

개정3판

Visual
Studio
2017

쉽게 풀어쓴



언어

EXPRESS




천인국 지음

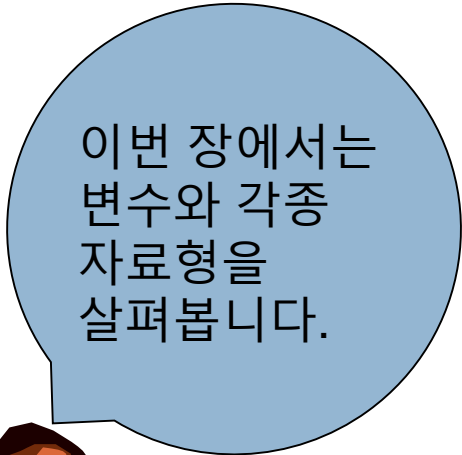
제4장 변수와 자료형



이번 장에서 학습할 내용

2

- 
- * 변수와 상수의 개념 이해
 - * 자료형
 - * 정수형
 - * 실수형
 - * 문자형
 - * 기호 상수 사용
 - * 오버플로우와 언더플로우 이해



이번 장에서는
변수와 각종
자료형을
살펴봅니다.





변수

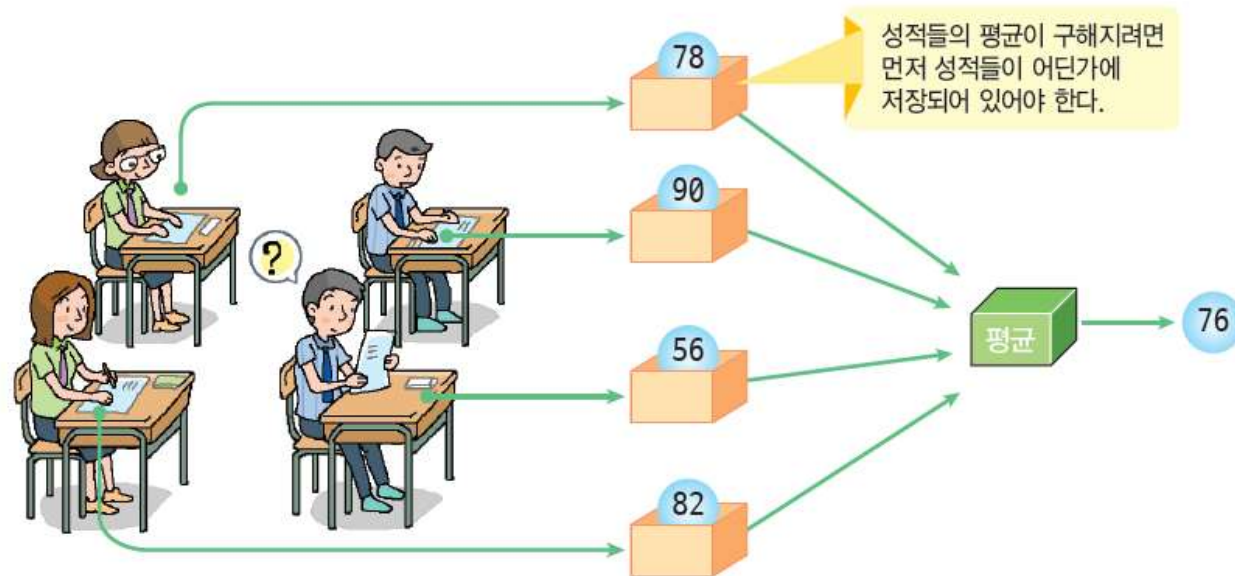
3

Q) 변수(variable)이란 무엇인가?

A) 프로그램에서 일시적으로 데이터를 저장하는 공간

Q) 변수는 왜 필요한가?

A) 데이터가 입력되면 어딘가에 저장해야만 다음에 사용할 수 있다.





변수를 사용하는 이유

4

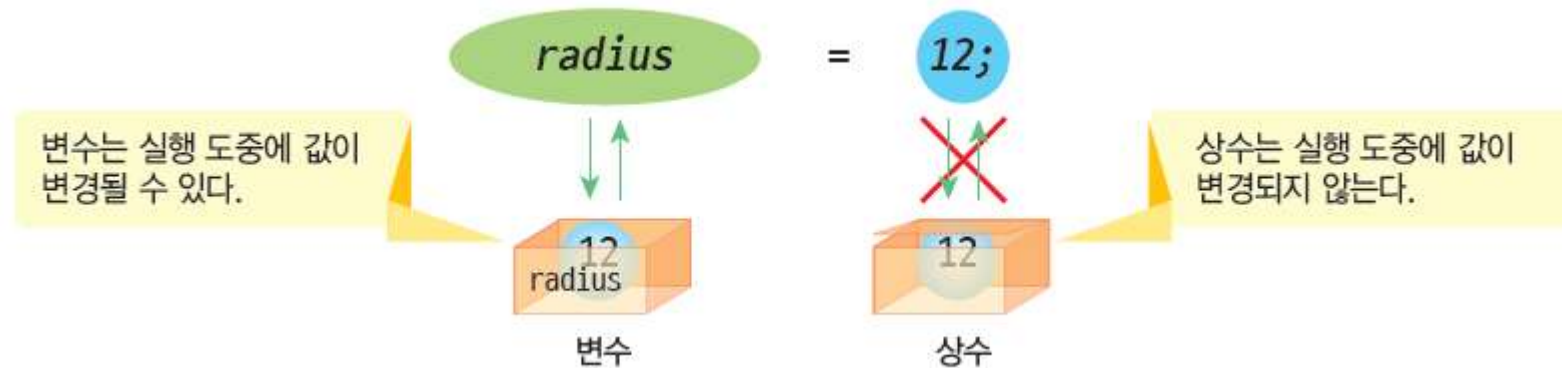
변수를 사용하지 않는 코드	변수를 사용하는 코드
<pre>// 크기가 100x200인 사각형의 면적 area = 100 * 200;</pre>	<pre>// 크기가 widthxheight인 사각형의 면적 width = 100; height = 200; area = width * height;</pre>



변수와 상수

5

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





예제: 변수와 상수

circle_area.c

6

```
/* 원의 면적을 계산하는 프로그램 */
#include <stdio.h>
int main(void)
{
    float radius;    // 원의 반지름
    float area;      // 원의 면적

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

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

    return 0;
}
```

변수

상수



자료형

7

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





다양한 자료형이 필요한 이유

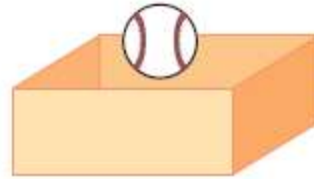
8

- 상자에 물건을 저장하는 것과 같다.

물건이 상자보다 크면
들어가지 않을 것이다.



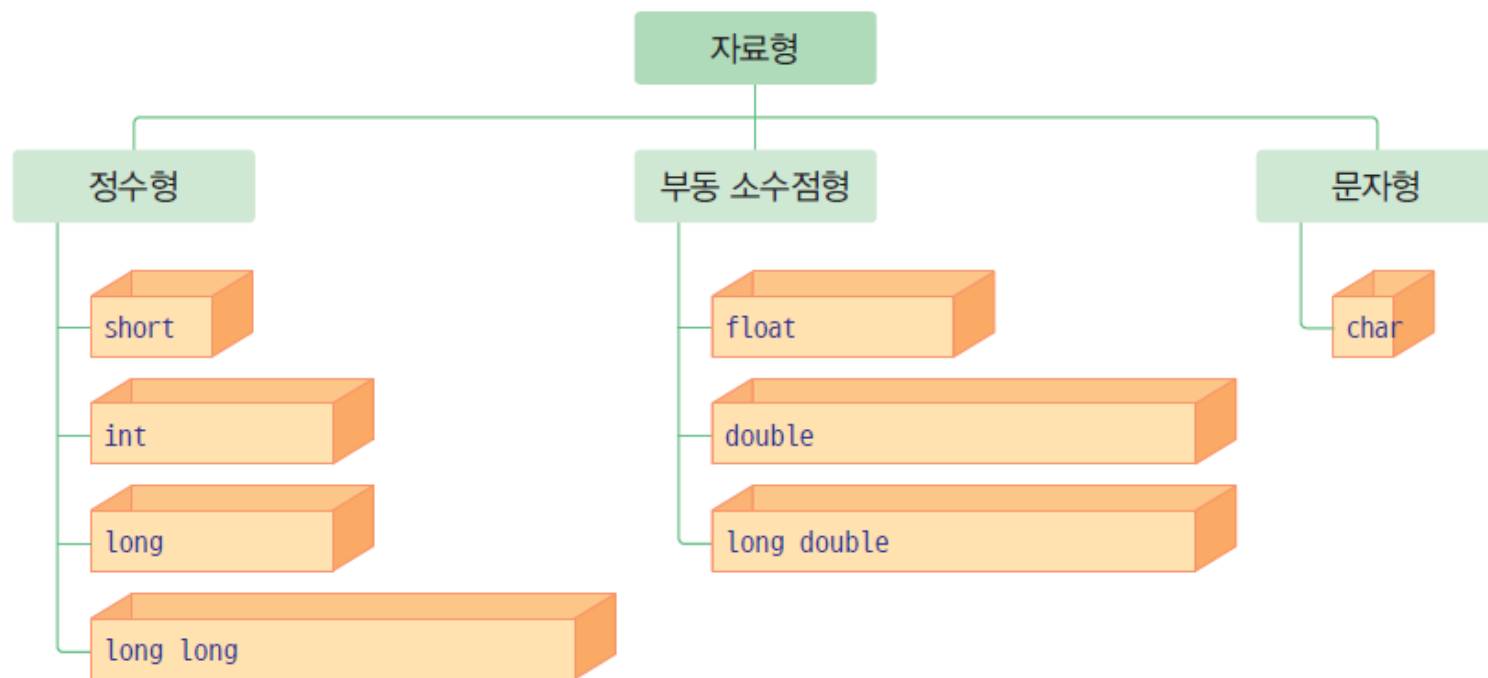
물건이 상자보다 너무 작으면
공간이 낭비될 것이다.





자료형

9





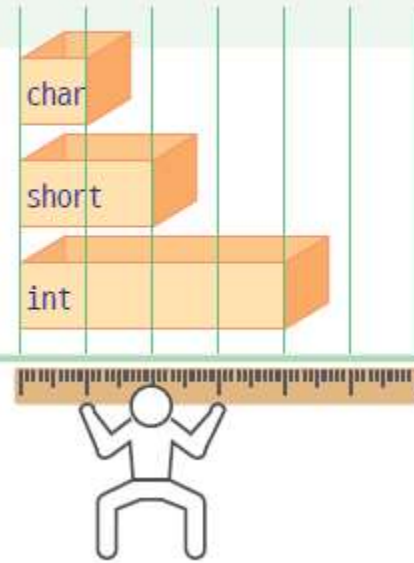
자료형의 크기

10

Syntax: sizeof()

예

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



sizeof 연산자는 변수나
데이터 타입의 크기를
바이트 단위로 반환합니다.





예제: 자료형의 크기

sizeof.c

11

```
#include <stdio.h>

int main(void)
{
    int x;
    printf("변수 x의 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("long long형의 크기: %d\n", sizeof(long long));
    printf("float형의 크기: %d\n", sizeof(float));
    printf("double형의 크기: %d\n", sizeof(double));
    return 0;
}
```



실행 결과

12

실행결과

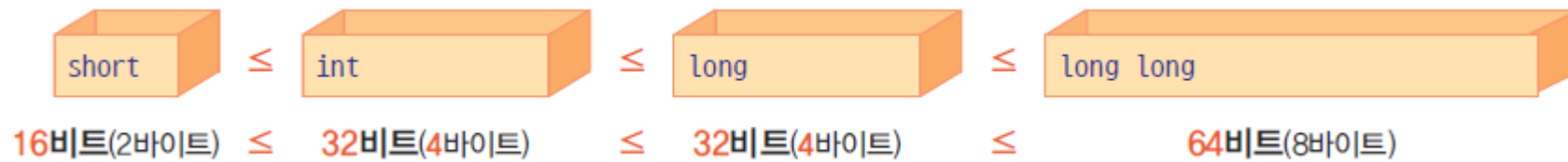
변수 x의	크기: 4
char형의	크기: 1
int형의	크기: 4
short형의	크기: 2
long형의	크기: 4
long long형의	크기: 8
float형의	크기: 4
double형의	크기: 8



정수형

13

- short, int, long, long long





정수형

14

자료형			비트	범위
정수형	short	부호있는 정수	16비트	-32768~32767
	int		32비트	-2147483648~2147483647
	long			-2147483648~2147483647
	long long		64비트	-9,223,372,036,854,775,808 ~9,223,372,036,854,775,807
	unsigned short	부호없는 정수	16비트	0~65535
	unsigned int		32비트	0~4294967295
	unsigned long			0~4294967295
	unsigned long long		64비트	0~18,446,744,073,709,551,615



정수형 선언의 예

15

- `short grade;` `// short형의 변수를 생성한다.`
- `int count;` `// int형의 변수를 생성한다.`
- `long distance;` `// distance형의 변수를 생성한다.`



정수형의 범위

16

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

17

integer.c

```
/* 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;

    year = 10;                // 약 3만2천을 넘지 않도록 주의
    sale = 200000000;         // 약 21억을 넘지 않도록 주의
    total_sale = year * sale; // 약 21억을 넘지 않도록 주의

    printf("total_sale = %d \n", total_sale);

    return 0;
}
```

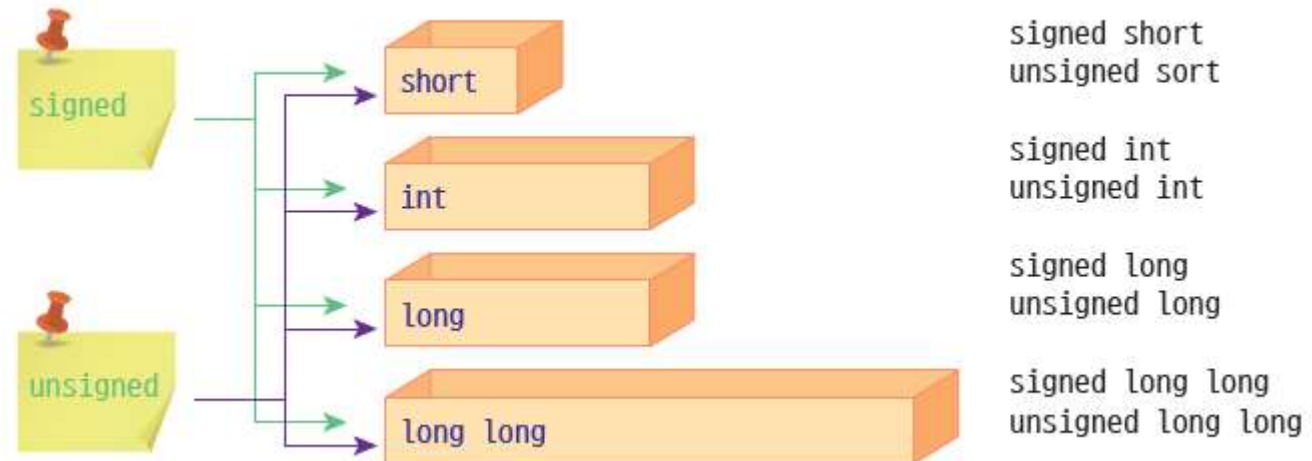
실행결과



signed, unsigned 수식자

18

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

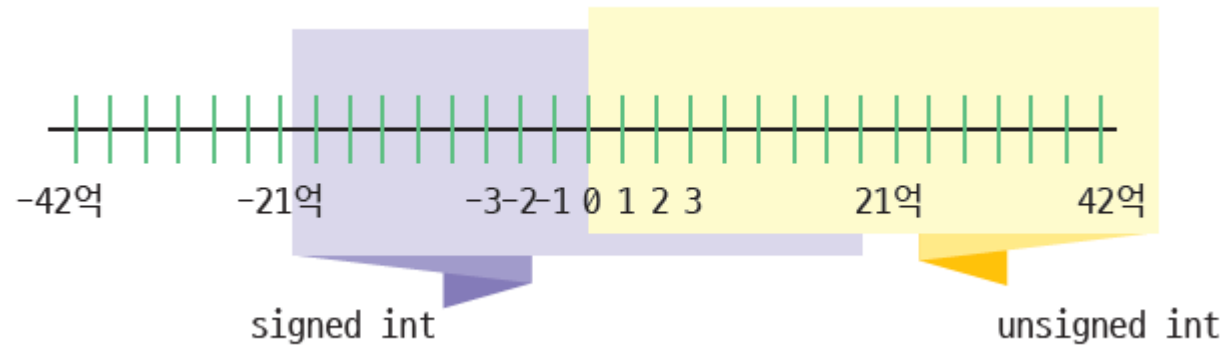




unsigned int

19

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





unsigned 수식자

20

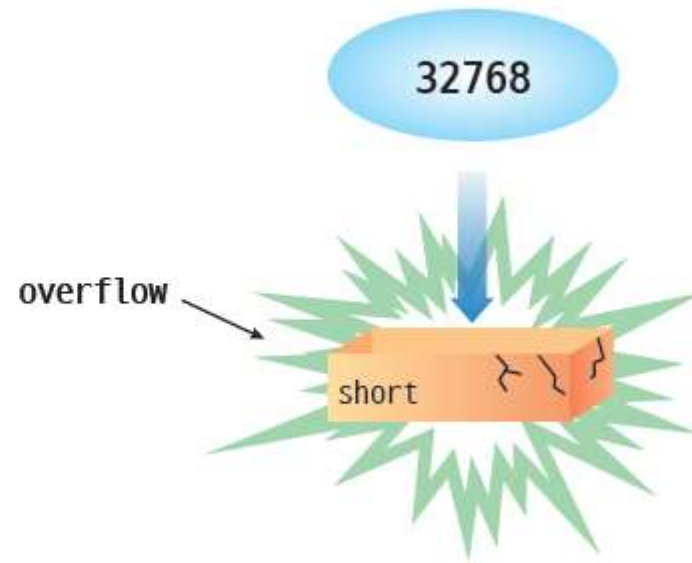
- `unsigned int speed;` // 부호없는 int형
- `unsigned distance;` // unsigned int distance와 같다.
- `unsigned short players;` // 부호없는 short형
- `unsigned long seconds;` // 부호없는 long형



오버플로우

21

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





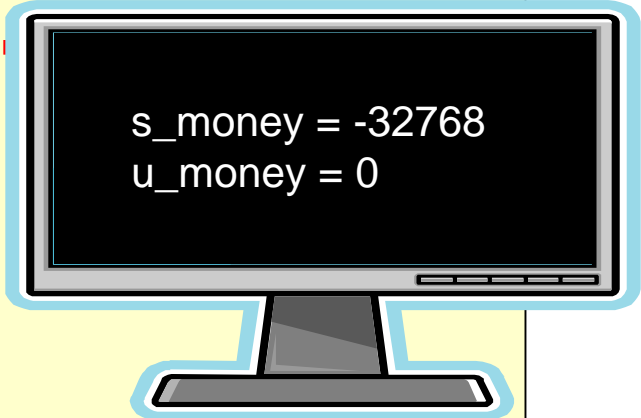
```
#include <stdio.h>
#include <limits.h>

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



s_money = -32768
u_money = 0



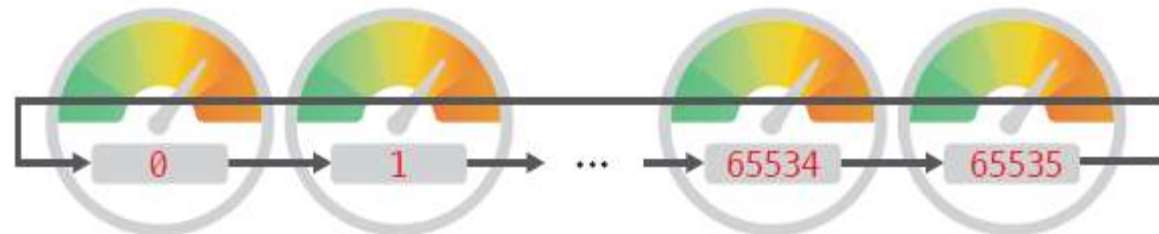
오버플로우

23

- 규칙성이 있다.
 - ▣ 수도 계량기나 자동차의 주행거리계와 비슷하게 동작



short의 경우



unsigned short의 경우



정수 상수

24

- 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

25

□ 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

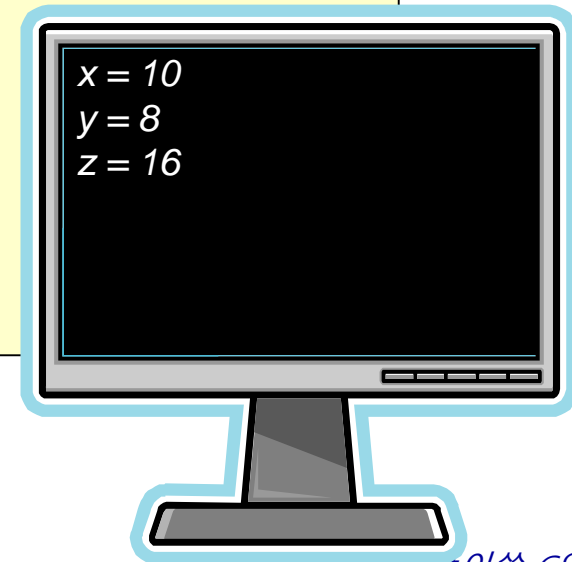


예제

int_const.c

26

```
/* 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\n", x);  
    printf("y = %d\n", y);  
    printf("z = %d\n", z);  
  
    return 0;  
}
```





기호 상수

27

- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것
- (예)
 - ▣ `area = 3.141592 * radius * radius;`
 - ▣ `area = PI * radius * radius;`

 - ▣ `income = salary - 0.15 * salary;`
 - ▣ `income = salary - TAX_RATE * salary;`
- 기호 상수의 장점
 - ▣ 가독성이 높아진다.
 - ▣ 값을 쉽게 변경할 수 있다.



기호 상수의 장점

28

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

```
int main(void)
```

```
{
```

```
...
```

```
won1 = 1120 * dollar1;
```

```
won2 = 1120 * dollar2;
```

```
...
```

```
}
```

1050

1050

리터럴 상수를 사용하는 경우:
등장하는 모든 곳을 수정하여야 한다.

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

```
#define EXCHANGE_RATE 1120
```

```
int main(void)
```

```
{
```

```
...
```

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

```
...
```

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

```
...
```

```
}
```

1050

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

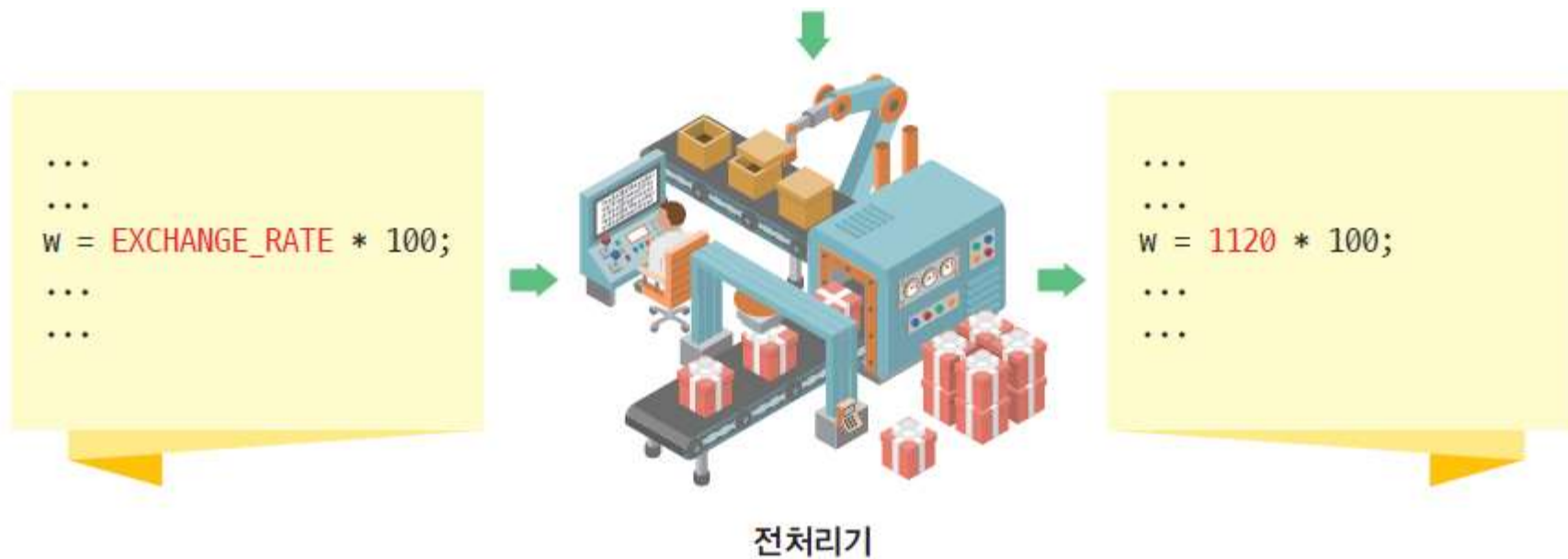


기호 상수를 만드는 방법 #1

29

EXCHANGE_RATE이라는 기호를 1120으로 정의

#define EXCHANGE_RATE 1120



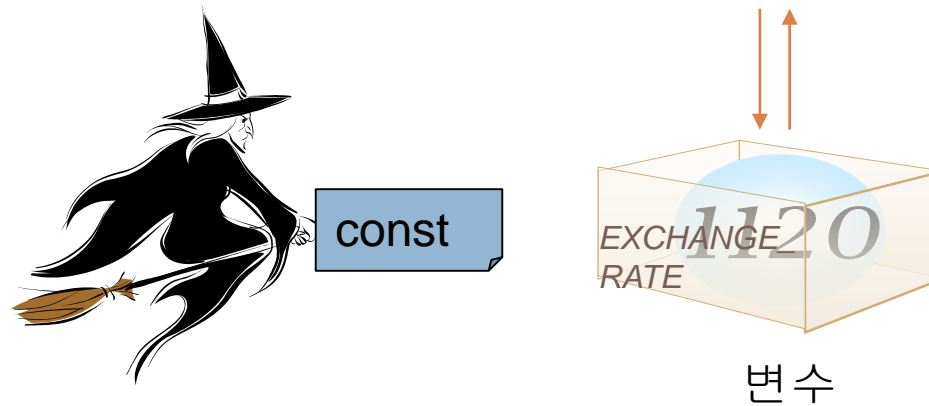


기호 상수를 만드는 방법 #2

30

변수가 값을 변경할 수 없게 한다.

const int EXCHANGE_RATE = 1120;





예제: 기호 상수

sym_const.c

31


```
#include <stdio.h>
#define TAX_RATE 0.2

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;
}
```

기호상수



월급을 입력하시요: 200
연봉은 2400입니다.
세금은 480.000000입니다.

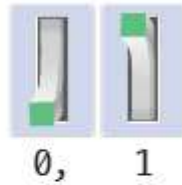


정수 표현 방법

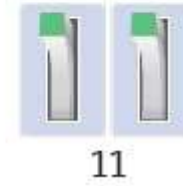
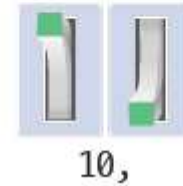
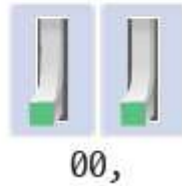
32

- 컴퓨터에서 정수는 이진수 형태로 표현되고 이진수는 전자 스위치로 표현된다.

스위치가 **하나** 있으면



스위치가 **둘** 있으면





정수 표현 방법

33

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

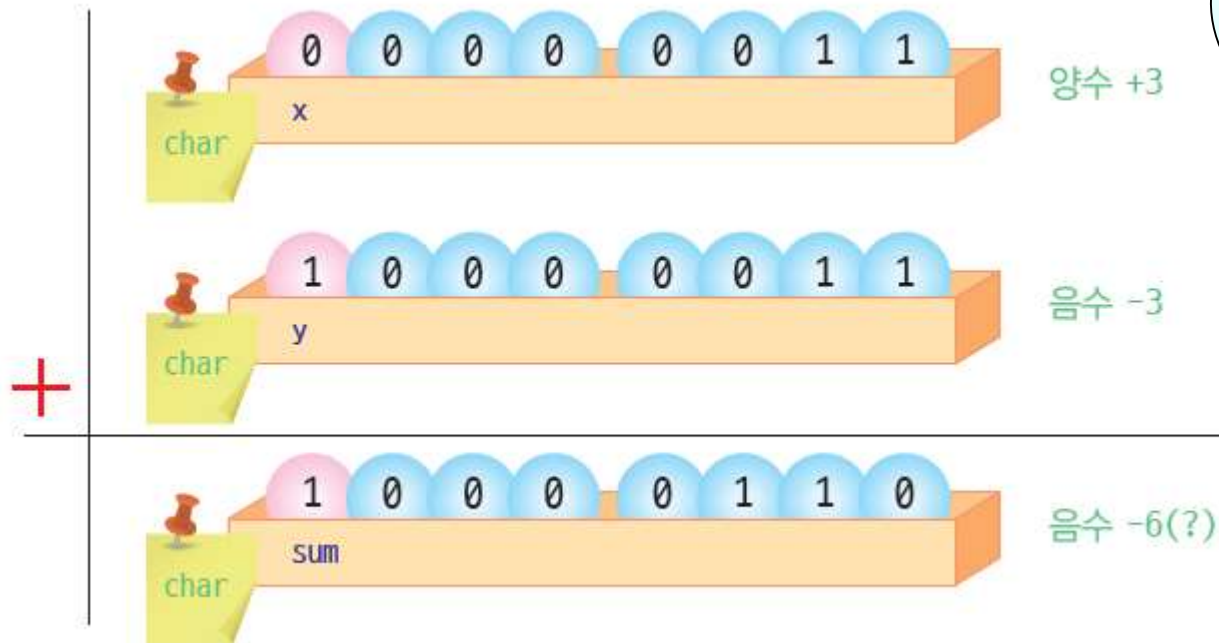




음수를 표현하는 첫번째 방법

34

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



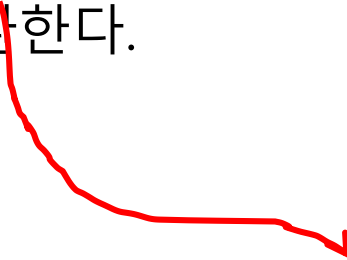
이 방법으로 표현된 이진수를 평범하게 더하면 결과가 부정확합니다.



컴퓨터는 덧셈만 할 수 있다

35

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

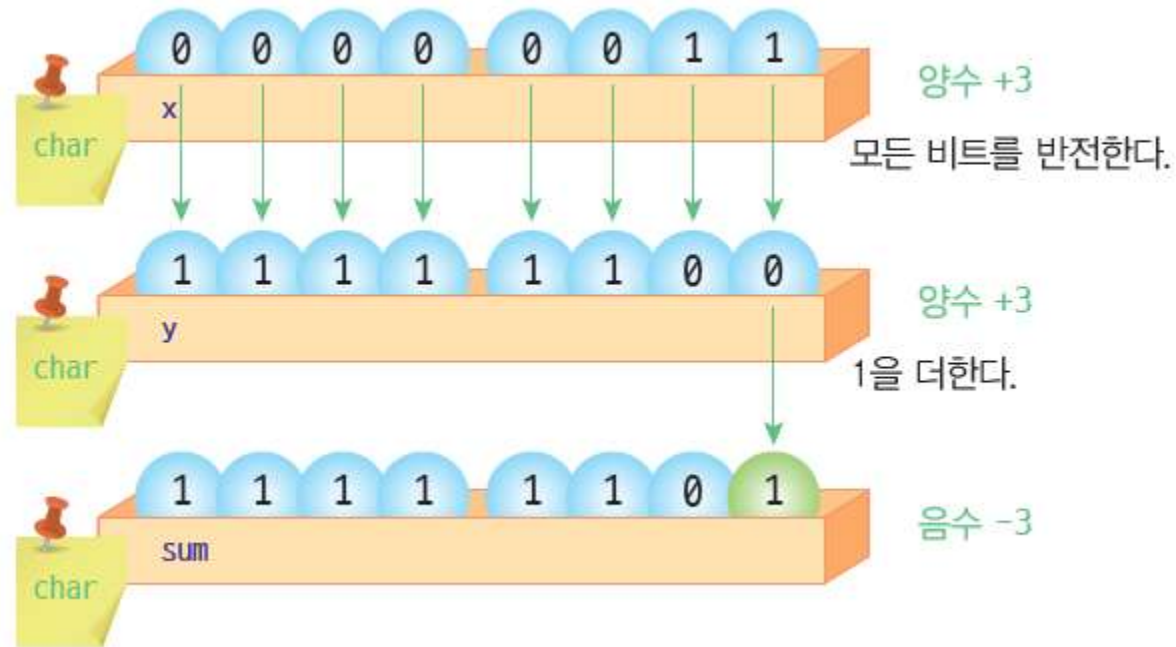

$$3-3 = 3+(-3)$$



음수를 표현하는 두번째 방법

36

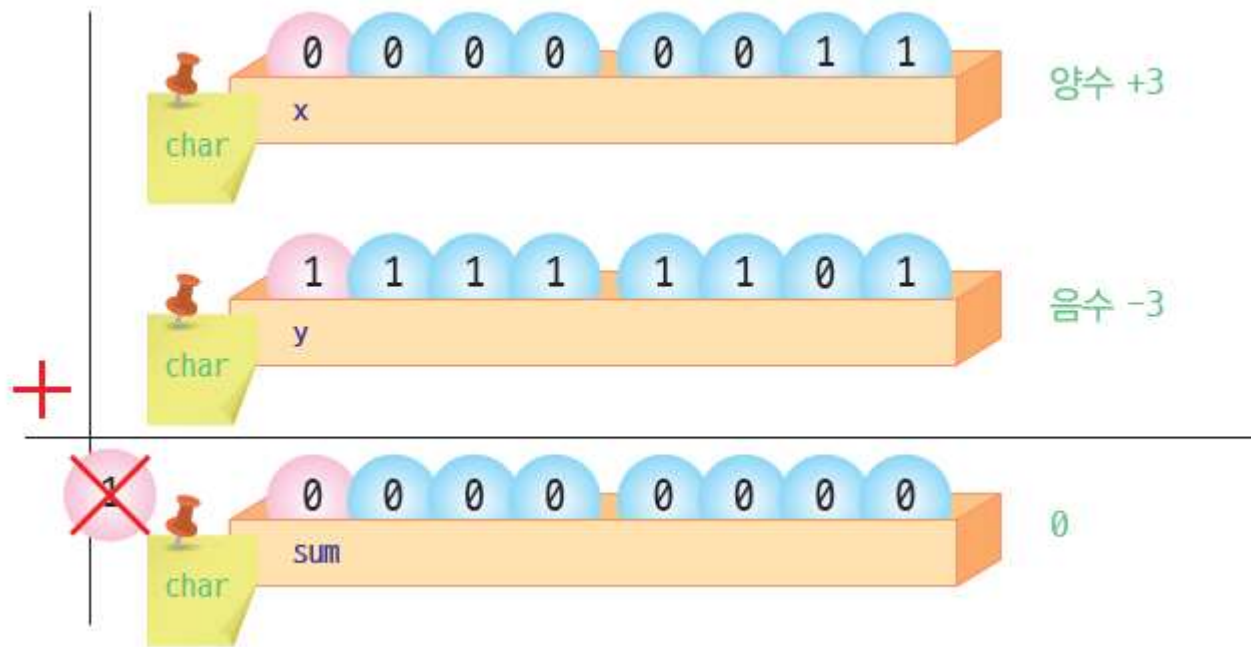
- 2의 보수로 음수를 표현한다. -> 표준적인 음수 표현 방법
- 2의 보수를 만드는 방법





2의 보수로 양수와 음수를 더하면

37



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





예제

38

two_com1.c

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

```
// 8자리의 16진수로 출력한다.  
// 8자리의 16진수로 출력한다.  
// 8자리의 16진수로 출력한다.
```

```
    return 0;
```

```
}
```

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



부동소수점형

39

- 컴퓨터에서 실수는 부동소수점형으로 표현
 - ▣ 소수점이 떠서 움직인다는 의미
 - ▣ 과학자들이 많이 사용하는 과학적 표기법과 유사

실수의 정밀도를 나타낸다.

실수의 표현 범위를 나타낸다.

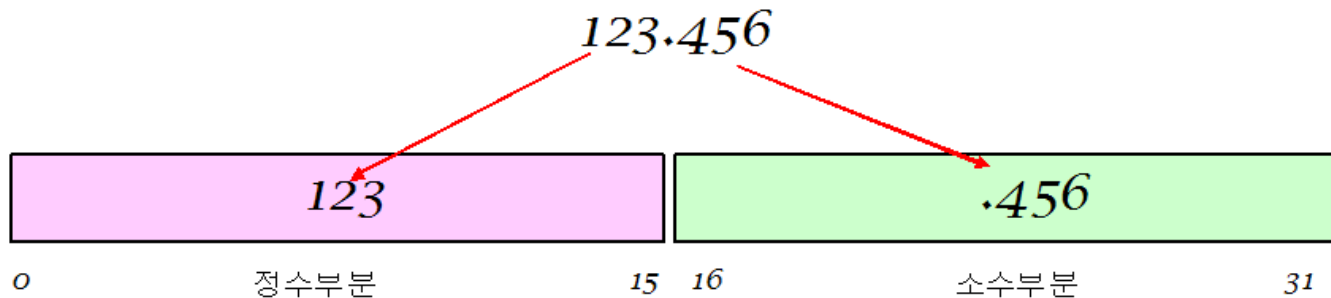
$$\underbrace{1.49598}_{\text{가수 부분}} \times \underbrace{10^8}_{\text{지수 부분}}$$



실수를 표현하는 방법

40

- #1 고정 소수점 방식
 - ▣ 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

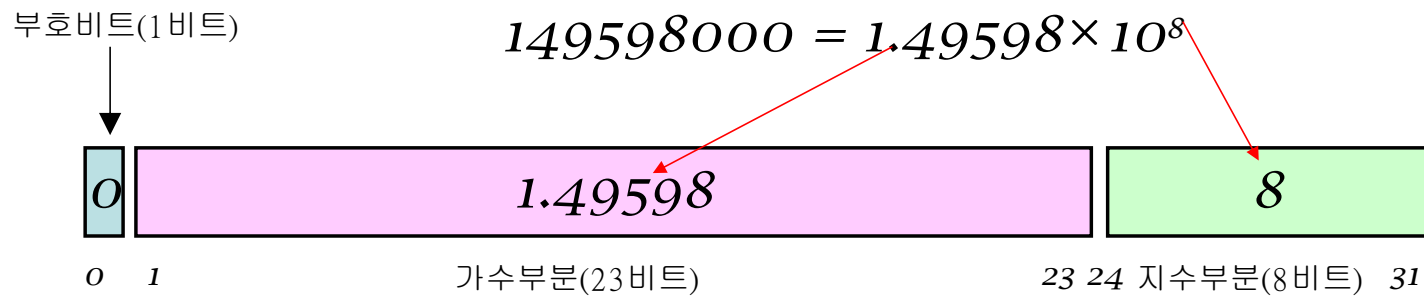




실수를 표현하는 방법

41

□ #2 부동 소수점 방식



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



부동 소수점 형

42



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	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$



예제

43

floating.c

```
/* 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

44

실수	지수 표기법	의미
123.45	1.2345e2	1.2345×10^2
12345.0	1.2345e5	1.2345×10^5
0.000023	2.3e-5	2.3×10^{-5}
2,000,000,000	2.0e9	2.0×10^9

1.23456

2. // 소수점만 붙여도 된다.

.28 // 정수부가 없어도 된다.

2e+10 // +나 -기호를 지수부에 붙일 수 있다.

9.26E3 // 9.26×10^3

0.67e-7 // 0.67×10^{-7}



부동 소수점 오버플로우

45

```
#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

underflow.c

46

```
#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

47

- 오차가 있을 수 있다!

floating_error.c

```
#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 계산된다.

0.000000



문자형

48

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현





문자형

49

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



아스키 코드표 (일부)

50

Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자
0	0	NULL	20	14	DC4	40	28	(60	3C	<
1	1	SOH	21	15	NAK	41	29)	61	3D	=
2	2	STX	22	16	SYN	42	2A	*	62	3E	>
3	3	ETX	23	17	ETB	43	2B	+	63	3F	?
4	4	EOL	24	18	CAN	44	2C	,	64	40	@
5	5	ENQ	25	19	EM	45	2D	-	65	41	A
6	6	ACK	26	1A	SUB	46	2E	.	66	42	B
7	7	BEL	27	1B	ESC	47	2F	/	67	43	C
8	8	BS	28	1C	FS	48	30	0	68	44	D
9	9	HT	29	1D	GS	49	31	1	69	45	E
10	A	LF	30	1E	RS	50	32	2	70	46	F
11	B	VT	31	1F	US	51	33	3	71	47	G
12	C	FF	32	20	space	52	34	4	72	48	H
13	D	CR	33	21	!	53	35	5	73	49	I
14	E	SO	34	22	"	54	36	6	74	4A	J
15	F	SI	35	23	#	55	37	7	75	4B	K
16	10	DLE	36	24	\$	56	38	8	76	4C	L
17	11	DC1	37	25	%	57	39	9	77	4D	M
18	12	DC2	38	26	&	58	3A	:	78	4E	N
19	13	DC3	39	27	'	59	3B	;	79	4F	O

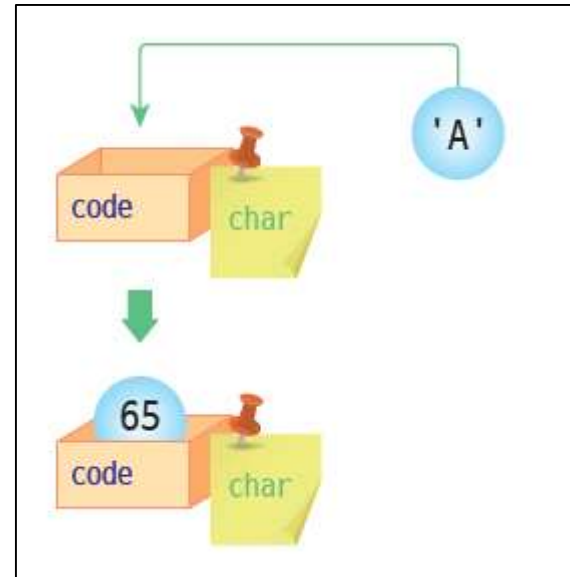


문자 변수

51

- char형을 사용하여 문자를 저장한다.

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





예제

52

char_var.c

```
/* 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
```

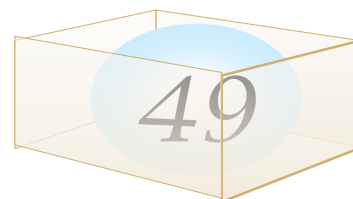
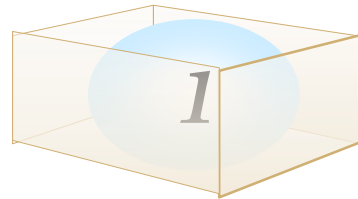


Quiz

53

(Q) 1과 '1'의 차이점은?

(A) 1은 정수이고 '1'은 문자 '1'을 나타내는
아스키코드이다.

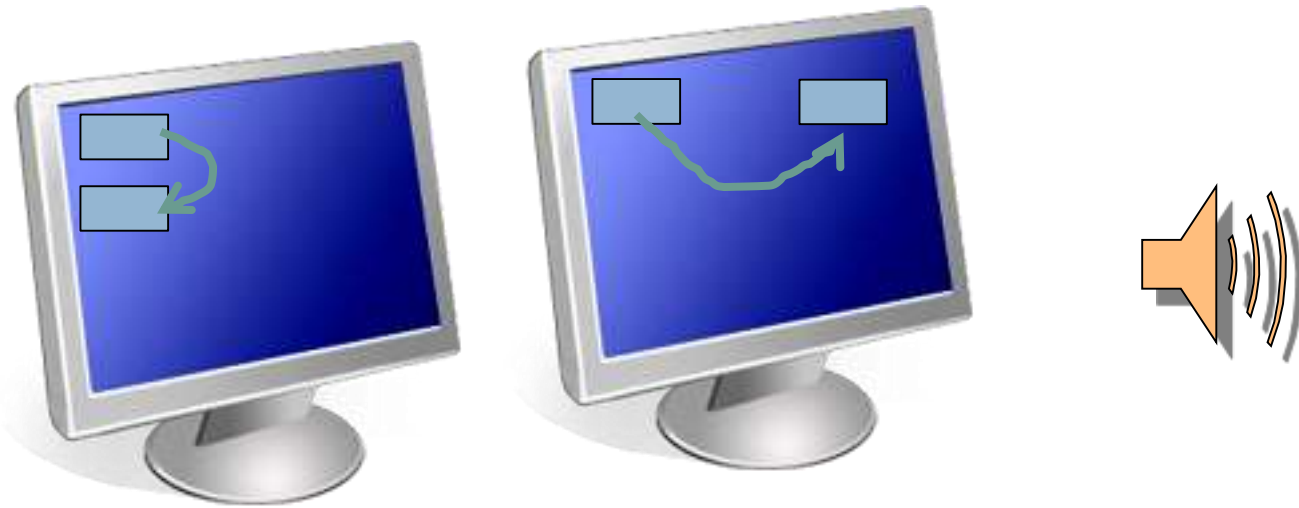




제어 문자

54

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



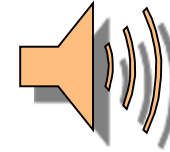


제어 문자를 나타내는 방법

55

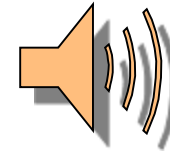
- 아스키 코드를 직접 사용

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



- 이스케이프 시퀀스 사용

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





이스케이프 시퀀스

56

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



예제

57

escape.c

```
#include <stdio.h>
int main()
{
    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

58

- 8비트의 정수를 저장하는데 char 형을 사용할 수 있다..

char.c

```
#include <stdio.h>

int main()
{
    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
```



중간 점검

59

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





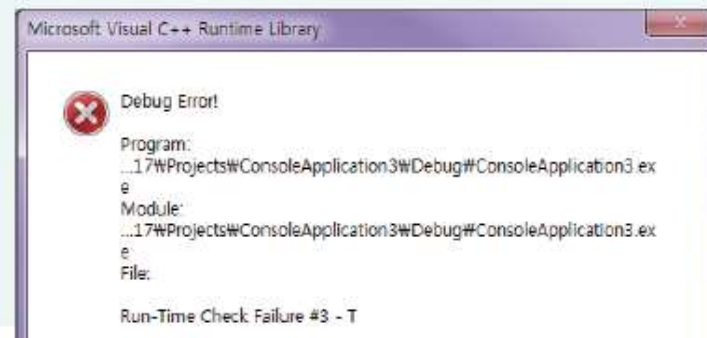
무엇이 문제일까?

60

sum_error.c

sum_error.c

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x, y, z, sum;
5      printf("3개의 정수를 입력하세요 (x, y, z): ");
6      scanf("%d %d %d", &x, &y, &z);
7      sum += x;
8      sum += y;
9      sum += z;
10     printf("3개 정수의 합은 %d\n", sum);
11     return 0;
12 }
```





무엇이 문제일까?

61

sum_error.c

sum_error.c

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x, y, z, sum;
5
6      sum = 0;
7      printf("3개의 정수를 입력하세요 (x, y, z): ");
8      scanf("%d %d %d", &x, &y, &z);
9      sum += x;
10     sum += y;
11     sum += z;
12     printf("3개 정수의 합은 %d\n", sum);
13     return 0;
14 }
```

▶ 실행결과

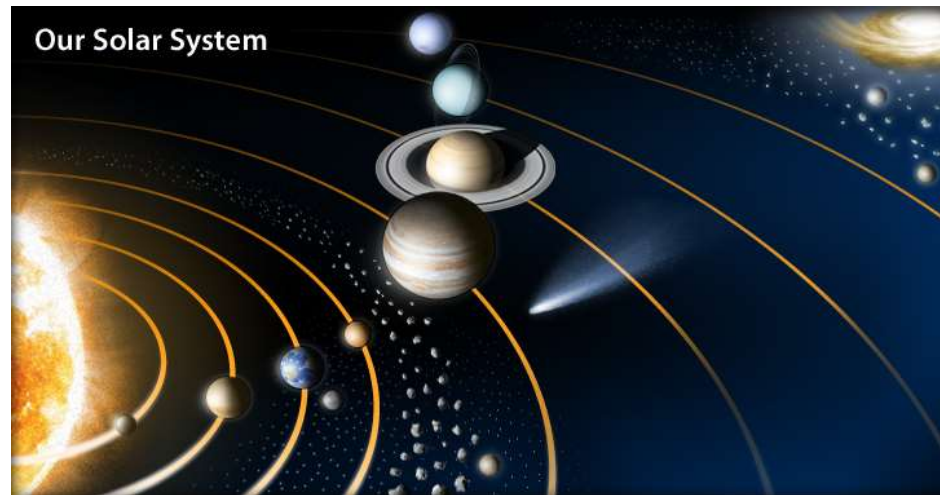
```
3개의 정수를 입력하세요 (x, y, z): 10 20 30
3개의 정수의 합은 60
```



mini project: 태양빛 도달 시간

62

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





실행 결과

63





힌트

64

- 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 %f나 %lf를 사용한다.



sun_light.c

```
#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
```



도전문제

66

- 위의 프로그램의 출력은 498.666667초로 나온다. 이것을 분과 초로 나누어서 8분 20초와 같은 식으로 출력하도록 변경하라. 나머지를 계산하는 연산자는 %이다. 추가적인 정수 변수를 사용하여도 좋다.





Q & A

67

