C Programming
Lecture 5 : Basic standard I/O

<u>Standard Input/Output (I/O)</u>

- Preconnected input and output channels between a computer program and its environment(typically a text terminal).

  - Standard input :
    - text input from keyboard
  - Standard output
    - text output written to display
  - Standard error :
    - another text output written to display for error messaging

Standard I/O library

- Library
  - A collection of subroutines (functions) used to develop software

- Standard library
  - Library that is made available in every implementation of a programming language
  - Same interface(parameter type) , same functionality in different systems

- Standard I/O library
  - Standard library for processing I/O

<u>printf function</u>

**printf*(control string,argument list);*

- Control string contains
  - Literal text to be displayed
  - format specifiers
  - Special characters

- Arguments can be
  - Variable , function, expression, constant
  - # of argument list must match the # of format identifiers

printf example

```c
7  #include <stdio.h>
8
9  int main()
10 {
11     int   i = 2;
12     double f = 3.14;
13     char  c = '5';
14
15     printf("i = %d\n", i);
16     printf("f = %f\n", f);
17     printf("c = %c\n", c);
18
19     return 0;
20 }
```

```
Output :
i =              2
f =     3.141593
c =              5
```

printf format specifiers

Specifier	Type
%c	character
%d	decimal integer
%o	octal integer (leading 0)
%x	hexadecimal integer (leading 0x)
%u	unsigned decimal integer
%ld	long int
%f	floating point
%lf	double or long double
%e	exponential floating point
%s	character string

printf examples

```c
#include <stdio.h>

int main()
{
    int    i = 2;
    double f = 3.14159265358979323846;
    char   c = '5';

    printf("i = %10d\n", i);
    printf("f = %10f\n", f);
    printf("c = %10c\n", c);

    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    double pi = 3.14159265358979323846;

    printf("pi = %10f\n", pi);
    printf("pi = %10.2f\n", pi);
    printf("pi = %10.12f\n", pi);

    return 0;
}
```

```
output :
i =          2
f =    3.141593
c =          5
```

```
output:
pi =    3.141593
pi =        3.14
pi = 3.141592653590
```

scanf function

- Accept formatted text input

```c
#include <stdio.h>

int main()
{
    int n = 0;

    scanf("%d", &n);
    printf("entered n   = %d\n", n);
    printf("double of n = %d\n", n+n);
    printf("triple of n = %d\n", n+n+n);

    return 0;
}
```

Output :
27   ←---- keyboard input
entered n   = 27
double of n = 54
triple of n = 81

gets(), puts() functions

- line based string I/O functions
- Prototype
  - **char* gets(char *BUF);**
    - Read characters from standard input until a newline is found
  - **int puts(const char *s);**
    - Writes a string s to the standard output.

```c
#include <stdio.h>

#define MAX_LINES 2

int main()
{
        char line[MAX_LINES];
        printf("string input :");
        gets(line);
        printf("the input string is : ");
        puts(line);

        return 0;
}
```

__redirection__

- Input redirection
  - Gets standard input from a file "inputFile.txt"
  - `program.exe < inputFile.txt`

- Output redirection
  - writes standard output to a file "outputFile.txt"
  - `program.exe > outputFile.txt`

- Combination
  - Gets standard input from a file "inputFile.txt" and writes standard output to a file "outputFile.txt"
  - `program.exe < inputFile.txt > outputFile.txt`

Exercise

- Write a program that **converts meter-type height into [feet(integer),inch(float)]-type height.** Your program should get one float typed height value as an input and prints integer typed feet value and the rest of the height is represented as inch type. **(1m=3.2808ft=39.37inch)**

  - **Ex) 1.80meter -> 5feet 10.9inch**

- use automatic type conversion
  - 1/2 = 0 (?) , 3/2 = 1 (?)

```
(ex)
int a;
float b;
b = 3.6/2.0;
a=b;
printf("a=%d, b=%f\n",a,b);
```