# Fundamentals of Computer System - Data and C

민기복

Ki-Bok Min, PhD

서울대학교 에너지자원공학과 조교수
Assistant Professor, Energy Resources Engineering

SEOUL NATIONAL UNIVERSITY

# Lecture 2 (Introducing C)
# Summary of last lecture

SEOUL NATIONAL UNIVERSITY

- Operator (연산자): =
- Functions (함수): main(), printf()
- Putting together a simple C program
- Creating variables, assigning them values, and displaying those values on screen
- The newline character (개행문자)

Actually this week

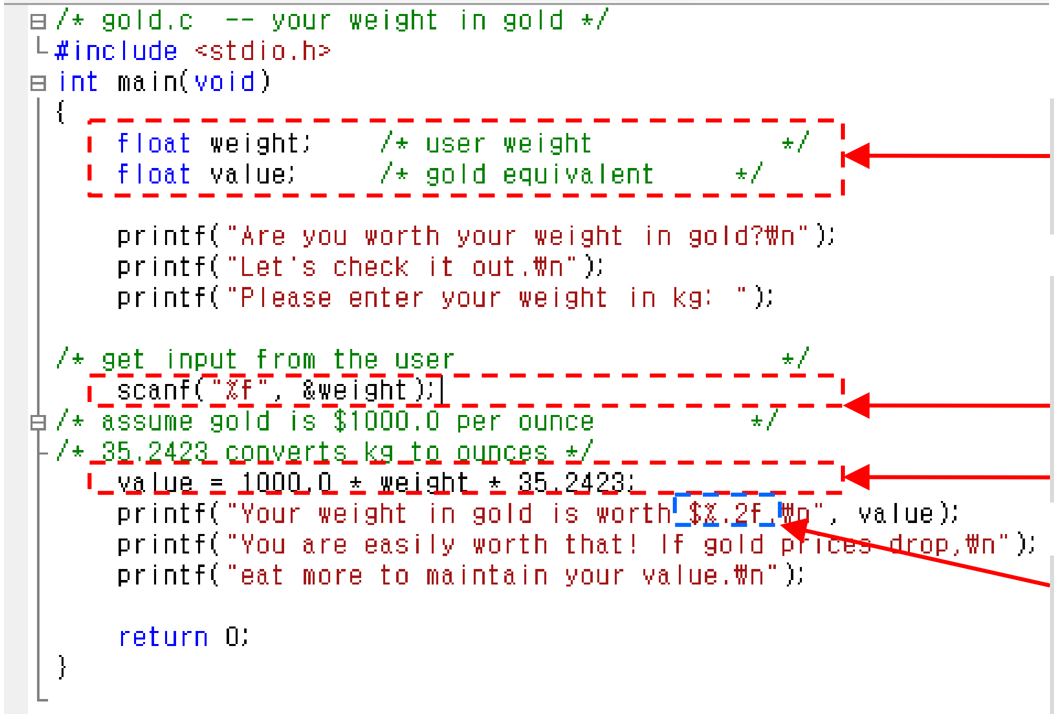
- Comments in your program, creating programs with more than one functions, and finding program errors
- Keywords – vocabulary of C

# Lecture 3 (Data and C)
## Content of this lecture

- Keywords:
  - int, short, long, unsigned, char, float, double, _Bool, _Complex, _Imaginary
- Basic data types: Integer
- integer types vs. floating-point types
- Writing constants & declaring variables of those types
- printf() & scanf()

# 1. A sample program (gold.c)
# an interactive program using scanf()

SEOUL NATIONAL UNIVERSITY



```
/* gold.c  -- your weight in gold */
#include <stdio.h>
int main(void)
{
   float weight;    /* user weight              */
   float value;     /* gold equivalent      */

   printf("Are you worth your weight in gold?₩n");
   printf("Let's check it out.₩n");
   printf("Please enter your weight in kg: ");

/* get input from the user                      */
   scanf("%f", &weight);
/* assume gold is $1000.0 per ounce            */
/* 35.2423 converts kg to ounces */
   value = 1000.0 * weight * 35.2423;
   printf("Your weight in gold is worth $%.2f ₩n", value);
   printf("You are easily worth that! If gold prices drop,₩n");
   printf("eat more to maintain your value.₩n");

   return 0;
}
```

Floating-point variable type (실수형)
– with decimal points

- Keyboard input
- %f  for floating-point number
- &weight to assign input  value to weight
- Why &? Has to do with address.

Floating-point specifier.
.2 specify two digits to the right of decimal
%f → 6 digits

- This program is interactive (대화식) – computer asks you for information and uses the number you enter.
- Compare this with "hello world" program – a significant improvement.

# Data Types

- Constants: keep their values unchanged
  - Ex) 35.2423 or 1000.0 in the previous example (gold.c)
  - Compiler can tell its type by the way it looks
- Variables: may change or be assigned values as the program runs
  - Ex) weight
  - Needs to be declared

# Data Types

## C data keywords

the way the data is stored

↓

integer type

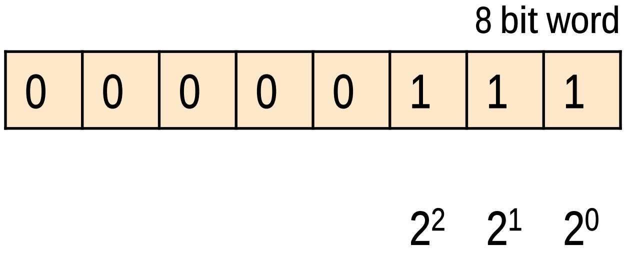
floating-point type

| Original K&R Keywords | C90 Keywords | C99 Keywords |
|---|---|---|
| int | signed | _Bool |
| long | void | _Complex |
| short | | _Imaginary |
| unsigned | | |
| char | | |
| float | | |
| double | | |

- You don’t need to be able to design an engine to drive a car.
- But good understanding of engine helps a great deal to be a good driver.

# Integer vs. Floating-Point Types

- Integer
  - Number with no fractional part
  - 7

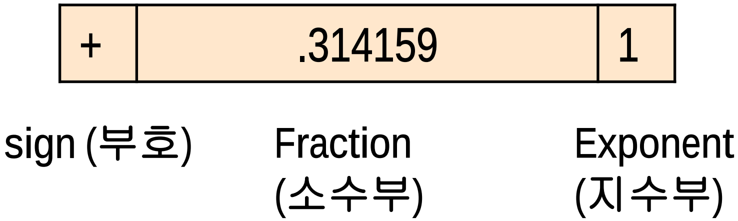


8 bit word

| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

$2^2$ $2^1$ $2^0$

- Floating-point
  - Number with fractional part
  - 7.00 3.16E7 2e-8
  - 3.14159E1



| + | .314159 | 1 |
|---|---|---|
| sign (부호) | Fraction (소수부) | Exponent (지수부) |

  - Computer floating-point numbers often approximate a true value
  - Slower than integer

# Basic C Data Types
## int Type

- 1 word – can be 16 bit (-32768 ~ 32767) or 32 bit ($-2^{31}$~$2^{31}-1$)
- Declaration (create storage)

  ex) int cows;

  Int hogs, cows, goats;

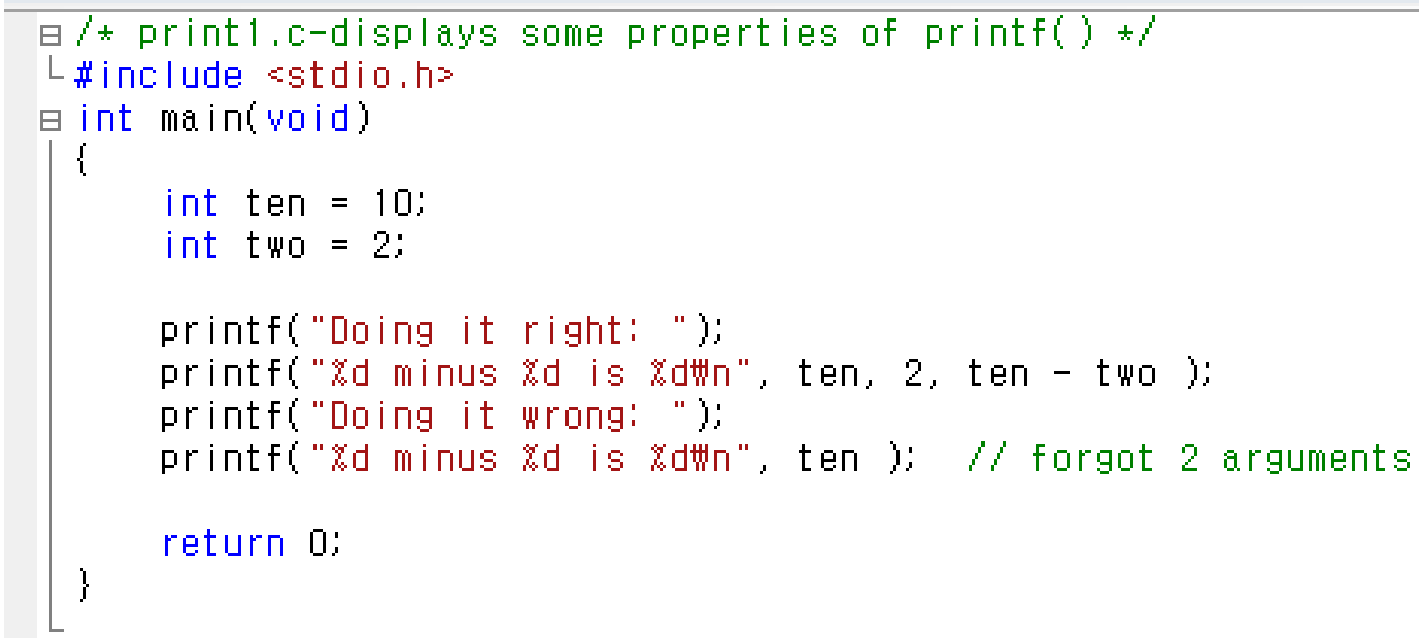
- Getting values (give it value)
  - 1) cows = 112;
  - 2) from functions ex) scanf()
  - 3) initialize ex) int cows = 112;

    Int hogs = 32, cows = 112;

Integer constant

# Basic C Data Types
## printing **int** values (decimal integer)

- Use printf() function to print **int** types
- For integer decima (십진수) use %d (a *format specifier*)
- Number of format specifier must matches the number of values to be displaced



```
/* print1.c-displays some properties of printf() */
#include <stdio.h>
int main(void)
{
    int ten = 10;
    int two = 2;

    printf("Doing it right: ");
    printf("%d minus %d is %d₩n", ten, 2, ten - two );
    printf("Doing it wrong: ");
    printf("%d minus %d is %d₩n", ten );  // forgot 2 arguments

    return 0;
}
```

# Basic C Data Types

## Octal (8진수) and hexadecimal (16진수)

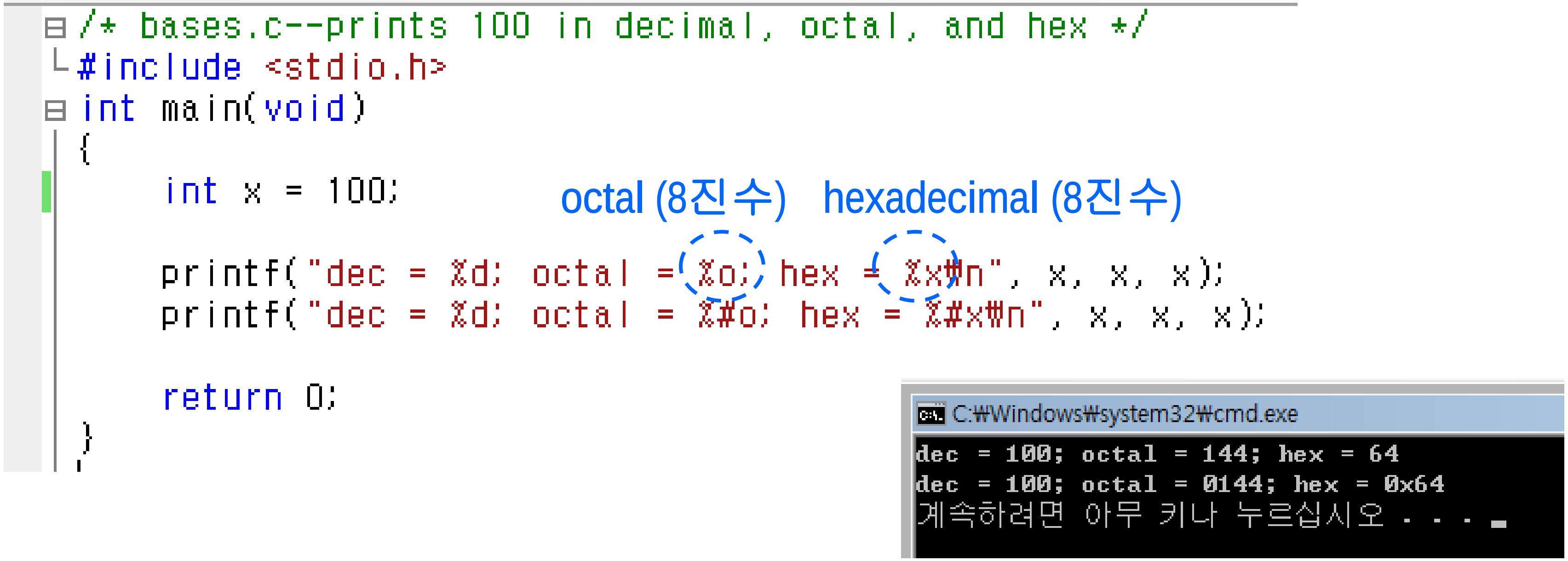
- Normally, C assumes that integer constants are decimal (십진수)
- However, octal (8진수) or hexadecimal (16진수) can be convenient in some situation
- How do we distinguish among decimal, octal and hexadecimal?
- Use prefix 0 for octal and 0x (or 0X) for hexadecimal

| | 10진수 | 8진수 | 16진수 |
|---|---|---|---|
| math | 16 | 20 | 10 |
| C language | 16 | 020 | 0x10 |

Same for C which uses binary code anyway

# Basic Data Type

## printing **int** values (octal and hexadecimal)



```
/* bases.c--prints 100 in decimal, octal, and hex */
#include <stdio.h>
int main(void)
{
    int x = 100;

    printf("dec = %d; octal = %o; hex = %x₩n", x, x, x);
    printf("dec = %d; octal = %#o; hex = %#x₩n", x, x, x);

    return 0;
}
```

octal (8진수) hexadecimal (8진수)

C:₩Windows₩system32₩cmd.exe

```
dec = 100; octal = 144; hex = 64
dec = 100; octal = 0144; hex = 0x64
계속하려면 아무 키나 누르십시오 . . .
```

- To display integer in octal and hexadecimal, use;
  - %o and %x
- To show c prefixes(such as 020 or 0x10), use
  - %#o and %#x

# Basic Data Type
## Other Integer Types

- **int** may be enough. But there are also other forms, too.

- **short int** (or **short**) : short may use less storage than **int**
- **long** int (or **long**) : may use more storage than **int**
- **long long int** (or **long long**) : may use more storage than **long int**
- **unsigned int** (or **unsigned**) : only nonnegative (more space)

- Why do we need these??? Optimal use of storage

# Basic Data Type
## Space of Types

- In a typical Windows 32 bit machine

| | | | |
|---|---|---|---|
| – short | 16 bit | $-2^{15} \sim 2^{15}-1$ | -32,768 ~ 32,767 |
| – unsigned short | 16 bit | $0 \sim 2^{16}-1$ | 0 ~ 65,535 |
| – Int | *32 bit | $-2^{31} \sim 2^{31}-1$ | -2,147,483,648 ~2,147,483,647 |
| – unsigned int | 32 bit | $0 \sim 2^{32}-1$ | 0 ~ 4,294,967,295 |
| – long | 32 bit | $-2^{31} \sim 2^{31}-1$ | -2,147,483,648 ~2,147,483,647 |
| – unsigned long | 32 bit | $0 \sim 2^{32}-1$ | 0 ~ 4,294,967,295 |
| – long long | 64 bit | $-2^{63} \sim 2^{63}-1$ | … |
| – unsigned long long | 64 bit | $0 \sim 2^{64}-1$ | … |

*: may be 16 bit for Windows 3.1 IBM PC.

- You have 32 bit system, if you need 16 bit, would you use **short** or **int**?
  - Answer: **short** ← saves memory

# Basic Data Type
## Maximum/Minimum range

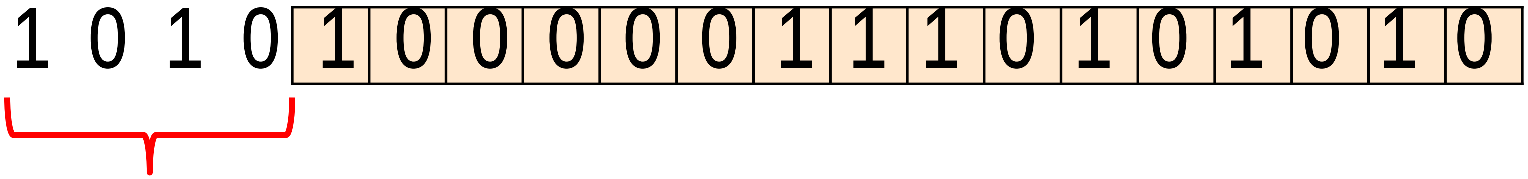
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/1 | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |

+/-

- $a(r^n-1)/(r-1) = 1(2^{15}-1)/(2-1)=2^{15}-1$ → maximum
- Minimum = - maximum -1 (because negative integer is stored as complement (보수)). Don’t worry about this.

# Basic Data Type
## Overflow

- If you have 16 bit storage and use 20 bit number.

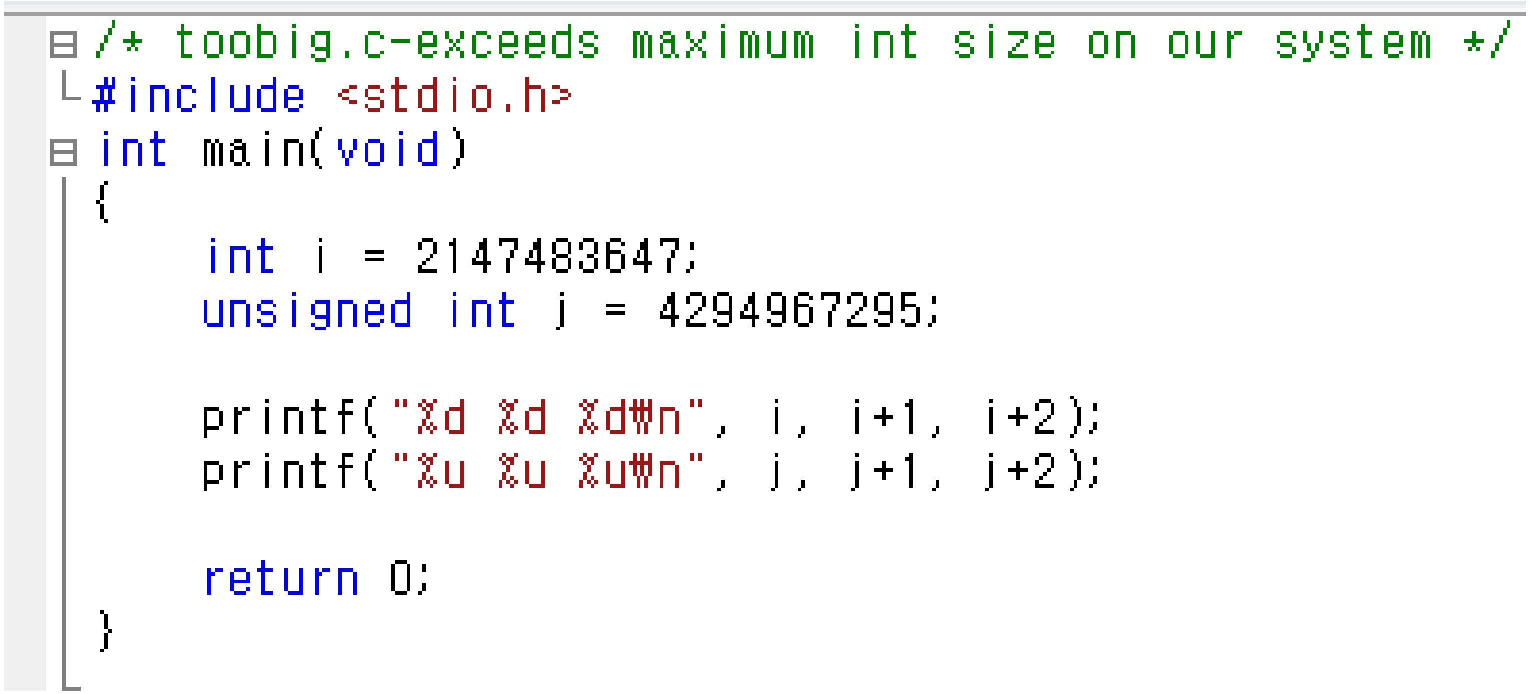


1 0 1 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

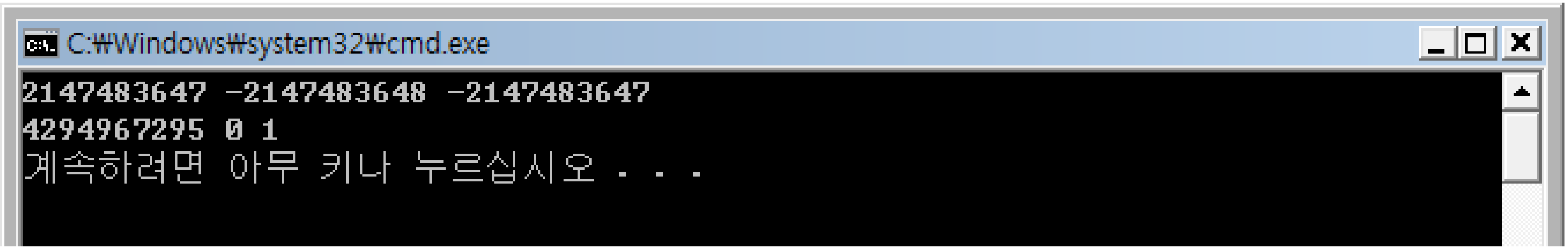
Overflow!!

- Importantly, compiler does not inform you of this.

# Basic Data Type
## Overflow – an example



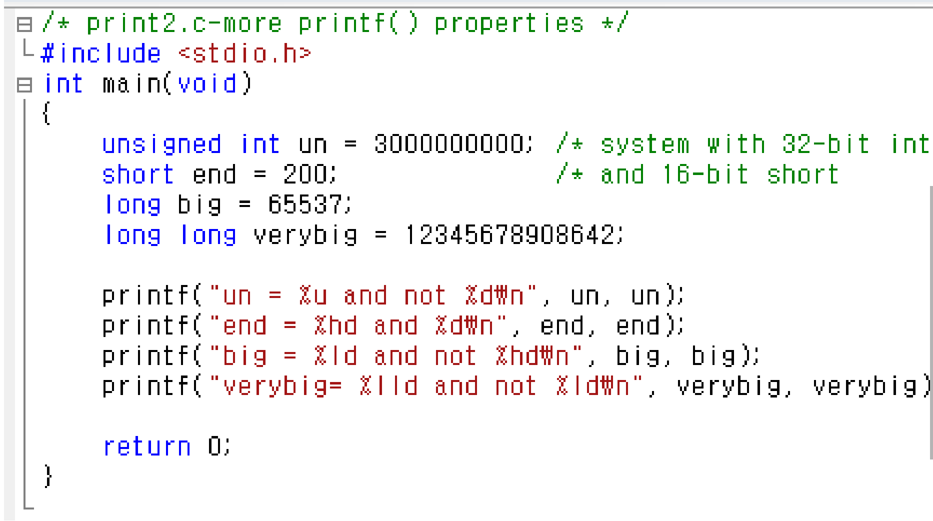
```c
/* toobig.c-exceeds maximum int size on our system */
#include <stdio.h>
int main(void)
{
        int i = 2147483647;
        unsigned int j = 4294967295;

        printf("%d %d %d₩n", i, i+1, i+2);
        printf("%u %u %u₩n", j, j+1, j+2);

        return 0;
}
```



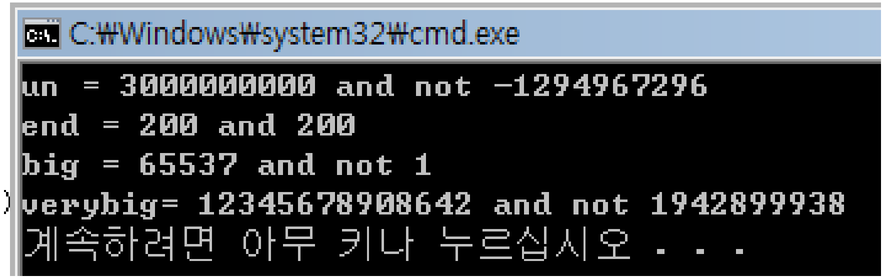
```
2147483647 -2147483648 -2147483647
4294967295 0 1
계속하려면 아무 키나 누르십시오 . . .
```

# Basic Data Type

## Printing **short**, **long**, **long long** and **unsigned** Types



```
/* print2.c-more printf() properties */
#include <stdio.h>
int main(void)
{
    unsigned int un = 3000000000; /* system with 32-bit int */
    short end = 200;              /* and 16-bit short       */
    long big = 65537;
    long long verybig = 12345678908642;

    printf("un = %u and not %d₩n", un, un);
    printf("end = %hd and %d₩n", end, end);
    printf("big = %ld and not %hd₩n", big, big);
    printf("verybig= %lld and not %ld₩n", verybig, verybig)

    return 0;
}
```

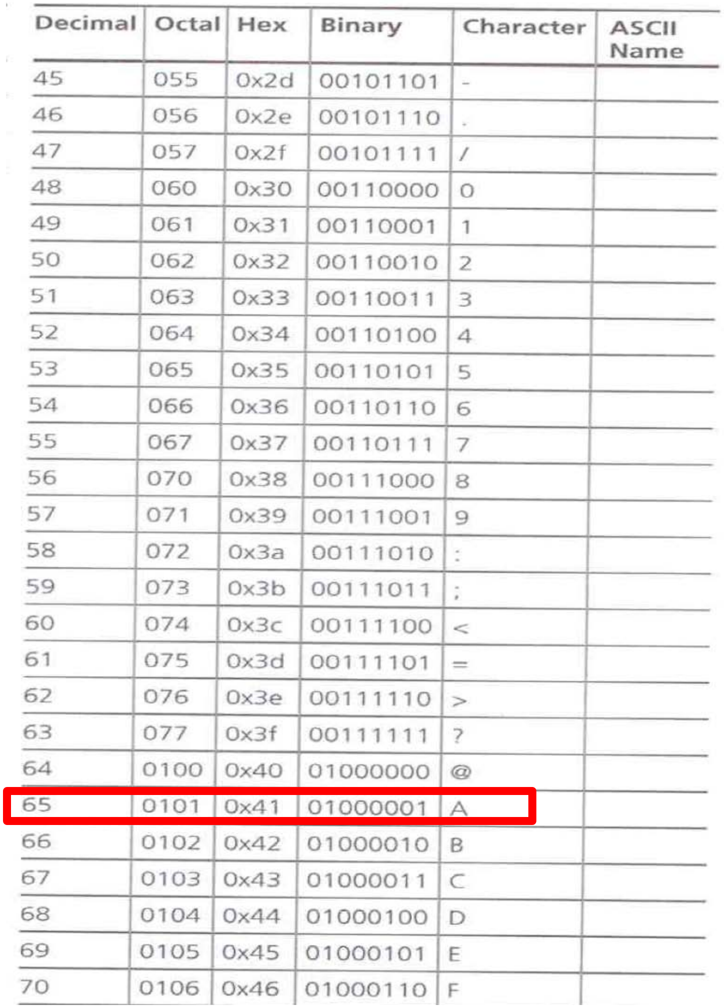


```
un = 3000000000 and not -1294967296
end = 200 and 200
big = 65537 and not 1
verybig= 12345678908642 and not 1942899938
계속하려면 아무 키나 누르십시오 . . .
```

| | | | |
|---|---|---|---|
| %d | integer decimal | %hd | short decimal |
| %u | unsigned int | %ld | long decimal |
| %lld | long long decimal | | |
| %lx | long hexadecimal (16진수) | %lo | long octal (8진수) |

# Basic Data Type
## Character type

- Char type: characters (letters & punctuation)
  - But technically it is integer
  - Because it actually stores integers
  - Computer uses a numerical code (certain integer represent certain characters). E.g., ASCII (American Standard Code for Information Interchange) code
  - 65 represents upper case A. To store A, you store 65.



| Decimal | Octal | Hex | Binary | Character | ASCII Name |
|---|---|---|---|---|---|
| 45 | 055 | 0x2d | 00101101 | - | |
| 46 | 056 | 0x2e | 00101110 | . | |
| 47 | 057 | 0x2f | 00101111 | / | |
| 48 | 060 | 0x30 | 00110000 | 0 | |
| 49 | 061 | 0x31 | 00110001 | 1 | |
| 50 | 062 | 0x32 | 00110010 | 2 | |
| 51 | 063 | 0x33 | 00110011 | 3 | |
| 52 | 064 | 0x34 | 00110100 | 4 | |
| 53 | 065 | 0x35 | 00110101 | 5 | |
| 54 | 066 | 0x36 | 00110110 | 6 | |
| 55 | 067 | 0x37 | 00110111 | 7 | |
| 56 | 070 | 0x38 | 00111000 | 8 | |
| 57 | 071 | 0x39 | 00111001 | 9 | |
| 58 | 072 | 0x3a | 00111010 | : | |
| 59 | 073 | 0x3b | 00111011 | ; | |
| 60 | 074 | 0x3c | 00111100 | < | |
| 61 | 075 | 0x3d | 00111101 | = | |
| 62 | 076 | 0x3e | 00111110 | > | |
| 63 | 077 | 0x3f | 00111111 | ? | |
| 64 | 0100 | 0x40 | 01000000 | @ | |
| 65 | 0101 | 0x41 | 01000001 | A | |
| 66 | 0102 | 0x42 | 01000010 | B | |
| 67 | 0103 | 0x43 | 01000011 | C | |
| 68 | 0104 | 0x44 | 01000100 | D | |
| 69 | 0105 | 0x45 | 01000101 | E | |
| 70 | 0106 | 0x46 | 01000110 | F | |

# Basic Data Type
## Character type (ASCII code)

- Standard ASCII code: 0 ~ 127 < 8 bit
- Commercial Unicode Initiative: 16 bit or 32 bit

# Basic Data Type (Char)
## Declaration

- Create three variables: response, itable, latan

```
char response;

char itable, latan;
```

- Do we need to memorize ASCII code to assign character?
  No!! → use ‘ ’

```
char grade = ‘A’
```

Character constant (문자상수)

```
grade = ‘A’;        /* OK*/

grade = A;          /* No! compiler think of A as a name of variables */

grade = “A”;        /* No! compiler think of A as a string */

grade = 65;         /* OK for ascii but poor style */
```

Will be covered later

# Basic Data Type (Char)
# Nonprinting characters

SEOUL NATIONAL UNIVERSITY

- Nonprinting characters: backspace, enter, speaker beep…
  - 1) use ASCII code,
- ex) char beep = 7 ; // beeping
- 2) use escape sequence
- Ex) char nerf = ‘\n’;

# Basic Data Type (Char)
## Nonprinting characters

SEOUL NATIONAL UNIVERSITY

- Nonprinting characters: backspace, enter, speaker beep

1) Use ASCII code

ex) char beep = 10; // new line

2) Use escape sequence

ex) char nerf = ‘\n’;

| sequence | Meaning |
|---|---|
| \b | Back space |
| \n | New line |
| \t | Horizontal tab |
| \\ | Backslash (\) |
| \’ | Single quote |
| \” | Double quote |
| \? | Question mark |

If you want to print this

He said, “a \ is a backslash.”

printf(“He said, \”a \\ is a backslash.\”\n”);

Use this

# Basic Data Type (Char)

## Number (수) & Number character (숫자상수)

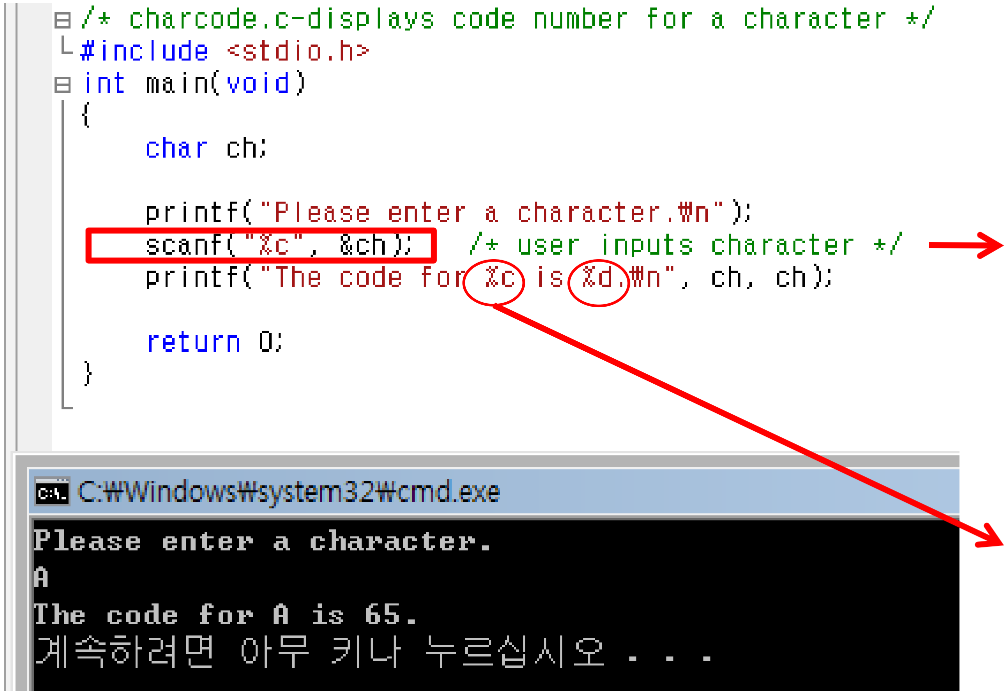
- Number vs. Number character
- Ex) character 4 is represented by ascii 52 or ‘4’

  He said, “a \ is a backslash.”

  printf(“He said, \”a \\ is a backslash.\”\n”);
- Within “ “, don’t use ‘ ’
- ‘/f’ versus ‘\014’
  - Use ‘/f’
- 032 versus ‘\032’
  - ‘\032’ is clear to represent a character code

# Basic Data Type (Char)
## Printing Characters

- Use %c
- As char is stored as 1 byte integer, if you print the value of a char variable with usual %d, you get an integer



```
/* charcode.c-displays code number for a character */
#include <stdio.h>
int main(void)
{
    char ch;

    printf("Please enter a character.₩n");
    scanf("%c", &ch);   /* user inputs character */
    printf("The code for %c is %d.₩n", ch, ch);

    return 0;
}
```

C:₩Windows₩system32₩cmd.exe

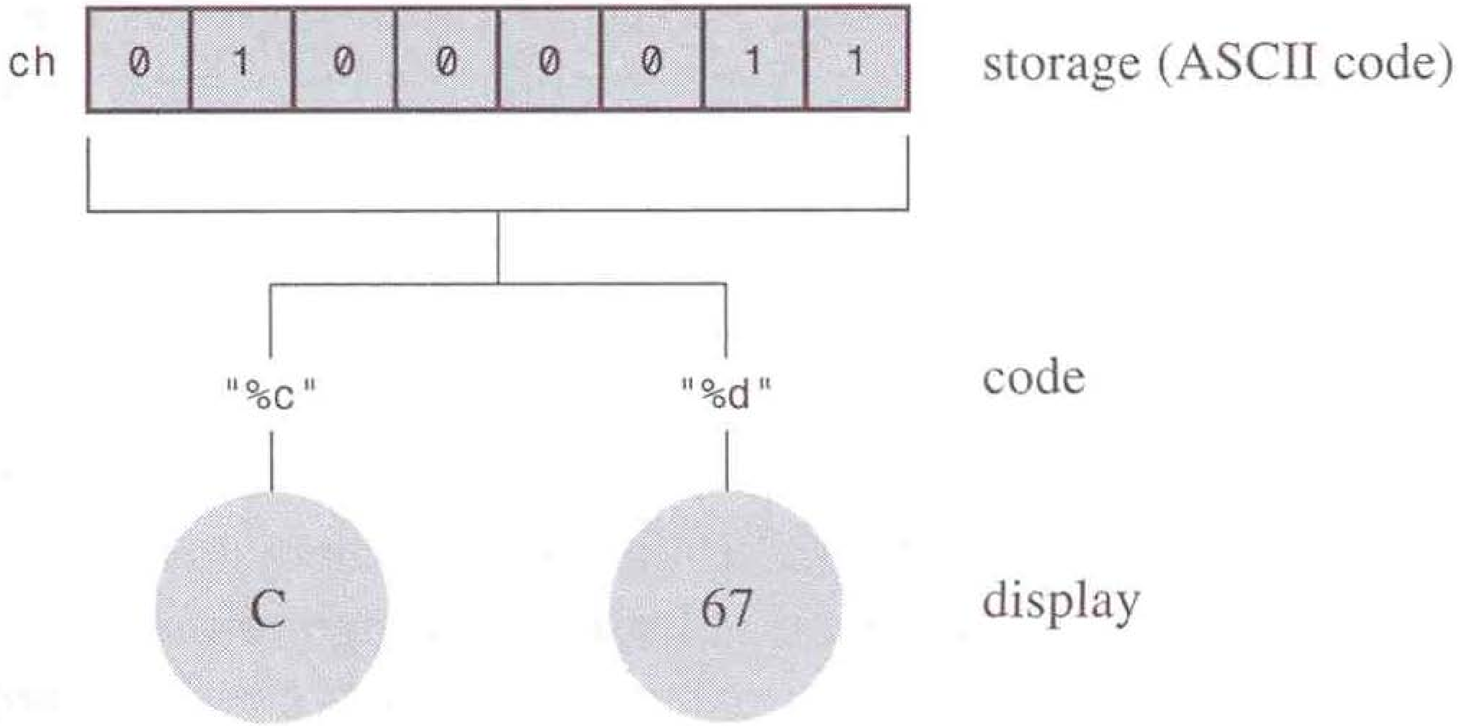
```
Please enter a character.
A
The code for A is 65.
계속하려면 아무 키나 누르십시오 . . .
```

scanf() function fetches the character you typed, & causes the character to be assigned to the variable ch

printf() has to do with display (not how it is stored)

# Basic Data Type (Char)
Printing Characters

- Data Display versus Data Printing



# Summary