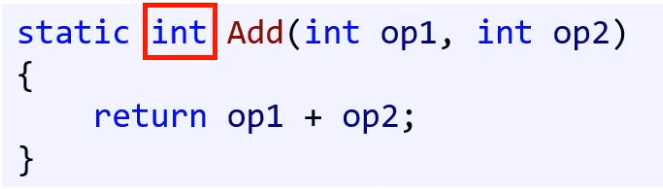
**1. 함수**

* **프로그래밍 세계에서의 함수**
* 함수에 **입력**이 들어가면 어떤 결과가 **출력**됨
* 중복되는 코드를 줄이는데 사용됨  
  - 수정 및 관리가 용이
* 자주 사용하는 코드의 재활용성이 올라감  
  - 함수 호출로 처리 가능
* **함수의 정의**

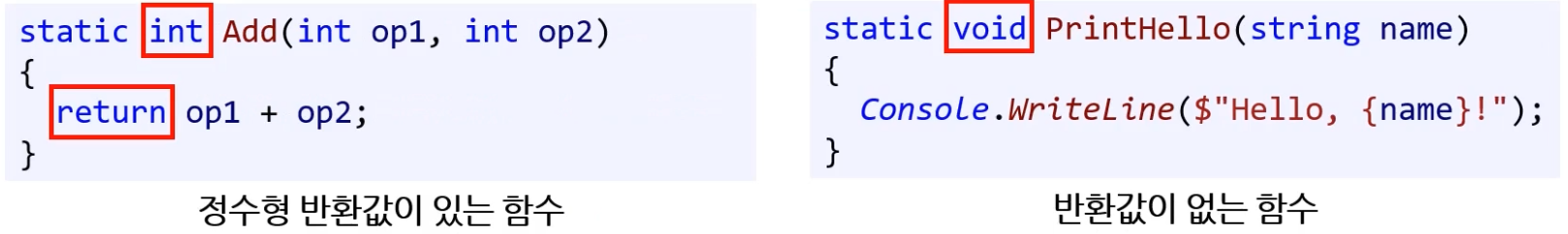
텍스트이(가) 표시된 사진

자동 생성된 설명

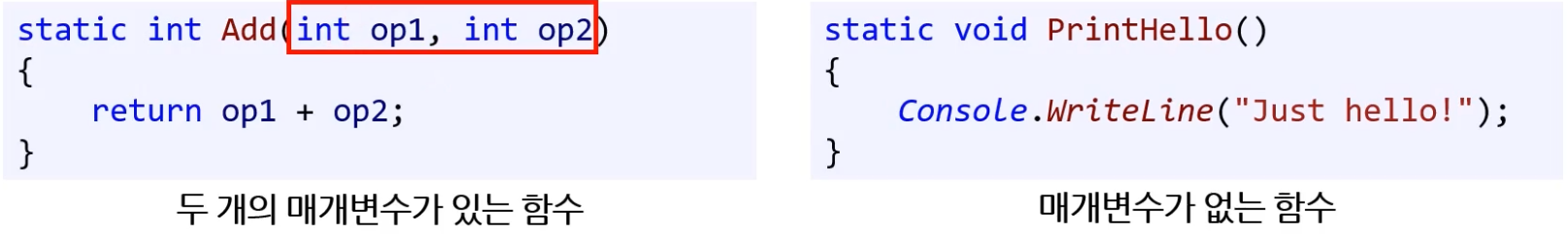
* 함수를 정의할 때는 다음의 것들을 포함해야 함  
  - 함수 시그니쳐  
   - 함수명(필수), 매개변수 목록(선택)  
  - 반환형(필수)  
  - 함수 바디(필수)
* **함수 – 반환형**



* 프로그래밍 세계에서의 **출력(output)**
* 반환형은 반드시 선언해야 함  
  - 안 하면 컴파일 오류
* 반환형이 없을 수도 있음  
  - 반환 값이 없다면 반환형에 void를 사용
* 반환형이 void가 아닐 경우  
  - 함수 바디에 return 키워드를 이용해 데이터를 반환해야 함  
  - 데이터를 반환하지 않으면 컴파일 오류



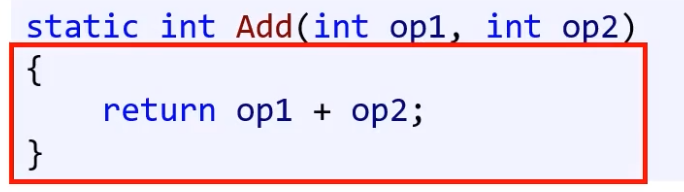
* **함수 – 매개변수**



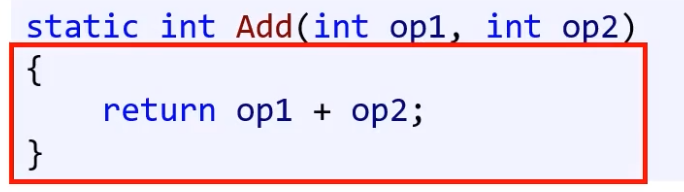
* 프로그래밍 세계에서의 입력
* 이를 매개변수(parameter)라고 부름
* 기본 자료형 뿐만 아니라 배열도 매개변수로 사용 가능
* 매개변수는 필수가 아님  
  - 함수는 0개 이상의 매개변수를 가짐
* 다른 용어로 인자(argument)라고도 함
* 엄밀히 말하면 매개변수와 인자는 다른 것  
  - 매개변수: 함수를 정의할 때 함수의 입력 값을 선언하는 것을 말함  
  - 인자: 함수를 호출할 때 함수로 전달되는 실제 값을 말함



* **함수 – 함수 바디**



* 함수의 기능을 구현한 코드 블록
* 중괄호를 이용하여 범위 표시
* 함수가 void가 아닌 반환형을 가지고 있다면 반드시 return 키워드를 사용하여 데이터를 반환해야 함
* **함수 – 함수명**



* 함수의 이름
* 엄밀히 말하면 함수 이름은 어떻게 짓든지 상관 없음
* 그러나 함수가 어떤 기능을 가졌는지 알 수 있게 지으면 가독성 증가  
  - 입력을 넣으면 내부구조(작동원리)를 알지 못하더라도 어떤 출력이 나오는지 알 수 있어야 함
* **함수는 반드시 블랙박스여야 함**
* 가독성을 위해
* 함수는 함수 호출자에 대해서 알아서는 안 됨
* 호출자가 함수 내부를 알 필요가 없게 **함수명을 명확하게** 지어야 함

**2. 코딩표준: 함수 명명 방법**

* **함수 - 정확하게 어떤 기능을 하는지 알려주는 단어 사용**
* 동사로 시작할 것 -> 함수는 행동(Action)이니까
* 제일 첫 글자는 대문자로
* 여러 단어를 연결한다면 두 번째 단어부터는 첫 글자를 대문자로  
  - 파스칼 표기법
* **변수 - 정확하게 어떤 정보를 담는지 알려주는 단어를 사용**
* 명사를 사용할 것 -> 함수는 행동(Action)이니까
* 제일 첫 글자는 소문자로
* 여러 단어를 연결한다면 두 번째 단어부터는 첫 글자를 대문자로  
  - 카멜 표기법
* **선조건과 후조건**
* 함수가 무슨 일을 하는지에 대한 약속
* 선조건  
  - 함수 실행 시작 전에 참으로 가정한 조건 – ex) Divide() 함수는 분모가 0이 아니어야 한다.  
  - 함수 이름이나 매개변수로 유추 가능하나, 부족하면 주석으로 추가 설명
* 후조건  
  - 함수 실행 후에 보장되는 조건 – ex) 두 정수를 더하면 정수의 결과가 나온다.  
  - 보통 함수 이름과 반환형으로 유추 가능
* 선조건을 만족하지 못하면 후조건을 보장할 수 없음
* **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”

**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

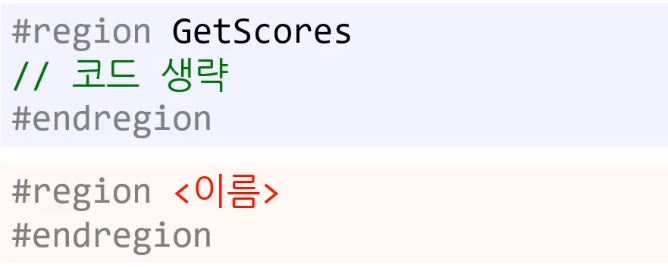
  자동 생성된 설명
* 값에 의한 전달(pass by value)이라고 함
* 올바른 값을 함수에 입력하면 약속된 값이 반환된다고 믿으면 고민 없이 그 함수를 호출 가능

**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