C프로그래밍

Lecture 04. 데이터 표현 방식의 이해

동덕여자대학교
데이터사이언스 전공
권 범

목차

01. 컴퓨터가 데이터를 표현하는 방식

02. 정수와 실수의 표현 방식
03. 비트 연산자
04. 연습 문제

01. 컴퓨터가 데이터를 표현하는 방식

❖ 2진수, 10진수, 16진수란 무엇인가? (1/6)

컴퓨터는 모든 데이터의 표현 및 연산을 2진수로 처리합니다.

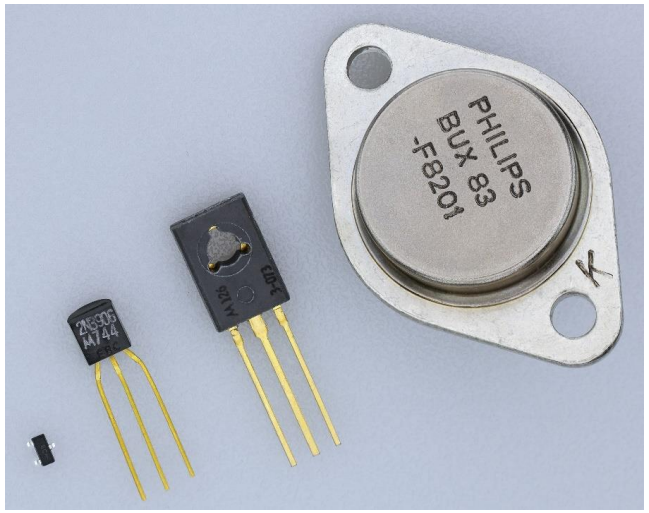
2진수를 이해해야 C언어를 보다 정확히 이해할 수 있습니다.

01. 컴퓨터가 데이터를 표현하는 방식

❖ 2진수, 10진수, 16진수란 무엇인가? (2/6)

컴퓨터와 2진법의 관계

- ✓ 컴퓨터 내부에는 수많은 트랜지스터(Transistor)가 존재합니다.
- ✓ 트랜지스터는 전기 신호로 작동하는 스위치이고, 전기 신호가 들어오면 켜지고 전기 신호가 들어오지 않으면 꺼집니다.
- ✓ 이때 전기 신호가 들어오면 ON 또는 TRUE 상태이고, 컴퓨터는 이것을 1로 인식합니다.
- ✓ 반대로 전기 신호가 없으면 OFF 또는 FALSE 상태이고, 컴퓨터는 이것을 0으로 인식합니다.
- ✓ 따라서, 컴퓨터는 트랜지스터를 통해 전기 신호를 0과 1로 구분하여 처리합니다.
- ✓ 즉 컴퓨터는 2진법을 기반으로 작동합니다.



[사진출처] https://en.wikipedia.org/wiki/File:Transistorer_(cropped).jpg

“트랜지스터 (Transistor)”

반도체 소자의 하나로써, 전기 신호와 전력을 증폭시키거나 스위치 하는데 사용됩니다.

01. 컴퓨터가 데이터를 표현하는 방식

❖ 2진수, 10진수, 16진수란 무엇인가? (3/6)

컴퓨터가 2진법을 사용하는 이유

- ✓ 오류의 최소화와 효율성(비용, 시간) 때문입니다.
- ✓ 컴퓨터는 전기신호를 활용하여 수많은 트랜지스터를 ON/OFF하는 행위를 반복합니다.
- ✓ 즉 2진수를 기반으로, 전기 신호를 0 또는 1로 표현하여 처리합니다.
- ✓ 만약 3이상의 N진수을 사용하게 되면, 전기 신호는 N가지의 경우로 구분됩니다.
- ✓ 그렇게 되면, 오류 발생량과 소요 시간 및 비용이 증가하게 됩니다.
- ✓ 결과적으로 연산 속도는 빨라지지만, 전기 신호를 구분하는 데에는 비효율적으로 동작하게 되는 것입니다.
- ✓ 따라서, 2진수를 기반으로 컴퓨터 시스템이 운영될 때, 오류를 최소화하고 효율적인 시스템 구축이 가능하다고 할 수 있습니다.

최근에는 3진법 반도체 구현 및 양자 컴퓨터 개발이 활발히 진행되고 있습니다.

**기존 컴퓨터가 0과 1만 구분할 수 있는 반면,
양자 컴퓨터에서는 0과 1이 동시에 공존할 수 있다고 합니다.**

01. 컴퓨터가 데이터를 표현하는 방식

❖ 2진수, 10진수, 16진수란 무엇인가? (4/6)

왜 16진수까지 설명하려고 하는 것인가요?

- ✓ 2진수로 데이터를 표현하게 되면, 그 길이가 길어져서 표현하기에도, 한눈에 파악하기에도 어려움이 따릅니다.
- ✓ 16진수로 표현하게 되면 이러한 것들이 한결 수월해 집니다.
- ✓ 그리고 많은 컴퓨터 공학에 관련된 책들이 16진수를 많이 사용합니다.

01. 컴퓨터가 데이터를 표현하는 방식

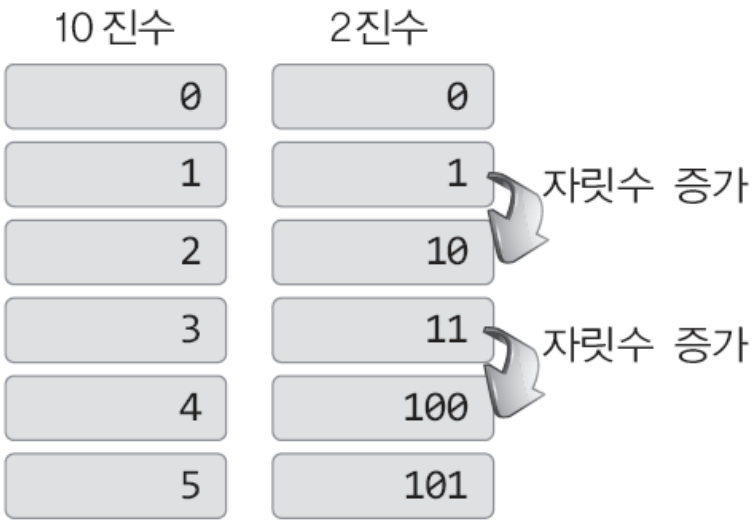
❖ 2진수, 10진수, 16진수란 무엇인가? (5/6)

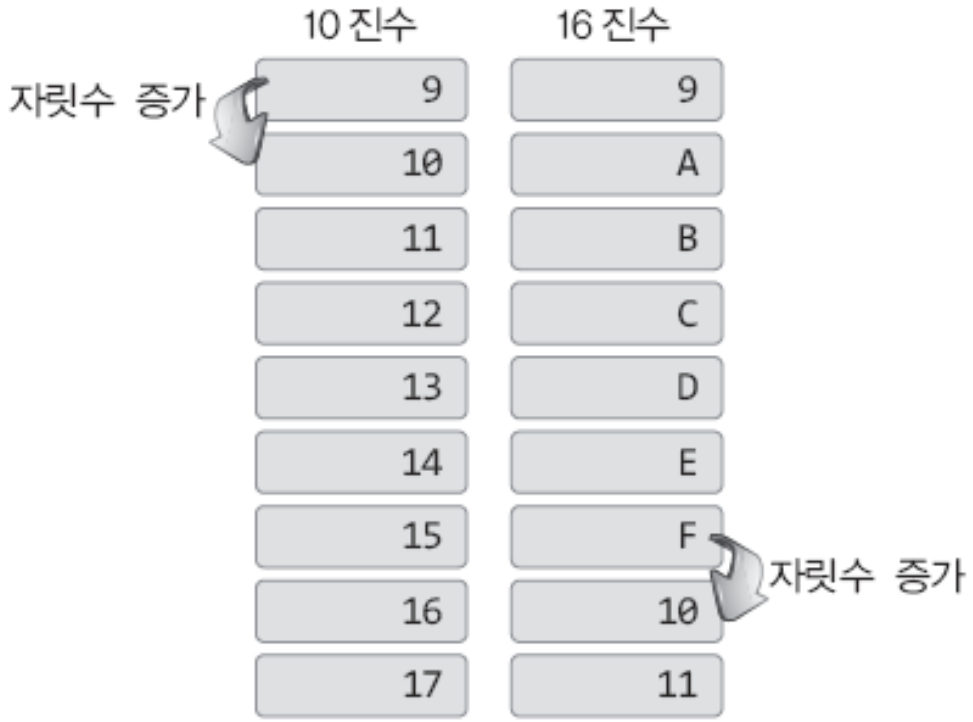
- 2진수: 두 개의 기호를 이용해서 값(데이터)를 표현하는 방식
- 10진수: 열 개의 기호를 이용해서 값(데이터)를 표현하는 방식
- N진수: N 개의 기호를 이용해서 값(데이터)를 표현하는 방식

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 컴퓨터가 데이터를 표현하는 방식

❖ 2진수, 10진수, 16진수란 무엇인가? (6/6)





[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 컴퓨터가 데이터를 표현하는 방식

❖ 데이터의 표현단위인 비트(Bit)와 바이트(Byte) (1/4)

- ✓ 컴퓨터 메모리의 주소 값은 1바이트당 하나의 주소가 할당되어 있습니다.
- ✓ 따라서 바이트는 컴퓨터에 있어서 상당히 의미가 있는 단위입니다.
- ✓ 1byte = 8bit

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 컴퓨터가 데이터를 표현하는 방식

❖ 데이터의 표현단위인 비트(Bit)와 바이트(Byte) (2/4)

문제

1비트가 표현할 수 있는 데이터의 수는 두 가지 (0, 1)입니다.
2비트가 표현할 수 있는 데이터의 수는 네 가지 (00, 01, 10, 11)입니다.
그렇다면 4비트, 1바이트, 그리고 4바이트로 표현할 수 있는 데이터의 개수는 몇 가지일까요?

01. 컴퓨터가 데이터를 표현하는 방식

❖ 데이터의 표현단위인 비트(Bit)와 바이트(Byte) (3/4)

문제

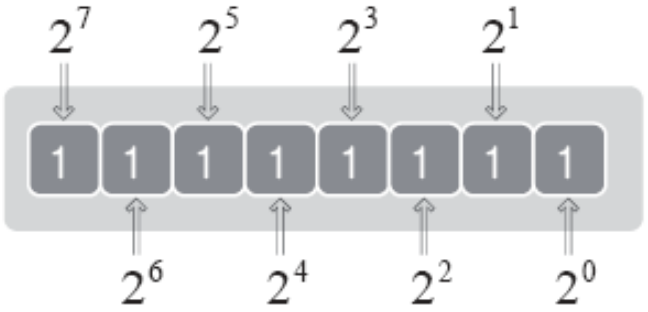
1비트가 표현할 수 있는 데이터의 수는 두 가지 (0, 1)입니다.
2비트가 표현할 수 있는 데이터의 수는 네 가지 (00, 01, 10, 11)입니다.
그렇다면 4비트, 1바이트, 그리고 4바이트로 표현할 수 있는 데이터의 개수는 몇 가지일까요?

해설

n개를 가지고 나타낼 수 있는 경우의 수는 2^n개입니다.
따라서 4비트를 가지고 표현할 수 있는 데이터의 수는 2^4개가 되고,
1바이트(8비트)를 가지고 표현할 수 있는 데이터의 수는 2^8개입니다.
마지막으로 4바이트(32비트)를 가지고 표현할 수 있는 데이터의 수는 2^{32}개가 됩니다.

01. 컴퓨터가 데이터를 표현하는 방식

❖ 데이터의 표현단위인 비트(Bit)와 바이트(Byte) (4/4)



왼쪽의 정보를 이용하면 2진수를 쉽게 10진수로 변환할 수 있습니다.

각 바이트들이 나타내는 값을 10진수로 표현해 볼까요?

0	0	0	1	0	0	0	1

$$0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 16 + 1 = 17$$

1	0	1	0	0	0	1	0

$$1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= 128 + 32 + 2 = 162$$

1	1	1	1	0	1	1	1

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 2^8 - 1 - 2^3 = 247$$

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 컴퓨터가 데이터를 표현하는 방식

❖ 8진수와 16진수로 데이터 표현하기 (1/2)

- C언어는 기본적으로 10진수 이외에 8진수와 16진수 표현도 허용됩니다.

```
int num1 = 10;   // 특별한 선언이 없으면 10진수 표현
int num2 = 0xA;  // 0x로 시작하면 16진수로 인식
int num3 = 012;  // 0으로 시작하면 8진수로 인식
```

- hexadecimal
  (형용사) 16진법의

8진수	10진수	16진수
7	7	7
10	8	8
11	9	9
12	10	a
13	11	b

주의

- ✓ 위의 세 가지 초기화 문장은 완전히 동일합니다.
- ✓ 숫자를 표현하는 방식만 다를 뿐입니다.
- ✓ num1, num2, num3은 완전히 동일한 값을 지니게 됩니다.
- ✓ 물론 컴퓨터 내부적으로는 2진수로 데이터를 저장하였을 것입니다.

01. 컴퓨터가 데이터를 표현하는 방식

❖ 8진수와 16진수로 데이터 표현하기 (2/2)

```
 1  /* notation.c */
 2  #include <stdio.h>
 3
 4  int main(void)
 5  {
 6      int num1 = 0xA7, num2 = 0x43;
 7      int num3 = 032, num4 = 024;
 8
 9      printf("0xA7의 10진수 정수 값: %d\n", num1);
10      printf("0x43의 10진수 정수 값: %d\n", num2);
11      printf(" 032의 10진수 정수 값: %d\n", num3);
12      printf(" 024의 10진수 정수 값: %d\n", num4);
13
14      printf("%d - %d = %d\n", num1, num2, num1 - num2);
15      printf("%d + %d = %d\n", num3, num4, num3 + num4);
16      return 0;
17  }
```

```
0xA7의 10진수 정수 값: 167
0x43의 10진수 정수 값: 67
 032의 10진수 정수 값: 26
 024의 10진수 정수 값: 20
167 - 67 = 100
26 + 20 = 46
```

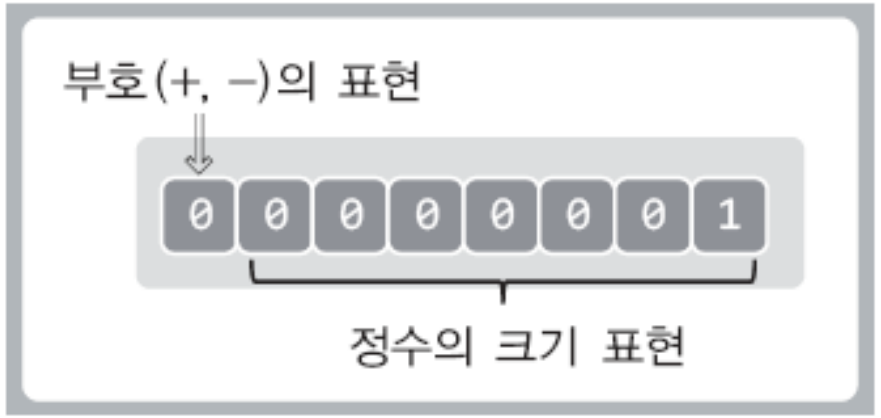
02. 정수와 실수의 표현 방식

01. 컴퓨터가 데이터를 표현하는 방식
03. 비트 연산자
04. 연습 문제

02. 정수와 실수의 표현 방식

❖ 정수(Integer)의 표현 방식

모든 정수의 가장 왼쪽에 있는 비트는 부호 비트입니다.



- ✓ 가장 왼쪽 비트를 MSB(Most Significant Bit)라고 합니다.
- ✓ MSB는 부호를 나타내는 비트입니다.
- ✓ MSB를 제외한 나머지 비트는 크기를 나타내는데 사용됩니다.

- ✓ int 형 변수의 경우 보통은 4바이트 메모리 공간을 이용해서 표현합니다.
- ✓ 본 강의에서는 편의를 위해 1바이트 메모리 공간을 이용해서 정수의 표현 방식을 설명합니다.
- ✓ 정수의 표현 방식을 이해하는 데에 있어 바이트의 크기는 중요하지 않습니다.
- ✓ 바이트의 크기가 크면 그만큼 넓은 범위의 정수를 표현할 수 있을 뿐입니다.

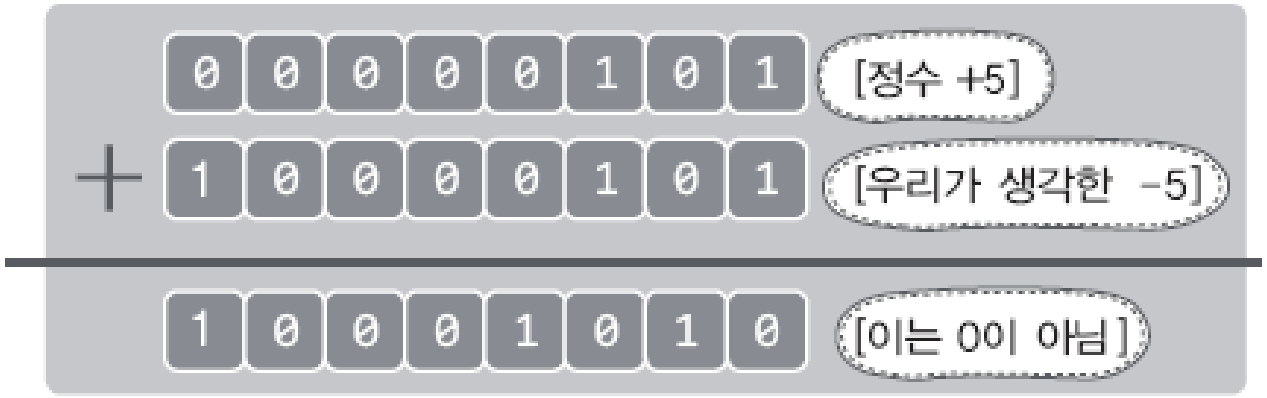
[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

02. 정수와 실수의 표현 방식

❖ 음의 정수의 표현 방식

음수를 표현할 때에는 2의 보수 체계를 기억해야 합니다.

음의 정수를 표현하는 방법(적절하지 못한)

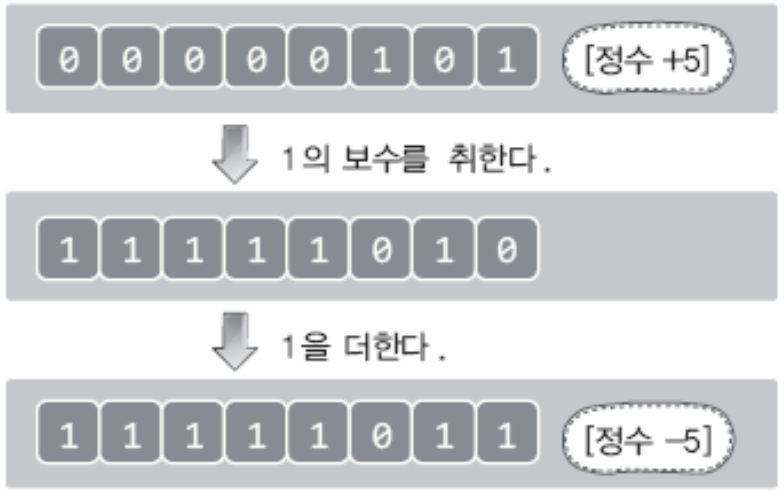


✓ 왼쪽에서는 양의 정수를 표현하는 방식으로 음의 정수를 표현하는 것이 적절치 않은 이유를 설명합니다.

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

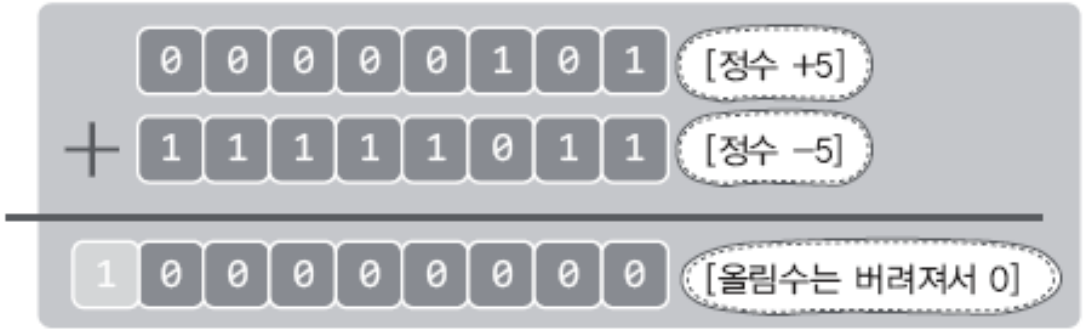
02. 정수와 실수의 표현 방식

❖ 음의 정수의 표현 방식: 2의 보수(Two's Complement) 체계



음의 정수를 표현하는 방법

- ✓ 1의 보수를 취하고 1을 더하는 과정을 2의 보수를 구하는 과정이라고 합니다.
- ✓ 2의 보수를 취하여 음의 정수를 표현합니다.



2의 보수 표현법이 음의 정수를 표현하는 데에 있어서 타당한지를 확인합니다.

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

02. 정수와 실수의 표현 방식

❖ 실수(Real Number)의 표현 방식 (1/3)

- 2바이트 메모리를 가지고 실수를 표현하는 방식을 생각해 봅시다.

- 가장 쉽게 생각할 수 있는 방식은 다음과 같을 것입니다.

① 일단 2바이트를 반으로 나눕니다.
② 절반은 소수점 이상을 표현합니다.
③ 나머지 절반은 소수점 이하를 표현합니다.

실수를 표현하는 방법(적절하지 못한)

✓ 소수점 이상이 0000001이므로 1
✓ 소수점 이하가 00000101이므로 5
✓ 따라서 +1.5라는 뜻으로 해석할 수 있습니다.

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

02. 정수와 실수의 표현 방식

❖ 실수(Real Number)의 표현 방식 (2/3)

- 정수를 표현하는 방식과 유사하게 실수를 표현하면 다음의 문제가 따릅니다.

① 표현할 수 있는 실수의 수가 몇 개 되지 않습니다.
② 3.12456과 3.12457 사이에 있는 무수히 많은 실수조차 제대로 표현하지 못합니다.

따라서 실수를 표현하는 방식은 정수를 표현하는 방식과 다릅니다.

실수를 표현하는 방법(적절하지 못한)

**이렇게 실수를 표현한다면,
나타낼 수 있는 실수의 범위가 얼마나 될까요?**

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

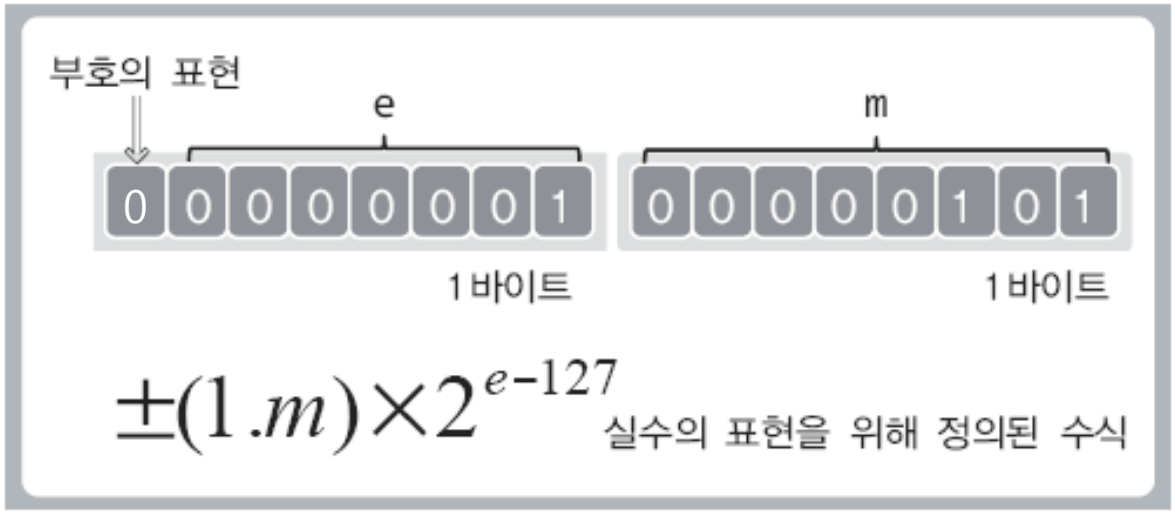
02. 정수와 실수의 표현 방식

❖ 실수(Real Number)의 표현 방식 (3/3)

- 실수를 표현하는 방식은 정수를 표현하는 방식과는 다릅니다

① 실수의 표현 방식에서는 정밀도를 포기하는 대신에 표현할 수 있는 값의 범위를 넓힌다.
② 따라서 컴퓨터는 완벽하게 정밀한 실수를 표현하지 못한다.

컴퓨터는 아래 수식을 이용해서 적은 비트 수를 가지고도 보다 넓은 범위의 실수를 표현할 수 있습니다.



왼쪽 식의 m과 e값에 적절한 값을 대입해서 0.0을 만들어 보세요. 가능한가요?

불가능합니다.
(2^n은 0보다 항상 큽니다.)

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

02. 정수와 실수의 표현 방식

❖ 실수 표현의 오차 확인하기

```
1  /* float_error.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int j;
7      float num = 0.0;
8
9      for(j = 0; j < 100; j++)
10         num = num + 0.1; // 이 연산을 총 100회 진행
11
12     printf("0.1을 100번 더한 결과: %f\n", num);
13     return 0;
14 }
```

```
0.1을 100번 더한 결과: 10.000002
```

✓ 이론적으로 오차 없이 모든 실수를 완벽하게 표현할 능력이 있는 컴퓨팅 환경은 존재하지 않습니다.
✓ 즉, 실수 표현에 있어서의 오차 발생은 C언어의 특성이 아닌 컴퓨터의 기본적인 특성입니다.

03. 비트 연산자

01. 컴퓨터가 데이터를 표현하는 방식
02. 정수와 실수의 표현 방식
04. 연습 문제

03. 비트 연산자

❖ 비트 단위 연산

비트 연산은 주로 하드웨어 관련 프로그래밍에 사용되지만,
그 이외의 프로그래밍 분야에서도 유용하게 사용됩니다.

비트 연산을 적절히 사용하면
요구되는 메모리 공간을 줄이고 성능 향상을 시킬 수 있습니다.

03. 비트 연산자

❖ 비트 연산자(비트 이동 연산자)

연산자	연산자의 기능
&	비트 단위로 AND 연산을 합니다. 예) num1 & num2;
|	비트 단위로 OR 연산을 합니다. 예) num1 | num2;
^	비트 단위로 XOR 연산을 합니다. 예) num1 ^ num2;
~	단항 연산자로서 피연산자의 모든 비트를 반전시킵니다. 예) ~num; // num은 변화 없음, 반전 결과만 반환
<<	피연산자의 비트 열을 왼쪽으로 이동시킵니다. 예) num << 2; // num은 변화 없음, 두 칸 왼쪽 이동 결과만 반환
>>	피연산자의 비트 열을 오른쪽으로 이동시킵니다. 예) num >> 2; // num은 변화 없음, 두 칸 오른쪽 이동 결과만 반환

03. 비트 연산자

❖ & 연산자(비트 단위 AND)

```
1  /* bitwise_operator_and.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 15;
7      int num2 = 20;
8      int num3 = num1 & num2;
9
10     printf("AND 연산의 결과: %d\n", num3);
11     return 0;
12 }
```

```
AND 연산의 결과: 4
```

- ✓ 0 & 0 — 0을 반환
- ✓ 0 & 1 — 0을 반환
- ✓ 1 & 0 — 0을 반환
- ✓ 1 & 1 — 1을 반환

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

03. 비트 연산자

❖ | 연산자(비트 단위 OR)

```
 1  /* bitwise_operator_or.c */
 2  #include <stdio.h>
 3
 4  int main(void)
 5  {
 6      int num1 = 15;
 7      int num2 = 20;
 8      int num3 = num1 | num2;
 9
10      printf("OR 연산의 결과: %d\n", num3);
11      return 0;
12  }
```

```
OR 연산의 결과: 31
```

- ✓ 0 | 0 0을 반환
- ✓ 0 | 1 1을 반환
- ✓ 1 | 0 1을 반환
- ✓ 1 | 1 1을 반환

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

03. 비트 연산자

❖ ^ 연산자(비트 단위 XOR)

```c
/* bitwise_operator_xor.c */
#include <stdio.h>

int main(void)
{
    int num1 = 15;
    int num2 = 20;
    int num3 = num1 ^ num2;

    printf("XOR 연산의 결과: %d\n", num3);
    return 0;
}
```

```
XOR 연산의 결과: 27
```

- ✓ 0 ^ 0 0을 반환
- ✓ 0 ^ 1 1을 반환
- ✓ 1 ^ 0 1을 반환
- ✓ 1 ^ 1 0을 반환

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

03. 비트 연산자

❖ ~ 연산자(비트 단위 NOT)

```
1  /* bitwise_operator_not.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 15;
7      int num2 = ~num1;
8
9      printf("NOT 연산의 결과: %d\n", num2);
10     return 0;
11 }
```

```
NOT 연산의 결과: -16
```

✓ ~ 연산 전 00000000 00000000 00000000 00001111

✓ ~ 연산 후 11111111 11111111 11111111 11110000

✓ ~0 1을 반환
✓ ~1 0을 반환

03. 비트 연산자

❖ << 연산자: 비트의 왼쪽 이동(Left Shift) (1/2)

```c
/* bitwise_operator_left_shift.c */
#include <stdio.h>

int main(void)
{
    int num = 15;

    int result1 = num << 1;
    int result2 = num << 2;
    int result3 = num << 3;

    printf("1칸 이동 결과: %d\n", result1);
    printf("2칸 이동 결과: %d\n", result2);
    printf("3칸 이동 결과: %d\n", result3);
    return 0;
}
```

```
1칸 이동 결과: 30
2칸 이동 결과: 60
3칸 이동 결과: 120
```

03. 비트 연산자

❖ << 연산자: 비트의 왼쪽 이동(Left Shift) (2/2)

```
 1  /* bitwise_operator_left_shift.c */
 2  #include <stdio.h>
 3
 4  int main(void)
 5  {
 6      int num = 15;
 7
 8      int result1 = num << 1;
 9      int result2 = num << 2;
10      int result3 = num << 3;
11
12      printf("1칸 이동 결과: %d\n", result1);
13      printf("2칸 이동 결과: %d\n", result2);
14      printf("3칸 이동 결과: %d\n", result3);
15      return 0;
```

```
1칸 이동 결과: 30
2칸 이동 결과: 60
3칸 이동 결과: 120
```

- ✓ 00000000 00000000 00000000 00001111 // 10진수로 15
- ✓ 00000000 00000000 00000000 00011110 // 10진수로 30
- ✓ 00000000 00000000 00000000 00111100 // 10진수로 60
- ✓ 00000000 00000000 00000000 01111000 // 10진수로 120

- ✓ 왼쪽으로 한 칸씩 이동할 때마다 정수의 값은 두 배씩 증가합니다.
- ✓ 오른쪽으로 한 칸씩 이동할 때마다 정수의 값은 반으로 줄어듭니다.

03. 비트 연산자

❖ >> 연산자: 비트의 오른쪽 이동(Right Shift) (1/2)

```
 1  /* bitwise_operator_right_shift.c */
 2  #include <stdio.h>
 3
 4  int main(void)
 5  {
 6      int num = -16;
 7
 8      printf("1칸 이동 결과: %d\n", num >> 1);
 9      printf("2칸 이동 결과: %d\n", num >> 2);
10      printf("3칸 이동 결과: %d\n", num >> 3);
11      return 0;
12  }
```

```
1칸 이동 결과: -8
2칸 이동 결과: -4
3칸 이동 결과: -2
```

- ✓ 11111111 11111111 11111111 11110000 // 10진수로 -16
- ✓ 11111111 11111111 11111111 11111000 // 10진수로 -8
- ✓ 11111111 11111111 11111111 11111100 // 10진수로 -4
- ✓ 11111111 11111111 11111111 11111110 // 10진수로 -2

- ✓ 왼쪽으로 한 칸씩 이동할 때마다 정수의 값은 두 배씩 증가합니다.
- ✓ 오른쪽으로 한 칸씩 이동할 때마다 정수의 값은 반으로 줄어듭니다.

03. 비트 연산자

❖ >> 연산자: 비트의 오른쪽 이동(Right Shift) (2/2)

- num >> 2;

✓ 11111111 11111111 11111111 11110000 // 10진수로 -16

CPU에 따라서 연산의 결과가 달라질 수 있습니다.

✓ 11111111 11111111 11111111 11111100 // 1이 채워지는 경우
✓ 00111111 11111111 11111111 11111100 // 0이 채워지는 경우

✓ 왼쪽 비트가 0으로 채워진 음수의 경우, 오른쪽으로 비트를 이동시킨 결과는 CPU에 따라서 달라집니다.
✓ 따라서 호환성이 요구되는 경우에는 >> 연산자의 사용을 제한해야 합니다.

04. 연습 문제

01. 컴퓨터가 데이터를 표현하는 방식
02. 정수와 실수의 표현 방식
03. 비트 연산자

04. 연습 문제

❖ 연습 문제 1.

- 입력받은 음의 정수 값을 양의 정수 값으로 바꿔서 출력하는 프로그램을 작성하세요.
  (단, 비트 단위 연산자를 사용해서 구현해야 하며, 양의 정수 값은 입력되지 않는다고 가정합니다.)
  예를 들어, -3이 입력되면 3이 출력되어야 합니다.

04. 연습 문제

❖ 연습 문제 1. 정답 및 해설

```
1  /* example1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int n;
8
9      printf("음의 정수 값을 입력하세요: ");
10     scanf("%d", &n);
11
12     n = ~n;
13     n = n + 1;
14
15     printf("%d\n", n);
16
17     return 0;
18 }
```

```
음의 정수 값을 입력하세요: -3
3
```

04. 연습 문제

❖ 연습 문제 2.

- 사용자로부터 입력받은 값의 두 배를 계산해서 출력해 주는 프로그램을 * (곱셈) 연산이 아닌, 비트 시프트 연산을 이용해서 구현해 보세요.
  (단, 입력값은 $-2^{30}-1$보다 크고, 2^{30}보다 작다고 가정합니다.)

04. 연습 문제

❖ 연습 문제 2. 정답 및 해설

```
1  /* example2.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int a;
8
9      printf("정수 값을 입력하세요: ");
10     scanf("%d", &a);
11
12     a = a << 1;
13     printf("%d\n", a);
14
15     return 0;
16 }
```

```
정수 값을 입력하세요: 5
10
```

끝맺음

- ❖ 01. 컴퓨터가 데이터를 표현하는 방식
- ❖ 02. 정수와 실수의 표현 방식
- ❖ 03. 비트 연산자
- ❖ 04. 연습 문제

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr