메소드(Method)

객체지향 프로그래밍 언어에서 사용하는 용어로 C와 C++에서는 함수(Function) 이라 불렸고, 파스칼에서는 프로시져(Procedure)라고 불렸습니다. 또는 서브루틴(Subroutine), 서브프로그램(Subprogram)이라고 부르기도 합니다.
메소드를 사용할때 매개변수 전달방식에 대해서 설명하고자 합니다.

참조에 의한 매개 변수 전달 (Call by reference)

C에서는 포인터를 이용하여 직접 데이터의 주소를 매개변수로 전달하여 함수에서 값을 컨트롤하였습니다.
C#에서는 프로그래머가 메모리를 참조한다는 것 자체가 위험하다고 판단하였기 때문에 포인터를 사용할 수 없게 했습니다. (하지만 설정으로 포인터를 사용가능하게 만들 수는 있습니다.)

Call by reference 를 이용하여 두개의 데이터의 값을 바꾸는 Swap 메소드 를 만들어 보겠습니다.

```
using System
; using System.Collections.Generic;
 using System.Linq; using System.Text;
 using System.Threading.Tasks;
namespace CsharpStudy
 {
 class Program
 {
 static void Swap(ref int x, ref int y)
 {
 int temp = x;
 x = y;
 y = temp
; }
 static void Main(string[] args)
{
int x = 3;
 int y = 5;
Swap(ref x, ref y);
 Console.WriteLine("{0} , {1}", x, y);
}
 }
 }
```

```
5 , 3
```

주의하실점은 매개변수를 전달할때, 함수에서 매개변수를 받을 때 ref 키워드를 붙여서 명확한 표현을 해주셔야 합니다.
그리고, C#에서는 전부 클래스 단위로 움직입니다. Main 메소드도 마찬가지로 클래스 안에 있습니다만, static 키워드로 정적으로 존재하기 때문에 Main에서 Swap메소드를 사용하기 위해서 Swap메소드도 static으로 선언되어야 합니다.

출력 전용 매개변수
메소드가 두개 이상의 반환 값을 갖고 싶을 때 사용합니다.

예를 들어 어떤 두 숫자를 가지고 몫과 나머지를 동시에 반환하고 싶을 때는 다음과 같이 함수를 만들면 됩니다.

```
void Divide ( int a , int b, ref int quotient, ref int remainder)
{
quotient = a/b;
remainder = a%b;
}
```

하지만 C#은 out 키워드를 이용한 출력전용 매개변수를 제공해 더욱 안전한 방법으로 사용할 수 있습니다.

```
using System;
using System.Collections.Generic;
using System.Linq; using System.Text;
 using System.Threading.Tasks;
 namespace CsharpStudy
 {
 class Program
 {
 static void Divide(int x, int y, out int quotient, out int remainder)
 {
 quotient = x / y;
 remainder = x % y;
 }
static void Main(string[] args)
 {
 int x = 5;
int y = 3;
int quo, rem; Divide(x, y, out quo, out rem);
 Console.WriteLine("quo: {0} , rem: {1}", quo, rem);
 }
 }
 }
```

```
quo: 1 , rem: 2
```

out 키워드를 사용할 경우
1. 메소드가 out 변수에 값을 저장하지 않을 경우 컴파일러가 에러를 발생합니다.
2. 호출된 메소드에서는 입력된 out 매개변수를 읽을 수 없고, 오직 쓰기만 가능합니다.(다른 용도로 사용되는 것을 금지합니다)

컴파일러가 에러를 발생한다는 것은 긍정적인 일입니다. 그 이유는 프로그래머에게 쉽게 에러를 고칠 수 있게 하지만, 런타임 오류는 프로그래머의 논리력으로 추적하여 고쳐야 하기 때문입니다.

메소드 오버로딩(Method Overloading)

메소드 오버로딩은 하나의 메소드 이름에 여러가지 구현을 overloading(과적) 하는 것입니다.
오버로딩을 할 경우 비슷한 기능을 하는 메소드를 매개 변수의 자료형이나 개수 등을 보고 여러가지 메소드 중에서 컴파일을 할때 컴파일러가 실행할 메소드를 선택하기 때문에, 프로그램의 성능 저하도 없을 뿐더러 한가지 메소드 이름으로 비슷한 기능의 메소드를 전부 묶을 수가 있습니다.
프로그래밍을 하다보면 클래스의 이름, 메소드의 이름 등 이름만 봐도 다른사람이 직관적으로 바로 알 수 있게 정하는 것이 아주 중요합니다. 이 때 유용하게 쓰입니다.

가변길이 매개변수

매개 변수의 수만 다른 같은 기능을 하는 메소드를 오버로딩 하고 싶을 때 사용할 수 있습니다.
메소드 오버로딩을 이용할 수도 있겠지만, 가변길이 매개변수는 수만 다른 경우 갯수만큼 메소드를 오버로딩 하는 것보다 한번에 형식은 같지만 변수의 개수가 정해져있는 것이 아닐 경우 가변길이 매개변수가 유용하게 쓰일 수 있습니다.

예를 들면 모든 수를 합하는 Sum 메소드를 가변길이 매개변수를 이용하여 만들어 보겠습니다.

```
using System;
using System.Collections.Generic;
 using System.Linq;
 using System.Text;
 using System.Threading.Tasks;
 namespace CsharpStudy
 {
 class Program
{
 static int Sum(params int[] args)
 {
 int sum = 0;
 foreach(int num in args)
 {
 sum += num;
 }
 return sum;
}
 static void Main(string[] args)
 {
 int result1 = Sum(2, 3, 5);
 int result2 = Sum(2, 3, 5, 7);
 int result3 = Sum(2, 3, 5, 7, 9);
 Console.WriteLine("result1 : {0} result2: {1} result3: {2}", result1, result2, result3);
 }
 }
 }
```

.

명명된 매개변수 (Named Parameter)

메소드를 호출할때 매개변수에 데이터를 넣을때 순서대로 지정합니다.
예를 들면 void Method(int a, int b, int c); 라는 메소드가 정의되어 있을때 Method(1,2,3); 로 호출하면 a,b,c 차례대로 값이 1,2,3이 들어가서 Method 기능을 수행합니다.
C#에서는 이런 방식외에 매개변수에 이름을 직접 명명하여 데이터를 할당할 수 있습니다.
매개 변수가 너무 많은 메소드를 사용할 때 어느 특정 매개변수에 어느 데이터를 할당 할때 유용합니다.

```
using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Text;
 using System.Threading.Tasks;
 namespace CsharpStudy
{
class Program
 {
static void ShowProfile(string name, int age)
 {
 Console.WriteLine("이름: {0} 나이: {1}", name, age);
 }
static void Main(string[] args)
 {
 ShowProfile("철수",10); ShowProfile(name: "영희", age: 15); ShowProfile(age: 20, name: "영수");
 }
 }
}
```

```
이름: 철수  나이: 10
이름: 영희  나이: 15
이름: 영수  나이: 20
```

주의할 점은 순서에 상관없이 매개변수 이름에 할당하기 때문에 하나라도 사용할 경우 전부 명명된 매개변수로 값을 할당해줘야야 합니다

선택적 매개변수
매개 변수에 디폴트 값을 넣어 매개변수를 전달하지 않을 경우 디폴트 값이 자동으로 할당됩니다.
명명된 매개변수와 함께 이용할 경우 매개변수가 많은 메소드에서 넣고 싶은 매개변수만 원하는 데이터를 쉽게 넣을 수 있기 때문에 유용하게 쓰입니다.

주의할 점은 메소드 오버로딩과 선택적 매개변수를 동시에 사용할 때의 모호함이 있으므로 같이 사용하면 안됩니다.
void Method(string str1="", string str2="");
void Method(string str1);
두 가지 메소드가 있을 경우
Method("abc"); 로 메소드 호출 할때 여러분이 컴파일러라면 어떤 함수를 호출해야 할지 판단이 되나요?

이상으로 C# 메소드에 대해 정리보았습니다.