упорядочивает массив по убыванию
2. упорядочивает массив по возрастанию
3. меняет максимальный и минимальный элементы местами
4. формирует массив из положительных элементов массива
5. **Фрагмент программы**

|  |
| --- |
| **i=0**  **Do**  **If a(i) = 0 Then**  **For j = i To n-1**  **a(j) = a(j+1)**  **Next j**  **n = n-1**  **Else**  **i = i + 1**  **End If**  **Loop While i <= n** |

1. удаляет из массива нулевые элементы
2. перемещает нулевые элементы влево
3. перемещает нулевые элементы вправо
4. оставляет все по-прежнему
5. **Фрагмент программы подсчитывает**

|  |
| --- |
| **k = 0 : s = 0**  **For i = 0 To 5**  **If a(i) = 0 Then k = k + 1 : If a(i) < 0 Then s = s + a(i)**  **Next i**  **TextBox1.Text=CStr(s) & " " & CStr(k)** |

1. сумму элементов, меньших 0, и количество положительных элементов
2. количество всех элементов и сумму положительных элементов
3. количество элементов, равных 0, и сумму отрицательных элементов
4. произведение отрицательных элементов и сумму элементов, равных 0
5. **Следующий фрагмент программы**

|  |
| --- |
| **j = -1**  **For i = 0 To n**  **If a(i) > 0 Then**  **j = j + 1 : m(j) = i**  **End If**  **Next i** |

1. записывает индексы положительных элементов массива **a()** в массив **m**
2. переписывает массив, **а()** в массив **m**
3. переписывает в массив m положительные элементы массива **a()**
4. изменяет порядок расположения элементов в массиве **a()**
5. **В результате работы программы на экран будет отображено значение переменной N, которая является**

|  |
| --- |
| **DIM B(100), М, I, N As Integer**  **Sub Button1\_Click( … )**  **For I = 0 To 100**  **B(I) = INT(RND\*100)**  **Next I**  **M = B(1)**  **For I = 1 Tо 100**  **If B(I)< M Then**  **M = B(I) : N = I**  **End If**  **Next I**  **TextBox1.Text = CStr(N)**  **End Sub** |

1. значением минимального элемента массива
2. индексом минимального элемента массива
3. значением максимального элемента массива
4. индексом максимального элемента массива
5. **Следующий фрагмент программы выполняет**

|  |
| --- |
| **k = 2 : n = 10**  **For i = k To n-1**  **a(i) = a(i+1)**  **Next i** |

1. Удаление **k**-го элемента массива
2. расширение массива «сдвигом вправо»
3. удаление **k** первых элементов массива
4. удаление последних **n-k** элементов массива
5. **В результате выполнения следующего фрагмента программы происходит**

|  |
| --- |
| **j = -1**  **For i = 0 To 10**  **If a(i) < 0 Then**  **j = j + 1 : b(j) = a(i)^2**  **End If**  **Next i** |

1. формирование массива **b()** из квадратов отрицательных элементов массива **a()**
2. упорядочение массива, **а()** по возрастанию
3. формирование нового массив **b()** по возрастанию
4. формирование нового массив **b()** по убыванию

## 4.7.7. Лабораторная работа по теме «Программирование алгоритмов формирования и обработки одномерных массивов»

**Цель работы:** овладение практическими навыками разработки и программирования вычислительного процесса, связанного с программированием алгоритмов формирования и обработки одномерных массивов, уяснение способов ввода, заполнения и вывода одномерных массивов, классических приемов обработки одномерных массивов.

### Вопросы, подлежащие изучению

1. Способы описания и объявления одномерных массивов.
2. Возможности резервирования памяти и хранения элементов массива.
3. Способы задания значений элементам массива: присваиванием; инициализацией; вводом по запросу с клавиатуры; заполнением массива данных случайными числами в заданном диапазоне.
4. Базовые алгоритмы обработки одномерных массивов: вычисление суммы (произведения) элементов массива; нахождение номера (и значения) минимального (максимального) значения элемента массива; формирование нового массива из исходного массива по заданному критерию; сортировка элементов массива от большего к меньшему (или от меньшего к большему); удаление элементов массива, имеющих равные значения (сжатие массива); удаление элементов массива по заданному критерию (сжатие по признаку).
5. Методы класса **Array**.
6. Методы работы с элементами управления **ListBox** и **ComboBox**.

### Общее задание на разработку проекта

1. ***Изучите вопросы программирование алгоритмов формирования и обработки одномерных массивов*** *(Тема 7).*
2. Создайте приложениес именем ***Проект-4.7***.
3. ***Выберите вариант задания*** *из табл. 4.7.7-1.*
4. ***Разработайте графический интерфейс*** *пользователя.*
5. ***Разработайте схемы алгоритмов*** *процедур пользователя в соответствии с индивидуальным заданием, предварительно проведя формализацию.*
6. ***Напишите программный код*** *процедур по разработанным алгоритмам.*
7. ***Разработайте*** *проект приложения, решающий поставленную задачу, который состоит из интерфейса пользователя и соответствующего программного кода, а также написанных ранее процедур обработки, ввода и вывода элементов массива. Все пользовательские процедуры должны находиться в модуле формы. Обмен данными между пользовательскими процедурами должен осуществляться через параметры, без использования глобальных переменных.*
8. ***Подготовьте входные массивы чисел*** *для решения задачи, если исходные данные не заданы.*
9. ***Выполните приложение*** *и получите результат.*
10. ***Докажите правильность*** *результата.*

### Варианты индивидуальных заданий

Таблица 4.7.7-1

|  |  |
| --- | --- |
| **1)** | Ввести одномерный массив x= {-1.5, 0.1, 12, 0, -2.2, 0.5, -1, 0.3}.  Заменить в нем все отрицательные элементы значением минимального элемента, а все положительные – максимальным значением. |
| **2)** | Ввести одномерный массив a = {5, -2, 0, 3, 4, 12, 7}.  Вычислить и вывести среднее арифметическое значение положительных элементов массива и заменить этим значением те элементы массива, которые больше среднего арифметического . |
| **3)** | Ввести одномерный массив x = {-1.5, 0, 0.8, 2.2, 3, 0.5, 0.1}.  Переписать элементы массива, принадлежащие отрезку [-1;1**]**, в массив y и найти сумму элементов, расположенных после максимального элемента в массиве y. |
| **4)** | Ввести одномерные массивы a(5) и b(5**)**, состоящие из произвольных чисел и определить в каком из массивов больше положительных элементов. Получить и вывести новый массив **с**, состоящий из положительных элементов массивов a и b. |
| **5)** | Ввести одномерный массив b = {7.35, 0.12, -7, 3.12, 2.87, -4.12, 5.32, 0, 6.5}.  Определить и вывести максимальный элемент массива и его номер. Сформировать новый массив из элементов одного знака, число которых больше |
| **6)** | Ввести одномерный массив a={2.35,-4.15,0,-3.1, 7.8, 6.3,-3.05,1.5}.  Найти и вывести среднее геометрическое положительных элементов массива a и индекс элемента, наиболее близкого к среднему геометрическому. Затем упорядочить массив по убыванию. |
| **7)** | Сформировать одномерный массив a(10**)** из случайных чисел, принадлежащих отрезку[-2, 6] и вычислить среднее арифметическое тех его элементов, значения которых не превышают заданного числа z. Заменить отрицательные элементы массива найденным средним арифметическим |
|  |
| **8)** | В одномерном массиве, состоящем из 20 вещественных элементов (значения элементов массива задать случайными числами из интервала [-5, 8]), вычислить  сумму элементов массива, расположенных до последнего положительного элемента. Сформировать из этих элементов новый массив. |
| **9)** | Ввести произвольно одномерный массив из 10 элементов. Найти максимальный и минимальный элементы массива и поменять их местами. В полученном массиве найти количество элементов, расположенных до первого отрицательного элемента. |
| **10)** | Ввести одномерные массивы x = {4.1, 16, 0, -3.2, 12}иY = {4, 5.1, 6}.  Объединить их в один массив z, поместив элементы массива y между третьим и четвертым элементами массива x. В новом массиве z найти сумму элементов, расположенных до максимального элемента. |
| **11)** | Ввести одномерные массивы z = {0, 1.6, 6.4, 3.8, -7, 1, -2**}** и a = {5,4,6.4,1}.  Найти среди элементов массивов aи z два одинаковых элемента с наименьшими индексами и вывести их значения и индексы. Элементы, расположенные между найденными числами, записать в новый массив. |
| **12)** | Ввести одномерный массив n = {3, 5, 7, 9, -11, 13, 15}.Переставить элементы массива **n** в обратном порядке и найти в нем произведение элементов, расположенных после минимального элемента. |
| **13)** | Ввести одномерные массивыx = {-6, 0.5, 0.12, 13, -10.1} и y = {13, 2.1, 14, 6, -2}. Создать одномерный массив r( **)** такой, что элементы с нечетными номерами являются элементами массива x, с четными номерами - массива y.  Вывести массив r. В новом массиве r найти произведение элементов, расположенных до минимального элемента. |
| **14)** | Ввести одномерный массив l = {13, 4, -2, 6, 7, -1, -5, 2, -3, 4}.  Вычислить и вывести m[0]n[0]+m[1]n[1]+…+m[k]n[k], где m[0], m[1],…m[p] - отрицательные элементы массива l**,** взятые в порядке их следования; n[0], n[1],…n[q] – положительные элементы массива l, взятые в обратном порядке их следования; k = min{p,q}. |
| **15)** | В одномерном массиве, состоящем из 20 вещественных элементов (значения элементов массива задать случайными числами из интервала [-3, 10]), вычислить  сумму элементов массива, расположенных между первым и последним отрицательными элементами. Сформировать из этих элементов новый массив. |
| **16)** | В одномерном массиве, состоящем из 15 вещественных элементов (значения элементов массива задать случайными числами из интервала [-10, 10]), вычислить произведение элементов массива, расположенных между максимальным и минимальным элементами. Сформировать из этих элементов новый массив. |
| **17)** | Преобразовать одномерный массив, состоящий из 20 целых элементов (значения элементов массива задать случайными числами из интервала [-4, 7]), таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине — элементы, стоящие в четных позициях. В полученном массиве найти минимальный элемент и его номер. |
| **18)** | Ввести одномерный массив m = {6, 10, 7, 14, 12, 12,-2, 3,-9, 6,-10}.  Вычислить и вывести количество и сумму тех элементов массива, которые делятся на2 и не делятся на 3. Сформировать из этих чисел массив k. |
| **19)** | В одномерном массиве, состоящем из 20 элементов целого типа (значения элементов массива задать случайными числами из интервала [-15, 15]), вычислить сумму элементов массива, расположенных между первым и последним положительными элементами. Сформировать из этих элементов новый массив |
| **20)** | Ввести одномерный массив l = {7, 6, 15, 17, 12, -12, 4, 0, -10, -22}.  Заменить в массиве нулями те элементы, модуль которых при делении на 5 дает в остатке 2. В полученном массиве найти максимальный элемент и его номер. |
| **21)** | Ввести одномерный массив k = {1, 2, 3, 4, 6, 5, 8, 9, 10}.  Если элементы массива образуют возрастающую последовательность вывести сообщение "ДА"; в противном случае - сообщение "НЕТ". Сжать массив, удалив из него все элементы, принадлежащие отрезку [a, b**].** |
| **22)** | Ввести упорядоченный массив q = {1.5, 2, 3.1, 4.2, 6, 7.5, 8.3, 9}.  Удалить из массива элемент с задаваемым индексом ***k***, а затем вставить элемент с вводимым значением ***s*** так, чтобы не нарушилась упорядоченность.  Вывести полученный массив. |
| **23)** | В одномерном массиве, состоящем из 15 вещественных элементов (значения элементов массива задать случайными числами из интервала [-7, 10]), вычислить  сумму положительных элементов массива. Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 4, а потом — все остальные. |
| **24)** | Ввести одномерный массив b={-15.1 ,0.8, 32.3, 7.5,-1.5, 2.4, -6.3, 15.5}.  Подсчитать и вывести среднее арифметическое значение элементов массива и количество элементов, меньших среднего арифметического, записывая их в новый массив. |
| **25)** | В одномерном массиве, состоящем из 20 вещественных элементов (значения элементов массива задать случайными числами из интервала [-3, 9]), вычислить сумму элементов массива, расположенных между первым и вторым отрицательными элементами. Сформировать из этих элементов новый массив |
| **26)** | Ввести одномерный массив m = {14, 6, 3, 0, 7, 12, -3, 1, 5, 2}.  Вычислить и вывести произведение элементов массива, кратных 3. Сформировать из этих чисел массив k |
| **27)** | Ввести одномерный массив m = {-1, 0, 10, -3, -5, 6, -2, 3, 4}. Вычислить сумму элементов с нечетными номерами. Сформировать и вывести массив n**,** элементами которого являются индексы положительных элементов массива m**.** |
| **28)** | Ввести два одномерных массива a= {-2, 0, -3.1, 4.6, -1}, b = {4, 7, -9.1, 1.2, -0.3}**.**  Сформировать из элементов массивов a и b массив z. В новом массиве z найти произведение элементов, расположенных после максимального по модулю элемента. |
| **29)** | В одномерном массиве, состоящем из 20 целых элементов (значения элементов массива задать случайными числами из интервала [-5, 5]), вычислить произведение элементов массива с четными номерами. Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы массива, а потом — все отрицательные. |
| **30)** | Ввести одномерный массив y = {2.5, -4.9, 10.2, -7.12, 3.1, -2, 6}.  Сформировать из него новый массив z, элементами которого будут являться отрицательные элементы массива **y**, и упорядочить по возрастанию массив z. |
| **31)** | Ввести одномерный массив x = {-1, 2, 3, 4, -5, 0,17, -6, 8, 9}.  Переписать в массив yподряд положительные элементы массива x. Подсчитать в массиве y количество элементов, которые являются простыми числами. |
| **32)** | Ввести одномерный массив g = {-3.1, 2.8, 0, 5, 7.7, -7.5, 0, 7.6, 3, 0}.Подсчитать произведение отрицательных элементов массива. Преобразовать массив таким образом, чтобы сначала располагались все элементы равные нулю, а потом - все остальные. |
| **33)** | Ввести одномерный массив k = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Переставить пары элементов k(i), k(i+1),где i = 0, 2, 4, 6, 8. Вывести полученный массив |
| **34)** | Ввести одномерный массив x = {6,3.8, -9.3, 2.87,16, 5, 0.2, -3.1,1,10}**.**  Сформировать массив Y, вычислив его элементы по правилу: |
| **35)** | Ввести одномерный массив z = {-2, 0, 3.5, 7, -12, 5, -1, 3}.  Расположить в массиве **r** сначала положительные элементы, а затем  неположительные элементы массива z. Вывести массив r**.** |

### Содержание отчёта

1. Тема и название лабораторной работы.
2. Фамилия, имя студента, номер группы, номер варианта.
3. Задание на разработку проекта.
4. Формализация и уточнение задания.
5. Элементы, разрабатываемого проекта:
6. графический интерфейс пользователя;
7. таблица свойств объектов;
8. схема алгоритма решаемой задачи;
9. программный код проекта.
10. Результаты выполнения проектов.
11. Доказательство правильности работы программы.

### Пример выполнения задания

1. **Тема и название лабораторной работы:**

Программирование алгоритмов формирования и обработки одномерных массивов.

Заполнение и преобразование одномерного массива различными   
 способами.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

Введите значения одномерного массива натуральных чисел **a(n)**, используя различные алгоритмы заполнения массива, и сформируйте его. Далее осуществите преобразование сформированного массива, используя различные методы обработки одномерных массивов, и вычислите одну из заданных характеристик сформированного массива

1. **Формализация и уточнение задания:**

В разрабатываемом приложении ввод значений исходного массива (процедура **vvod()**) может осуществляться одним из следующих способов: вводом с клавиатуры; с помощью датчика случайных чисел; формированием по формуле.

Преобразование исходного массива может осуществляться несколькими способами: сортировка по возрастанию методом «пузырька»; сортировка по убыванию методом «выбора»; удаление отрицательных элементов.

1. **Элементы, разрабатываемого проекта:**
2. **Графический интерфейс пользователя:**

Разработайте форму приложения, которая может иметь такой же

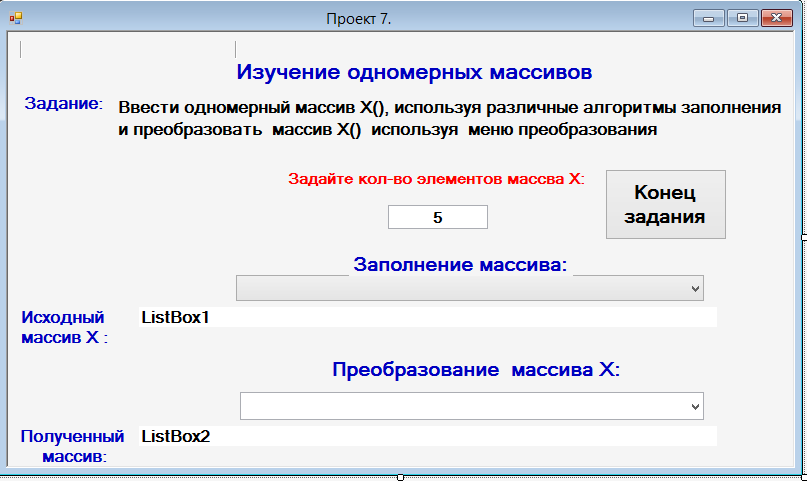
вид, как на рис. 4.7.7-1.

Для обеспечения возможности выбора способов ввода и обработки массива используйте элементы управления **ComboBox** (поле со списком или раскрывающийся список).

По своим функциям список типа **ComboBox** совмещает функции списка **ListBox** и текстового поля **TextBox**, поэтому из списка можно не только выбирать, но и вводить данные в находящееся в верхней части поле ввода. Элементы могут добавляться в список и удаляться из него во время разработки приложения с помощью свойства Items. Программные подобные действия осуществляются с использованием методов коллекции **Items** (например, **Add()** – добавление, **Remove()** – удаление, **Insert()** – вставка), элемента управления **ComboBox**. Метод **Clear()** удаляет все элементы из списка.

Также надо обратить внимание на расположенное в окне **Propertie**s свойство SelectionMode. С его помощью устанавливается количество выбираемых элементов. Например, при значении этого свойства *One* (один) в списке можно выделить только одну строку.

Для доступа к элементам списка используется свойство Items. Значения этого свойства являются массивом, размер которого равен количеству элементов в списке, а обращение к элементу осуществляется через индекс, определяющий его положение в списке, причем нумерация начинается с нуля. Например, значение первого элемента списка будет равно **ComboBox1.Items(0*).***



*Рис. 4.7.7-1. Форма проекта* **Проект 7-1*:***

*Заполнение и преобразование одномерного массива*

1. **Таблица свойств объектов:**

Установите и сведите в табл. 4.7.7-2 свойства объектов.

Таблица 4.7.7-2

|  |  |  |
| --- | --- | --- |
| **Имя**  **объекта** | **Свойство** | **Значение свойства** |
| **Label1** | ForeColor | *Red(Красный)* |
| Font | *Times New Roman, Bold (Жирный), 10-point (10 пунктов)* |
| AutoSize | *True* |
| Name | *Label1* |
| Text | *1.* |
| **Label2** | ForeColor | *Red(Красный)* |
| Font | *Times New Roman, Bold (Жирный), 10-point (10 пунктов)* |
| AutoSize | *True* |
| Name | *Label2* |
| Text |  |
| **Label3** | Font | *Times New Roman, Bold (Жирный), 14-point (14 пунктов)* |
| ForeColor | *Синий* |
| Name | *Label3* |
| Text | *Изучение одномерных массивов* |
| **Label5** | Font | *Times New Roman, Bold (Жирный), 10-point (10 пунктов)* |
| Name | *Label5* |
| Text | *Введите одномерный массив Х(), используя различные способы заполнения и преобразуйте его, используя алгоритмы преобразования* |
| **Label8** | ForeColor | *Red(Красный)* |
| Name | *Label8* |
| Text | *Задайте количество элементов массива х:* |
| **Label10** | Font | *10 пунктов* |
| Name | *Label10* |
| Text | *Заполнение массива* |
| **Label11** | AutoSize | *False* |
| ForeColor | *Синий* |
| Name | *Label11* |
| Text | *Преобразование массива х:* |
| **Label6** | Font | *Times New Roman, Bold (Жирный), 10-point (10 пунктов)* |
| Name | *Label6* |
| Text | *Исходный массив Х* |
| **Label7** | Font | *Times New Roman, Bold (Жирный), 10-point (10 пунктов)* |
| Name | *Label7* |
| Text | *Полученный массив* |
| **ComboBox1** | Items  (коллекция) | *Ввод массива с клавиатуры*  *Ввод массива по генератору случайных чисел*  *Ввод массива по формуле* |
| Name | *ComboBox1* |
| **ComboBox2** | Items  (коллекция) | *Сортировка по возрастанию методом пузырька*  *Сортировка по убыванию методом выбора*  *Удаление отрицательных элементов* |
|  | Name | *ComboBox2* |
| **ListBox1** | Name | *ListBox1* |
| **ListBox2** | Name | *ListBox2* |
| Text |  |
| **Button1** | Name | *Button1* |
| Text | *Конец задания* |

1. **Схемы алгоритмов решаемой задачи:**

Схемы алгоритмов отдельных процедур приведены в **Теме 4.7**.

1. Программный код проекта:

Программный код разработанного проекта приведен на   
 рис. **П.1.7**-2.

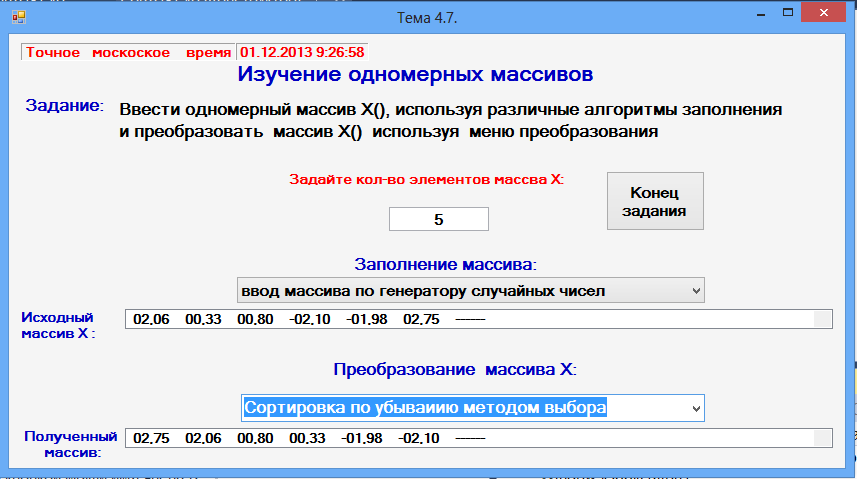
|  |
| --- |
| **Option Strict On**  **Option Explicit On**  *'Imports System.Math*  **Public Class Проект7**  **Dim x() As Single**  *'Процедура ввода массива*  **Sub vvod(ByRef x() As Single, ByVal index As Integer)**  **Dim y, r As String**  **Dim i As Integer**  **Select Case index**  **Case 0** *'ввод с клавиатуры*  **For i = 0 To UBound(x)**  **r = Str(i)**  **y = InputBox("элемент массива X("+ r +")=", \_**  **"Ввод значений элементов массива X()")**  **x(i) = CSng(Val(y))**  **Next i**  **Case 1** *'ввод генератором случ. чисел*  **For i = 0 To UBound(x)**  **x(i) = Rnd(1) \* 10 - 5**  **Next i**  **Case 2** *'ввод по формуле*  **For i = 0 To UBound(x)**  **x(i) = i \* 5 - 15**  **Next i**  **End Select**  **End Sub**  *'Процедура вывода массива*  **Public Sub PrintL(ByRef x() As Single,ByRef LB As ListBox)**  **Dim i As Integer**  **Dim y As String = " "**  **For i = 0 To UBound(x)**  **y = y + Format(x(i), "00.00") + Space(4)**  **Next i**  **y = y + "------"**  **LB.Items.Add(y)**  **End Sub**  *'Процедура сортировки эл-тов массива по возрастанию*  **Public Sub sort1(ByRef x() As Single)**  **Dim Box As Single**  **Dim i, j As Integer**  **For i = 0 To UBound(x) - 1**  **For j = i + 1 To UBound(x)**  **If x(i) > x(j) Then**  **Box = x(j) : x(j) = x(i) : x(i) = Box**  **End If**  **Next j**  **Next i**  **End Sub**    *'Процедура сортировки эл-тов массива по убыванию*  **Public Sub sort2(ByRef x() As Single)**  **Dim i, j, m As Integer**  **Dim xmax As Single**  **For i = 0 To UBound(x) - 1**  **xmax = x(i) : m = i**  **For j = i + 1 To UBound(x)**  **If x(j) > xmax Then**  **xmax = x(j)**  **m = j**  **End If**  **Next j**  **x(m) = x(i)**  **x(i) = xmax**  **Next i**  **End Sub**  *'Процедура удаления отрицательных эл-тов массива*  **Public Sub mass\_udal(ByRef x() As Single)**  **Dim j, i, n As Integer**  **n = UBound(x)**  **i = 0**  **Do While i <= n**  **If x(i) < 0 Then**  **For j = i To n - 1**  **x(j) = x(j + 1)**  **Next j**  **n = n - 1**  **Else**  **i = i + 1**  **End If**  **Loop**  **ReDim Preserve x(n)**  **End Sub**  *'Процедура обработки события загрузки формы*  **Private Sub Form1\_Load(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles MyBase.Load**  **Label1.Text = "Точное" + Space(3) + "москоское" + \_**  **Space(3) & " время"**  **Label2.Text = CStr(DateTime.Now)**  **End Sub**    *'Процедура обработки события выбора элем. из ComboBox1*  **Private Sub ComboBox1\_SelectedIndexChanged \_**  **(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles ComboBox1.SelectedIndexChanged**  **Dim index, n As Integer**  **index = ComboBox1.SelectedIndex**  *'индекс выбр-го эл-та ComboBox1*  **ListBox1.Items.Clear()**  **ListBox2.Items.Clear()**  **n = CInt(TextBox1.Text) 'кол=во эл-тов массива**  **ReDim x(n)**  **vvod(x, index)**  *'ввод массива в зависимости от индекса выбора*  **PrintL(x, ListBox1) *'вывод исходного массива***  **End Sub**  ***'Процедура обработки события выбора элем. из ComboBox2***  **Private Sub ComboBox2\_SelectedIndexChanged \_**  **(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles \_**  **ComboBox2.SelectedIndexChanged**  **Dim ind As Integer**  **ind = ComboBox2.SelectedIndex**  **ListBox2.Items.Clear()**  **Select Case ind**  **Case 0**  **sort1(x)*'вызов проц. сортировки массива по возр.***  **Case 1**  **sort2(x)*'вызов проц. сортировки массива по убыв.***  **Case 2**  **mass\_udal(x)***'вызов проц.удаления отриц. эл-тов*  **End Select**  **PrintL(x, ListBox2)***'вызов проц.вывода результ. массива*  **End Sub**  **Private Sub Button6\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles Button6.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.7.7. Программный код проекта* **Проект 7-1*:***

*Заполнение и преобразование одномерного массива*

1. **Результаты выполнения проекта:**

Результаты выполнения проекта приведены на рис. 4.7.7-3.



*Рис. 4.7.7-3. Результаты выполнения* **Проект 7-1*:***

*Заполнение и преобразование одномерного массива*

1. **Доказательство правильности работы программы:**

Получен исходный массив из 8 элементов: 2.06, 0.33, 0.80, -2.10, -1.98, 2.75, -4.86, 2.61, 3.14.

Удалены отрицательные элементы 2.06, 0.33, 0.80, 2.75, 2.61, 3.14.

Вывод: анализируя результирующий массив, приходим к выводу о том, что отрицательные элементы отсутствуют, следовательно, для введенных данных, задача решена верно.

### Контрольные вопросы

1. Какие структуры данных известны?
2. Какую информацию должен включать оператор объявления массива?
3. Что такое массив?
4. Верно ли, что все элементы массива должны быть одного типа?
5. В чем преимущество объединения отдельных элементов в массив?
6. Какие правила объявления массива?
7. Что такое индекс массива?
8. Что такое размер массива и размерность массива?
9. Совпадает ли размерность массива с количеством элементов в нем?
10. Может ли массив содержать один элемент?
11. Может ли массив совсем не содержать элементов?
12. Что произойдет, если индекс массива выйдет за допустимые пределы?
13. Каким образом различаются два элемента массива, имеющие одинаковые значения?
14. В каком случае можно сказать, что два массива равны?
15. Каким образом можно поменять местами значения двух массивов?
16. Может ли одномерный массив иметь больше одного индекса?
17. Можно ли в качестве индекса одномерного массива использовать выражение?
18. Почему одномерный массив иногда называют списком или вектором?
19. Приведите пример массива целых чисел, массива вещественных чисел, массива строк.
20. Какими способами инициализируется массив?
21. Что такое статический массив?
22. Что такое динамический массив?
23. Как изменить размер массива, сохранив имеющиеся в нём значения?
24. Как определить размер массива в ходе выполнения программы?
25. Чем отличаются списки **ListBox** и **ComboBox**?
26. Как настраиваются свойства списков **ListBox**?
27. Как настраиваются свойства списков **ComboBox**?
28. Как добавить, изменить и удалить элемент списка?
29. Как определить элемент, выбранный пользователем?
30. Каким образом массив передается в процедуру?
31. Какие базовые алгоритмы обработки одномерных массивов известны?
32. Какие методы сортировки массивов известны?
33. В чем суть метода сортировки элементов массива выбором?
34. В чем суть метода сортировки элементов массива пузырьком?
35. Какие методы класса **Array** известны?
36. Каким образом объявляются одномерные массивы в программах?