**1. 묵시적 형 변환(Implicit Conversion)**

* **기본 자료형 간의 변환을 컴파일러가 ‘알아서’ 함**
* 모든 기본 자료형 간의 변환이 가능한 것은 아님
* **특별한 문법이 필요하지 않음**

ex) int num1 = 100000;  
ex) long num2 = num1;

* **정수형을 부동소수점 실수형으로 변환할 때는 손실이 생길 수밖에 없다**
* 정밀도에서 차이가 발생하여 다른 값이 들어갈 수도 있음
* **정밀도 손실(Loss of Precision)**
* 아래의 경우 자료의 값이 달라지는 경우가 발생할 수 있음
* int/long에서 float으로 변환
* long에서 double로 변환
* **승격(Promotion)**
* 컴파일러가 자동으로 실수형이나 부동소수형 자료의 이진 표현을 확장
* 산술 연산자나 논리 연산자가 제대로 동작하게 하거나 ALI(산술논리장치)가 좀 더 효율적으로 돌 수 있게 하려고 사용  
  ex) double num1 = 3.7;  
  ex) int num2 = 10;  
  ex) Console.WriteLine(num1 + num2); // 연산의 결과는 double 형으로 승격됨
* **작은 형에서 큰 형으로 변환**
* ex) 32비트에서 64비트 형으로 변환  
  - 문제 없음(64비트가 32비트를 포함할 수 있음)
* **큰 형에서 작은 형으로 변환**
* 값이 충분히 작으면 문제가 없음  
  - 문제 없음(64비트에 32비트만 포함되어 있다면 32비트가 수용할 수 있음)  
  - 따라서 수용 가능한 비트 수를 넘어가면 문제가 발생함
* 프로그래밍 언어에 따라서 변환을 못하게 할 수 있음  
  - C, C++에서는 가능하지만 C#에서는 불가능
* 값이 크다면 문제가 발생함  
  - 런타임 중에 값이 어떻게 될지 모름  
  - 이런 상황을 ‘정보의 손실’이라고 함

**2. 명시적 형 변환**

* **두 double 형 수를 더하기**
* 이때 정수형으로 명시적 형변환을 한 경우 소수점은 버려지게 됨
* **명시적 형 변환**
* long형에서 int형으로  
  - long형의 비트 표현을 고려해 봤을 때 32비트까지 담긴 내용만을 int에 저장함  
  - 문제가 발생할 여지가 있음
* double에서 int형으로  
  - 컴파일러가 double형 값을 대상으로 int형으로 표현하고자 할 때 정수 부분만 표현해줌  
  - double형과 int형의 2진수 비트 패턴은 완전히 다른 방식으로 작동하기 때문에 연관성은 없음
* 모든 자료형이 변환되는 것은 아님  
  - 예를 들어, int형에서 bool형으로의 형변환은 불가능함
* **명시적 형 변환은 프로그래머의 의도를 나타냄**
* 강한 타입 언어인 C#에서는 묵시적 형 변환이 불가능한 상황에서 명시적으로 표현해주지 않는 경우에는 컴파일러가 오류를 표시함

**3. 산술 연산자**

**4. 증감 연산자**

* **부동 소수점형에는 사용하지 않는다**
* 원칙적으로는 사용이 가능하나, 상식적으로 어떻게 작동할지 모두가 만족할 만한 하나의 규칙이 나오기 굉장히 어려움  
  - 따라서 증감연산자는 **정수형에만 사용하는 것**이 실무적으로나 상식적으로나 적절함