

Procedures to Simplify Programs

A function is very specific (some inputs and one output). A procedure is much more general. In fact, a procedure can be written to duplicate the effect of almost any block of code.

One of the best reasons to use procedures is to simplify our main program by isolating complicated code as procedures (and sometimes functions, where appropriate).

Procedures to Simplify Programs

```
% Main Program %  
initDisplay()  
validateUser()  
mathQuiz()  
shutdown()
```

By using procedures (with good names) in our main program, it is much simpler to understand.

The procedures hide some of the complicated details, which is called encapsulation.

Passing Parameters

The phrase “passing parameters” is used to describe the input and output that occurs with procedures.

The parameters are the input values we pass to the procedure when we call it.

For example:

```
doSomething(input1, input2, input3)
```

The parameters are input1, input2, and input3.

Changing Parameters

A procedure can change the values of parameters. In many languages (including Turing), this ability is set when declaring the procedure.

Use the var command when declaring the procedure to decide which parameters can be changed.

Changing Parameters

For example:

```
procedure doThis(input1:int, input2:real)
```

- none of these parameters can be changed

```
procedure doThat(var input3:string, input4:int)
```

- input3 can be changed, but input4 cannot

Changing Parameters - Example

```
% the parameter x can be changed
procedure double ( var x : real )
    x := 2 * x
end double

%%%%% Main Program %%%%%%
var y : real
y := 3.14
double ( y ) % This doubles the value of y
```