Getting Started With Turing – Assigning & Outputting Values of Variables

Assigning Values to Variables

Declaring a variables reserves space in the computer's memory for information of a particular data type (e.g., int, or real, or char). This variable does not become useful until we assign a value to it. There are several ways to assign a value to a variable. All of them make some use of the **assignment operator**.

1. When declaring the variable

```
var total : int := 15
```

2. Immediately after declaring the variable

```
var total : int
total := 15
```

3. After declaring the variable, but later in the program

```
var total : int
.
.
.
total := 15
```

4. Using the value from another variable

```
var a : int := 5
var b : int

b := a
```

5. Note: An integer can be assigned to a real value, but not the other way around

```
var a : int := 5
var b : real

b := a
```

Output of Variables

The put command can be used to output variables or combinations of variables. Keep in mind that the variable must have been given a value before you try to output it, or you will get an error.

Program:

```
var len : int := 5
var width : int := 10
put "The length is ", len ..
put " and the width is ", width
```

Output:

```
The length is 5 and the width is 10
```

There are a number of details that should be noted from this example.

1. A single put statement can be used to output multiple items. In this case, each put is used to print a string constant (the text inside the quotation marks) and a variable value. You can separate multiple printable items using commas.

2. Notice that the variable for length is actually named `"len"`. This is because `"length"` is a reserved keyword in Turing, and cannot be used as a variable identifier.

One of the reasons we use variables is so we can change their values as the program runs. There are several ways for this to happen, but the simplest is to use the **assignment operator** to change the value. Keep in mind that this will destroy the previous value.

Program:

```
% declare variables
var sample : int

% begin program
sample := -5
put "Sample is now: ", sample
sample := 2074
put "Now it is: ", sample
```

Output:

```
Sample is now: -5
Now it is: 2074
```