# Simple Functions

There are many similarities between functions and procedures. We use a function when we are trying to produce a single answer. We have seen a number of predefined Turing functions already. The sqrt function takes a number as a parameter and computes the result of square rooting the number. We say the function returns the square root as a result. Turing insists that we do something constructive with this result. This means that we call functions in a different way from procedures.

If sqrt was a procedure, we would call it by simply saying:

sqrt(x)

Since it is a function, however, we would get an error if we tried to call it like that. We need to use the result somehow. Assuming that x is an integer or real variable, here are a number of different ways we could call sqrt:

* **var** y : **real** := sqrt(x)
* **if** sqrt(x) > 10 **then**
* **put** sqrt(x)
* drawline(round(sqrt(x)), 10, 20, 30, blue)

Now let's take a look at how we can define our own functions. We'll call them the same way as the predefined ones. Let's create a function that finds the largest of three integers. We will also call the function a couple of times to see how it works.

|  |  |
| --- | --- |
| |  | | --- | | **function** largest (a, b, c : **int**) : **int**  **var** big : **int**    **if** a > b **and** a > c **then**  big := a  **elsif** b > c **then**  big := b  **else**  big := c  **end** **if**    **result** big  **end** largest  *%main program*  **put** largest(5, 2, 8)  **var** x : **int**  x := largest(11, -9, 3)  **put** x | |

This program would produce the following output:

|  |
| --- |
| 8  11 |

Let us consider the parts of the function. The function heading is similar to a procedure heading with two differences. One is we use the word function instead of procedure. The other is that we need to indicate what kind of a value we are trying to find. We do that by putting a colon and the type after the list of parameters (if there are any). In the body of the function, we need at least one result statement. When a function hits a result statement it will return the value after the word result, it will stop executing and return to where the function was called.

In the function largest, we could have avoided the local variable big, by using three result statements. It could be rewritten like this:

|  |  |
| --- | --- |
| |  | | --- | | **function** largest (a, b, c : **int**) : **int**  **if** a > b **and** a > c **then**  **result** a  **elsif** b > c **then**  **result** b  **else**  **result** c  **end** **if**  **end** largest | |

Although this version is shorter, you need to be careful with this approach. Some would argue that a function should have only one result statement. You definitely want to keep the number of result statements to a minimum; otherwise, your function will be too hard to follow.

Here is another example. It is a function that can be used to simulate rolling a single six-sided die.

|  |  |
| --- | --- |
| |  | | --- | | **function** roll : **int**  **var** die : **int**    randint(die, 1, 6)  **result** die  **end** roll  *% main program*  **put** "you rolled a " , roll | |