메소드

Update 2023

01. 메소드 개요

- 메소드는 반복하여 사용하도록 이름 하나로 만들어 놓은 코드 집합이다.
- 특정한 기능을 처리하는 독립적인 하나의 단위 또는 모듈이다.
- C#에서는 이러한 함수(function)를 메서드(method)라고 한다.
- 메소드란 어떤 값을 받아서 그 값을 가지고 가공을 거쳐 어떤 결과값을 반환시켜 주는데 입력 값을 인자(Parameter)라고 하고 결과값을 반환할 경우에는 return 키워드를 사용한다.

02. 메소드의 종류

- 메소드는 내장 메소드와 사용자 정의 메소드로 구분할 수 있다.
- 내장 메소드는 C#이 자주 사용하는 기능을 미리 만들어서 제공하는 메소드로 사용 용도에 따라 문자열 관련 함수, 날짜 및 시간 관련 메소드, 수학 관련 메소드, 형식 변환 관련 메소드 등으로 나눌 수 있다. 이러한 내장 메소드를 API(Application Programming Interface)라고도 한다.
- 사용자 정의 메소드는 프로그래머가 필요할 때마다 새롭게 기능을 추가하여 사용하는 메소드이다.

03. 매개변수와 반환 값

- **매개변수(인자, 파라미터)**란 함수에 어떤 정보를 넘겨주는 데이터를 나타내며 콤마를 기준으로 여러 개 설정할 수 있다.

- **매개변수가 있는 메소드:** 특정 함수에 인자 값을 1개 이상 전달하는 방식으로 정수형, 실수형, 문자형, 문자열형, 개체형 등 여러 가지 데이터 형식을 인자 값으로 전달할 수 있다.
- **매개변수가 가변(여러 개)인 메소드 :** C#에서는 클래스 하나에 매개변수의 형식과 개수를 달리하여 이름이 동일한 **메소드**를 여러 개 만들 수 있으며 이를 가리켜 함수 중복 또는 함수 오버로드(overload)라고 한다.
- **반환값이 있는 메소드(결과값이 있는 메소드): 메소드**의 처리 결과를 함수를 호출한 쪽으로 반환할때는 return 키워드를 사용하여 데이터를 돌려줄 수 있다.

04. 메소드 정의와 호출

- 메소드 정의하기

```
static void 메소드명(매개변수)
{
        명령문;
}
```

```
static 데이타형 메소드명(매개변수)
{
        …
        return 값;
}
```

- 메소드 호출하기

```
메소드명();
메소드명(매개변수값);
데이타형 결과변수 = 메소드명(매개변수값);
```

04. 메소드 정의와 호출

- 함수 선언은 함수를 호출하기에 코드가 앞에 위치해야 하지만 C#에서는 Main() 메서드 앞 또는 뒤에 위치해도 상관없다.
- 반환값이 없는 경우 즉 return 문이 없다면 void 키워드를 사용한다

```
// 매개변수 X, 반환값 X
참조 1개
static void show()
{
    Console.WriteLine("Hellow world");
}
```

04. 메소드 정의와 호출

```
참조 0개
static void Main(string[] args)
{
    // 함수 호출
    show();

}
```

05. 매개변수가 있는 메소드

- 메서드 정의시 괄호 안에는 매개변수를 선언할 수 있는데 콤마 기호를 구분으로 하나 이상 줄 수 있으며 함수를 호출할 때는 동일한 데이터 형식으로 리터럴값을 전달해야 한다

```
// 매개변수 o, 반환값 x
참조 2개
static void showMessage(string message)
{
    Console.WriteLine("message = {0}", message);
}
```

```
// 매개변수가 있는 메서드 호출
showMessage("Good morning");
showMessage("Good Night");
```

05. 매개변수가 있는 메소드

```
// 매개변수가 여러개인 경우 o, 반환값 x
참조 0개
static void showMessage2(string user, int clock, string message)
{
    Console.WriteLine($"{user}님, {clock}시 입니다. {message}");
}
```

```
// 매개변수가 여러개인 있는 메서드 호출
showMessage2("박은우", 12, "운동할 시간입니다...");
showMessage2("마동호", 9, "택배가 도착했습니다...");
```

```
박은우님, 12시 입니다. 운동할 시간입니다...
마동호님, 9시 입니다. 택배가 도착했습니다...
```

퀴즈

- 1부터 100까지 숫자 중 5의 배수만 출력하는 메서드를 정의하고 호출하여라.
- 한 줄에 3개씩 출력하여야 한다.
- 메서드 호출과 출력 예시 화면은 아래를 참조한다

```
// 1~100 안에서 5의 배수 출력 함수 호출
PrintNumber();
```

```
 5   10   15
20   25   30
35   40   45
50   55   60
65   70   75
80   85   90
95
```

퀴즈

- 구구단 전체 출력 메소드를 정의하고 호출하여라
- 구구단 출력 메서드 호출과 출력 예시 화면은 아래를 참조한다

```
Console.WriteLine("구구단 출력함수 호출");
GuguShow();
```

```
----------------------
      출력화면
----------------------
2 X 1 = 2
...

9 X 9 = 81
```

퀴즈

- "Q"나 "q"를 입력하여 종료하기 전까지는 문자열을 계속 입력받게 한 후 입력 종료시 입력 받은 모든 문자열을 출력하는 메소드를 작성하고 호출하여라.
- 구구단 출력 메서드 호출과 출력 예시 화면을 아래를 참조한다
- 아무것도 입력 받지 않는 경우 (처음 실행시 바로 "Q"나 "q" 입력)에는 특정 메시지가 출력되도록 한다.

```
// 메소드 호출
Fridge();
```

```
// 출력 화면 예시 1 - 처음에 q나 Q를 입력한 경우
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... Q
냉장고에는 [물 ]가(이) 있다
```

퀴즈

```
// 출력 화면 예시 2
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... 소고기
----------
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... 삼겹살
----------
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... 치킨
----------
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... 피자
----------
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... 수박
----------
입력 종료(q/Q)
우리집 냉장고에 있는 것은? ... q
```

냉장고에는 [물 소고기 삼겹살 치킨 피자 수박]가(이) 있다

퀴즈

- 매개 변수 값이 있는 형태로 특정 숫자의 구구단을 출력할 수 메소드를 정의하고 호출하여라.
- 메서드 호출과 출력 예시 화면을 아래를 참조한다

```
// 특정 숫자의 구구단 출력  메소드 호출
GuguShow2(2);
```

```
----------------------
      출력화면
----------------------
2 X 1 = 2
...
```

퀴즈

- 1부터 n까지의 누적합을 구하는 메소드를 정의하고 호출하여라.
- 메서드 호출과 출력 예시 화면을 아래를 참조한다

```
// 100까지 누적합 메소드 호출
Nsum(100);

// 1000까지 누적합 메소드 호출
Nsum(1000);
```

```
// 출력화면
1부터 100까지의 합은? 5050
1부터 1000까지의 합은? 500500
```

퀴즈

- 1부터 100사이에서 특정 숫자의 배수만 출력하는 메소드를 정의하고 호출하여라.
- 메서드 호출과 출력 예시 화면을 아래를 참조한다

```
// 1~100 사이에서 7 배수를 한 줄에 7개씩 배치
PrintNumber2(7, 7);

// 1~100 사이에서 11 배수를 한 줄에 2개씩 배치
PrintNumber2(11, 2);
```

```
  7   14   21   28   35   42   49
 56   63   70   77   84   91   98
```

```
11   22
33   44
55   66
77   88
99
```

06. 반환 값이 있는 메소드

- 반환값(return value)은 메서드에서 명령어를 실행한 후 그 결과를 다시 함수를 호출한 부분으로 되돌려 주는 결과값을 의미하며 return이라는 키워드를 이용한다.
- 반환값이 없는 메서드의 경우 메서드 정의시 void로 지정해야 한다.
- 반환값이 있는 메서드의 경우 반환값이 없다는 의미인 void 키워드 자리에 반환되는 데이터 형식을 지정해야 한다.

```
static void 메소드명(매개변수)
{
        명령문;
}
```

```
static 데이타형 메소드명(매개변수)
{
        …
        return 값;
}
```

06. 반환 값이 있는 메소드

```
// 매개변수 x, 반환값 o
// 반환값이 문자열인 경우
참조 1개
static string GetMessage()
{
    return "message = Hellow Wordl";
}


    // 반환값이 있는 함수 호출
    string message = GetMessage();
    Console.WriteLine($" return value : {message} ");
```

06. 반환 값이 있는 메소드

```
// 매개변수 o, 반환값 o
// 반환값이 정수형인 경우
// 사각형의 넓이를 반환하는 메서드 정의
참조 1개
static int GetSquare(int x, int y)
{
    return x * y;
}
```

06. 반환 값이 있는 메소드

```
// 매개변수와 반환값이 있는 메서드 호출
// 사각형의 가로와 세로 길이를 매개변수로 전달
// 넓이 구하기 함수 호출
Console.WriteLine("\n 사각형 넓이 구하기 ");
int x = 20, y = 10;
int r = GetSquare(x, y);
Console.WriteLine($" 가로 : {x}, 세로 : {y}, 넓이 : {r} ");
```

06. 반환 값이 있는 메소드

```
static bool checkNumber(int number)
{
    if (number % 2 != 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```
100은 홀수인가? False
23은 홀수인가? True
```

```
int n = 100;
Console.WriteLine($"{n}은 홀수인가? {checkNumber(n)}");
n = 23;
Console.WriteLine($"{n}은 홀수인가? {checkNumber(n)}");
```

퀴즈

- 정수형으로 정의된 세 수의 곱하기 결과값을 반환하는 메소드를 정의하고 호출하여라.
- 메서드 호출 예시 화면은 아래를 참조한다
- 메서드 안에는 return 문이 존재 해야 한다.

```
int x=78, y=5, z=11;
Console.WriteLine($"{x} X {y} X {z}의 결과값은? ");

// 함수 호출
Console.WriteLine(ThreeMultiplied(x, y, z));
```

07. 기본 매개 변수

- 기본 매개변수(default parameter) 또는 선택적 인수(optional argument)는 매개변수의 초기값이 지정되어 있는 형태로 매개변수 값을 지정하지 않아도 기본값이 자동 설정된다.
- 필요에 따라 데이터를 할당하거나 할당하지 않아도 된다는 특징이 있다.

```
static void 메소드명(매개변수=값, ...)
{
        명령문;
}
```

```
static 데이타형 메소드명(매개변수=값, ...))
{
        ...
        return 값;
}
```

07. 기본 매개 변수

```
// 매개변수 o, 반환값 x
참조 4개
static void getNumber(int x = 0, int y = 0, int z = 0)
{
    Console.WriteLine("\t x = {0}, y = {1}, z = {2} \n", x, y, z);
}
```

```
getNumber();
getNumber(10);
getNumber(10, 20);
getNumber(10, 20, 30);
```

```
x = 0, y = 0, z = 0

x = 10, y = 0, z = 0

x = 10, y = 20, z = 0

x = 10, y = 20, z = 30
```

07. 기본 매개 변수

```
// 매개변수 o, 반환값 o
참조 4개
static string getNumberSum(int x = 0, int y = 0, int z = 0)
{
    return $"\t {x} + {y} + {z} = {x+y+z}\n";
}
```

```
Console.WriteLine(getNumberSum());
Console.WriteLine(getNumberSum(10));
Console.WriteLine(getNumberSum(10, 20));
Console.WriteLine(getNumberSum(10, 20, 30));
```

```
0 + 0 + 0 = 0

10 + 0 + 0 = 10

10 + 20 + 0 = 30

10 + 20 + 30 = 60
```

07. 기본 매개 변수

```
// 매개변수 o, 반환값 x
// 매개변수 일부만 초기값이 지정되어 있는 형태
static void getNumber2(int x, int y = 0, int z=0)
{
    Console.WriteLine("\t x = {0}, y = {1}, z = {2} \n", x, y, z);
}
```

```
//getNumber2(); // 오류발생
getNumber2(10);
getNumber2(10, 20);
getNumber2(10, 20, 30);
```

```
x = 10, y = 0, z = 0

x = 10, y = 20, z = 0

x = 10, y = 20, z = 30
```

07. 기본 매개 변수

```
static string getCalc(char sign, int x = 0, int y = 0)
{
    if (sign == '+')
    {
        return $"\t {x} + {y}  = {x + y}\n";
    }
    else if (sign == '-')
    {
        return $"\t {x} - {y}  = {x - y}\n";
    }
    else
    { return "계산 오류"; }
}
```

07. 기본 매개 변수

```
// 매개변수 o, 반환값 o
// 기호에 따라 더하기 빼기 수행
Console.WriteLine(getCalc('*'));
Console.WriteLine(getCalc('+'));
Console.WriteLine(getCalc('+', 10));
Console.WriteLine(getCalc('+', 10, 20));
Console.WriteLine(getCalc('-'));
Console.WriteLine(getCalc('-', 10));
Console.WriteLine(getCalc('-', 10, 20));
```

```
계산 오류
        0 + 0  = 0

        10 + 0  = 10

        10 + 20  = 30

        0 - 0  = 0

        10 - 0  = 10

        10 - 20  = -10
```

퀴즈

- 원의 넓이를 구하는 메소드를 정의하고 호출하여 출력 화면과 같은 결과 화면이 나오도록 프로그래밍 하여라.
- 반지름을 지정하지 않은 경우에는 넓이가 0으로 출력되어야 한다.

(메소드 지정시 기본 매개변수 값 정의)

```
// 매개변수를 전달하지 않음 => 기본매개변수값은 0
CircleArea();

// 반지름 5로 전달
CircleArea(5);

// 반지름 12로 전달
CircleArea(12);
```

```
// 출력화면
원의 넓이는? 0 X 0 X 3.14 = 0.000
원의 넓이는? 5 X 5 X 3.14 = 78.500
원의 넓이는? 12 X 12 X 3.14 = 452.160
```

퀴즈

- 국어, 영어, 수학 점수를 입력 받아 총점, 평균, 합격 여부를 출력하도록 프로그래밍하여라.
- 메서드 호출은 예시를 참조하고 국어, 영어, 수학 점수 데이터가 없는 경우 0점으로 처리한다.
- 합격은 평균 70점 이상으로 한다.

퀴즈

```
// 성적표 메서드 호출 예시
gradePrint();
gradePrint(80);
gradePrint(80, 95);
gradePrint(80, 95, 70);
gradePrint(77, 35, 70);
```

```
국어 : 0 영어 : 0 수학 : 0
총점 : 0 평균 : 0.00 => 불합격
==================================
국어 : 80 영어 : 0 수학 : 0
총점 : 80 평균 : 26.67 => 불합격
==================================
국어 : 80 영어 : 95 수학 : 0
총점 : 175 평균 : 58.33 => 불합격
==================================
국어 : 80 영어 : 95 수학 : 70
총점 : 245 평균 : 81.67=> 합격
==================================
국어 : 77 영어 : 35 수학 : 70
총점 : 182 평균 : 60.67 => 불합격
```

08. 명명된 매개변수

- 명명된 매개변수(named parameter)를 사용하면 메소드를 호출할 때 필요한 매개변수 이름을 직접 지정할 수 있다 . 콜른(:)을 이용하여 매개변수와 값을 지정한다.
- 메서드 호출
  매개변수명(변수명:리터럴값);
- 명명된 매개변수를 사용할 경우에는 매개변수의 순서가 변경되어도 된다.

```
static 자료형 메소드명(매개변수)
{
        명령문;
}
```

```
메소드명(매개변수:값, ...);
```

08. 명명된 매개변수

```
// 매개변수 o, 반환값 x
참조 2개
static void printProfile(string name, string phone)
{
    Console.WriteLine($" 이름 : {name}");
    Console.WriteLine($" 모바일 : {phone}");
    Console.WriteLine();
}

printProfile(name: "장원영", phone: "010-1234-7890");
printProfile(phone: "010-6789-0000", name: "이승기");
```

08. 명명된 매개변수

```
// 매개변수 o, 반환값 o
// 기호에 따른 계산 결과 출력
참조 4개
static string calculator(char symbol, double n1, double n2)
{
    if (symbol == '+') return $"{n1} + {n2} = {(n1 + n2):f2}";
    else if (symbol == '-') return $"{n1} - {n2} = {(n1 - n2):f2}";
    else if (symbol == '*') return $"{n1} x {n2} = {(n1 * n2):f2}";
    else if (symbol == '/') return $"{n1} / {n2} = {(n1 / n2):f2}";
    else if (symbol == '%') return $"{n1} % {n2} = {n1 % n2}";
    else return "계산 오류";
}
```

08. 명명된 매개변수

```
// 명명된 매개변수 함수 호출
Console.WriteLine("\t계산식1 : "+ calculator(symbol:'*', n1:34.5, n2:4.569));
Console.WriteLine("\t계산식2 : " + calculator(n1:52, n2: 7, symbol: '%'));
Console.WriteLine("\t계산식3 : " + calculator(n2:25.89, n1: 33.5, symbol:'/'));
Console.WriteLine("\t계산식4 : " + calculator(n2: 34, n1: 267.89, symbol: '^'));
```

```
계산식1 : 34.5 x 4.569 = 157.63
계산식2 : 52 % 7 = 3
계산식3 : 33.5 / 25.89 = 1.29
계산식4 : 계산 오류
```

퀴즈

- 고객의 키와 몸무게를 이용하여 bmi 지수와 메시지를 출력하도록 프로그래밍하여라.
- 메서드 정의 영역에는 매개변수와 return 문이 있어야 한다.
- 메서드 호출은 예시를 참조한다.
- Bmi 지수 산출과 관련 메시지

```
//    BMI지수 = 몸무게(kg) ÷ (키(m) × 키(m))
//     BMI < 20, 저체중
//     20 <= BMI < 25, 정상체중
//     25 <= BMI < 30, 경도비만
//     30 <= BMI < 40, 비만
//     BMI >= 40, 고도비만
```

퀴즈

```
// 퀴즈 : 명명된 매개변수 함수 호출
// 키, 몸무게 => bmi지수 메세지
Console.WriteLine($"고객001 167.5cm , 55.7kg => BMI {printBMI(height: 167.5, weight: 55.7)}");
Console.WriteLine($"고객002 188cm , 89.1kg => BMI {printBMI(weight:89.1, height:188)} ");
Console.WriteLine($"고객003 145cm , 50.35kg => BMI {printBMI(weight: 50.35, height: 145)}");
```

```
고객001 167.5cm , 55.7kg => BMI 19.85 저체중
고객002 188cm , 89.1kg => BMI 25.21 경도비만
고객003 145cm , 50.35kg => BMI 23.95 정상체중
```

09. 가변길이 매개변수

- 매개변수의 길이가 가변인 메소드로 매개변수의 개수가 유연하게 달라질 수 있다.
- 매개변수는 배열형태의 args로 지정되며 메소드 정의시 **params 자료형[] args** 키워드를 사용한다.

```
static 자료형 메소드명(params 자료형[] args)
{
        명령문;
}
```

```
메소드명(매개변수값1);
메소드명(매개변수값1, 매개변수값2, 매개변수값3);
```

09. 가변길이 매개변수

```
// 매개변수 o, 반환값 x
참조 3개
static void printTxt(params string[] args)
{
    Console.WriteLine("매개변수 전체 갯수 = {0}", args.Length);
    // 가변매개변수 args의 각각의 값을 출력
    for (int i = 0; i < args.Length; i++)
    {
        Console.WriteLine($"{i} => {args[i]}");
    }
}
```

09. 가변길이 매개변수

```
// 가변길이 매개변수 값을 모두 출력하는 메서드 호출
printTxt("바나나");
printTxt("바나나", "사과", "포도");
printTxt("오렌지", "바나나", "토마토", "메론", "석류");
```

```
매개변수 전체 갯수 = 1
0 => 바나나

매개변수 전체 갯수 = 3
0 => 바나나
1 => 사과
2 => 포도

매개변수 전체 갯수 = 5
0 => 오렌지
1 => 바나나
2 => 토마토
3 => 메론
4 => 석류
```

09. 가변길이 매개변수

```
// 매개변수의 모두의 합 구하기
// 매개변수 o, 반환값 o
참조 3개
static int getParameterSum(params int[] args)
{
    int sum = 0;
    for (int i = 0; i < args.Length; i++)
    {
        sum += args[i];
    }
    return sum;
}
```

09. 가변길이 매개변수

```
Console.WriteLine("{0}+{1}={2}", 10, 30, getParameterSum(10, 20));
Console.WriteLine("{0}+{1}+{2}={3}", 10, 30, 50, getParameterSum(10, 30, 50));
Console.WriteLine("{0}+{1}+{2}+{3}={4}", 10, 30, 50, 100,
                        getParameterSum(10, 30, 50, 100));
```

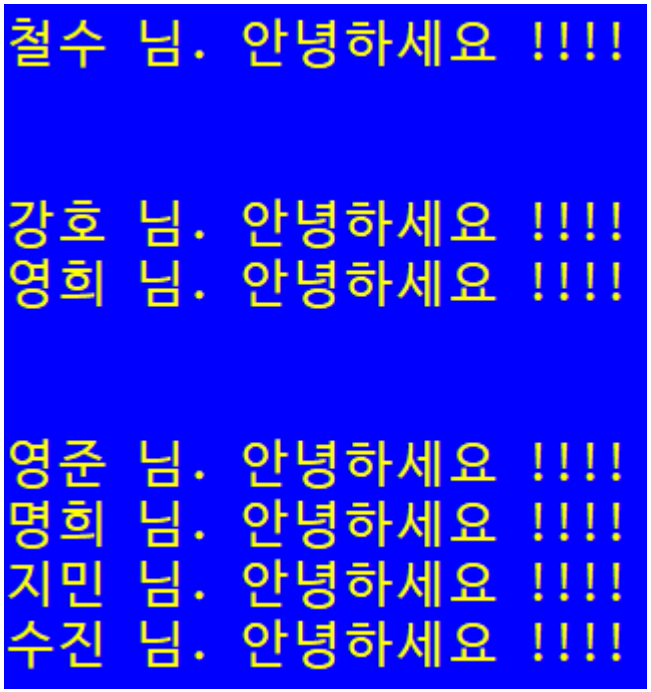
```
10+30=30
10+30+50=90
10+30+50+100=190
```

퀴즈

- 가변길이 매개변수 방식으로 메소드를 정의하여 메소드를 호출하면 출력화면이 표시되도록 프로그래밍 하여라.

```
// 가변 매개변수 메소드 호출 = 인사말 출력
printMessage("철수");
printMessage("강호", "영희");
printMessage("영준", "명희", "지민", "수진");
```

퀴즈


철수 님. 안녕하세요 !!!!
강호 님. 안녕하세요 !!!!
영희 님. 안녕하세요 !!!!
영준 님. 안녕하세요 !!!!
명희 님. 안녕하세요 !!!!
지민 님. 안녕하세요 !!!!
수진 님. 안녕하세요 !!!!

퀴즈

- 가변길이 매개변수 방식으로 메소드를 정의하여 호출하면 매개변수 전달 값 중 가장 큰 수가 출력되도록 프로그래밍 하여라.
- 매개 변수 값이 없거나 1개인 경우에는 오류 메세지를 출력한다.

```
Console.WriteLine(getMaxNumber());
Console.WriteLine(getMaxNumber(55.5));
Console.WriteLine(getMaxNumber(12.678, 55.5));
Console.WriteLine(getMaxNumber(123.5, -90, 678));
Console.WriteLine(getMaxNumber(-90, 678, 999.456, 123));
```

```
오류발생
오류발생
큰수는? 55.5
큰수는? 678
큰수는? 999.456
```

10. 메소드 오버로딩

- 하나의 메소드 이름에 여러 개의 메소드를 정의하는 기능으로 메소드 다중 정의라고 한다.
- 자료형과 매개변수 갯수등을 달리해서 이름이 동일한 메소드를 정의할 수 있다.

```
// 두수의 합을 반환 => 매개변수 정수형, 반환값도 정수형
static int Plus(int x, int y)
{
    return x + y;
}
// 두수의 합을 반환 => 매개변수 정수형, 반환값은 실수형
static double Plus(double x, double y)
{
    return x + y;
}
```

10. 메소드 오버로딩

```
// 메소드 오버로딩 호출
Console.WriteLine(" 정수형 Plus() 메서드 호출 => " + Plus(12, 56));
Console.WriteLine(" 실수형 Plus() 메서드 호출 => " + Plus(3.14, 7.8));
```

```
정수형 Plus() 메서드 호출 => 68
실수형 Plus() 메서드 호출 => 10.94
```

10. 메소드 오버로딩

```
// 메서드 오버로딩 - 매개변수의 데이타형과 갯수가 틀린 경우
// 매개변수 X, 반환값 X
참조 1개
static void Hi()
{
    Console.WriteLine("Hi!!!");
}
```

```
// 매개변수 1개 , 반환값 X
참조 1개
static void Hi(string msg)
{
    for (int i = 0; i < 3; i++)
    {
        Console.WriteLine(msg);
    }
}
```

10. 메소드 오버로딩

```
// 매개변수 2개 , 반환값 X
// 매개변수 n 만큼 메세지 출력하기
참조 1개
static void Hi(string msg, int n)
{
    for (int i = 0; i < n; i++)
    {
        Console.WriteLine($"{i} => {msg}");
    }
}
```

```
// 메세지를 출력하는 3개의 메서드 오버로딩 호출
Hi();
Console.WriteLine("============");
Hi("Hello C#");
Console.WriteLine("============");
Hi("Good Night", 10);
```

10. 메소드 오버로딩

```
Hi!!!
===========
Hello C#
Hello C#
Hello C#
===========
```

```
0 => Good Night
1 => Good Night
2 => Good Night
3 => Good Night
4 => Good Night
5 => Good Night
6 => Good Night
7 => Good Night
8 => Good Night
9 => Good Night
```

퀴즈

- 메소드 오버로딩 기능을 이용하여 Caculator 메소드를 동일한 이름으로 정의하여 두수의 정수형 int 사칙연산, 세수의 double 실수형 사칙연산 메소드로 호출할 수 있도록 프로그래밍 하여라

```
// 정수 int 형으로 메소드 호출
Calculator(100, 4);

// 실수 double 형으로 메소드 호출
Calculator(15.78, 3.456, 23.456);
```

퀴즈

```
// 출력 화면
----------------
정수 int형의 2수의 사칙연산
----------------
100 + 4 = 104
100 - 4 = 96
100 * 4 = 400
100 / 4 = 25
```

```
----------------
실수 double형의 3수의 사칙연산
----------------
15.78 + 3.456  + 23.456 = 42.69
15.78 - 3.456 - 23.456 = -11.13
15.78 * 3.456 * 23.456 = 1279.19
15.78 / 3.456 / 23.456 = 0.19
```

퀴즈

- 메소드 오버로딩 기능을 이용하여 원, 사각형, 사다리꼴의 면적을 구하는 GetArea 메서드를 호출할 수 있도록 프로그래밍 하여라

```
// 타원, 사각형, 사다리꼴 면적을 구하는 메서드 오버로딩 호출
Console.WriteLine($"반지름 2.5cm인 타원의 면적 = " +
    $"{GetArea(2.5)}cm");
Console.WriteLine($"가로 5.5cm, 세로 7.5cm인 사각형 면적 = " +
    $"{GetArea(5.5, 7.5)}cm");
Console.WriteLine($"윗변 2.5cm, 아랫변 4.3cm, 높이 5.2cm인 사다리꼴의 면적 = " +
    $"{GetArea(2.5, 4.3, 5.2)}cm");
```

```
반지름 2.5cm인 타원의 면적 = 19.625cm
가로 5.5cm, 세로 7.5cm인 사각형 면적 = 41.25cm
윗변 2.5cm, 아랫변 4.3cm, 높이 5.2cm인 사다리꼴의 면적 = 17.68cm
```