UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #14 – Functions (I)

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept

Administrative Issues

- Lab#6
  - Due **5pm, Wednesday, March 1**
- Homework#3
  - Due **Friday, March 3**

Review of Lectures #13 (Loops)

- Counter-controlled repetition vs. sentinel/event-controlled repetition
- Three *C* loop statements
  - *while* loops
  - *do…while* loops
  - *for* loops

```
while (expression)
{ loop_body
}
```

```
do
{loop_body
} while (expression);
```

```
for (statement1;statement2;statement3)
{
        loop_body
}
```

- break/continue statements can be used to change the flow of control in loops
  - *break* is used to escape from a loop or cause a loop to terminate.
  - *continue* is used to skip the remaining statements in the body of a structure and skip to the next iteration.

Outline

- **Basic concepts**
- Function declaration, call, and definition
- Function types
- Common programming errors

What is a function?

A function is an independent module that somebody calls it in order to perform a specific task.

In general, the purpose of a function is to receive zero or more pieces of data, operate on them, and return zero or some pieces of data.

main()

- main() is also a function.
- In C, a program consists of one or more functions. One and only one function is called main(), and that is where program execution always starts.
- Who calls main()?
  - The operating system does.
- main() can call other functions to perform some part of the job

Functions and Variables

- Like variables, functions have types associated with them; and functions and their type must be declared prior to their use in a program

- Like variable names, function names must conform to the naming rules for identifiers

Identifier Name Rules (Review, L#3)

- The first character can not be a digit. It has to be an alphabetic character or underscore.
- The identifier name must consist only of alphabetic characters, digits, or underscores.
- First 31 characters of an identifier are significant/used.
- DO NOT use a C reserved word /keywords (e.g., **int**).

An Example

- Define a function to compute the average of two integer numbers

```
/* defining the Function */
float average_2(int num1, int num2)
 {
  float local_average;
  local_average =  (num1 + num2)/2.0;
  return local_average;
 }
```

Function Usage

- Function declaration
  - Function prototype
- Function call (using it)
  - Statement section of the function that calls it
  - It transfers program control to the function
- Function definition
  - Usually after the function that calls it
  - Contains the code needed to complete the task

An Example (Cont'd)

```
#include <stdio.h>

/* Declaration of the function prototype */
float average_2(int num1, int num2);

int main(void)
{
   int n1;
   int n2;
   int n3;
   int n4;
  float avg_num;
  printf("Enter  two integers: ");
  scanf("%d %d", &n1, &n2);

  /* Calling the Function: function_name
     followed by arguments enclosed in () */
  avg_num = average_2(n1,n2);

  printf("The average of the first pair  is
     %f\n",avg_num);

  printf("Enter next two integers: ");
  scanf("%d %d", &n3, &n4);

  /* Calling the Function again */
  avg_num = average_2(n3,n4);

  printf("The average of the second pair  is
     %f\n",avg_num);

  return 0;
}
```

```
/* defining the Function to compute average of 2
      numbers */
float average_2(int num1, int num2)
 {
  float local_average;
  local_average =  (num1 + num2)/2.0;
  return local_average;
 }
```

Explanation

- How do we call a function?
  - Write function name followed by arguments enclosed in parentheses
- What does a function call do?
  - Transfers program control to the function
- What happens after the function finishes execution?
  - Control goes back to the location at which the function was called
- In this example program, have we passed any information from average_2() to main()?
  - The value of the variable local_average

Explanation (Cont'd)

In this example program, have we passed any information from main() to average_2()?

- When the function average_2 is called for the first time, the value of n1 in main() is assigned to the variable num1 in average_2, the value of n2 in main() is assigned to the variable num2 in average_2

- When the function average_2 is called for the second time, the value of n3 in main() is assigned to the variable num1 in average_2, the value of n4 in main() is assigned to the variable num2 in average_2

Advantages of Using Functions

- Reusability of code.
  - A function can be called in many different parts of a program, by using a simple statement.
- Data protection.
  - Data of a function are considered local to the function. Nobody else can manipulate them.

Scope

- Scope determines the region of a program in which a defined object is visible.
- Global scope.
  - Variables here are visible to every part of the program.
- Local scope.
  - Variables defined with a block {} or function have local scope. They are invisible outside the function or block.

Types of Functions

- Like data, functions also have types. This is the type of the data they return.

- A function can return
  - an int, a float, etc...
  - or it can return nothing (void).

- A function can have arguments
  - int, float,etc…
  - or it can have no arguments (void).

Types of Functions (Examples)

- float average_2(int num1, int num2)
- void function_name(int arg1, int arg2, float arg3)
- int function_name(char arg1, float arg2, int arg3)
- void function_name(void)
  - Example: void main(void)
- float function_name(void)

Common Programming Errors (1)

- Make sure that the function prototype matches exactly the function's definition (return type, function name, number, types, and order of arguments). Otherwise, you will get a compile error.
- Put a semicolon at the end of the function prototype.
- DO NOT use a semicolon at the end of the function header definition.

Common Programming Errors (2)

- Ensure that what you are returning from the function matches the return type of the function.
- The type of each function argument must be individually defined.
- DO NOT define a function inside another function.
- When the function has no arguments, remember to put the parentheses when you call them

An Example *(revisit)*
(compute the average of 2 numbers)

```
#include <stdio.h>

/* Declaration of the function prototype */
float average_2(int num1, int num2);

int main(void)
{
   int n1;
   int n2;
   int n3;
   int n4;
  float avg_num;
  printf("Enter  two integers: ");
  scanf("%d %d", &n1, &n2);

  /* Calling the Function: function_name
     followed by arguments enclosed in () */
  avg_num = average_2(n1,n2);

  printf("The average of the first pair  is
     %f\n",avg_num);

  printf("Enter next two integers: ");
  scanf("%d %d", &n3, &n4);

  /* Calling the Function again */
  avg_num = average_2(n3,n4);

  printf("The average of the second pair  is
     %f\n",avg_num);

  return 0;
}

/* defining the Function */
float average_2(int num1, int num2)
 {
  float local_average;
  local_average =  (num1 + num2)/2.0;
  return local_average;
 }
```

Exercises (1)

Find the errors, if any, in the following definitions of functions:

a)
```
void int(void)
{
    printf("Hello\n");
}
```

b)
```
void f1(int x, y)
{
    printf("Hi\n");
}
```

c)
```
void  f1(int x, int y)
{
    printf("Hi\n");
    void f2(void)
    {
    printf("Hello\n");
    }
}
```

d)
```
int f2(void)
{
    printf("Hello\n");
}
```

e)
```
void f3 (void)
{
    printf("Hi\n");
    return 0;
}
```

Exercises (2)

Given the following function prototypes and variable declarations, find errors, if any, in the function calls

```
void func1(void);
void func2(int n, double x);
void func3(double n1, int n2, double n3, int n4);
void func4(int y, int z, int w, int x);
void main(void)
{
        int a,b,c,d,e;
        double r,s,t,u,v;
        ……
```

```
func1(a);

func2(a, b);

func2(r, s);

func3(r,a,s,b);

func3(r,a,r,a);

func4(a,b,c,d,e);

func4(r,s,t,u);

}
```

Summary of Lectures #14

- A function is an independent module that somebody calls it in order to perform a specific task.
- Every C program contains one and only one main()
- Type of a function is the type of the data it returns.
- Functions must be declared before being used in a program
- Information can be passed between a function and the function that calls it
- A list of common programming errors about functions

Things To Do

- Homework #3
  - **Due by March 3 (Friday)**

Next Topic

- Functions (Cont'd)