

SIEC: BASIC C PROGRAMMING

L #04: PRINFT() FUNCTION AND VARIABLES

Seung Beop Lee

School of International Engineering and Science

CHONBUK NATIONAL UNIVERSITY

Outline

- **printf() funtion**
- **Variables, Types and Declarations**

printf() funtion

printf() funtion

- printf() Function

- The printf() function provides a superbly versatile way of **printing text**.

- Way to use the printf() function

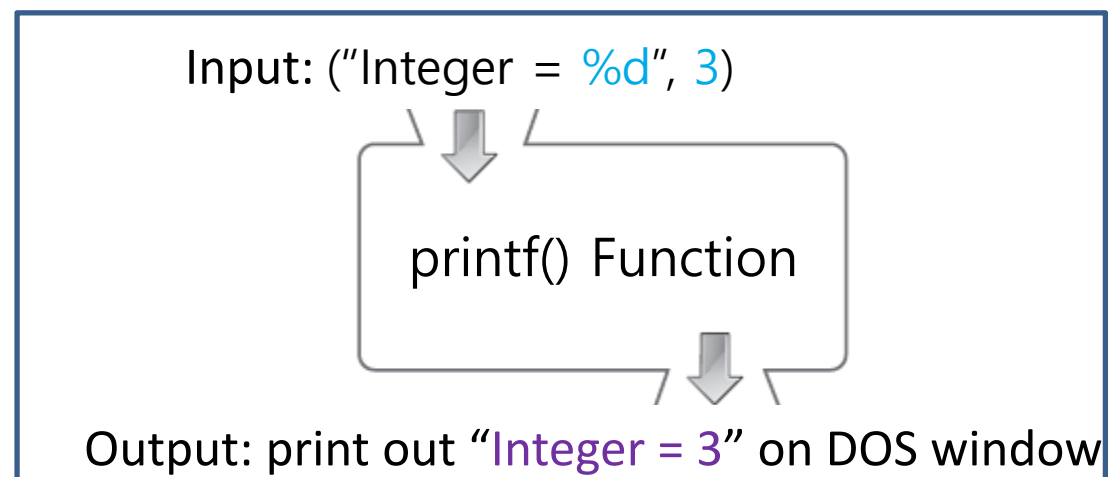
1. The **first way** is to print out **a literal string**:

```
printf("... some string ... ");
```

2. The **second way** is to print out **the contents of variables** by using **'control sequence'**.

Ex) To print out an integer, the **control sequence** **%d** is used as follows:

```
printf("Integer = %d", 3)
```



printf() funtion

- Examples for printf() Function

```
1  #include <stdio.h>
2
3  int main()
4  {
5      /* my first program in C */
6      int a, b;
7      a = 1;
8      b = 3;
9      printf("%s = %d \n", "1 곱하기 3 ", a*b);
10
11
12     printf("Astronomy is %d derful \n", 1);
13     printf("And interesting %d \n", 2);
14     printf("The ear %d volves around the sun \n", 3);
15     printf("And makes a year %d you \n", 4);
16
17
18     return 0;
19 }
```

%d is the control sequence for an **integer**.
%s is the control sequence for an **literal string**.

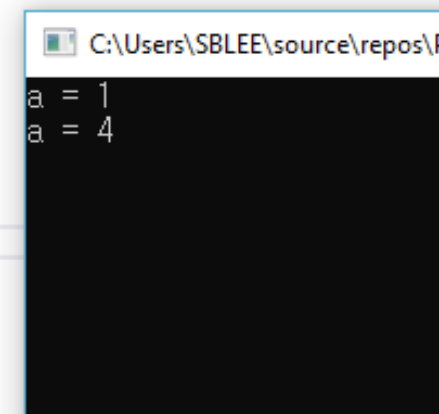
Variables, Types and Declarations

Variables, Types and Declarations

■ Variables

- A variable is a sequence of program code with a name (also called its identifier).
- A variable is nothing but a **name given to a storage area** that our programs can **manipulate**.
- The storage area of the name *a* in the following example is a space allocated to computer memory.
- So, you can manipulate the value of this memory.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      /* my first program in C */
6      int a, b;
7      a = 1;
8      printf("%s = %d \n", "a", a);
9      a = 4;
10     printf("%s = %d \n", "a", a);
11
12     b = 3;
13     printf("%s = %d \n", "1 곱하기 3 ", a*b);
14
15     return 0;
16 }
```



```
C:\Users\SBLEE\source\repos\
a = 1
a = 4
1 곱하기 3 
```

Variables, Types and Declarations

■ Variables

- A variable (i.e. name or identifier) in C can be **anything from a single letter to a word**.
- A variable (i.e. name or identifier) starts with a letter A to Z or a to z or an underscore `_` followed by zero or more letters, underscores, and digits (0 to 9), as follows:

```
mohd      zara      abc      move name  a 123  
myname50  _temp     j       a23b9     retVal
```

- However, C does **not allow punctuation characters** (i.e. `@`, `$`, and `%`) **within variable** (i.e. name or identifier).

Variables, Types and Declarations

■ Types

- In the C programming language, **data types** indicate an extensive system used for declaring **variables or functions of different types**.
- The **type of a variable** determines **how much space** it occupies in storage and **how the bit pattern** stored is **interpreted**.
- The types in C can be classified as follows:

S.N.	Types and Description
1	Basic Types: They are arithmetic types and consists of the two types: (a) integer types and (b) floating-point types.
2	Enumerated types: They are again arithmetic types and they are used to define variables that can only be assigned certain discrete integer values throughout the program.
3	The type void: The type specifier <i>void</i> indicates that no value is available.
4	Derived types: They include (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types.

Variables, Types and Declarations

▪ Integer Types(Basic Types)

- Following table gives you details about **standard integer types** with its storage sizes and value ranges:

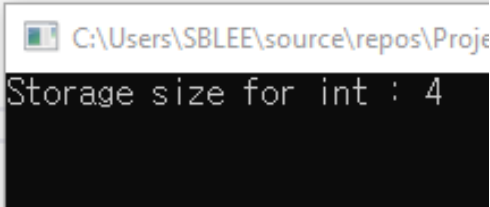
Type	Storage size	Value range	Representation
Char	1 byte	-128 to 127 or 0 to 255	A single ASCII character
unsigned char	1 byte	0 to 255	
signed char	1 byte	-128 to 127	
Int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	A standard integer
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295	
Short	2 bytes	-32,768 to 32,767	
unsigned short	2 bytes	0 to 65,535	
Long	4 bytes	-2,147,483,648 to 2,147,483,647	
unsigned long	4 bytes	0 to 4,294,967,295	

Variables, Types and Declarations

▪ Integer Types(Basic Types)

- To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions **sizeof(type)** yields the **storage size of the object or type in bytes**. Following is an example to get the size of *int* type on any machine:

```
1  #include <stdio.h>
2  #include <limits.h>
3
4  int main()
5  {
6      printf("Storage size for int : %d \n", sizeof(int));
7      return 0;
8  }
```



Variables, Types and Declarations

▪ Floating-Point Types(Basic Types)

- Following table gives you details about **standard floating-point types** with storage sizes and value ranges and their precision:

Type	Storage size	Value range	Representation
float	4 bytes	1.2E-38 to 3.4E+38 (6 decimal places)	A floating point or real number (short)
double	8 bytes	2.3E-308 to 1.7E+308 (15 decimal places)	A long floating point number
long double	10 bytes	3.4E-4932 to 1.1E+4932 (19 decimal places)	

- The header file **float.h** defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs. Following example will print storage space taken by a float type and its range values:

```
1 #include <stdio.h>
2 #include <float.h>
3
4 int main()
5 {
6     printf("Storage size for float : %d \n", sizeof(float));
7     printf("Minimum float positive value: %E\n", FLT_MIN);
8     printf("Maximum float positive value: %E\n", FLT_MAX);
9     printf("Precision value: %d\n", FLT_DIG);
10
11     return 0;
12 }
```

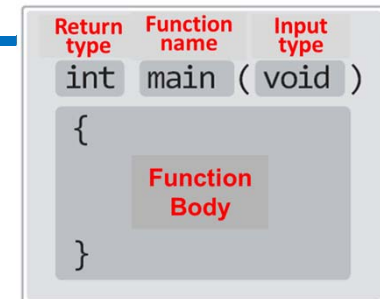
```
C:\Users\SBLEE\source\repos\Project1\Debug\Project1.exe
Storage size for float : 4
Minimum float positive value: 1.175494E-38
Maximum float positive value: 3.402823E+38
Precision value: 6
```



Variables, Types and Declarations

■ Void Type

- The void type specifies that **no value is available**.
- It is used in **three kinds of situations**:



S.N.	Types and Description
1	Function returns as void There are various functions in C which do not return value or you can say they return void. A function with no return value has the return type as void. For example, void exit (int);
2	Function arguments as void There are various functions in C which do not accept any parameter. A function with no parameter can accept as a void. For example, int main(void);
3	Pointers to void A pointer of type void * represents the address of an object, but not its type. For example, a memory allocation function void *malloc(size_t size); returns a pointer to void which can be casted to any data type.

```
1  #include <stdio.h>
2  #include <float.h>
3
4  int main(void)
5  {
6      printf("main function with no parameter = int main(void)");
7      return 0;
8  }
```

```
C:\Users\SBLEE\source\repos\Project1\Debug\Project1.exe
main function with no parameter = int main(void)
```

Variables, Types and Declarations

■ Void Type

- The void type specifies that **no value is available**. It is used in **three kinds of situations**:

S.N.	Types and Description
1	Function returns as void There are various functions in C which do not return value or you can say they return void. A function with no return value has the return type as void. For example, void exit (int status);
2	Function arguments as void There are various functions in C which do not accept any parameter. A function with no parameter can accept as a void. For example, int rand(void);
3	Pointers to void A pointer of type void * represents the address of an object, but not its type . For example, a memory allocation function void *malloc(size_t size); returns a pointer to void which can be casted to any data type.

- The void type may not be understood to you at this point, so let us proceed and we will cover these concepts in the upcoming classes.

Variables, Types and Declarations

▪ Declarations

- To declare a variable in a C program, one writes the type followed by a list of variable names which are to be treated as being that type:

```
typename variablename1,...,variablenameN;
```

- For example:

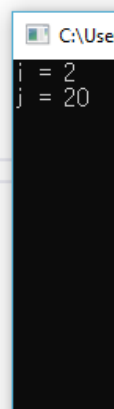
```
int i, j;  
char ch;  
double x, y, z, fred;  
unsigned long int Name_of_Variable;
```

Variables, Types and Declarations

■ Where to declare things

- There are **two kinds of place** in which declarations can be made.
 - **One place** is **outside all of the functions**. That is, in the space between function definitions. (Generally, the declaration is after the **#include** lines.) Variables declared here are called **global variables**. There are also called static and external variables in special cases.
 - The **global variables** can use in any function **without additional declaration**.

```
1 #include <stdio.h>
2 #include <float.h>
3
4 // Variable declaration
5 int i, j;
6 char ch;
7 double x, y, z, fred;
8
9 // function declaration
10 int func();
11
12 int main(void)
13 {
14     i = 2;
15     printf("i = %d \n", i);
16
17     func();
18
19     return 0;
20 }
21
22 int func()
23 {
24     j = 20;
25     printf("j = %d \n", j);
26     return 0;
27 }
28 }
```



```
C:\Use
i = 2
j = 20
```


Variables, Types and Declarations

■ Where to declare things

- There are **two kinds of place** in which declarations can be made.
 - The **other place** where declarations can be made is in the opening brace, {}, of a block. Generally, the declaration follows immediately after the opening brace. Variables of this kind **only work inside their braces {}** and are often called **local variables**.
 - The local variables are **useful only inside their braces**.

```
1 #include <stdio.h>
2 #include <float.h>
3
4 // function declaration
5 int func();
6
7 int main(void)
8 {
9     // Variable declaration
10    int i, j;
11
12    i = 2;
13    printf("i = %d \n", i);
14
15    func();
16
17    return 0;
18 }
19
20 int func()
21 {
22     j = 20;
23     // Error message: identifier "j" is undefined
24 }
25
26
```



Change

```
1 #include <stdio.h>
2 #include <float.h>
3 // Variable declaration
4 int j;
5
6 // function declaration
7 int func();
8
9 int main(void)
10 {
11     // Variable declaration
12     int i, j;
13
14     i = 2;
15     printf("i = %d \n", i);
16
17     func();
18
19     return 0;
20 }
21
22 int func()
23 {
24     j = 20;
25     printf("j = %d \n", j);
26     return 0;
27 }
28
```

Variables, Types and Declarations

■ Where to declare things

- There are **two kinds of place** in which declarations can be made.
 - The **other place** where declarations can be made is in the opening brace, {}, of a block. Generally, the declaration follows immediately after the opening brace. Variables of this kind **only work inside their braces {}** and are often called **local variables**.
 - The local variables are **useful only inside their braces**.

```
1 #include <stdio.h>
2 #include <float.h>
3
4 // function declaration
5 int func();
6
7 int main(void)
8 {
9     // Variable declaration
10    int i, j;
11
12    i = 2;
13    printf("i = %d \n", i);
14
15    func();
16
17    return 0;
18 }
19
20 int func()
21 {
22     j = 20;
23
24     identifier "j" is undefined
25
26 }
```

Error message



Change

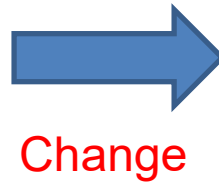
```
1 #include <stdio.h>
2 #include <float.h>
3
4 // function declaration
5 int func();
6
7 int main(void)
8 {
9     // Variable declaration
10    int i, j;
11
12    i = 2;
13    printf("i = %d \n", i);
14
15    func();
16
17    return 0;
18 }
19
20 int func()
21 {
22     // Variable declaration
23     int j;
24
25     j = 20;
26     printf("j = %d \n", j);
27     return 0;
28 }
29 }
```

Variables, Types and Declarations

■ Declarations and Initialization

- When a variable is declared in C, the language allows a neat piece of syntax which means that **variables can be declared and assigned a value in one go**.

```
1  #include <stdio.h>
2  #include <float.h>
3
4  // function declaration
5  int func();
6
7  int main(void)
8  {
9      // Variable declaration
10     int i, j;
11
12     i = 2;
13     printf("i = %d \n", i);
14
15     func();
16
17     return 0;
18 }
19
20 int func()
21 {
22     // Variable declaration
23     int j;
24
25     j = 20;
26     printf("j = %d \n", j);
27     return 0;
28 }
29 }
```



```
1  #include <stdio.h>
2  #include <float.h>
3
4  // function declaration
5  int func();
6
7  int main(void)
8  {
9      // Variable declaration
10     int i=2;
11
12     printf("i = %d \n", i);
13
14     func();
15
16     return 0;
17 }
18
19 int func()
20 {
21     // Variable declaration
22     int j=20;
23
24     printf("j = %d \n", j);
25     return 0;
26 }
27 }
```

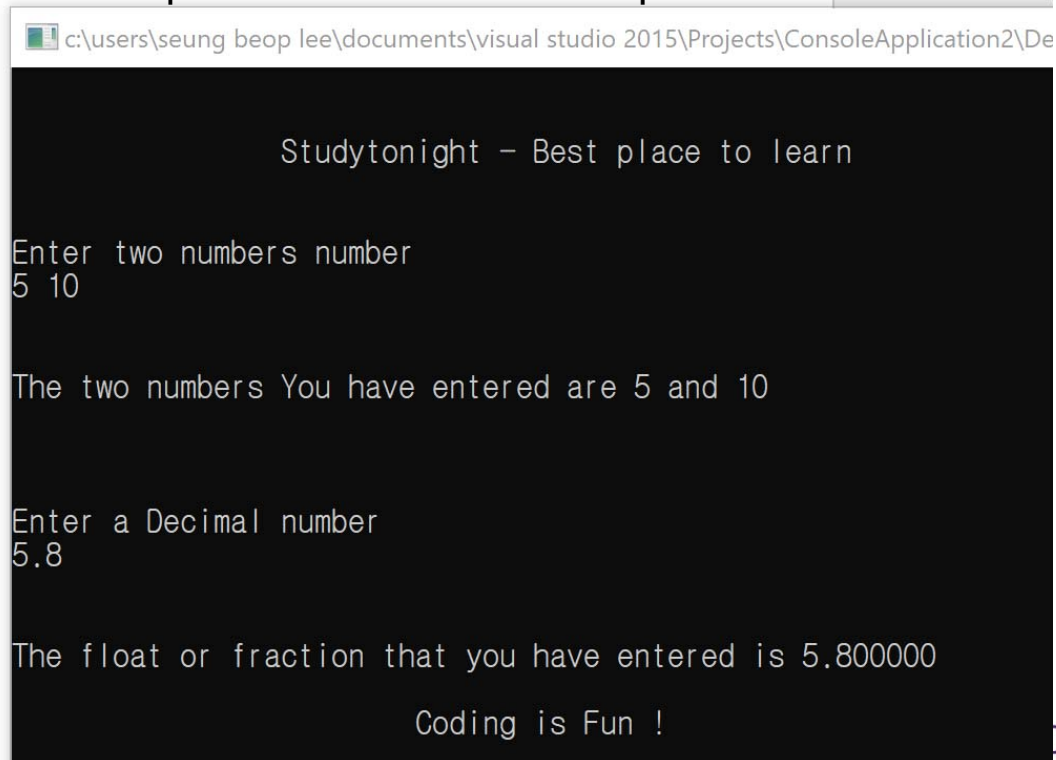
Thank You

(seungbeop.lee@gmail.com)

Self-coding

■ Problem definition

- Make a program which outputs the following result by using the input of various datatypes from user in C. The different datatypes are int(integer values), float(decimal values) and char(character values)
- ✓ `printf()` is used to display text onto the screen. `&` is used to assign the input value to the variable and store it at that particular location. `scanf_s()` is used to take input from the user using format specifier discussed in upcoming class. `%d` and `%i`, both are used to take numbers as input from the user. `%f` is the format specifier to take float as input from the user.



```
c:\users\seung beop lee\documents\visual studio 2015\Projects\ConsoleApplication2\De

Studytonight - Best place to learn

Enter two numbers number
5 10

The two numbers You have entered are 5 and 10

Enter a Decimal number
5.8

The float or fraction that you have entered is 5.800000

Coding is Fun !
```