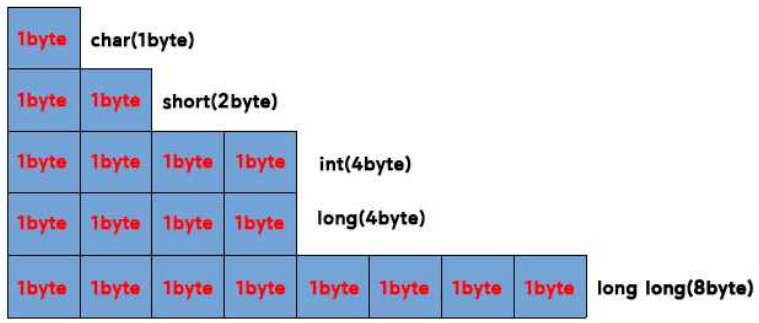
# 3.3 : 기본 자료형(Data Type)

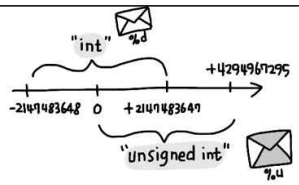
## □ 기본 자료형 - signed 정수 자료형




| 자료형 (Data Type) | 할당되는 메모리 크기 (Byte) | 표현 가능 데이터 범위 | 출력 서식 문자 |
|---|---|---|---|
| char | 1 | -128 ~ 127 | %c (문자로 출력) 또는 %d |
| short | 2 | -32768 ~ 32767 | %d |
| int | 4 | -214748 ~ 2147483647 | %d |
| long | 4 | -214748 ~ 2147483647 | %ld |
| long long | 8 | $-2^{63}$ ~ $2^{63}$ - 1 | %lld |

| ※ 자료형의 int(4byte)와 long(4byte)의 크기가 같은 이유 |
|---|
| 자료형 int는 운영체제에 따라서 2byte와 4byte로 사용할 수 있는 메모리 크기가 달라짐<br>- 16bit의 운영체제에서 int의 크기는 2byte, long의 크기는 4byte<br>- 32bit와 64bit 운영체제에서는 int의 크기는 4byte, long의 크기는 4byte<br>- 윈도우가 아닌 다른 운영체제에서는 long의 메모리 크기가 8byte인 경우도 있음 |

## □ 기본 자료형 - unsigned 정수 자료형




- 정수 자료형은 양수와 음수 모두 저장함
- **unsigned** 정수 자료형은 양수만 저장함 - 2배 넓은 범위의 값을 저장할 수 있음

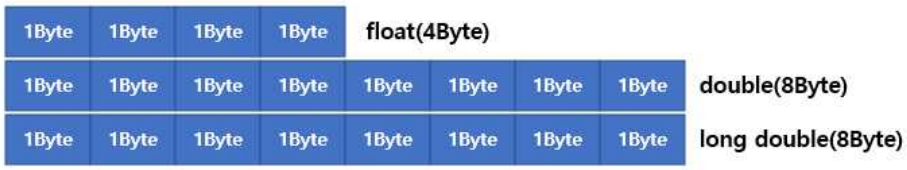
| 자료형<br>(Data Type) | 할당되는 메모리<br>크기 (Byte) | 표현 가능 데이터 범위 | 출력 서식 문자 |
|---|---|---|---|
| **unsigned** char | 1 | 0 ~ 255 | %u |
| **unsigned** short | 2 | 0 ~ 65535 | %u |
| **unsigned** int | 4 | 0 ~ 4294967295 | %u |
| **unsigned** long | 4 | 0 ~ 4294967295 | %lu |
| **unsigned** long long | 8 | 0 ~ $2^{64}$ - 1 | %llu |

**실습예제 3-4: 04integer.c**

**【생각해 보기】** 실행 결과는 ?

## 3.3 : 기본 자료형(Data Type)

### □ 기본 자료형 - 부동소수(실수) 자료형




| 자료형 (Data Type) | 할당되는 메모리 크기 (Byte) | 표현 가능 데이터 범위 | 출력 서식 문자 |
|---|---|---|---|
| float | 4 | $1.17549E^{-38}$ ~ $3.40282E^{38}$ | %f |
| double | 8 | $2.22507E^{-308}$ ~ $1.79769E^{308}$ | %f |
| long double | 8 | $2.22507E^{-308}$ ~ $1.79769E^{308}$ | %f |

※ long double은 운영체제와 플랫폼마다 크기가 다름

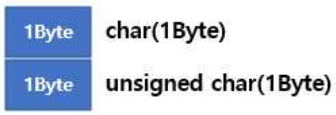
| 운영체제 | CPU(플랫폼) | 바이트 크기 | 비트 크기 |
|---|---|---|---|
| MS Windows | X86(32bit) | 8 | 64 |
| | X86-64(64bit) | 8 | 64 |
| Linux | X86(32bit) | 12 | 96 |
| | X86-64(64bit) | 16 | 128 |

**실습예제 3-5: 05floatdouble.c**

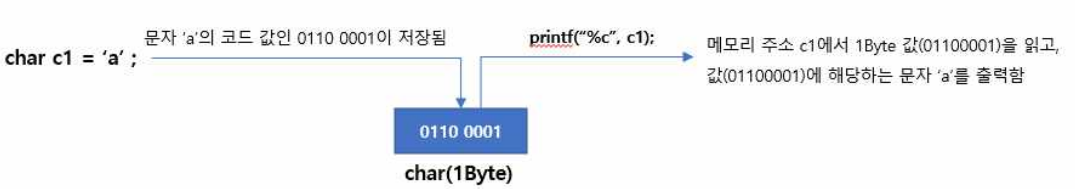
**【생각해 보기】** 실행 결과는 ?

## 3.3 : 기본 자료형(Data Type)

### □ 기본 자료형 - 문자 자료형




| 자료형 (Data Type) | 할당되는 메모리 크기 (Byte) | 표현 가능 데이터 범위 | 출력 서식 문자 |
|---|---|---|---|
| char | 1 | -128 ~ 127 (문자는 실제 0~127까지 사용) | %c |
| unsigned char | 1 | 0 ~ 255 | %c |




※ 표3-6 아스키 코드표 참조 (1Byte인 char로는 한글 문자 저장 불가)

실습예제 3-6: 06char.c

```
```

**【생각해 보기】** 실행 결과는 ?

```
```

## 3.3 : 기본 자료형(Data Type)

### □ 자료형의 크기 알아내기 - sizeof 연산자

실습예제 3-7: 07sizeflow.c

```
```

**【생각해 보기】** 실행 결과는 ?

```
```

## 3.4 : 상수 표현 방법

□ 상수(Constant)의 개념과 표현 방법

| 구분 | 정의 | 값 변경 | 메모리 역할 | 예시코드 |
|---|---|---|---|---|
| 상수 | 프로그램 실행 중 변하지 않는 값 | **불가능** | 고정된 값 저장 | #define PI 3.14<br>const int MAX = 100; |
| | #define PI 3.14 | 3.14 (이름은 있지만 변경은 안 됨) | | |

□ 상수(Constant)

- 데이터는 어딘가 저장되어야 다음에 사용할 수 있음
- 상수 : 저장된 값의 변경이 불가능한 공간 (예: 3.14, 10, 'A', "Hello World")
- **보통 상수는 이름이 없음**, 이런 상수를 **리터럴(literal) 상수**라고 함 (예: 3.14, 10)
- 이름을 붙일 수 있음, 이를 **기호(symbolic) 상수**라 함 (#define PI 3.14)

□ 상수(Constant) 표현법 (정수, 실수, 문자, 문자열)

| 자료형 | 설명 | 상수 예시 | 코드 예시 | 메모리 저장 예시 |
|---|---|---|---|---|
| **정수** | 소수점 없는 수 | **10, -25** | 20<br>const int NUM = 10;<br>#define YEAR 2025 | 이진 정수로 저장됨 |
| | • 정수 상수는 **10진수, 8진수, 16진수**로 표현 가능<br>▸정수 10 표현 방법: 10(10진수), 0xA(16진수). 012(8진수) | | | |
| **실수** | 소수점 있는 수 | **3.14, -0.5** | 3.33<br>const float DISTANCE = 2.89;<br>#define PI 3.14 | 부동소수점 방식으로 저장 |
| | • 실수 상수는 **소수점 형태, 지수 형태로** 표현 가능<br>▸**소수점 형태**: 3.14, -1.5, **-10.** ▸**지수 형태** : 3.14e-5, 0.314E-4 | | | |
| **문자** | 한 개 문자<br>(작은 따옴표) | **'A', '1'** | 'A'<br>const char grade = 'A';<br>#define TOP 'A' | (문자 1개) →<br>아스키코드 65 |
| | • 아스키(ASCII) 코드 : 문자를 메모리에 표현하기 위한 약속<br>▸문자 'A'의 아스키 코드 값 : 65 (0x41, 01000001) | | | |
| **문자열** | 문자들 묶음<br>(큰 따옴표) | **"Hello", "AB"** | "DONGYANG"<br>const char str[] = "AB";<br>#define NAME "DONGYANG" | 'A' + 'B' + '\0'<br>(널문자 포함) |

**실습예제** 3-10: 10const.c

**【생각해 보기】** 실행 결과는 ?