

C Programming Language

How to write & variable declaration

Dr. Mostafa A. Elhosseini

Revise

- ✂ Hello world layout
- ✂ Function
 - Parenthesis
 - Argument
 - void as an argument
 - Int as a return type
 - return
- ✂ Block
 - braces
- ✂ Header file
- ✂ String constant

Agenda

✂ How to print output?

- printf – print formatted
- Format specifier
- How to print constants
- Escape sequence

✂ Variable

- Variable?
- Types
- Variable declaration
- Variable definition

printf()

≡ To print formatted output on the screen

```
printf("this is the number 10");
```

message - string constant

printf()

≡ Escape sequence

- \n : new line
- \t : tab
- \\ : backslash
- \" : double quotes

```
printf("My name is Mostafa");  
printf("I am an associate professor ");  
printf("in the field of Control engineering");
```

```
My name is MostafaI am an associate professor in the field of Control engineering
```

printf()

≡ Escape sequence

- \n : new line : has the effect of a carriage return and linefeed, but stored in memory as a single character “linefeed”
- \t : tab
- \\ : backslash
- \" : double quotes



```
printf("My name is Mostafa\n");  
printf("I am an associate professor\n\n ");  
printf("in the field of Control engineering");
```

printf()

- ≡ To print **f**ormatted output on the screen
- ≡ Format specifier
 - **%d** : signed decimal integer
- ≡ Takes the value on the right of the comma and plugs it into the string on the left – in place of format specifier

```
printf("Number of students is %d", 45);
```



format specifier

The diagram consists of a red rectangular box containing the code `printf("Number of students is %d", 45);`. A red arrow points from the text "format specifier" below to the `%d` in the string. A red curved arrow points from the `45` to the `%d`, indicating its substitution.

printf()

- ≡ To print formatted output on the screen
- ≡ Format specifier
- ≡ %s : string: double quotes
- ≡ %c : character single quote



The diagram shows a red rectangular box containing the code `printf("the grade of %s is %c", "Ali", 'A');`. Two curved arrows originate from above the code. An orange arrow starts at the first `%s` and points down to the string `"Ali"`. A purple arrow starts at the first `%c` and points down to the character `'A'`.

```
printf("the grade of %s is %c", "Ali", 'A');
```

Variable

- ≡ A **space in computer memory set aside for a certain kind of data** and given a name for easy reference
- ≡ You should declare a variable first before the first time it used, and should be in the first of program
 - Placing variable definitions among executable statements causes syntax error in microsoft Visual C++ compiler
- ≡ Variable definition
 - Name
 - Type
 - value
- ≡ Variable types
 - Integer %d
 - Float %f
 - Character %c
 - String %s

Variable Name

- ≡ Only letter, digits, and the underscore characters are allowed
- ≡ The first character may be a character or underscore but not numbers
- ≡ Avoid the use of underscore as the first letter of a variable
 - To prevent conflicts with compiler-generated identifier, and standard library identifier
- ≡ theta in_radian, index32,sine_value → right
- ≡ 32index, theta-in-radian, sin\$value → wrong
- ≡ thetaInRadian, SineValue → perfect
- ≡ Please use **meaningful identifier** name

■ Variable

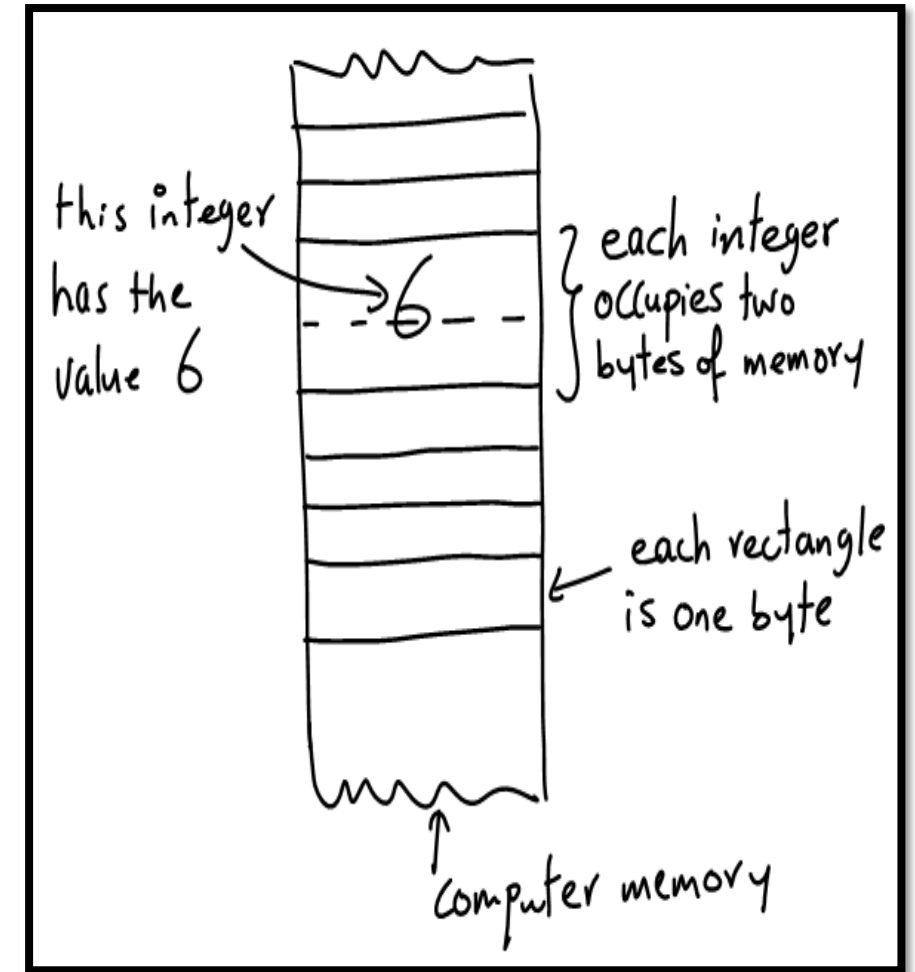
```
#include <stdio.h>
int main()
{
    int x;
    x = 10;
    printf("value of x is %d:",x);
    return 0;
}
```

"E:\Courses\Taibah\Programming essential using C\Code\printf01\bin\Debug\printf01.exe"

```
value of x is 10:
Process returned 0 (0x0)   execution time : 0.014 s
Press any key to continue.
```

■ Variable

- ≡ If you have **more than one variable** of the **same type**, you can define them all with one type name, **separating** the variable names with commas
- ≡ When **you define a variable**, the compiler **sets aside an appropriate amount of memory** to store that variable — in the present case we've specified an **integer** variable, so the compiler will set aside **two bytes** of memory
- ≡ Actually, the **amount of memory** used for an integer (and other variable types) is **not** defined as **part of the C** language — it is dependent on the particular **computer system** and **compiler** being used



Printing graphics characters

- ≡ every **character** (**letters**, **digits**, **punctuation**, and so forth) is represented in the computer by a **number**
- ≡ **True ASCII codes** run from **0 to 127** (decimal). These cover the upper and lowercase letters, digits 0 to 9, punctuation, and control characters such as linefeed and back-space
- ≡ **PC computers** use an **additional 128 characters**, with codes running from **128 to 255**. These additional characters consist of foreign-language symbols and **graphics character**
- ≡ **Graphics characters** are printed using their hex code in an escape sequence, such as `'\xB0'`