Functions in C (part 1 of 3)

CMSC 104 Section 02
April 10, 2024

Administrative Notes

Remember that Homework 6 is due next Monday

Classwork 7 will start today; it’s due next Wednesday (April 17)

Quiz 4 is in two weeks - April 24

Functions: what, why and how

Remember that the best way to write a software program is to “build a little, test a little.”

- Functions are a way to do that
  - You write and test one function at a time
  - It’s a key step in the process of “top-down design” of a program
    - Take your big problem; break it up into smaller problems
    - Take some of these smaller problems and break them up into even smaller, simpler programs
    - The goal is to get to a set of really simple problems, each of which is straightforward to code a solution for
- Functions have the side benefit that they encourage the reuse of software
  - Once you write a function to solve a problem, you can put that same function in any program that you want
    - Import or copy-and-paste

Okay, so what's a "function?"

In C, it's a block or module of code that:

- Has a specific name by which it's called
- Takes zero or more values as inputs or *parameters*
- Executes specific tasks
- Returns zero or more values

More specifically:

- When you call a function, program control passes to that function
  - The computer executes the code statements that make up that function
    - Sequential code, conditional code, loops, …
  - No other function executes while this function is executing
- When the function ends, program control passes back to the calling location
  - The function or main program that called the function

The special function “main”

Every C program must include a function called ***main***

- Program execution starts with ***main*** - the first line in ***main*** is the first line of code executed in the program.
- In this class:
  - ***main*** returns one value, which is of type int and we set it to 0
  - ***main*** does not take any parameters; there are no inputs to main
- That is not required by C; there are programs where ***main*** takes inputs and there are programs where ***main*** returns other types or returns nothing at all

Predefined functions

There are a lot of functions that have been written by other people and that are generally useful

They are packaged with C as part of the language or part of a library that is included in a program

Examples:

- printf()
- scanf()
- getchar()
- srandom()
- random() (from CW 6)

Parameters and return values

The items you input to a function are called ***parameters***

- ***Formal parameters*** are identified when you define a function in your program
- ***Actual parameters*** are the values (variables, constants or literals) you pass to the function when you call it

Return values are items you put in a “return” statement in the function

- They are associated with the function’s name and are available for use in the calling routine (e.g., the main program)
- Sometimes called “***output parameters***”

Enough of this nonsense: let's write your own function

My example "hw6.c" program from Monday

```
/**************************************
#include <stdio.h>
// Constant values to use if you don't want to do any Extra Credit
#define LOW 1
#define HIGH 100
int main() {
  // Variables to use without any Extra Credit embellishments
  char response;  /* read in h/l/y answer from the user */
  char cr;        /* read in carriage return, but don't really need to use */
  int guess;      /* program's guess of user's secret number */
  // Variable(s) to use for the Extra Credit embellishments
  int min = LOW;        /* lowest number of user's range for program to guess */
  int max = HIGH;       /* highest number of user's range for program to guess */

  printf("Think of a number between %d and %d.\n", min, max);
  printf("I will guess the number, then tell me if my guess is\n");
  printf("too high (enter 'h'), too low (enter 'l'), or correct\n");
  printf("(enter 'y' for 'yes').\n\n");
    do {
    guess = (min + max) /2;
    printf("Is it %d  [(h)igh, (l)ow, (y)es]", guess);
    scanf("%c", &response);
    scanf("%c", &cr);
    switch(response) {
      case 'h':max = guess; break;
      case 'l': min = guess; break;
    case 'y': printf("YAY!  I got it \n"); break;
    default: printf("[WARNING]: Invalid response, must be h/l/y! \n");
    }
    //     printf("min %d and max %d \n", min, max);

    if (min == max){
      printf("You're cheating; that must be the right answer\n");
    }
  } while ((response != 'y') &&( min != max));

return 0;
}
```

Now rewritten to use a function

```
#include <stdio.h>
// Constant values to use if you don't want to do any Extra Credit
#define LOW 1
#define HIGH 100
int main() {
  // Variables to use without any Extra Credit embellishments
  char response;  /* read in h/l/y answer from the user */
  //char cr;        /* read in carriage return, but don't really need to use */
  int guess;      /* program's guess of user's secret number */
  // Variable(s) to use for the Extra Credit embellishments
  int min = LOW;        /* lowest number of user's range for program to guess */
  int max = HIGH;       /* highest number of user's range for program to guess */
  void PrintMessage(int min, int max);
  char CheckGuess( char g);

 PrintMessage(min, max);
  // do-while loop to guess user's secret number
      guess = (min + max) /2;
    response = CheckGuess(guess);

    switch(response) {
      case 'h':max = guess; break;
      case 'l': min = guess; break;
    case 'y': printf("YAY!  I got it \n"); break;
    default: printf("[WARNING]: Invalid response, must be h/l/y! \n");
    }
    //     printf("min %d and max %d \n", min, max);

    if (min == max){
      printf("You're cheating; that must be the right answer\n");
    }
  } while ((response != 'y') &&( min != max));

return 0;
}
```

```
void PrintMessage(int min, int max) {
   printf("Think of a number between
%d and %d.\n", min, max);
   printf("I will guess the number, then
tell me if my guess is\n");
   printf("too high (enter 'h'), too low
(enter 'l'), or correct\n");
   printf("(enter 'y' for 'yes').\n\n");
}

char CheckGuess(char guess) {
   char response;  /* read in h/l/y
answer from the user */
   char cr;        /* read in carriage
return, but don't really need to use */
     printf("Is it %d  [(h)igh, (l)ow,
(y)es]", guess);
     scanf("%c", &response);
     scanf("%c", &cr);
     return response;

}
```

Function prototype, function definition and *function call*

```
#include <stdio.h>
// Constant values to use if you don't want to do any Extra Credit
#define LOW 1
#define HIGH 100
int main() {
   // Variables to use without any Extra Credit embellishments
   char response;  /* read in h/l/y answer from the user */
   //char cr;        /* read in carriage return, but don't really need to use */
   int guess;      /* program's guess of user's secret number */
   // Variable(s) to use for the Extra Credit embellishments
   int min = LOW;        /* lowest number of user's range for program to guess */
   int max = HIGH;       /* highest number of user's range for program to guess */
   void PrintMessage(int min, int max);
   char CheckGuess( char g);


   PrintMessage(min, max)
   // do-while loop to guess user's secret number
      guess = (min + max) /2;
    response = CheckGuess(guess);

     switch(response) {
       case 'h':max = guess; break;
       case 'l': min = guess; break;
     case 'y': printf("YAY!  I got it \n"); break;
     default: printf("[WARNING]: Invalid response, must be h/l/y! \n");
     }
         if (min == max){
       printf("You're cheating; that must be the right answer\n");
     }
    } while ((response != 'y') &&( min != max));
return 0;
}
```

Function prototype

Function definition

Function call

```
void PrintMessage(int min, int max) {
   printf("Think of a number between
%d and %d.\n", min, max);
   printf("I will guess the number, then
tell me if my guess is\n");
   printf("too high (enter 'h'), too low
(enter 'l'), or correct\n");
   printf("(enter 'y' for 'yes').\n\n");
}

char CheckGuess(char guess) {
   char response;  /* read in h/l/y
answer from the user */
   char cr;        /* read in carriage
return, but don't really need to use */
     printf("Is it %d  [(h)igh, (l)ow,
(y)es]", guess);
     scanf("%c", &response);
     scanf("%c", &cr);
     return response;


}
```

General syntax for function definitions in C

```
type functionName(parameter_1, ..., parameter_n) {
    variable declaration(s);
    statement(s);
    return varName; // if there is something to return
}
```

- If there are no parameters, either `functionName()` or `functionName(void)` is acceptable.
- If the **function type** (**return type**) is void, a return statement is not required.

Comments in functions

Good practice:

A function header comment before the definition of a function is a good practice, and is required by the CMSC 104 Coding Standards. Your header comments should be neatly formatted and contain the following information:

- Function name
- Function description – what it does
- A list of any input parameters and their meanings
- A list of any return values (output parameters) and their meanings
- A description of any special conditions, if any.

An example

```
/*********************************************
*PrintMessage: prints a welcoming message
* explaining what the program will do and what
* the user will be expected to do
* parameters: min, max: the lowest and highest
numbers that can be guessed in this run of the
program
* returns: none
*********************************************/
```