Using Arrays with Subprograms

Review - What is an Array?

Recall: An array is a collection of one type of data (e.g., integer, string) that is used for a single purpose (e.g., grades, addresses).

Each box is called an element of the array, and the position of each element is the index.

an array with 5 elements

Subprograms using Arrays

For the purposes of a function or procedure, an array is just like any other variable. You can pass it to the subprogram as a parameter (just like an integer, string, or real)

A function can return an array, and a procedure can modify an array if required.

Since the declaration of an array is more complicated than a variable, we must show the same care when using arrays with functions and procedures.

Arrays in Turing

To declare an array in Turing:

var name : array low .. high of dataType

name – the name of the array

low – the lower index value (usually 1 or 0)
high – the upper index value

dataType – integer, string, real, etc...

Example – Passing a Specific Array to a Function

```
procedure printArray (arr : array 1 .. 3 of string)
    for i : lower (arr) .. upper (arr)
        put arr (i)
    end for
end printArray

var firstname : array 1 .. 3 of string
% initialize the array
firstname(1) := "fred"
firstname(2) := "wilma"
firstname(3) := "pebbles"

% output the array
printArray(firstname)
```

Array Functions

Like strings, there are specialized functions in Turing to help handle arrays.

In the previous example, we used the upper() and lower() functions:

upper(arrayName) – returns the highest index value in the array

lower(arrayName) – returns the lowest index value in he array

Example – Passing a General Array to a Function

```
procedure print_array (arr : array 1 .. * of string)
    for i : lower (arr) .. upper (arr)
        put arr (i)
    end for
end print_array

var firstname : array 1 .. 3 of string
% initialize the array
firstname(1) := "fred"
firstname(2) := "wilma"
firstname(3) := "pebbles"

% output the array
printArray(firstname)
```

Initializing Arrays

It is good practise to initialize (i.e., set a starting value) for all variables, including the elements of an array.

Sometimes, we want to set all elements to the same value. A for loop is best for this:

```
for i : 1 .. upper(grades)
   grades(i) := 0
end for
```

Initializing Arrays

If each value will be different, the array elements can be filled one at a time:

```
var firstname : array 1 .. 3 of string
% initialize the array
firstname(1) := "fred"
firstname(2) := "wilma"
firstname(3) := "pebbles"
```

Initializing Arrays

There is a short form for initializing arrays, but the problem is that it results in very long lines of code (less readable):

```
var firstname : array 1 .. 3 of string
   : init (“fred”, “wilma”, “pebbles”)
```