C PROGRAMMING FOR ENGINEERS

Modularity
User-Defined Functions

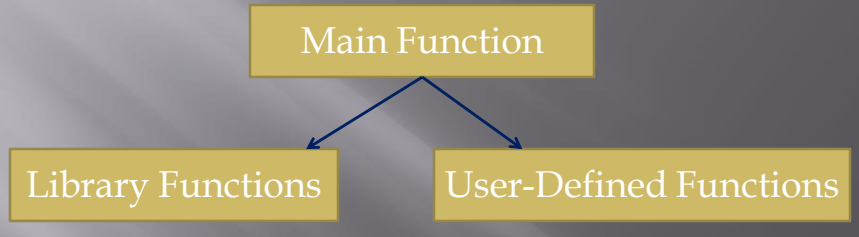
Modularity

- Execution of a program begins in the **main** function.
- The **main** function can call other functions.
  - Functions can be defined in the same file.
  - Functions can be defined in other files or libraries.
- Functions are also referred to as ***modules***.
- A module is a set of statements that performs a specific task or computes a value.

Why use modules?

- Modules can be written and tested separately.
- Large projects can be written in parallel.
- Makes the program more readable by reducing the program length.
- Modules can be reused.
- Concept of abstraction: no need to know the details of the module, but still be able to use it by knowing the general purpose of the module.

Main Function/Module Association



Functions defined in the Standard C library.
Examples:
- pow(x,y)
- sqrt(x)
- sin(x)
- printf(…)

Developed for specialized operations or calculations not contained in the standard C library.
Requires a driver program to execute.

User-Defined Function

- General form:

Type of value calculated by the function.

Name of the function.

Information passed to the function.

```
numeric_type  function_name(parameter_list)
{
        declarations;
        statements;
        return variable;
}
```

The variable type must be the same as the numeric_type defined in the function.

NOTE: the return variable can be an algebraic expression.

All functions must have a return statement.

Function Prototype Statement

```
/* Main function */
int main(void)
{
        double compute1(double x, double y);
        int compute2(int a, int b);
        …
        return 0;
}

/* A user-defined function */
double compute1(double x, double y)
{
        returns a floating-point value
}

/* A user-defined function */
int compute2(int a, int b)
{
        returns an integer value
}
```

User-defined function prototype statements.

The first prototype statement is associated to the first function, compute1. The function expects two double type values as the input and it returns a double type calculation.

The second prototype statement is associated to the second function, compute2. The function expects two integer type values as the input and it returns an integer type calculation.

Passing Parameters

- Call by value:
  - The formal parameters receive the value of the actual parameters.
  - The function cannot change the value of the actual parameters (arrays are the exception).
- Call by reference:
  - The actual parameters are pointers.

Formal and Actual Parameters

```
#include <stdio.h>
int main(void)
{
        double tempF, tempC;
        double convert(double temp);
        printf("Enter a temperature in degrees Fahrenheit:  ");
        scanf("%lf", %tempF);
        tempC = convert(tempF);
        printf("\nThe temperature in degrees Celsius is %4.1f", tempC);
        return 0;
}

double convert(double temp)
{
        double C_temp;
        C_temp = (5.0/9.0)*(temp-32.0);
        return C_temp;
}
```

Formal Parameter: holds the value needed for the function.

Actual Parameter: used when calling (invoking) the function.

Function Parameter: holds the value received from the calling function. Does not have to be the same name as the Formal Parameter, just the same numeric type.

NOTE: When the function is invoked, the value for the actual parameter will be copied to the formal parameter.

Coercion of Arguments

- It is important that the numeric type of the formal parameters matches the numeric type of the actual parameters. In fact the parameters must match in order, number, and type. If they do not, the program will coerce the values.
- Example:

```
int maximum(int a, int b)
{
    if(a > b)
        return a;
    else
        return b;
}
```

Example 1:

Actual Parameters		Formal Parameters
x	3	a
y	8	b

The function will return _____ as the maximum.

Example 2:

Actual Parameters		Formal Parameters
x	2.8	a
y	4.6	b

The function will return _____ as the maximum.

Void Functions

- Void functions are used for:
  - Performing a specific task.
  - Modifying data.
- Void functions do not return a value to the calling program.

Void Function (general form)

The word void must be first.

Should have a meaningful function name.

```
void function_name(parameter list)
{
        declarations;
        statements;
        return;
}
```

The return statement in a void function does not contain a variable or expression, and does not return a value to the calling location.