Records

Record structures allow a number of data items to be grouped together and named. These structures are useful as array elements to store a number of data items in memory that can be of different types. As with arrays, the data items may be input from a file or they may be created by a program.

Record Structures

A record structure is first defined as a type in a Visual Basic class.

```
Private Structure <recordname>
        <Variable declaration 1>
        . . .
        <Variable declaration n>
End Structure
```

The variables declared in the structure are the names of fields in the record structure.

Example:

```
Private Structure petrecord
        Dim petname, owner, species as String
        Dim weight As Single
End Structure
```

Setting and Using Fields of a Record Structure

A regular variable can be declared to have the type of a record structure.

```
Dim pet As petrecord
```

The fields are accessed by using a . (dot) notation and the name of the field.

```
pet.petname = “Fluffy”
pet.owner = “Jay Jones”
pet.species = “Domestic Oriental”
pet.weight = 7.2
```

These values can be used in further computations, such as

```
If pet.weight > 10.0 Then
        Label1.Text = “Large”
End If
```

Declaration of arrays whose elements are fields

A common use of record structures is as elements of arrays:

```
Dim <arrayname> ( 0 to <maxnumber> ) as <recordtype>
```

Example:

```
Dim pets ( 0 to 24 ) as petrecord
```

Now each element of the array can be accessed via array subscripting. Since this element is a record structure, then the dot notation is used to access individual elements.

Example assigning elements of the pet record in array location 0:

```
Dim n as Integer = 0
pets(n).petname = "Fluffy"
pets(n).owner = "Jay Jones"
pets(n).species = "Domestic Oriental"
pets(n).weight = 7.2
```

It is also quite common to get the values of an array of records from a file, using the input command. Suppose that you have a file with the elements of one pet record on each line:

```
Dim n as Integer = 0
While Not EOF(1)
        Input ( 1, pets(n).petname)
        Input ( 1, pets(n).owner)
        Input ( 1, pets(n).species)
        Input ( 1, pets(n).weight)
        n = n + 1
End While
```

This will read each pet record from the file and store it into the pets array.

"With" notation for record structures

As an abbreviation for giving the name of the record structure variable with the dot notation for each field, if you are doing a lot of dot notations on a single element, the "with" notation allows an abbreviation. This abbreviation allows you to omit the name of the record. The goal of the abbreviation is just to save typing:

```
Dim n as Integer = 0
        While Not EOF(1)
                With pets(n)
                        Input ( 1,  .petname)
                        Input ( 1,  .owner)
                        Input ( 1,  .species)
                        Input ( 1,  .weight)
                End With
                n = n + 1
        End While
```

Hierarchical Record Structures

Record structures can be further organized by having an element of one record structure be another record structure. This is a useful tool for organizing information.

```
Private Structure addressrecord
        Dim street, city, state, zip as String
End Structure

Private Structure petrecord
        Dim petname, owner, species as String
        Dim weight As Single
        Dim address As addressrecord
End Structure
```

Setting and using elements of Hierarchical Records:

For each element of an inner record, an additional dot notation is used to access the element. For example, the zip code of a single pet:

```
pet.address.zip
```

This additional notation can also be used in arrays of record structures.