

## -Part1-

# 제6장 자료형이란 무엇인가

(교재 133페이지 ~ 168페이지)

## 학습목차

### 6. 1 자료형이란

-교재 135페이지 -

### 6. 2 정수형

-교재 139페이지 -

### 6. 3 실수형

-교재 147페이지 -

### 6. 4 문자형

-교재 153페이지 -

### 6. 5 자료형 변환

-교재 157페이지 -

### 6. 6 typedef를 이용한 자료형의 재정의

-교재 163페이지 -

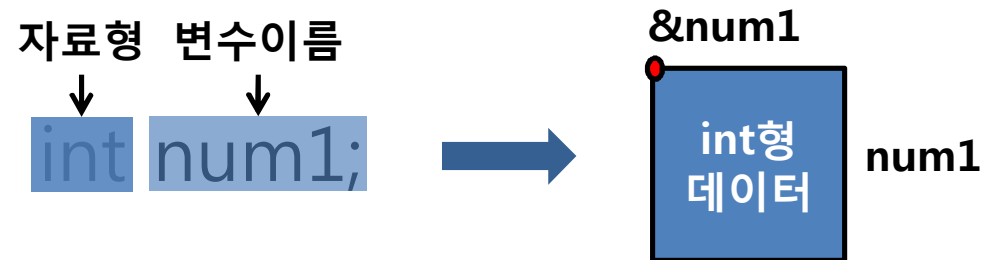
## 6.1 자료형이란

-교재 135페이지 -

## 6.1 자료형이란 (1/4)

### ▶ 자료형

- ✓ 변수가 저장하는 데이터 형식



### ▶ 자료형의 종류

- ✓ 정수형: 정수를 표현하는 데이터 타입
- ✓ 실수형: 소수점이 포함된 값을 표현하는 데이터 타입

정수형				실수형		
char	short	int	long	float	double	long double
문자형						

## 6.1 자료형이란 (2/4)

### ▶ 자료형의 크기

✓ **sizeof 연산자**: 자료형의 크기를 구하는 연산자

사용법	예	설명
sizeof(자료형)	<code>printf("%d", sizeof(int) );</code>	자료형의 메모리 크기를 출력
sizeof(변수)	<code>int num1 = 3; printf("%d", sizeof(num1) );</code>	변수의 메모리 크기를 출력

✓ **sizeof 연산자의 장점**

- '자료형에 할당되는 메모리의 크기를 구할 수 있다.'

## 6.1 자료형이란 (3/4)---[6-1.c 실습]

```
char num1=10;
short num2=20;
int num3=30;
long num4=40;
```

정수형			
char	short	int	long
1바이트	2바이트	4바이트	4바이트

```
printf("Wn-----정수형 자료형과 변수의 메모리 크기-----Wn");
printf(" char형의 크기 %d바이트, %d바이트 Wn", sizeof(char), sizeof(num1) );
printf("short형의 크기 %d바이트, %d바이트 Wn", sizeof(short), sizeof(num2) );
printf(" int형의 크기 %d바이트, %d바이트 Wn", sizeof(int), sizeof(num3) );
printf(" long형의 크기 %d바이트, %d바이트 Wn", sizeof(long), sizeof(num4) );
```

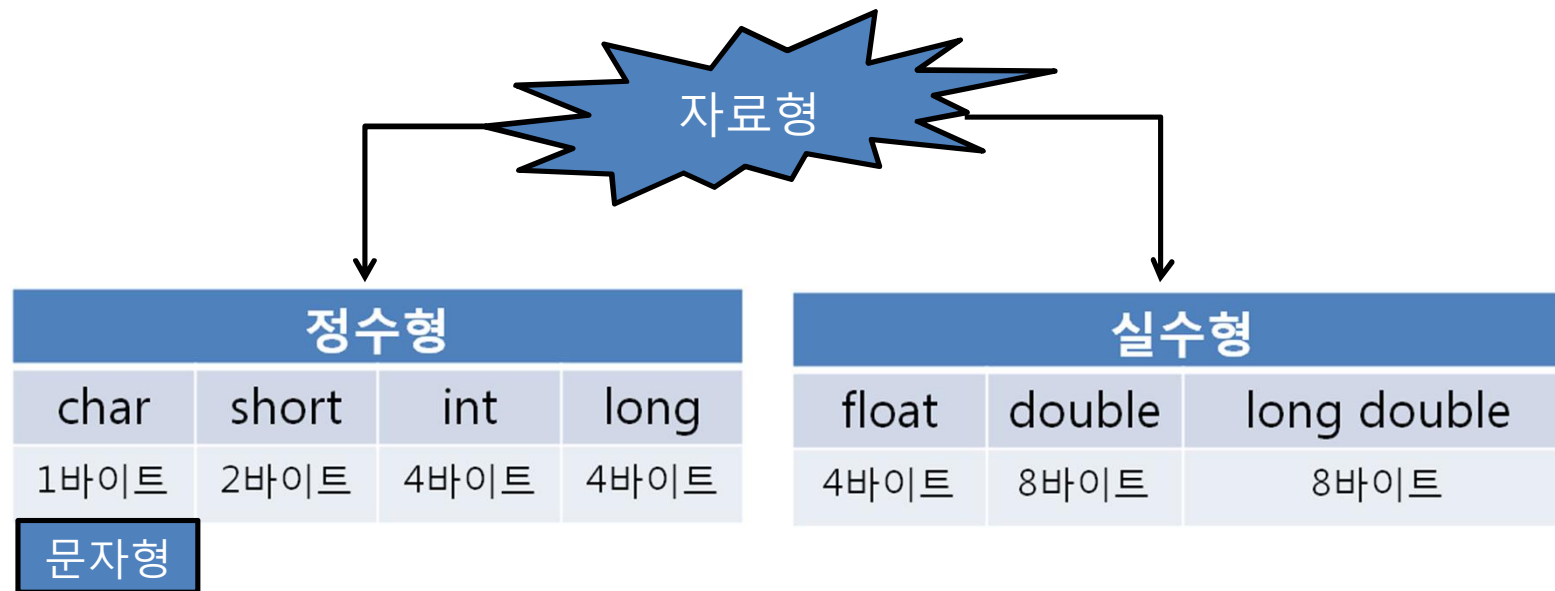
```
float num5=3.14;
double num6=3.15;
long double num7=3.17;
```

실수형		
float	double	long double
4바이트	8바이트	8바이트

```
printf("Wn-----실수형 자료형과 변수의 메모리 크기-----Wn");
printf(" float형의 크기 %d바이트, %d바이트 Wn", sizeof(float), sizeof(num5) );
printf(" double형의 크기 %d바이트, %d바이트 Wn", sizeof(double), sizeof(num6) );
printf("long double형의 크기 %d바이트, %d바이트 Wn", sizeof(long double), sizeof(num7) );
```

## 6.1 자료형이란 (4/4)

### ▶ 기본 자료형의 메모리 크기



## 6.2 정수형

-교재 139페이지 -



## 6.2 정수형 (1/10)

### ▶ 정수형 종류

✓ char(1바이트), short(2바이트), int(4바이트), long(4바이트)

정수형	메모리 크기	데이터 표현 범위
char	1바이트(8비트)	-128 ~ +127
short	2바이트(16비트)	-32768 ~ +32767
int	4바이트(32비트)	-2147483648 ~ +2147483647
long	4바이트(32비트)	-2147483648 ~ +2147483647

✓ 데이터의 표현 범위를 구하는 공식

n은 비트 수(1바이트는 8비트)

$$-2^{n-1}$$

최솟값(MIN)

~

$$+2^{n-1}-1$$

최댓값(MAX)

## 6.2 정수형 (2/10)

### ▶ 정수형 데이터 표현 범위를 자동으로 알려주는 라이브러리

✓ **limits.h** : 정수형 데이터 표현 최솟값(MIN)과 최댓값(MAX) 상수 제공

정수형	메모리 크기	데이터 표현 범위
char	1바이트(8비트)	-128 ~ +127
short	2바이트(16비트)	-32768 ~ +32767
int	4바이트(32비트)	-2147483648 ~ +2147483647
long	4바이트(32비트)	-2147483648 ~ +2147483647

정수형	상수(최솟값)	상수(최댓값)
char	CHAR_MIN	CHAR_MAX
short	SHRT_MIN	SHRT_MAX
int	INT_MIN	INT_MAX
long	LONG_MIN	LONG_MAX

## 6.2 정수형 (3/10)---[6-2.c 실습]

```
#include <stdio.h>
#include <limits.h> // 정수형의 최솟값(MIN), 최댓값(MAX) 상수 정의
int main(void)
{
    printf(" char의 최솟값 %d, 최댓값 %d \n", CHAR_MIN, CHAR_MAX);
    printf("short의 최솟값 %d, 최댓값 %d \n", SHRT_MIN, SHRT_MAX);
    printf(" int의 최솟값 %d, 최댓값 %d \n", INT_MIN, INT_MAX);
    printf(" long의 최솟값 %d, 최댓값 %d \n", LONG_MIN, LONG_MAX);

    return 0;
}
```

정수형	메모리 크기	데이터 표현 범위
char	1바이트(8비트)	-128 ~ +127
short	2바이트(16비트)	-32768 ~ +32767
int	4바이트(32비트)	-2147483648 ~ +2147483647
long	4바이트(32비트)	-2147483648 ~ +2147483647

## 6.2 정수형 (4/10)

▶ '정수형의 양수 범위를 두 배로 늘리는 unsigned 자료형이 있다.'

✓ unsigned: 0과 양수만을 표현

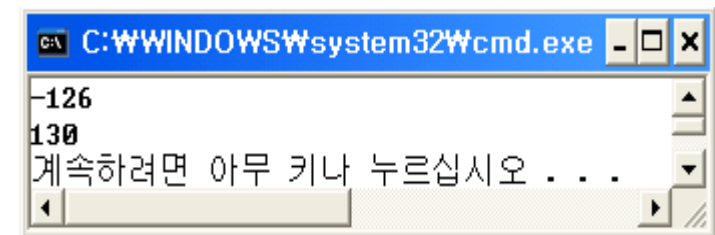
정수형	메모리 크기	데이터 표현 범위
char (signed char) <b>unsigned char</b>	1바이트(8비트) 1바이트(8비트)	-128 ~ +127 0 ~ (127 + 128)
short (signed short) <b>unsigned short</b>	2바이트(16비트) 2바이트(16비트)	-32768 ~ +32767 0 ~ (32767 + 32768)
int (signed int) <b>unsigned int</b>	4바이트(32비트) 4바이트(32비트)	-2147483648 ~ + 2147483647 0 ~ (2147483647 + 2147483648)
long (signed long) <b>unsigned long</b>	4바이트(32비트) 4바이트(32비트)	-2147483648 ~ + 2147483647 0 ~ (2147483647 + 2147483648)

## 6.2 정수형 (5/10)---[6-3.c 실습]

```
#include <stdio.h>
int main(void)
{
    signed char num1=130;           // -128 ~ 127의 데이터 표현 범위
    unsigned char num2=130;        // 0 ~ 256의 데이터 표현 범위

    printf("%d \n", num1);          // -126 출력
    printf("%u \n", num2);          // 130 출력

    return 0;
}
```



## 6.2 정수형 (6/10)

▶ **limits.h** 에서 제공하는 unsigned형 상수의 **최댓값(MAX)**

unsigned 정수형	상수(최댓값)
unsigned char	UCHAR_MAX
unsigned short	USHRT_MAX
unsigned int	UINT_MAX
unsigned long	ULONG_MAX

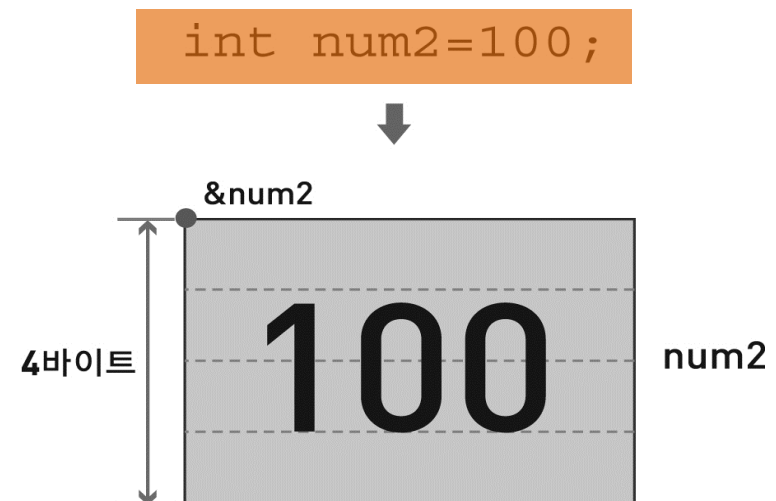
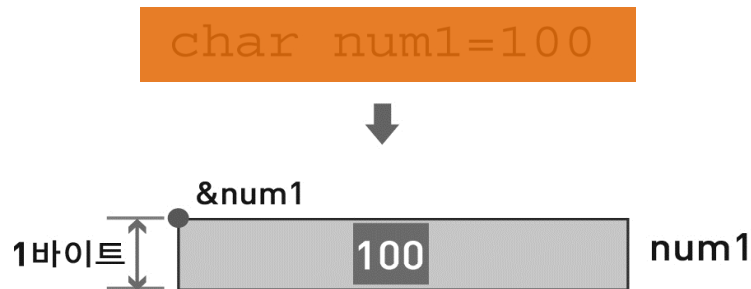


unsigned 정수형	데이터 표현 범위
unsigned char	0 ~ (127 + <b>128</b> )
unsigned short	0 ~ (32767 + <b>32768</b> )
unsigned int	0 ~ (2147483647 + <b>2147483648</b> )
unsigned long	0 ~ (2147483647 + <b>2147483648</b> )

## 6.2 정수형 (7/10)

### ▶ '정수형은 int형을 선호한다.'

✓ char형 변수와 int형 변수의 차이



✓ CPU가 int형을 가장 빠르게 처리하는 이유

- 개발된 대부분의 컴퓨터들은 32비트 이상의 시스템
- CPU가 연산하는 기본 단위가 최소 32비트

## 6.2 정수형 (8/10)

### ▶ '정수형의 오버플로우와 언더플로우는 순환된 값을 출력한다.'

✓ '오버플로우' : 자료형에 저장할 수 있는 최대 범위보다 큰 수 저장



✓ '언더플로우' : 자료형에 저장할 수 있는 최소 범위보다 작은 수 저장





## 6.2 정수형 (9/10)---[6-4.c 실습]

```
#include <stdio.h>
int main(void)
{
    char num1=-129;        // 최솟값(-128)보다 -1만큼 작은 값 저장(언더플로우)
    char num2=128;         // 최댓값(127)보다 +1만큼 큰 값 저장(오버플로우)

    printf("%d \n", num1); // 127 출력
    printf("%d \n", num2); // -128출력

    num1=-130;             // 최솟값(-128)보다 -2만큼 작은 값 저장(언더플로우)
    num2=129;              // 최댓값(127)보다 +2만큼 큰 값 저장(오버플로우)

    printf("%d \n", num1); // 126 출력
    printf("%d \n", num2); // -127출력

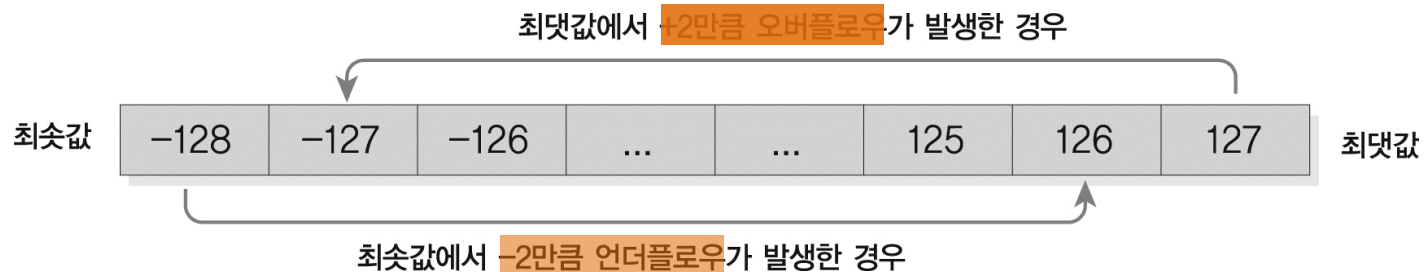
    return 0;
}
```

## 6.2 정수형 (10/10) ---[6-4.c 분석]

```
char num1=-129;           // 최솟값보다 -1만큼 작은 값 저장(언더플로우)  
char num2=128;           // 최댓값보다 +1만큼 큰 값 저장(오버플로우)
```



```
num1=-130;               // 최솟값(-128)보다 -2만큼 작은 값 저장(언더플로우)  
num2=129;                // 최댓값(127)보다 +2만큼 큰 값 저장(오버플로우)
```



## 6.3 실수형

-교재 147페이지 -

## 6.3 실수형 (1/6)

### ▶ 실수형이란

- ✓ 실수형 데이터를 저장하는 변수의 자료형
- ✓ 소수점을 가진 실수의 값을 표현할 수 있는 자료형

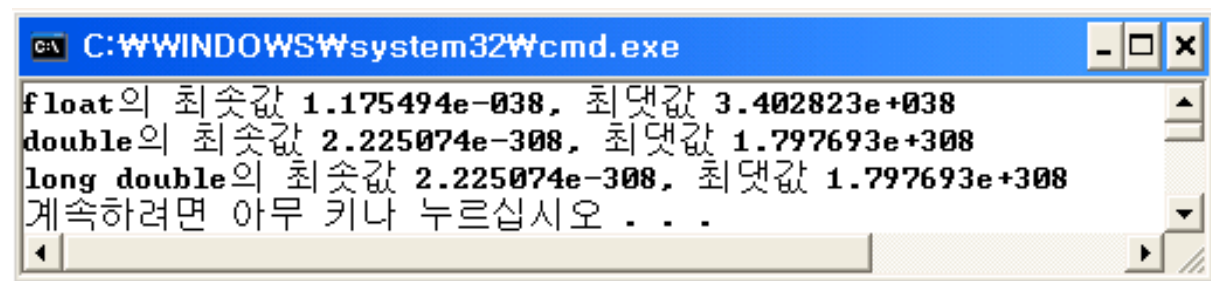
실수형	메모리 크기	데이터 표현 범위
float	4바이트(32비트)	$1.17 \times 10^{-38} \sim 3.40 \times 10^{38}$
double	8바이트(64비트)	$2.22 \times 10^{-308} \sim 1.79 \times 10^{308}$
long double	8바이트(64비트)	$2.22 \times 10^{-308} \sim 1.79 \times 10^{308}$

## 6.3 실수형 (2/6)---[6-5.c 실습]

### ▶ 실수형 데이터 표현 범위를 자동으로 알려주는 라이브러리

✓ **float.h** : 실수형 데이터 표현 최솟값(MIN)과 최댓값(MAX) 상수 제공

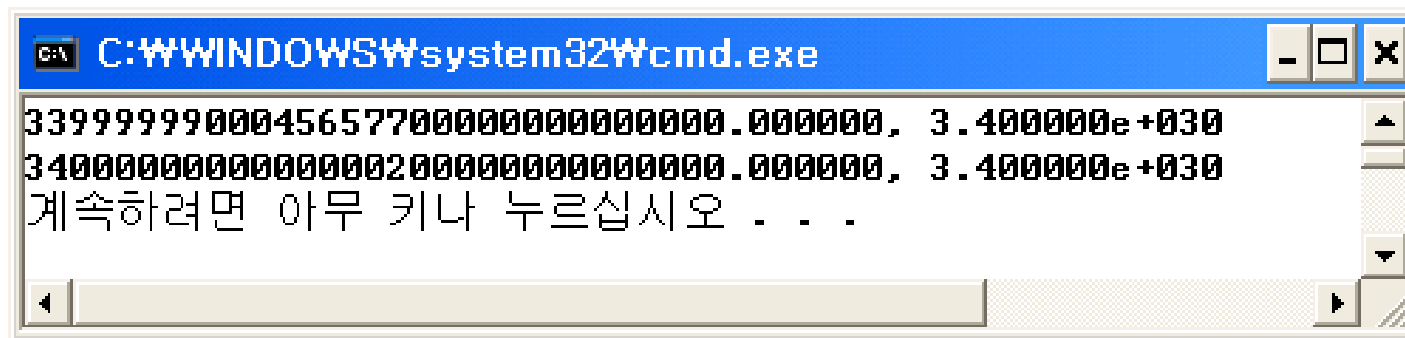
```
#include <stdio.h>
#include <float.h>           // 실수형의 데이터 표현 범위 상수 정의
int main(void)
{
    printf(" float의 최솟값 %e, 최댓값 %e \n", FLT_MIN, FLT_MAX);
    printf("double의 최솟값 %e, 최댓값 %e \n", DBL_MIN, DBL_MAX);
    printf("long double의 최솟값 %e, 최댓값 %e \n", LDBL_MIN, LDBL_MAX);
    return 0;
}
```



## 6.3 실수형 (3/6)---[6-6.c]

```
#include <stdio.h>
int main(void)
{
    float num1=3.4e+30;
    double num2=3.4e+30;

    printf("%f, %e \n", num1, num1);    // float형 오차발생
    printf("%lf, %le \n", num2, num2);  // double형은 정상
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
33999999000456577000000000000000.000000, 3.400000e+030
34000000000000000200000000000000.000000, 3.400000e+030
계속하려면 아무 키나 누르십시오 . . .
```

## 6.3 실수형 (4/6)

### ▶ 실수형은 데이터의 정밀도를 높이기 위해 사용

- ✓ 99.9
- ✓ 99.99
- ✓ 99.999
- ✓ 99.9999

실수형	표현 가능한 소수점 이하 자리 수
float	소수점 이하 6자리
double	소수점 이하 15자리
long double	소수점 이하 15자리 또는 그 이상

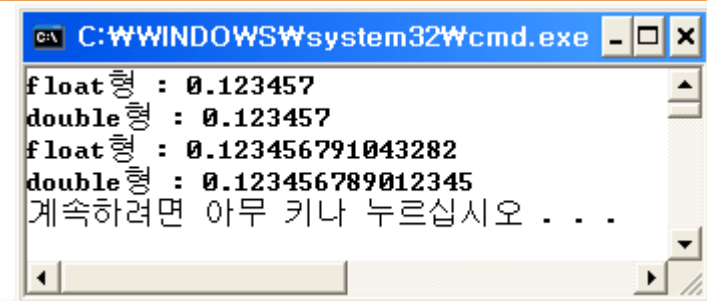
## 6.3 실수형 (5/6)---[6-7.c]

```
#include <stdio.h>
int main(void)
{
    float num1=0.123456789012345;
    double num2=0.123456789012345;

    printf("float형 : %f \\\n", num1);    // 0.123457 출력
    printf("double형 : %lf \\\n", num2);    // 0.123457 출력

    printf("float형 : %.15f \\\n", num1);    // 0.123456791043282 출력
    printf("double형 : %.15lf \\\n", num2);    // 0.123456789012345 출력

    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
float형 : 0.123457
double형 : 0.123457
float형 : 0.123456791043282
double형 : 0.123456789012345
계속하려면 아무 키나 누르십시오 . . .
```



## 6.3 실수형 (6/6)---[6-8.c 실습]

### ▶ '실수형은 double형을 선호한다.'

- ✓ 오차를 줄이기 위해서 .....[6-6.c 참조]
- ✓ 정밀도를 높이기 위해서 .....[6-7.c참조]
- ✓ 컴파일러는 기본적으로 실수형을 double로 인식 ....[6-8.c 참조]

```
#include <stdio.h>
int main(void)
{
    float num1=0.123456;           // float num1=0.123456F;
    printf("float형 : %f \n", num1); // 0.123456(소수점 6자리까지 출력)
    printf("float형 : %.2f \n", num1); // 0.12(소수점 2자리까지 출력)

    return 0;
}
```

컴파일하고있습니다...

C:\W6-8.c(5) : **warning** C4305: '초기화중' : 'double'에서'float'(으)로 잘립니다.

## 6.4 문자형

-교재 153페이지 -

## 6.4 문자형 (1/3)

▶ '컴퓨터(CPU)는 문자를 인식하지 못한다.'

▶ '컴퓨터는 **ASCII** 코드를 참조해서 문자를 인식한다.'

✓ **A**merican **S**tandards **C**ommittee for **I**nformation **I**nterchange

✓ ASCII 표는 교재 703페이지~707페이지 참조

▶ '문자형은 **char**형을 선호한다.'

✓ 작은따옴표 안에 문자 하나를 입력 (ASCII에 지정된 숫자, 문자만 저장)

• 사용 예) `char c = 'a';`

✓ 잘못 사용한 문자형의 사례

• 사용 예 1) `char c = '가';` // 한글은 2바이트

• 사용 예 2) `char c = a;` // 작은 따옴표가 없음

• 사용 예 3) `char c = "o";` // 큰 따옴표의 사용

## 6.4 문자형 (2/3)---[6-9.c 실습]

```
#include <stdio.h>
int main(void)
{
    char val1;

    val1='A';
    printf("%d %c \n", val1, val1); // 65  A 출력

    val1='B';
    printf("%d %c \n", val1, val1); // 66  B 출력

    val1='C';
    printf("%d %c \n", val1, val1); // 67  C 출력

    return 0;
}
```

## 6.4 문자형 (3/3)---[6-10.c]

```
#include <stdio.h>
int main(void)
{
    char val1;
    int val2;

    printf("문자 입력 : ");
    scanf("%c", &val1);
    printf("ASCII 코드 값 %d입니다. \n", val1);

    printf("ASCII 코드 값 입력 : ");
    scanf("%d", &val2);
    printf("문자로 %c입니다.\n", val2);

    return 0;
}
```

## 6.5 자료형 변환

-교재 157페이지 -

## 6.5 자료형 변환 (1/7)

### ▶ 자료형 변환의 종류

#### ✓ 자동 형변환

- '컴파일러가 자동 형변환 시킨다.'

#### ✓ 강제 형변환

- '프로그래머 강제 형변환 시킨다.'

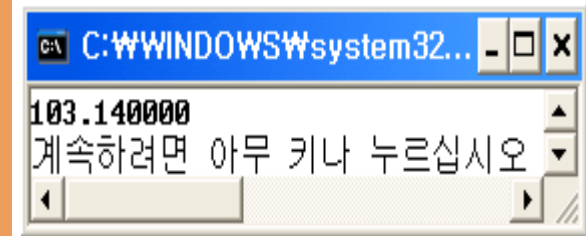
## 6.5 자료형 변환 (2/7)---[6-11.c 실습]

### ▶ '컴파일러가 자동으로 형변환을 해준다.' - 자동 형변환

- ✓ 다른 자료형 간 산술 연산의 경우에 작은형에서 큰형으로 자동 형변환
  - 정수 + 실수 또는 실수 + 정수와 같은 산술 연산을 하는 경우

```
#include <stdio.h>
int main(void)
{
    int    num1=100;           // 정수
    double num2=3.14;          // 실수

    printf("%lf \n", num1+num2); // 정수 + 실수
    return 0;
}
```



- ✓ 자료형 변환 우선순위 (작은형에서 큰형으로...)
  - char < int < long < float < double < long double



## 6.5 자료형 변환 (3/7)---[6-12.c 실습]

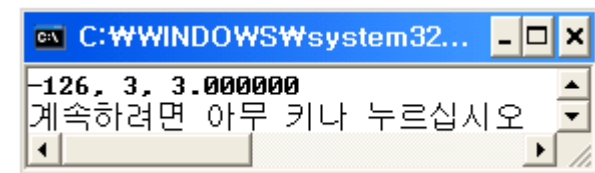
### ▶ '컴파일러가 자동으로 형변환을 해준다.' - 자동 형변환

✓ 대입 연산을 하는 경우

- 대입 연산자를 기준으로 오른쪽에서 왼쪽으로 자동 형변환

```
#include <stdio.h>
int main(void)
{
    char    num1=130;
    int     num2=3.14;
    double  num3=3;

    printf("%d, %d, %lf \n", num1, num2, num3);
    return 0;
}
```



## 6.5 자료형 변환 (4/7)---[6-12.c 분석]

4행 char num1=130;



자동 형변환

char형

int형

num1

=

130

대입 연산자를 기준으로 오른쪽에서 왼쪽으로 자동 형변환



오버플로우 발생

5행 int num2=3.14;



자동 형변환

int형

double형

num2

=

3.14

대입 연산자를 기준으로 오른쪽에서 왼쪽으로 자동 형변환



0.14만큼의 데이터가 손실됨

6행 double num3=3;



자동 형변환

double형

int형

num3

=

3

대입 연산자를 기준으로 오른쪽에서 왼쪽으로 자동 형변환



데이터 손실이 없음

## 6.5 자료형 변환 (5/7)

### ▶ '컴파일러가 자동으로 형변환을 해준다.' - 자동 형변환

- ✓ 데이터 손실이 없는 경우
  - 예) int형 (작은 자료형) 에서 double형(큰 자료형)으로 변환되는 경우
- ✓ 데이터 손실이 있는 경우
  - 예) double형 (큰 자료형) 에서 int형(작은 자료형)으로 변환되는 경우

### ▶ 데이터 손실이 있는 경우, 데이터 손실을 최소화 하는 방법은?

- ✓ '강제 형변환' - 자동 형변환의 문제점 보완

## 6.5 자료형 변환 (6/7)

### ▶ 프로그래머가 강제로 형변환을 해준다 - 강제 형변환

- ✓ 이미 정의된 자료형을 강제로 다른 자료형으로 변환하는 것
- ✓ 괄호 연산자 ( )를 이용

```
int num1=2;  
(double) num1;
```

↑                      ↑  
자료형                  변수

## 6.5 자료형 변환 (7/7)---[6-13.C] <강제 형변환의 필요성>

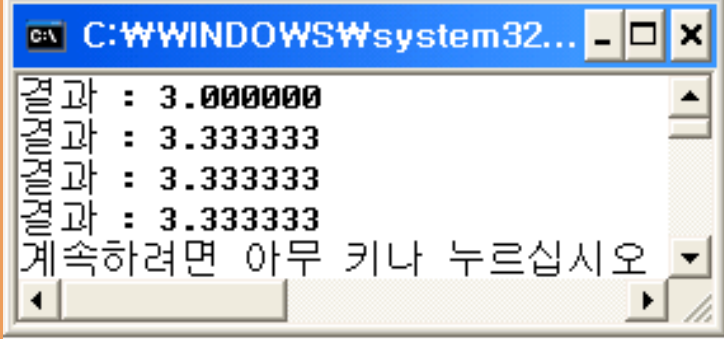
```
#include <stdio.h>
int main(void)
{
    int num1=10, num2=3;
    double result;

    result=num1/num2;
    printf("결과 : %lf \n", result);

    result=(double)num1/num2;
    printf("결과 : %lf \n", result);

    result=num1/(double)num2;
    printf("결과 : %lf \n", result);

    result=(double)num1/(double)num2;
    printf("결과 : %lf \n", result);
    return 0;
}
```



```
C:\WINDOWS\system32...
결과 : 3.000000
결과 : 3.333333
결과 : 3.333333
결과 : 3.333333
계속하려면 아무 키나 누르십시오
```

## 6.6 typedef를 이용한 자료형의 재정의

-교재 163페이지 -

## 6.6 typedef를 이용한 자료형의 재정의 (1/2)

### ▶ 기본 자료형들에 새로운 이름을 붙이는 용도

- ✓ 자료형을 간결하게 표현 가능
- ✓ 프로그램의 가독성을 높일 수 있음
- ✓ 너무 남용하면 자료형 분석 시 혼란 초래

```
typedef int mytype;
```



기본 자료형



사용자 정의 자료형

## 6.6 typedef를 이용한 자료형의 재정의(2/2)---[6-14.c]

```
#include <stdio.h>

typedef int money;

int main(void)
{
    money num1=3000;
    money num2=10000;
    money num3=2000;
    money num4=0;

    num4=num1+num2+num3+num4;
    printf("total money : %d won ₩n", num4);

    return 0;
}
```



## 공부한 내용 떠올리기

- ▶ 자료형이란 무엇인가
- ▶ 정수 자료형
  - ✓ char형, short형, int형, long형
- ▶ 실수 자료형
  - ✓ float형, double형, long double형
- ▶ 자료형 변환
  - ✓ 자동 형변환
  - ✓ 강제 형변환
- ▶ typedef를 이용한 자료형 재정의

## 1m 철학 (출처: 사랑과 지혜의 탈무드)

