FUNCTIONS IN C++

Objectives/Topics
- Defining Functions
- Calling Functions
- Parameters
- Passing Arguments to Parameters
- Prototypes

How are they defined?

```cpp
return_type function_name(parameters){
    statements constituting the function
}
```

- The parameter list inside () may be empty; however, the () must still be there.
- The function_name can be any user defined identifier that hasn’t been used already for some other purpose.
Example: Read an integer.

```cpp
int GetInt( )
{
    int N;
    cout << "Enter an integer: ";
    cin >> N;
    return N;
}
```

Example

```cpp
int GetPosInt( )
{
    int N;
    cout << "Enter an integer."
    do{
        cout << " It must be > 0: ";
        cin >> N;
    }while(N <= 0);
    return N;
}
```

Calling the functions.

Once defined, functions can be called from numerous places to accomplish the specified task; i.e., the code to accomplish the task is written once and called many times.

```cpp
m = GetInt( );
n = GetPosInt( );
```

Notice the presence of ( ) in the call statements. They must be there, even if there are no parameters in the function, as they specify that these are function calls.
Call From Where?

The call to a function can be done from any function, including the function itself. To start with, our call to user-defined functions will take place in `main()`. If a function calls itself, that process is called recursion. We will discuss recursively defined functions later.

Using Parameters

Consider the following function call, and suppose that `a` and `b` are integers containing positive values.

\[ c = \text{GCD}(a, b); \]

The function GCD should compute the greatest common divisor of `a` and `b` (the largest integer that divides both `a` and `b`) and return that value.

```c
int GCD(int m, int n){
    //Invoke the Euclidean Algorithm
    //to compute greatest common divisor. Assume m,n are > 0
    int rem = m % n;
    while(rem > 0){
        m = n;
        n = rem;
        rem = m % n;
    }
    return n;
}
```
Passing Parameters

In the previous example m and n are the parameters. In the call c = GCD(a, b); the variables a and b are arguments. These are passed to the corresponding parameters (a copied into m, and b copied into n). Since m and n are variable locations associated with the function GCD and a and b are copied into them, a and b are left unchanged in their program. This is called pass by value. The values of a and b are copied to the function.

General format for functions in the same file.

include directives
using statement
function prototype list
main function
other function definitions. Their prototypes will normally be listed in the prototype list above.

Prototypes

A prototype consists primarily of the first line (header) of the function definition followed by a semicolon. Actual parameter variable names are often omitted. If variable names are present in the prototype, they are ignored.

int GetInt();
int GetPosInt();
int GCD(int, int); //Only the parameter types are specified.
Why Prototypes

Prototypes specify each function in advance, so that when a function is referred to in `main`, its meaning will have already been given. Recall that the function definitions come after the `main` program, so without a prototype, an expression such as `GCD(a,b)` would not have meaning.