

## 8 Functions

Two new concepts

**scope** of variable names

**call by reference** versus **call by value** for **input parameters** (a.k.a. **function arguments**)

## Scope of variable names

The **scope** of a variable name is the region of the program in which that variable is defined. The scope in essence is the outer most open-close braces in which that variable is defined.

```
void main(void)
{ // opening of scope for x and y in the next line
  double x,y; // variables x and y "local" to the main program

  x = 4.6;

  y = xsquared(x);

} // end main, and closing of scope for the x and y of the main

double xsquared(double x)
{ // opening of scope for y in the next line
  double y; // variable y local to the xsquared function

  y = x*x; // the variable x is the value passed in

  return(x);
} // end xsquared, and closing of scope for the y local to xsquared
```

On the one hand, this makes it easier to encapsulate pieces of programs meaningfully and to make the program read the way the thought process thinks.

On the other hand, this is a “feature” of C that can be abused.

We refer to the **calling program** and the **called function**.

Note: C uses the term *function* to refer to everything, including the “main program.” For historical reasons, borrowing from other languages (mostly Fortran), the terms “routine” and “subroutine” are also used. Sometimes the terms “module” and “submodule” will be used.

## Rules for declaration of functions

- The function name must be declared prior to its use.
- The usual way to declare the function name prior to its use is to declare a **prototype** prior to the main program.
- Functions that have a data type must be declared as such both in the prototype and in the function definition, and then they must **return** a value of the appropriate data type.
- Function arguments must match up (in their order) by data type in the prototype, in the calling of the function, and in the definition of the function.
- Functions that do not have a data type and thus do not return a value should be prototyped to be of the **void** data type. The default data type for a function is **int** (like the main program) and you can get an error/warning if you fail to provide a prototype but define the function later to be of void data type.
- The main program is in essence just another function.

## Call by reference versus call by value

In **call by value**, a *copy* of the value of an input parameter is passed to the function. The function can change the value of that parameter, but that value is not propagated back to the calling program.

What actually happens is that when the function is called, space for a variable that is an input parameter is **allocated** dynamically in the memory space allocated for use to the called function. When the function executes down to the last line and control returns back to the calling program, that space is **freed** and returned to the pool of unused memory.

The example earlier in these notes is call by value.

## Call by reference

The **scanf** function is an example of a function that passes its arguments by reference instead of value. By using the ampersand, what is passed to the **scanf** function is the *address in memory* where the value of the argument is stored. Since the function has been passed an address, if it stores a new value in that variable (as is the purpose of the **scanf** function), then the value is stored in a location *in the memory space of the calling program*, not in the memory space of the called program.

In general, for a C variable **myvalue**, the expression **&myvalue** refers to the *address* of the variable **myvalue** and not to the value stored.

A MAJOR SOURCE OF ERROR: The calling program refers to a variable **myvalue** and to its address **&myvalue**.

The called program will normally refer to a variable **\*myvalue** and to its address **myvalue**.

This is simply an artifact of the way that C is written.

You will make mistakes with this, but you will also learn to recognize what the compiler tells you as warning messages of this problem.