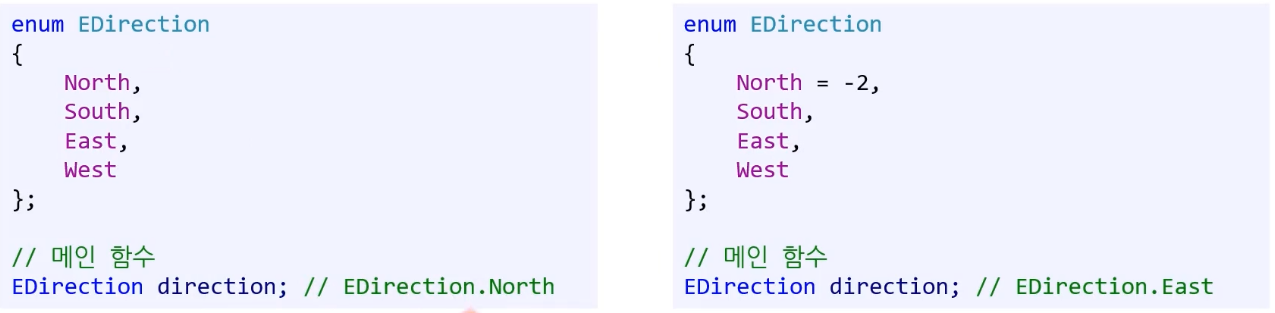
**1. 열거형**

* **열거형**
* 정수형 상수의 집합  
  - 부동소수점은 불가능함
* 각 원소마다 고유의 이름을 가짐
* 집합 역시 고유의 이름을 가짐
* Enum은 변수로 사용 가능
* **열거형의 정의**
* 정의는 함수 밖에서 함
* 첫 번째 원소의 기본 값은 0
* 아무 값도 대입해주지 않으면 원소의 값은 1씩 증가
* 각 원소에 원하는 값을 대입 가능  
  - 상수 및 계산식
* **열거형 변수의 정의 및 대입**



* 대입 없이 변수를 정의만 하면 값이 **0인 원소가 기본으로 할당**됨

텍스트이(가) 표시된 사진

자동 생성된 설명

* 열거형 변수의 정의와 대입은 위와 같이 한다
* 열거형 변수에는 해당 열거형의 원소만 대입 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

* **Enum을 사용하면 좋은 점**
* 코드가 읽기 좋아짐
* 다른 값이 들어가면 컴파일 오류 발생  
  - 문제 발생 여지를 사전 예방
* 함수 매개 변수로 쓰이면 함수가 요구하는 인자형을 빨리 알 수 있음  
  - 함수에 잘못된 값이 넘어가는 것을 예방
* **Enum과 정수형**

테이블이(가) 표시된 사진

자동 생성된 설명

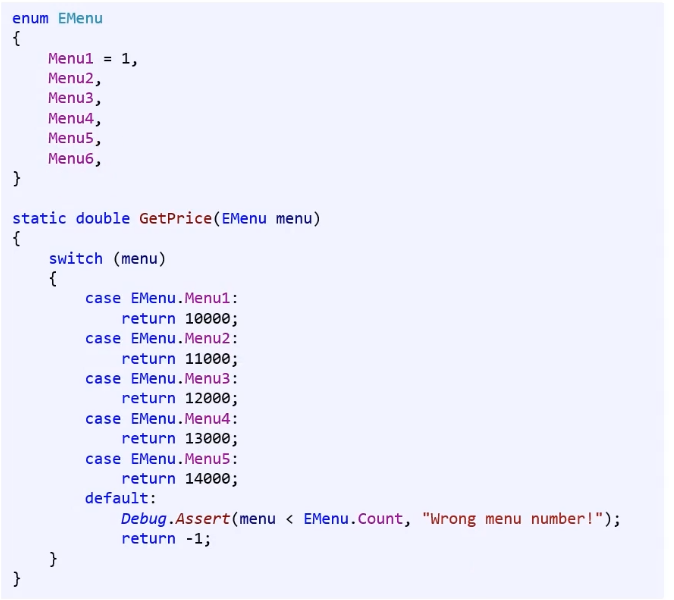
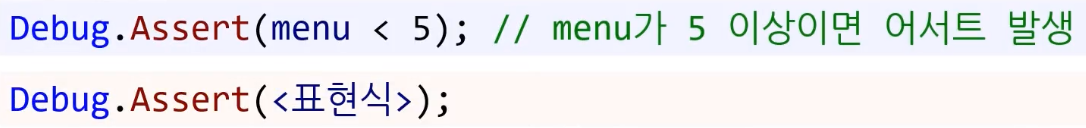
* **유사한 성질을 가진 정보들을 나열해야 할 때 정수형 상수 보다는 enum을 사용할 것**

텍스트, 시계, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**2. 어써트(assert)**

* **절대로 발생하지 않아야 하는 조건을 런타임 중에 검사**
* 만약 발생한다면 코드가 올바르게 동작하지 않는다는 의미
* 함수의 선조건 검사에 쓰기 적당함
* **디버그 모드에서만 동작함**
* 릴리즈 모드에서는 어써트 함수는 무시됨  
  - 주석처럼 무시됨
* 릴리즈 모드에서 동작하면 성능 저하가 발생
* **최종 제품의 성능저하 없이 개발 중에 문제를 고치는 바람직한 방법**
* **함수 시그니쳐가 약속하는 것**텍스트이(가) 표시된 사진

  자동 생성된 설명
* 함수명  
  - “두 수를 나눈 결과를 얻을 수 있는 함수구나”
* 매개변수  
  - “첫 번째 매개변수가 분자고, 두 번째 매개변수는 분모구나”
* 반환형  
  - “부동소수점형이 반환되는 구나”
* **어써트의 코드**
* **Debug.Assert()**
* 사용하기 위해서는 System.Diagnostics 라이브러리를 추가해야 함  
  
* Assert() 안에 들어가는 조건이 거짓일 때 프로그램은 일시 중단되고, 어써트 메시지가 출력창에 출력됨  
  - Assert() 안에 들어가는 조건이 언제나 참이라고 가정하고 프로그램을 짜고 있어야 함
* **Debug.Assert() 호출하기**
* 방식1 – 특별한 메시지 출력이 없는 경우  
  
* 방식2 – 메시지 출력이 있는 경우**(권장)**  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* 어써트는 내부의 조건은 항상 ‘참’임을 가정(단언)하는 조건을 넣어야 함  
  - 좋은 예) Assert(denominator != 0, “ ”); // denominator가 0이 아님을 가정하고 코드를 짰음을 의미  
  - 나쁜 예) Assert(denominator == 0, “ ”); // 따라서 이 구문은 적절한 예가 아님

**3. 함수와 범위**

* **기본적으로 어떤 범위 내에서 선언된 것은 범위 밖에서 쓰지 못함**텍스트이(가) 표시된 사진

  자동 생성된 설명
* **상위 범위에서 선언한 변수/상수는 하위 범위에서 사용 가능**텍스트이(가) 표시된 사진

  자동 생성된 설명
* name은 main 함수의 범위에 포함됨
* for문은 main 함수에 포함됨  
  - 따라서 for문 내에서 main 범위 내의 변수/상수들을 사용할 수 있음
* **‘기본적으로’ 함수 안에서 선언한 모든 것은 그 함수에서만 사용 가능**
* 지역 변수(Local Variable)라 부름
* **‘기본적으로’ 함수 밖에 있는 변수/상수는 사용할 수 없음**
* 지역 변수(Local Variable)라 부름
* **함수 매개변수, 반환값 모두 복사된 것**텍스트이(가) 표시된 사진

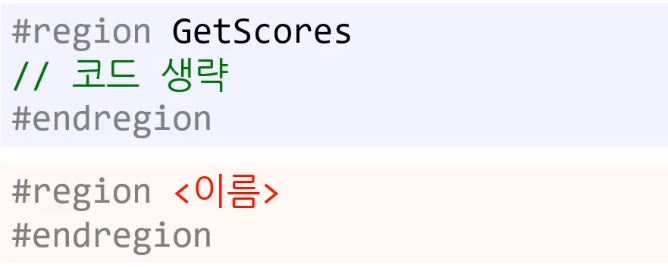
  자동 생성된 설명
* 값에 의한 전달(pass by value)이라고 함
* 올바른 값을 함수에 입력하면 약속된 값이 반환된다고 믿으면 고민 없이 그 함수를 호출 가능

**3. 값에 의한 전달, 참조에 의한 전달**

* **값에 의한 전달**
* 원본 변수와 매개 변수는 엄연히 다른 변수임  
  - 함수 매개변수에 원본 변수의 사본(값)이 전달(복사)됨
* 호출된 함수의 매개변수 값이 변경돼도 호출자 함수에 반영되지 않음
* **참조에 의한 전달**
* ref 키워드
* 원본 변수와 매개 변수가 같은 변수임  
  - 함수 매개변수에 원본 변수가 전달됨
* 호출된 함수의 매개변수 값이 변경되면 호출자 함수에 반영됨
* **Ref 키워드 (C# 전용)**
* 참조에 의한 전달을 위해 C#에서 사용
* 함수 호출 시 인자에 ref 키워드를 붙임
* Ref 키워드는 다른 프로그래밍 언어에 널리 쓰이지 않음  
  - 그러나 일부 언어에서 비슷한 개념이 존재 -> 참조에 의한 전달을 지원하고 싶어하기 때문

**4. 함수를 사용하는 상황**

* **처음부터 함수로 시작하지 말 것**
* 원칙상 쭉 코딩하는 것이 올바른 방법이긴 함
* **함수를 사용하는 적기**
* 현재 존재하는 혹은 향후에 발생 가능성이 높은 코드 중복을 피하고자 할 때
* 코드 중복은 좋지 않음  
  - 다음 사람이 중복 코드에 있는 버그를 고칠 때, 모든 코드를 수정할 것이라는 보장이 없기 때문
* **함수가 길어지면 동일한 이름의 지역변수가 생기는 경우가 있음**
* 중괄호를 사용하여 범위를 분리시키면 문제 해결 가능  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* **함수 대신 #region과 #endregion**
* C# 전용
* 긴 함수를 짧게 만들 수 있는 방법  
  - 하지만 실제로 함수 길이가 줄어드는 것은 아님
* 비주얼 스튜디오에서 코드를 접거나 펼 수 있게 해 줌  
  
* **결론: 함수는 코드 중복을 피하기 위해서 만들어라**
* 만약 처음부터 함수를 만들기 어렵다면 함수 없이 코드를 모두 다 작성하고, 그 다음에 함수로 분리하라