C Programming & Lab

3. Variables and Types

Sejong University

Outline

1) Variables and Data Types
2) Variable Declaration and Initialization
3) Data Representation
4) Integer Type
5) Floating point type
6) Character Type
7) Printf
8) Scanf

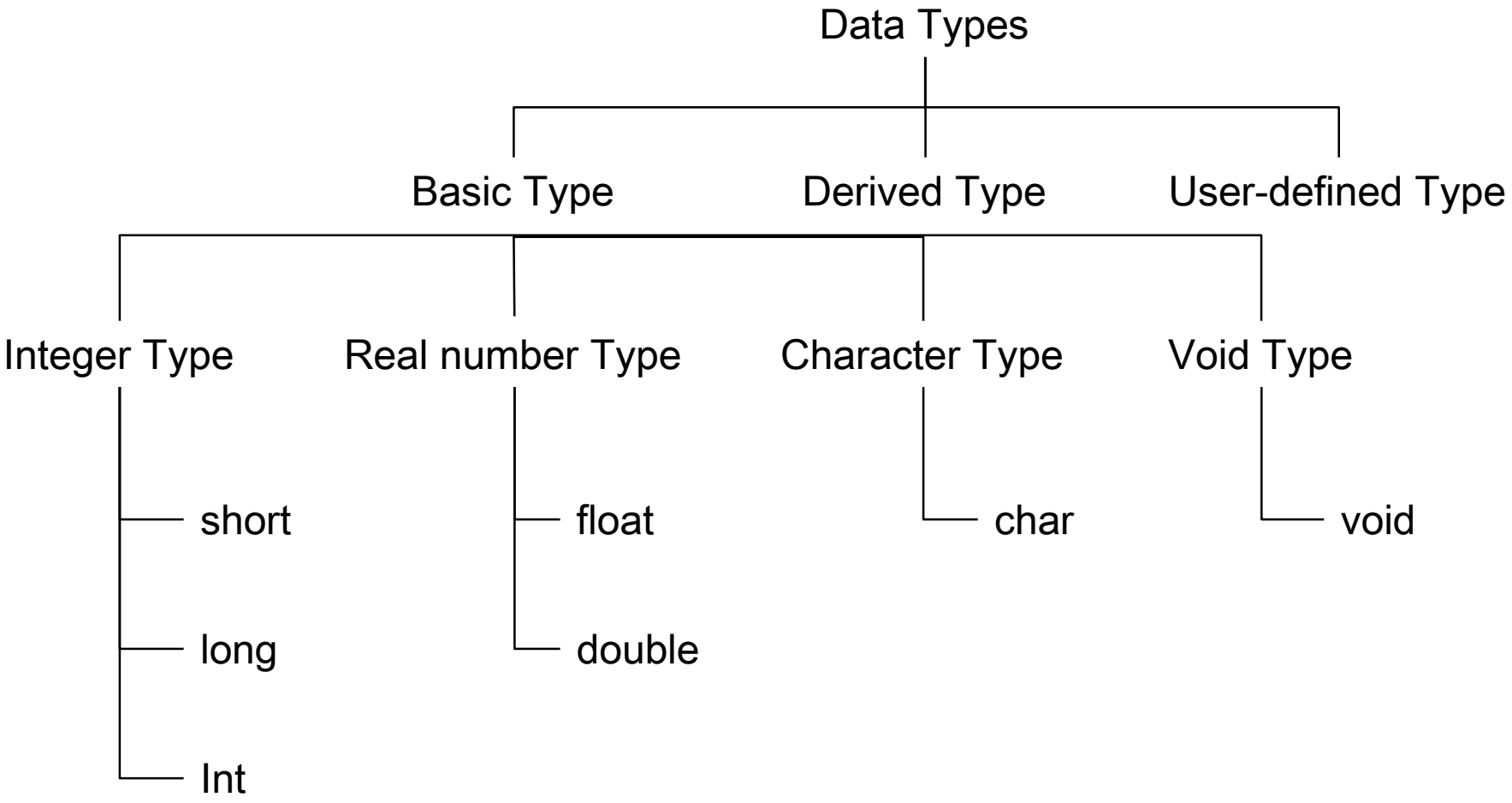
Variables

- Most programs need a way to store data temporarily during program execution

- The temporal storage locations are called **variables**

Data Types

- Every variable must have a (data) **type**



Variable Declaration

- Variables must be declared before they are used
- Declaration examples

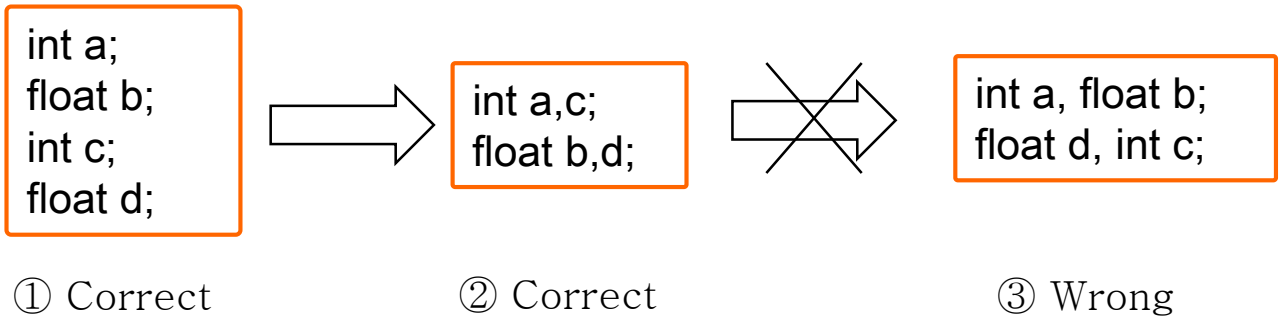
```
int height;          //Declare a variable height as type int
double weight;       //Declare a variable weight as type double
float radius;        //Declare a variable radius as type float
```

- Identifiers
  - Names for variables, functions, macros, and others
  - May contain letters, digits, underscores
  - Must begin with a letter or underscore
  - Using lower-case letters is common
  - e.g.) classroom, classroom10, _classroom

- Keywords
  - int, double, float are keywords
  - Cannot use as identifiers

Variable Declaration

- More examples



- Examples of incorrect declaration

```
int math*;          //Cannot use special character *
float my height;    //Cannot include a space inbetween
double 2016year;    //Cannot begin with a number
int for;            //Cannot use a keyword for
int int;            //Cannot use a datatype int
short year-2017;    //Cannot use special character –
```

Assignment

- A variable can be given a value using the assignment operator ‘=‘

```
int age;          //Declare a variable of type int
age = 20;         //A value of 20 is assigned to the variable age
age = 21;         //Now the variable age stores 21
```

[Example 1] Variable Declaration

```
01  #include<stdio.h>
02
03  int main()
04  {
05      int snum;
06      int credits;
07
08      snum = 20160120;
09      credits = 18;
10
11      printf("ID : %d₩n", snum);
12      printf("Credits : %d₩n", credits);
13      return 0;
14  }
```

Display

```
ID : 20160120
Credits : 18
```

Use a variable in stead of a constant!!

Variable Initialization

- If a variable has not been assigned a value, it may carry a garbage value, causing a program crash

- Variable name = equation, constant, variable
  → Initialize the variable with the designated value

- Possible to initialize a variable when it is declared

- Multiple variables can be declared and initialized simultaneously

```
void main()
{
    int num;
    printf("%d", num);
}
```

↓

ERROR!!

```
void main()
{
    int num=123;
    printf("%d", num);
}
```

↓

123

```
int a;
int b;
a = 123;
b = 456;
```

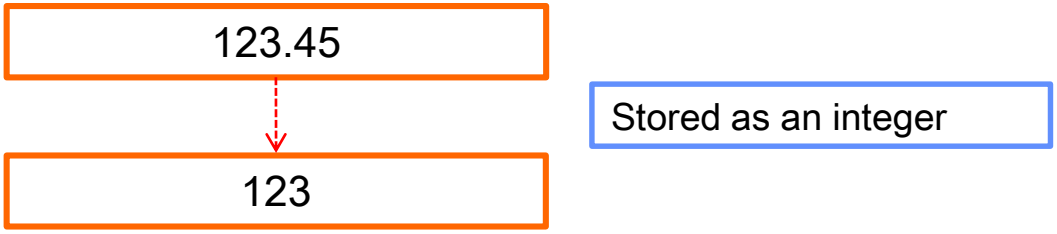
=

```
int a = 123, b = 456;
```

Variable Initialization

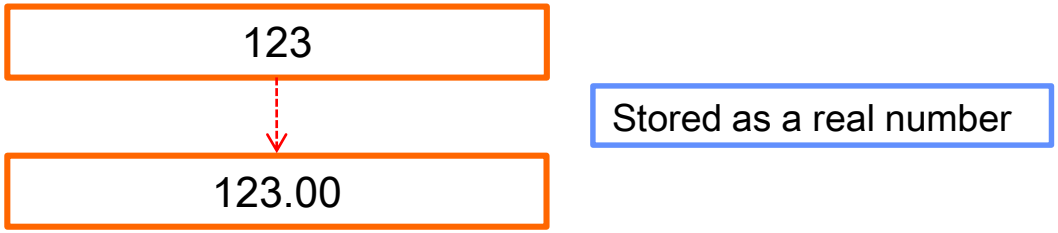
- Initialization with a different data type
  - A real number is assigned to a variable of int type

    ex) int a = 123.45;



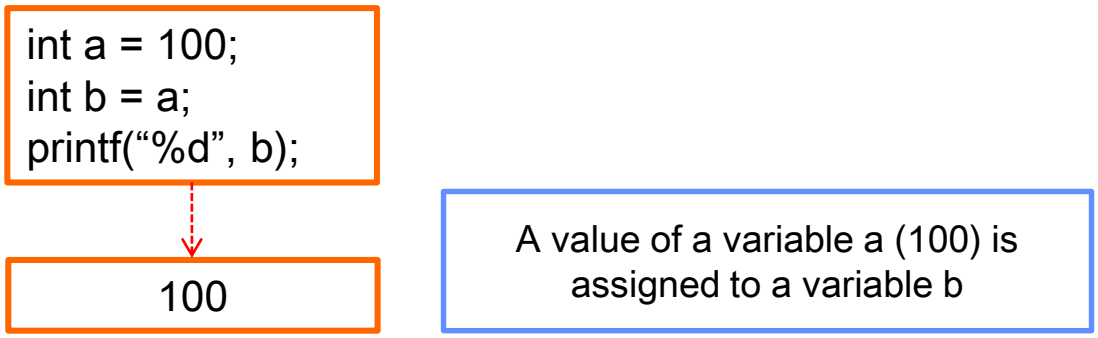
  - An integer value is assigned to a variable of float type

    ex) float a = 123;

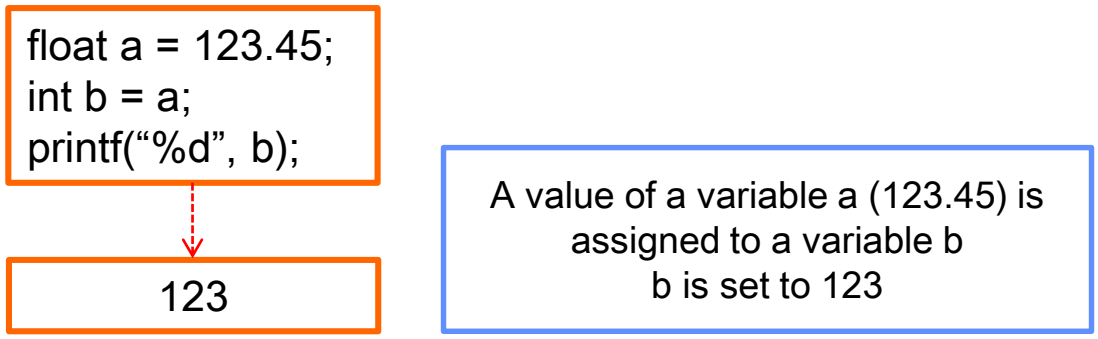


Variable Initialization

- Initialization with variables
  - Initialize a variable of int type with a variable of int type



  - Initialize a variable of int type with a variable of float type



[Quiz] Initialization with a different data type

- What will be the result of the following source code?

```
01  #include<stdio.h>
02
03  int main()
04  {
05      float fnum1 = 13.5;
06      float fnum2 = 12.5;
07      int inum1 = fnum1;
08      int inum2 = fnum2;
09
10      printf("fum1+fum2 = %f₩n", fnum1+fnum2);
11      printf("inum1+inum2 = %d₩n", inum1+inum2);
12
13      return 0;
14  }
```

Variable Initialization

- Initialize multiple variables with the same value
  - Initialize 4 variables with 100

```
int a,b,c,d;
a=b=c=d=100;
printf("%d %d %d %d", a, b, c, d);
```

```
100 100 100 100
```

```
a = b = c =d =100;
```

=

```
d = 100;
c = d;
b = c;
a = b;
```

[Example 2] Variable Initialization 1

```
01  #include<stdio.h>
02
03  int main()
04  {
05      int a, b, c, d;
06      a = 10;
07      b = a + 10;
08      c = a + b;
09      d = a + b + c;
10
11      printf("Value of a,b,c,d-> %d, %d, %d, %d₩n",a,b,c,d);
12
13      a = b = c = d;
14      printf("Value a,b,c,d-> %d, %d, %d, %d₩n",a,b,c,d);
15
16      return 0;
16  }
```

Display

```
Value of a,b,c,d->10, 20, 30, 60
Value of a,b,c,d->10, 10, 10, 10
```

[Example 3] Variable Initialization 2

```
01  #include<stdio.h>
02
03  int main()
04  {
05      int math = 99;
06      int korean = 90;
07      int science;
08      science = 94;
09
10       int total=math+korean+science;
11      //Store the total sum to the variable total
12      // using the addition operator +
13      printf("Math : %d₩n", math);
14      printf("Korean : %d₩n", korean);
15      printf("Science : %d₩n", science);
16      printf("Total : %d₩n", total);
17
18      return 0;
19  }
```

Display

```
Math : 99
Korean : 90
Science : 94
Total : 283
```

[Quiz] Data Type

1. Read 3 students' GPA as floating type
   e.g.) 3.6, 3.7, 3.8
2. Print the total sum as a real number and an integer.

Display1

```
Student1 = 3.6
Student2 = 3.7
Student3 = 3.8

Total(integer) => 11.1
Total(real number) => 11
```

1. Read the GPA as floating type and store it in a variable of int type
2. Print the total sum as an integer

Display2

```
Student1 = 3.6
Student2 = 3.7
Student3 = 3.8

Total(Real number) => ???
```

Data Representation

- Bit
  - The smallest unit of data in a compute
  - A single binary value, either 0(On) or 1(Off), like a electronic switch
    - ✓ The total number of possible combinations with n bits: 2^n

Binary	Decimal
00	0
01	1
10	2
11	3

Data Representation

- Binary number : possible number 0, 1
- Decimal number : possible number 0~9
- Hexadecimal number :
  possible number 0~9, A~F

- Representation
  - Binary number : 10_2
  - Decimal number : 10_{10}
  - Hexadecimal number : 10_{16}

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

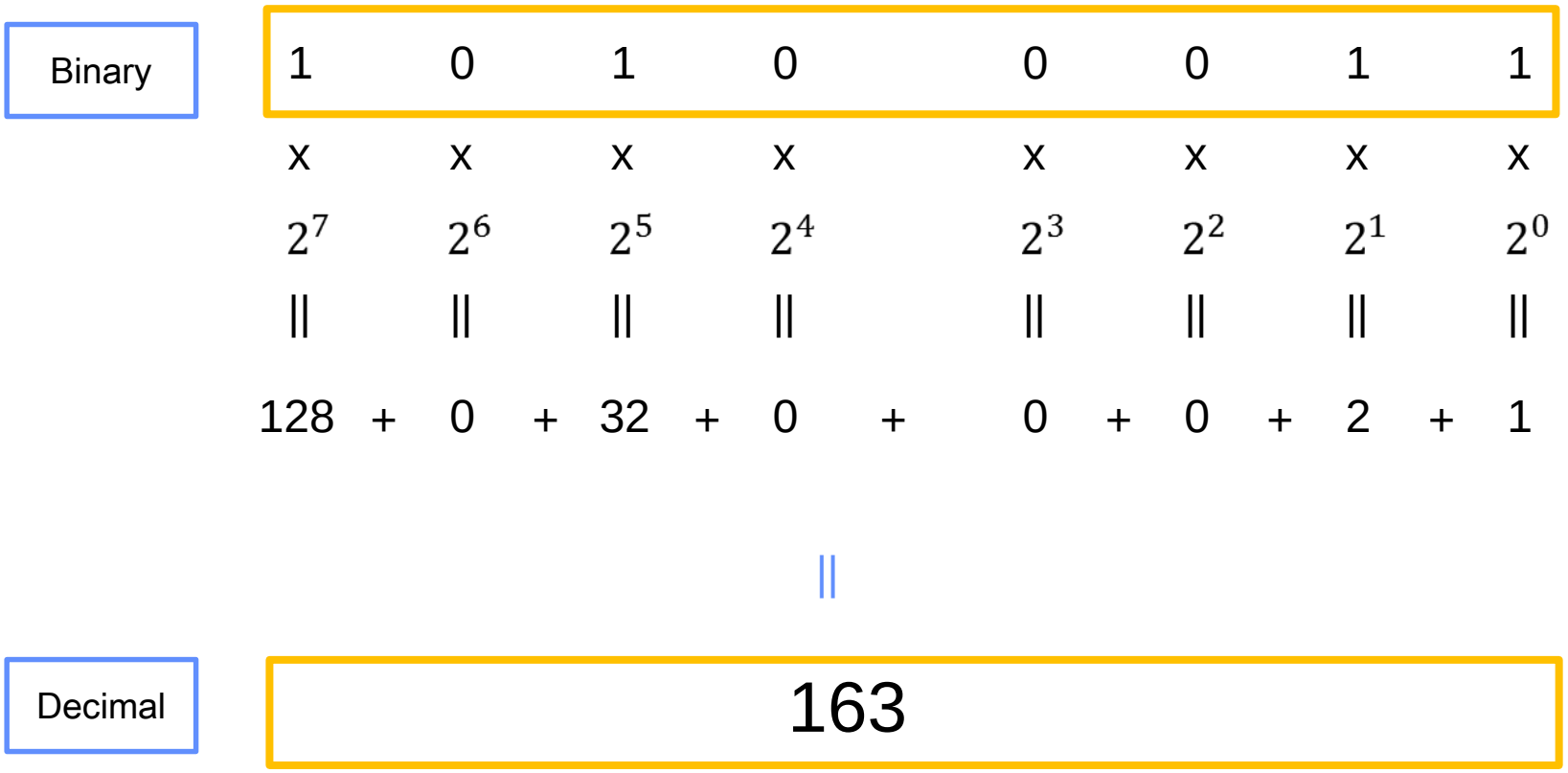
Data Representation

- Byte
  - Consists of 8 bits

# of bits	# of bytes	# of combinations	Binary	Decimal	Hexadecimal
1	-	$2^1 = 2$	0 ~1	0 ~ 1	0 ~ 1
2	-	$2^2 = 4$	0 ~11	0 ~ 3	0 ~ 3
4	-	$2^4 = 16$	0 ~1111	0 ~ 15	0 ~ F
8	1	$2^8 = 256$	0 ~11111111	0 ~ 255	0 ~ FF
16	2	$2^{16} = 65,536$	0 ~11111111 11111111	0 ~ 63,355	0 ~ FFFF
32	4	$2^{32} =$ ~4.2B	0 ~ …	0 ~ 4.2B	0 ~ FFFF FFFF
64	8	$2^{64} =$ …	0 ~ … …	0 ~ … …	0 ~ … …

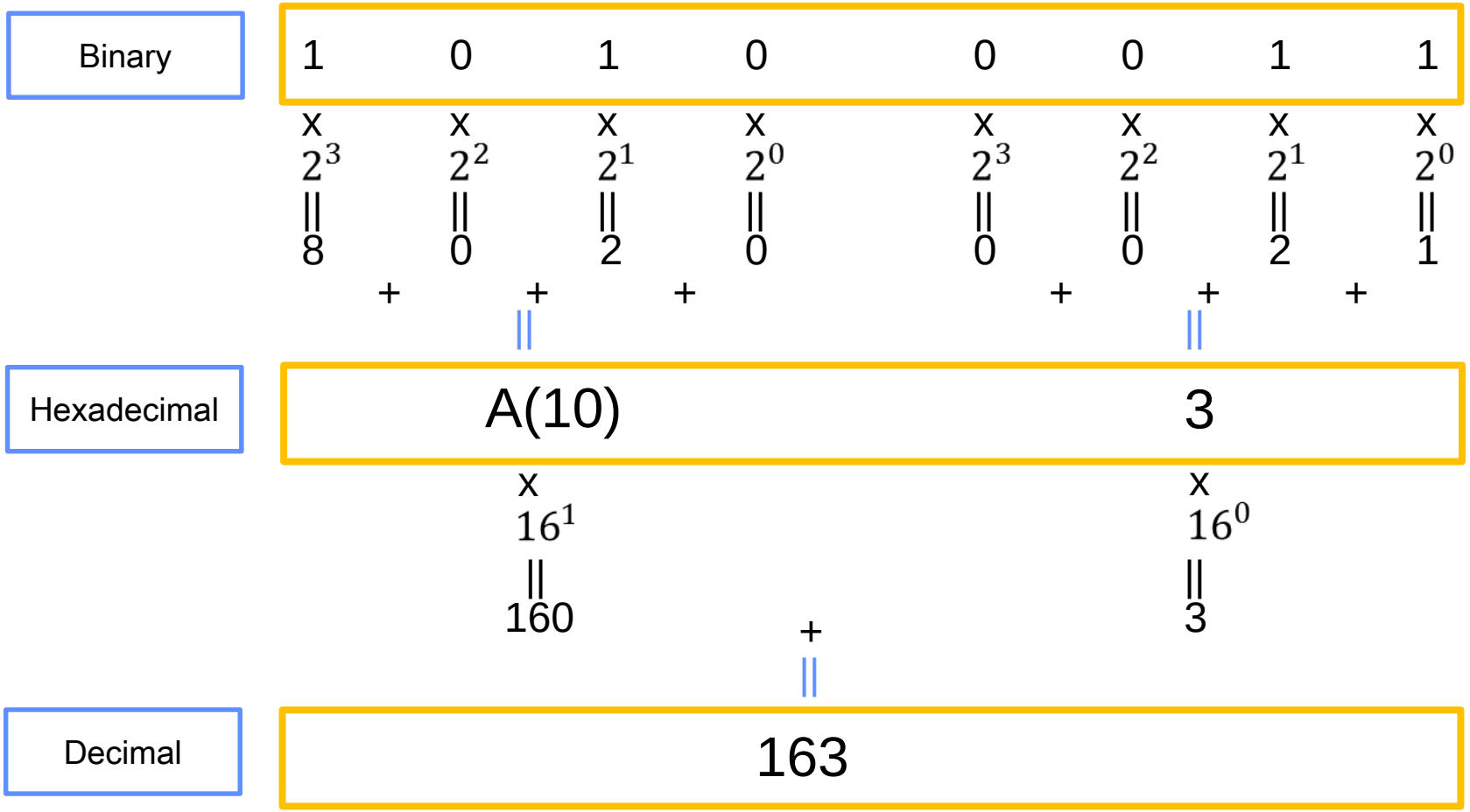
Data Representation

- Numeral system conversion
  - Binary –> Decimal



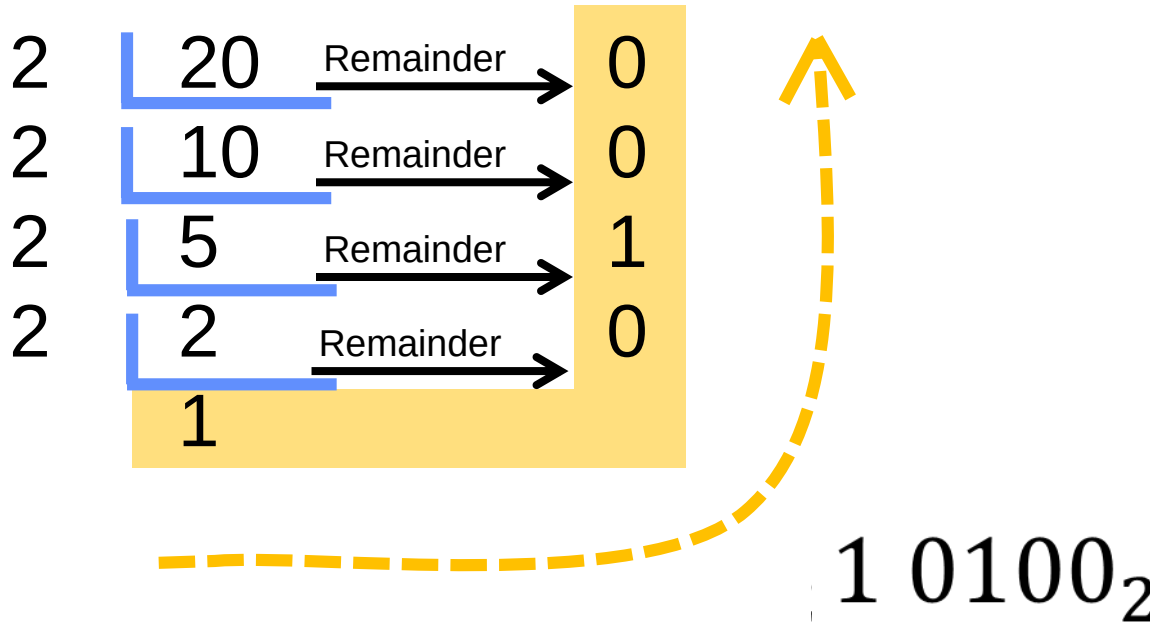
Data Representation

- Numeral system conversion
  - Binary –> Hexadecimal –> Decimal



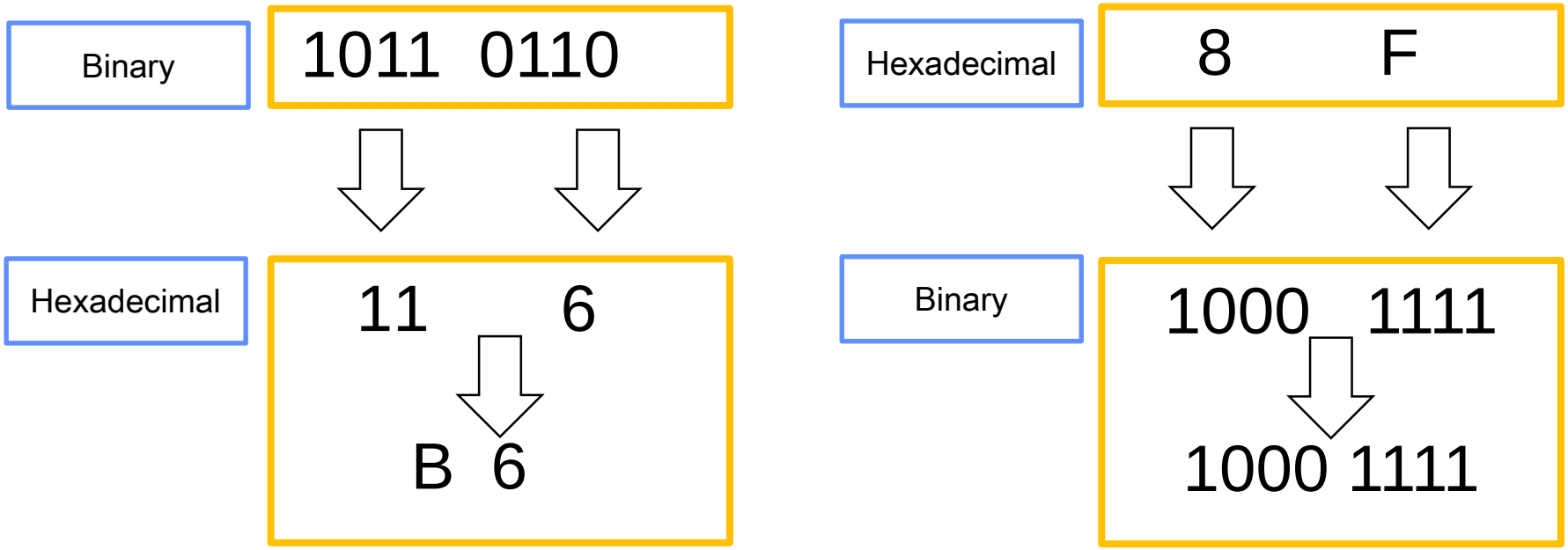
Data Representation

- Numeral system conversion
  - Decimal ->Binary
    - ✓ Keep dividing by 2 and record remainders



Data Representation

- Numeral system conversion
  - Hexadecimal ->Binary, Binary -> Hexadecimal
    - ✓ 1 hexadecimal digit corresponds to 4 binary digits
    - ✓ Convert 4 digits



[Quiz] Numeral system conversion

- Complete the following table

번호	Binary	Hexadecimal	Decimal
1	0101 1100		
2	10 1110 0100		
3		FA9	
4			983

Basic Data Type

- Integer data type
  - Basic keyword is int
    - ✓ Short: smaller than or equal to int, long: bigger than or equal to int
  - Integer type short, int, long can carry positive, 0, negative numbers
  - Keyword unsigned only handles 0 and positive numbers

Type	Keyword	Memory	Number Range
Integer	short	2 bytes	-32,768~32,767
	int	4 bytes	-2,147,483,648 ~ 2,147,438,647
	long	4 bytes	-2,147,483,648 ~2.147.483.647
Unsigned Integer	unsigned short	2 bytes	0~65,535
	unsigned int	4 bytes	0~4,294,967,295
	unsigned long	4 bytes	0~4,294,967,295

[Example 4] Integer Data Type

```
03  int main()
04  {
05      short sVar = 32000;                 //from -32767 to 32767
06      int iVar      = -2140000000;        //from 0 to ~2.1B
07
08      unsigned short usVar = 65000; //from 0 to 65535
09      unsigned int uiVar = 4280000000; //from 0 to ~4.2B
10
11      printf("Value : %d %d₩n", sVar, iVar);
12      printf("Value : %u %u₩n", usVar, uiVar);
13
14      return 0;
15  }
```

%u: designated for unsigned type

Display

```
Value : 32000 -2140000000
Value : 65000 4280000000
```

→ Practice 1: Print the exact maximum and minimum values of different data types
→ Practice 2: Print the minimum value −1 and maximum value +1

Basic Data Type

- Floating Point Data Type
  - Represent real numbers such as 3.14, 3.26567
  - Keyword: float, double
  - double type represents a finer and wider number range than float type
  - A constant 3.14 is considered as double type
    - ✓ float type constant: a constant with suffix f ex)3.14f

Type	Keyword	Memory	Number Range
floating point	float	4 bytes	~ -3.4×10^{38} to ~3.4×10^{38}
	double	8 bytes	~ -1.79×10^{38} to ~1.79×10^{38}

[Example 5] floating Point Data Type

```
01  #include<stdio.h>
02
03  int main()
04  {
05      float x = 0.12345678901234567890;
06      double y = 0.12345678901234567890;
07
08      printf("x=%.20f₩ny=%.20f₩n", x, y);
09
10      return 0;
11  }
```

Display

```
x=0.12345679104328156000
y=0.12345678901234568000
```

Basic Data Type

- Character Data Type
  - Keywords: char, unsigned char
  - Each type has a 1 byte

Type	Keyword	Memory	Range
Character	Char	1 byte	-128 ~ 127
	unsigned char	1 byte	0 ~ 255

→ You can regard character type as integer type of size 1 byte

Basic Data Type

- ASCII Table
  - A table showing special characters (or symbols) and their corresponding integer values

`char ch = ‘a’;` = `char ch = 97;`

  - ASCII Table contains 127 characters

Type	Decimal	Hexadecimal
0 ~ 9	48 ~ 57	0x30 ~ 0x39
A ~ Z	65 ~ 90	0x41 ~ 0x5A
a ~ z	97 ~ 122	0x61 ~ 0x7A

  - The print result may vary depending on the format

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

[Example 6] Character Data Type

```
01  #include<stdio.h>
02
03  int main()
04  {
05      char c1 = ‘a’;
06      char c2 = 65;
07
08      printf(“Character : %c %c", c1, c2);
09      printf(“Integer : %d %d", c1, c2);
10
11      return 0;
12  }
```

Display

```
Character : a A
Integer : 97 65
```

printf() Format

- Formats

Type	Example	Description
%d	10	Integer (decimal)
%x	100	Integer (hexademical)
%o	1234	Integer (octal)
%f or %lf	0.5	Real number
%c	‘a’ ‘A’	Character
%s	“Hi” “Hello”	Character string

printf() Discordance between Conversion Specification and Data Type

- The number conversion specifications > the number of output items: garbage value is shown

printf("%d+%d=%d",100,50);	⇨	100+50=19346782

- The number conversion specifications < the number of output items: ignored

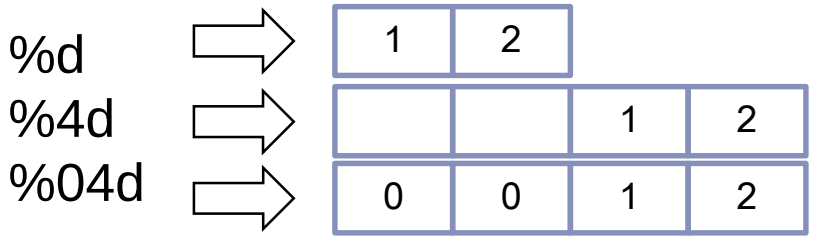
printf("%d+%d=",100,50,150);	⇨	100+50=

- The conversion specification does not match the data type of the output item: error

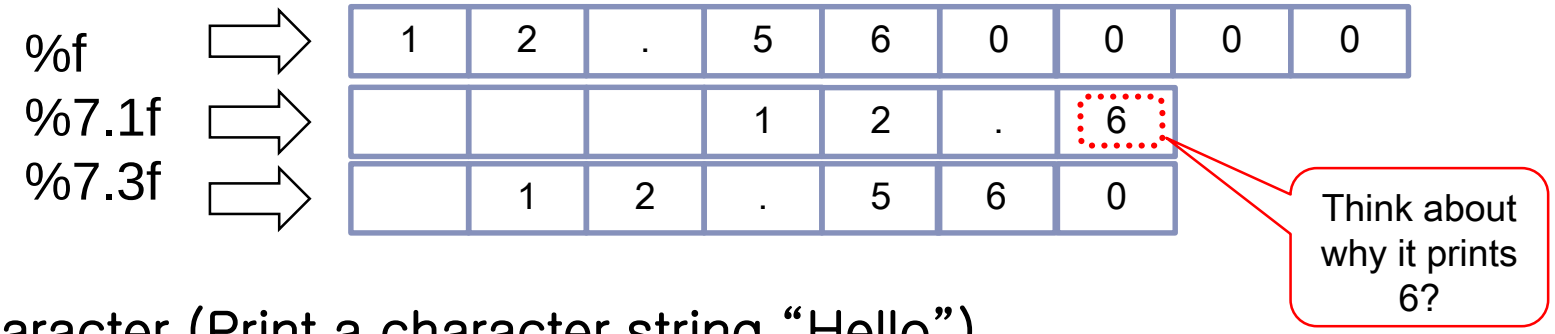
printf("%d/%d=%d",1,2,0.5);	⇨	???

printf() Format

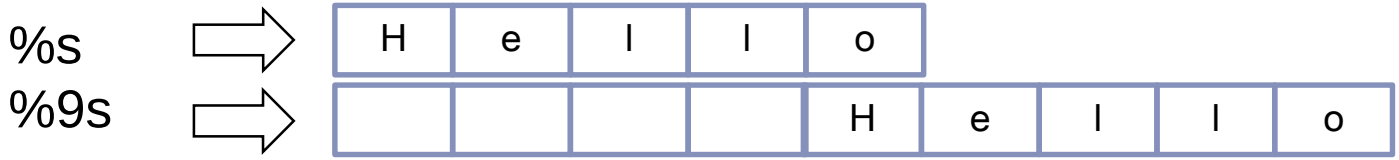
- Integer (Print an integer 12)



- Floating point (Print a real number 12.56)



- Character (Print a character string “Hello”)



[Example 7] printf() Format

```
01 #include <stdio.h>
02
03 void main()
04 {
05
06        printf("%d₩n", 125);
07        [Empty]
08        [Empty]
09
10        printf("%f₩n", 12.56);
11        [Empty]
12        [Empty]
13
14        printf("%s₩n",  " Sejong");
15        [Empty]
16 }
```

Display

1	2	5							
			1	2	5				
0	0	1	2	5					
1	2	.	5	6	0	0	0	0	
		1	2	.	6				
	1	2	.	5	6	0			
S	e	j	o	n	g				
			S	e	j	o	n	g	

printf() Format

- Formats

Type	Description
\n	New line
\t	Tab
\b	Backspace
\r	Carriage return (move to the start of the current line)
\a	Alert user
\\	Print \
\’	Print ‘
\”	Print “

[Example 8] Formats

```
01  #include<stdio.h>
02
03  int main()
04  {
05      printf("Beep sound.[①]₩n");
06      printf("Name : [②]Unknown[③][④] ID : [⑤]₩n",1600000);
07      printf("Target grade : [⑥]₩n",4.5);
08      printf("Emotion : Thrilled[⑦]₩n");
09
10      return 0;
11  }
```

Display

```
Beep sound.
Name : "Unknown"
ID : '1600000'
Target grade : 4.5
Emotion : Thrilled////
```

scanf()

- scanf()
  - Read an input from a user via Command window
  - Require a format string in between (“”) to specify the appearance of the input
  - Put & symbol in front of a variable
  - The same number and type of format specifications and variables

- Syntax

```
scanf(“format”,&variable);
```

```
int a=0;
float b=0;

scanf(“%d”,&a);
scanf(“%f”,&b);
```

Note) Do not use \n in scanf()

scanf()

- Can read multiple values from a user
- Conversion specifications match variables one by one

```
scanf("%d%d",&k1,&k2);
```

- Normally, use a space or a new line to specify the boundary between multiple values
- Delimiter may be used, but must use the same format

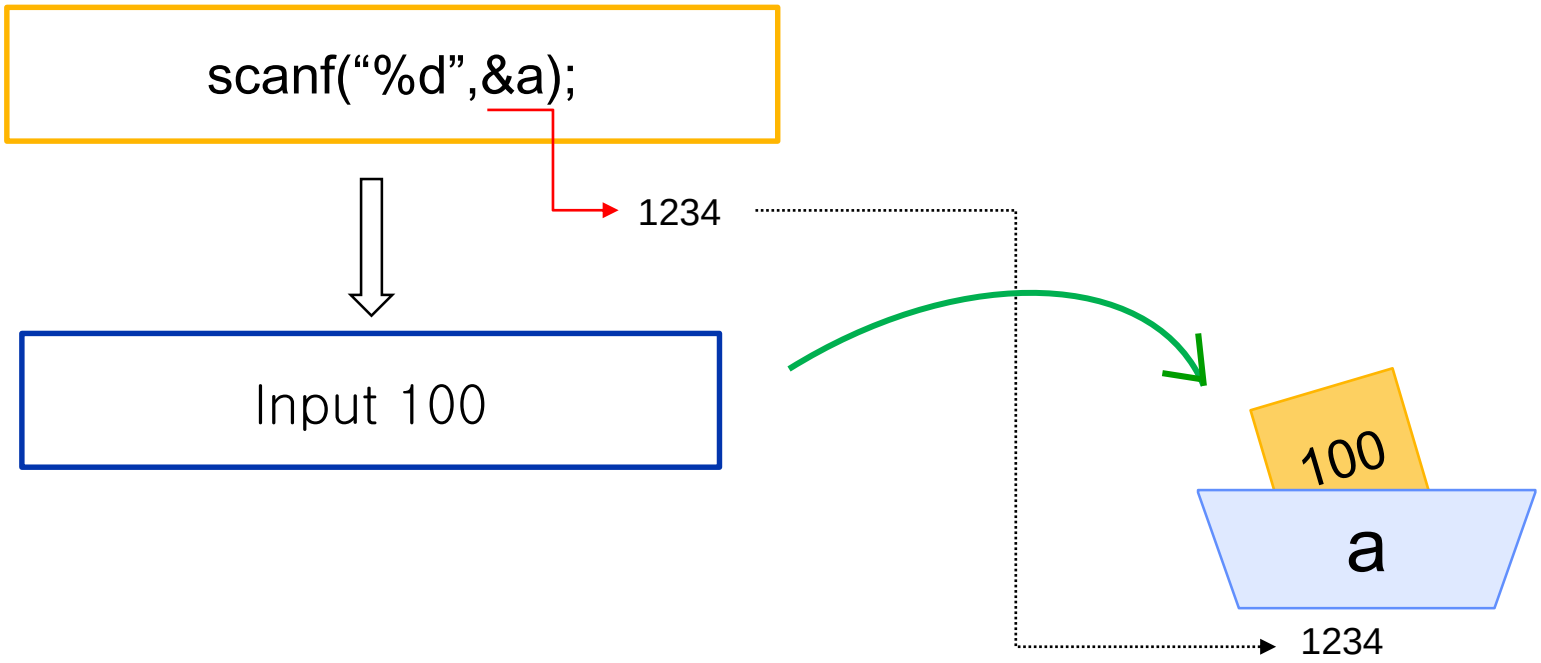
```
scanf("%d%d",&a,&b);
```
⇨ `100 50` or
```
100
50
```

```
scanf("%d,%d",&a,&b);
```
⇨ `100,50`

```
scanf("%d=%d",&a,&b);
```
⇨ `100=50`

scanf()

- & symbol, preceding a variable, is the address-of operator, indicating the address of the variable
- scanf() takes an address of a variable and assigns the received value



[Example 9] scanf()

```
01  #include<stdio.h>
02
03  int main()
04  {
05      int id;
06
07      printf(“Student ID : ");
08      scanf("%d",&id);
09      printf(“My student ID is %d.₩n",id);
10
11      return 0;
12  }
```

Display

```
Student ID : 16011111
My student ID is 16011111.
```

[예제 10] scanf()

```
01  #include<stdio.h>
02
03  int main()
04  {
05      int a,b,c;
06
07      printf(“Inupt : ");
08      scanf("%d-%d-%d",&a,&b,&c);
09      printf("%d-%d-%d₩n",c,b,a);
10
11      return 0;
12  }
```

Display

```
Input : 3-4-5
5-4-3
```