Оглавление

Введение

При обработке информации часто возникает необходимость одновременного доступа к различным характеристикам объекта. При этом удобно, когда все характеристики объекта хранятся вместе в специально разработанной структуре. Поэтому в Visual Basic 2005 помимо массивов, представляющих собой нумерованный набор элементов одного типа, существует возможность создавать произвольные структуры данных, состав которых определятся программистом. Про такие типы данных говорят, что они определены пользователем, подразумевая при этом, что программист является пользователем среды и языка программирования. Как правило, типы данных, определенные пользователем (программистом) называют структурами или записями.

Структура – это непустая совокупность нескольких элементов, каждый из которых может иметь свой тип данных. Причем Visual Basic 2005 допускает использование одной структуры внутри дугой. Отдельный элемент структуры называется полем. Помимо полей структура может содержать еще различные подпрограммы. Подпрограммы, входящие в состав структуры называются методами.

1. Описание структуры. Область видимости. Понятие метода

Описание структуры должно располагаться в самом начале модуля перед описанием всех подпрограмм, глобальных и локальных переменных. Как правило, описание структуры располагают сразу после слов `Public Class`. Описывать структуры внутри программы или подпрограммы нельзя.

При описании структуры необходимо перечислить все ее поля, указав для каждого из них тип данных. Затем надо описать все методы, входящие в состав структуры. В общем случае описание структуры выглядит следующим образом.

```
ОбластьВидимости Structure ИмяСтруктуры
    ОбластьВидимости ИмяПоля1 As ТипДанных
    ОбластьВидимости ИмяПоля2 As ТипДанных
    ...
    ОбластьВидимости ИмяПоляN As ТипДанных
    Описание методов
End Structure
```

Оглавление

Имена структуры, полей и методов должны строиться в соответствии с общим правилом имен. В качестве типов данных при описании полей структуры могут использоваться любые типы данных, существующие в Visual Basic 2005, в том числе и ранее объявленные структуры.

Область видимости задается одним из двух ключевых слов: `Private` или `Public`. Она определяет доступность структуры, поля или метода из других частей проекта или программы. Структура, описанная как `Private`, доступна только в том модуле, в котором она описана. Структура, описанная как `Public`, доступна во всех модулях проекта.

Поле, имеющее область видимости `Private`, доступно только внутри структуры. Если возникает необходимость обработать такое поле извне, то для этого в состав структуры включают специальные методы. Поле с областью видимости `Public` может быть обработано вне структуры. При описании полей вместо ключевого слова `Public` можно использовать ключевое слово `Dim`. Для структур разница между `Dim` и `Public` полями отсутствует.

Поле структуры может быть одномерным или многомерным массивом. В этом случае после имени поля ставятся круглые скобки. Если массив должен быть многомерным, то между скобками ставится необходимое количество запятых. Обратите внимание, что в составе структуры могут использоваться только массивы с неизвестным номером последнего элемента. Обработка массивов, являющихся полями структуры, ничем не отличается от обработки аналогичных массивов, описанных вне структуры.

В состав структуры мимо полей могут входить еще и методы. Метод – это подпрограмма, входящая в состав структуры и предназначенная для обработки полей только той структуры, где она описана. Поля других структур метод обрабатывать не может. Метод может быть как функцией, так и процедурой. В зависимости от этого несколько меняется описание метода.

Описание метода-процедуры.

```
ОбластьВидимости Sub ИмяПроцедуры(Параметры)
    Операторы
End Sub
```

Описание метода-функции.

```
ОбластьВидимости Function ИмяФункции(Параметры) _
```

Оглавление

```
                    As ТипРезультата
        Операторы
        Return Результат
    End Function
```

Область видимости метода определяет, будет ли данный метод доступен вне структуры. `Public` метод предназначен для обработки полей структуры и может быть вызван из любой другой подпрограммы, в том числе и не принадлежащей данной структуре. `Private` метод является вспомогательным и не может быть вызван извне структуры.

В качестве примера рассмотрим описание структуры, предназначенной для хранения информации о городах. Для каждого города требуется хранить его название, значения дневной и ночной температуры.

Чтобы эту структуру можно было использовать в других модулях программы, зададим для нее область видимости `Public`. А самой структуре дадим имя `Gorod`.

```
Public Structure Gorod
```

Последовательно опишем все поля структуры. Для каждого данного заведем отдельное поле. Первым будет поле для хранения названия города. Очевидно, что это поле будет иметь строковый тип данных.

```
    Public Nazvanie As String
```

Два следующих поля предназначены соответственно для хранения дневной и ночной температуры. Они имеют целый тип.

```
    Public Den As Integer
    Public Noch As Integer
```

Для удобства работы включим в состав структуры два метода-функции. Первый из них предназначен для упрощения вывода структуры. Он формирует и возвращает строку, в которую входят значения всех полей. Для формирования колонок используется константа `vbTab`. Числовые поля преобразуются в строки с помощью функции `Str`. Строковые поля преобразовывать не надо.

```
    Public Function Print() As String
       Return Nazvanie + vbTab + Str(Den) + _
              vbTab + Str(Noch)
    End Function
```

Второй метод предназначен для вычисления среднесуточной температуры в городе. Для этого складываются значения дневной и ночной температур, и полученная сумма

Оглавление

делится на два. Очевидно, что значение среднесуточной температуры, полученное в результате деления, будет рациональным числом. Следовательно, эта функция должна иметь рациональный тип результата.

```
    Public Function Srednee() As Single
        Return (Den + Noch) / 2
    End Function
End Structure
```

2. Оператор With

Для того чтобы обратиться к Public полю структуры или вызвать Public метод надо указать имя переменной, имеющей тип структуры, и через точку напечатать имя нужного поля или метода. Например.

```
Dim g as Gorod
g.Nazvanie = "Москва"
g.Den = 34
g.Noch = 21
```

Когда нужно обработать сразу несколько полей структуры, можно использовать оператор `With`. Он позволяет упростить текст программы, сделав его более наглядным. Рассмотрим общей вид этого оператора.

```
With Имя переменной, имеющей структурный тип
    Операторы
End With
```

Внутри оператора `With` имя обрабатываемой переменной не указывается, а обращение к нужному полю или методу начинается с точки.

```
Dim g as Gorod
With g
    .Nazvanie = "Москва"
    .Den = 34
    .Noch = 21
End With
```

3. Ввод массива структур

Ввод массива структур практически не отличается от ввода одномерного массива. Он также состоит из двух этапов. На первом этапе указывается количество элементов в массиве и соответствующим образом переопределяется размер массива. На втором этапе организуется цикл, на каждом шаге которого вводится значение одного

Оглавление

элемента. Рассмотрим особенности программной реализации этого алгоритма применительно к обработке массива структур.

Сначала описывается массив структур `a()`. Так как его размер заранее неизвестен, то массив описывается без указания верхней границы.

```
Dim a()As Gorod
```

Для работы с массивом нам необходимо знать его размер. Он будет храниться в переменной `n`. Поскольку массивы всегда обрабатываются в цикле, то для организации цикла `For` нам потребуется счетчик `i`. Очевидно, что обе эти переменных всегда будут иметь целый тип.

```
Dim n, i As Integer
```

Задание массива начинается с определения его размера. Мы просим пользователя указать количество городов. Так как количество городов может быть только положительным, то при вводе этого значения необходима проверка, которую мы организуем с помощью цикла `Do Loop Until`.

```
Do
    n = Val(InputBox("Введите количество городов"))
Loop Until n > 0
```

В Visual Basic 2005 нумерация элементов массива всегда начинается с нуля. Следовательно, номер последнего элемента будет на единицу меньше общего количества элементов массива. Поэтому уменьшаем значение переменной `n` на единицу. Теперь в ней хранится не количество элементов, а номер последнего элемента массива.

```
n -= 1
```

Задаем размер массива `a()`, указывая в операторе `ReDim` номер последнего элемента массива.

```
ReDim a(n)
```

Организуем цикл для ввода значений элементов массива. Элементы массива последовательно пронумерованы от 0 до `n`. Следовательно, счетчик цикла должен изменяться в этом же диапазоне. Тогда на `i`-м шаге цикла мы будем вводить элемент массива с номером `i`.

```
For i = 0 To n
```

Теперь надо ввести значения всех полей для очередного элемента массива. Для этого будем использовать функцию `InputBox`. По правилам языка Visual Basic 2005

Оглавление

значение каждого поля структуры должно вводиться по отдельности. Сразу ввести значения всех полей нельзя. При вводе значения строкового поля никаких дополнительных преобразований не требуется.

```
a(i).Nazvanie = InputBox("Название города")
```

При вводе значений числовых полей необходимо использовать преобразование `Val`.

```
a(i).Den = Val(InputBox("Дневная температура"))
a(i).Noch = Val(InputBox("Ночная температура"))
Next
```

После завершения цикла массив структур полностью введен и может обрабатывать в соответствии с условием задачи.

4. Вывод массива структур

Для вывода массива структуры, как правило, используется окно списка, хотя можно организовать вывод и в другие элементы управления, например в текстовое поле. Рассмотрим фрагмент программы, реализующий вывод массива структур в окно списка.

Сначала очищаем окно списка от результатов предыдущих запусков программы.

```
lstGorod.Items.Clear()
```

Выводим заголовки колонок. Колонки организуются с помощью константы `vbTab`.

```
lstGorod.Items.Add("Город" + vbTab + "День" + _
                   vbTab + "Ночь")
```

Выводим горизонтальную черту, чтобы зрительно отделить заголовки колонок от значений соответствующих полей.

```
lstGorod.Items.Add("-----------------------------")
```

Организуем цикл для обработки всех элементов массива структур.

```
For i = 0 To n
```

Для формирования одной строки из значений полей структуры мы разработали специальный метод (см. раздел 1). В окно списка будем выводить строку, которую нам вернет метод `Print`.

```
    lstGorod.Items.Add(a(i).Print)
Next
```

Оглавление

5. Поиск в массиве структур

Алгоритмы обработки массивов структур практически не отличаются от аналогичных алгоритмов обработки одномерных массивов. Рассмотрим задачу поиска минимального и максимального элементов в массиве структур. В качестве примера будем использовать массив городов, описанный в предыдущих разделах. В этом массиве будем искать город с максимальной дневной температурой и город с минимальной ночной температурой.

Для решения задач нам потребуются переменные для хранения максимального и минимального элементов массива. Очевидно, что эти переменные будут иметь такой же тип данных, что и обрабатываемый массив.

```
Dim max, min As Gorod
```

Поиск, как обычно, начинаем с нулевого элемента массива.

```
max = a(0)
min = a(0)
```

Организуем цикл для обработки всех элементов массива. Так как нулевой элемент мы же использовали при задании начальных значений переменных, то обработку массива можно начать с первого элемента.

```
For i = 1 To n
```

На каждом шаге массива сравниваем дневную температуру в текущем городе с дневной температурой в ранее найденном городе. Обратите внимание на то, что структуры сравниваются по значению одного конкретного поля. Сравнивать структуры полностью нельзя. Сравнение всегда должно происходить по одинаковым полям.

```
    If a(i).Den > max.Den Then
```

Если в текущем городе температура выше, то мы должны обновить максимум, записав в него анализируемый элемент массива. Заметьте, что элемент массива запоминается целиком, хотя сравнение было только по одному полю. Это делается для того, чтобы в переменной максимум сохранилось вся информация о данном городе, а не только значение дневной температуры.

```
        max = a(i)
    End If
```

Теперь мы сравниваем ночную температуру в текущем городе с ночной температурой в ранее найденном городе.

```
    If a(i).Noch < min.Noch Then
```

Оглавление

Если в текущем городе температура ниже, то мы должны обновить минимум, записав в него анализируемый элемент массива.

```
            min = a(i)
        End If
    Next
```

После завершения цикла нам остается только вывести полученные значения. Сначала выводим горизонтальную черту, чтобы зрительно отделить исходные данные от результатов.

```
lstGorod.Items.Add("------------------------------")
```

Выводим поясняющий текст.

```
lstGorod.Items.Add("Город с макс. дневной темп.")
```

С помощью специального метода (см. раздел 1) формируем строку из данных, записанных в переменной `max`. Эту строку выводим в окно списка.

```
lstGorod.Items.Add(max.Print)
```

Выводим поясняющий текст.

```
lstGorod.Items.Add("Город с мин. ночной темп.")
```

С помощью метода `Print` формируем строку из данных, записанных в переменной `min`. Эту строку выводим в окно списка.

```
lstGorod.Items.Add(min.Print)
```

Полный текст программы представлен в приложении 1. Пример работы программы приведен на рис. 1.

Оглавление

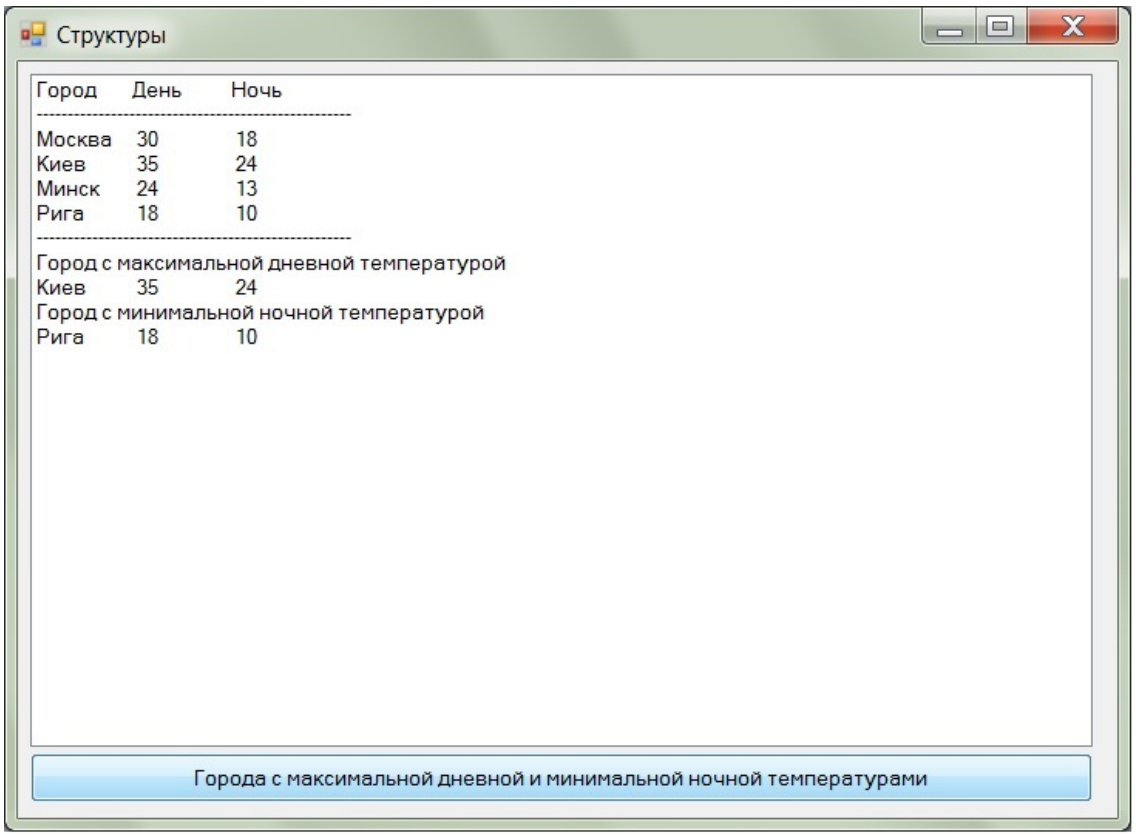

Рис. 1. Пример работы программы обработки массива структур

6. Формирование нового массива из некоторых элементов исходного массива

В качестве примера возьмем задачу формирования массива городов, в которых дневная температура выше средней. Эта задача решается в два этапа. На первом этапе вычисляется средняя дневная температура. Для этого находится сумма всех дневных температур и делится на количество городов. На втором этапе происходит непосредственное формирование нового массива. Анализируется каждый элемент исходного массива. Если дневная температура в этом городе больше средней, то он переносится в новый массив. Рассмотрим особенности программной реализации этого алгоритма.

Начнем с описания необходимых переменных. Для решения задачи нам потребуется еще один массив. В него мы перенесем те элементы исходного массива, которые будут удовлетворять оставленному условию. Другими словами, это результирующий массив. Очевидно, что тип элементов в этом массиве должен совпадать с типом элементов в исходном массиве.

```
Dim b() As Gorod
```

Оглавление

В отдельной переменной будем хранить номер последнего элемента в формируемом массиве.

```
Dim k As Integer
```

Для вычисления средней дневной температуры нам необходимо вычислить сумму всех дневных температур. Значение этой суммы будем записывать в специальную переменную.

```
Dim Summa As Integer
```

Для средней дневной температуры заведем отдельную переменную. Так как средняя температура получается в результате деления, то эта переменная должна иметь рациональный тип данных.

```
Dim Sred As Single
```

Первый этап решения задачи – это вычисление средней дневной температуры. Для этого необходимо вычислить сумму всех дневных температур. Задаем начальное значение суммы.

```
Summa = 0
```

Организуем цикл для обработки всех элементов исходного массива.

```
For i = 0 To n
```

На каждом шаге цикла добавляем очередное значение дневной температуры к ранее накопленной сумме.

```
    Summa += a(i).Den
Next
```

После завершения цикла можно вычислить среднее значение дневной температуры. Для этого необходимо найденную сумму разделить на количество городов. Заметим, что `n` – это номер последнего города. Так как нумерация городов в массиве идет с нуля, то количество городов будет на единицу больше, чем номер последнего элемента массива.

```
sred = Summa / (n + 1)
```

Выводим горизонтальную черту, чтобы зрительно отделить промежуточные результаты от исходных данных.

```
lstGorod.Items.Add("------------------------------")
```

Выводим значение средней дневной температуры, чтобы пользователь мог проконтролировать правильность работы программы.

```
lstGorod.Items.Add("Средняя дневная температура:" + _
                   Str(Sred))
```

Оглавление

Переходим ко второму этапу – формированию нового массива. До начала формирования массив пуст. В нем не содержится ни одного элемента. Поэтому номер последнего элемента в этом массиве находится за его пределами и равен -1.

```
k = -1
```

Организуем цикл для обработки всех элементов исходного массива.

```
For i = 0 To n
```

На каждом шаге цикла анализируем очередной город.

```
If a(i).Den > Sred Then
```

Если дневная температура в этом городе больше средней, то мы должны перенести всю информацию об этом городе в новый массив. При этом номер последнего элемента в новом массиве увеличится на единицу.

```
k += 1
```

Изменяем размер результирующего массива. Использование ключевого слова `Preserve` позволит нам сохранить в массиве все ранее найденные элементы.

```
ReDim Preserve b(k)
```

В новый массив на последнюю позицию записываем анализируемый элемент. Обратите внимание на то, что элемент записывается целиком, хотя сравнение проводится только по одному полю. Это делается для того, чтобы в новом массиве сохранялась вся информация о городе, а не только значение дневной температуры.

```
b(k) = a(i)
End If
Next
```

После завершения цикла выводим полученный массив. Сначала печатаем горизонтальную черту, чтобы зрительно отделить полученные результаты от исходных данных.

```
lstGorod.Items.Add("------------------------------")
```

Затем печатаем поясняющий заголовок и еще одну горизонтальную черту.

```
lstGorod.Items.Add("Результирующий массив")
lstGorod.Items.Add("------------------------------")
```

Анализируем номер последнего элемента в сформированном массиве.

```
If k < 0 Then
```

Если номер последнего элемента меньше нуля, значит, в новом массиве нет ни одного элемента. Поэтому вместо значений элементов массива выводим поясняющее сообщение.

Оглавление

```
            lstGorod.Items.Add("Таких городов нет")
        Else
```

В противном случае, организуем цикл для вывода полученного массива.

```
            For i = 0 To k
```

На каждом шаге цикла с помощью метода `Print` (см. раздел 1) формируем строку из данных, записанных в очередном элементе массива. Эту строку выводим в окно списка.

```
                lstGorod.Items.Add(b(i).Print)
            Next
        End If
```

Полный текст программы представлен в приложении 2. Пример работы программы приведен на рис. 2.

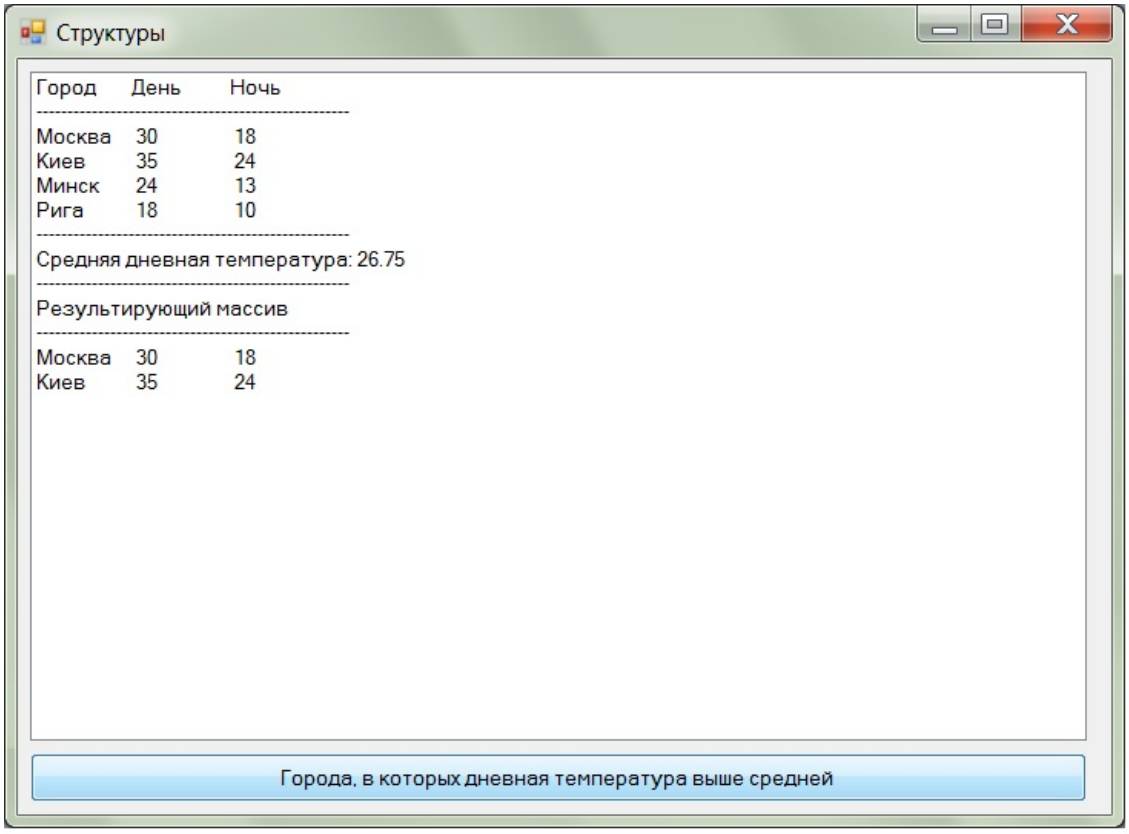

Рис. 2. Пример работы программы формирования нового массива структур из элементов исходного массива

7. Сортировка массива структур

Рассмотрим задачу сортировки массива структур. Она решается практически также как задача сортировки одномерного массива. Как правило, при сортировке

Оглавление

массива структуры используется метод пузырька, хотя все другие методы сортировки тоже можно использовать для решения этой задачи. В качестве примера возьмем задачу сортировки городов по убыванию среднесуточной температуры.

Так как любая сортировка предполагает перестановку элементов, то для нее нам потребуется дополнительная переменная. Очевидно, что эта переменная будет иметь тот же тип, что и элементы массива.

```
Dim z As Gorod
```

Сортировка методом пузырька предполагает использование логической переменной для хранения информации о состоянии массива: отсортирован он или нет. Опишем соответствующую переменную.

```
Dim sort As Boolean
```

Организуем внешний цикл сортировки. Он будет повторяться до тех пор, пока массив не станет упорядоченным.

```
Do
```

Перед началом внутреннего цикла предполагаем, что массив отсортирован.

```
    sort = True
```

Организуем цикл для обработки всех элементов массива от нулевого до предпоследнего. На каждом шаге цикла мы будем сравнивать очередной элемент массива со следующим. Поэтому предпоследний элемент будет сравнивать с последним, а последним элемент сравнивать не с чем. Следовательно, внутренний цикл надо остановить на предпоследнем элементе массива.

```
    For i = 0 To n – 1
```

Сравниваем значение среднесуточной температуры в текущем элементе и в следующем за ним. Для вычисления среднесуточной температуры мы используем специальный метод `Srednee`, который описан в разделе 1.

```
        If a(i).Srednee < a(i + 1).Srednee Then
```

Если среднесуточная температура в `i`-м городе меньше, чем среднесуточная температура в `(i+1)`-м городе, значит, эти города в массиве идут в неправильном порядке, и, следовательно, массив неупорядочен. Меняем значение переменной `sort` на противоположное.

```
            sort = False
```

И переставляем местами элементы массива так, чтобы они шли в правильном порядке. Для этого мы используем дополнительную переменную. Обратите внимание на то, что структуры переставляются целиком, хотя сравнение шло только по одному признаку.

Оглавление

```
                z = a(i)
                a(i) = a(i + 1)
                a(i + 1) = z
            End If
        Next
```

Перестановка элементов будет продолжаться до тех пор, пока массив не станет отсортированным.

```
    Loop Until sort
```

После завершения внешнего цикла сортировки выводим измененный массив. Сначала выводим горизонтальную черту, чтобы зрительно отделить исходные данные от полученных результатов.

```
    lstGorod.Items.Add("------------------------------")
```

Затем печатаем поясняющий заголовок и еще одну горизонтальную черту.

```
    lstGorod.Items.Add("Массив после сортировки")
    lstGorod.Items.Add("------------------------------")
```

Организуем цикл для вывода всех элементов массива структур.

```
    For i = 0 To n
```

На каждом шаге цикла с помощью метода `Print` (см. раздел 1) формируем строку из данных, записанных в очередном элементе массива. Эту строку выводим в окно списка.

```
        lstGorod.Items.Add(a(i).Print)
    Next
```

Полный текст программы представлен в приложении 3. Пример работы программы приведен на рис. 3.

Оглавление

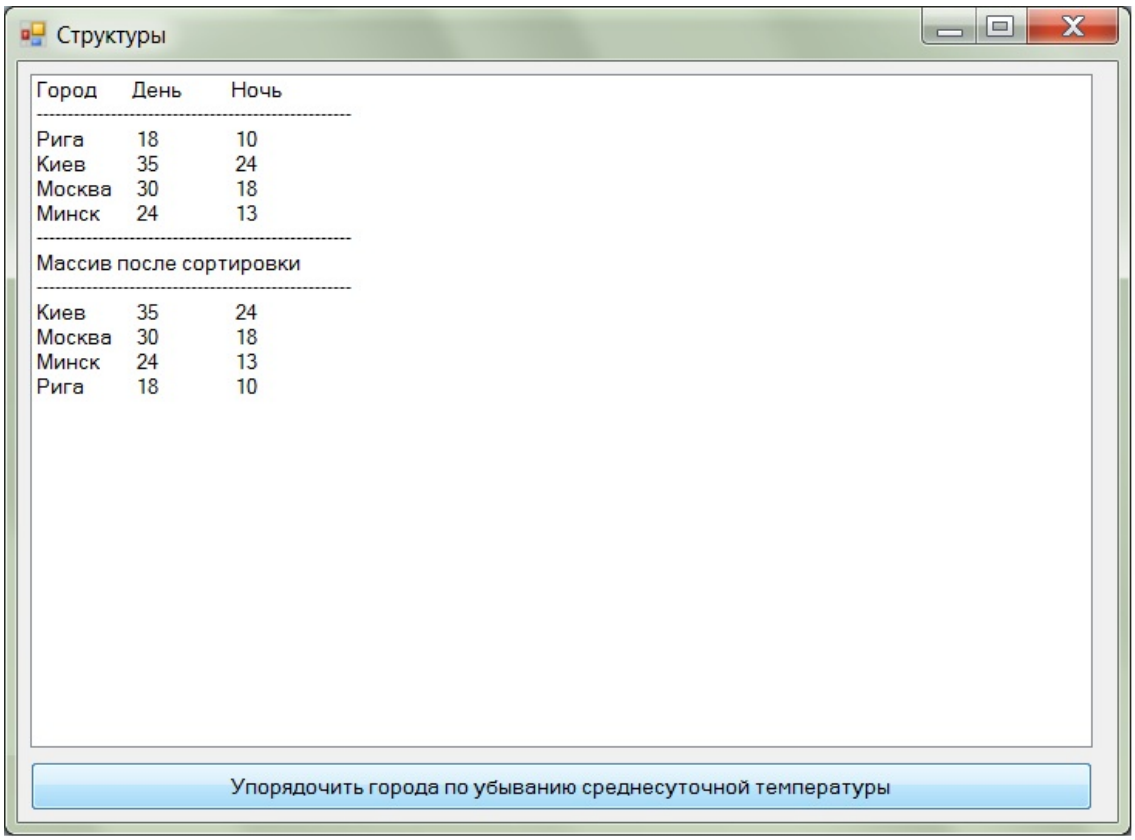

Рис. 3. Пример работы программы сортировки нового массива структур

Приложение 1

Дан одномерный массив городов. Известно название каждого города, дневная и ночная температура. Найти город с максимальной дневной температурой и город с минимальной ночной температурой. Исходные данные и полученные результаты вывести в окно списка.

Описание структуры

```
Public Structure Gorod
    Public Nazvanie As String
    Public Den As Integer
    Public Noch As Integer
    Public Function Print() As String
        Return Nazvanie + vbTab + Str(Den) + vbTab + _
               Str(Noch)
    End Function
    Public Function Srednee() As Single
        Return (Den + Noch) / 2
```

Оглавление

```
    End Function
End Structure
```

Текст программы.

```
Dim a() As Gorod
Dim n, i As Integer
Dim max, min As Gorod
lstGorod.Items.Clear()
Do
    n = Val(InputBox("Введите количество городов"))
Loop Until n > 0
n -= 1
ReDim a(n)
For i = 0 To n
    a(i).Nazvanie = InputBox("Название города")
    a(i).Den = Val(InputBox("Дневная температура"))
    a(i).Noch = Val(InputBox("Ночная температура"))
Next
lstGorod.Items.Add("Город" + vbTab + "День" + vbTab + _
                   "Ночь")
lstGorod.Items.Add("-----------------------------------")
For i = 0 To n
    lstGorod.Items.Add(a(i).Print)
Next
max = a(0)
min = a(0)
For i = 1 To n
    If a(i).Den > max.Den Then
        max = a(i)
    End If
    If a(i).Noch < min.Noch Then
        min = a(i)
    End If
Next
lstGorod.Items.Add("-----------------------------------")
```

Оглавление

```
lstGorod.Items.Add("Город с макс. дневной температурой")
lstGorod.Items.Add(max.Print)
lstGorod.Items.Add("Город с мин. ночной температурой")
lstGorod.Items.Add(min.Print)
```

Приложение 2

Дан одномерный массив городов. Известно название каждого города, дневная и ночная температура. В новый массив перенести города, в которых дневная температура выше средней. Исходные данные и полученные результаты вывести в окно списка.

Описание структуры

```
Public Structure Gorod
    Public Nazvanie As String
    Public Den As Integer
    Public Noch As Integer
    Public Function Print() As String
        Return Nazvanie + vbTab + Str(Den) + vbTab + _
               Str(Noch)
    End Function
    Public Function Srednee() As Single
       Return (Den + Noch) / 2
    End Function
End Structure
```

Текст программы.

```
Dim a() As Gorod
Dim n, i As Integer
Dim b() As Gorod
Dim k As Integer
Dim Summa As Integer
Dim Sred As Single
lstGorod.Items.Clear()
Do
    n = Val(InputBox("Введите количество городов"))
Loop Until n > 0
n -= 1
ReDim a(n)
```

Оглавление

```
For i = 0 To n
    a(i).Nazvanie = InputBox("Название города")
    a(i).Den = Val(InputBox("Дневная температура"))
    a(i).Noch = Val(InputBox("Ночная температура"))
Next
lstGorod.Items.Add("Город" + vbTab + "День" + vbTab + _
                   "Ночь")
lstGorod.Items.Add("-----------------------------------")
For i = 0 To n
    lstGorod.Items.Add(a(i).Print)
Next
Summa = 0
For i = 0 To n
    Summa += a(i).Den
Next
sred = Summa / (n + 1)
lstGorod.Items.Add("-----------------------------------")
lstGorod.Items.Add("Средняя дневная температура:" + _
                   Str(Sred))
k = -1
For i = 0 To n
    If a(i).Den > Sred Then
        k += 1
        ReDim Preserve b(k)
        b(k) = a(i)
    End If
Next
lstGorod.Items.Add("-----------------------------------")
lstGorod.Items.Add("Результирующий массив")
lstGorod.Items.Add("-----------------------------------")
If k < 0 Then
    lstGorod.Items.Add("Таких городов нет")
Else
    For i = 0 To k
```

Оглавление

```
        lstGorod.Items.Add(b(i).Print)
    Next
End If
```

Приложение 3

Дан одномерный массив городов. Известно название каждого города, дневная и ночная температура. Упорядочить города по убыванию среднесуточной температуры. Исходные данные и полученные результаты вывести в окно списка.

Описание структуры

```
Public Structure Gorod
    Public Nazvanie As String
    Public Den As Integer
    Public Noch As Integer
    Public Function Print() As String
        Return Nazvanie + vbTab + Str(Den) + vbTab + _
               Str(Noch)
    End Function
    Public Function Srednee() As Single
       Return (Den + Noch) / 2
    End Function
End Structure
```

Текст программы.

```
Dim a() As Gorod
Dim n, i As Integer
Dim z As Gorod
Dim sort As Boolean
lstGorod.Items.Clear()
Do
    n = Val(InputBox("Введите количество городов"))
Loop Until n > 0
n -= 1
ReDim a(n)
For i = 0 To n
    a(i).Nazvanie = InputBox("Название города")
    a(i).Den = Val(InputBox("Дневная температура"))
```

Оглавление

```
    a(i).Noch = Val(InputBox("Ночная температура"))
Next
lstGorod.Items.Add("Город" + vbTab + "День" + vbTab + _
                   "Ночь")
lstGorod.Items.Add("-------------------------------------")
For i = 0 To n
    lstGorod.Items.Add(a(i).Print)
Next
Do
    sort = True
    For i = 0 To n - 1
        If a(i).Srednee < a(i + 1).Srednee Then
            sort = False
            z = a(i)
            a(i) = a(i + 1)
            a(i + 1) = z
        End If
    Next
Loop Until sort
lstGorod.Items.Add("-------------------------------------")
lstGorod.Items.Add("Массив после сортировки")
lstGorod.Items.Add("-------------------------------------")
For i = 0 To n
    lstGorod.Items.Add(a(i).Print)
Next
```

Список литературы

1. Волчёнков Н.Г. Программирование на Visual Basic 6: В 3-х ч. Часть 1. – М.: ИНФРА-М, 2000. – 286 с.
2. Шевякова Д.А., Степанов А.М., Карпов Р.Г. Самоучитель Visual Basic 2005 / под общ. ред. А.Ф. Тихонова. – СПб.: БХВ-Петербург, 2007. – 576 с.
3. Богданов М.Р. Visual Basic 2005 на примерах. – СПб.: БХВ-Петербург, 2007. – 592 с.

Оглавление