
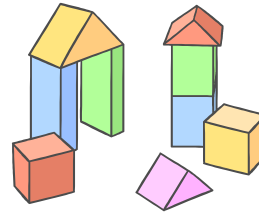


# 제 3장 C프로그램 구성요 소

# 이번 장에서 학습할 내용

- 
- \* 주석
  - \* 변수, 상수
  - \* 함수
  - \* 문장
  - \* 출력 함수 printf()
  - \* 입력 함수 scanf()
  - \* 산술 연산
  - \* 대입 연산

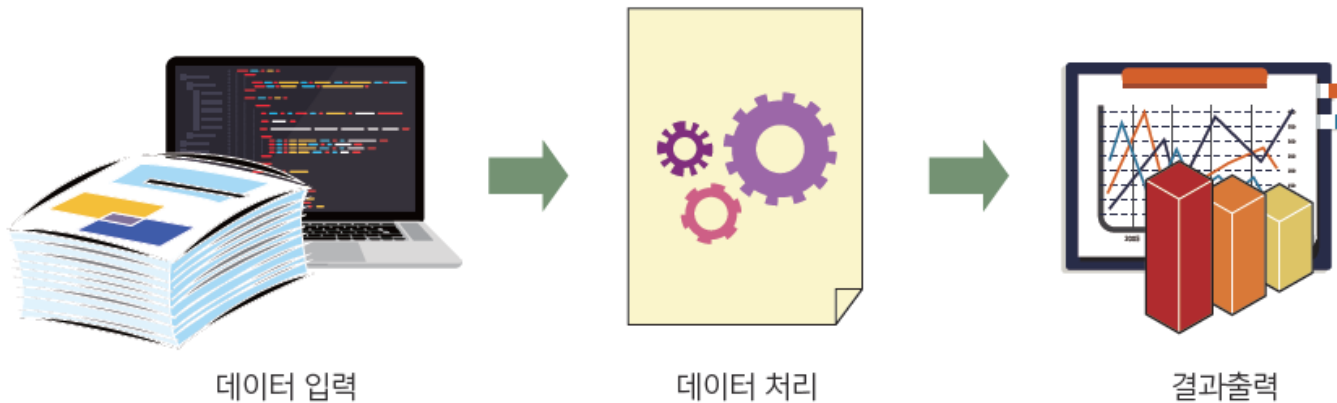


이번 장에서는  
C프로그램을  
이루는  
구성요소들을  
살펴봅니다.



# 일반적인 프로그램의 형태

- 데이터를 받아서(입력단계), 데이터를 처리한 후에(처리단계), 결과를 화면에 출력(출력단계)한다.



# 덧셈 프로그램 #1

add1.c

```
1  /* 두개의 숫자의 합을 계산하는 프로그램 */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int x;    // 첫 번째 정수를 저장할 변수
7      int y;    // 두 번째 정수를 저장할 변수
8      int sum;  // 두 정수의 합을 저장하는 변수
9
10     x = 100;
11     y = 200;
12
13     sum = x + y;
14     printf("두수의 합: %d", sum);
15
16     return 0;
17 }
```

주석

전처리기

변수 선언

함수

연산

두수의 합: 300

# 주석(comment)

```
/* 두 개의 숫자의 합을 계산하는 프로그램 */  
#include <stdio.h>  
int main(void)  
{  
    int x; // 첫 번째 정수를 저장할 변수  
    int y; // 두 번째 정수를 저장할 변수  
    int sum; // 두 정수의 합을 저장하는 변수  
    x = 100;  
    y = 200;  
    sum = x + y;  
    printf("두수의 합 : % d", sum);  
    return 0;  
}
```

주석은 코드를  
설명하는 글입니다.



## 2가지 주석 방법

```
/* 한 줄로 된 주석*/
```

```
/* 여러  
   줄로  
   된 주석*/
```

```
// 이 줄은 전체가 주석이다.
```

```
int x; // 여기서부터 줄의 끝까지가 주석이 된다.
```

# 주석의 중요성

- 다른 사람이 프로그램을 보았을 때, 주석이 있다면 훨씬 쉽게 프로그램의 내용을 알 수 있다. 많은 시간이 흘렀다면, 만든 사람이라고 하더라도 내용을 잘 기억할 수 없다.
- 좋은 주석은 코드를 반복하거나 코드를 설명하지 않는 것이다. 주석에는 코드를 작성한 의도를 명확히 나타내어야 한다.

# 주석 스타일

```
/*  
파일 이름: add.c  
설명   : 두수를 더하는 프로그램  
작성자 : 홍길동  
*/
```

```
/******  
  
* 파일 이름: add.c  
* 설명      : 두수를 더하는 프로그램  
* 작성자    : 홍길동  
  
*****/
```



# 들여쓰기

- **들여쓰기(indentation):** 같은 수준에 있는 문장들을 왼쪽 끝에서 몇 자 안으로 들여쓰는 것

```
#include <stdio.h>

int main(void)
{
    int x;
    int y;
    int sum;

    ...

    return 0;
}
```

빈줄을 넣어서 의미별로 구별을 한다.

프로그램의 의도를 주석으로 설명한다.

// 첫 번째 정수를 저장할 변수  
// 두 번째 정수를 저장할 변수  
// 두 정수의 합을 저장하는 변수

같은 내용의 처리이면 들여쓰기를 한다.

# 주석과 들여 쓰기가 없다면..

```
#include <stdio.h>
int main(void) {
int x; int y; int sum;    x = 100; y = 200; sum = x
+ y;    printf("두수의 합: %d", sum); return 0;
}
```

실행은 되지만 무슨  
처리를 하고 있는 프  
로그램인지 알기가 힘  
들고 또한 들여쓰기가  
안 되어 있어서 같은  
수준에 있는 문장들을  
구분하기 힘듭니다.



# 전처리기

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

- 외부 파일을 포함시키라는 의미의 전처리기
- #기호로 시작

- **stdio.h**는 표준 입출력에 대한 라이브러리 함수의 정의가 들어 있다.

# 전처리기

```
/* 첫번째 프로그램 */
```

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

```
int main(void)
{
    printf("Hello World!");
    return 0;
}
```

hello.c

```
// stdio.h
...
int printf(char *,...);
...
```

stdio.h

# 함수

- 함수(function): 특정 기능을 수행하는 처리 단계들을 괄호로 묶어서 이름을 붙인 것
- 함수는 프로그램을 구성하는 기본적인 단위(부품)

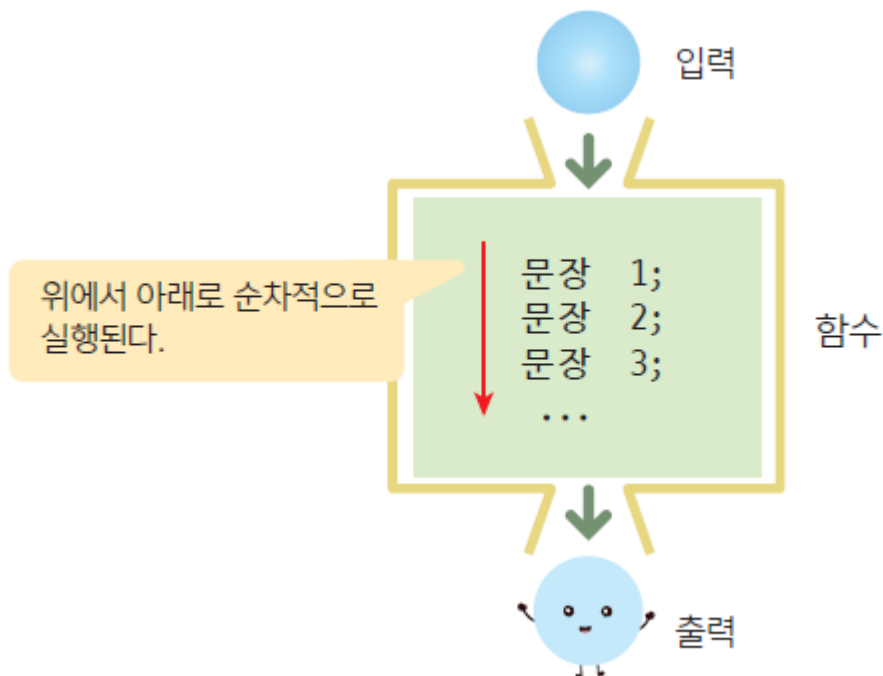
```
int main(void)
{
    ...
    ...
}
```



# 함수안에 들어 있는 것

Q) 그렇다면 함수 안에 들어 있는 것은 무엇인가?

A) 함수 안에는 함수가 처리하는 처리 단계(문장)들이 중괄호 안에 나열



# 함수의 구조

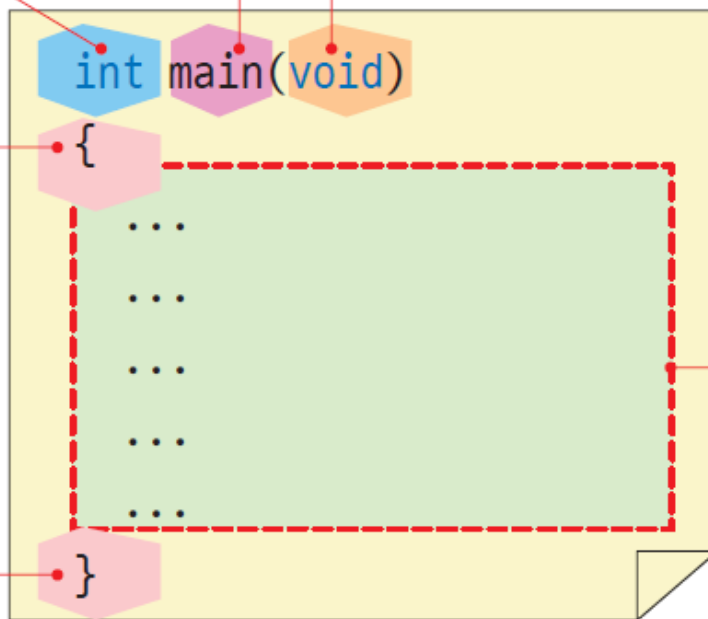
함수의 출력 타입

함수의 이름

함수의 입력 타입, void는 입력이 없다는 의미이다.

함수의 시작

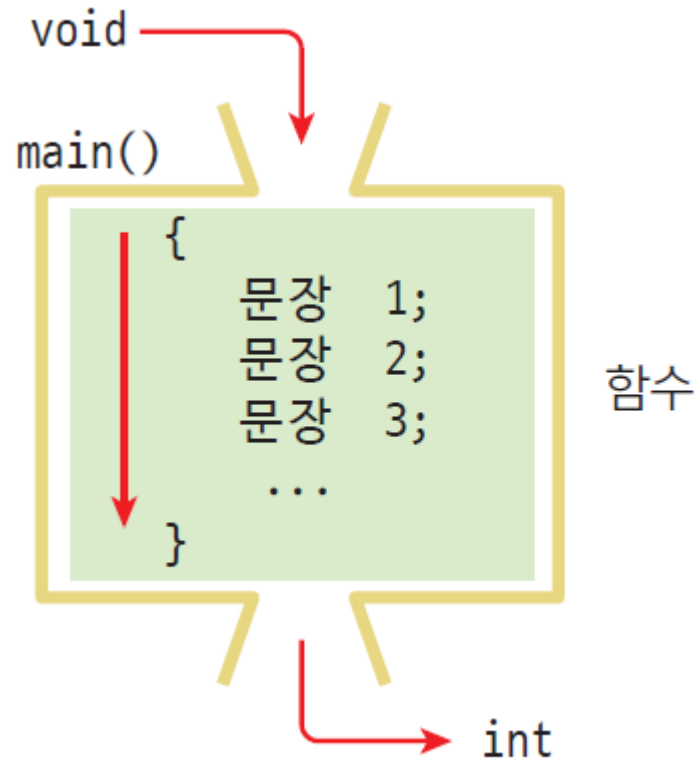
함수의 종료



함수의 몸체, 함수가 수행하는 작업에 해당하는 문장들이 들어간다.

# 함수

- 작업을 수행하는 문장은 함수 안에 들어가야 함





# return 문장

- return은 함수를 종료시키면서 값을 반환하는 키워드이다.
- 값을 반환하기 위해서는 return 다음에 반환값을 써주면 된다.

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

```
int main(void) {
```

```
    ...
```

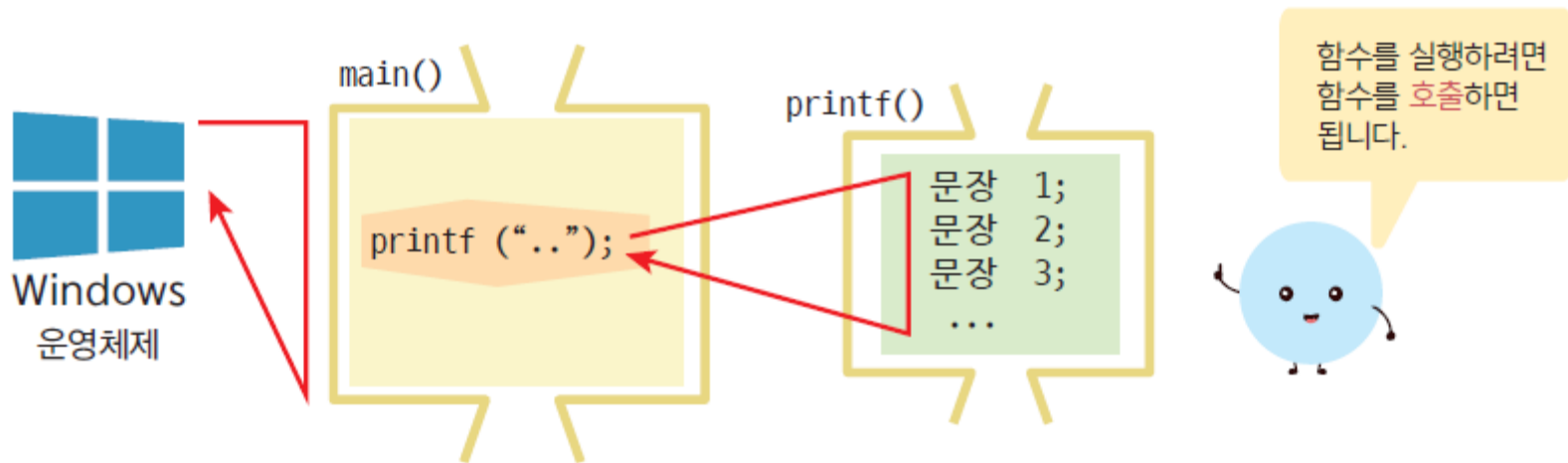
```
    ...
```

```
    return 0;
```

```
}
```



# main()은 누가 호출할까?



# 변수

- 프로그램이 사용하는 데이터를 일시적으로 저장할 목적으로 사용하는 메모리 공간

Syntax

변수 선언

예

자료형

int

x;

변수 이름

int

y;

int

sum;

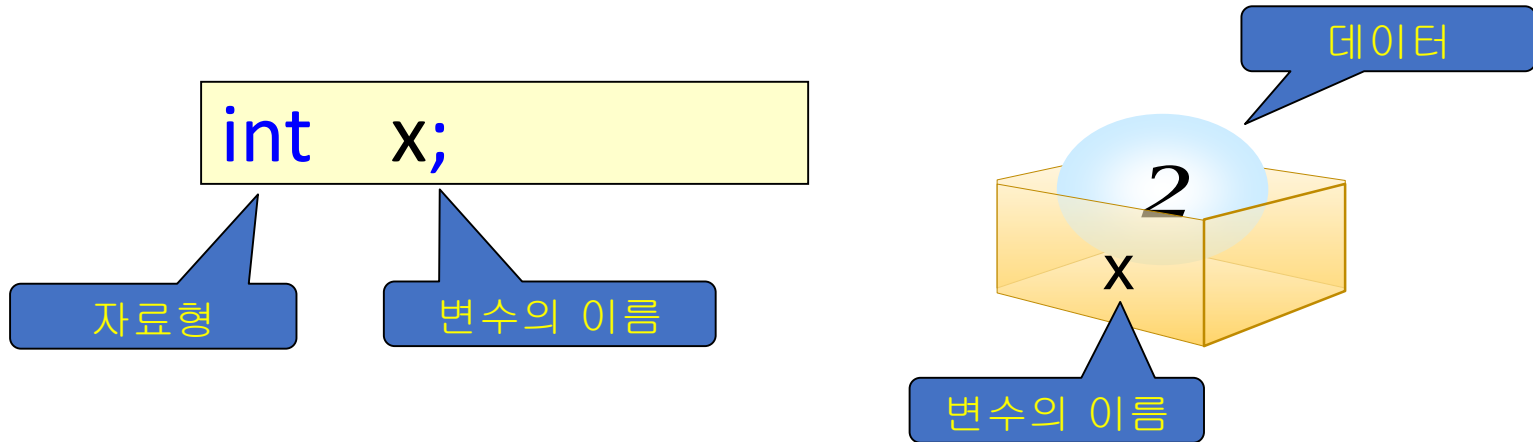
# 변수는 왜 필요한가?

- 변수는 데이터 값을 일시적으로 저장하는 역할을 한다.



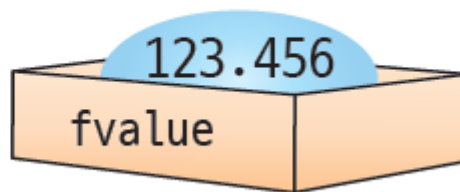
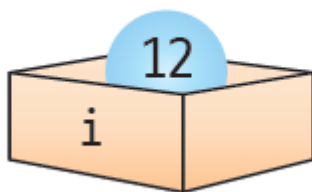
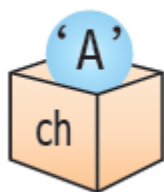
# 변수의 종류

- 변수는 데이터를 담는 상자로 생각할 수 있다.



# 변수의 종류

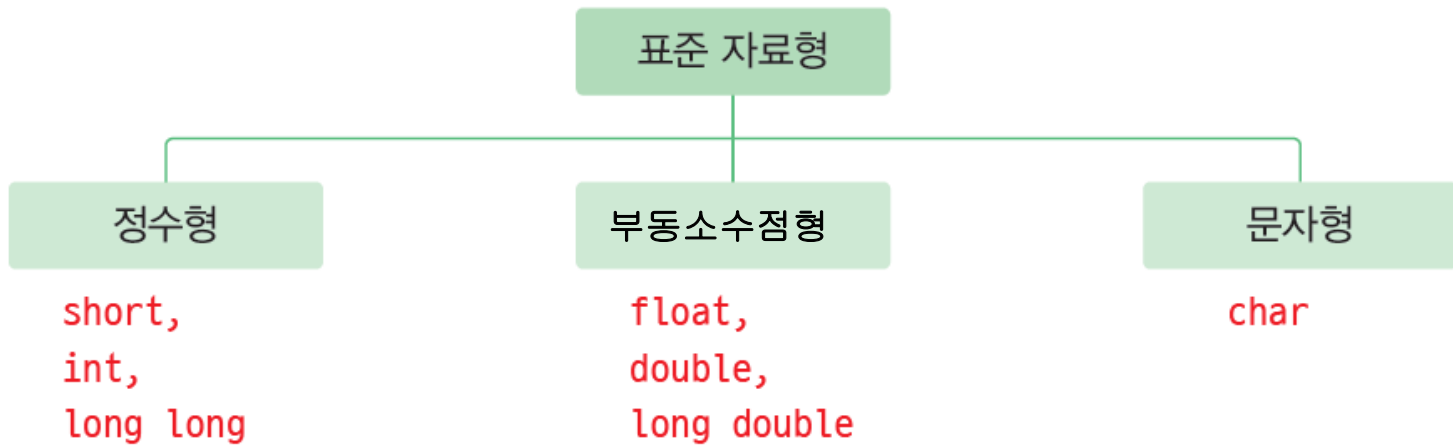
- 변수에는 데이터의 종류에 따라 여러 가지 타입이 존재한다.



Note: Not to Scale

# 자료형

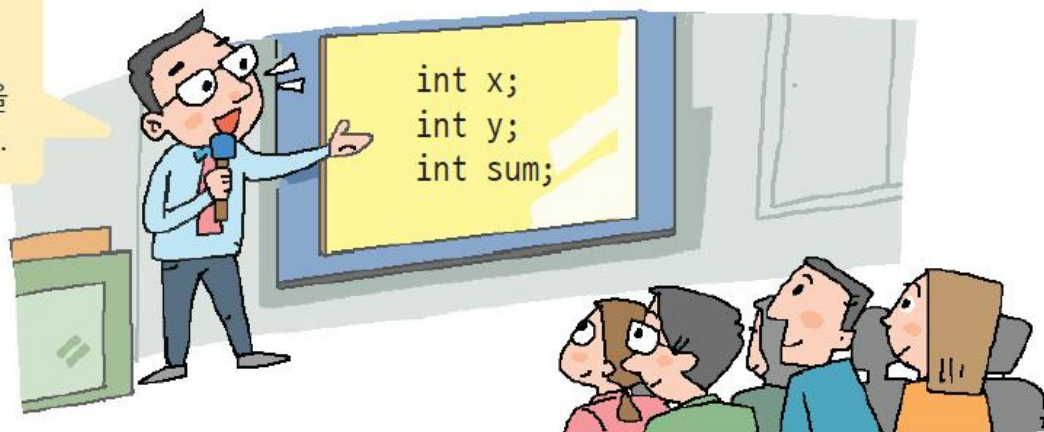
- 변수가 저장할 데이터가 정수인지 실수인지, 아니면 또 다른 어떤 데이터인지를 지정하는 것이다.
- 자료형에는 정수형, 부동소수점형(실수형), 문자형이 있다.



# 변수 선언

- 변수 선언: 컴파일러에게 어떤 타입의 변수가 사용되는지를 미리 알리는 것

지금부터 이  
프로그램에서  
사용될 변수들을  
소개하겠습니다.



컴파일러



# 변수 선언

자료형은 정수형

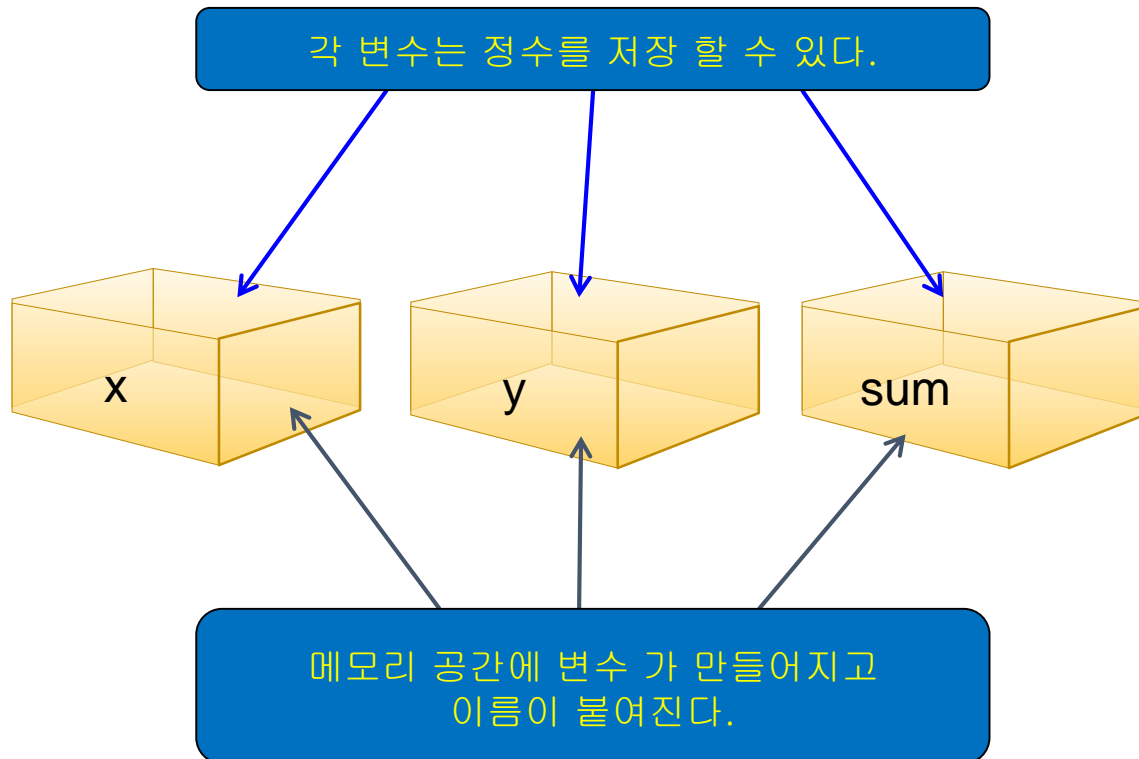
변수 이름

```
int x;  
int y;  
int sum;
```

```
// 첫 번째 정수를 저장하는 변수  
// 두 번째 정수를 저장하는 변수  
// 두 정수의 합을 저장하는 변수
```

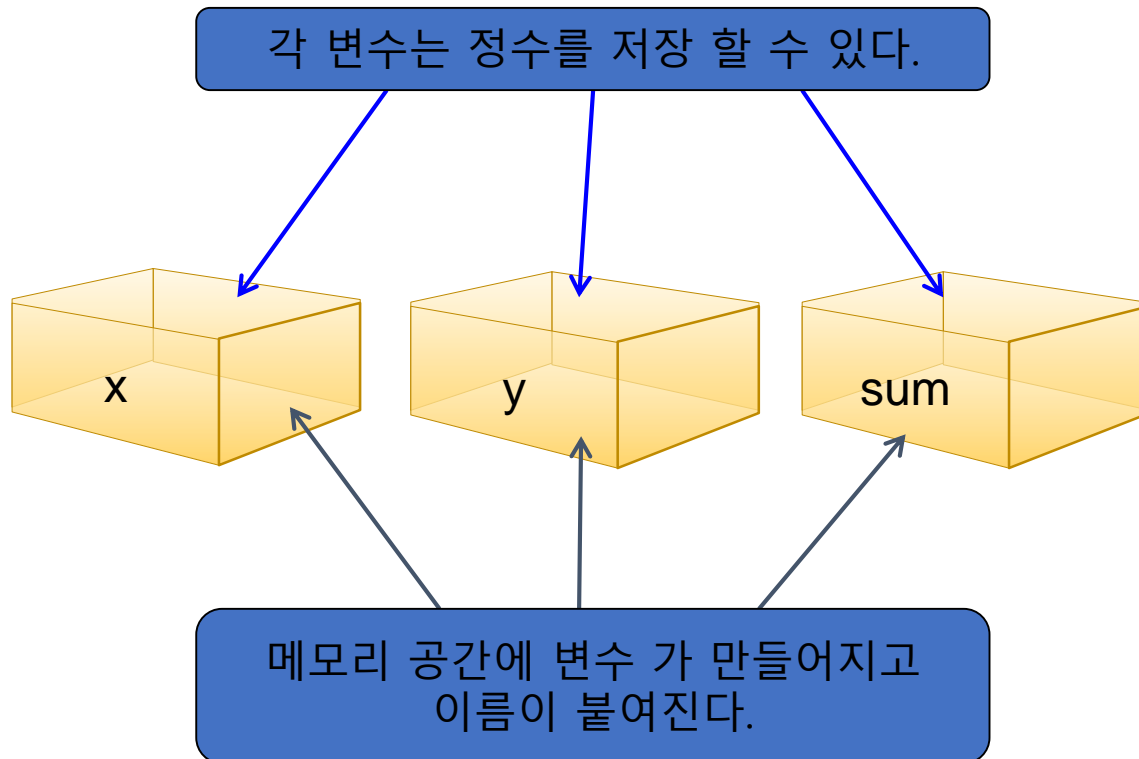
# 변수 선언

```
int x;    // 첫번째 정수를 저장하는 변수  
int y;    // 두번째 정수를 저장하는 변수  
int sum;  // 두 정수의 합을 저장하는 변수
```



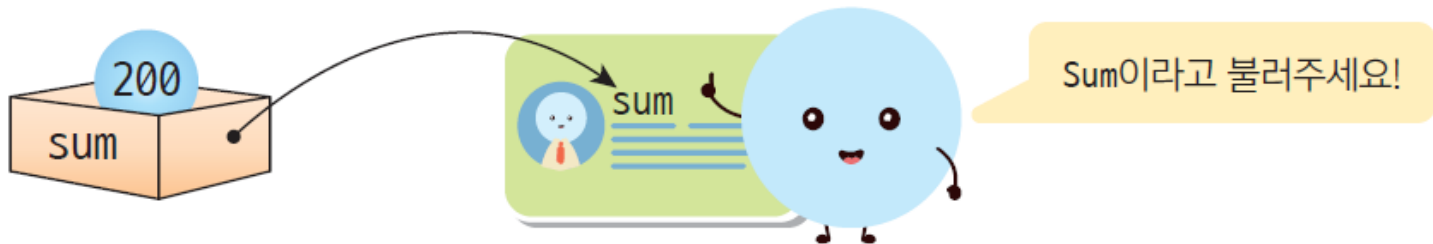
# 한 줄에 여러 개의 변수 선언

```
int x, y, sum;    //가능!!
```



# 변수의 이름

- 식별자(identifier): 변수의 이름은 프로그래머가 마음대로 지을 수 있지만 몇 가지의 규칙을 지켜야 한다. “홍길동”, “김영희” 등의 이름이 사람을 식별하듯이 변수의 이름은 변수와 변수들을 식별하는 역할을 한다.



# 변수의 이름

- 식별자 만드는 규칙

- 식별자는 영문자와 숫자, 밑줄 문자 \_로 이루어진다.
- 식별자의 중간에 공백이 들어가면 안 된다.
- 식별자의 첫 글자는 반드시 영문자 또는 밑줄 기호 \_이어야 한다. 식별자는 숫자로 시작할 수 없다.
- 대문자와 소문자는 구별된다. 따라서 변수 index와 Index, INDEX은 모두 서로 다른 변수이다.
- C언어의 키워드와 똑같은 식별자는 허용되지 않는다.

The diagram illustrates the naming rules for variables and functions in C. It shows a code snippet with two annotations:

- An orange arrow points from the text "함수 이름" (Function Name) to the word "sub" in the function signature "int sub(void)".
- Another orange arrow points from the text "변수 이름" (Variable Name) to the variable "x" in the declaration "int x;" inside the function body.

```
int sub(void)
{
    int x;
}
```

# 키워드

- 키워드(keyword): C언어에서 고유한 의미를 가지고 있는 특별한 단어 예약어(reserved words) 라고도 한다.

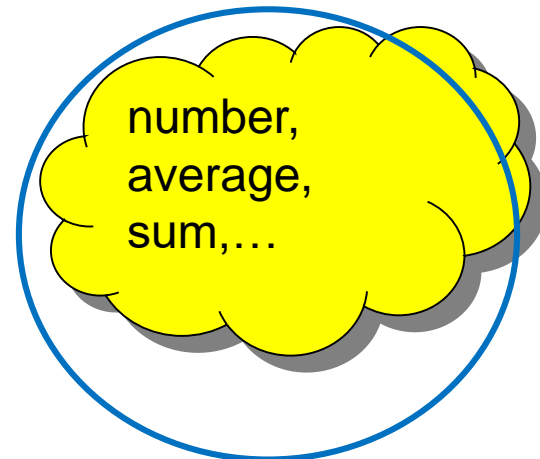
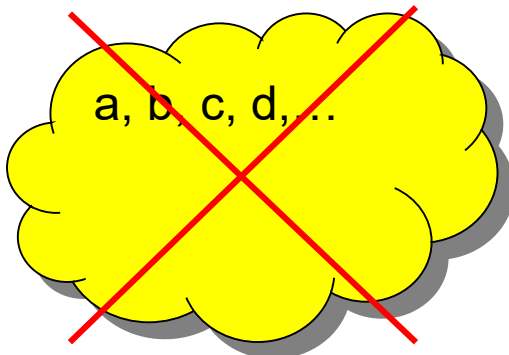
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

# 변수의 이름

- `sum` *// 영문 알파벳 문자로 시작*
- `_count` *// 밑줄 문자로 시작할 수 있다.*
- `number_of_pictures` *// 중간에 밑줄 문자를 넣을 수 있다.*
- `King3` *// 맨 처음이 아니라면 숫자도 넣을 수 있다.*
  
- `2nd_base(X)` *// 숫자로 시작할 수 없다.*
- `money#` *// #과 같은 기호는 사용할 수 없다.*
- `double` *// double은 C 언어의 키워드이다.*

# 좋은 변수 이름

- 변수의 역할을 가장 잘 설명하는 이름을 선택하여야 한다.
  - i, j, k (X)
  - year, month, date (O)
- 여러 단어로 된 이름을 만드는 방법
  - 밑줄 방식: bank\_account
  - 단어의 첫번째 글자를 대문자: BankAccount





# 변수의 초기화

- 변수에 초기값을 줄 수 있다.
  - `int x = 10;`
  - `int y = 20;`
  - `int sum = 0;`
- 동일한 타입의 변수인 경우, 같은 줄에서 선언과 동시에 변수들을 초기화할 수 있다.
  - `int width = 100, height = 200;`
- 다음과 같이 초기화하는 것은 문법적으로는 오류가 아니지만 피하는 것이 좋다.
  - `int width, height = 200;`

width는 초기화되지  
않는다.

# 수식

- 수식(expression): 피연산자와 연산자로 구성된 식
- 수식은 **결과값**을 가진다.

x가 3일때 수식  
 $x^2 - 5x + 6$ 의 값을  
계산하라.

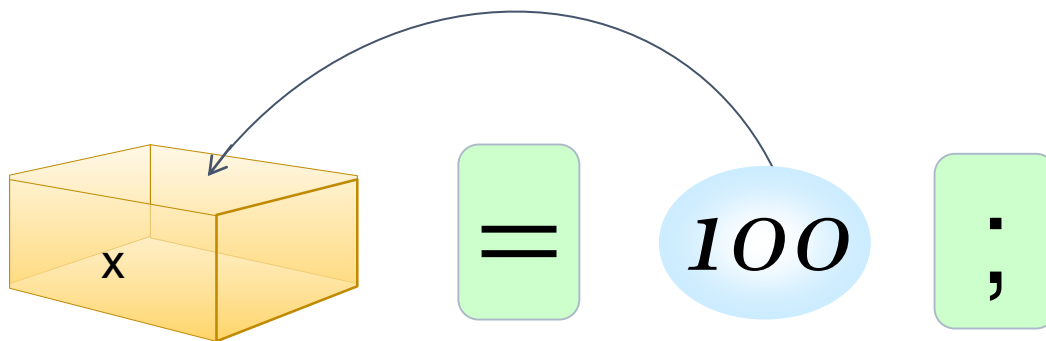


```
int x, y;  
  
x = 3;  
y = x * x - 5 * x + 6;  
printf("%d\n", y);
```

# 변수에 값 저장하기

```
x = 100;
```

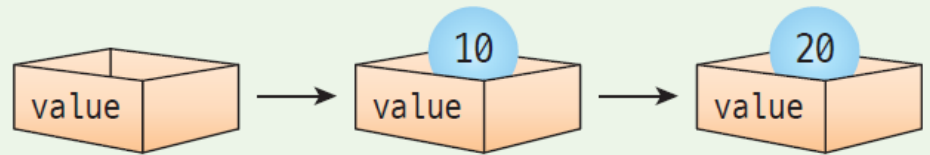
- 대입 연산(assignment operation): 변수에 값을 저장하는 연산
- 대입 연산 = 배정 연산 = 할당 연산



# 다양한 대입 연산

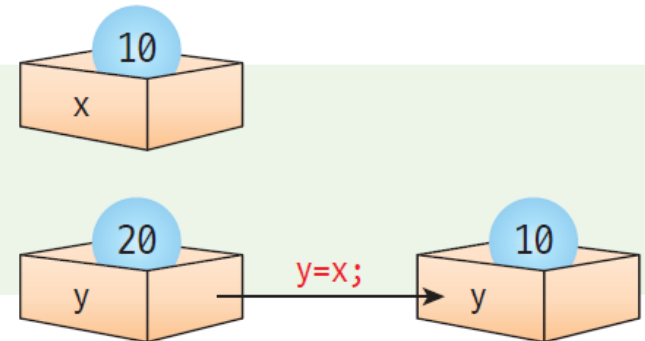
- 변수에는 = 기호를 이용하여 값을 저장할 수 있고 변수의 값은 몇 번이든지 변경이 가능하다.

```
int value;  
value = 10;  
value = 20;
```



```
int x = 10;  
int y = 20;  
y = x;
```

// y는 10이 된다.



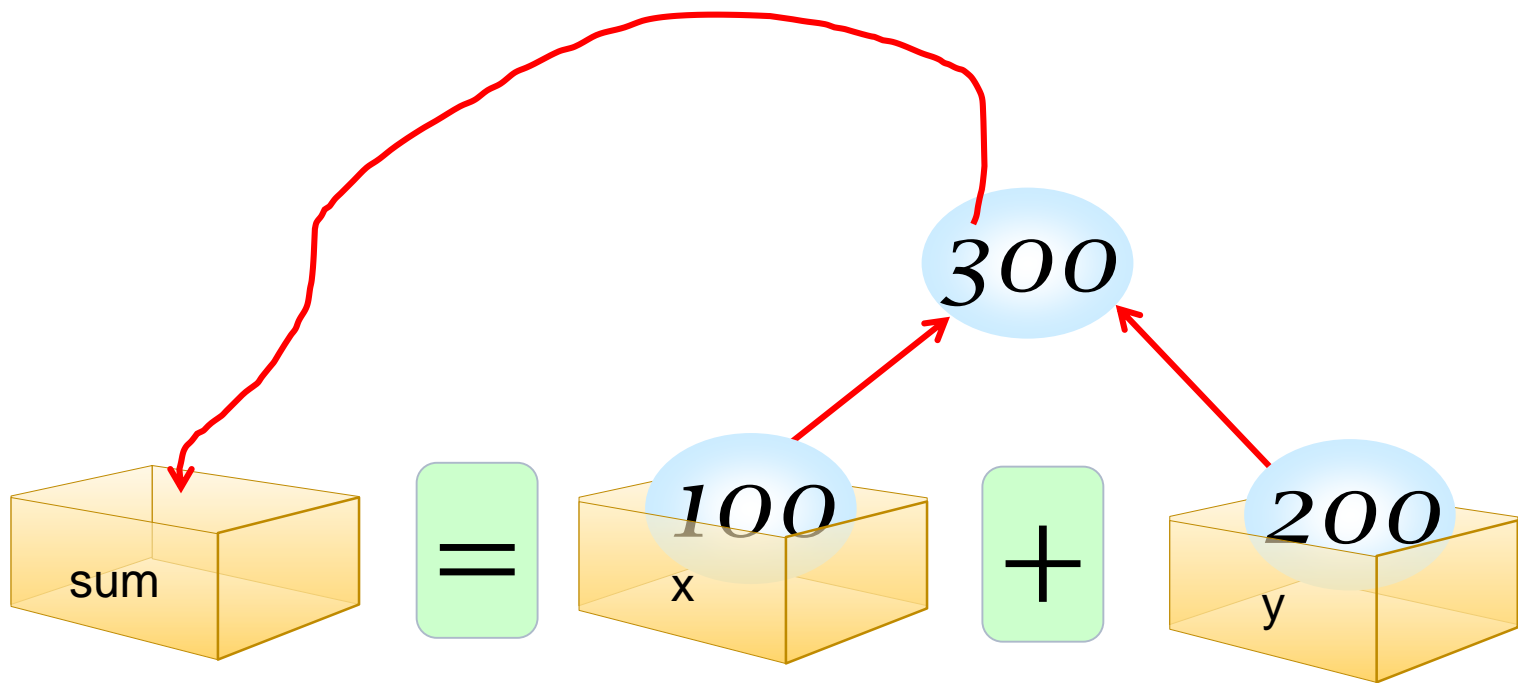
# 산술 연산

- 산술 연산자는 일반적으로 수학에서 사용하는 연산 기호와 유사하다.

연산	연산자	C 수식	수학에서의 기호
덧셈	+	$x + y$	$x + y$
뺄셈	-	$x - y$	$x - y$
곱셈	*	$x * y$	$xy$
나눗셈	/	$x / y$	$x/y$ 또는 $\frac{x}{y}$ 또는 $x \div y$
나머지	%	$x \% y$	$x \bmod y$

# 산술 연산

```
sum = x + y;
```



# 정리

프로그램

```
#include <stdio.h>
int main(void)
{
    int x;
    int y;
    int sum;

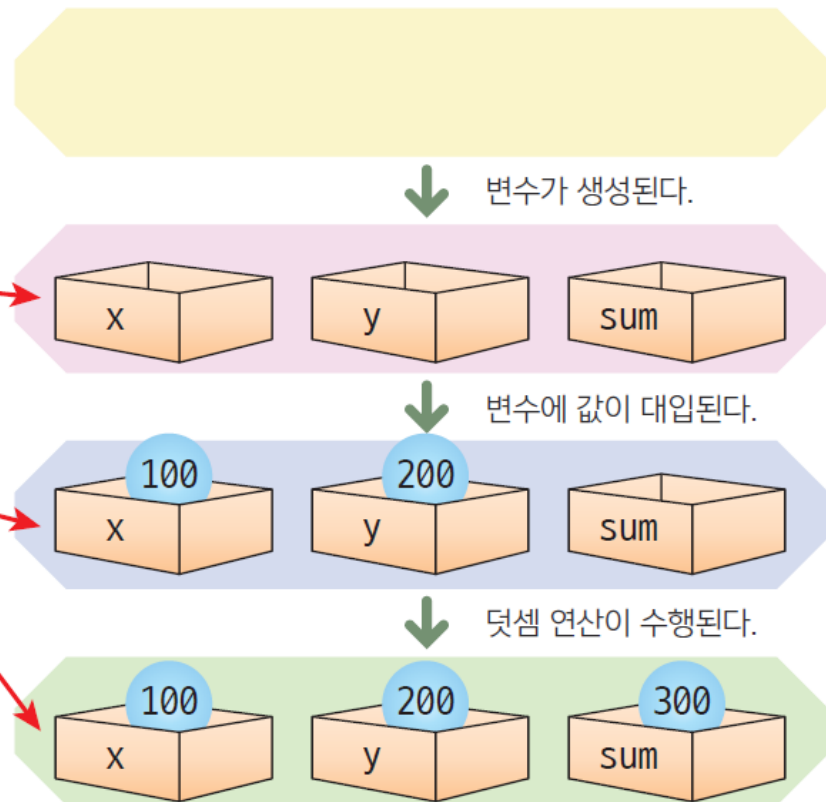
    x = 100;
    y = 200;

    sum = x + y;

    printf("두수의 합: %d", sum);

    return 0;
}
```

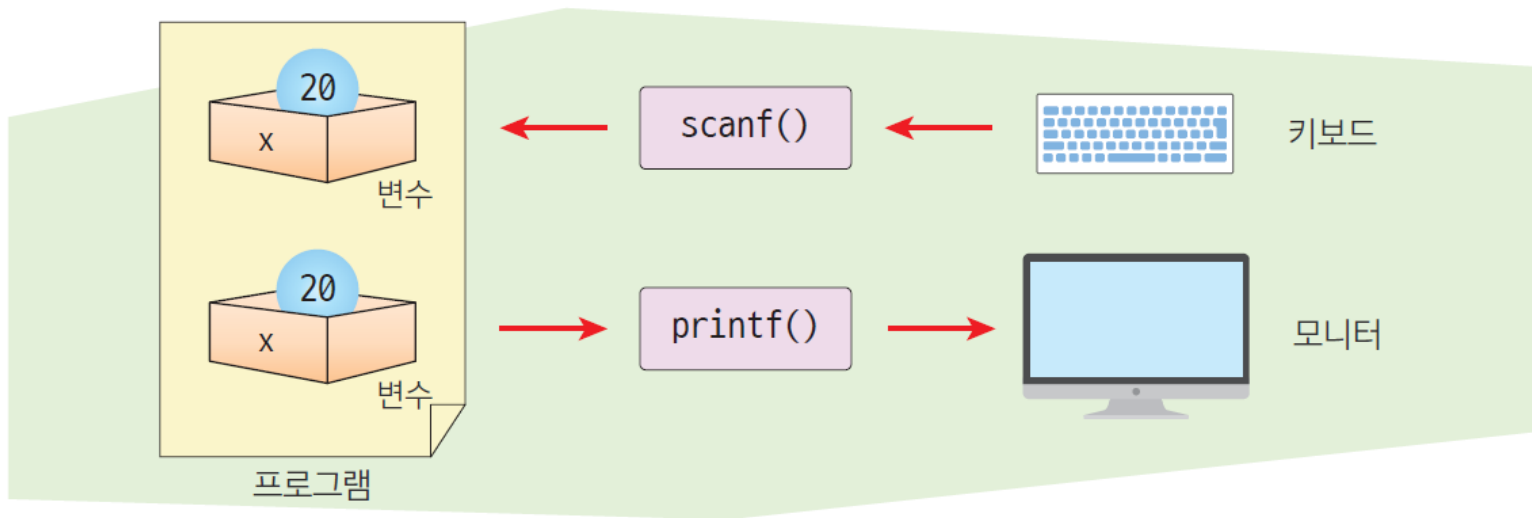
컴퓨터 내부



# 라이브러리 함수

- 라이브러리 함수: 라이브러리 함수란 컴파일러가 프로그래머가 사용할 수 있도록 제공하는 함수

- printf(): 모니터에 출력을 하기 위한 표준 출력 함수
- scanf(): 키보드에서의 입력을 위한 표준 입력 함수





# 문자열 출력

```
printf("Hello World!\n");
```

- 문자열(string): "Hello World!\n"와 같이 문자들을 여러 개 나열한 것



# 변수값 출력

- 출력 형식

```
printf("두수의 합: %d \n, sum");
```



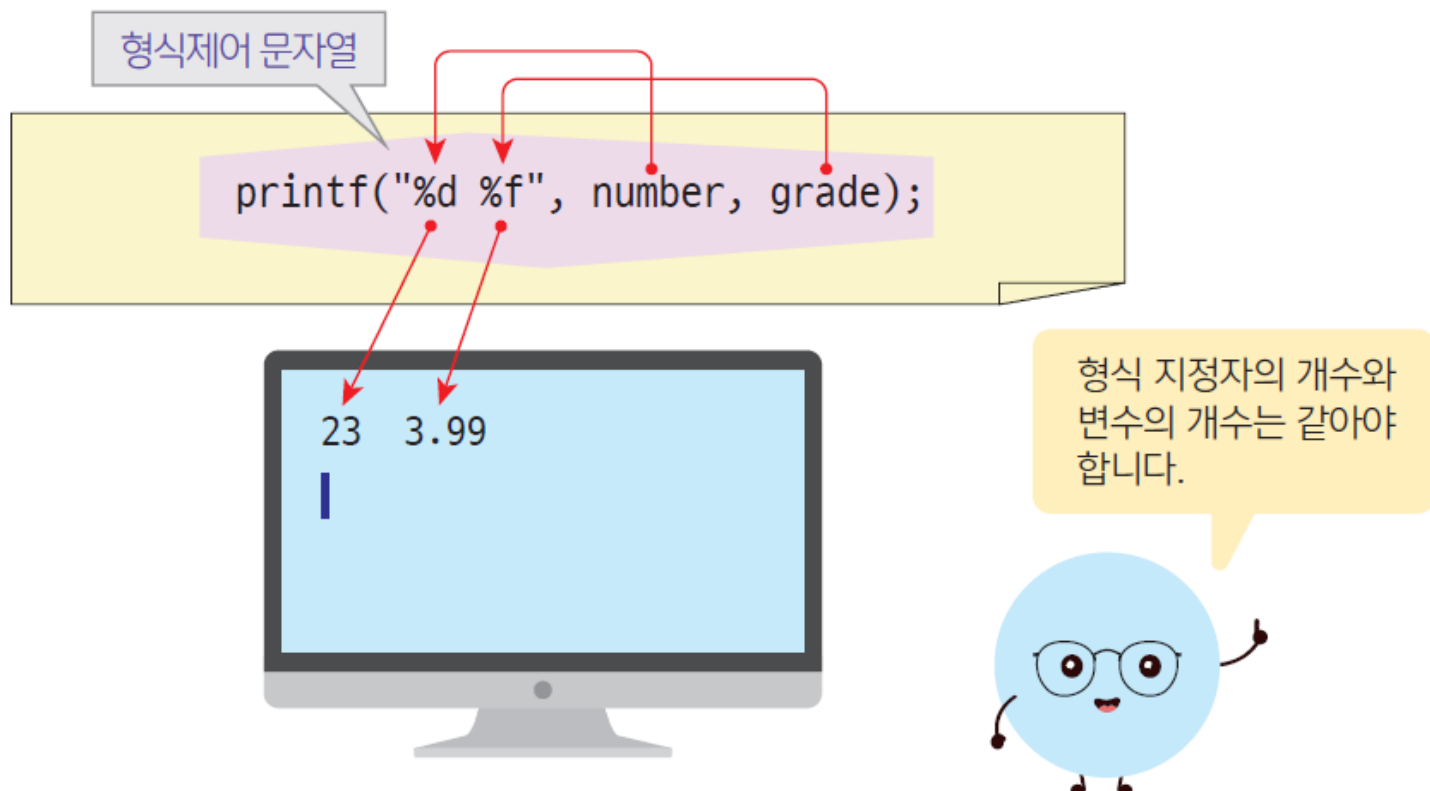
# 형식 지정자

- 형식 지정자: printf()에서 값을 출력하는 형식을 지정한다.

형식 지정자	의미	예	실행 결과
%d	10진 정수로 출력	<code>printf("%d \n", 10);</code>	10
%f	실수로 출력	<code>printf("%f \n", 3.14);</code>	3.14
%c	문자로 출력	<code>printf("%c \n", 'a');</code>	a
%s	문자열로 출력	<code>printf("%s \n", "Hello");</code>	Hello

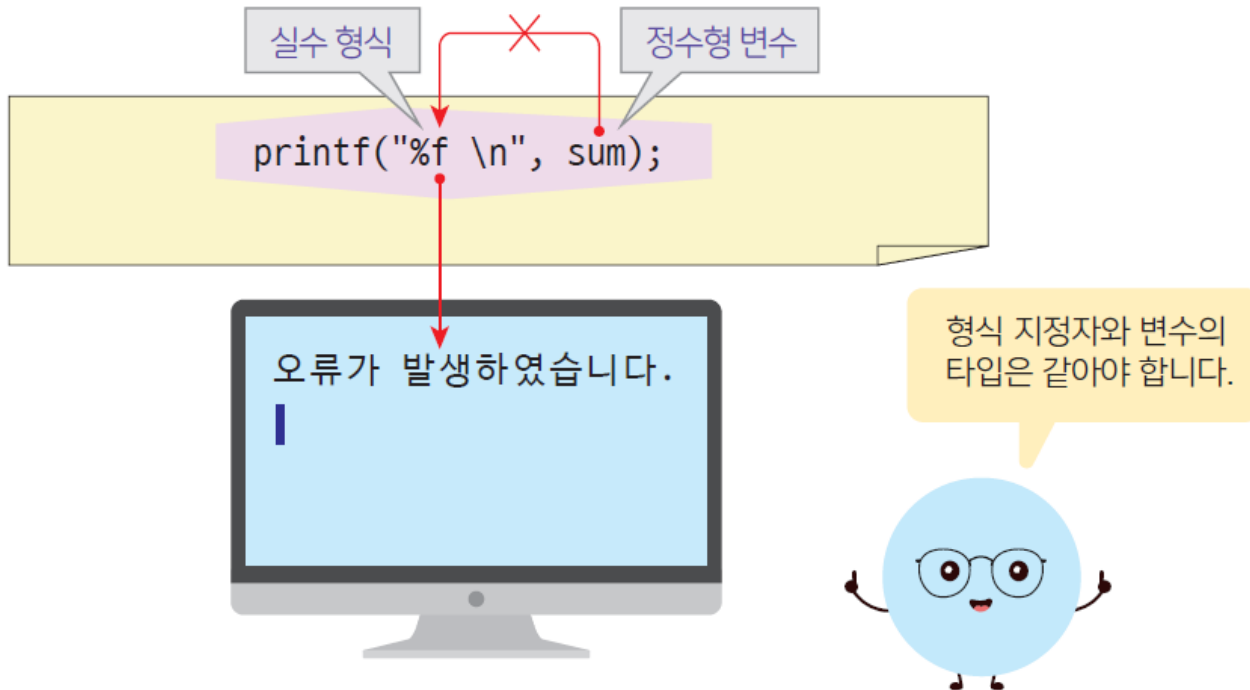
# 여러 개의 변수값 출력

- 형식 지정자의 자리에 변수의 값이 대치되어서 출력된다고 생각하며 되다



# 주의!

- 형식과 변수의 자료형은 반드시 일치하여야 한다는 점이다



# 필드폭(width)과 정밀도(precision)


- printf( )를 사용하여 출력할 때, 데이터가 출력되는 필드의 크기에  
저장하는 이다

출력 문장	출력 결과	설명										
printf("%10d", 123);	<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>2</td><td>3</td></tr></table>								1	2	3	폭은 10, 우측정렬
							1	2	3			
printf("%-10d", 123);	<table><tr><td>1</td><td>2</td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	1	2	3								폭은 10, 좌측정렬
1	2	3										

출력 문장	출력 결과	설명										
printf("%f", 1.23456789);	<table><tr><td>1</td><td>.</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>8</td><td></td><td></td></tr></table>	1	.	2	3	4	5	6	8			소수점 이하 6자리
1	.	2	3	4	5	6	8					
printf("%10.3f", 1.23456789);	<table><tr><td></td><td></td><td></td><td></td><td></td><td>1</td><td>.</td><td>2</td><td>3</td><td>5</td></tr></table>						1	.	2	3	5	소수점 이하 3자리
					1	.	2	3	5			
printf("%-10.3f", 1.23456789);	<table><tr><td>1</td><td>.</td><td>2</td><td>3</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr></table>	1	.	2	3	5						좌측 정렬
1	.	2	3	5								
printf("%.3f", 1.23456789);	<table><tr><td>1</td><td>.</td><td>2</td><td>3</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr></table>	1	.	2	3	5						소수점 이하 자리만 표시
1	.	2	3	5								

# Lab: 사칙 연산

- 변수  $x$ 와  $y$ 에 20과 10을 저장하고  $x+y$ ,  $x-y$ ,  $x*y$ ,  $x/y$ 을 계산하여서 변수에 저장하고 이들 변수를 화면에 출력하는 프로그램을 작성해보자.



두수의 합: 30  
두수의 차: 10  
두수의 곱: 200  
두수의 몫: 2

# Solution

```
// 정수 간의 가감승제를 계산하는 프로그램
#include <stdio.h>

int main(void)
{
    int x;                // 첫 번째 정수를 저장할 변수
    int y;                // 두 번째 정수를 저장할 변수
    int sum, diff, mul, div; // 두 정수 간의 연산의 결과를 저장하는 변수

    x = 20;                // 변수 x에 20을 저장
    y = 10;                // 변수 y에 10을 저장

    sum = x + y;           // 변수 sum에 (x+y)의 결과를 저장
    diff = x - y;          // 변수 diff에 (x-y)의 결과를 저장
    mul = x * y;           // 변수 mul에 (x*y)의 결과를 저장
    div = x / y;           // 변수 div에 (x/y)의 결과를 저장
```



# Solution

```
printf("두수의 합: %d\n", sum);           // 변수 sum의 값을 화면에 출력
printf("두수의 차: %d\n", diff); // 변수 diff의 값을 화면에 출력
printf("두수의 곱: %d\n", mul);           // 변수 mul의 값을 화면에 출력
printf("두수의 몫: %d\n", div); // 변수 div의 값을 화면에 출력

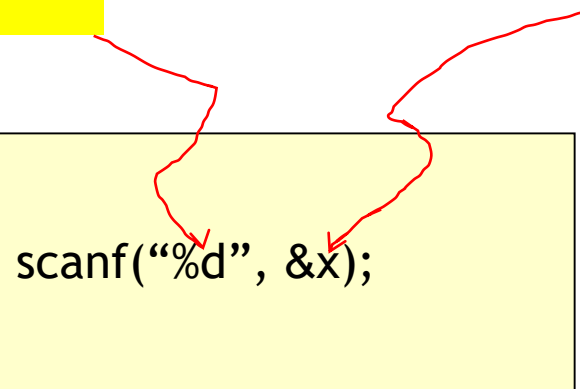
return 0;
}
```

# scanf()

- 키보드로부터 값을 받아서 변수에 저장한다.
- 변수의 주소를 필요로 한다.

형식 지정자

값을 저장할 변수의 주소



scanf("%d", &x);

The diagram illustrates the two arguments of the scanf function. A red arrow originates from the yellow box labeled '형식 지정자' (Format specifier) and points to the '%d' in the code. Another red arrow originates from the yellow box labeled '값을 저장할 변수의 주소' (Address of variable to store value) and points to the '&x' in the code. The code itself is contained within a light yellow rectangular box.

# 주소가 필요한 이유

- 우리가 인터넷에서 제품을 구입하고, 집으로 배달시키려면 쇼핑몰에 구매자의 주소를 알려주어야 하는 것과 비슷하다.



&x

& 연산자는 변수의 주소를 계산한다.

# scanf()의 형식지정자

- 대부분 printf()와 같다.

형식 지정자	의미	예
%d	10진 정수를 입력한다	scanf("%d", &i);
%f	float 형의 실수를 입력한다.	scanf("%f", &f);
%lf	double 형의 실수를 입력한다.	scanf("%lf", &d);
%c	하나의 문자를 입력한다.	scanf("%c", &ch);
%s	문자열을 입력한다.	char s[10]; scanf("%s", s);

아직 학습하지 않았음!  
너무 신경쓰지 말것!

# 실수 입력시 주의할 점

- float 형은 %f 사용

```
float ratio = 0.0;  
scanf("%f", &ratio);
```

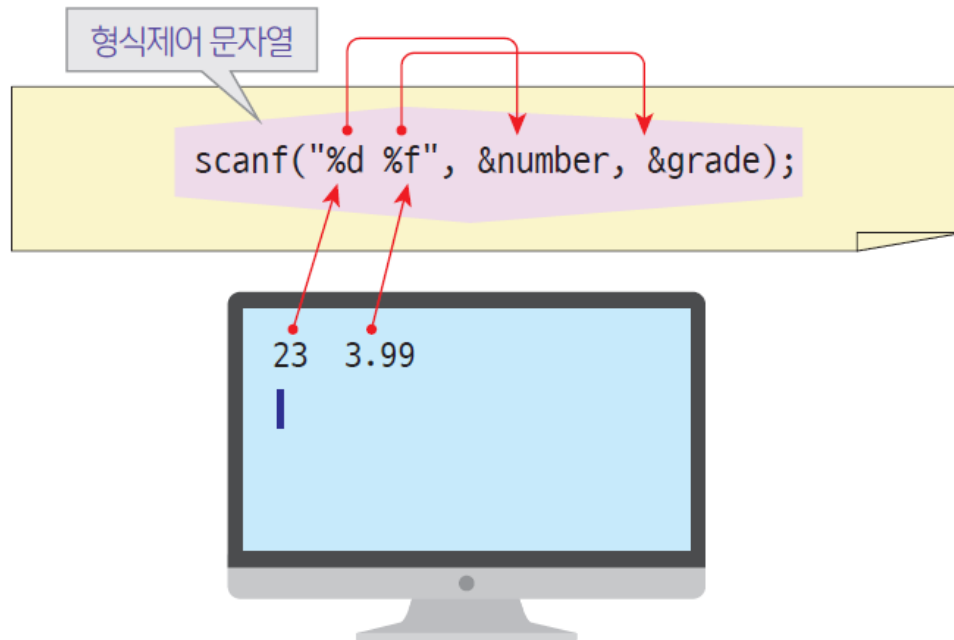
```
double scale = 0.0;  
scanf("%lf", &scale);
```

- double 형은 %lf 사용

잘못 사용하면 오류가  
발생합니다.



# scanf()



형식 지정자의 개수와  
변수의 개수는 같아야  
합니다.



# scanf() 사용시 컴파일 오류가 난다면?

⚠ C6031 반환 값이 무시되었습니다. 'scanf'.

✖ C4996 'scanf': This function or variable may be unsafe. Consider using scanf\_s instead. To disable deprecation, use \_CRT\_SECURE\_NO\_WARNINGS. S

scanf()가 안전하지 않으니  
scanf\_s()를 사용하라는  
의미이다.

만약 이런 오류가 발생하면 다음  
페이지와 같이  
\_CRT\_SECURE\_NO\_WARNINGS  
를 정의해준다.

# 정수를 받아들이는 프로그램

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main(void)
{
    int x;                // 정수를 저장할 변수
    printf("정수를 입력하시오: ");
    scanf("%d", &i);
    printf("입력된 정수 = %d \n", i);
    return 0;
}
```

만약 scanf() 오류가 발생하면 소스 파일의 처음에서 \_CRT\_SECURE\_NO\_WARNINGS를 정의해준다.

Microsoft Visual Studio 디버그 콘솔

정수를 입력하시오: 20  
입력된 정수 = 20

C:\Users\chun\source\repos\Project8\Debug\Project8.exe(28548 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.



# scanf() 정리

책의 소스에서 scanf() 오류가 발생하면  
소스 파일의 첫부분에서  
\_CRT\_SECURE\_NO\_WARNINGS를  
정의해준다.

Windows



소스의 첫 부분에  
한 줄을 추가한다.

```
#define _CRT_SECURE_NO_WARNINGS  
...  
...
```

Linux




소스를 그대로  
사용한다.

```
...  
...
```

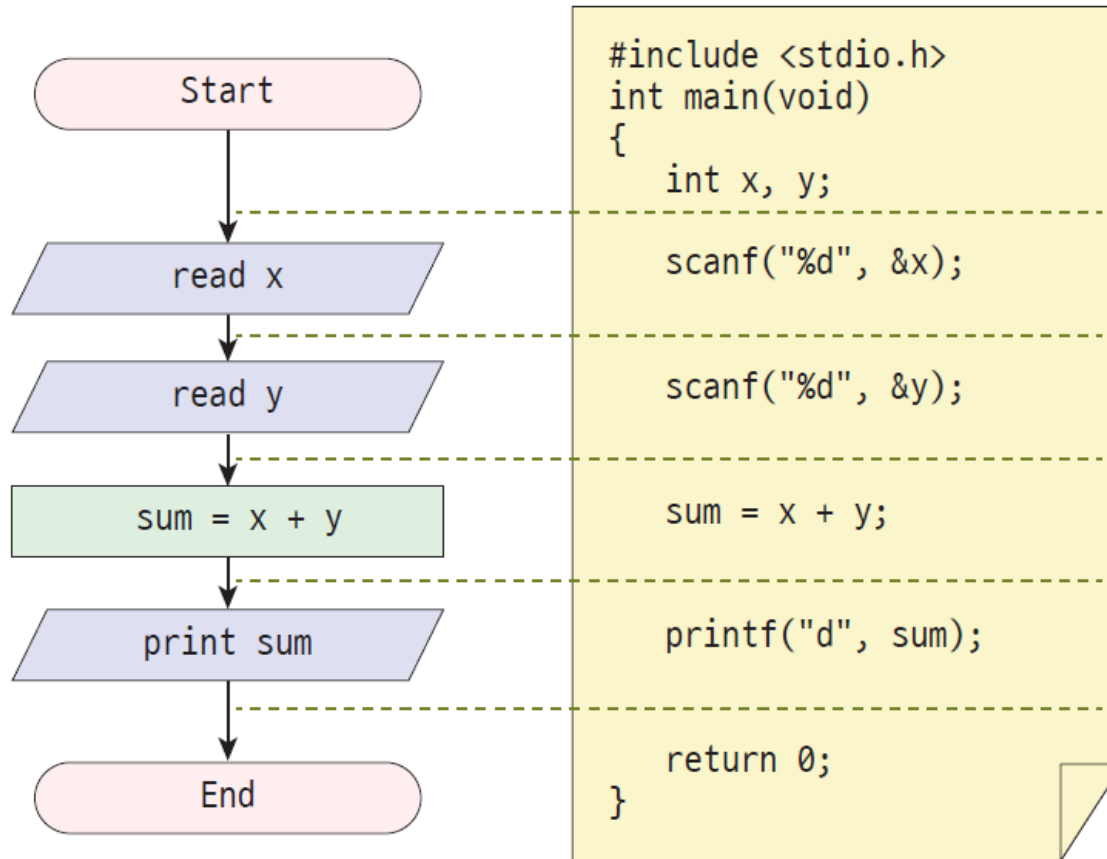
# 덧셈 프로그램 #2

- 사용자로부터 입력을 받아보자.



첫번째 숫자를 입력하시오: 10  
두번째 숫자를 입력하시오: 20  
두수의 합: 30

# 알고리즘



# 두번째 덧셈 프로그램

```
// 사용자로부터 입력받은 2개의 정수의 합을 계산하여 출력
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    int x;                // 첫번째 정수를 저장할 변수
    int y;                // 두번째 정수를 저장할 변수
    int sum;              // 2개의 정수의 합을 저장할 변수

    printf("첫번째 숫자를 입력하십시오:"); // 입력 안내 메시지 출력
    scanf("%d", &x);                // 하나의 정수를 받아서 x에 저장


    printf("두번째 숫자를 입력하십시오:"); // 입력 안내 메시지 출력
    scanf("%d", &y);                // 하나의 정수를 받아서 x에 저장

    sum = x + y;                // 변수 2개를 더한다.
    printf("두수의 합: %d", sum); // sum의 값을 10진수 형태로 출력

    return 0;                // 0을 외부로 반환
}
```

# 원의 면적 계산 프로그램

- 사용자로부터 원의 반지름을 입력받고 이 원의 면적을 구한 다음, 화면에 출력한다.



반지름을 입력하시오: 10.0  
원의 면적: 314.000000

# 원의 면적 계산 프로그램

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>


int main(void)
{
    float radius; // 원의 반지름
    float area; // 면적

    printf("반지름을 입력하시오 : ");
    scanf("%f", &radius);
    area = 3.14 * radius * radius;
    printf("원의 면적 : %f\n", area);

    return 0;
}
```

# 환율 계산 프로그램

- 사용자가 입력하는 원화를 달러화로 계산하여 출력하는 프로그램은 작성하여 보자.



환율을 입력하시오: 1400  
원화 금액을 입력하시오: 1000000  
원화 1000000원은 714.285714달러입니다.

```
/* 환율을 계산하는 프로그램*/
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    double rate; // 원/달러 환율
    double usd; // 달러화
    int krw; // 원화는 정수형 변수로 선언

    printf("환율을 입력하시오: "); // 입력 안내 메시지
    scanf("%lf", &rate); // 사용자로부터 환율입력

    printf("원화 금액을 입력하시오: "); // 입력 안내 메시지
    scanf("%d", &krw); // 원화 금액 입력

    usd = krw / rate; // 달러화로 환산


    printf("원화 %d원은 %lf달러입니다.\n", krw, usd); // 계산 결과 출력

    return 0; // 함수 결과값 반환
}
```



# 평균 계산하기 프로그램

- 사용자로부터 세 개의 double형의 실수를 입력받은 후, 합계와 평균값을 계산하여 화면에 출력하는 프로그램을 작성하라.



3개의 실수를 입력하시오: 10.2 21.5 32.9  
합계=64.60  
평균=21.53

# 평균 계산하기 프로그램

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    double num1, num2, num3;
    double sum, avg;

    printf("3개의 실수를 입력하시오: ");
    scanf("%lf %lf %lf", &num1, &num2, &num3);           // 3개의 실수 입력

    sum = num1 + num2 + num3;
    avg = sum / 3.0;

    printf("합계=%.2lf\n", sum);                          // 소수점 이하를 2자리로 표시
    printf("평균=%.2lf\n", avg);

    return 0;
}
```

# Q & A

