정수 타입

▣ 정수 타입

- 변수는 선언될 때의 타입에 따라 저장할 수 있는 값의 종류와 허용 범위가 달라진다.

- 정수형에서 [byte] 와 [short] 는 리터럴이 없기 때문에 [int형의 리터럴]를 사용한다.

- 기본적으로 컴파일러는 정수 리터럴을 int 타입 값으로 인식한다.
  단, 변수의 타입이 int형이 아니라도, 자료형의 범위에 속하는 값이면 저장할수 있다.

- long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙여 표시한다.

타입	메모리 크기		저장되는 값의 허용 범위	
byte	1byte*	8bit	$-2^7 \sim (2^7-1)$	−128 ~ 127
short	2byte	16bit	$-2^{15} \sim (2^{15}-1)$	−32,768 ~ 32,767
char	2byte	16bit	$0 \sim (2^{16}-1)$	0 ~ 65535 (유니코드)
int	4byte	32bit	$-2^{31} \sim (2^{31}-1)$	−2,147,483,648 ~ 2,147,483,647
long	8byte	64bit	$-2^{63} \sim (2^{63}-1)$	−9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

* 1byte = 8bit, bit는 0과 1이 저장되는 단위

- long 타입은 수치가 큰 데이터를 다루는 프로그램에서 사용된다. <예> 은행, 과학 분야

- 기본적으로 컴파일러는 정수 리터럴을 int 타입 값으로 간주하기 때문에,
  int 타입의 허용 범위를 초과하는 리터럴은 뒤에 소문자 “l”이나 대문자“L”을 붙여 long 타입 값임을
  컴파일러에게 알려줘야 한다.

```
long var3 = 1000000000000;   //컴파일러는 int로 간주하기 때문에 에러 발생
```

int	4byte	32bit	$-2^{31} \sim (2^{31}-1)$	-2,147,483,648 ~ 2,147,483,647
long	8byte	64bit	$-2^{63} \sim (2^{63}-1)$	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

- underscore 표기법은 콤마 1000,000,000 로 표현하듯이,
  프로그래밍에선 콤마 대신 밑줄 문자로 표현해도 실제로는 숫자로 읽혀진다.

```
int cost = 1000_000_000;  //1000000000
```

- 변수 와 리터럴의 타입은 반드시 일치해야 한다.

1. 범위가 '변수' > '리터럴' 인 경우, OK

```
int  i = 'A';                  // int > char   <참고> 'A'=65
long  l = 123;                 // long > int
double  d = 3.14f;             // double > float
```

2. 범위가 '변수 < '리터럴' 인 경우, 에러

```
int  i = 300_000_0000;  // int의 범위(-20억~20억) 벗어남
long  l = 3.14f;              // long(8byte) < float(4byte)
float  f = 3.14;              // float < double
```

3. byte, short 변수에 int 리터럴 저장가능 (단, 변수의 타입의 범위 이내어야 함)

```
byte  b = 100;     //OK, byte의 범위(-128~127)에 속함
byte  b = 128;     //에러, byte의 범위를 벗어남
```

- 10진수, 2진수, 8진수, 16진수 변환

1byte = 8bit =11111111	= 2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	= 128	64	32	16	8	4	2	1
	= 255							

<예>
10진수 = 8
2진수 = 00001000 => 0b00001000
8진수 = 00 001 000 => 0010
16진수 = 0000 1000 => 0x08

- 코드에서 프로그래머가 직접 입력한 값을 리터럴(Literal)이라고 부르는데,
  변수에 대입할 정수 리터럴은 진수에 따라 작성하는 방법이 다르다.

• 2진수: 0b 또는 0B로 시작하고 0과 1로 작성

```
int x = 0b1011;     //10진수 값 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 11
int y = 0B10100;    //10진수 값 = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0 = 20
```

• 8진수: 0으로 시작하고 0~7 숫자로 작성

```
int x = 013;        //10진수 값 = 1x8^1 + 3x8^0 = 11
int y = 0206;       //10진수 값 = 2x8^2 + 0x8^1 + 6x8^0 = 134
```

• 10진수: 소수점이 없는 0~9 숫자로 작성

```
int x = 12;
int y = 365;
```

• 16진수: 0x 또는 0X로 시작하고 0~9 숫자나 A, B, C, D, E, F 또는 a, b, c, d, e, f로 작성

```
int x = 0xB3;       //10진수 값 = 11x16^1 + 3x16^0 = 179
int y = 0x2A0F;     //10진수 값 = 2x16^3 + 10x16^2 + 0x16^1 + 15x16^0 = 10767
```