Introduction to Software Design

C02. Data Representation

Yoonsang Lee
Spring 2020

Topics Covered

- 진법: 2진수, 10진수, 16진수
- 비트(bit)와 바이트(byte)
- 컴퓨터에서
  - 정수를 표현하는 방법
  - 실수를 표현하는 방법
  - 문자를 표현하는 방법
- 자료형 (data type)
- 자료형의 변환 (type casting)
- More about printf(), scanf()

10진수

0
1
2
3
4
5
6
7
8
9
10
11
12
...

10진수: 열 개의 기호를 이용해서 값(데이터)을 표현하는 방식

자릿수 증가 →

왜? 사용할 수 있는 기호가 10개 밖에 없기 때문에 (0~9)

2진수

10진수	2진수
0	0
1	1
2	→ 10
3	11
4	→ 100
5	101
6	110
7	111
8	→ 1000
	자릿수 증가
…	…

2진수: 두 개의 기호를 이용해서 값(데이터)을 표현하는 방식

16진수

10진수	16진수
0	0
1	1
…	…
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	→ 10
17	11
…	…

자릿수 증가

16진수: 16개의 기호를 이용해서 값(데이터)을 표현하는 방식

데이터의 표현 방식

- 진법을 왜 알아야 할까?

- 컴퓨터는 0과 1로 모든 데이터를 표현하기 때문

- 2, 3, 4, … 100, 200 등을 표현하려면?

-> 0과 1로 정수를 표현하는 방법이 바로 2진수

- 2.3333…, 3.141592… 등은 어떻게 표현할까?

-> 0과 1로 실수를 표현하는 별도의 방법이 있다

데이터의 표현 방식

- 그렇다면 알파벳 a, b, c,… 등은 어떻게 표현하는가?

-> 각각의 알파벳 문자를 특정 숫자와 대응시켜 표현 : ASCII 코드

- 그렇다면 16진수는 어디에 쓰일까?

-> 2진수로만 데이터를 표현하면 숫자의 길이가 길어져서 표현하기도, 파악하기도 어렵다.

16진수 하나로 네 자리 2진수를 한꺼번에 표현할 수 있기 때문에 간단하게 표현될 수 있다.

예) 16진수 F는 2진수 1111

16진수 E는 2진수 1110

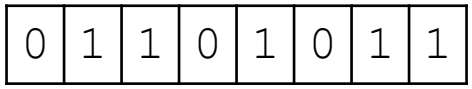
…

비트(Bit)와 바이트(Byte)

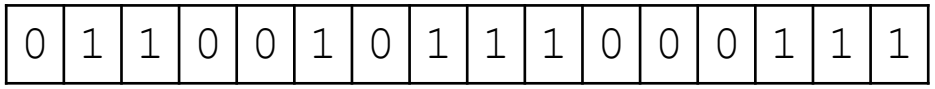
- 비트(Bit): 컴퓨터가 표현하는 데이터의 최소 단위. 0 혹은 1 두 개의 값만 표현할 수 있다.

1비트

- 바이트(Byte): 비트를 **8**개 묶은 것.



1바이트



2바이트

비트(Bit)와 바이트(Byte)

- 1비트로 표현할 수 있는 데이터의 개수는?
  - 2개: 0, 1
- 2비트?
  - 4개: 00, 01, 10, 11
- 4비트? → 경우의 수: 2 x 2 x 2 x 2

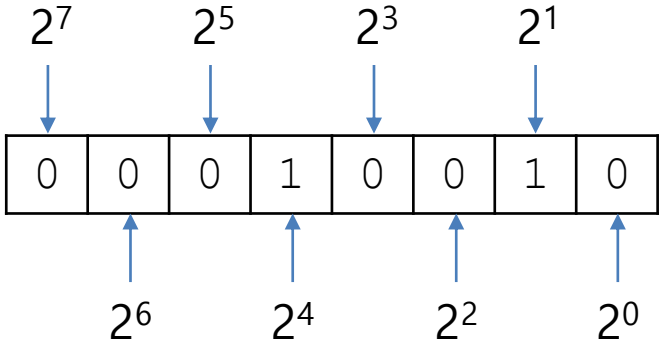
0	1	1	0

  - 2^4 = 16개
- 1바이트?
  - 2^8 = 256개
- 2바이트?
  - 2^{16} = 65,536개

비트(Bit)와 바이트(Byte)

- 다음의 1바이트로 표현된 2진수들은 각각 10진수로 얼마인가?

2진수	10진수	
00000001	= 1	= 2^0
00000010	= 2	= 2^1
00000100	= 4	= 2^2
00001000	= 8	= 2^3
00010000	...	= 2^4
00100000		= 2^5
01000000		= 2^6
10000000		= 2^7



이런 식으로 계산할 수 있다.
10010 = 2^4 + 2^1 = 18

Quiz #1

- Go to https://www.slido.com/
- Join #isd-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked as “attendance”.

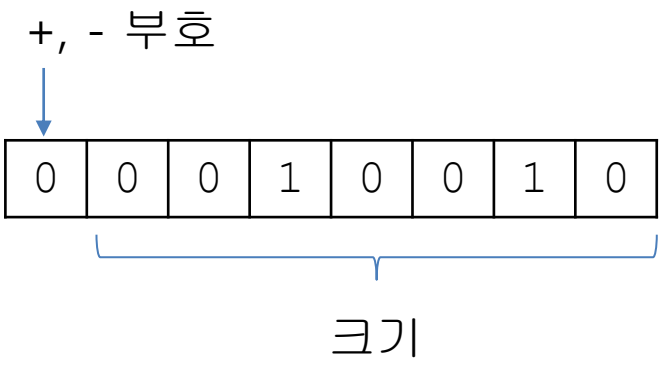
- C

```c
#include <stdio.h>
int main()
{
    int num1 = 10;  // decimal number
    int num2 = 0xA; // hexadecimal number
    int num3 = 012; // octal number

    // print num1, num2, num3 in decimal
    printf("%d\n", num1);
    printf("%d\n", num2);
    printf("%d\n", num3);

    printf("%x\n", num1);   // print num1 in hexadecimal
    printf("%o\n", num1);   // print num1 in octal
    return 0;
}
```

- Python

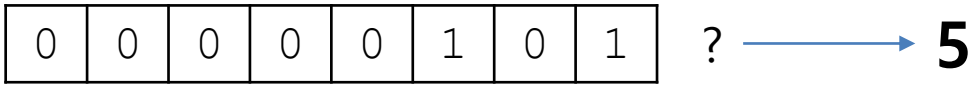
```python
num1 = 10   # decimal number
num2 = 0xA  # hexadecimal number
num3 = 0o12 # octal number

# print num1, num2, num3 in decimal
print('%d'%num1)
print('%d'%num2)
print('%d'%num3)

print('%x'%num1)     # print num1 in hexadecimal
print('%o'%num1)     # print num1 in octal
```

정수의 표현방식



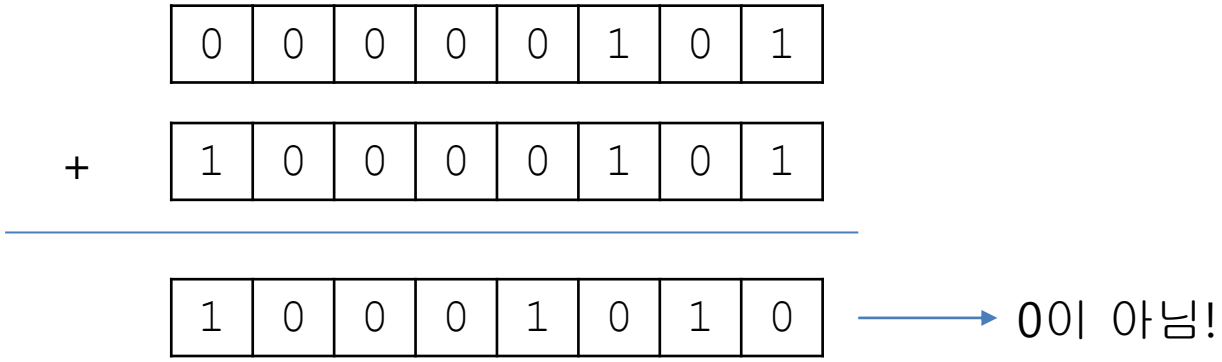
- 가장 왼쪽의 부호를 나타내는 비트를 MSB(Most Significant Bit)라고 함.
- **양수인 경우 0, 음수인 경우 1**

(C에서 정수는 보통 4바이트로 표현되지만 여기에서는 편의상 1바이트로 설명)

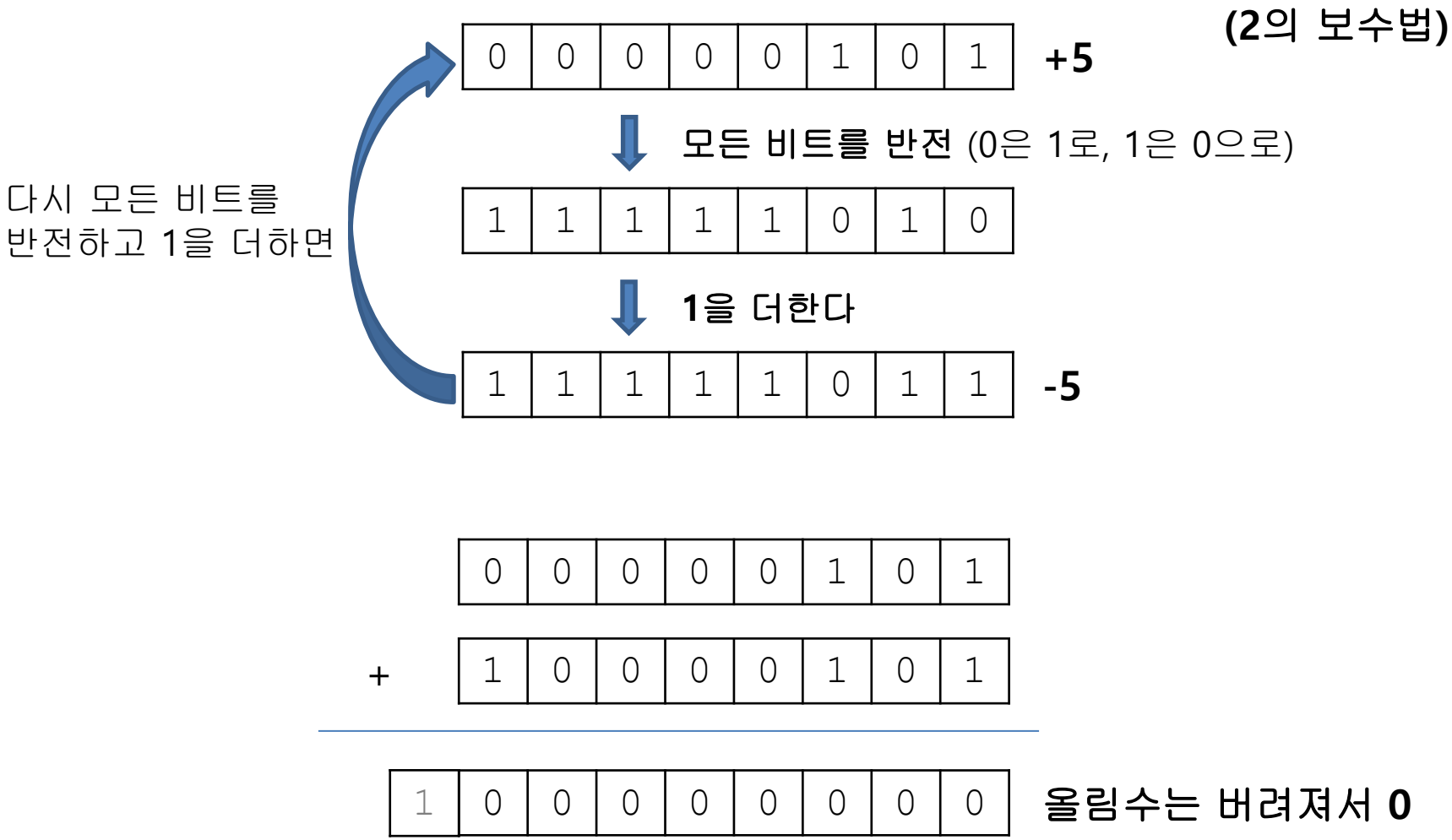


그렇다면 -5는? 10000101?

잘못된 음의 정수 표현 방식



실제 음의 정수 표현 방식


(2의 보수법)
0 0 0 0 0 1 0 1
+5
모든 비트를 반전 (0은 1로, 1은 0으로)
1 1 1 1 1 0 1 0
1을 더한다
1 1 1 1 1 0 1 1
-5
다시 모든 비트를
반전하고 1을 더하면
0 0 0 0 0 1 0 1
+
1 0 0 0 0 1 0 1
1 0 0 0 0 0 0 0 0
올림수는 버려져서 0

Quiz #2

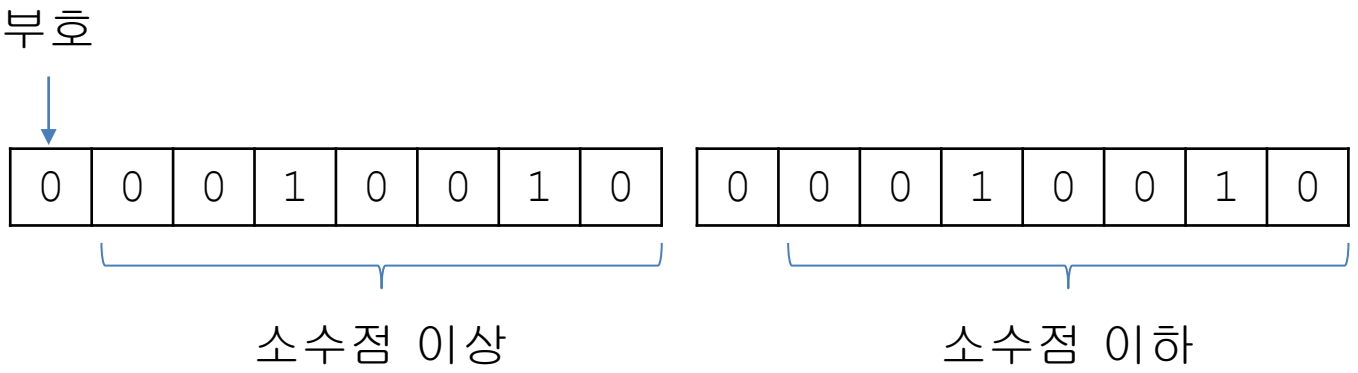
- Go to https://www.slido.com/
- Join #isd-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked as “attendance”.

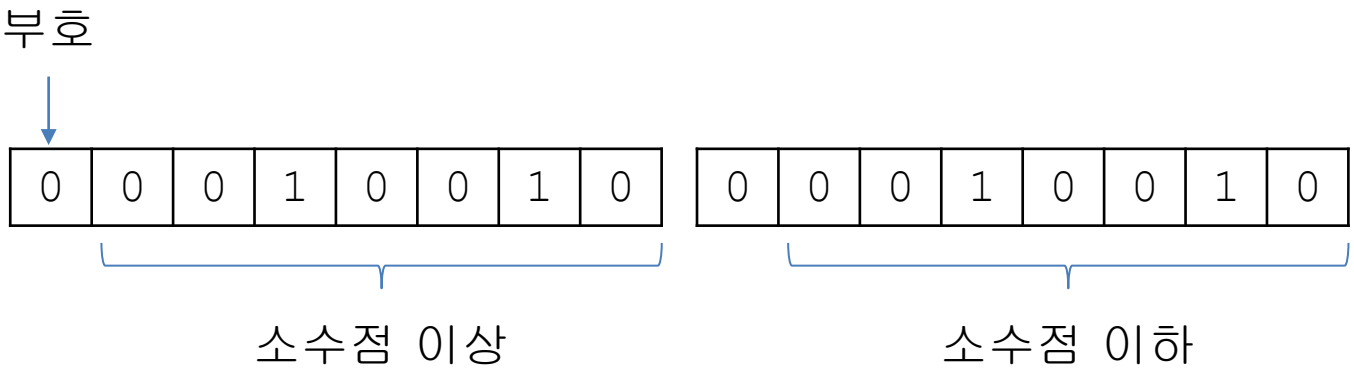
실수의 표현방식

- 가장 먼저 생각해볼 수 있는 방법



- 문제는?

잘못된 실수의 표현방식



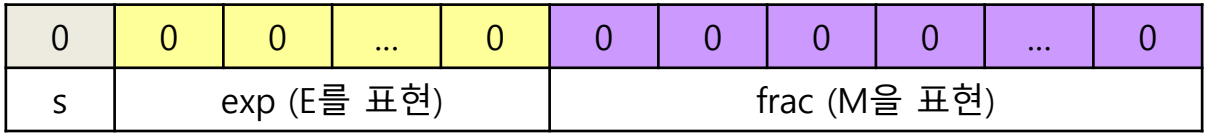
- 표현할 수 있는 실수의 범위가 너무 좁음 (소수점 이하는 0~255까지만 표현 가능)

- 적은 비트로 넓은 범위의 실수를 표현하기 위한 방법 필요

실제 실수의 표현방식

- IEEE 754 format

- $(-1)^s \times M \times 2^E$
  - s : 부호(sign). 양수일 때 0, 음수일 때 1.
  - M : 가수(fraction), E : 지수(exponent)



- float형: exp 8비트, frac 23비트
- double형: exp 11비트, frac 52비트

실수 표현의 “유한한 정밀도”

- **전체 실수 중 유한한 수만을 정확하게 표현할 수 있다는 것을 이해하는 것이 중요!**

-> 유한한 개수의 비트를 사용하기 때문에

- 0.1211과 0.1212사이에도 무한한 실수가 존재
- 컴퓨터는 표현 가능한 “가장 가까운 값”으로 표현
- 컴퓨터의 실수 표현은 **“유한한 정밀도” (finite precision)**을 가진다.

실수의 표현방식 | float, double

- float형: sign 1비트 + exp 8비트 + frac 23비트 = 32비트 = 4바이트

- double형: sign 1비트 + exp 11비트 + frac 52비트 = 64비트 = 8바이트

- double형 변수가 더 많은 바이트를 사용한다

-> double형 변수가 표현할 수 있는 숫자의 범위가 더 넓고, 더 정밀하게 표현할 수 있다.

C Example 1

```c
#include <stdio.h>

int main()
{
    float x = 3.33333333333333333333333333333333333333333333;
    double y = 3.33333333333333333333333333333333333333333333;

    printf("float: %.30f\n", x);
    printf("double: %.30f\n", y);

    return 0;
}
```

C Example 2

```c
#include <stdio.h>

int main()
{
    float num1 = 0.0;
    double num2 = 0.0;

    for(int i=0; i<100; i++)
    {
        num1 += 0.1;
        num2 += 0.1;
    }

    printf("%.20f\n", num1);
    printf("%.20f\n", num2);

    return 0;
}
```

실수의 표현방식 | 참고자료

- 참고자료:
  - https://ko.wikipedia.org/wiki/부동소수점
  - https://namu.wiki/w/컴퓨터에서의 수 표현#s-3
  - IEEE 754 format https://en.wikipedia.org/wiki/IEEE_floating_point

문자의 표현 방식

- 컴퓨터는 모든 데이터를 0과 1로 구성된 숫자로 표현

- 숫자를 이용해 문자를 표현하려면?
- -> 특정 숫자가 특정 문자를 의미하도록 미리 정해놓은 규약을 사용하면 된다.

- 가장 기본적인 규약: ASCII 코드
  - 출력 불가능 제어 문자 33개 + 출력 가능 문자 95개 = 총128개의 문자 표현
  - American Standard Code for Information Interchange (미국정보교환표준부호)
  - https://ko.wikipedia.org/wiki/미국정보교환표준부호

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

https://commons.wikimedia.org/wiki/File:ASCII-Table.svg

자료형(Data Type)이란?

- 데이터를 표현하는 방법

- 숫자를 하나 저장하는 경우를 생각해보자.
  - 정수인지? 실수인지?
  - 정수를 저장한다면 몇 바이트를 사용할 것인가?

- 정수를 저장할 것이고, 저장공간의 크기는 4바이트로, 변수의 이름은 num이라고 하겠다.
- -> int num;

C Data Types

- C에서 변수의 자료형은 크게 두 부류로 나뉨
- 정수형 변수: char, short, int, long 형
- 실수형 변수: float, double 형
- 정수를 저장하는 방식과 실수를 저장하는 방식이 다르기 때문에 정수형/실수형 변수로 나뉜다

C Data Types (64-bit gcc compiler)

Type	Storage size	Value range
char	**1 byte**	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	**4 bytes**	-2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes	-(2^63) to (2^63)-1
unsigned long	8 bytes	0 to (2^64)-1

Type	Storage size	Value range	Precision
float	**4 byte**	1.2E-38 to 3.4E+38	6 decimal places
double	**8 byte**	2.3E-308 to 1.7E+308	15 decimal places
long double	16 byte		

C Data Types

- C 표준에서는 자료형 별 크기를 구체적으로 지정하진 않는다.
  - “short와 int는 최소 2바이트이되, int는 short와 크기가 같거나 더 커야한다”
  - system에 따라, compiler에 따라 다를 수 있다.

- 왜 이렇게 다양한 자료형이 있는지?
- -> 1. 정수와 실수의 데이터 표현방식이 다르다.
- -> 2. 표현할 값의 크기에 따라 선택하여 메모리 공간을 효율적으로 사용할 수 있도록

sizeof 연산자

- 현재 사용하고 있는 C컴파일러의 자료형 별 바이트 크기를 알 수 있음.

```c
#include <stdio.h>

int main()
{
    char a = 30;
    short b = 30;
    int c = 30;
    float d = 1.4;
    double e = 1.4;

    printf("%ld\n", sizeof(a));
    printf("%ld\n", sizeof(b));
    printf("%ld\n", sizeof(c));
    printf("%ld\n", sizeof(d));
    printf("%ld\n", sizeof(e));

    return 0;
}
```

Python 3 Data Types

- **Numeric**
  - *int / float / complex*
  - Types that describe numeric content
- **Boolean**
  - *bool*
  - Types that define true/false relationships
  - *Logical operators: and / or / not*
- **Text**
  - *str*
  - Immutable string objects
  - 1,2, or 3 quotes
- **Sequence**
  - **Lists**
    - Typically homogeneous sequences of objects
    - An mutable, ordered array
    - Square Brackets
  - **Tuple**
    - Typically heterogeneous sequences of objects
    - An immutable collection
    - Parenthesis
- **Mapping**
  - **Dictionaries**
    - A mutable, unordered, associative array
    - Curly Brackets

이 많은 자료형 중 어떤 것을 사용해야 하나요? - 정수의 경우

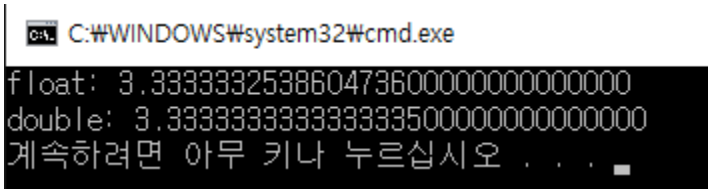
- 일반적으로 **int**를 사용하면 된다.
- 127보다 작은 양수는 char을 쓰는 것이 낫지 않나요? 1바이트 밖에 안 쓰니까…
- CPU가 가장 빠르게 연산할 수 있는 기본 단위: 4 byte인 int형
- -> char, short형 변수는 덧셈을 하면 int형으로 자동으로 변환. 즉, 변환을 위해 불필요한 자원이 소모.
- 연산속도보다 데이터 크기를 줄이는 것이 중요한 경우 char, short형이 유용할 수 있음.

이 많은 자료형 중 어떤 것을 사용해야 하나요? - 실수의 경우

- 일반적으로 **double**을 사용하면 된다.

- float은 정밀도가 너무 낮다.

자료형	소수점 이하 정밀도
float	6자리
double	15자리



- 컴퓨팅 환경의 발전으로 8바이트 double형을 쓰는 것이 그다지 부담스럽지 않다.

실제 코드에서 문자를 어떻게 표현하나?

- '영어 알파벳 문자 하나'는 **char**형 변수에 저장
- -> ASCII 코드 값은 0~127까지로 이루어져 있음
- -> 1byte char형 변수로 충분히 표현 가능

```
int main()
{
    char ch1 = 'A';
    char ch2 = 'C';
}
```

=

```
int main()
{
    char ch1 = 65;
    char ch2 = 67;
}
```

작은 따옴표를 사용하여
표현된 문자 하나는,

해당 문자에 대응하는 **숫자값**을 의미
// A의 ASCII코드 값은 65
// C의 ASCII코드 값은 67

주의: 문자(character)와 문자열(string)

- C

```c
char ch = 'a';        // 문자(character)하나. (사실은 숫자)
char str1[] = "a";    // 길이가 1인 문자열(string)
char str2[] = "abc"; // 길이가 3인 문자열(string)
```

- Python

```python
str1 = 'a'     # 길이가 1인 문자열(string)
str2 = 'abc'   # 길이가 3인 문자열(string)
str3 = "a"     # 길이가 1인 문자열(string)
str4 = "abc"   # 길이가 3인 문자열(string)
```

C & Python Examples

- C

```c
#include <stdio.h>

int main()
{
    int i = 97;  // 알파벳 a의 ASCII 코드값
    char ch = 'a';
    printf("character: %c %c\n", i, ch);
    printf("ASCII code: %d %d\n", i, ch);
    return 0;
}y
```

- Python

```python
i = 97
ch = 'a'
print('%s %s'%(chr(i), ch))
print('%d %d'%(i, ord(ch)))
```

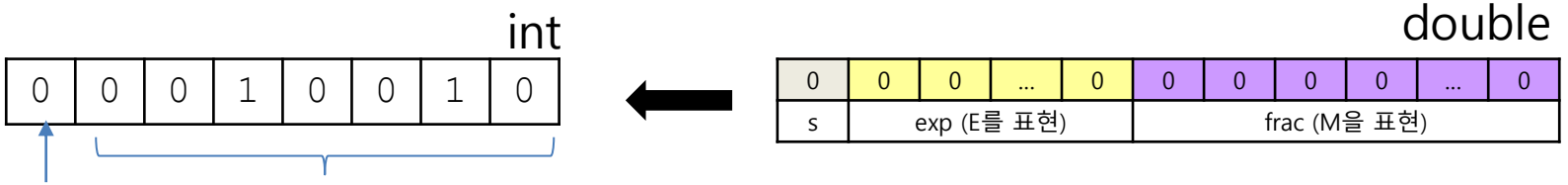
- Python에는 C처럼 문자의 ASCII 코드값을 의미하는 'a'와 같은 문법이 없다.
- chr(i): Unicode 코드값이 정수 i 인 문자를 나타내는 문자열을 리턴
- ord(c): 하나의 Unicode 문자를 나타내는 문자열 c가 주어지면 해당 문자의 Unicode 코드값를 나타내는 정수를 리턴

- Unicode: 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현할 수 있도록 설계된 문자 코드 표. ASCII 코드 표를 포함한다.

자료형의 변환 (Type Casting)

- 자료형: 데이터를 표현하는 방법
- 자료형의 변환: 데이터의 표현방식을 바꾸는 것

```
double num1 = 10;    // int형 정수 10을 double형으로 자동 형 변환
int num2 = 3.14;     // double형 실수 3.14를 int형으로 자동 형 변환
```



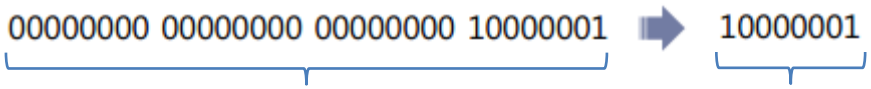
  - num1에는 245.0이 대입
  - num2에는 3이 대입

- 자동으로 발생하는 형 변환 -> 자동 형 변환

자동 형 변환 | 대입연산의 경우

```
double num1 = 10;     // num1에 10.0 대입
int num2 = 3.14;      // num2에 3 대입
```

```
int num3 = 129;
char ch = num3;       // ch에 ? 대입
```



int로 표현된 10진수 129 (4 바이트)

char형으로 변환 (1 바이트로 줄임) : 10진수로 -127

- 데이터 표현 범위가 좁은 자료형으로의 형 변환은, 그 과정에서 데이터의 손실이 발생할 수 있다.
  - 실수 -> 정수, 큰 바이트 형 -> 작은 바이트 형

자동 형 변환 | 정수의 승격

- 정수 데이터는 일반적으로 int형을 사용
  - 왜? CPU가 가장 빠르게 연산할 수 있는 기본 단위: 4 byte인 int형 -> char, short형 변수는 덧셈을 하면 int형으로 자동으로 변환 -> 느려짐

```
int main()
{
    short num1 = 1, num2 = 2;
    short num3 = num1 + num2;    // num1, num2가 int형으로 형 변환
                // -> 정수의 승격 (integral promotion)
}
```

- 컴파일러가 알아서 함. 데이터의 손실을 따로 신경 쓸 필요 없음.

C & Python Examples

- C

```c
#include <stdio.h>
int main()
{
    double num1 = 245;
    int num2 = 3.1415;
    int num3 = 129;
    char ch = num3;
    printf("num1: %f\n", num1);
    printf("num2: %d\n", num2);
    printf("ch: %d\n", ch);
    return 0;
}
```

- C는 정적타입언어(statically typed language, compile time에 변수의 타입을 결정)이므로 대입연산에 의해 자동 형 변환이 일어난다.
- 또한 char, short의 자료형이 있기 때문에 정수의 승격에 의한 자동 형 변환도 일어난다.

- Python

```python
num1 = 245
num2 = 3.1415
num1 = num2
print(num1)
```

- Python은 동적타입언어(dynamically typed language, run time에 변수의 타입을 결정)이므로 대입연산에 의한 자동 형 변환이 존재하지 않는다.
- Python의 정수형 타입은 int 하나밖에 없기 때문에 정수의 승격에 의한 자동 형 변환도 존재하지 않는다.

자동 형 변환 | 피연산자 자료형 불일치의 경우

```
double num1 = 5.15 + 19;
```

- 연산을 하려면 5.15를 정수로 형 변환 or 19를 실수로 형 변환
- -> 실제로는 19를 실수로 형 변환

- 데이터의 손실을 최소화 하는 방향으로
  - 정수형보다 실수형으로
  - 바이트 크기가 큰 자료형으로
  - int → long → long long → float → double → long double

  - ex: long형 정수와 double형 실수를 더하는 경우, long형 정수가 double형 실수로 자동 형 변환됨

명시적 형 변환

- **형 변환 연산자**를 사용해 강제로 형 변환을 명령하는 것

```
double r1 = (double)n1 / n2;
```

  - 참고로 아래 두 코드는 정확히 같은 일을 한다

```
int main()
{
    int num1 = 3;
    double num2 = 2.5 * num1;
}
```

```
int main()
{
    int num1 = 3;
    double num2 = 2.5 * (double)num1;
}
```

어차피 double형으로 자동 형 변환 되기 때문

C & Python Examples

- C

```c
#include <stdio.h>

int main()
{
    // 피연산자 자료형 불일치에 의한 자동 형 변환
    double num1 = 5.15 + 19;
    printf("%f\n", num1);

    // 명시적 형 변환
    double num2 = (int)5.15 + 19;
    printf("%f\n", num2);

    // 주의! 나누기 연산자(/)
    // C에서는 정수/정수 = 정수(몫)
    printf("%f\n", 5/2); // 2.0
    printf("%f\n", (double)5/2); //2.5

    return 0;
}
```

- Python

```python
# Python도 피연산자 자료형 불일치에 의한 자동 형 변환이 일어난다.
num1 = 5.15 + 19
print(num1)

# 명시적 형 변환
num2 = int(5.15) + 19
print(num2)

# 주의! 나누기 연산자(/)
# Python 3에서는 정수/정수 = 실수
print(5 / 2)     # 2.5
```

printf 함수 | Escape Sequence (특수문자)

이스케이프 시퀀스	표현
₩a	벨(경고)
₩b	백스페이스
₩f	폼 피드
₩n	줄 바꿈
₩r	캐리지 리턴
₩t	가로 탭
₩v	세로 탭
₩'	작은따옴표
₩"	큰따옴표
₩₩	백슬래시
₩?	리터럴 물음표

꼭 알아야 하는 것

나머지는 굳이 알 필요 없다!

종종 쓸 일이 있는 것

printf 함수 | Format Specifier (서식문자) printf("%d", 3);

자주 쓰는 것

specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000

C & Python Examples

- C

```c
#include <stdio.h>

int main()
{
    double a = 0.123;
    double b = 0.0000123;

    printf("%f \n", a);

    // 소수점 이하 10번째 자리까지 출력
    printf("%.10f \n", b);

    printf("%e \n", a);
    printf("%e \n", b);

    printf("%g \n", a);
    printf("%g \n", b);
    return 0;
}
```

- Python

```python
a = 0.123
b = 0.0000123

print('%f'%a)
print('%.10f'%b)
print('%e'%a)
print('%e'%b)
print('%g'%a)
print('%g'%b)
```

Quiz #3

- Go to https://www.slido.com/
- Join #isd-hyu
- Click “Poll”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked as “attendance”.

printf 필드 폭 지정

- 보기 좋은 출력을 위해 필드의 폭을 지정 가능
  - %8d : 필드 폭을 8칸 확보하고, 오른쪽 정렬해서 출력
  - %-8d : 필드 폭을 8칸 확보하고, 왼쪽 정렬해서 출력
  - 숫자: 필드의 폭을 의미
  - 부호: 왼쪽 혹은 오른쪽 정렬 지정

```c
#include <stdio.h>
int main()
{
    printf("%-8s%-25s%5s\n", "name", "major", "score");
    printf("%-8s%-25s%5s\n", "Ann", "Computer Science", "100");
    printf("%-8s%-25s%5s\n", "Donald", "Mechanical Engineering", "80");
    printf("%-8s%-25s%5s\n", "Emma", "Electrical Engineering", "90");
    return 0;
}
```

```
name    major                          score
Ann     Computer Science                 100
Donald  Mechanical Engineering            80
Emma    Electrical Engineering            90
```

scanf 함수

- printf와 동일한 서식문자 사용

- **double형 입력**을 위해 %f가 아닌 **%lf**가 사용된다는 점만 다르다.

Next Time

- Labs in this week:
  - Lab1: 과제 7-1
  - Lab2: 과제 7-2

- Next lecture:
  - 8-C03. Functions