EASTERN WASHINGTON UNIVERSITY
start something big

C Function

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Recall Last Class

- More Basic I/O with I/O redirection
- Looping has same usage as in Java
  - While
  - For
  - Do while

Today Class

- C functions
  - Why use functions?
  - How to write(define) functions?
  - How to use(call) a function?
  - Good practices

Concept of Function

- A function is a group of statements that together perform a task.
- The C standard library provides numerous built-in functions that your program can call.
  - For example, function **strcat()** to concatenate two strings, function **memcpy()** to copy one memory location to another location and so on.

Concept of Function

- Every C program has at least one function, which is **main()**,
  - and in your programs you can define additional functions.
  - A function is known with various names like a method or a sub-routine or a procedure.

Why Use Functions?

- The divide-and-conquer approach makes program development more manageable.
  - With functions, we break program into smaller pieces.
- Another motivation is software reusability
  - Using existing functions as building blocks to create new programs.
  - Software reusability is a major factor in the object-oriented programming also.

Why Use Functions?

- Another motivation is software reusability
  - E.g. You write an application that can do both remote file transfer and online video chat, like the instant message apps.
  - You can write the low-level netIO library first,
    - define functions that sending/receiving stream of characters or array of bytes, or do Error validation and corrections.
  - Then the netIO library can be shared by file transfer module and video chat module in your application.

Why Use Functions?

- We use **abstraction** each time we use standard library functions like printf, scanf and pow.
  - Information hiding.
  - We do **not** care about how printf() display a double or a float value on screen.
  - We just use the service provided by printf() function.
- A third motivation is to avoid repeating code in a program.
  - Packaging code as a function allows the code to be executed from other locations in a program simply by calling the function.

Define a C Function

```
return_type   function_name( parameter list )
{
    body of the function
}
```

Components of a Function

- Return Type: A function may return a value of a particular type, such as double or int.
  - Some functions perform the desired operations without returning a value.
  - In this case, the return_type is the keyword void.
- Function Name: this is the actual name of the function.
  - The function name and the parameter list together constitute the **function signature**.

Components of a Function

- Parameters:
  - A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as **actual parameter** or **argument**.
    - E.g. int i = 100;
    - printf("%d", i); // variable i is the actual parameter here!
  - The parameter list refers to the type, order, and number of the parameters of a function.
    - A function may contain no parameters.

Components of a Function

- Function Body
  - The function body contains a collection of statements that define what the function does.

Define a Function

```
/* function returning the max between two numbers */
int max(int num1, int num2)
//num1 and num2 are called formal parameters
{
   /* local variable declaration */
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;

    return result;
}
```

Function Declaration/Definition

- A **function declaration** tells the compiler about a function name and how to call the function.
  - Tell the compiler that the function has been defined and implemented somewhere.
  - For the above defined function max(), following is the function declaration:
    - int max(int num1, int num2);
    - int max(int, int);
- A **function definition** is about writing the function body and completing the function.

Function Declaration/Definition

- Function declaration is required when you define a function in one source file and you call that function in another file.
- In such case you should **declare** the function at the top of the file calling the function.

Call a Function

- To call a function, you simply need to pass the required parameters along with function name,
  - and if function returns a value, then you can store returned value.

Demo Code

- maxDemo.c
  - Demo of function declaration, definition and invocation.
- globalWarming.c

Good Practices and Tips

- Each function should be limited to performing a single well-defined task.
- The function name should express that task. This helps abstraction and promotes software reusability.
- If you cannot choose a concise name that expresses what the function does, it is possible that your function is attempting to perform too many diverse tasks.
  - Then it's best to break such as function into several smaller functions. This is called **decomposition**.

Good Practices and Tips

- Defining a parameter **again** as a local variable in a function is a compilation error.
- Defining a function inside another function is a syntax error.
- **Small** functions promote software reusability,
  - But some people many say this undermines the performance.

Good Practices and Tips

- Programs should be written as collections of **small functions**. This makes programs easier to write, debug, maintain and modify.
  - A function requiring a large number of parameters may be performing too many tasks. Decompose it.
  - The function header should fit on one line if possible.

Good Practices and Tips

- **Thoroughly test each of functions you defined, before you use them in another function of your own, which is called Unit Test.**
  - **E.g. to test int max(int a, int b) function**
  - **we call this function in a small program, in order to validate the function by calling**
  - **max(4, 9)**
  - **max(8, 2)**
  - **max(0, 0)**
  - **max(-2, 7)**
  - **The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.**

Summary

- C functions
  - Why use functions?
  - How to write(define) functions?
  - How to use(call) a function?
  - Good practices and tips

Next Class

- C arrays