Advanced Data Types

Introduction to Records

Review – Data Types

Useful information is stored in a variable. Each variable needs to be declared, where it is assigned an identifier (i.e., the variable name) and a data type.

There are a number of data types which are used in most programming languages:

- integer - real -
- string - character -
- boolean -

Example – Student Data

To store information about a single student, you might use the following variables:

```
var firstname : string
var lastname : string
var age : int
var address : string
var average : real
var graduating : boolean
```

Example – Student Records

For 100 students, arrays could be used, but it might still be a bit awkward:

```
var firstname : array 1..100 of string
var lastname : array 1..100 of string
var age : array 1..100 of int
var address : array 1..100 of string
var average : array 1..100 of real
var graduating : array 1..100 of boolean
```

Data Records

A record is a more advanced data type, which actually combines multiple data types into a single structure.

Rather than having multiple variables for each piece of information, a record is created as a new data type. Each piece of information is a field in the new record.

Example – Student Record

At first glance, it actually seems a bit more complicated:

```
type studentData:
   record
      firstname : string
      lastname : string
      age : int
      address : string
      average : real
      graduating : boolean
   end record
```

Example – Student Record

To access a student record, use a dot to choose any field in the record.

```
var student : studentData

put "What is the student's first name?"
get student.firstname

put "What is the student's age?"
get student.age
```

Why Use Records

Remember that programmers want to make their code efficient and readable. Someone else might have to modify or maintain your code later in its life-cycle.

Part of good programming is organization, and records allow us to group related data together, even if it is made up of different data types.

Example – Multiple Student Records

Consider again the multiple arrays we declared in before using records. With our new data type called `studentData`, it is much simpler to have records for multiple students.

```
var students : array 1..100 of studentData
```

The next question how to access each record, and the fields within each record.

Arrays of Records

To access elements in an array, we use the index of the array (i). This works with arrays of records, but you need to remember that the index goes with the record, and not the field.

For example:

```
put student(3).firstname      % correct!
```

not

```
put student.firstname(3)      % incorrect!
```