

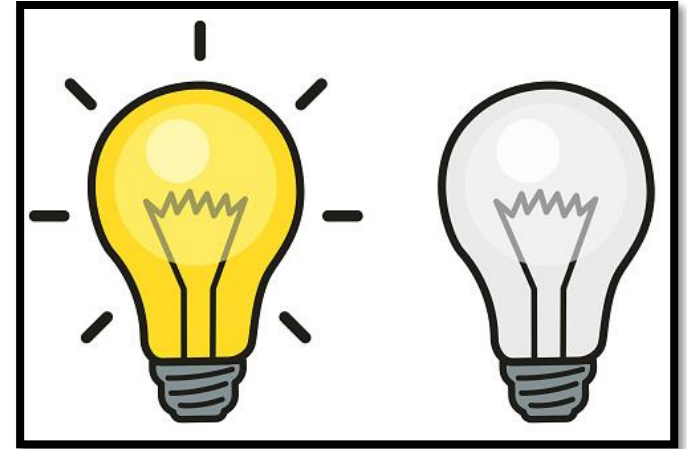
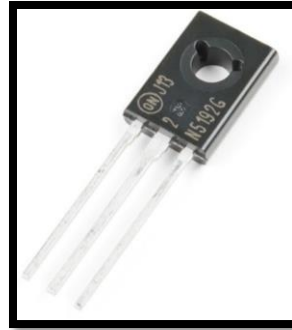
Variables and Standard I/O

- Section 0** 비트, 바이트, 그리고 진법
- Section 1** 변수의 이해
- Section 2** 표준 입출력 함수
- Section 3** Regular Expression (Regex)



0 비트, 바이트, 그리고 진법 비트





● 컴퓨터에서 표현하는 가장 작은 단위



- 하나의 비트로 **0**과 **1**을 표현
- 전압이 걸리면 1, 안걸리면 0 → 보통 전구의 ON / OFF로 묘사

● 비트를 나열하여 만든 수: 2진수

표 3-3 전기 스위치로 표현 가능한 가짓수

전기 스위치	의미	2진수(0, 1)	10진수
	꺼짐, 꺼짐	00	0
	꺼짐, 켜짐	01	1
	켜짐, 꺼짐	10	2
	켜짐, 켜짐	11	3

전기 스위치 n 개로 표현할 수 있는 가짓수 = 2^n

0 비트, 바이트, 그리고 진법 바이트

● 컴퓨터에서 의미를 갖는 데이터의 가장 작은 단위

- 1 bit로는 두 개 밖에 구분하지 못하니까 bit를 여러 개 묶어서 쓰자!
- 비트 8개가 합쳐진 단위

표 3-5 비트와 바이트의 크기에 따른 숫자의 범위

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
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	$2^{16}=65536$	0~11111111 11111111	0~65535	0~FFFF
32	4	2^{32} =약 42억	0~...	0~약 42억	0~FFFF FFFF
64	8	2^{64} =약 1800경	0~...	0~약 1800경	0~...

● 1 bit는 몇 개의 데이터를 표현 가능한가?

● 왜 가능한가?

● 2진수: 두 개의 숫자로 표현한 수

- 각 자리에 들어갈 수 있는 숫자는 단 두개
- 0과 1

● 10진수: 10개의 숫자로 표현한 수

- 각 자리에 들어갈 수 있는 숫자 0~9

● 16진수: 16개의 숫자로 표현한 수

- 0~9와 여섯 개의 알파벳 a, b, c, d, e, f

● 진수를 구분하는 방법

- 2진수 : 10_2 또는 0b10
- 10진수 : 10_{10}
- 16진수 : 10_{16} 또는 0x16

표 3-4 10진수, 2진수, 16진수로 나타낸 0~15

10진수(0~9)	2진수(0, 1)	16진수(0~F)
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

0 비트, 바이트, 그리고 진법 진법 변환

● 2진수를 10진수로 변환

- 2진수의 맨 뒤에서부터 각 자리에 해당하는 가중치인 $2^0, 2^1, 2^2, \dots$ 을 곱한 후
- 각 자리의 결과를 모두 합한다.

1	0	0	1			0	0	1	1	→ 2진수
×	×	×	×			×	×	×	×	
2^7	2^6	2^5	2^4			2^3	2^2	2^1	2^0	
128	0	0	16			0	0	2	1	→ 10진수
+										
147										→ 10진수

그림 3-17 2진수를 10진수로 변환하는 방법

0 비트, 바이트, 그리고 진법 진법 변환

● 2진수를 16진수로 변환한 후 10진수로 변환

- 2진수의 4자리는 16진수의 한 자리임.
- 16진수를 10진수로 바꾸려면 맨 뒤에서부터 각 자리에 해당하는 가중치인 16^0 , 16^1 을 곱한 후 각 자리의 결과를 모두 합하면 된다.

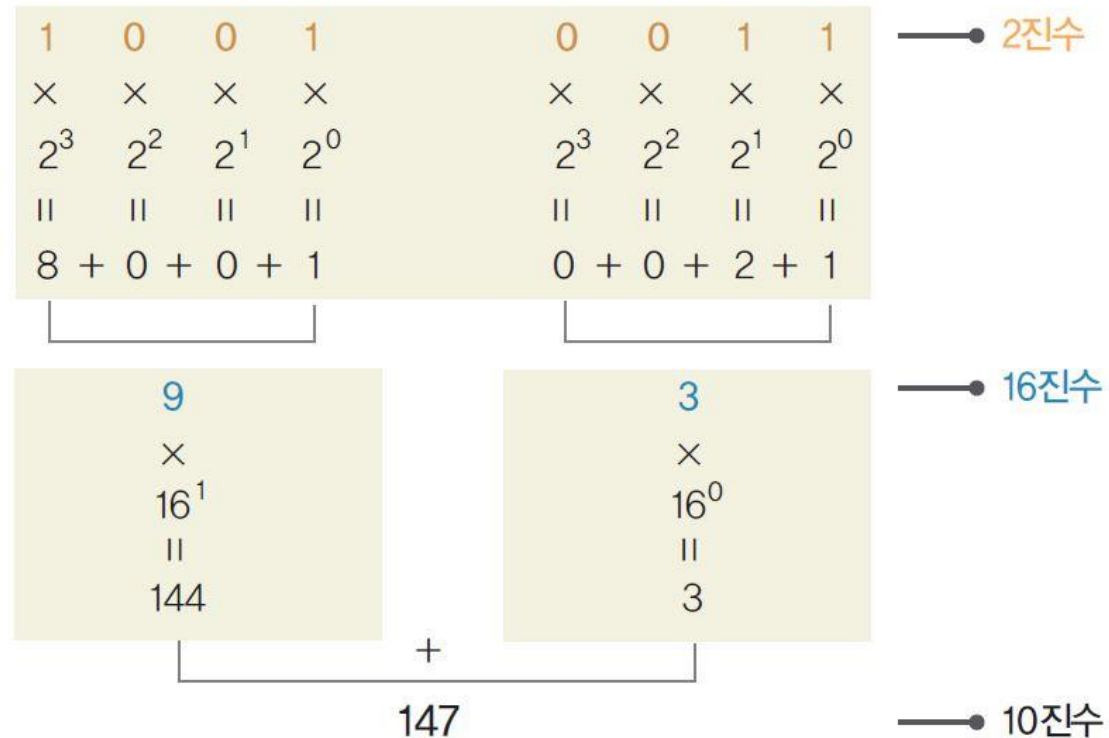


그림 3-18 2진수를 16진수로 변환한 후 10진수로 변환하는 방법

0 비트, 바이트, 그리고 진법 진법 변환

● 2진수 변환 연습

- 10진수를 2진수로 변환
 - 10진수를 계속 2로 나눠 나가면서 그 몫과 나머지를 구함.
- 16진수를 2진수로 변환
 - 16진수를 계속 2로 나눠 나가면서 그 몫과 나머지를 구함 2진수를 16진수로 변환한 후 10진수로 변환

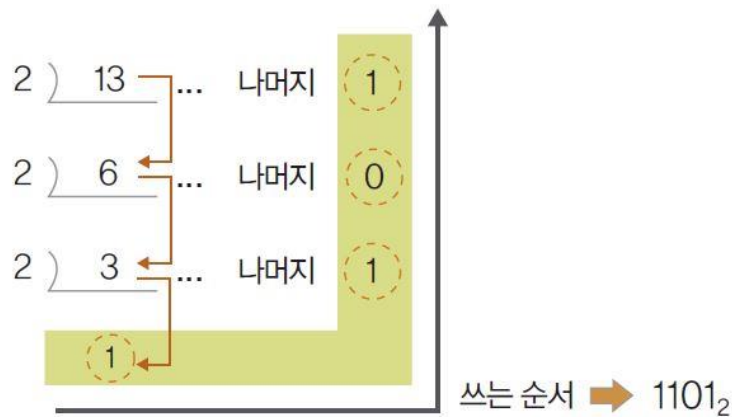


그림 3-19 10진수를 2진수로 변환하는 방법

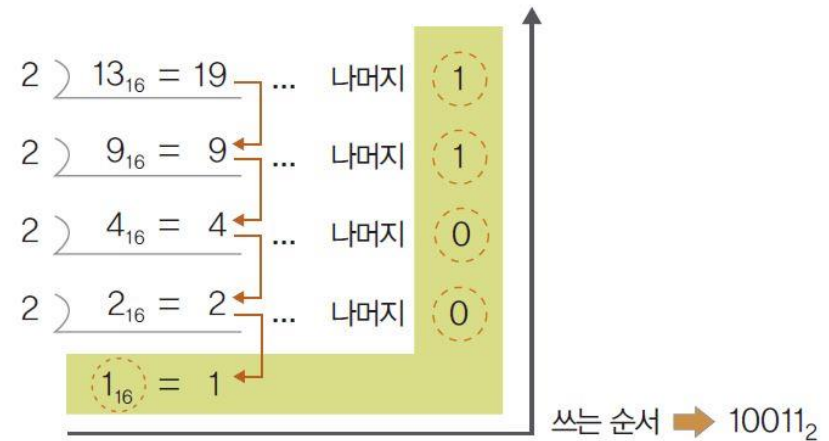


그림 3-20 16진수를 2진수로 변환하는 방법

0 비트, 바이트, 그리고 진법 진법 변환

● 16진수를 2진수로 변환 (간편한 방법)

표 3-6 16진수와 2진수

16진수	2진수	16진수	2진수
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

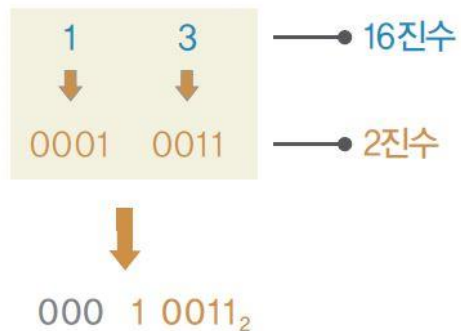


그림 3-21 16진수를 2진수로 변환하는 간편한 방법 1

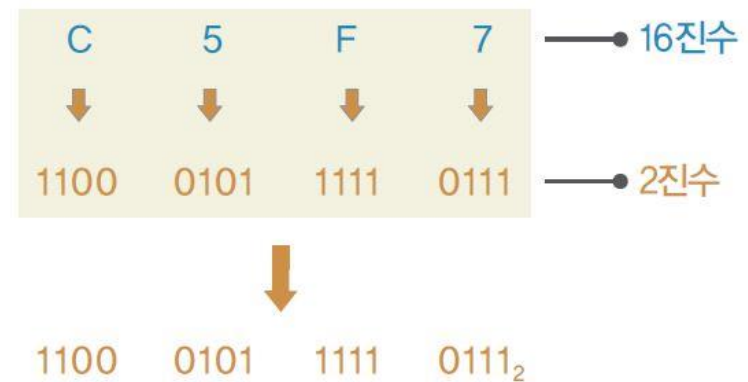


그림 3-22 16진수를 2진수로 변환하는 간편한 방법 2

● 정리

- 진법 변환하는 방법을 숙달하자. → 컴퓨터공학의 출발
- 컴퓨터에 10진수를 효율적으로 저장할 수 없기 때문에 2진수로 저장하고
- 2진수는 시각적으로 불편하기 때문에 16진수로 표현한다.
- 우리에게 익숙한 숫자: 0xFF

● 진법 변환 숙달의 요령

- 16진수 또는 2진수를 보고 빠르게 대략적인 10진수의 값을 파악하도록
- 또는
- 10진수를 보고 16진수나 2진수의 대략적인 자릿수를 추정하는 것을
- 반드시 숙달하도록 하자.

● 정리

- 진법 변환하는 방법을 숙달하자. → 컴퓨터공학의 출발
- 컴퓨터에 10진수를 효율적으로 저장할 수 없기 때문에 2진수로 저장하고
- 2진수는 시각적으로 불편하기 때문에 16진수로 표현한다.
- 우리에게 익숙한 숫자: 0xFF

● 진법 변환 숙달의 요령

- 16진수 또는 2진수를 보고 빠르게 대략적인 10진수의 값을 파악하도록
- 또는
- 10진수를 보고 16진수나 2진수의 대략적인 자릿수를 추정하는 것을
- 반드시 숙달하도록 하자.

1 변수의 이해

변수와 조작법

● 변수란?

- 수학에서의 변수: 수, 바뀔 수 있다 ex) $y=2x+1$
- 컴퓨터의 변수: 메모리 특정 위치의 값, 바뀔 수 있다.

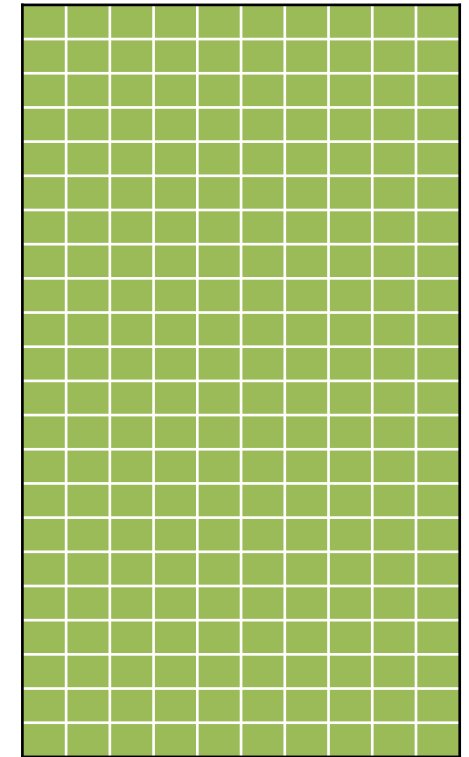
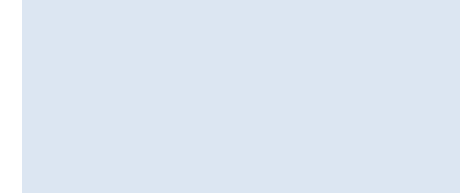
● C언어에서 변수의 조작 순서

- 1) 선언 - 종류 선택 필수
- 2) 대입 (입력)
- 1)과 2)가 최소 1회 수행되어야 가져오기를 할 수 있다.

● 각 구문별 의미

- 1) 선언: 메모리에서 특정 용도를 위해 공간을 확보
- 2) 대입: 1)에서 확보된 공간에 값을 기록
- 3) 가져오기: 기록된 값을 가져온다.

(만약 1)과 2)를 한번이라도 하지 않는다면?)



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해 변수의 선언

● 변수 선언

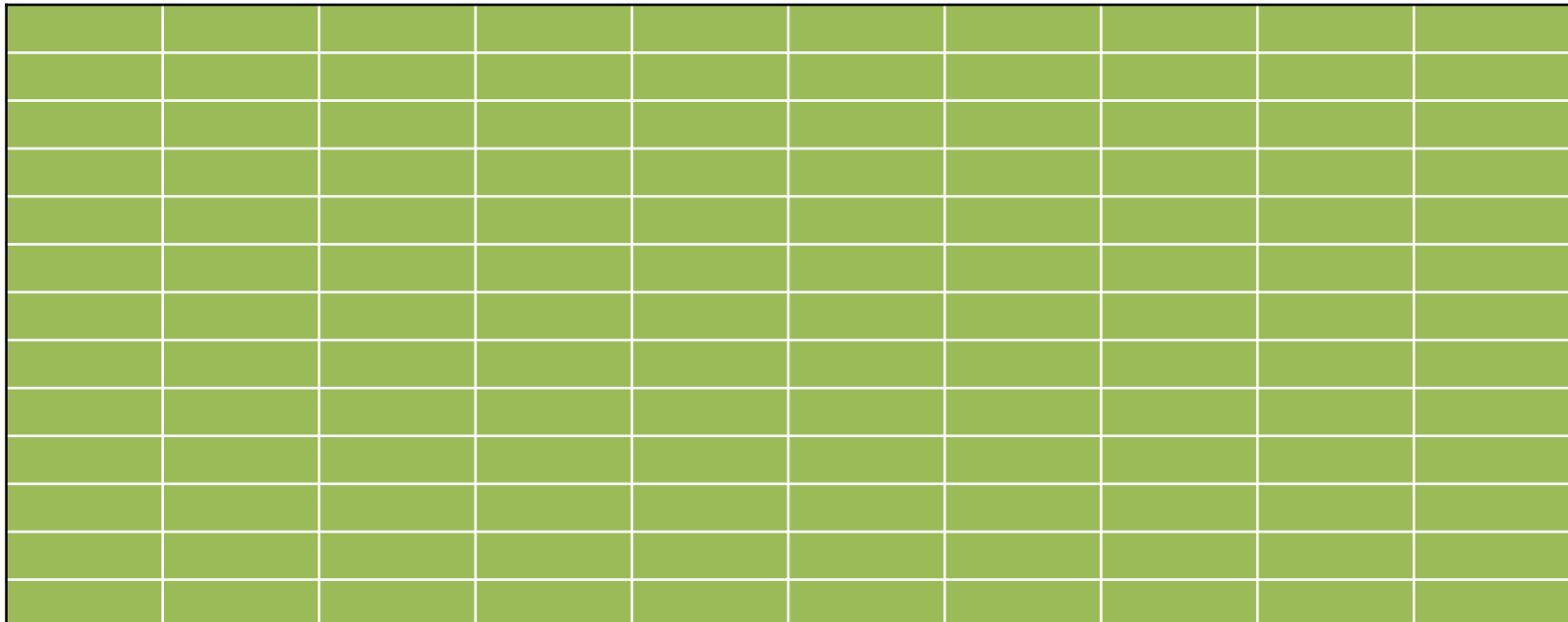
- 메모리 특정 위치의 값을 어떻게 사용하겠다고 사전 지정
- 1) 메모리 공간을 특정한 유형으로 쓸 것이다
- 2) 메모리 공간을 달라

```
#!/usr/bin/python

counter = 100      # An integer assignment
miles   = 1000.0   # A floating point
name    = "John"   # A string

print counter
print miles
print name
```

Python에서의 변수 사용



RAM 또는 가상 메모리

같이 그림을 그려보자

1 변수의 이해

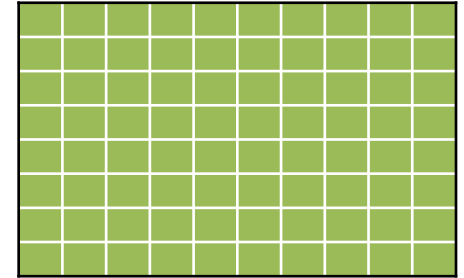
변수의 선언

● 변수 선언

- 소수점이 없는 값과 소수점이 있는 값을 담는 변수 선언

```
int a;
float b;
```

```
int a, b;
float a, b;
```



RAM 또는 가상 메모리
같이 그림을 그려보자

- 변수 a보다 변수 b 크기가 더 큰 것은 정수형 변수와 실수형 변수의 크기가 다르기 때문
- 변수 종류가 같으면 변수를 개별적으로 선언해도 되고 comma(,)를 사용하여 연속 선언해도 된다.

```
int a;
int b;
```

=

```
int a, b;
```

❶ 가능

```
int a;
float b;
int c;
float d;
```

==

❷ 가능

```
int a, c;
float b, d;
```

≠

❸ 불가능

```
int a, float b;
int c, float d;
```

1 변수의 이해

변수의 종류

● 숫자형 변수 (Number Type)

- 정수형: 소수점 이하의 자리가 없는 이산적인 값

표 3-7 정수형 데이터 형식

데이터 형식	의미	크기	값의 범위
short	작은 정수형	2바이트	$-2^{15}(-32768) \sim 2^{15}-1(32767)$
unsigned short	부호 없는 작은 정수형	2바이트	$0 \sim 2^{16}-1(65535)$ <small>→ 정수 표현</small>
int	정수형	4바이트	$-2^{31}(\text{약 } -21\text{억}) \sim 2^{31}-1(\text{약 } 21\text{억})$
unsigned int	부호 없는 정수형	4바이트	$0 \sim 2^{32}-1(\text{약 } 42\text{억})$
long int(또는 long)	큰 정수형	4바이트	$-2^{31} \sim 2^{31}-1$
unsigned long	부호 없는 큰 정수형	4바이트	$0 \sim 2^{32}-1$

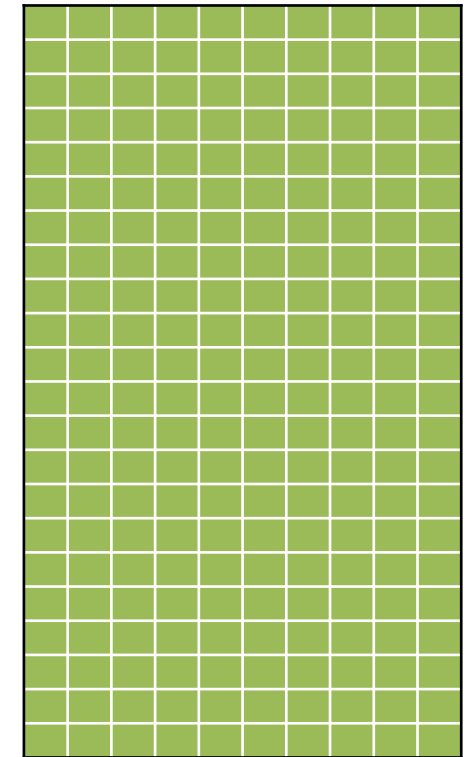
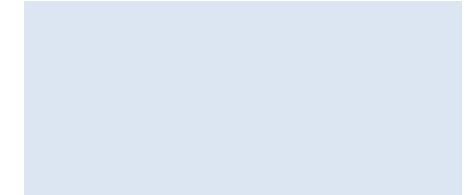
- 가장 많이 쓰는 것은 int형
- 최소, 최대값의 크기에 따라 선택적 사용

- 실수형: 소수점 이하의 자리가 있는 연속적인 값

표 3-8 실수형 데이터 형식

데이터 형식	의미	크기	값의 범위
float	실수형	4바이트	약 $-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$
double	큰 실수형	8바이트	약 $-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$
long double	큰 실수형	8바이트	약 $-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$

- 가장 많이 사용하는 것은 float과 double
- 정밀도 요구사항에 따라 선택 사용



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해

변수의 종류

● 문자형 변수 (Character Type)

- 문자형을 표시하는 방법: 숫자에 문자를 대응시킨다.
- ascii 코드: 아스키 코드 – [전체 테이블](#)

표 3-9 아스키코드

아스키코드	10진수	16진수
0 ~ 9	48 ~ 57	0x30 ~ 0x39
A ~ Z	65 ~ 90	0x41 ~ 0x5A
a ~ z	97 ~ 122	0x61 ~ 0x7A

- 숫자 0은 48, 대문자 A는 65, 소문자 a는 97 등은 알아두면 편함.
- C 언어에서는 숫자를 문자로도 표현함.

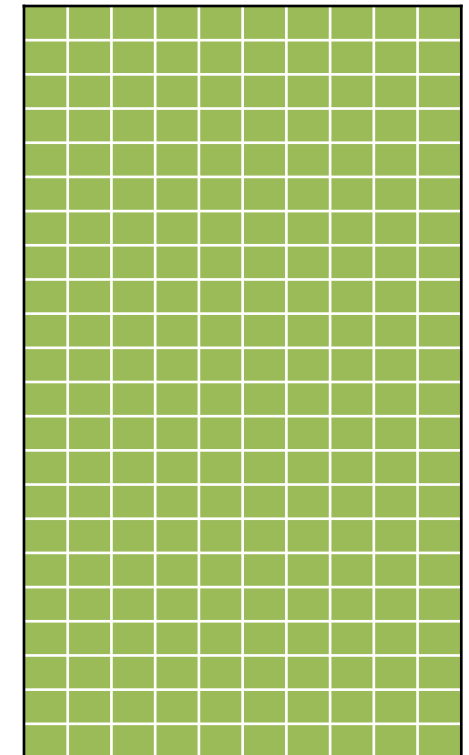
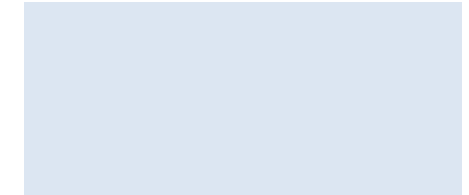
`char ch = 'a';` == `char ch = 97;`

- **문자형:** C에서 문자형 (Character Type)은 글자 하나를 뜻한다.

표 3-10 문자형 데이터 형식

데이터 형식	의미	크기	값의 범위
char	문자형 또는 정수형	1바이트	$-2^7(-128) \sim 2^7-1(127)$
unsigned char	문자형 또는 부호 없는 정수형	1바이트	$0 \sim 2^8-1(255)$

- cf) String형



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해

변수의 종류

● 문자열형 변수 (String Type)

- 엄밀히 C에는 String Type의 변수가 없다.
- 문자열은 한 문자가 여러 개 이어 붙인 것.
- 자신의 영문명 중 Last Name을 옆에 그려보자
 - 어떤 문제를 발견했는가?

● String형 문자를 지원하기 위한 C언어의 방법

- 문자형 변수를 이어 붙일 수 있도록 한다. → `char str[7];`
- 널(null) 문자 : `\0`
 - **출력 함수 동작 시**, 문자열의 끝을 알려주는 역할
 - 화면에 출력되지 않음
- 문자열은 문자 배열을 사용

```
char str[7];
```

str

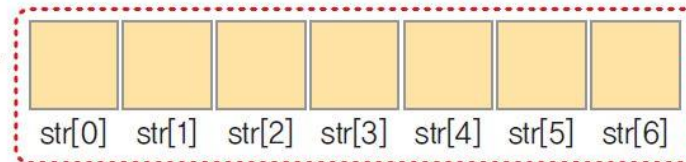
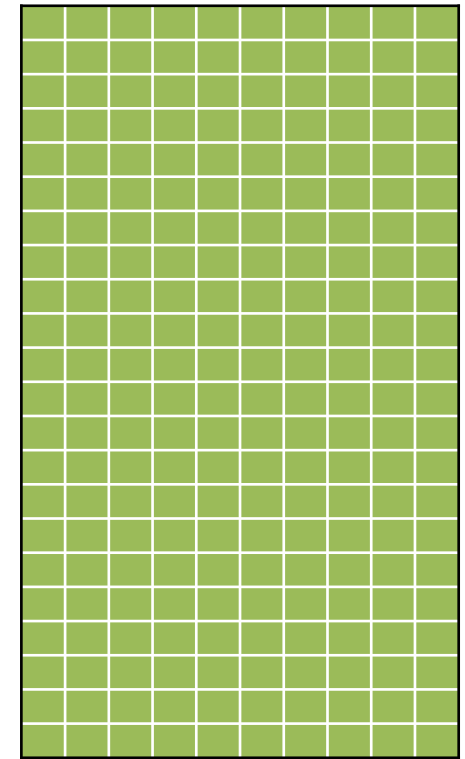
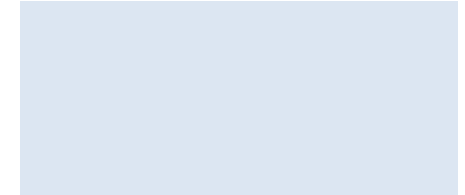


그림 3-25 배열로 표현한 문자열



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해 변수의 종류

● String형 문자를 지원하기 위한 C언어의 방법

- 출력 함수 동작 시

```
char str[10] = "0123456789";
```

↓
인덱스 자릿수 의미

```
char str[10] = "0123456789";
str[0] = 'I';
str[1] = 'T';
str[2] = 'C';
str[3] = 'o';
str[4] = 'o';
str[5] = 'k';
str[6] = '\0';
```



그림 3-28 [응용 3-15]의 str 배열 내용 1

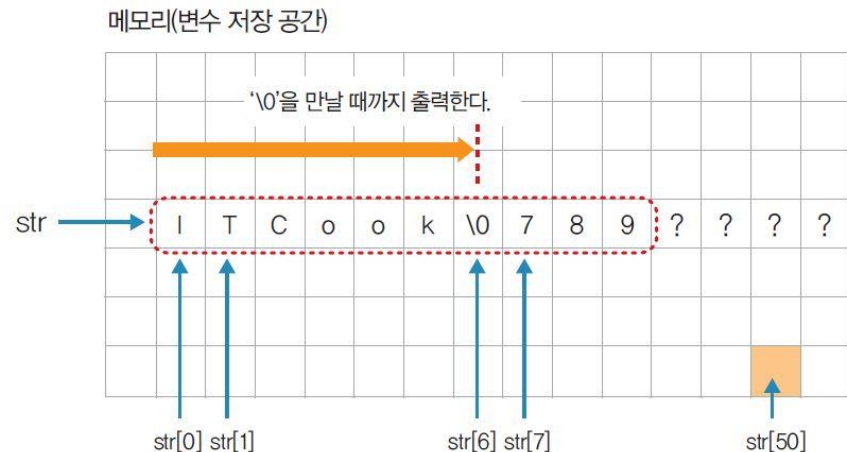


그림 3-29 [응용 3-15]의 str 배열 내용 2

1 변수의 이해 변수의 대입

● 대입 연산자 (Assignment Operator): =

- 같다는 의미의 등호가 아니다.
- 원칙적으로 정수형 변수에는 정수 값을, 실수형 변수에는 실수 값을 사용

```
int ten;  
ten = 10;
```

```
double sqrt_two;  
sqrt_two = 1.414;
```

```
char grade;  
grade = 'A';
```

```
char grade;  
grade = 65;
```

```
char grade;  
grade = 0x41;
```

- 문자형 값은 앞뒤로 작은따옴표 사용, 문자열 값은 큰따옴표 사용

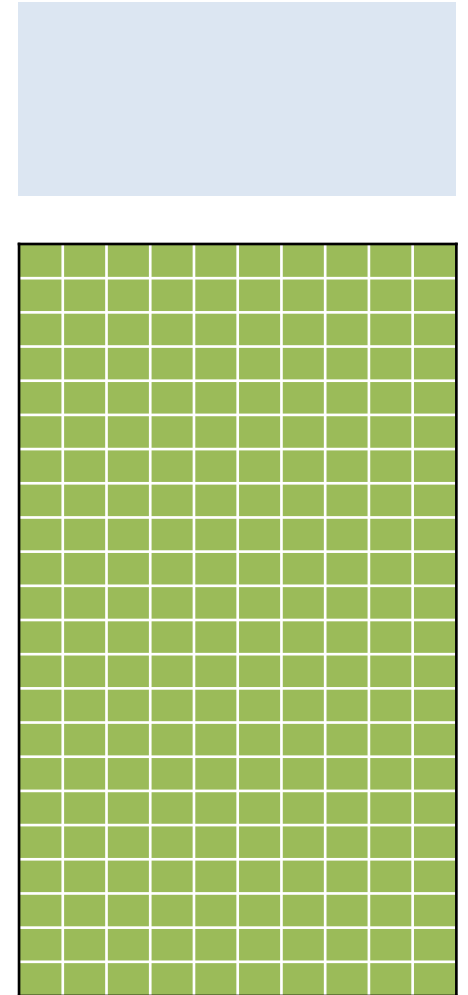
```
char fname[10];  
fname = "Seunggyu";
```

```
char fname[10];  
strcpy(fname, "Seunggyu");
```

```
char fname[10] = "Seunggyu";
```

```
fname[5] = 'G';
```

```
fname[5] = '\0';
```



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해

변수의 대입

● 대입 연산자 (Assignment Operator): =

- 정수형 변수 a와 실수형 변수 b를 선언하고 변수에 값을 넣는 방법

```
int a;
float b;
a = 100;
b = 123.45;
```

==

```
int a = 100;
float b = 123.45;
```

- 변수 선언 후에 값을 대입해도 되고 변수 선언과 동시에 값을 대입해도 됨
- 변수 선언과 동시에 값을 대입하면 프로그램이 좀 더 간결해진다.

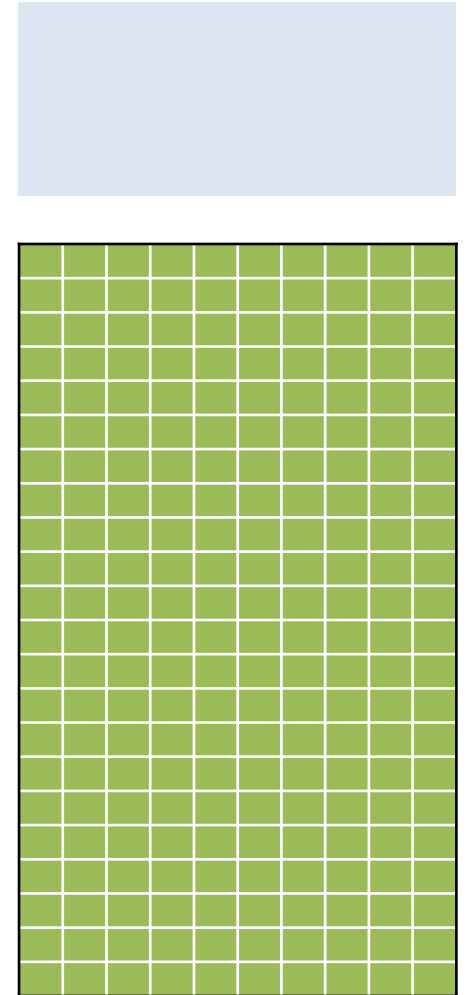
- 변수 a, b가 모두 정수형 변수라면 한 줄로 표현 가능

```
int a;
int b;
a = 100;
b = 200;
```

==

```
int a = 100, b = 200;
```

- 지정된 데이터 형식과 다른 값을 변수에 대입하는 경우



RAM 또는 가상 메모리
같이 그림을 그려보자

변수의 대입 - 대입 연산자 사용 시 주의 사항

● 일치하지 않는 자료형의 대입

- 의도적으로 사용가능하기는 하다.

```
int a;  
a = 123.45;
```

- 실수(123.45)를 대입해도 변수가 정수형이므로 소수점 아래가 떨어져 정수(123)만 저장된다.

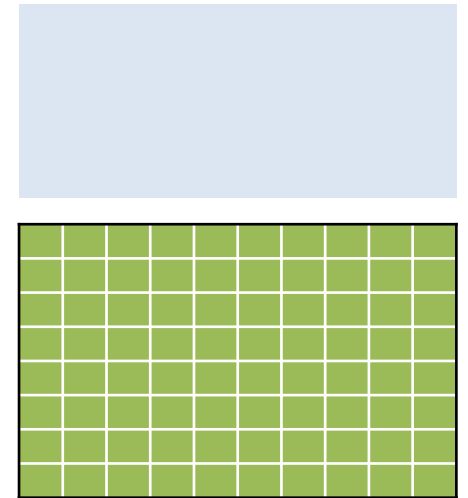
- 오른쪽 명령은 제대로 수행될까?
- 되어도 안되어도 수정해야 한다.

```
double b;  
b = 200;
```

```
b = 200.0;
```

● 변수에 변수의 값을 대입하는 것도 가능

- 변수명만을 사용하는 연산이 어떤 기능을 한다고 볼 수 있을까?



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해

변수의 연산

● 숫자형의 이항 5칙 연산 + 대입 연산

- $+$, $-$, $*$, $/$, $\%$, $=$
- (다음주차 강의에 더 자세히 다룰 예정)

● 5칙 연산의 과정과 변수 사용의 이해

```
int a;
a = 100 + 100;
```

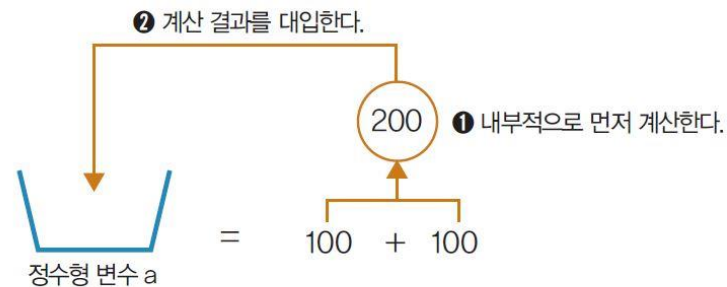


그림 3-11 숫자끼리의 계산 결과를 대입하는 방식

```
int b;
b = a + 100;
```

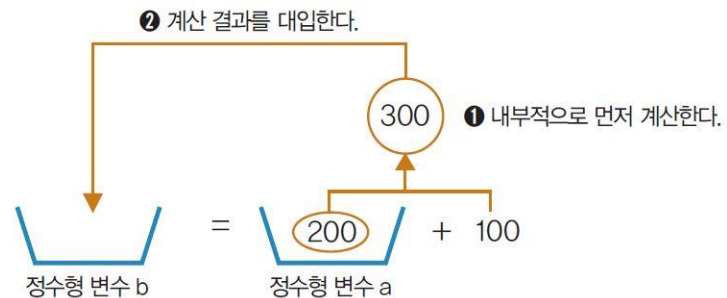
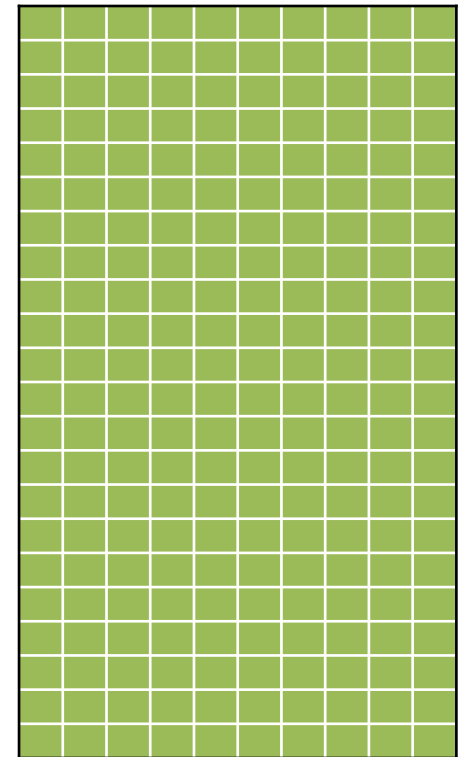
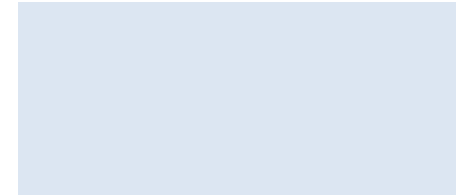


그림 3-12 변수와 숫자의 계산 결과를 대입하는 방식



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해 변수의 연산

● 5칙 연산의 과정과 변수 사용의 이해

```
int c, d;  
a = b = c = d;
```

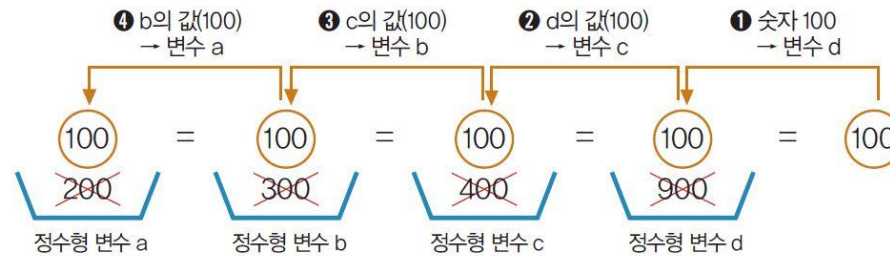


그림 3-13 연속된 값의 대입 방식

```
a = a + 200;
```

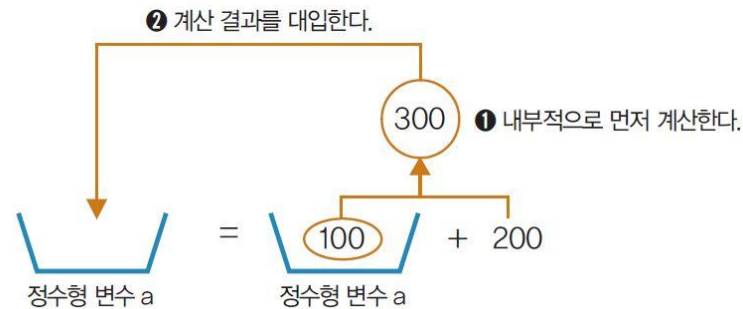
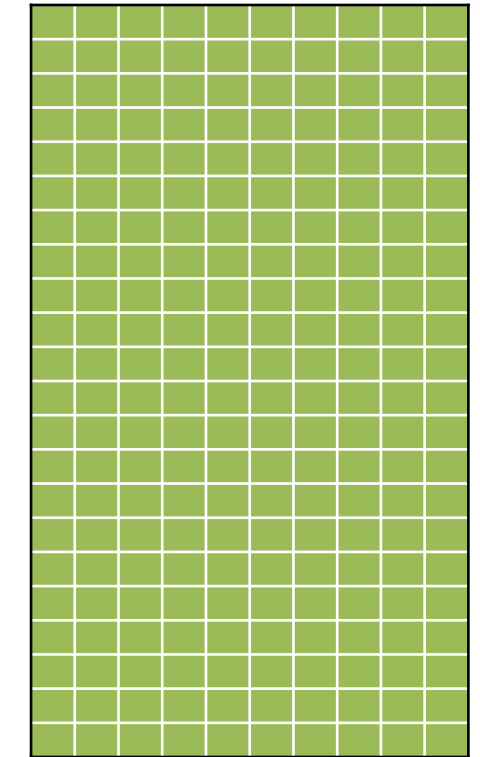


그림 3-14 자신의 값을 다시 계산 결과에 대입하는 방식



RAM 또는 가상 메모리
같이 그림을 그려보자

1 변수의 이해

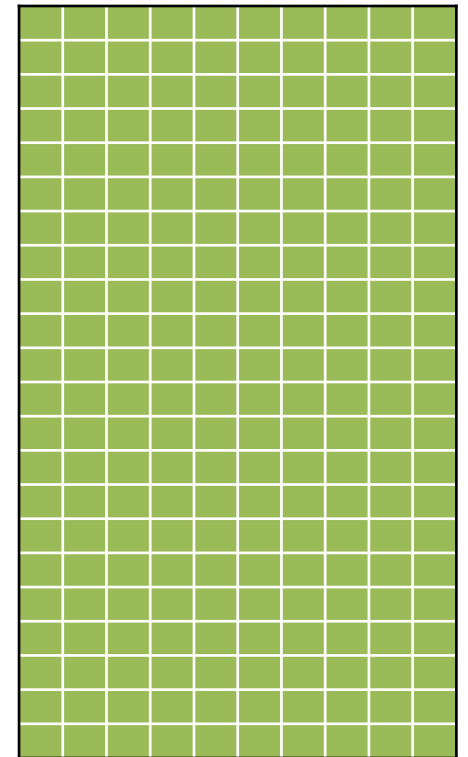
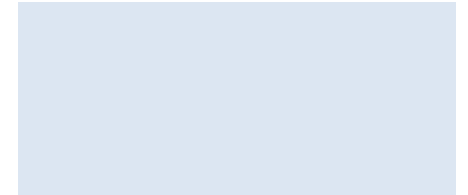
변수의 연산

● 숫자형의 이항 5칙 연산 + 대입 연산

- =
- +, -, *, /, %, =
- 다음주차 강의에 더 자세히 다룰 예정)

● 5칙 연산의 특징

- 연산 좌, 우 항의 형식이 다를 경우 크기가 더 큰 형식으로 변환된다.
 - 정수 + 정수 이면
- 연산 좌, 우 항의 형식이 다르고 크기가 같으면 실수형이 우선된다.



RAM 또는 가상 메모리
같이 그림을 그려보자

printf()와 scanf()의 개요

● %스타일 포매팅과 {} 스타일 포매팅 – 기억나니?

- printf에서
- f는 format의 약자

% 스타일 포매팅

```
print("%d %s" % (10000, 'lelecoder'), end='')
```

```
print("%d %s" % (10000, 'lelecoder'), end='') 10000 lelecoder
```

정수 *문자열*

{} 스타일 포매팅

```
print("number={0}, name={1}".format(10000, 'lelecoder'))
```

```
print("number={0}, name={1}".format(10000, 'lelecoder')) number=10000, name=lelecoder
```

● C언어의 printf와 scanf는 % 스타일 포매팅을 사용

printf("%d %s", 10000, lelecoder)

scanf("%d %s", &inp, lelecoder)

- 해석 1: "정수형 변수를 10진수로 하나 + 공백 + 문자열"을 출력한다.
첫번째 변수의 정수는 10000, 두번째 변수의 문자열은 lelecoder변수에 저장된 값
- 해석 2: "정수형 데이터를 10진수로 하나 + 공백 + 문자열"을 사용자가 입력한다.
그 중 공백 앞의 정수형 데이터는 inp에 저장하고, 공백 뒤의 문자열은 lelecoder에 저장

printf()의 기본 형태

- **printf() 함수: 화면(모니터)에 큰따옴표(" ") 안의 내용을 출력하라는 의미**

- 표준 출력: 모니터, PC 스피커로 출력하는 것 등 PC의 필수 출력 요소를 사용하는 것

```
printf("안녕하세요?");
```

실행 결과 ▶

안녕하세요?

- **printf는 문자만 좋아해**

```
printf("100");  
printf("%d", 100);
```

실행 결과 ▶

100
100

- 첫 번째 printf("100")의 결과인 100은 '글자 100'.
- printf()는 큰따옴표 안에 있는 것을 무조건 문자(열)로 취급한다.
- 두 번째 printf("%d", 100)의 결과인 100은 '숫자 100(백)'. 서식(%d)이 지정된 숫자는 그대로 숫자의 의미를 지니기 때문이다.

- **변수는 %, %뒤의 값에 따라 변수형을 결정**

printf()의 기본 형태

- %로 변수가 시작됨을 표시, 알파벳으로 변수가 끝남을 표시

- %뒤의 값에 따라 변수형과 출력 포맷을 결정

표 3-1 printf()의 대표적인 서식

서식	서식값의 예	설명
%d, %x, %o	10, 100, 1234	정수(10진수), 정수(16진수), 정수(8진수)
%f 또는 %lf	0.5, 1.0, 3.14	실수(소수점이 있는 수)
%c	'a', 'b', 'F'	문자(한 글자이며 '로 둘러싸야 한다)
%s	"안녕", "abcdefg", "a"	문자열(한 글자 이상이며 "로 둘러싸야 한다)

- d: decimal (10진수), x: hexadecimal (16진수), o: octet (8진수)
- f: float (소수점), lf: long float (긴 소수점)
- c: character (문자), s: string (문자열)

- 정수형을 출력할 경우 %와 d 사이에 다른 무언가가 들어갈 수 있음을 의미

- 우선은 %와 알파벳 하나를 사용하는 기본 형태를 익혀보자!
- 4가지 기본 형태: %d, %f, %c, %s

printf()의 기본 형태

● Reading Comprehension

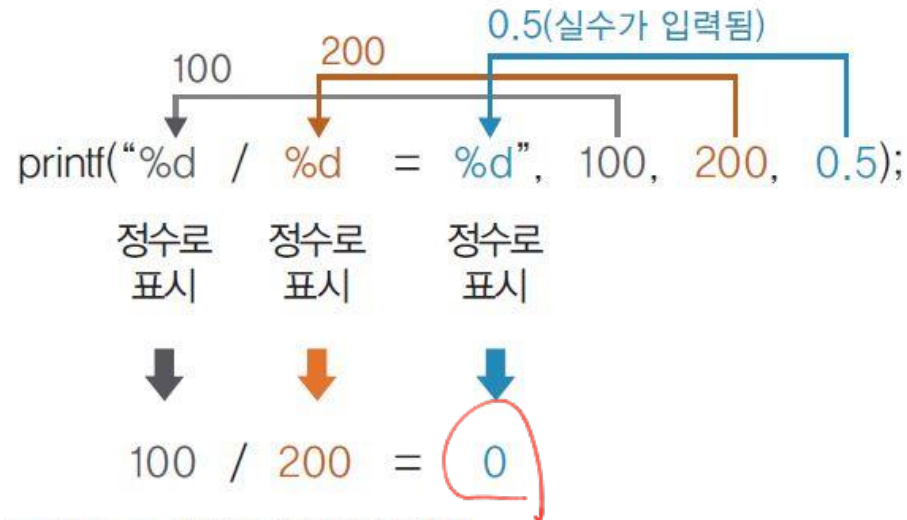


그림 3-2 서식과 숫자의 불일치

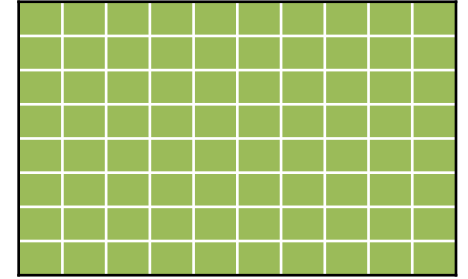
● 100 / 200의 계산 결과가 0.5가 아닌 0인 이유

- 0.5는 실수 (소수점이 있는 수)
- 출력 형식은 정수 (소수점이 없는 수)
- 즉, 0.5에서 소수점 아래를 버리고 정수형의 값만 출력한단

scanf()의 기본 형태

- %로 변수가 시작됨을 표시, 알파벳으로 변수가 끝남을 표시

- 기본 서식의 사용법은 printf와 동일
- 단, 주의 사항: 입력의 어절은 공백으로 구분한다.
 - 공백: 스페이스, 탭, 엔터



RAM 또는 가상 메모리 같이 그림을 그려보자

- Int x = 9;로 주어졌을 때, 변수명 x의 의미는?

● 생각하고 독해해보자

- 오른쪽과 같은 코드라면
- X가 의미상 옳은 코드인가?
- 방향성을 기억하자.

```
int x;  
scanf("%d", x);
```

● scanf의 진짜 사용 방법

```
int x;  
scanf("%d", &x);
```

```
float y;  
scanf("%f", &x);
```

```
char ch;  
scanf("%c", &ch);
```

```
char str[10];  
scanf("%s", str);
```

```
char str[10];
scanf("%c", &ch[5]);
```

printf()의 기본 형태

● Reading Comprehension

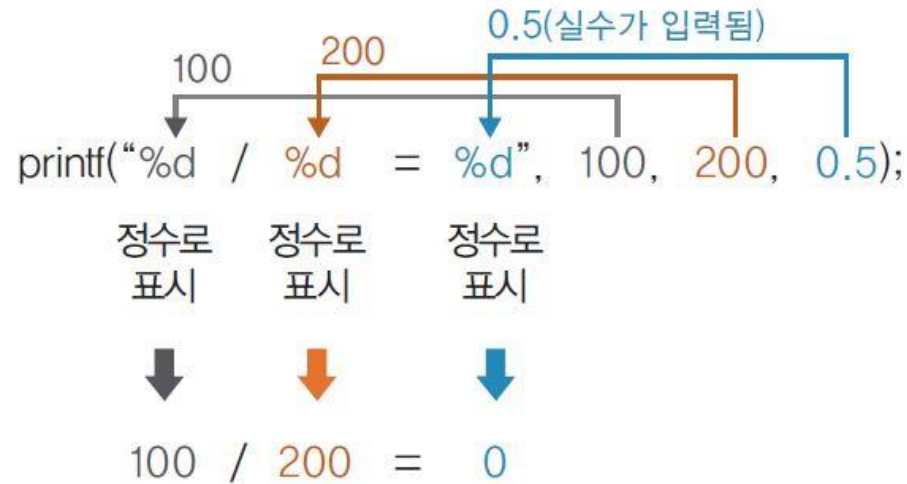


그림 3-2 서식과 숫자의 불일치

● 100 / 200의 계산 결과가 0.5가 아닌 0인 이유

- 0.5는 실수 (소수점이 있는 수)
- 출력 형식은 정수 (소수점이 없는 수)
- 즉, 0.5에서 소수점 아래를 버리고 정수형의 값만 출력한당

printf()의 서식 지정

정수형 데이터의 서식 지정

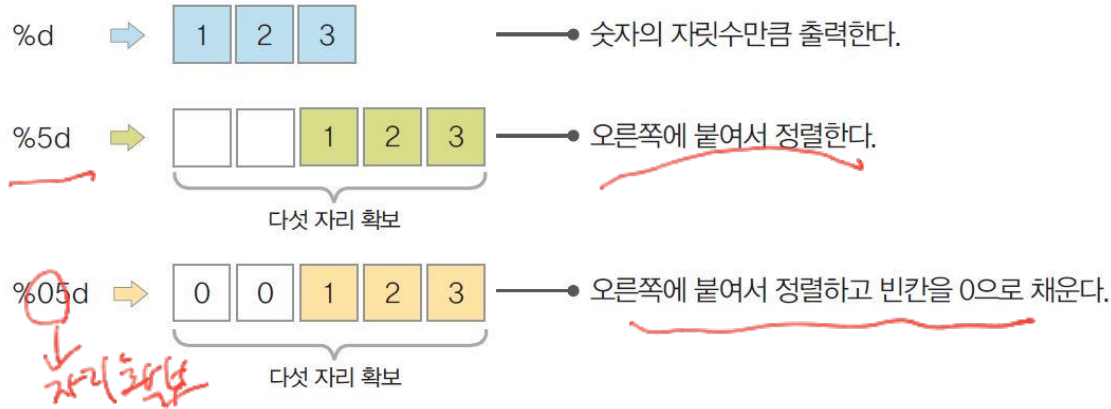


그림 3-3 정수형 데이터 서식 지정

실수형 데이터의 서식 지정

- 두 번째의 `%7.1f`는 소수점을 포함한 전체 자리인 일곱 자리 확보한 후
- 소수점 아래는 한 자리만 차지한다는 의미.
- 소수점 위 정수 부분은 다섯 자리가 된다.

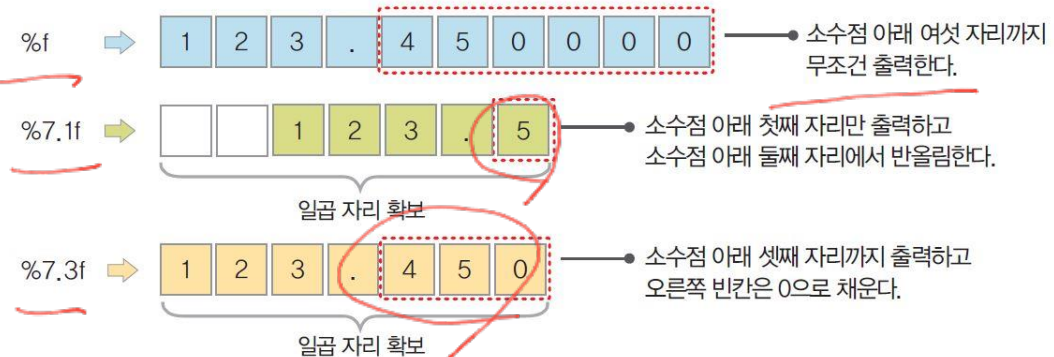


그림 3-4 실수형 데이터 서식 지정

printf()의 서식 지정

● 문자열의 서식 지정

- 길이 지정시 우측 정렬된다.

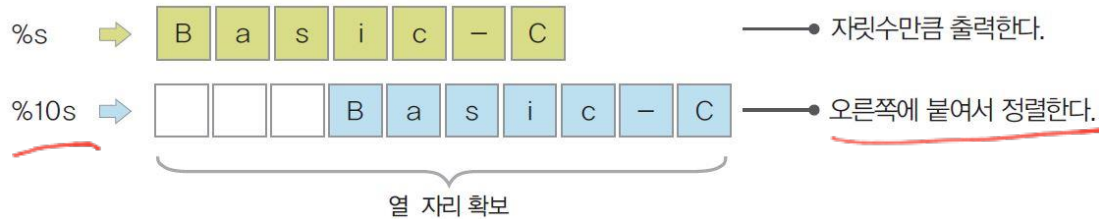


그림 3-5 문자열형 데이터 서식 지정

● 다양한 기능의 서식 문자

- 서식 문자는 앞에 반드시 '\' 또는 'w'가 붙는다. 이런 문자를 탈출(escape) 문자라고도 한다.

표 3-2 다양한 서식 문자

서식 문자	역할	비고
\n	새로운 줄로 이동한다.	Enter를 누른 효과와 동일하다.
\t	다음 탭으로 이동한다.	Tab을 누른 효과와 동일하다.
\b	뒤로 한 칸 이동한다.	Back Space를 누른 효과와 동일하다.
\r	줄의 맨 앞으로 이동한다.	Home을 누른 효과와 동일하다.
\a	'뽵' 소리를 낸다.	
\\	\를 출력한다.	
\'	'를 출력한다.	
\"	"를 출력한다.	

입력의 정규 표현식

● Regex란?

- 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어
- 문자를 수식의 개념을 적용하여 표현하는 일반적인 방법
- 문자열을 나타내는 일반적인 수식 (규칙)

● Regex의 필요성

- 숫자만 입력하게 하고 싶는데 사용자가 돌발적으로 문자열을 입력하면 프로그램은 어떻게 되나?
- if-else로 걸러낸다? try-catch로 걸러낸다?
- 좋은 방법이지는 하지만...

● Regex의 활용사례

- 숫자만 입력 받고 싶을 때
- 특정 문자만 입력 받고 싶을 때 → ex) 0123456789ABCDEFabcdef
- 특정 문자 앞까지만 입력 받고 싶을 때, 공백으로 구분되는 scanf의 규칙을 초월할 수 있다!
- 기타 등등

scanf에서 문자열의 정규식 사용

● scanf의 기본 규칙

- 변수에 들어갈 데이터를 공백으로 구분
- %d, %s, %f 등 %로 시작되는 모든 데이터에 해당됨

● scanf의 변수 표시는 s대신에 대괄호로 Regex를 사용한다

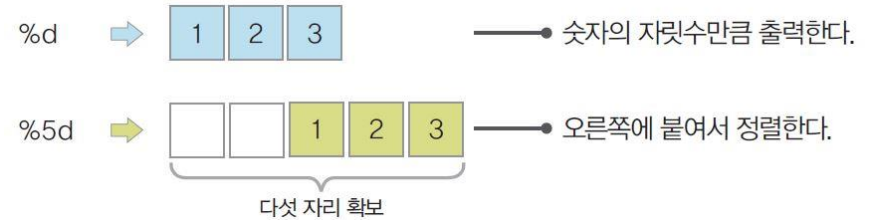
- %s에서 → %[....]
- []안에 규칙을 써넣는 방법으로 문자열의 규칙을 지정
- 문자의 나열: 나열된 문자만 받는다.
 - Ex) [abc]: a, b, c만 허용한다. Ex) [a-c]: a에서 c 사이의 문자만 허용
 - Ex) [0-9]: 0에서 9사이의 문자만 허용 Ex) [0-9A-Za-z]: 0에서 9사이의 문자, 영문 알파벳을 허용
- ^ : ^ 뒤의 문자가 입력되기 직전의 문자들을 입력
 - Ex) %[^123]: 123이라는 문자열이 입력되기 전까지 입력 받은 문자를 허용
 - Ex) %[^Wn]: 줄바꿈 문자 즉, 엔터 입력 전까지의 모든 문자

3 Regular Expression

scanf에서 서식의 사용

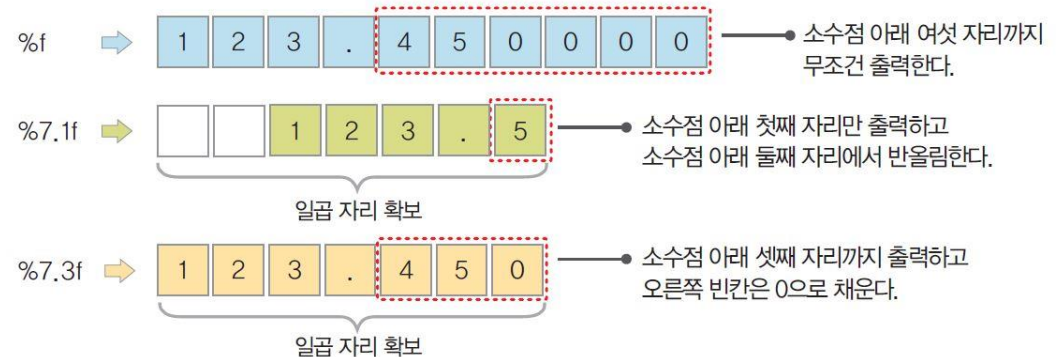
● scanf의 정수 서식 사용

- %d: 정수를 입력
- %5d: 입력 받은 정수에서 최대 다섯자리를 취함



● scanf의 실수 서식 사용

- %f: 실수를 입력
- %7.1f: 입력 받은 실수에서 최대 7자리, 소수점 아래 한자리를 취함. 소수점 포함
- %.3f: 입력 받은 실수에서 최대 자리수는 상관없지만, 소수점 아래는 세자리를 취함



● scanf에서 문자 서식 사용

- %5s: 다섯자리 문자 (공백 허용 안함)