Computer Programming Using C
COP 3275 - Summer 2017

Lecture 3: C Fundamentals

By Ahmed E. Khaled

/* Programming */
Recap to previous lecture!

• the development environment

• Structure of any C program
The General Form of a Simple Program

• C programs rely on three key language features:
  – Directives
  – Functions
  – Statements
The General Form of a simple program

directives

main()
{
    statements
}
Create simple C program

#include <stdio.h>

main()
{
    printf("Hello World\n");
}
Compiling and Linking

• Before a program can be executed, three steps are usually necessary:
  – Preprocessing: The preprocessor obeys commands that begin with # (known as directives)
  – Compiling: A compiler translates the program into machine instructions (object code).
  – Linking: A linker combines the object code produced by the compiler with any additional code needed to yield a complete executable program.

• The preprocessor is usually integrated with the compiler.
Lecture 3: C Fundamentals

• Printing Strings
• Comments
• Variables and Types
• Declaration and assignment
• Printing variables
Printing Strings

- `printf` function displays a *string literal*

- String is a set of characters enclosed in double quotation marks—it doesn’t show the quotation marks.

- `printf` doesn’t automatically advance to the next output line when it finishes printing.

- To make `printf` advance one line, include `\n` (the *new-line character*) in the string to be printed.
Printing Strings

• The statement

```c
printf("Hello World.\n");
```

could be replaced by two calls of `printf`:

```c
printf("Hello World.\n");
printf("\n");
```

• The new-line character can appear more than once in a string literal:

```c
printf("Hello World:  \n C fundamentals\n");
```
Comments

- Our application source files, still lack something important which is “Documentation”.

- Any program should contain some information that include: the author, the purpose of this program, and some details.

- In programming languages, such information is placed in “Comments”
Comments

• A comment begins with /* and end with */.

/* This is a comment */

• Comments may appear almost anywhere in a program, either on separate lines or on the same lines as other program text.

• printf("Cat"); /* Printing Cat */
Comments

• Comments may extend over more than one line.

/* Name: HelloWorld.c
   Purpose: Prints a greeting message.
   Author: XXX */

#include <stdio.h>
main()
{
    printf("Hello World\n");
}

Comments

• *Warning*: Forgetting to terminate a comment may cause the *compiler to ignore part of your program*:

```c
printf("My "); /* forgot to close this comment...
printf("cat ");
printf("has "); /* so it ends here */
printf("fleas");
```
Comments

• Single line comment can also be written in the following way:

```
// This is a comment

printf("Cat");    // Printing Cat
```

• This style of comment ends automatically at the end of a line.

• Advantages of // comments:
  – Safer: there’s no chance that an un-terminated comment will accidentally consume part of a program.
Guess the output in these cases

```c
printf("Cat");    // Printing Cat

//printf("Dog");    // Printing Cat

//printf("house");    //printing house

/*
   printf("Car");    //printing cat
*/
```
Variables

• Most programs need to perform a series of calculations before producing outputs.

• Such programs need a way to store data temporarily during program execution.

• In most of the programming languages, such storage locations are called variables.
Variables

• Variables are used to store information to be referenced and manipulated in a computer program.

• They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves.

• It is helpful to think of variables as containers that hold information and store data in memory. Such data can then be used throughout your program.
Types

• Every variable must have a Type.

• The type of the variable specifies what kind of data such variable can hold.

• Proper selection of the variable type is very important, since the type affects:
  – How the variable is stored in memory
  – What operations can be performed on the variable
Bits and Bytes!

- Computer is nothing more than a *set of switches*, either on / off.
- These 0’s and 1’s are interpreted as *digits in the binary number system* and are called *bits*.
- Eight bits form *Byte*, which is the minimum storage unit in a computer.

- Data of various kinds (numbers and characters), are encoded as *a series of bytes*. E.g. character C is represented as 01000011 in *one byte*. 
• A computer’s storage capacity is measured in *multiples of the byte*:
  – Kilobyte (KB) is about 1,000 bytes.
  – Megabyte (MB) is about 1 million bytes.
  – Gigabyte (GB) is about 1 billion bytes.
  – Terabyte (TB) is about 1 trillion bytes.
Types

- Type for a numeric variable determines the range of values that this variable can store, and whether or not digits are allowed after the decimal point.
Types

- C language has a wide variety of types, for today’s lecture we will introduce `int` and `float`.

- A variable of type `int` (short for `integer`) can store a *whole number* such as 0, 1, 392, or –2553.
  - The range `int` value is typically -32,768 to 32,767, to be stored in 2 `bytes`
Types

• A variable of type float (short for floating-point) can store much larger numbers than an int variable.

• Also, a float variable can store numbers with digits after the decimal point, like 379.125.

• The float variable is stored in 4 bytes.

• Drawbacks of float variables:
  – Slower arithmetic
  – Approximate nature of float values
• Variables must be *declared* before they are used, for the benefit of the compiler.

• Variables are declared by *specify the type*, give a *name* for the variable and end with a *semicolon* as follows:

```plaintext
int height;
```
Declarations

```
int height;
```

The compiler created variable named height, of type int (this variable can store an integer value).

```
float profit;
```

The compiler now creates variable named profit, of type float (this variable can store a floating point value).
Declarations

• Any statement in C ends with *semicolon*.
• Alternatively, several can be declared at the same time:

```c
int height, length, width, volume;
float profit, loss;
```
Assignment

• A variable can be given a value by means of assignment:

\[
\text{height} = 8;
\]

the compiler assign value of 8 to the variable height, that is of type int (how?).
Assignment

• Before a variable can be assigned a value—or used in any other way—it must first be declared.
Assignment

```cpp
int height;  // declare height
int width;   // declare width
height = 8;  // assign value to height
Width = 5;   // assign value to width

float profit; // declare variable profit
Profit = 58.4; // assign a float value to profit
```
Assignment

int height; //declare height

//what is the value of height?
(it has an indeterminate value)

height = 8; //assign value to height
//what is the value of height?

height = 10; //assign value to height
//what is the value of height?
Assignment

• Once a variable has been assigned a value, it can be used to help *compute the value of another variable*:

```c
int x;
int y;
x = 8;
y = x + 2;  // y holds the value of 10
```
Assignment

- The right side of an assignment can be a formula (or *expression*, in C terminology) involving constants, variables, and operators.

```c
int x;
int y;
int z;
x = 8;
y = 10;
z = x * y;  // z holds the value of 8 * 10 = 80
```
Declare and Assign

Some time to avoid the *undetermined values* of the variable, you can assign value during the *declaration phase*.

\[
\begin{align*}
\text{int X;} & = \quad \text{int X = 10;} \\
X = 10; & \\
\text{float profit;} & = \quad \text{float profit = 50.23;} \\
\text{profit = 50.23;} & 
\end{align*}
\]
Printing the Value of a Variable

• To write the message \texttt{Height: } h where \( h \) is the current value of the \texttt{height} variable, we’d use the following call of \texttt{printf}:

\[
\texttt{printf("Height: \%d\n", height);}\]

• \texttt{\%d} is a placeholder indicating where the value of \texttt{height} \textit{is to be filled in}. 
Printing the Value of a Variable

- There’s no limit to the number of variables that can be printed by a single call of `printf`:

```c
printf("Height: %d  Length: %d\n", height, length);
```