

기초컴퓨터프로그래밍

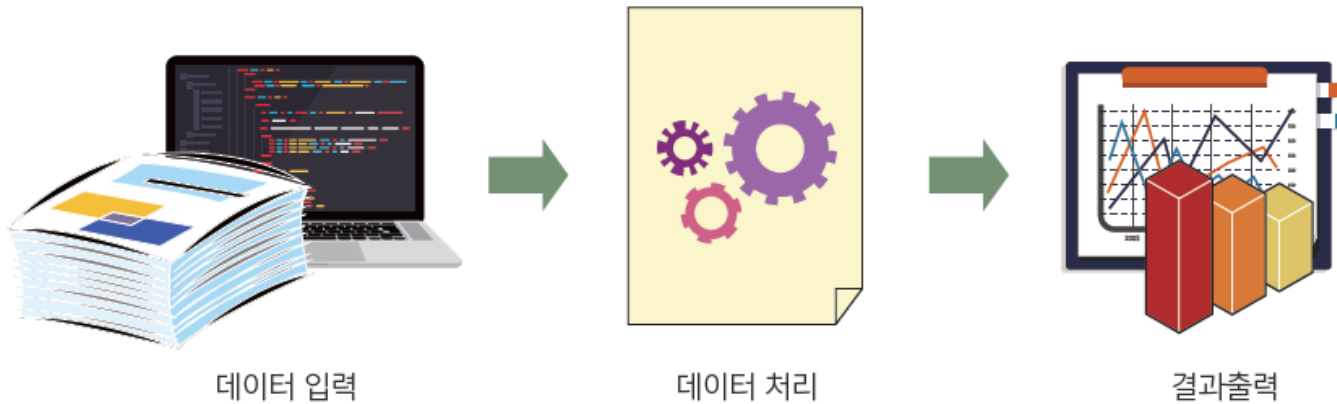
제 3장 C언어 프로그램 구성 요소

양우석 yws87874912@gmail.com



일반적인 프로그램의 형태

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





더셈 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

연산

함수

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
- C언어는 들여쓰기를 잘못해도 에러가 나진 않음. 하지만 가독성을 위해!

```
#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.
같은내용, 같은블럭, 같은구문



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.





작가 점검

1. 주석은 /* /* */ */와 같이 중첩할 수 있을까?
2. 주석은 한 줄 이상이 될 수 있는가?
3. 주석에는 어떤 내용을 쓰면 좋은가?
4. 주석은 프로그램의 동작에 어떤 영향을 끼치는가?





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



중간 점검

1. `printf()`를 사용하기 위하여 포함시켜야 하는 헤더 파일은 무엇인가?
2. 전처리기 `#include`의 의미는 무엇인가?





함수

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

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

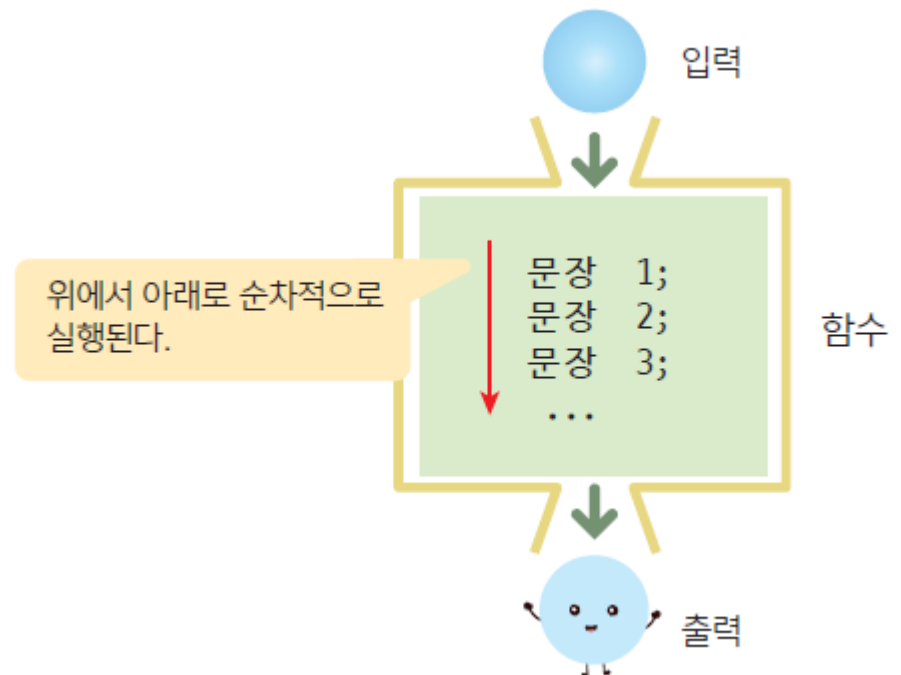




함수 안에 들어 있는 것

Q) 그렇다면 함수 안에 들어 있는 것은 무엇인가?

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





함수의 구조

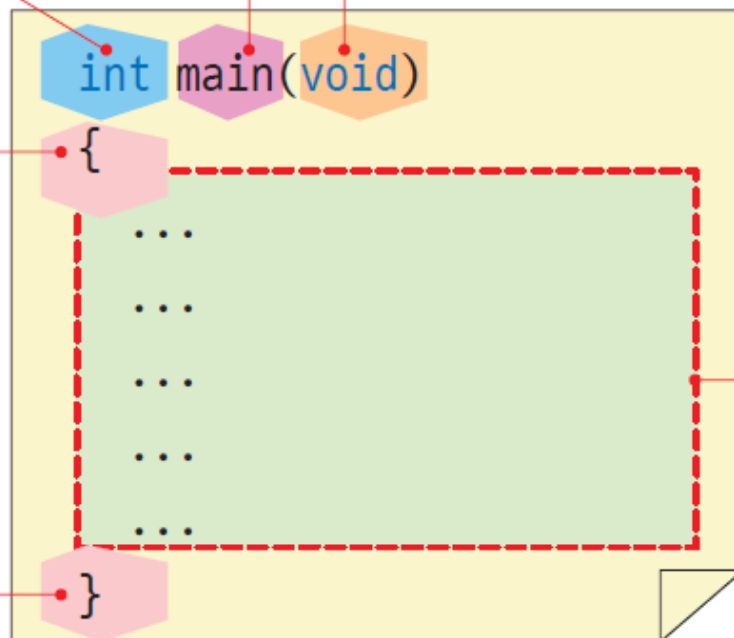
함수의 출력 타입

함수의 이름

함수의 입력 타입, void는 입력이 없다는 의미이다.

함수의 시작

함수의 종료

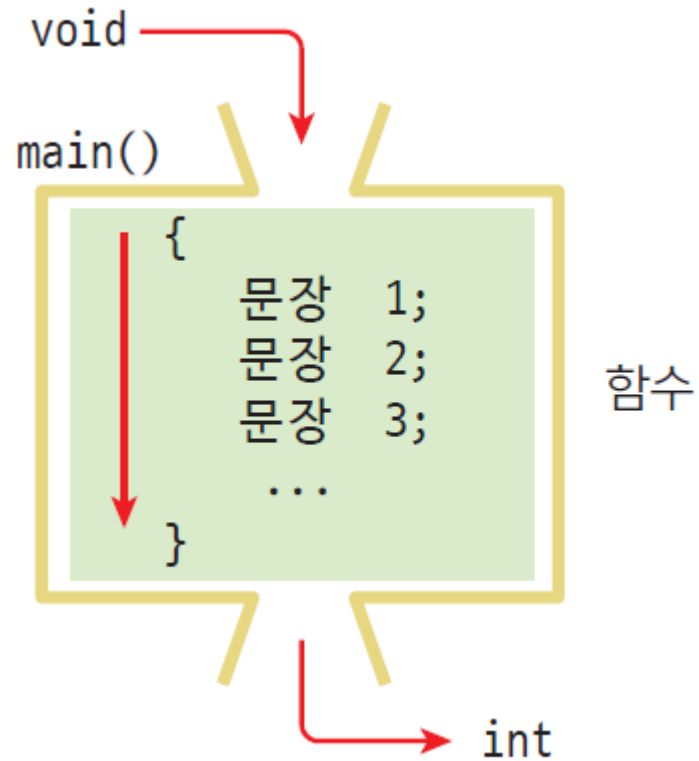


함수의 몸체, 함수가 수행하는 작업에 해당하는 문장들이 들어간다.



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) {
```

```
    ...
```

```
    ...
```

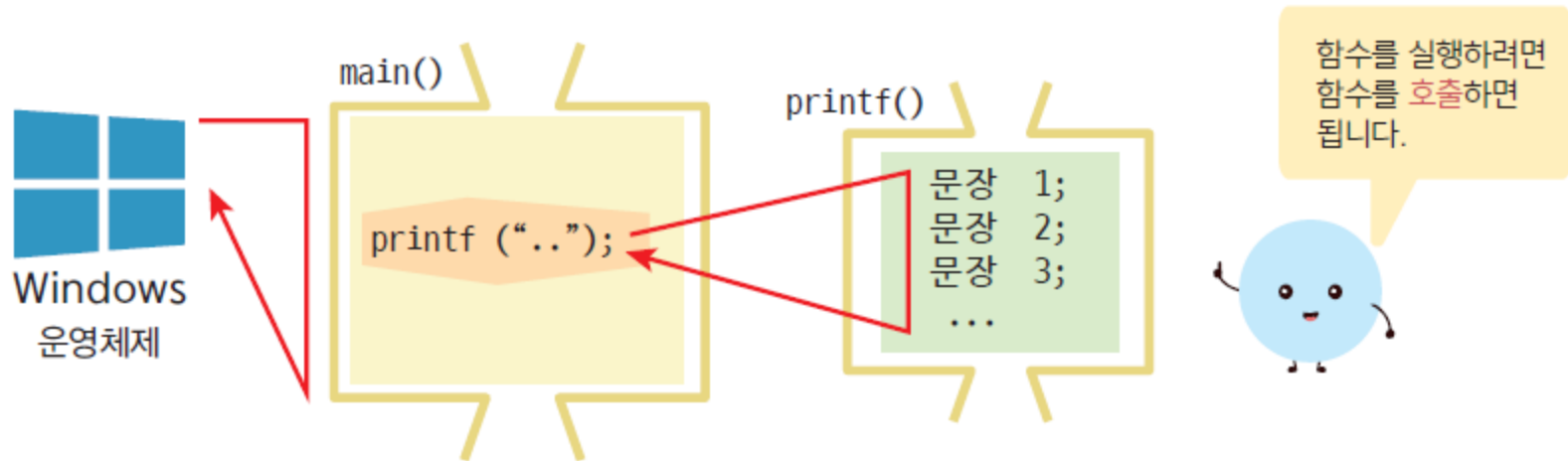
```
    return 0;
```

```
}
```





main()은 누가 호출할까?





장간 점검

1. 모든 C 프로그램에 반드시 있어야 되는 함수는 무엇인가?
2. 함수의 시작과 끝을 나타내는 기호는 무엇인가?
3. 모든 문장은 어떤 기호로 끝나는가?





변수

- 프로그래밍이 사용하는 데이터를 일시적으로 저장할 목적으로 사용하는 메모리 공간

Syntax

변수 선언

예

자료형

int

x;

변수 이름

int y;

int sum;



변수는 왜 필요한가?

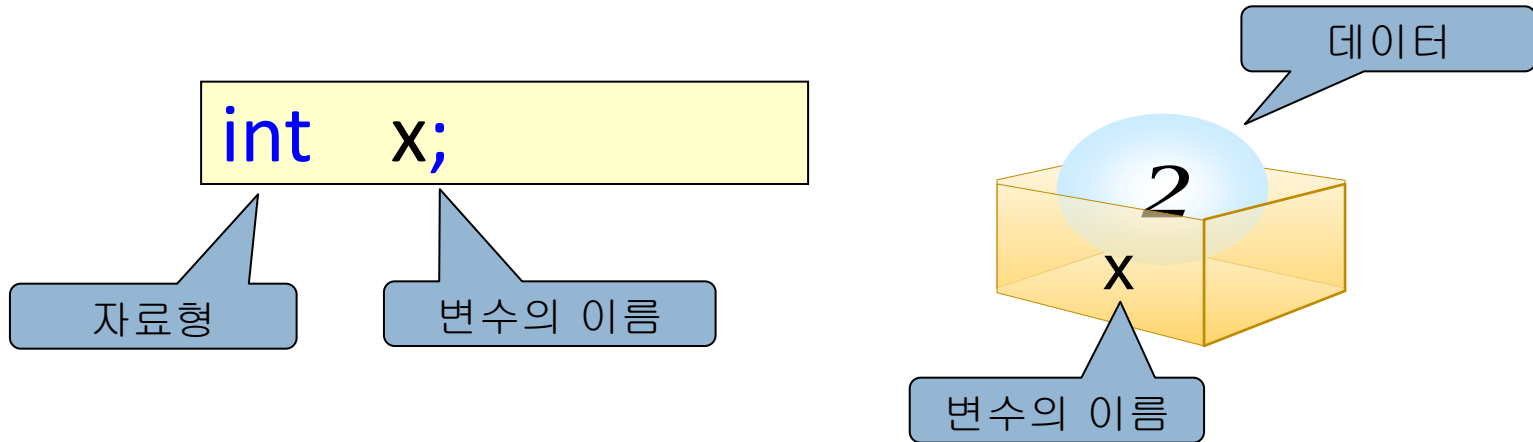
- 변수는 데이터 값을 일시적으로 저장하는 역할을 한다.





변수의 종류

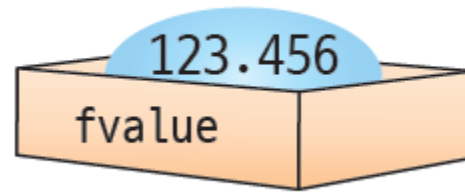
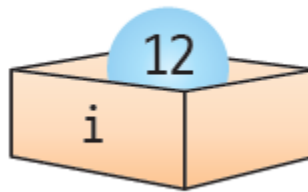
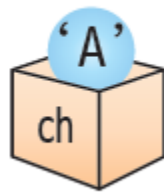
- 변수는 데이터를 담는 상자로 생각할 수 있다.





변수의 종류

- 변수에는 데이터의 종류에 따라 여러 가지 타입이 존재한다.

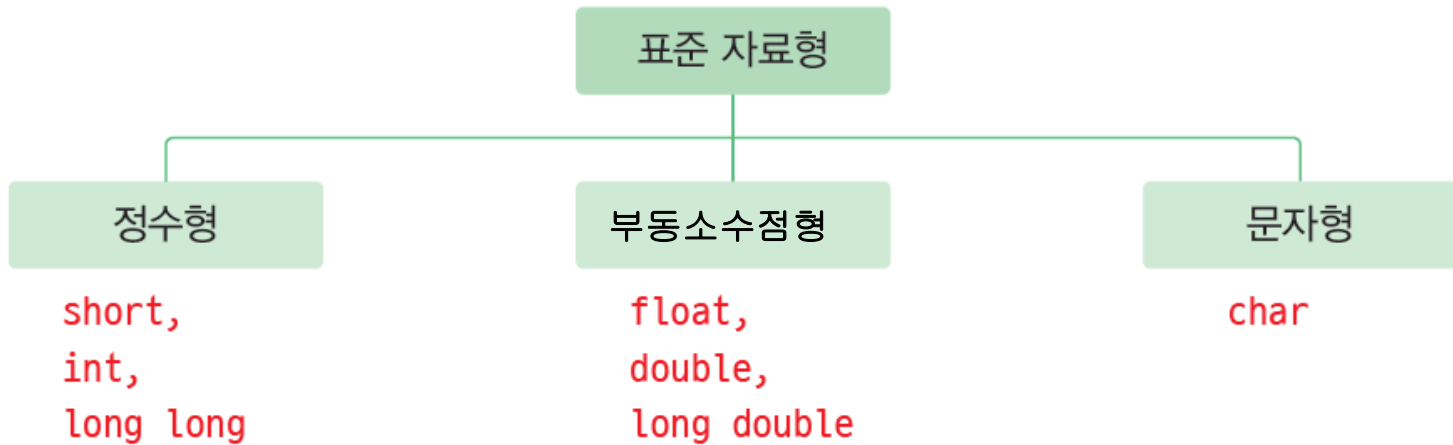


Note: Not to Scale



자료형

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

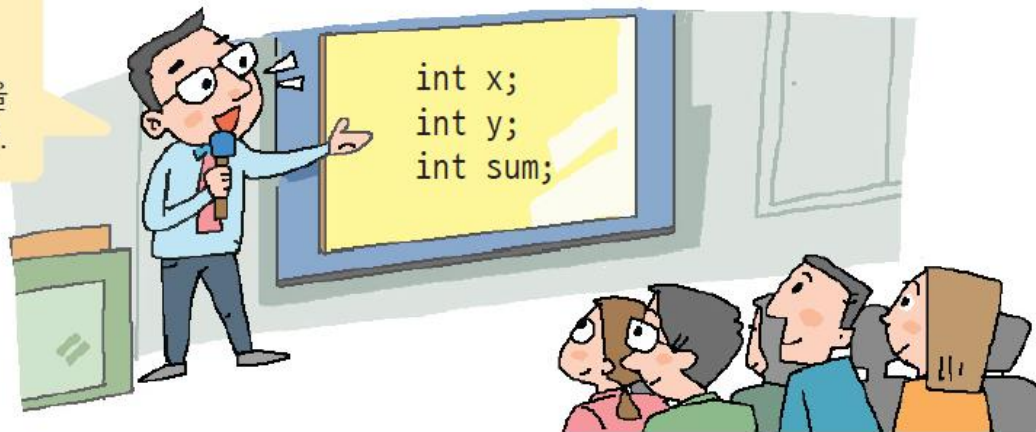




변수 선언

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

지금부터 이
프로그램에서
사용될 변수들을
소개하겠습니다.



컴파일러



변수 선언

자료형은 정수형

변수 이름

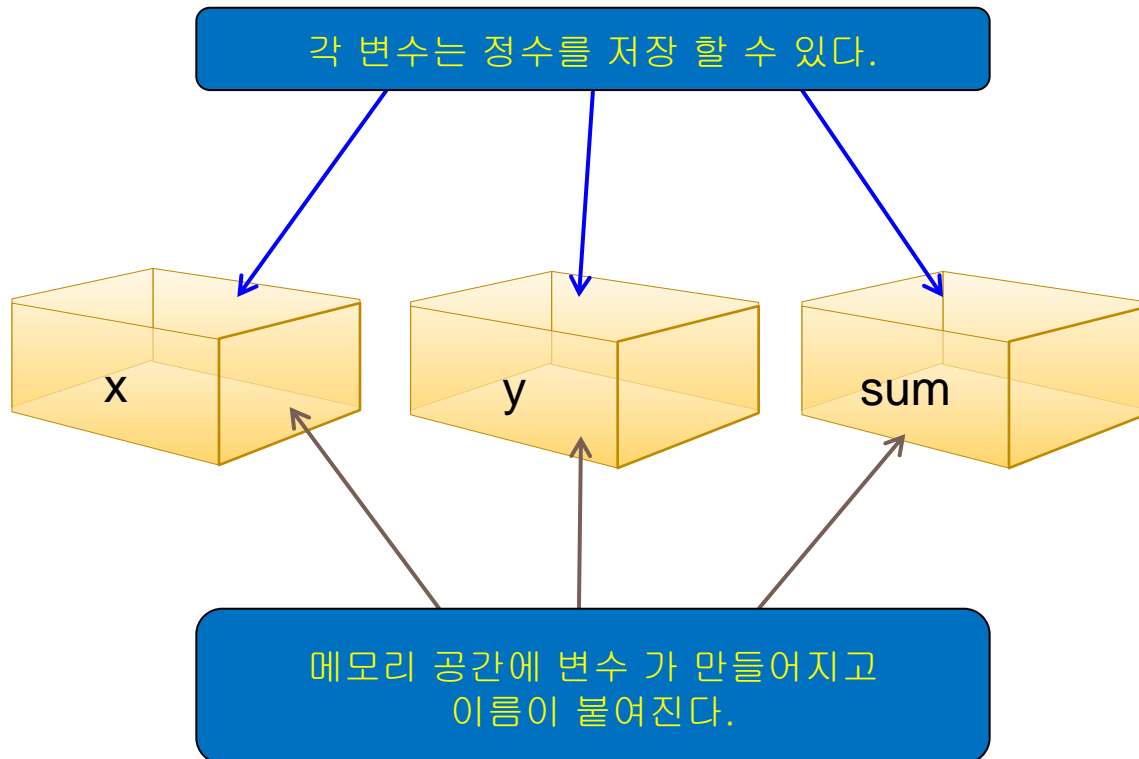
```
int x;  
int y;  
int sum;
```

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



변수 선언

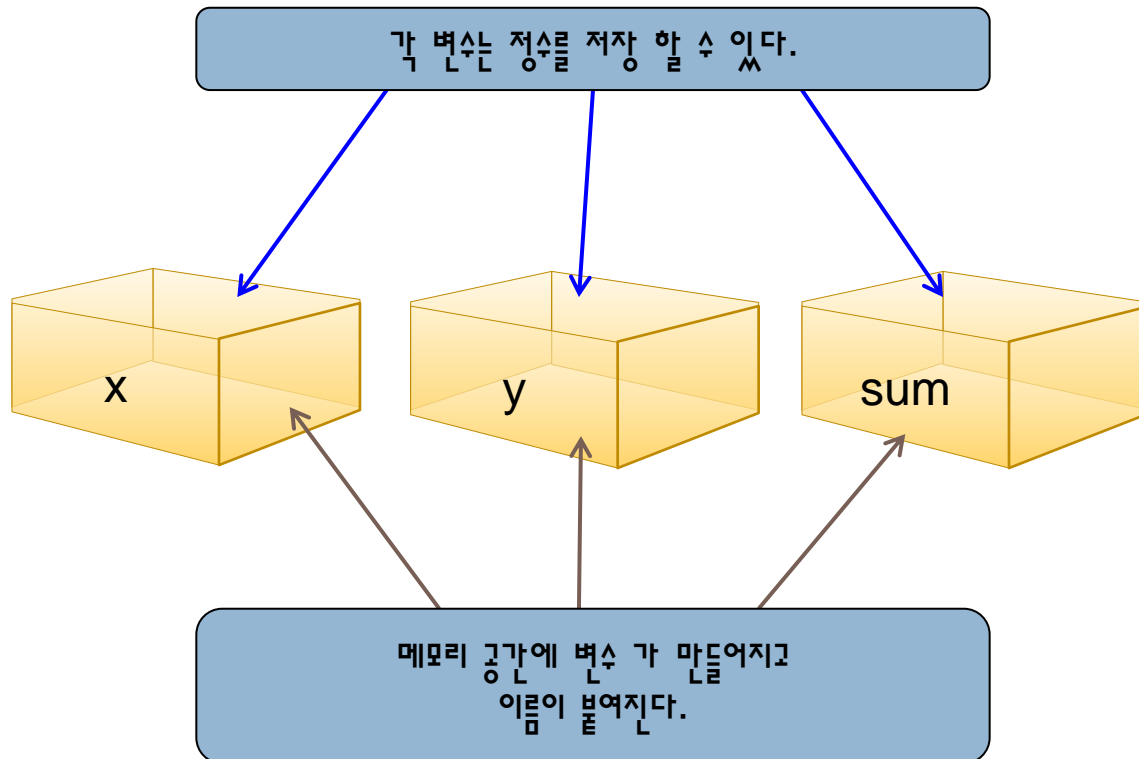
```
int x;    // 첫번째 정수를 저장하는 변수  
int y;    // 두번째 정수를 저장하는 변수  
int sum;  // 두 정수의 합을 저장하는 변수
```





한 줄에 여러 개의 변수 선언

```
int x, y, sum;    //가능!!
```





중간점검

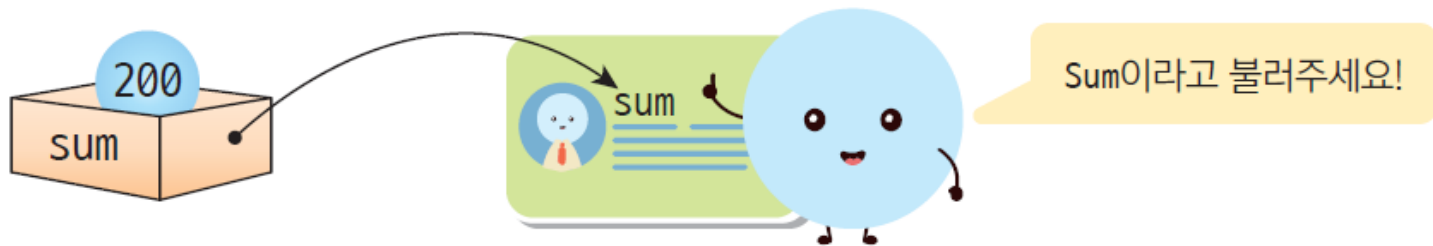
1. `double`형 변수 `f`를 선언하는 문장을 작성하여 보자.
2. 변수 선언은 함수의 어떤 위치에서 하여야 하는가?
3. 변수 선언은 함수의 어떤 위치에서 하여야 하는가?





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 `index`와 `Index`, `INDEX`은 모두 서로 다른 변수이다.
- C언어의 키워드와 똑같은 식별자는 허용되지 않는다.

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

함수 이름

변수 이름



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

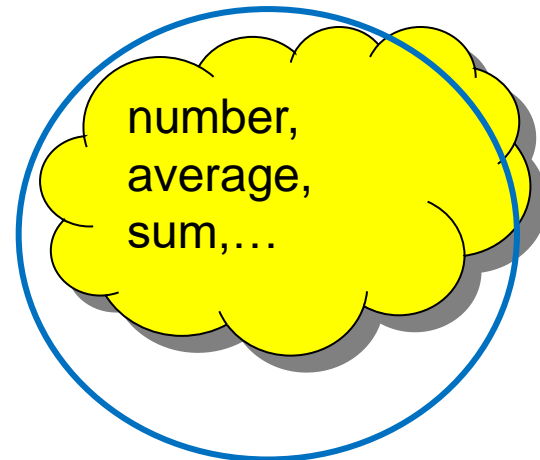
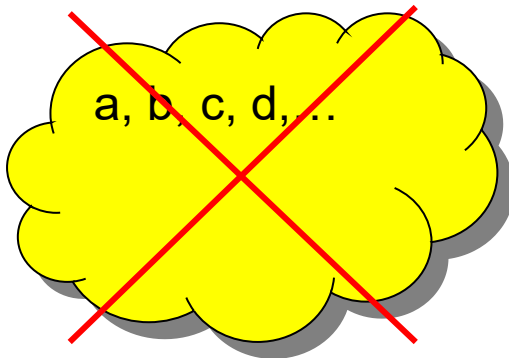
- `sum` *// 영문 알파벳 문자로 시작*
- `_count` *// 밑줄 문자로 시작할 수 있다.*
- `number_of_pictures` *// 중간에 밑줄 문자를 넣을 수 있다.*
- `King3` *// 맨 처음이 아니라면 숫자도 넣을 수 있다.*

- `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`





중간 점검

1. 변수 이름을 만들 때 지켜야 하는 규칙은 무엇인가?
2. 변수 이름의 첫 번째 글자로 허용되는 것은 무엇인가?
3. C에서 고유한 의미를 가지고 있는 단어들을 무엇이라고 하는가?





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.



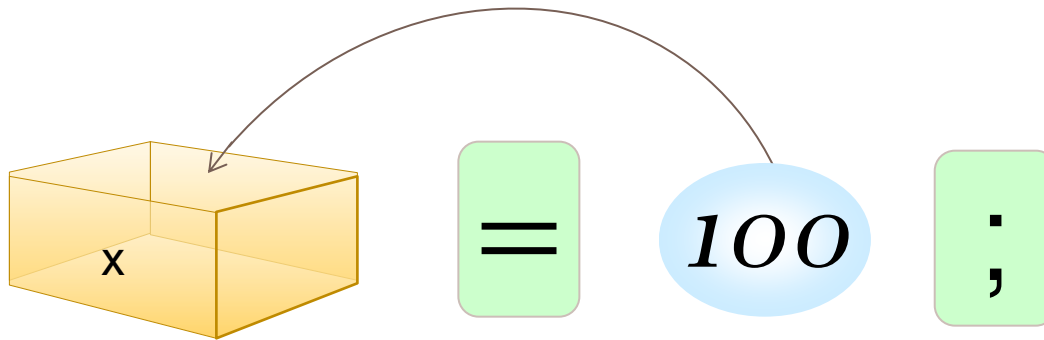
```
int x, y;  
  
x = 3;  
y = x * x - 5 * x + 6;  
printf("%d\n", y);
```



변수에 값 저장하기

```
x = 100;
```

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

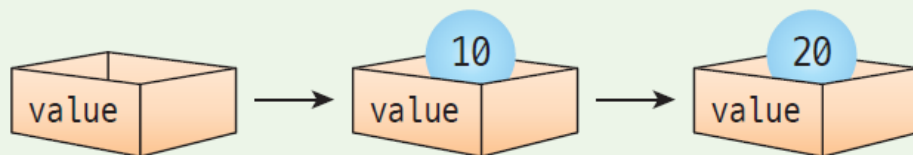




다양한 대입 연산

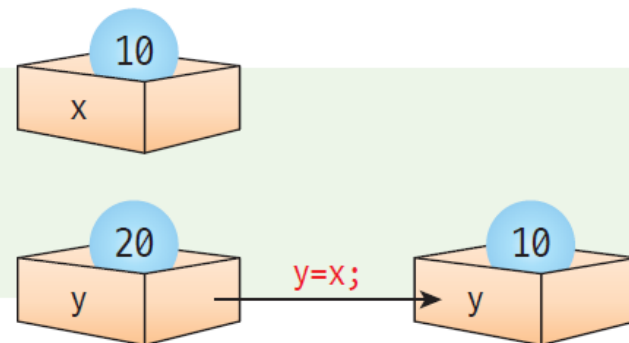
- 변수에는 = 기호를 이용하여 값을 저장할 수 있고 변수의 값은 몇 번이든지 변경이 가능하다.

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



- 변수에는 다른 변수의 값도 대입할 수도 있다.

```
int x = 10;  
int y = 20;  
y = x;           // y는 10이 된다.
```





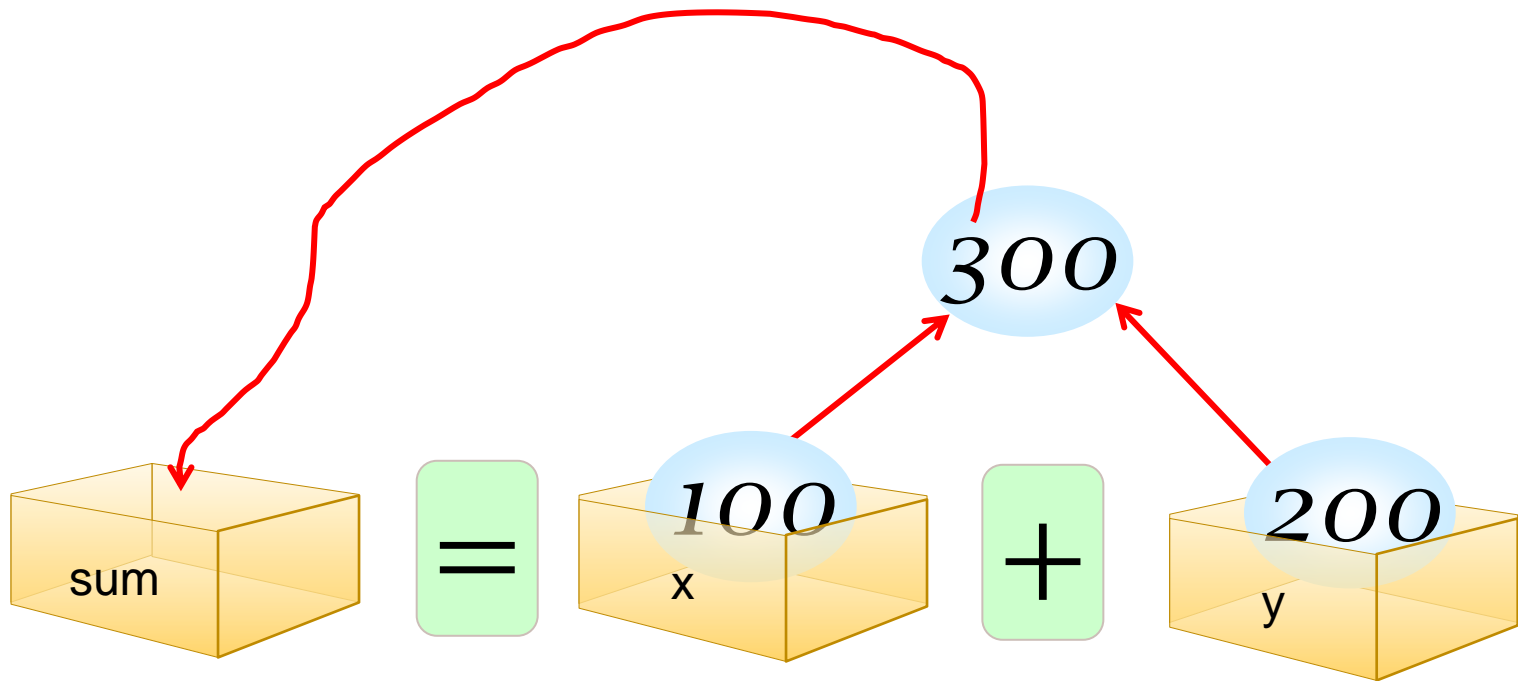
산술 연산

- 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	%	$x \% y$	$x \bmod y$



sum = x + y;





정리

프로그램

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

    x = 100;
    y = 200;

    sum = x + y;

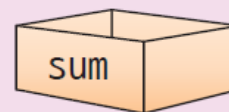
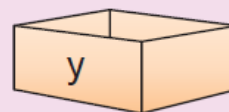
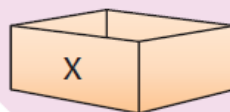
    printf("두수의 합: %d", sum);

    return 0;
}
```

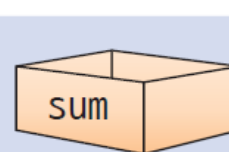
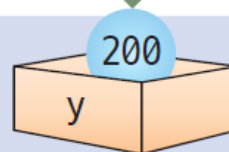
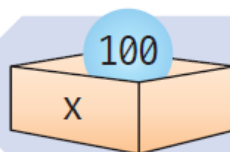
컴퓨터 내부



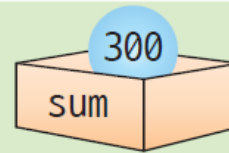
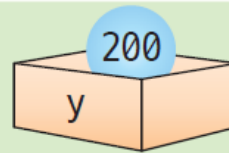
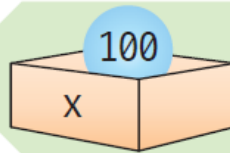
변수가 생성된다.



변수에 값이 대입된다.



덧셈 연산이 수행된다.





작업 점검

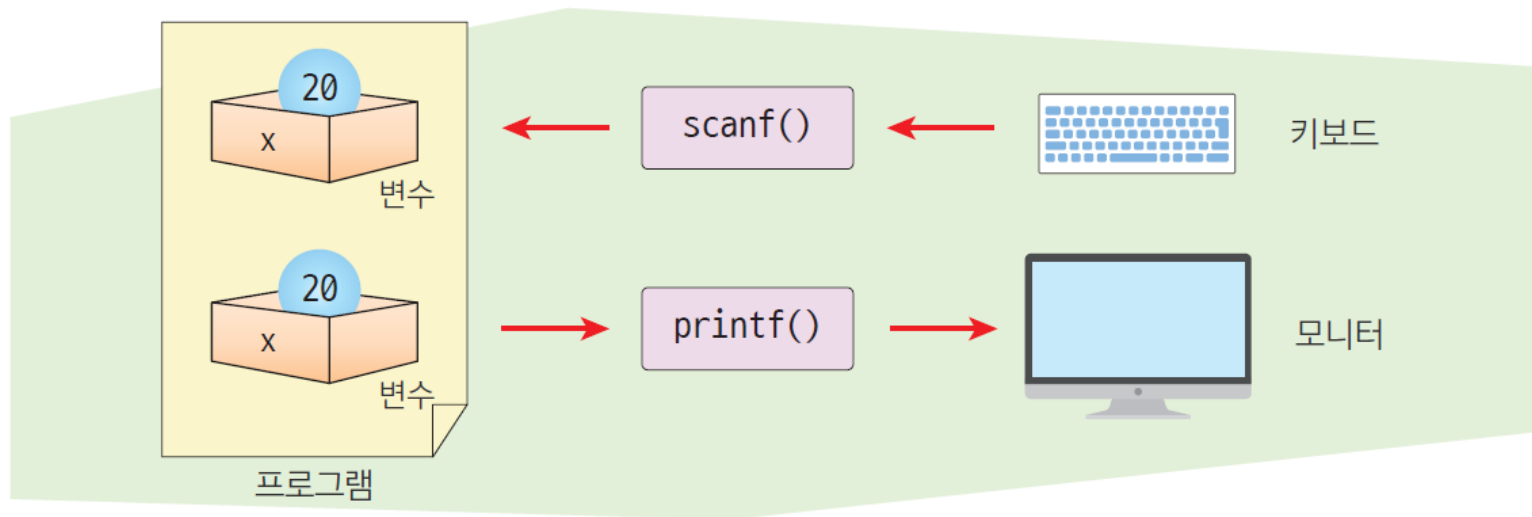
1. 함수의 중간에서 변수를 선언할 수 있는가?
2. int형 변수 `x`와 `y`를 한 줄에 선언하고 1과 0으로 각각 초기화하라.
3. 변수 `a`와 변수 `b`의 곱을 변수 `product`에 저장하는 문장을 작성하여 보자.
4. 변수 `a`를 변수 `b`로 나눈 값을 변수 `quotient`에 저장하는 문장을 작성하여 보자.





라이브러리 함수

- 라이브러리 함수: 라이브러리 함수란 컴파일러가 프로그래머가 사용할 수 있도록 제공하는 함수
- 아래 두 함수는 `<stdio.h>` 로부터 제공 받은 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





변수가
잘
출력

- 출력 형식

```
printf("두수의 합: %d \n, sum");
```





형식 지정자

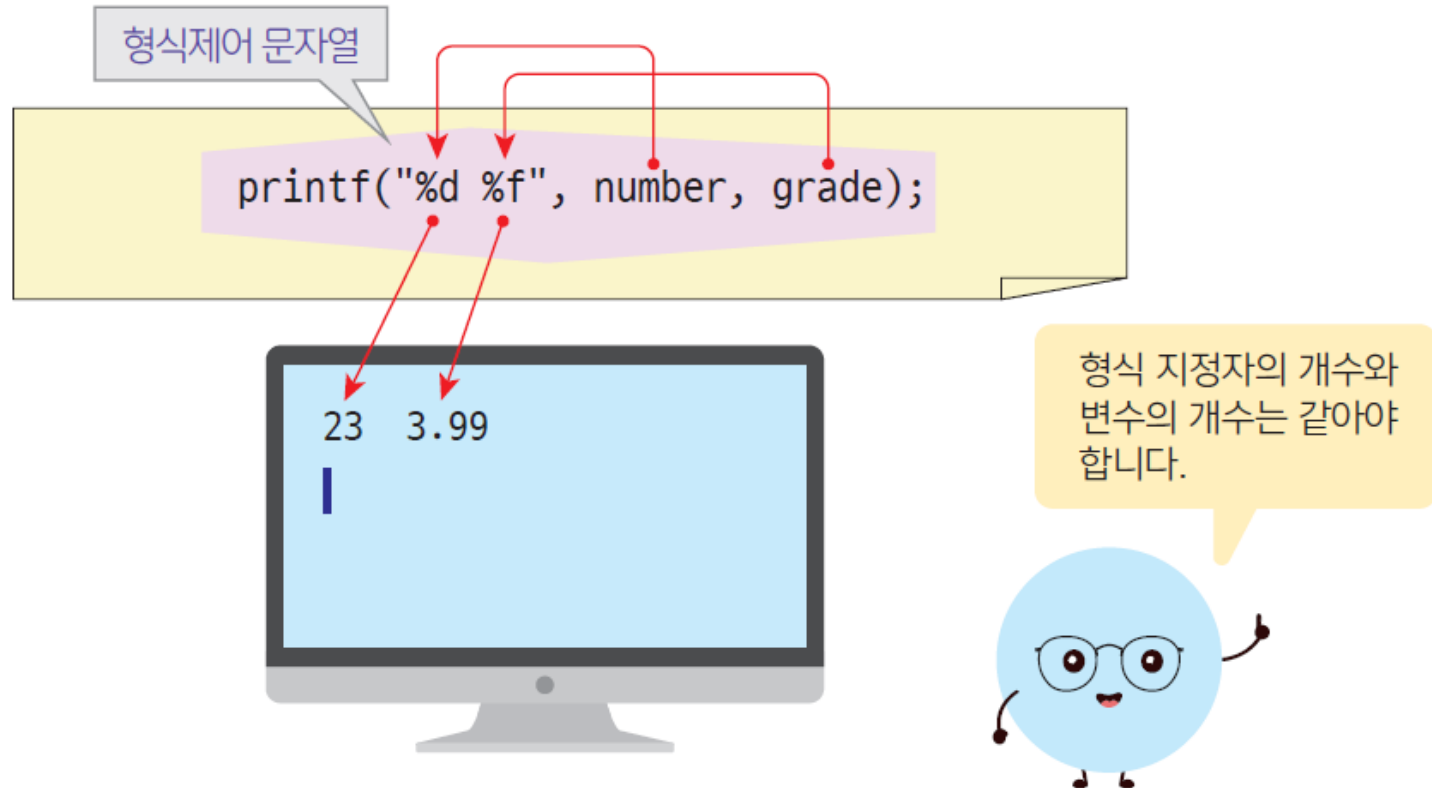
- 형식 지정자: `printf()`에서 값을 출력하는 형식을 지정한다.

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 변수가 출력

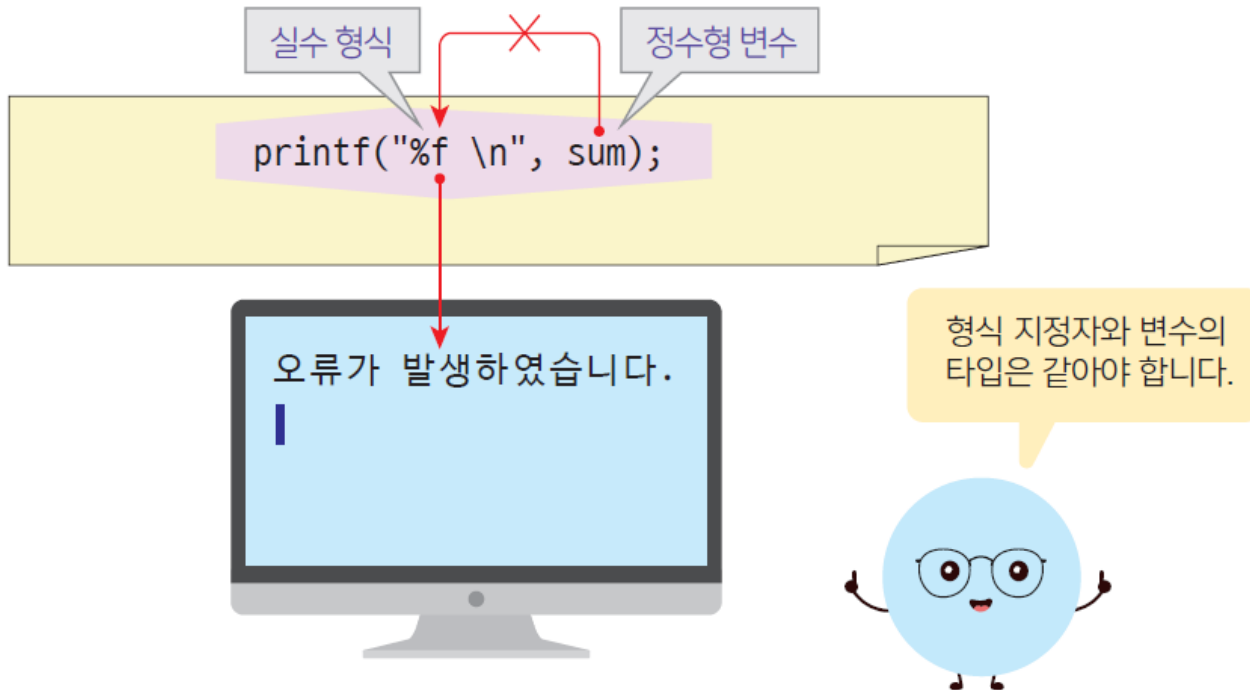
- 형식 지정자의 자리에 변수의 값이 대치되어서 출력된다고 생각하면 된다.





주의!

- 형식과 변수의 자료형은 반드시 일치하여야 한다는 점이다





필드폭(width)과 정밀도(precision)

- printf()를 사용하여 출력할 때, 데이터가 출력되는 필드의 크기를 지정할 수 있다.

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								



중간 점검

1. `printf()`에서 변수의 값을 실수 형태로 출력할 때 사용하는 형식 지정자는 무엇인가?
2. `printf()`를 사용하여 정수형 변수 `k`의 값을 출력하는 문장을 작성하여 보자.





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);
```




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()

형식제어 문자열

```
scanf("%d %f", &number, &grade);
```

23 3.99

형식 지정자의 개수와
변수의 개수는 같아야
합니다.





비주얼 스튜디오 2022에서 scanf() 오류

The screenshot shows the Visual Studio 2022 IDE with a C program named 'hello.c'. The code is as follows:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i;
6     printf("정수를 입력하시오: ");
7     scanf("%d", &i);
8     return;
9 }
```

The '오류 목록' (Error List) window at the bottom shows two errors. The first error, C6031, is a warning about a variable that is not used. The second error, C4996, is a warning about the use of the deprecated 'scanf()' function. A red box highlights the C4996 error, and a yellow lightning bolt points to it.

코드	설명	프로젝트	파일	줄
C6031	변화 값이 무시되었습니다. 'scanf'.	hello	hello.c	7
C4996	'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.	hello	hello.c	7



비주얼 스튜디오 2022에서의 scanf() 함수 오류

- scanf()는 안전하지 않으니, scanf_s()를 대신 사용하라는 오류
- scanf_s()와 같이 기존의 함수에 _s를 붙이는 안전한 함수들은 2011년도 발표된 C11의 선택적인 표준(Annex K)
- 하지만 선택적인 표준이기 때문에 비주얼 스튜디오를 제외하고는 gcc를 비롯한 다른 컴파일러에서는 아직도 활발히 도입되지 않고 있다(현장 적용 보고서에서는 ‘그다지 유익하지 않은’ 것으로 평가했다. 다음 표준에서 삭제할 것이 권고되었다)
- 결론: 소스 코드의 맨 첫 부분에 _CRT_SECURE_NO_WARNINGS를 정의하고 기존의 함수들을 그대로 사용
- stdio.h 헤더 파일을 포함하기 전에 정의하여야 함(1번 코드라인에 작성)
- 참고: <https://programmerjoon.tistory.com/4>



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_WARNINGS를
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

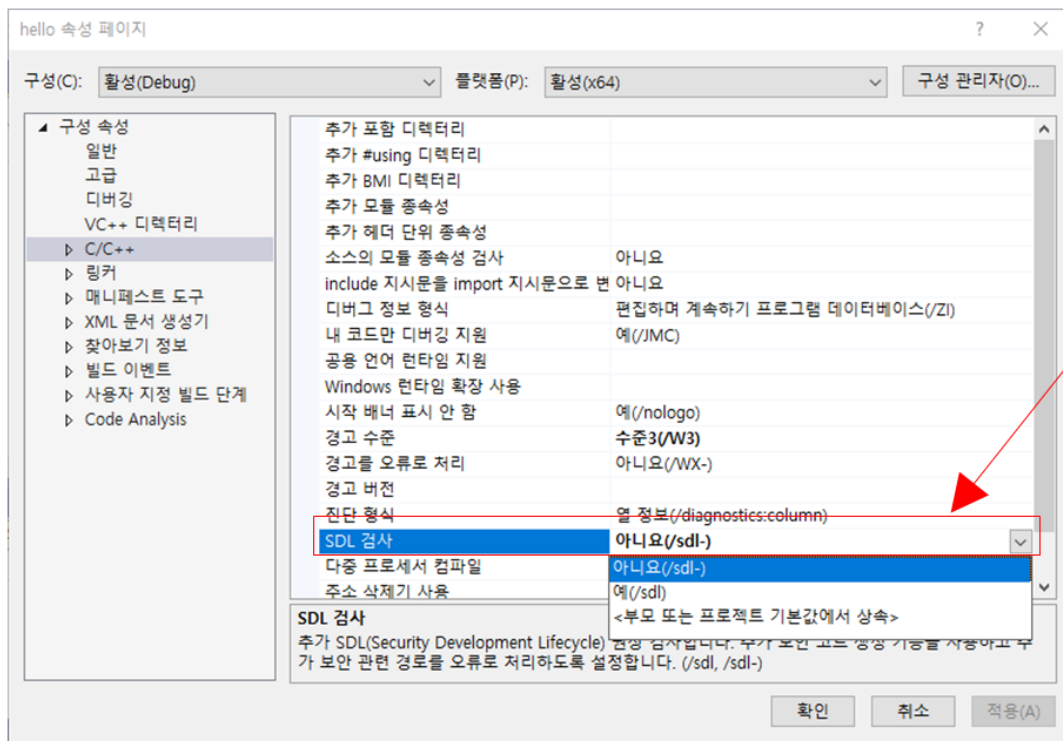
정수를 입력하시오: 20
입력된 정수 = 20

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



다른 방법

- [프로젝트]->[프로젝트 속성(P)]로 들어가서 [C/C++]→[일반]→[SDL검사]를 “아니요”로 설정하여도 된다. 각자 편리한 방법을 사용하면 된다.



SDL 검사를 “아니요”로 설정한다.



scanf() 정리

책의 소스에서 scanf() 오류가 발생하면 소스 파일의 첫부분에서 `_CRT_SECURE_NO_WARNINGS`를 정의해준다.

Windows



소스의 첫 부분에
한 줄을 추가한다.

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

Linux



소스를 그대로
사용한다.

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



값 저장

1. `scanf()`를 사용하여 사용자로 부터 실수값을 받아서 `double`형의 변수 `value`에 저장하는 문장을 작성하여 보자.





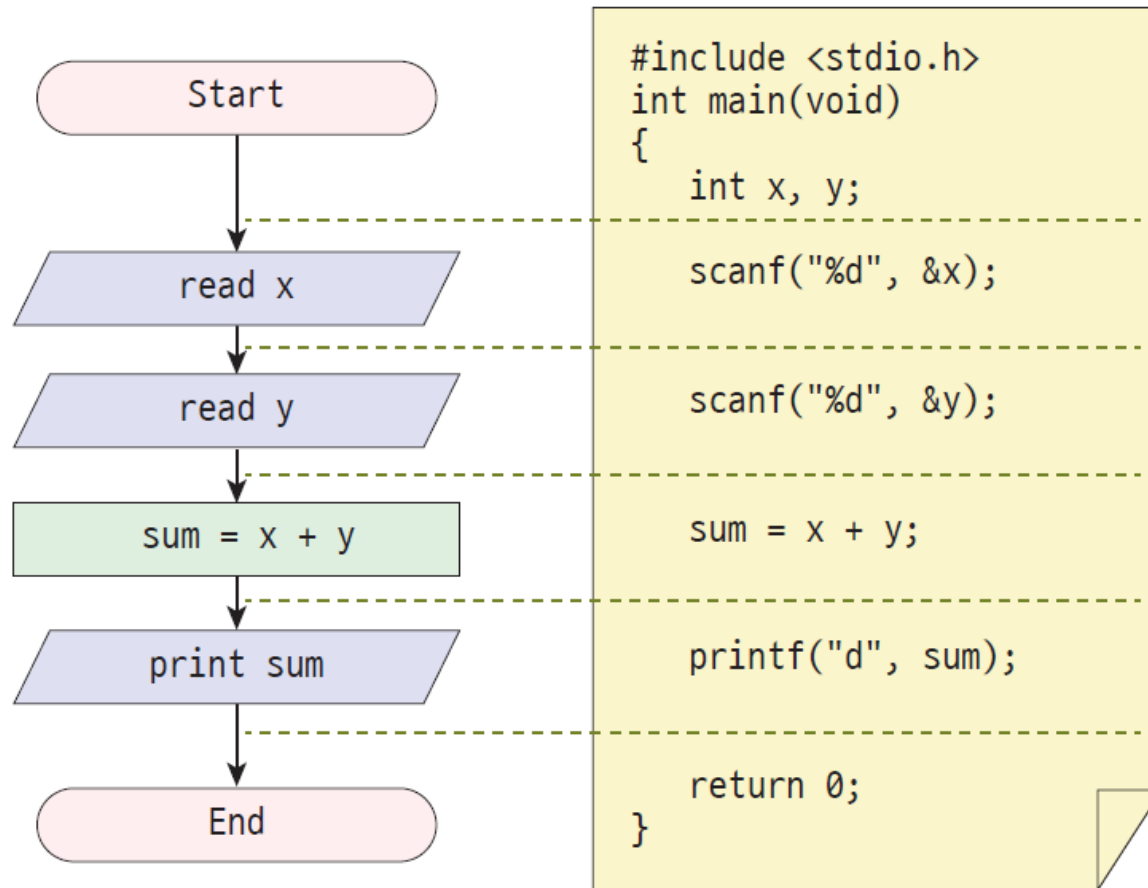
더샘 프로그램 #2_ (꼭 한번씩 해보고 이해하기)

- 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 **double**형의 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;

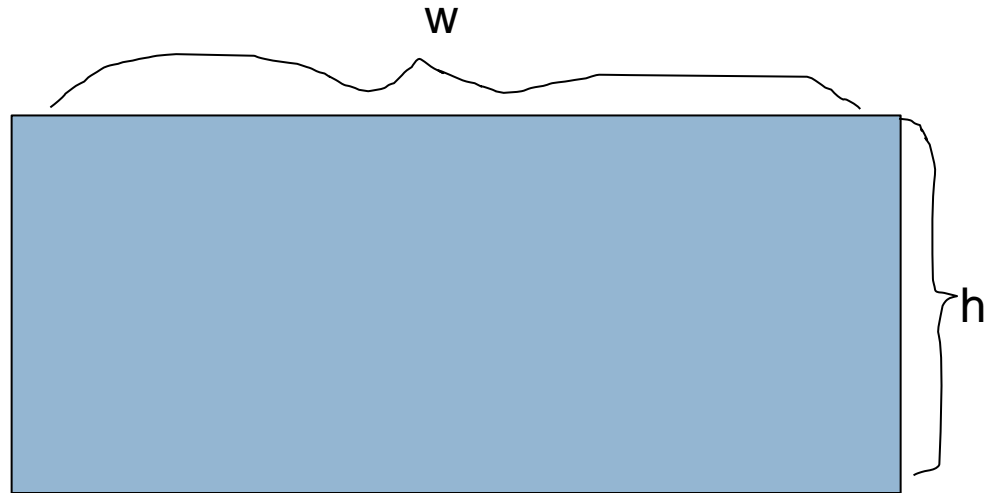
    printf("합계=%.2lf\n", sum);                          // 소수점 이하를 2자리로 표시
    printf("평균=%.2lf\n", avg);

    return 0;
}
```



Mini Project: 사각형의 둘레와 면적

- 필요한 변수는 w , h , $area$, $perimeter$ 라고 하자.
- 변수의 자료형은 실수를 저장할 수 있는 **double**형으로 하자.
- $area = w * h$;
- $perimeter = 2 * (w + h)$;





프로그램의 실행 화면

사각형의 넓이: 50.000000
사각형의 둘레: 30.000000



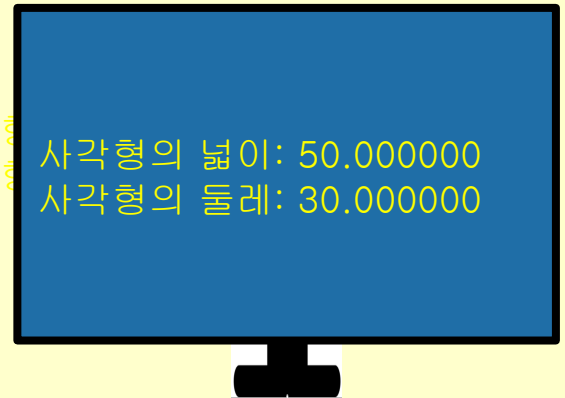
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main(void)
{
    double w;
    double h;
    double area;
    double perimeter;

    w = 10.0;
    h = 5.0;
    area = w * h;
    perimeter = 2 * (w + h);

    printf("사각형의 넓이: %lf\n", area);
    printf("사각형의 둘레: %lf\n", perimeter);
    return 0;
}
```

사각형의
사각형의





도전문제

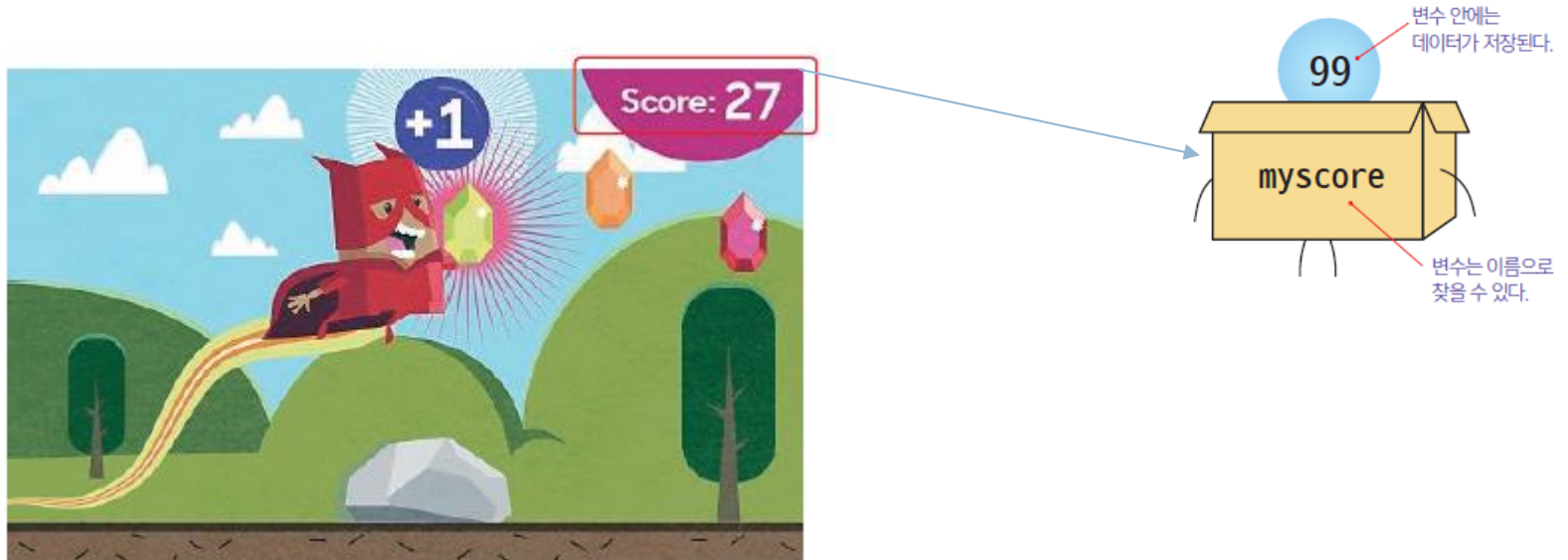
1. 한번의 `printf()` 호출로 변수 `perimeter`와 `area`의 값이 동시에 출력되도록 변경하라.
2. 변수들을 한 줄에 모두 선언하여 보자.
3. `w`와 `h`의 값을 사용자로부터 받도록 변경하여 보자. `%lf`를 사용한다.





변수

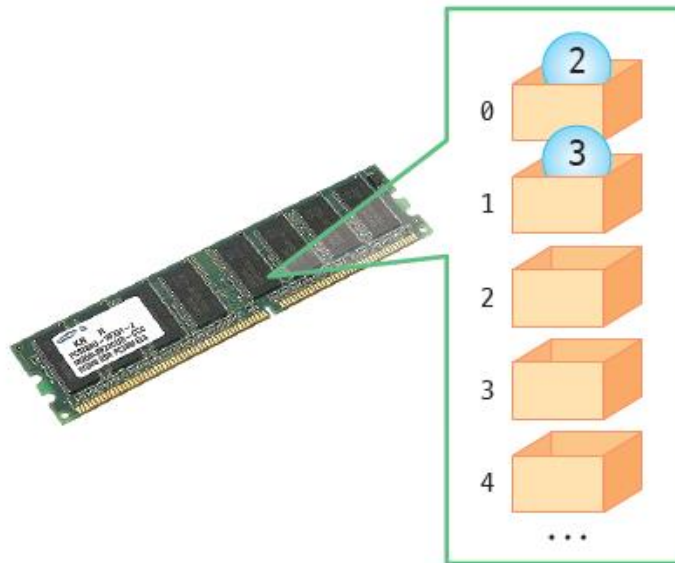
- 컴퓨터 프로그램은 값을 저장하기 위하여 변수(variable)를 사용한다.
- 변수는 게임에서 점수를 저장하는데 사용될 수 있고, 대형 마트에서 우리가 구입한 물건들의 가격을 저장할 수도 있다.





변수는 어디에 만들어지는가?

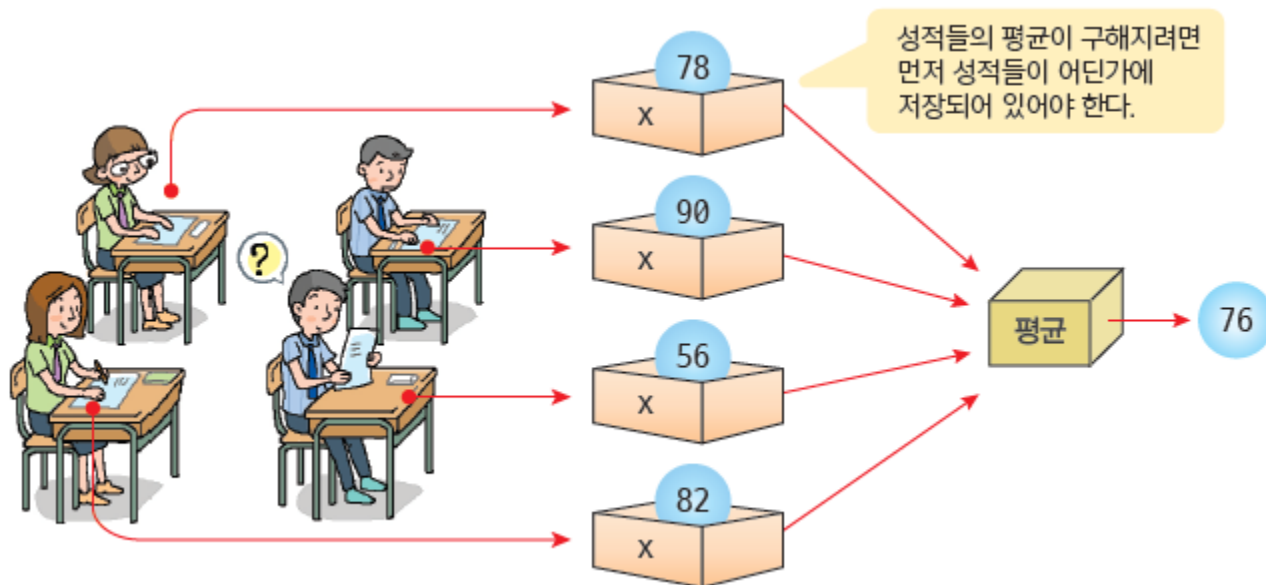
- 변수는 바로 메인 메모리(main memory)에 만들어진다.
- 우리는 변수 이름을 사용하여 메모리 공간을 사용하게 된다





변수가 왜 필요한가? #1

- 사용자에게서 받는 데이터를 저장하는 장소이다. – 변수가 없다면 사용자로부터 받은 데이터를 어디에 저장할 것인가?





변수가 왜 필요한가? #2

변수를 사용하지 않는 코드	변수를 사용하는 코드
<pre>// 크기가 100×200인 사각형의 면적 area = 100 * 200;</pre>	<pre>// 크기가 width×height인 사각형의 면적 width = 100; height = 200; area = width * height;</pre>

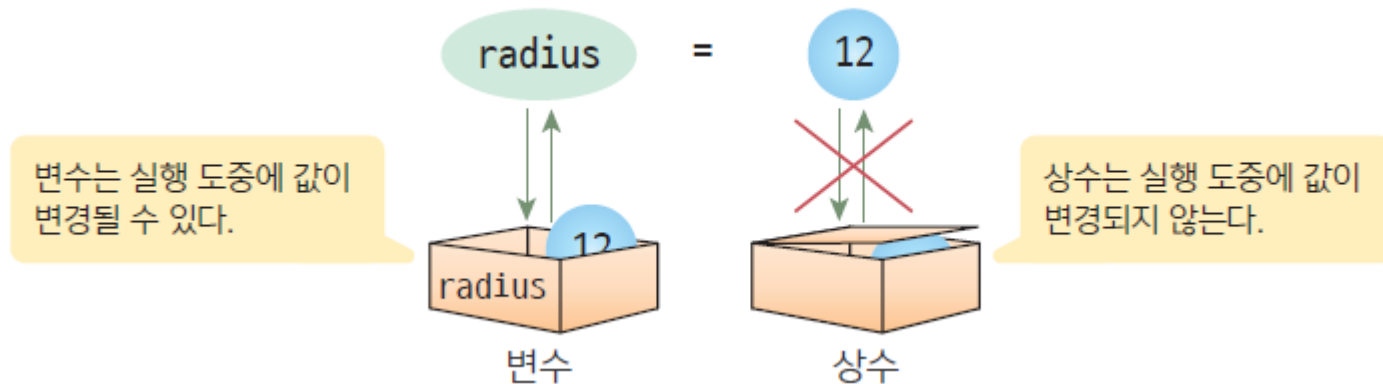
어떤 코드가 더
유연한가요?
변경에 더 잘 적응할 수
있나요?





변수와 상수

- **변수(variable)**: 저장된 값의 변경이 가능한 공간
- **상수(constant)**: 저장된 값의 변경이 불가능한 공간
 - (예) 3.14, 100, 'A', "Hello World!"





예제: 변수와 상수

/* 원의 면적을 계산하는 프로그램 */

#include <stdio.h>

int main(void)

{

float radius;
float area;

// 원의 반지름
// 원의 면적

printf("원의 면적을 입력하세요:");
scanf("%f", &radius);

area = 3.141592 * radius * radius;
printf("원의 면적: %f \n", area);

return 0;

}

상수

scanf() 오류가 발생하면 소스의

#define _CRT_NO_SECURE_W

을 넣으세요.





공간 점검

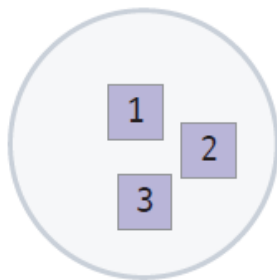
1. 변수와 상수는 어떻게 다른가?
2. 변수가 실제로 만들어 지는 공간은 컴퓨터 부품 중에서 어디인가?



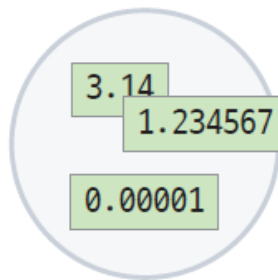


자료형

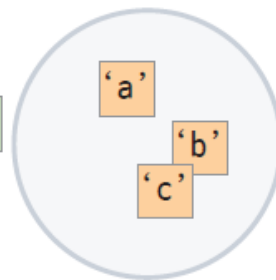
- 자료형(data type): 데이터의 타입(종류)
 - short, int, long: 정수형 데이터(100)
 - double, float: 부동소수점형 데이터(3.141592)
 - char: 문자형 데이터('A', 'a', '한')



정수형



부동 소수점형



문자형





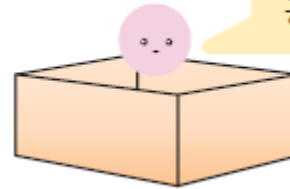
다양한 자료형이 필요한 이유

- 상자에 물건을 저장하는 것과 같다.

물건이 상자보다 크면
들어가지 않을 것이다.



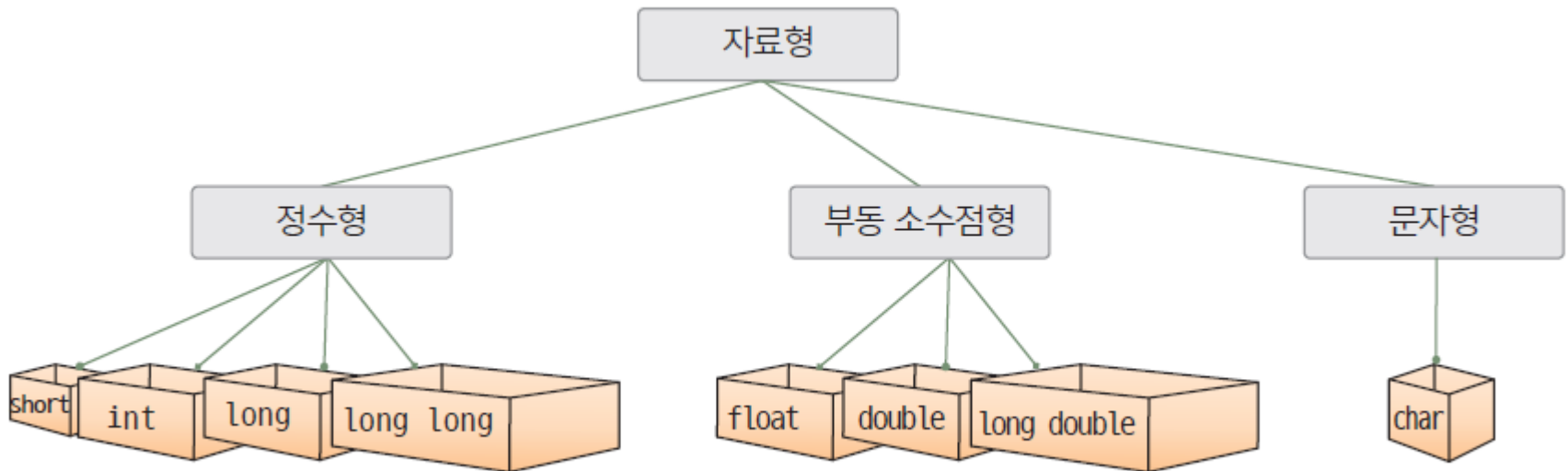
물건이 상자보다 너무 작으면
공간이 낭비될 것이다.





자료형의 분류

- 자료형을 크게 나누면 정수형(integer type), 부동소수점형(floating-point type), 문자형(character type)으로 나눌 수 있다.





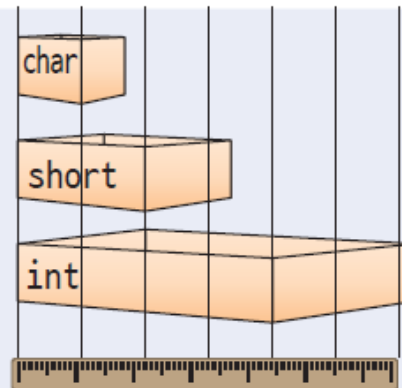
자료형의 크기

- 자료형의 크기를 알아보려면 **sizeof** 연산자를 사용하면 된다. **sizeof**는 변수나 자료형의 크기를 바이트 단위로 반환하는 연산자이다.

Syntax sizeof()

예

```
sizeof(x)      // 변수  
sizeof(10)     // 값  
sizeof(int)    // 자료형  
sizeof(double) // 자료형
```



sizeof 연산자는 변수나
자료형의 크기를 바이트
단위로 반환합니다.





예제: 자료형의 크기

```
#include <stdio.h>

int main(void)
{
    int x;

    printf("변수x의   크기: %d\n", sizeof(x));

    printf("char형의   크기: %d\n", sizeof(char));
    printf("int형의   크기: %d\n", sizeof(int));
    printf("short형의  크기: %d\n", sizeof(short));
    printf("long형의   크기: %d\n", sizeof(long));
    printf("float형의  크기: %d\n", sizeof(float));
    printf("double형의 크기: %d\n", sizeof(double));

    return 0;
}
```

변수x의 크기: 4
char형의 크기: 1
int형의 크기: 4
short형의 크기: 2
long형의 크기: 4
float형의 크기: 4
double형의 크기: 8

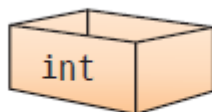


정수형

자료형		비트	범위
정수형	short	16비트	-32768~32767
	int	32비트	-2147483648~2147483647
	long		
	long long	64비트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807



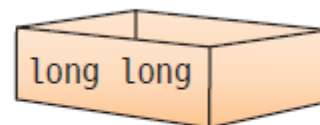
≤



≤



≤



16비트(2바이트)

≤

32비트(4바이트)

≤

32비트(4바이트)

≤

64비트(8바이트)



C에서는 왜 이렇게 많은 종류의 정수형이 있을까?

- 용도에 따라 프로그래머가 선택하여 사용할 수 있게 하려는 것이다.
- 비트수를 늘리면 정수의 범위는 확대시킬 수 있지만 메모리 공간을 더 많이 필요로 한다.



어떤 크기던지
돌릴 수 있습니다.
하지만 복잡하고
크기가 큼니다.



크기에 맞추어
선택하여
사용합니다.



정수형의 범위

약 -21억에서
+21억

- int형

$-2^{31}, \dots, -2, -1, 0, 1, 2, \dots, 2^{31} - 1$
(-2147483648 ~ +2147483647)

- short형

$-2^{15}, \dots, -2, -1, 0, 1, 2, \dots, 2^{15} - 1$
(-32768 ~ +32767)

- long형

- 보통 int형과 같음





예제

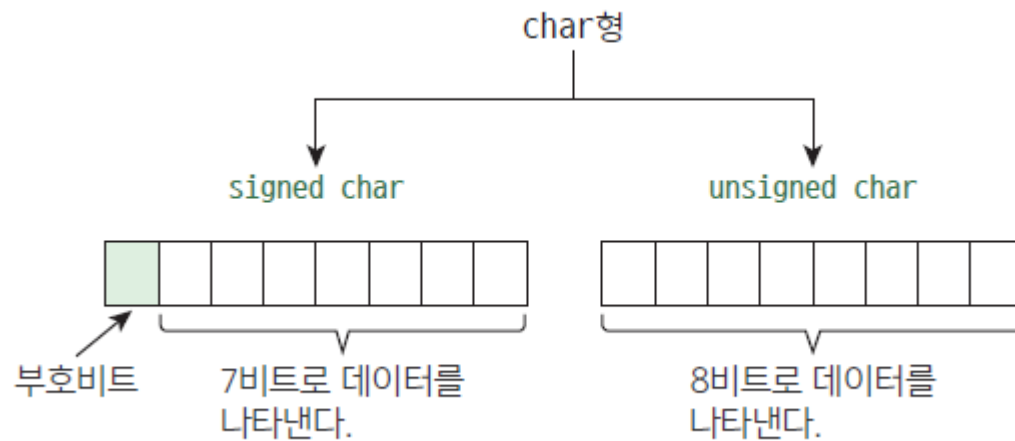
```
/* 정수형 자료형의 크기를 계산하는 프로그램*/  
#include <stdio.h>  
  
int main(void)  
{  
    short year = 0;           // 0으로 초기화한다.  
    int sale = 0;             // 0으로 초기화한다.  
    long total_sale = 0;      // 0으로 초기화한다.  
    long long large_value;    // 64비트 자료형  
  
    year = 10;                // 약 3만2천을 넘지 않도록 주의  
    sale = 2000000000;        // 약 21억을 넘지 않도록 주의  
    total_sale = year * sale; // 약 21억을 넘지 않도록 주의  
  
    printf("total_sale = %d \n", total_sale);  
  
    return 0;  
}
```

total_sale = 2000000000



signed, unsigned 수식자

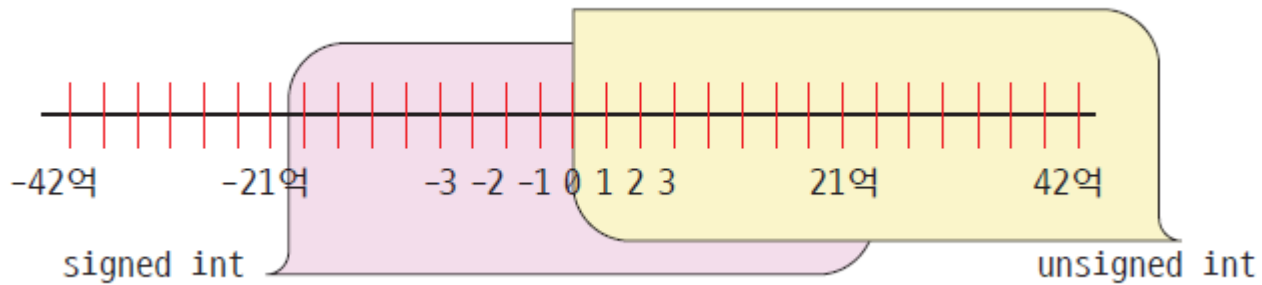
- unsigned
 - 음수가 아닌 값만을 나타냄을 의미
 - unsigned int
- signed
 - 부호를 가지는 값을 나타냄을 의미
 - 흔히 생략





unsigned int

$0, 1, 2, \dots, 2^{32} - 1$
(0 ~ +4294967295)





unsigned 예제

```
unsigned int  speed;           // 부호없는 int형
unsigned      distance;        // unsigned int distance와 같다.
unsigned short players;        // 부호없는 short형
```

```
unsigned int sales = 2800000000; // 약 28억
printf("%u \n", sales);          // %d를 사용하면 음수로 출력된다
```

unsigned는
%u로
출력하세요.





오버플로우

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

int main(void)
{
    short s_money = SHRT_MAX;           // 최대값으로 초기화한다. 32767
    unsigned short u_money = USHRT_MAX; // 최대값으로 초기화한다. 65535

    s_money = s_money + 1;
    printf("s_money = %d", s_money);

    u_money = u_money + 1;
    printf("u_money = %d", u_money);
    return 0;
}
```

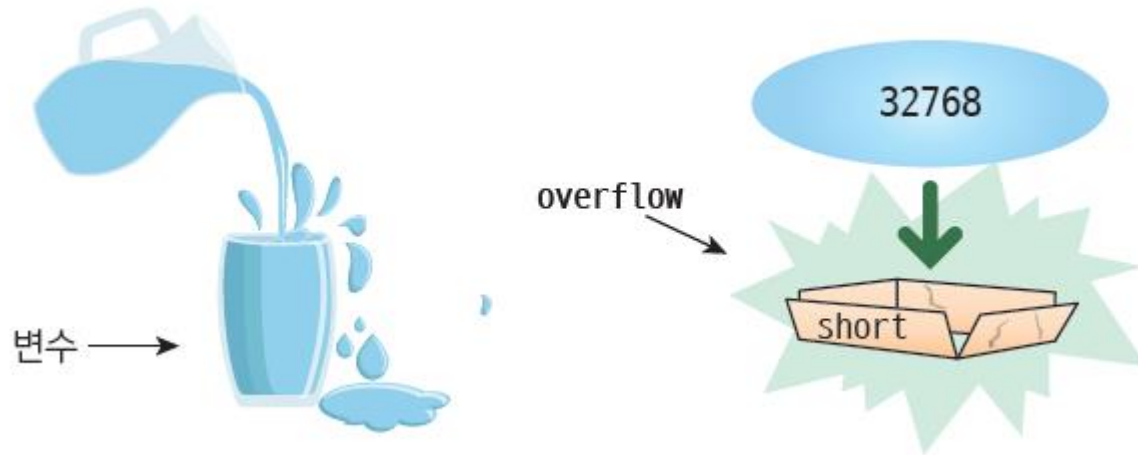
오버플로우 발생!!

S_money = -32768
U_money = 0



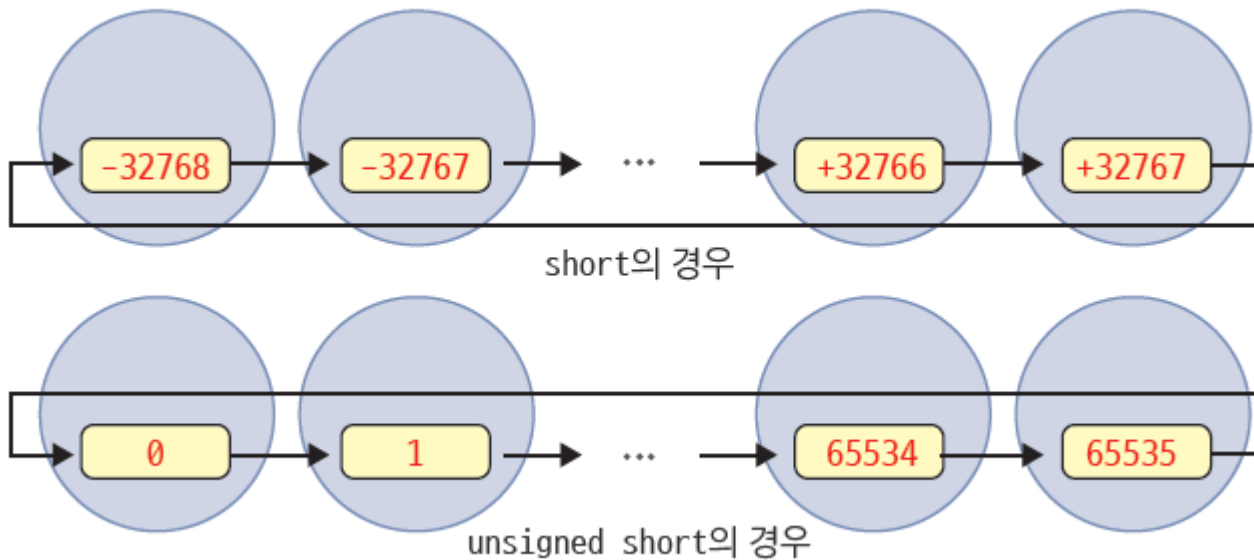
오버플로우

- **오버플로우(overflow)**: 변수가 나타낼 수 있는 범위를 넘는 숫자를 저장하려고 할 때 발생





- 규칙성이 있다.
 - 수도 계량기나 자동차의 주행거리계와 비슷하게 동작





참고

참고사항

각 자료형의 최대값과 최소값은 limits.h에 정의되어 있다.

```
#define CHAR_MIN    (-128)
#define CHAR_MAX    127

#define SHRT_MIN    (-32768)        /* minimum (signed) short value */
#define SHRT_MAX    32767          /* maximum (signed) short value */
#define USHRT_MAX   0xffff         /* maximum unsigned short value */

#define INT_MIN     (-2147483647 - 1) /* minimum (signed) int value */
#define INT_MAX     2147483647       /* maximum (signed) int value */
#define UINT_MAX    0xffffffff      /* maximum unsigned int value */
```



정수 상수

- 숫자를 적으면 기본적으로 int형이 된다.
 - `sum = 123;` `// 123은 int형`
- 상수의 자료형을 명시하려면 다음과 같이 한다.
 - `sum = 123L;` `// 123은 long형`

접미사	자료형	예
u 또는 U	unsigned int	123u 또는 123U
l 또는 L	long	123l 또는 123L
ul 또는 UL	unsigned long	123ul 또는 123UL



10진법, 8진법, 16진법

- 8진법

- $012_8 = 1 \times 8^1 + 2 \times 8^0 = 10$

- 16진법

- $0xA_{16} = 10 \times 16^0 = 10$

10진수	8진수	16진수
0	00	0x0
1	01	0x1
2	02	0x2
3	03	0x3
4	04	0x4
5	05	0x5
6	06	0x6
7	07	0x7
8	010	0x8
9	011	0x9
10	012	0xa
11	013	0xb
12	014	0xc
13	015	0xd
14	016	0xe
15	017	0xf
16	020	0x10
17	021	0x11
18	022	0x12



예제

```
/* 정수 상수 프로그램*/  
#include <stdio.h>  
  
int main(void)  
{  
    int x = 10; // 10은 10진수이고 int형이고 값은 십진수로 10이다.  
    int y = 010; // 010은 8진수이고 int형이고 값은 십진수로 8이다.  
    int z = 0x10; // 010은 16진수이고 int형이고 값은 십진수로 16이다.  
  
    printf("x = %d", x);  
    printf("y = %d", y);  
    printf("z = %d", z);  
  
    return 0;  
}
```

x = 10
y = 8

x = 10
y = 8
z = 16



기호 상수

- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것
- 용법 : `#define EXCHANGE_RATE 1120`
- 전처리기 지시자(`#include` 와 비슷하게 생각)로 일반적으로 매크로라고 표현
- (예)
 - `won = 1120 * dollar;` // (1) 실제의 값을 사용
 - `won → EXCHANGE_RATE * dollar;` // (2) 기호상수 사용
- 기호 상수의 장점
 - 가독성이 높아진다.
 - 값을 쉽게 일괄 변경할 수 있다.



기호 상수의 장점

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

```
int main(void)
```

```
{
```

```
...
```

```
won1 = 120 * dollar1;
```

```
won2 = 120 * dollar2;
```

```
...
```

```
}
```

1050

1050

리터럴 상수를 사용하는 경우:
등장하는 모든 곳을 수정하여야 한다.

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

```
#define EXCHANGE_RATE 120
```

```
int main(void)
```

```
{
```

```
...
```

```
won1 = EXCHANGE_RATE * dollar1;
```

```
won2 = EXCHANGE_RATE * dollar2;
```

```
...
```

```
}
```

1050

기호 상수를 사용하는 경우:
기호 상수가 정의된 곳만 수정하면 한다.



기호 상수를 만드는 방법 #1

Syntax

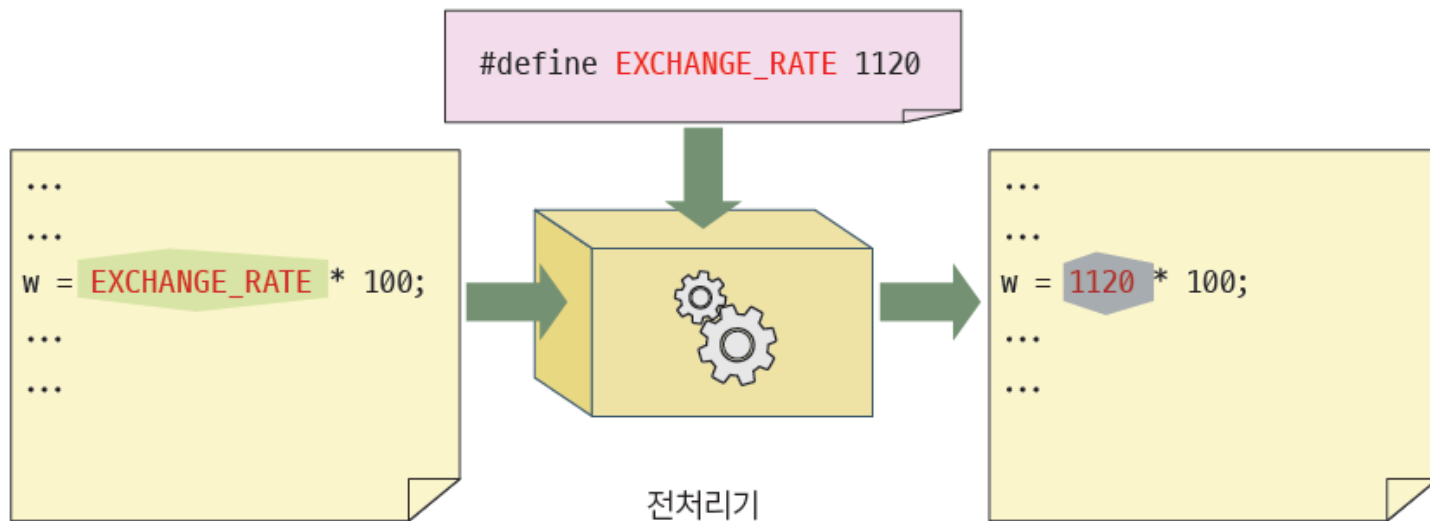
기호상수선언

예

```
#define EXCHANGE_RATE 1120
```

기호상수

값





기호 상수를 만드는 방법 #2

Syntax

기호상수선언

예

```
const int EXCHANGE_RATE = 1120;
```

기호상수

값



예제: 기호 상수

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

```
#define TAX_RATE 0.2
```

기호상수

```
int main(void)
```

```
{
```

```
    const int MONTHS = 12;
```

```
    int m_salary, y_salary;
```

// 변수 선언

```
    printf( "월급을 입력하시요: "); // 입력 안내문
```

```
    scanf("%d", &m_salary);
```

```
    y_salary = MONTHS * m_salary; // 순수입 계산
```

```
    printf("연봉은 %d입니다.", y_salary);
```

```
    printf("세금은 %f입니다.", y_salary*TAX_RATE);
```

```
    return 0;
```

```
}
```

월급을 입력하시요: 100
연봉은 1200입니다.
세금은 240.000000입니다.



정수 점검

1. 정수형에 속하는 자료형을 모두 열거하라.
2. 숫자값을 직접 사용하는 것보다 기호 상수를 사용하는 것의 이점은 무엇인가?
3. 왜 정수를 하나의 타입으로 하지 않고 **int**, **short**, **long** 등의 여러 가지 타입으로 복잡하게 분류하여 사용하는가?
4. 부호가 없는 **unsigned int**형의 변수에 음수를 넣으면 어떤 일이 벌어지는가?
5. 변수가 저장할 수 있는 한계를 넘어서는 값을 저장하면 어떻게 되는가? 구체적인 예로 **short**형의 변수에 **32768**을 저장하면 어떻게 되는가?





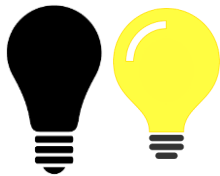
정수 표현 방법

- 컴퓨터에서 정수는 이진수 형태로 표현되고 이진수는 전자 스위치로 표현된다.

스위치가 **하나** 있으면



0, 1



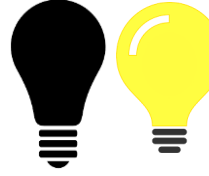
스위치가 **둘** 있으면



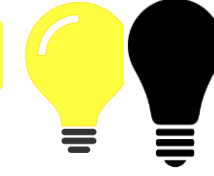
00,



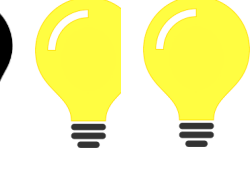
01,



10,



11





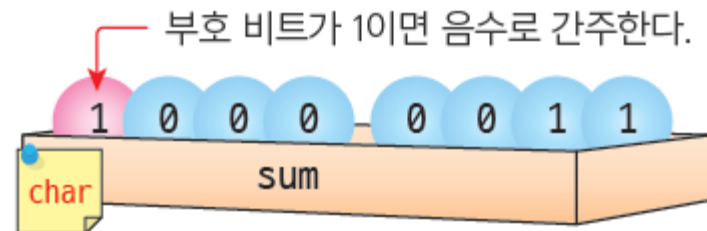
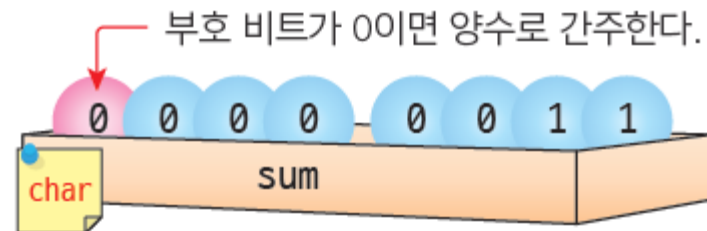
short형

비트 패턴	정수	비고
0000000000000000	0	양의 정수
0000000000000001	1	
0000000000000010	2	
0000000000000011	3	
0000000000000100	4	
0000000000000101	5	
...	...	



정수 표현 방법

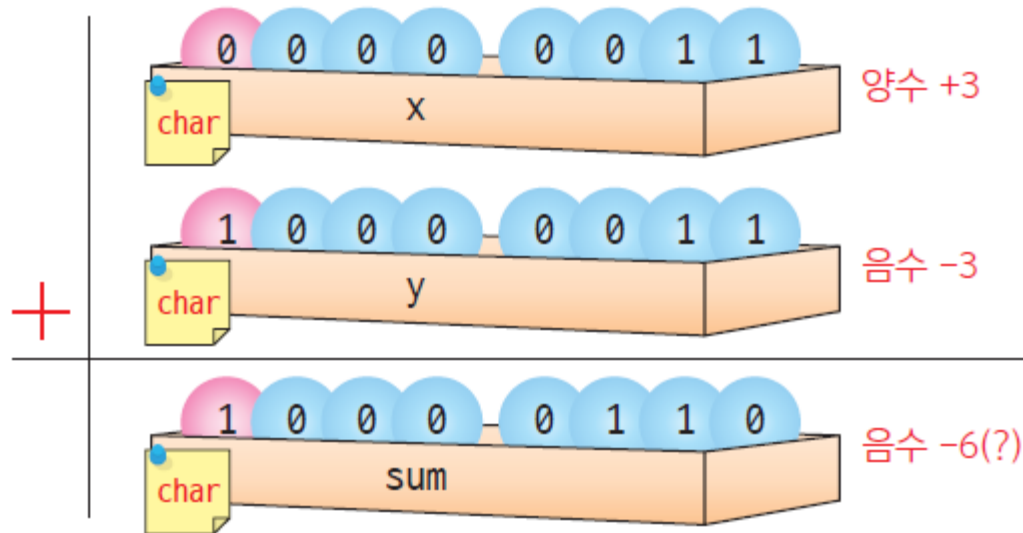
- 양수
 - 십진수를 이진수로 변환하여 저장하면 된다.
- 음수
 - 보통은 첫번째 비트를 부호 비트로 사용한다.
 - 문제점이 발생한다.





음수를 표현하는 첫번째 방법

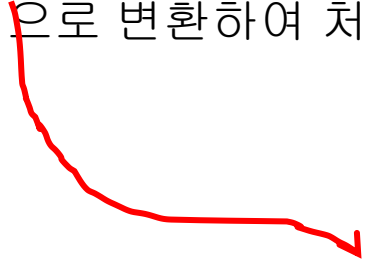
- 첫번째 방법은 맨 처음 비트를 부호 비트로 간주하는 방법입니다.
- 양수와 음수의 덧셈 연산을 하였을 경우, 결과가 부정확하다.
 - (예) $+3 + (-3)$





컴퓨터는 덧셈만 할 수 있다

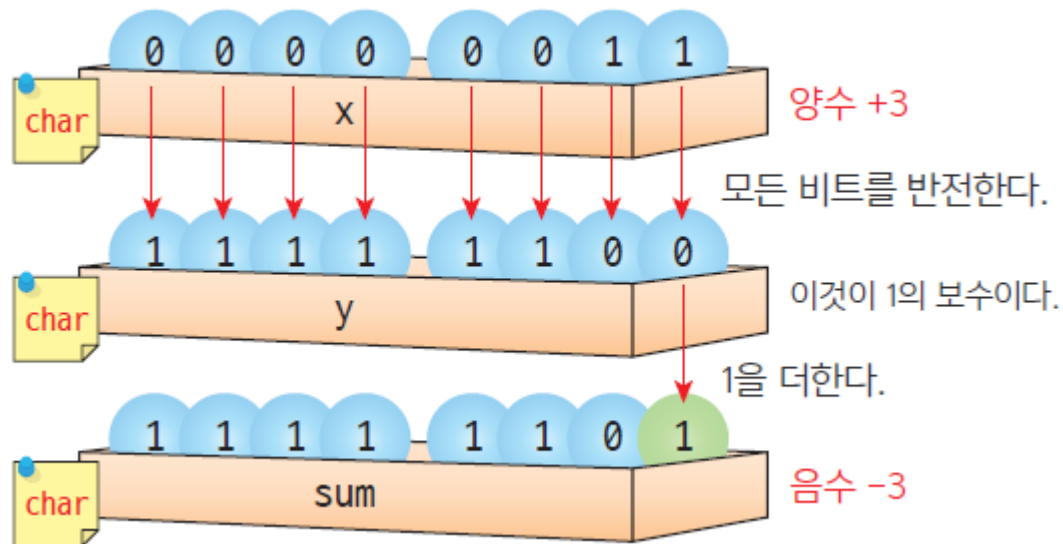
- 컴퓨터는 회로의 크기를 줄이기 위하여 덧셈회로만을 가지고 있다.
- 뺄셈은 다음과 같이 덧셈으로 변환하여 처리한다.


$$3-3 = 3+(-3)$$



음수를 표현하는 두번째 방법

- N의 보수 : 어떤 수가 N이 되기 위해 부족한 수만큼 보충 해야하는 수
- 2의 보수로 음수를 표현한다. -> 표준적인 음수 표현 방법
- 2의 보수를 만드는 방법



모든 비트의 0과 1을 바꾼 뒤(1의 보수로 만든 뒤)
1을 더해주면 부호를 바꿀 수 있다.(2의 보수로 만든다)



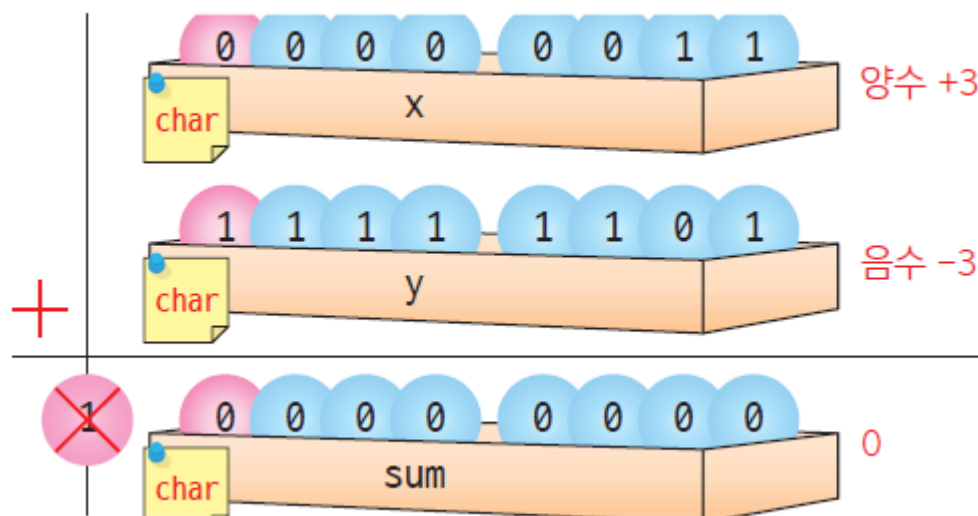
2의 보수

비트	부호없는 정수	부호있는 정수 (2의 보수)
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

비트	부호없는 정수	부호있는 정수 (2의 보수)
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
0111 1110	126	126
0111 1111	127	127
1000 0000	128	-128
1000 0001	129	-127
1000 0010	130	-126
1111 1110	254	-2
1111 1111	255	-1



2의 보수로 양수와 음수를 더하면



음수를 2의 보수로 표현하면 양수와 음수를 더할 때 각각의 비트들을 더하면 됩니다.





예제

```
/* 2의 보수 프로그램*/  
#include <stdio.h>
```

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

```
    int x = 3;
```

```
    int y = -3;
```

음수가 2의 보수로
표현되는지를 알아보자.

```
    printf("x = %08X\n", x);
```

// 8자리의 16진수로 출력한다.

```
    printf("y = %08X\n", y);
```

// 8자리의 16진수로 출력한다.

```
    printf("x+y = %08X\n", x+y);
```

// 8자리의 16진수로 출력한다.

```
    return 0;
```

```
}
```

```
x = 00000003  
y = FFFFFFFD  
x+y = 00000000
```



중간 점검

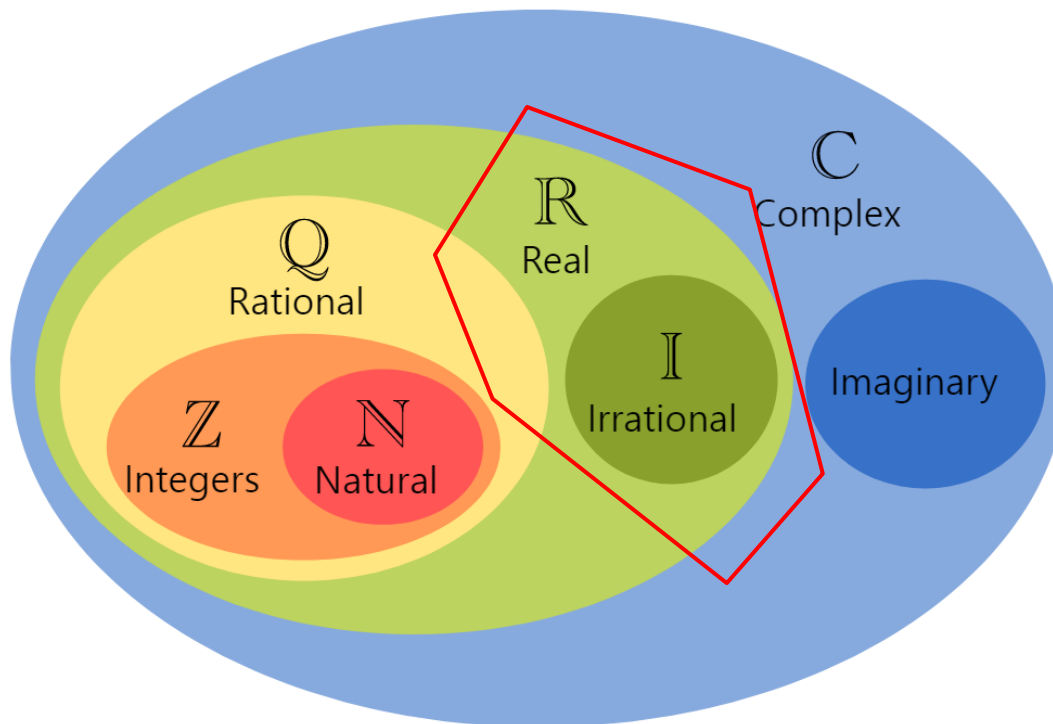
1. 음수의 표현 방법으로 2의 보수를 사용하는 이유는 무엇인가?
2. 이진수 01000011의 1의 보수를 구해보자.
3. 이진수 01000011의 2의 보수를 구해보자.





실수를 나타내는 방법

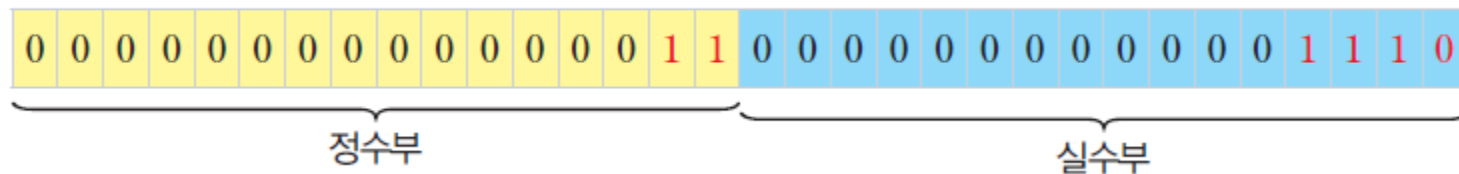
- 수학에서의 실수는 3.14와 같이 소수점을 가진 수이다. 실수는 매우 큰 수나 매우 작은 수를 다루는 과학이나 공학 분야의 응용 프로그램을 작성할 때는 없어서는 안 될 중요한 요소이다.





실수를 표현하는 방법 #1

- 정수 부분을 위하여 일정 비트를 할당하고 소수 부분을 위하여 일정 비트를 할당
- 전체가 32비트이면 정수 부분 16비트, 소수 부분 16비트 할당
- 과학과 공학에서 필요한 아주 큰 수를 표현할 수 없다



이 방법으로
나타낼 수 있는
최대수는
얼마일까요?





실수를 표현하는 방법 #2

- 부동 소수점 방식

부호비트(1비트)



s

e

지수부분(8비트)

m

가수부분(23비트)

$$149598000 = 1.49598 \times 10^8$$

$$\text{실수값} = (-1)^s * (1.m) * 2^{e-127}$$

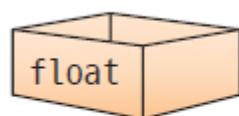
- 표현할 수 있는 범위가 대폭 늘어난다.
- 10^{-38} 에서 10^{+38}

한동안 아주 다양한 부동소수점 방법이 사용되었지만 1985년부터는 IEEE 754 로 표준화되었습니다.



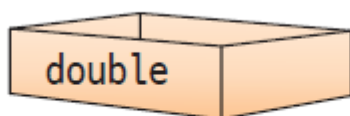


부동 소수점 형



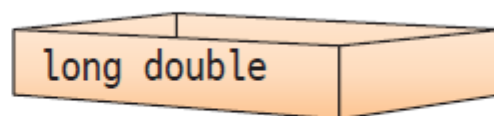
32비트

<=



64비트

<=



64비트

자료형	명칭	크기	범위
float	단일 정밀도(single-precision) 부동 소수점	32비트	$\pm 1.17549 \times 10^{-38} \sim \pm 3.40282 \times 10^{+38}$
double long double	두배 정밀도(double-precision) 부동 소수점	64비트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$



실수를 출력하는 형식 지정자

- %f
 - printf(“%f”, 0.123456789); // 0.123457 출력
- %e
 - printf(“%e”, 0.123456789); // 1.234568e-001 출력



예제

```
/* 부동 소수점 자료형의 크기 계산*/  
#include <stdio.h>  
int main(void)  
{  
    float x = 1.234567890123456789;  
    double y = 1.234567890123456789;  
  
    printf("float의 크기=%d\n", sizeof(float));  
    printf("double의 크기=%d\n", sizeof(double));  
  
    printf("x = %30.25f\n", x);  
    printf("y = %30.25f\n", y);  
    return 0;  
}
```

float의 크기=4

double의 크기=8

x = 1.23456788063049320000000000

y = 1.23456789012345670000000000



부동 소수점 상수

실수	지수 표기법	의미
123.45	1.2345e2	1.2345×10^2
12345.0	1.2345e4	1.2345×10^4
0.000023	2.3e-5	2.3×10^{-5}
2,000,000,000	2.0e9	2.0×10^9

1.23456

2.

.28

2e+10

9.26E3

0.67e-9

// 소수점만 붙여도 된다.

// 정수부가 없어도 된다.

// +나 -기호를 지수부에 붙일 수 있다.

// 9.26×10^3

// 0.67×10^{-9}



부동소수점 오버플로우

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

```
int main(void)
```

```
{
```

```
float x = 1e39;
```

```
printf("x = %e\n",x);
```

```
}
```

숫자가 커서 오버플로우
발생

x = inf

계속하려면 아무 키나 누르십시오 . . .



부동 소수점 언더플로우

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

```
int main(void)
```

```
{
```

```
    float x = 1.23456e-38;
```

```
    float y = 1.23456e-40;
```

```
    float z = 1.23456e-46;
```

```
    printf("x = %e\n",x);
```

```
    printf("y = %e\n",y);
```

```
    printf("z = %e\n",z);
```

```
}
```

숫자가 작아서
언더플로우 발생

```
x = 1.234560e-038
```

```
y = 1.234558e-040
```

```
z = 0.000000e+000
```



부동소수점형 사용시 주의사항

- 오차가 있을 수 있다!

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

```
int main(void)
```

```
{
```

```
    float value = 0.1;
```

```
    printf("%.20f \n", value);
```

출력하라는 의미이다.

```
    return 0;
```

```
}
```

이진법으로는 정확하게 나타낼 수 없는 값들이 있기 때문이다. 0.1도 그 중의 하나이다.

// %.20f는 소수점 이하를 20자리로

0.10000000149011611938



부동소수점 연산 사용시 주의사항

- 오차가 있을 수 있다!

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

```
int main(void)
```

```
{
```

```
    double x;
```

```
    x = (1.0e20 + 5.0) - 1.0e20;
```

```
    printf("%f \n", x);
```

```
    return 0;
```

```
}
```

부동소수점 연산에서는
오차가 발생한다.
5.0이 아니라 0으로 계산
된다.

0.000000



정간 점검

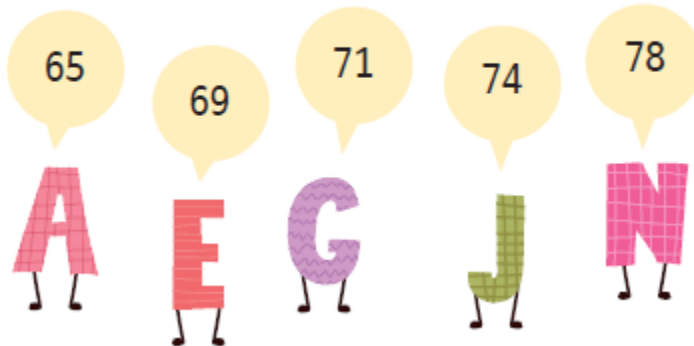
1. float형과 double형의 크기는?
2. 부동 소수점 상수인 1.0×10^{25} 를 지수 형식으로 표기하여 보자.
3. 부동 소수점형에서 오차가 발생하는 근본적인 이유는 무엇인가?





문자형

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현
- 공통적인 규격이 필요하다.
- 아스키 코드(**ASCII**: American Standard Code for Information Interchange)





아스키 코드표 (일부)

Dec	Hex	문자
0	0	NULL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOL
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	HT
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3

Dec	Hex	문자
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US
32	20	space
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	'

Dec	Hex	문자
40	28	(
41	29)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;

Dec	Hex	문자
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

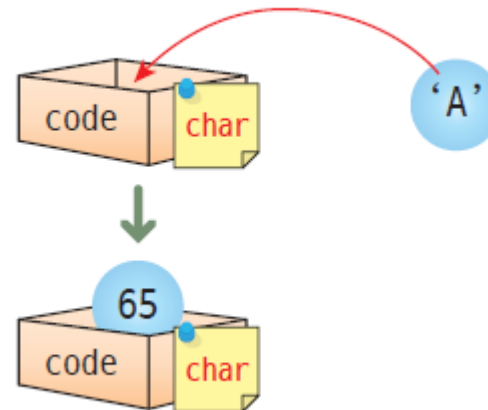
[illegible]



문자 변수

- char형을 사용하여 문자를 저장한다.

```
char code;  
code = 'A';
```





예제

```
/* 문자 변수와 문자 상수*/  
#include <stdio.h>  
  
int main(void)  
{  
    char code1 = 'A';    // 문자 상수로 초기화  
    char code2 = 65;     // 아스키 코드로 초기화  
  
    printf("code1 = %c\n", code1);  
    printf("code2 = %c\n", code2);  
}
```

```
code1 = A  
code2 = A
```



Q&A

1
Q A 문자가 정수로 표현된다면 char형의 변수가 문자를 저장하는지, 정수를 저장하는지를 어떻게 구별하는가?

char형의 변수에 저장된 값을 문자로 해석하면 문자라고 간주된다. 예를 들어서 화면에 출력할 때 %c를 사용하여 출력하면 변수에 들어 있는 값을 아스키 코드로 해석한다. 반면에 %d를 사용하면 문자가 아니고 정수로 해석한다.



Tip: 한글 표현 방법

한글 표현 방법

한글은 8비트로 표현할 수 없다. 글자의 개수가 영문자에 비하여 많기 때문이다. 컴퓨터에서 한글을 표현하는 방식은 크게 2가지 방법이 있다. 첫 번째는 이는 각 글자마다 하나의 코드를 부여하는 것이다. 예를 들어서 '가'에는 0xb0a1이라는 코드를 부여하는 것이다. 한글에서 표현 가능한 글자의 개수는 11172개이고 따라서 이들 글자에 코드를 부여하려면 8비트($2^8=256$)로는 부족하고, 16비트($2^{16}=65536$)가 되어야 가능하다. 이것을 완성형이라고 한다. 대표적인 코드 체계가 유니코드(unicode)이다. 유니코드(unicode)는 전세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준이다. 유니코드 협회(unicode consortium)가 제정하며, 현재 최신판은 유니코드 10.0이다. 이 표준에는 문자 집합, 문자 인코딩, 문자 정보 데이터베이스, 문자들을 다루기 위한 알고리즘 등이 포함된다. 또 하나의 방법은 똑같이 16비트를 사용하는 방법이지만 글자의 초성, 중성, 종성에 각각 5비트씩을 할당하고, 가장 앞의 비트는 영숫자와 한글을 구분 짓는 기호로 하는 방법이다. 즉 맨 처음 비트가 1이면 한글, 0이면 영숫자로 판단하는 것이다. 이런 식으로 한글을 표현하는 방법을 조합형이라고 한다.

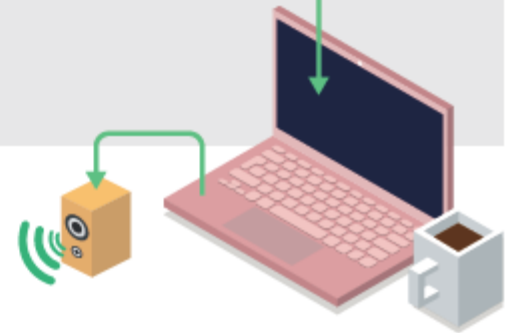


제어 문자

- 인쇄 목적이 아니라 제어 목적으로 사용되는 문자들
 - (예) 줄바꿈 문자, 탭 문자, 벨소리 문자, 백스페이스 문자

```
char beep = 7;  
printf("%c", beep);
```

```
printf("%c", 7);
```

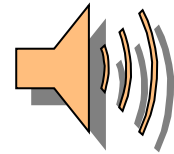




제어 문자를 나타내는 방법

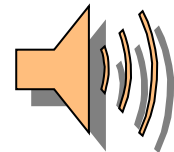
- 아스키 코드를 직접 사용

```
char beep = 7;  
printf("%c", beep);
```



- 이스케이프 시퀀스 사용

```
char beep = '\a';  
printf("%c", beep);
```





이스케이프 시퀀스

제어 문자	이름	의미
\0	널문자	
\a	경고(bell)	"삐"하는 경고음 발생
\b	백스페이스(backspace)	커서를 현재의 위치에서 한 글자 뒤로 옮긴다.
\t	수평탭(horizontal tab)	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
\n	줄바꿈(newline)	커서를 다음 라인의 시작 위치로 옮긴다.
\v	수직탭(vertical tab)	설정되어 있는 다음 수직 탭 위치로 커서를 이동
\f	폼피드(form feed)	주로 프린터에서 강제로 다음 페이지로 넘길 때 사용된다.
\r	캐리지 리턴(carriage return)	커서를 현재 라인의 시작 위치로 옮긴다.
\"	큰따옴표	원래의 큰따옴표 자체
\'	작은따옴표	원래의 작은따옴표 자체
\\	역슬래시(back slash)	원래의 역슬래시 자체



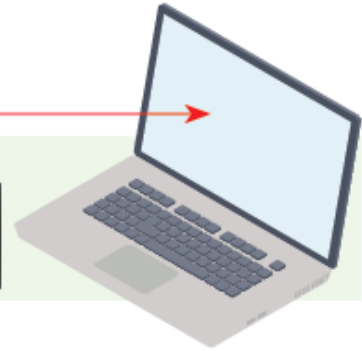
프로그램에서 경고음을 내려면?

특수 문자열을 사용하여 프로그램에서 경고음을 내려면 다음과 한다.

```
char beep = '\a';  
printf("%c", beep);
```

```
printf("\a");
```

beep





역슬래시 \

- 특수한 기능을 가진 문자 앞에 역슬래시 \를 위치시키면 문자의 특수한 의미가 사라진다.
- 키보드 백스페이스 왼쪽의 원화 표시와 같음

```
printf("\나만의 할리우드\" UCC 열풍 ");
```



“나만의 할리우드” UCC 열풍

```
printf("\\는 제어 문자를 표시할 때 사용한다. ");
```



\는 제어 문자를 표시할 때 사용한다.



예제

```
#include <stdio.h>
int main(void)
{
    int id, pass;

    printf("아이디와 패스워드를 4개의 숫자로 입력하세요:\n");

    printf("id: ____\b\b\b\b");
    scanf("%d", &id);

    printf("pass: ____\b\b\b\b");
    scanf("%d", &pass);
    printf("\a입력된 아이디는 \"%d\"이고 패스워드는 \"%d\"입니다.", id, pass);

    return 0;
}
```

아이디와 패스워드를 4개의 숫자로 입력하세요:
id: 1234
pass: 5678
입력된 아이디는 "1234"이고 패스워드는 "5678"입니다.



정수형으로서의 char형

- 8비트의 정수를 저장하는데 char 형을 사용할 수 있다..

```
#include <stdio.h>

int main(void)
{
    char code = 'A';
    printf("%d %d %d \n", code, code + 1, code + 2); // 65 66 67이 출력된다.
    printf("%c %c %c \n", code, code + 1, code + 2); // A B C가 출력된다.
    return 0;
}
```



```
65 66 67
A B C
```



주간 점검

1. 컴퓨터에서는 문자를 어떻게 나타내는가?
2. C에서 문자를 가장 잘 표현할 수 있는 자료형은 무엇인가?
3. 경고음이 발생하는 문장을 작성하여 보자.





Lab: 변수의 초기값

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

    printf("3개의 정수를 입력하세요 (x, y, z): ");
    scanf("%d %d %d", &x, &y, &z);
    sum += x;
    sum += y;
    sum += z;
    printf("3개 정수의 합은 %d\n", sum);
    return 0;
}
```

Microsoft Visual C++ Runtime Library



Debug Error!

Program: ..\documents\visual studio
2017\Projects\hello\Debug\hello.exe
Module: ..\documents\visual studio
2017\Projects\hello\Debug\hello.exe
File:

Run-Time Check Failure #3 - The variable 'sum' is being used
without being initialized.

(Press Retry to debug the application)

종단(A)

다시 시도(R)

무시(I)



무엇이 문제일까?

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

```
int main(void)
```

```
{
```

```
    int x, y, z, sum;
```

```
    sum = 0;
```

```
    printf("3개의 정수를 입력하세요 (x, y, z): ");
```

```
    scanf("%d %d %d", &x, &y, &z);
```

```
    sum += x;
```

```
    sum += y;
```

```
    sum += z;
```

```
    printf("3개 정수의 합은 %d\n", sum);
```

```
    return 0;
```

```
}
```

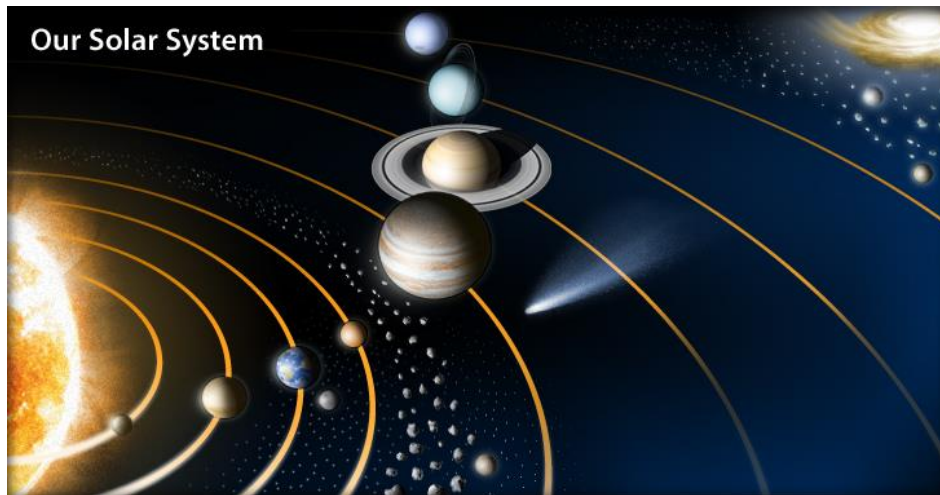
변수는 사용하기 전에 반드시 초기화
시켜야 함!

3개의 정수를 입력하세요 (x, y, z): 10 20 30
3개 정수의 합은 60



Mini Project: 태양빛 도달 시간

- 태양에서 오는 빛이 몇 분 만에 지구에 도착하는지를 컴퓨터로 계산해보고자 한다.
- 빛의 속도는 1초에 30만 km를 이동한다.
- 태양과 지구 사이의 거리는 약 1억 4960만 km이다.





실험 결과

빛의 속도는 300000.000000km/s
태양과 지구와의 거리 $149600000.000000\text{km}$
도달 시간은 498.666667초



힌트

- 문제를 해결하기 위해서는 먼저 필요한 변수를 생성하여야 한다. 여기서는 빛의 속도, 태양과 지구 사이의 거리, 도달 시간을 나타내는 변수가 필요하다.
- 변수의 자료형은 모두 실수형이어야 한다. 왜냐하면 매우 큰 수들이기 때문이다.
- 빛이 도달하는 시간은 (도달 시간 = 거리 / (빛의 속도))으로 계산할 수 있다.
- 실수형을 `printf()`로 출력할 때는 `%f`나 `%lf`를 사용한다.



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

```
int main(void)
```

```
{
```

```
    double light_speed = 300000;
```

```
// 빛의 속도 저장하는 변수
```

```
    double distance = 149600000;
```

```
// 태양과 지구 사이 거리 저장하는 변수
```

```
// 149600000km로 초기화한다.
```

```
    double time;
```

```
// 시간을 나타내는 변수
```

```
    time = distance / light_speed;
```

```
// 거리를 빛의 속도로 나눈다.
```

```
    time = time / 60.0;
```

```
// 초를 분으로 변환한다.
```

```
    printf("빛의 속도는 %fkm/s \n", light_speed);
```

```
    printf("태양과 지구와의 거리 %fkm \n", distance);
```

```
    printf("도달 시간은 %f초\n", time); // 시간을 출력한다.
```

```
    return 0;
```

```
}
```

```
빛의 속도는 300000.000000km/s
```

```
태양과 지구와의 거리 149600000.000000km
```

```
도달 시간은 498.666667초
```



Q & A

