

2. c 언어 기본 원소/ 데이터유형

Autumn 2019

Today

1. 지난 시간 실습 복습
2. 변수와 수리연산
3. 간단한 실습2
4. 데이터유형

[복습] 100-50 출력하기

■ .c 확장자를 갖는 파일 작성하고 실행시키기

1. Visual Studio를 실행한다.
2. 메뉴에서 [파일]-[열기]-[프로젝트/솔루션] 선택 후 앞서 작업했던 'C:\AEN2Codes\First' 폴더에서 'First.sln'을 선택한다.
3. 왼쪽 [솔루션 탐색기]의 프로젝트 이름(현재 First) 아래 '소스 파일' 폴더에서 마우스 오른쪽 버튼을 클릭하여 [추가]-[새 항목]을 선택한다.
4. [새 항목 추가] 창에서 'C++ 파일(.cpp)' 선택한 뒤, 이름에 'First.c' 입력하고, <추가>를 클릭한다.
5. 코드 편집 창에 100과 50을 빼고 출력하는 프로그램을 코딩한다.
6. 틀린 글자가 없는지 확인 후 메뉴의 [파일]-[모두 저장]으로 저장한다.
7. [빌드]-[솔루션 빌드] 선택하거나 단축키 [F7]로 코드를 빌드하고 [ctrl+F5]로 실행시킨다.

코드 분석 1/2

```
01 #include <stdio.h>
```

---- 무조건 들어가는 부분이다(표준 입/출력 라이브러리에 대한 정보 포함).

```
02
```

```
03 int main()
```

---- 프로그램의 시작 부분으로 항상 고정되어 있다.

```
04 {
```

---- 항상 { }로 묶어준다.

```
05     printf (" %d ", 100 - 50 );
```

---- 100-50의 값을 모니터에 출력하라는 의미이다.

```
06 }
```

- 첫 번째 줄의 `#include <stdio.h>` 구문은 컴파일러에게 입/출력 정보를 가지고 있는 “표준라이브러리”를 포함시키도록 한다.
 - 대부분의 C 코드 시작 부분에 사용된다.
 - (표준라이브러리 관련은 추후에 별도로 소개함)

코드 분석 2/2

- C 프로그램은 코드의 크기와 상관없이 항상 **변수**와 **함수**로 구성된다.
- 함수는 실행시키고자 하는 **명령들(문장)**을 포함하고 있음
 - 위의 예제에서는 **main** 함수가 있다.
 - 함수의 이름은 보통 자유롭게 지정할 수 있지만, **main** 이라는 이름을 갖는 함수는 특별하다.
 - **프로그램은 항상 main 함수에서부터 시작**하도록 되어있다. 따라서, 모든 프로그램은 **main** 함수를 가지고 있어야 한다.
 - 함수의 명령들은 항상 꺾은 괄호“{ }” 안에 선언되어야 한다.
- 함수들은 “**매개변수**”라고 부르는 것을 통해서 함수들끼리 데이터를 주고 받는다.
 - 매개변수는 함수이름 뒤의 괄호 안에 넣어주는데, 이 예제의 **main** 함수는 괄호 안이 비어 있으므로 **매개변수가 없는 함수**이다.
 - 위 예제의 **main** 함수는 하나의 명령 (`printf(“%d”, 100-50)`) 만 포함하고 있다.
 - 함수는 매개변수를 넣는 괄호와 함께 이름을 적으면 실행된다.
 - 예제의 **main** 함수는 **함수 printf** 를 매개변수 “%d”와 100-50으로 호출하고 있다.

c언어 구조

- c언어는 기본적으로 다음의 파트들로 구성된다.

- 전처리 명령어
- 함수
- 변수
- 실행문장 & 표현
- 주석

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     printf(" %d ", 100 - 50 );
```

```
06 }
```

c언어의 기본 문법

- c에서의 구분기호

- c 문법은 여러가지 구분기호로 이루어짐

`printf("Hello, World! \n");` 는 다섯개의 구분기호로 구성되어있음

- 세미콜론 (;)

- 문장을 종결할 때 항상 필요함

- 주석 (// 또는 /**/)

- 사용자가 알아보기 쉽게 쓰는 구문들로 컴파일러는 무시하는 구간

- 식별자 (Identifiers)

- 변수 또는 함수를 나타내는 이름

- 예약어 (keywords)

- c에서 예약한 구문들 (int, double 등)

Today

1. 지난 시간 실습 복습
2. 변수와 수리연산
3. 간단한 실습2
4. 데이터유형

두번째 예제 코드

```
01 #include <stdio.h>
02 /* addition code */ --- 주석 (컴파일러가 무시하는 부분)
03 int main()
04 {
05     int a, b; ---계산할 두 수를 저장할 변수 a, b와
06     int result; ---결과를 넣을 변수 result를 선언한다
07
08     a=100; ---a에 100을, b에 50을 넣는다.
09     b=50;
10
11     result = a + b; ---a와 b를 더한 결과를 result에 넣는다.
12     printf(" %d + %d = %d \n", a, b, result); ---모니터에 a, b, result를 출력한다.
13
14     result = a - b; ---a에서 b를 뺀 결과를 result에 넣는다.
15     printf(" %d - %d = %d \n", a, b, result); ---모니터에 a, b, result를 출력한다.
16
17     result = a * b; ---a와 b를 곱한 결과를 result에 넣는다.
18     printf(" %d * %d = %d \n", a, b, result); ---모니터에 a, b, result를 출력한다.
19
20     result = a / b; ---a를 b로 나눈 결과를 result에 넣는다.
21     printf(" %d / %d = %d \n", a, b, result); ---모니터에 a, b, result를 출력한다.
22 }
```

변수 선언하기

- 변수는 프로그램이 실행되는 동안 필요한 값들을 저장하고 있다.
- C언어에서 모든 변수들은 사용되기 전에 선언한다.
 - 일반적으로 함수의 시작부분에서, 다른 명령어를 정의하기 전에 선언한다.
 - “선언”은 변수의 유형과 변수 이름들을 나열한다.
 - 예를 들어, 정의한 변수의 유형 `int`는 나열한 변수들의 값이 정수여야 한다는 것을 뜻한다. (소수부를 갖는 실수 값으로 선언하려면 `float` 유형을 사용한다.)
 - `int n;`
 - `float phi;`

변수 초기화

- 변수를 초기화하지 않으면, 컴파일러는 변수가 기본값을 갖는 것으로 가정한다.
- 변수 초기화는 할당(assign) 연산자를 통해서 한다.
프로그래밍에서의 = 은 같다가 아닌 할당한다는 뜻임
 - `a = 100;`
 - `b=50;`
- 변수를 선언할 때 초기화할 수 있다.
 - `int a = 100;`
 - `float phi = 1.618;`
- 여러 변수들의 선언과 초기화를 한번에 할 수 있다.
 - `int a = 0, b= 4;`

수리연산

- x 와 y 라는 변수가 있다고 가정해보자.

- $x + y$,
- $x - y$
- $x * y$
- x / y

- 간단한 명령:

- $y = x + 3 * x / (y - 4);$
- 세미콜론 ; 으로 문장을 끝내야 한다.

- 연산하면서, 동시에 할당할 수 있다.

- $x += y$
- $x -= y$
- $x /= y$
- $x \% = y$

두번째 예제 코드 분석 1/2

- 1행의 `/**`은 주석으로 컴파일러가 무시하는 부분이다. 이는 여러 줄로 써도 되고, 공백이 들어가도 된다.
- 변수란 값을 저장하는 그릇(또는 방)이라고 생각할 수 있다.
- 5행과 6행에서 변수(그릇) 3개를 준비한다.



그릇 이름: a



그릇 이름: b



그릇 이름: result

- 8행과 9행에서 a 그릇에는 100, b 그릇에는 50을 넣는다.



그릇 이름: a

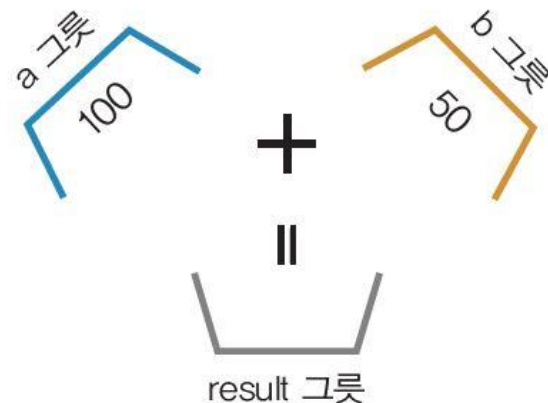


그릇 이름: b



그릇 이름: result

- 11행에서 a 그릇 값과 b 그릇 값을 더한 결과를 result 그릇에 넣는다.



두번째 예제 코드 분석 2/2

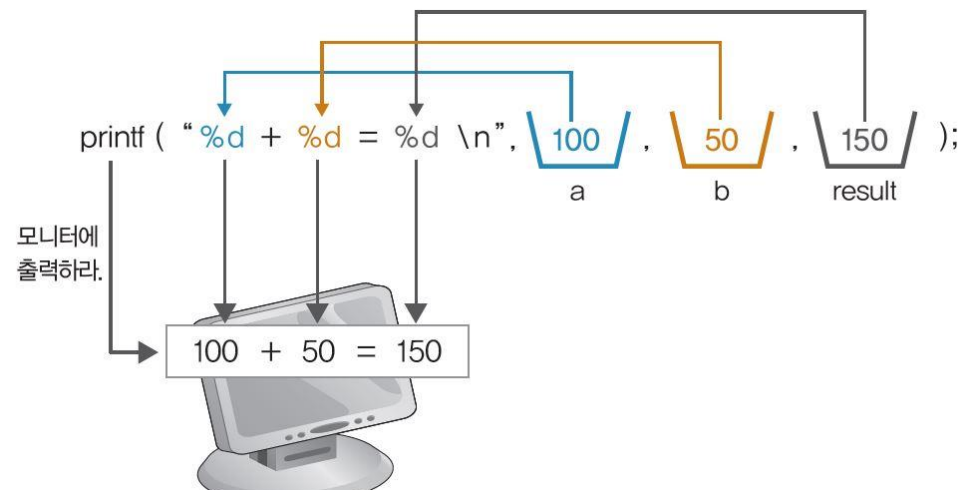
- 12행의 **printf()**는 결과를 출력하기 위해 사용하는 함수로 괄호 안의 매개변수를 모니터에 출력하라는 의미이다.
 - 함수 printf의 괄호 안에 “ ”으로 구성된 매개변수는 **문자열**을 의미한다.
 - 문자열에 있는 **\n**은 c 문법으로 다음 줄로 넘어가도록 하는 문자열이다.
 - \n 이 없으면 다음 줄로 넘어가서 출력하지 않는다.
 - 다음과 같이 출력했을 때 결과는? `printf(“%d + %d = %d”, a, b, result);`
 - **% 기호**는 %의 자리를 다른 값으로 대체하겠다는 것을 뜻한다.
 - %기호는 뒤이어 오는 매개변수와 쌍으로 이루어져야 한다.
 - **%d**는 이 자리에 뒤에 오는 매개변수 값을 **정수**로 대체하겠다는 것을 뜻한다.
 - **%f**는 매개변수 값을 **실수**로 대체한다.
 - 숫자를 사용해서 실수부의 자리수를 표현할 수 있다.

printf(“%f”, 1234.123456)

%f - 실수값

%.3f - 실수의 소수부를 3자리만 출력

- **%d**를 썼을 때의 결과는?
- **%c**는 매개변수 값을 **문자하나**로 대체한다.
문자 하나는 **작은따옴표**로 둘러싸야한다.
- **%s**는 매개변수 값을 **문자열**로 대체한다.
문자열은 **큰따옴표**로 둘러싸야한다.



printf()

- printf 함수는 C 언어의 일부가 아니다.
- C 프로그램에서 접근할 수 있는 ANSI 표준라이브러리로 이용한 함수이다.
- C 언어 자체만으로 입/출력을 할 수 없다.
- <https://en.cppreference.com/w/c/header>

Today

1. 지난 시간 실습 복습
2. 변수와 수리연산
3. 간단한 실습2
4. 데이터유형

[실습1] 화씨-섭씨 변환하기

- 다음의 공식 $^{\circ}\text{C} = \left(\frac{5}{9}\right) * (^{\circ}\text{F} - 32)$ 을 이용해서 다음의 값을 출력하도록 하시오.
- 다음의 화씨온도와 섭씨 온도는 동일함을 확인하세요.
 - main 함수에서 작성할 것
 - 변수명은 `fahr`, `cels` 두 개를 쓸 것
 - 변수 선언은 정수형으로 하고, 각각 0으로 초기화 할 것
 - 화씨 온도($^{\circ}\text{F}$)에 대한 섭씨온도($^{\circ}\text{C}$)를 `printf` 함수를 사용해서 출력할 것

$^{\circ}\text{F}$	$^{\circ}\text{C}$
0	-17
20	-6
40	4
60	15
80	26
100	37

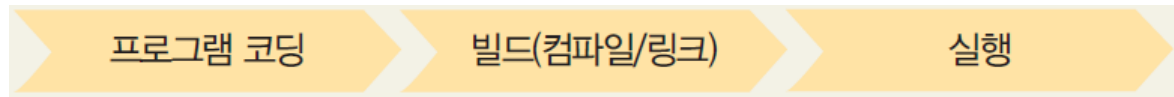
[실습2] 화씨-섭씨 변환하기

- 섭씨온도(°C)에 대한 화씨 온도(°F) 를 [실습1]과 같은 과정으로 출력하시오.

°C	°F
-17	0
-6	20
4	40
15	60
26	80
37	100

Summary

- c 프로그램 작성 순서



- c 언어 기본 원소: 함수와 변수

- 함수

- ① 여러 개의 다양한 이름을 갖는 함수를 가질 수 있지만, **main** 이라는 이름을 갖는 함수는 반드시 한 개 존재해야 한다.
- ② 프로그램은 **main** 함수부터 실행하게 된다.
- ③ 함수는 매개변수를 통해서 데이터들을 전달할 수 있고, 이는 함수명 뒤 괄호안에 입력한다. 괄호안이 비어있는 경우에는 데이터 전달을 하지 않는 것을 뜻한다.

- 변수

- ① 변수는 사용되기 전에 반드시 선언해야 한다.
- ② 선언 후 초기값을 줘야 하며, 선언과 동시에 값을 할당할 수 있다.
- ③ 변수는 값을 저장하는 그릇과 비슷한 개념으로, 변수에 한 번 들어간 값은 다른 값이 들어오기 전까지 그대로 유지된다.

- **printf()** 함수: 매개변수를 화면에 출력할 때 사용하는 함수이다.

- **scanf()** 함수: 키보드로 값을 입력받을 때 사용하는 함수이다.

- ① 변수에 값을 입력받으려면 반드시 변수 앞에 **&** 기호를 붙여야 한다.

Today

1. 지난 시간 실습 복습
2. 변수와 수리연산
3. 간단한 실습2
4. 데이터유형

변수

- 변수와 상수는 프로그램에서 다루는 기본 데이터 유형
- 변수 만들기 규칙
 - 문자와 숫자로 만들 수 있음
 - Underscore “_”도 문자로 간주함 (긴 문자 쓸 때 가독성을 줌)
 - 하지만, underscore “_”로 시작하는 변수명은 권장하지 않음
 - 대/소문자는 구별됨
 - 예를 들어, x 와 X 는 다른 변수임
 - 일반적으로 C언어에서 변수명은 소문자, 상수명은 대문자로 표기함
 - C 언어에서 예약한 단어들은 변수명으로 사용할 수 없음
 - 예를 들어, int , float 는 변수명이 될 수 없음
 - 프로그램 목적하는 바와 관련된 단어로 정의하는 것을 권장함

기본 데이터유형 및 크기

■ 기본 데이터유형

- char: 문자형 or 정수형¹
- int: 정수형
- float: 실수형
- double: 큰 실수형

■ 위 데이터유형들에

short, long, unsigned, signed의 수식어를 붙여서 쓸 수 있다.

- 이 수식어들을 쓰는 이유는 일반적으로 정수형의 크기를 다르게 주기 위함이다.
 - short: 16 bits
 - long: 32 bits
- 다음은 문자형이나 정수형의 크기를 다르게 주기 위함이다.
 - unsigned: 항상 0이상의 값을 나타냄, 크기는 $0 \sim 2^n$ ($n = \text{bit}$)
 - signed: $-2^{n-1} \sim 2^{n-1}$
- 수식어를 쓰지 않는 경우에는 컴파일하는 기계에 따라 변수의 크기가 달라질 수 있다.

1 문자형 데이터의 정수

■ char

- 한 글자를 뜻하는 문자형

■ 아스키 코드 (**ASCII: American Standard Code for Information Interchange**)

- 키보드에 있는 모든 기호에 할당된 부호
- 1바이트만 표기가능해서 현재는 **유니코드**를 많이 사용함 (**UTF-8**)
- 숫자 0은 48, 대문자 A는 65, 소문자 a는 97 등은 알아두면 편함
- C 언어에서는 숫자를 문자로도 표현함.

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

==

```
char ch = 97;
```

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

데이터 크기 - 진수

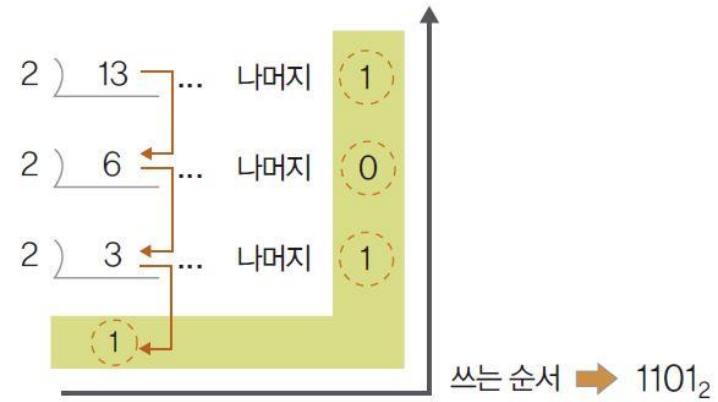
- 2진수 : 표현 가능한 숫자 0, 1
- 10진수 : 표현 가능한 숫자 0~9
- 16진수 : 표현 가능한 숫자 0~9, A~F
- 진수를 구분하여 표기하는 방법
 - 2진수 : 10_2
 - 10진수 : 10_{10}
 - 16진수 : 10_{16}

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

진수 변환 연습 - 10진수와 2진수

■ 10진수를 2진수로 변환

- 10진수를 계속 2로 나뉘 **나가면서** 그 몫과 나머지를 구함



■ 2진수를 10진수로 변환

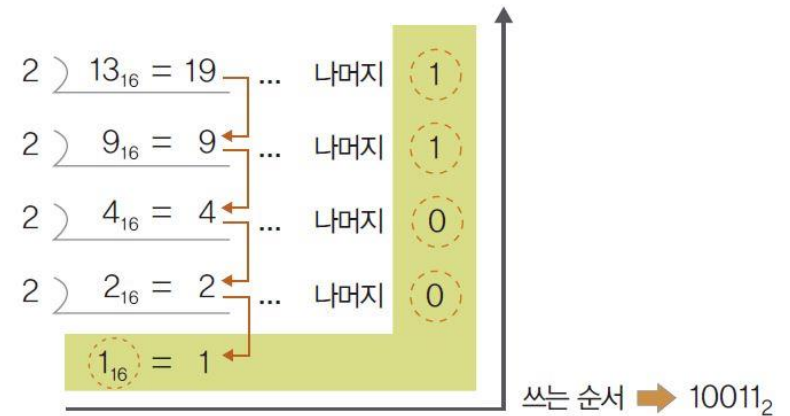
- 2진수의 맨 뒤에서부터 각 자리에 해당하는 가중치인 $2^0, 2^1, 2^2, \dots$ 을 곱한 후 각 자리의 결과를 모두 합한다.

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
+ <div style="border-top: 1px solid black; width: 100%; height: 10px; margin: 5px 0;"></div>									
147									→ 10진수

진수 변환 연습 - 16진수와 2진수

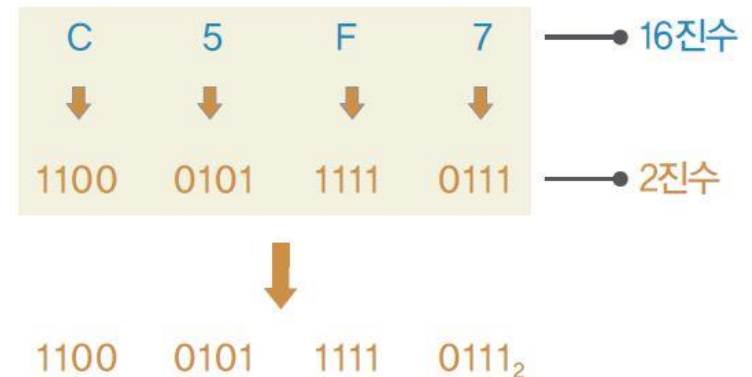
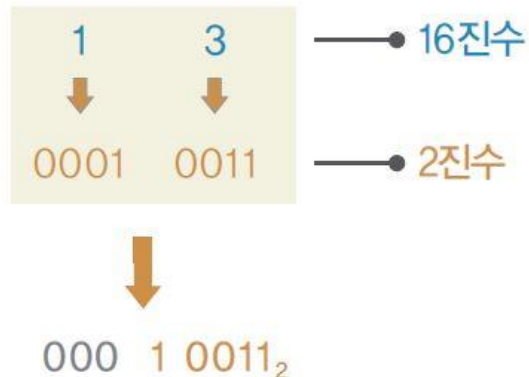
■ 16진수를 2진수로 변환법 1

- 16진수를 10진수로 만든 뒤, 계속 2로 나눠 **나가면서** 그 몫과 나머지를 구함



■ 16진수를 2진수로 변환법 2

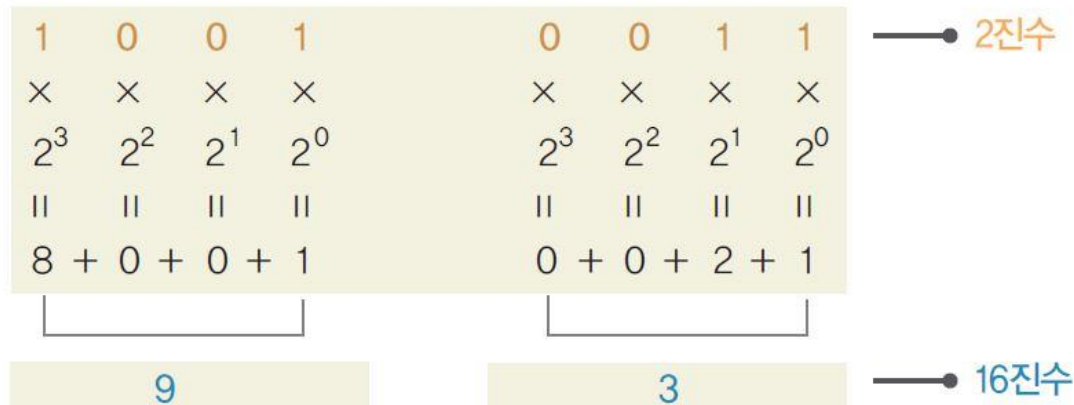
- 16진수의 한 자리마다 2진수 4자리를 부여함



진수 변환 연습 - 16진수와 2진수, 10진수

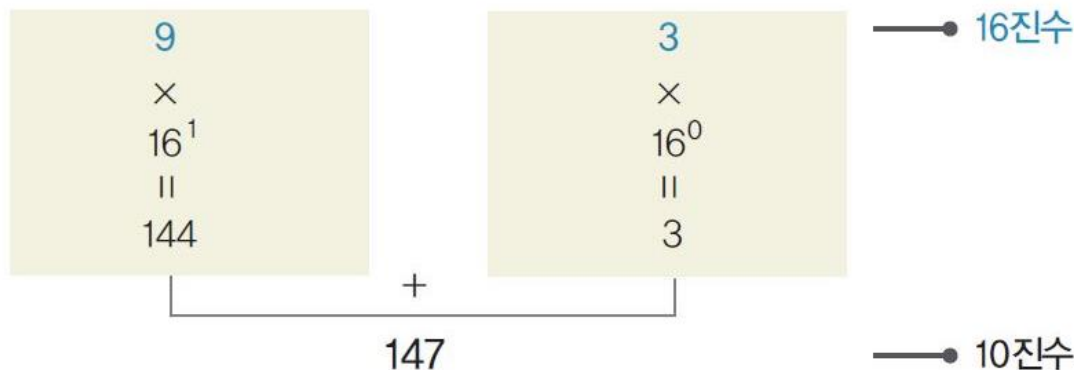
■ 2진수를 16진수로 변환

- 2진수의 4자리가 16진수의 한 자리임



■ 16진수를 10진수로 변환





- 각 자리에 해당하는 가중치인 $16^0, 16^1, 16^2, \dots$ 을 곱해서 만들어줌



데이터 크기 - 비트

- 컴퓨터에서 표현하는 가장 작은 단위로, 0과 1만 존재.
- 비트는 전기스위치와 비슷한 개념으로 0(OFF)과 1(ON)만 존재함.

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

데이터 크기 - 바이트

- 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~...

데이터유형 별 크기

데이터유형	의미	크기	값의 범위
char	문자형 (정수형)	1바이트	$-2^7 \sim 2^7 - 1$
unsigned char	문자형 (부호없는 정수)	1바이트	$0 \sim 2^8 - 1$
int	정수형	4바이트	$-2^{31} \sim 2^{31} - 1$
unsigned int	부호없는 정수	4바이트	$0 \sim 2^{32} - 1$
short int (short)	작은 정수	2바이트	$-2^{15} \sim 2^{15} - 1$
unsigned short	부호없는 작은 정수	2바이트	$0 \sim 2^{16} - 1$
long int (long)	큰 정수	4바이트	$-2^{31} \sim 2^{31} - 1$
unsigned long	부호없는 큰 정수	4바이트	$0 \sim 2^{32} - 1$
float	실수형	4바이트	소수점 6자리 표현
double	큰 실수형	8바이트	소수점 15자리 표현

[실습1-1] printf() 복습하기

- 다음 코드를 보고 결과를 예상해보시오.
- 코딩해서 그 결과를 비교해보시오.

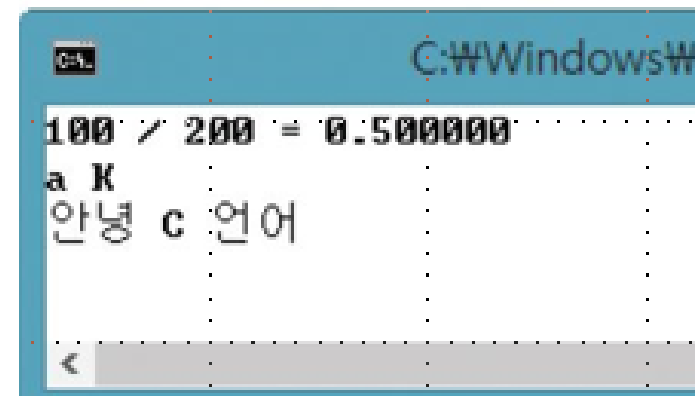
```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("100+100");
06     printf("\n");
07     printf("%d", 100+100);
08     printf("\n");
09 }
```

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("%d", 100, 200);
06     printf("\n");
07     printf("%d %d", 100);
08     printf("\n");
09 }
```

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("%d / %d = %d", 100, 200, 0.5);
06     printf("\n");
07 }
```

- 다음 결과가 나오도록 빈 칸을 적절한 코드로 채우시오.

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("%d / ____①____ = ____②____ \n", 100, 200, 0.5);
06     printf(" ____③____ %c \n", 'a', 'K');
07     printf("%s %s \n", "안녕", "c 언어");
08 }
```



[실습1-2] printf()의 다양한 기능

- 서식을 출력할 수 있다.
 - 서식을 나타낼 때는 반드시 역슬래시 '\ '가 붙는다.
 - 이를 escape 문자 (탈출문자) 라고도 한다.

서식 문자	역할
\n	새로운 줄로 이동한다.
\t	다음 탭으로 이동한다.
\b	뒤로 한 칸 이동한다.
\r	줄의 맨 앞으로 이동한다.
\a	'뽁' 소리를 낸다.
\\	\를 출력한다.
\'	'를 출력한다.
\"	"를 출력한다.

- 다음의 서식을 출력하는 연습을 해보자.

```
01: #include <stdio.h>
02:
03: int main()
04: {
05:     printf("\n줄 바꿈\n연습 \n");
06:     printf("\t탭키\t연습 \n");
07:     printf("이것을\x뎌어씁니다 \n");
08:     printf("\a\a\a뽁소리 3번 \n");
09:     printf("글자가 \"강조\"되는 효과 \n");
10:     printf("\\\t\\\t\\\t\\\t\\\t\\\t 역슬래시 세개 출력 \n");
11: }
```

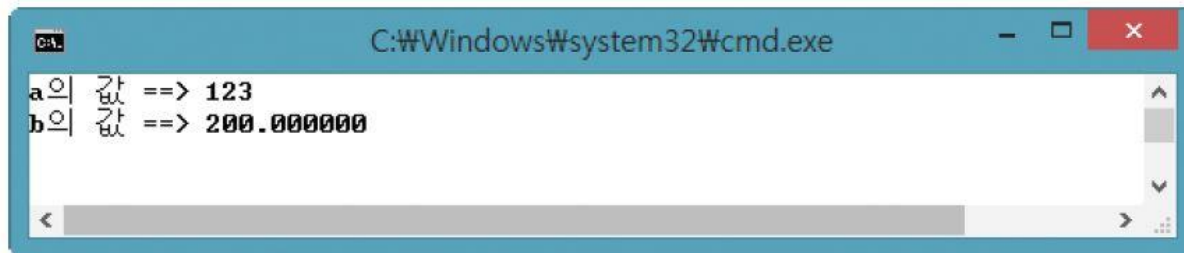

[실습1-2] printf()의 다양한 기능 코드 해석

- 6행의 \t는 [Tab]에 지정된 만큼의 간격을 벌린다.
- 7행에서는 ‘이것을’이라는 문구를 출력했다가 \r을 만나서 다시 처음 위치로 돌아가 ‘덮어씹니다’를 출력하므로 ‘이것을’이 지워져서 보이지 않는다.
- 8행에서는 ‘뽀’ 소리를 낸다.
- 9행에서는 강조라는 단어 앞뒤에 “ ”를 출력한다.“ 를 출력하기 위해서는 \를 입력해야 한다.
- 10행은 역슬래시(\) 하나를 출력하려면 \를 두 번 입력해야 한다는 것을 보여 준다.

[실습2] 변수의 이해 - [기본 3-7] 변수에 값 대입 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a;          ---- 정수형 변수 a를 선언한다.
06     float b;        ---- 실수형 변수 b를 선언한다.
07
08     a = 123.45;      ---- 정수형 변수에 실수를 대입한다. ← 바람직하지 않다.
09     b = 200;        ---- 실수형 변수에 정수를 대입한다. ← 바람직하지 않다.
10
11     printf ("a의 값 == > %d \n", a);
12     printf ("b의 값 == > %f \n", b);
13 }
```

실행 결과 ▼

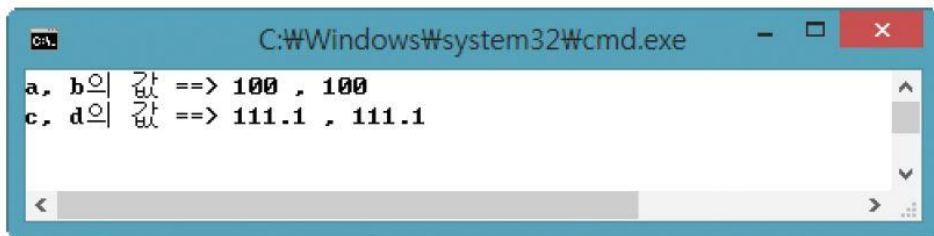


```
C:\Windows\system32\cmd.exe
a의 값 ==> 123
b의 값 ==> 200.000000
```

[실습2] 변수의 이해 - [응용 3-8] 변수에 변수 대입 예 ①

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;      ---- 정수형 변수 2개를 선언한다.
06     float c, d;    ---- 실수형 변수 2개를 선언한다.
07
08     a = 100;       ---- a에 정수 100을 대입한다.
09     ____①____     ---- b에 a 값을 대입한다.
10
11     c = 111.1f;    ---- c에 실수 111.1을 대입한다.
12     ____②____     ---- d에 c 값을 대입한다.
13
14     printf ("a, b의 값 ==> %d , %d \n", a, b);
15     printf ("c, d의 값 ==> %5.1f , %5.1f \n", c, d);
16 }
```

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
a, b의 값 ==> 100 , 100
c, d의 값 ==> 111.1 , 111.1
```

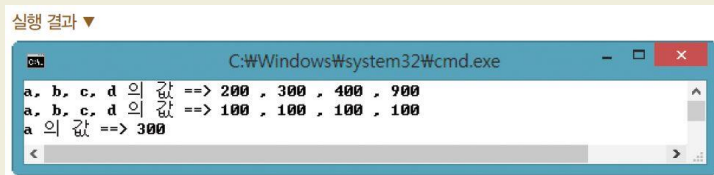
자판: a = b 1, c = d 2

[실습2] 변수의 이해 - [응용 3-9] 변수에 변수를 대입하는 예 ②

```

01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b, c, d;
06
07     a = 100 + 100;          --- a에 두 수의 계산 결과를 대입한다.
08     b = a + 100;           --- b에 변수와 수의 계산 결과를 대입한다.
09     c = a + b - 100;       --- c에 변수와 수의 계산 결과를 대입한다
10     ①                      --- d에 a, b, c의 덧셈 결과를 대입한다.
11     printf ("a, b, c, d 의 값 == > %d, %d, %d, %d \n", a, b, c, d);
12
13     ②                      --- a, b, c, d에 모두 100을 대입한다(한 문장으로 처리한다).
14     printf ("a, b, c, d 의 값 == > %d, %d, %d, %d \n", a, b, c, d);
15
16     a = 100;
17     a = a + 200;           --- a에 자신의 a 값과 200을 더한 값을 다시 대입한다
18     printf ("a 의 값 == > %d \n", a);
19 }

```


$$\text{예시 1} \quad d = a + b + c; \quad \text{예시 2} \quad a = b = c = d = 100;$$