

# Lecture02: Overview of C

## 9/17/2012

Slides modified from Yin Lou, Cornell CS2022: Introduction to C

1

## Administrative things

- Office hours (Instructor & TAs) are on the course webpage
  - [http://mll.csie.ntu.edu.tw/course/comp\\_prog\\_f12/](http://mll.csie.ntu.edu.tw/course/comp_prog_f12/)
- Assignment #1 is on the course webpage due next week.
  
- Does everyone registered for the course have an account?
- Please email your studentID to the TA.

2

## Today's Do List

- Review last week
- Variable declaration
- printf()
- scanf()
- In-class practice exercises
- Introduce the Judge system (for submitting your assignments)

3

## Review: Hello World

**what is #include <stdio.h>? a library of functions**

**what are main() and printf()? called functions**

**function: a set of code enclosed in { ... } that takes some inputs, perform some operation, and produces output;**

**example: rect\_area(height, width) = height \* width**

**more about how to define and write functions next week**

```
#include <stdio.h>

int main()
{
    printf("Hello World :)\n");
    return 0;
}
```

4

## Review: Compile and Run in CodeBlocks

- Run CodeBlocks
- Create a new project
  - File → New → Project
  - Select “Console application”
  - Click C → Next
  - Type in Project title: (e.g., “hello”)
  - Click finish
- Open and edit the main.c
  - File → Open
  - Find “main.c”
- Compile and run
  - Build → Build and Run

5

## What Happened?

- Compile (Build → Build)
  - Compile “main.c” to machine code named “hello.exe”
- Run (Build → Run)
  - Execute the program “hello.exe”

```
main.c (Hello World)

include <stdio.h> /* printf() is declared in this header file. */

int main() /* Main point of execution */
{
    printf("Hello World\n"); /* Output "Hello World" to console */
    return 0; /* Tell OS the program terminates normally */
}
```

6

## main.c: Variables

a name for a place in memory that stores a value

what is computer memory?

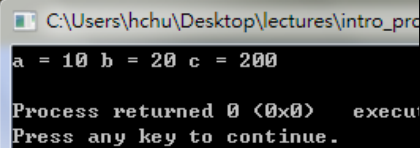
each variable must have a type

C punctuations: ; , { } ( ) " "

arithmetic (math) operators: + - \* / =

```
#include <stdio.h>

int main()
{
    int a;
    int b, c;
    a = 10;
    b = 20;
    c = a * b;
    printf("a = %d b = %d c = %d\n", a, b, c);
    return 0;
}
```



```
C:\Users\hchu\Desktop\lectures\intro_prog
a = 10 b = 20 c = 200
Process returned 0 (0x0) execution completed.
Press any key to continue.
```

## printf()

print (output) data on the user screen

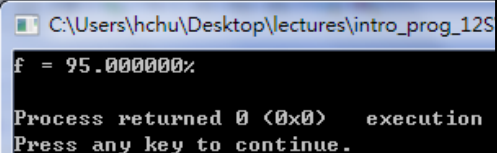
must specify each type of the output data in the format string

what's the format string?

- printf(format string, val1, val2);
  - format string can include placeholders that specify how the arguments val1, val2, etc. should be formatted
  - %c : format as a character
  - %d : format as an integer
  - %f : format as a floating-point number
  - %% : print a % character

- Examples

```
double f = 0.95;
printf("f = %f%%\n", f * 100);
```

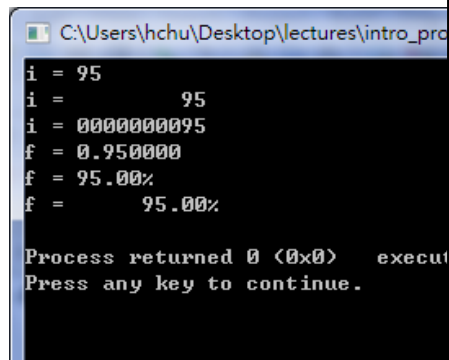


```
C:\Users\hchu\Desktop\lectures\intro_prog_125
f = 95.000000%
Process returned 0 (0x0) execution completed.
Press any key to continue.
```

## Even more on printf()

- Placeholders can also specify widths and precisions
  - `%10d` : add spaces to take up at least 10 characters
  - `%010d` : add zeros to take up at least 10 characters
  - `%.2f` : print only 2 digits after decimal point
  - `%.5.2f` : print only 2 digits after decimal point, add spaces to take up 5 chars
- Examples

```
int i = 95;
double f = 0.95;
printf("i = %d\n", i);
printf("i = %10d\n", i);
printf("i = %010d\n", i);
printf("f = %f\n", f);
printf("f = %.2f%%\n", f * 100);
printf("f = %10.2f%%\n", f * 100);
```

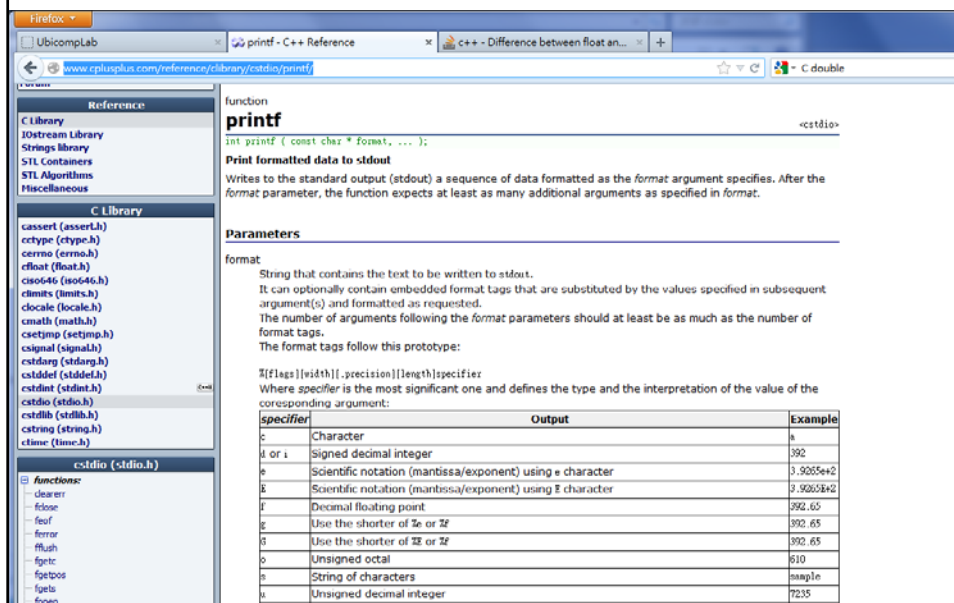


```
C:\Users\hchu\Desktop\lectures\intro_pro
i = 95
i =          95
i = 0000000095
f = 0.950000
f = 95.00%
f =          95.00%

Process returned 0 (0x0) executed successfully.
Press any key to continue.
```

## Even more on printf() format string

<http://www.cplusplus.com/reference/cstdio/printf/>



function

### printf

```
int printf ( const char * format, ... );
```

Print formatted data to stdout

Writes to the standard output (stdout) a sequence of data formatted as the *format* argument specifies. After the *format* parameter, the function expects at least as many additional arguments as specified in *format*.

#### Parameters

**format**

String that contains the text to be written to *stdout*. It can optionally contain embedded format tags that are substituted by the values specified in subsequent argument(s) and formatted as requested. The number of arguments following the *format* parameters should at least be as many as the number of format tags. The format tags follow this prototype:

`[[flags][width][.precision][length]specifier]`

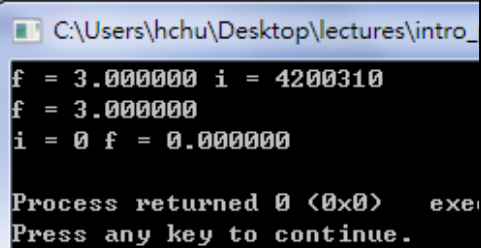
Where *specifier* is the most significant one and defines the type and the interpretation of the value of the corresponding argument:

specifier	Output	Example
c	Character	a
i or I	Signed decimal integer	392
e	Scientific notation (mantissa/exponent) using e character	3.90554e2
E	Scientific notation (mantissa/exponent) using E character	3.90554E2
f	Decimal floating point	392.60
g	Use the shorter of %e or %f	392.65
G	Use the shorter of %E or %F	392.65
o	Unsigned octal	610
s	String of characters	sample
u	Unsigned decimal integer	7235

## Warning about printf

- printf is powerful, but potentially dangerous
- What does this code output?

```
int i = 90;
double f = 3;
printf("f = %f i = %d\n", f);
printf("f = %f\n", f, i);
printf("i = %d f = %f\n", f, i);
```



```
C:\Users\hchu\Desktop\lectures\intro_...
f = 3.000000 i = 4200310
f = 3.000000
i = 0 f = 0.000000
Process returned 0 (0x0)   exe
Press any key to continue.
```

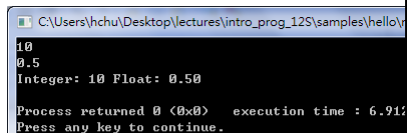
## main.c: scanf

read data from user input

again must specify the type for the input data in the format string  
note the '&' in front of the variable (called the address operator – a bit complicated to explain right now, so just remember to use it in scanf)

```
#include <stdio.h>
```

```
int main()
{
    int i;
    double f;
    scanf("%d", &i);
    scanf("%lf", &f);
    printf("Integer: %d Float: %.2f\n", i, f);
    return 0;
}
```



```
C:\Users\hchu\Desktop\lectures\intro_prog_125\samples\hello\...
10
0.5
Integer: 10 Float: 0.50
Process returned 0 (0x0)   execution time : 6.912
Press any key to continue.
```

## In-Class Exercise #1

- Write a program that calculates travel reimbursement for a pizza delivery person at a rate of NT\$6.97 per kilometer. Your program should interact with the user in the following manner: **(It is okay not to get the program right the first time. Look at the compilation errors and fix them.)**

KILOMETER REIMBURSEMENT CALCULATOR

Enter beginning odometer readings=> 13505.2

Enter ending odometer reading=> 13810.6

You traveled 305.4 kilometers. At \$6.97 per kilometer,

Your reimbursement is \$2128.63

13

## In-Class Exercise #2

- Write a program that ask for two numbers and output the results of four arithmetic operations (+, -, \*, /). Below shows the sample output.

enter a> 5

enter b> 2

a+b = 7

a-b = 3

a\*b = 10

a/b = 2

14

**After you are done with in-class  
exercises, start the assignment #1.**

**TA will talk about the judge system**

15