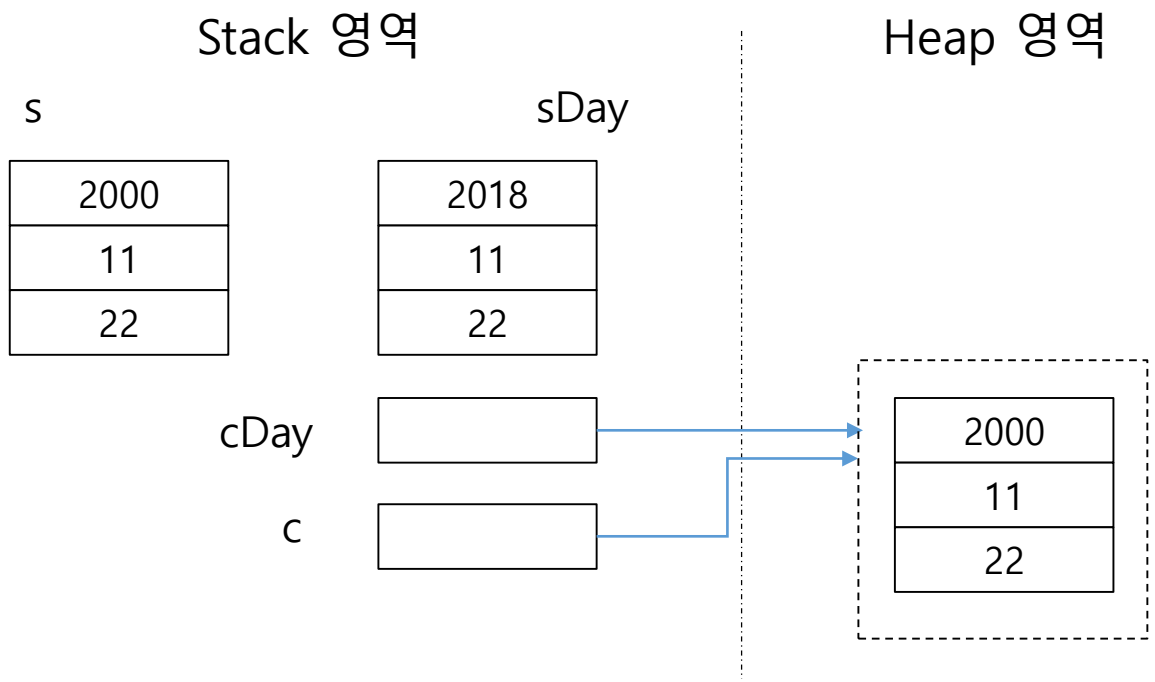
객체지향 프로그래밍

C# - Class

KYUNGSUNG UNIVERSITY SINCE 1955

1) 구조체와 클래스는 class와 struct 키워드만 빼고 구문이 똑같다.
2) 구조체는 값형이고 클래스는 참조형이다. 즉 구조체는 메모리의 스택 영역에 공간을 갖게 되고 클래스는 스택에 참조만 위치하게 된다. 클래스는 반드시 new 키워드를 사용하여 객체를 생성한다.
3) 구조체와 클래스는 모두 세 가지 위치에 만들 수 있다. ① 클래스 내 ② 같은 파일의 클래스 밖 ③ 다른 파일에 만든다. 클래스 내에 만들면 그 클래스에서만 사용할 수 있고, 다른 파일에 만들 때는 같은 프로젝트 안에 파일이 있어야 한다.
4) 구조체와 클래스는 모두 new 키워드로 만들 수 있는데, 이 경우에는 필드의 값들이 모두 디폴트 값으로 초기화된다.



클래스와 구조체는 .NET Framework의 공용 형식 시스템의 기본 구문입니다. 각각은 기본적으로 하나의 논리 단위에 속하는 '데이터' 및 '동작 ' 을 캡슐화하는 데이터 구조입니다.
클래스나 구조체의 멤버는 데이터와 동작을 정의하며 필드, 메소드, 속성, 이벤트 등을 포함합니다.

```
using System;

namespace A065_ClassAndStruct
{
  struct DateStruct
  {
    public int year, month, day;
  }

  class DateClass
  {
    public int year, month, day;
  }
  class Program
  {
    static void Main(string[] args)
    {
      DateStruct sDay;
      sDay.year = 2018;
      sDay.month = 11;
      sDay.day = 22;
      Console.WriteLine("sDay: {0}/{1}/{2}", sDay.year, sDay.month, sDay.day);
```

```
        DateClass cDay = new DateClass();
        cDay.year = 2018;
        cDay.month = 11;
        cDay.day = 22;
        Console.WriteLine("cDay: {0}/{1}/{2}", cDay.year, cDay.month, cDay.day);

        DateStruct sDay2 = new DateStruct();
        Console.WriteLine("sDay2: {0}/{1}/{2}", sDay2.year, sDay2.month, sDay2.day);
        DateClass cDay2 = new DateClass();
        Console.WriteLine("cDay2: {0}/{1}/{2}", cDay2.year, cDay2.month, cDay2.day);

        DateStruct s = sDay;
        DateClass c = cDay;

        s.year = 2000;
        c.year = 2000;

        Console.WriteLine("s: {0}/{1}/{2}", s.year, s.month, s.day);
        Console.WriteLine("c: {0}/{1}/{2}", c.year, c.month, c.day);
        Console.WriteLine("sDay: {0}/{1}/{2}", sDay.year, sDay.month, sDay.day);
        Console.WriteLine("cDay: {0}/{1}/{2}", cDay.year, cDay.month, cDay.day);

    }
  }
}
```

클래스 또는 구조체에 포함될 수 있는 멤버의 종류

멤버	설명
필드	필드는 클래스 또는 구조체에서 직접 선언되는 모든 형식의 변수이다. 필드는 기본 제공 형식 또는 다른 클래스의 인스턴스일 수 있다.
상수	상수는 값이 컴파일 시간에 설정되면 변경할 수 없는 필드나 속성이다.
속성	해당 클래스의 필드처럼 액세스되는 클래스의 메소드이다.
메소드	클래스가 수행하는 작업을 정의한다. 입력으로 매개변수를 사용할 수 있으며, 매개변수를 통해 출력 데이터를 반환할 수 있다. 매개변수를 통하지 않고 직접 값을 반환할 수도 있다.
이벤트	이벤트는 버튼 클릭, 성공적인 메소드 완료 등의 사건이 발생했을 때 알림을 다른 객체에 제공한다. 이벤트는 대리자를 사용하여 정의 및 트리거된다.
연산자	오버로드된 연산자는 클래스 멤버로 간주되며 클래스에서 public static 메소드로 정의한다.
인덱서	인덱서를 사용하면 배열과 유사한 방법으로 객체를 인덱싱 할 수 있다.
생성자	생성자는 객체를 처음 만들 때 호출되는 메소드이다. 보통 객체의 데이터를 초기화하는데 사용된다.
소멸자	C#에서 드물게 사용되며, 메모리에서 객체를 제거할 때 런타임 실행 엔진이 호출하는 메소드이다.

```
using System;

namespace A066_FieldsAndConstants
{
  class Product
  {
    public string name;
    public int price;
  }

  class MyMath
  {
    public static double PI = 3.14;
  }

  class MyCalendar
  {
    public const int months = 12;
    public const int weeks = 52;
    public const int days = 365;
    public const double daysPerWeek = (double)days / (double)weeks;
    public const double daysPerMonth = (double)days / (double)months;
  }

  class FieldsAndConstants
  {
    static void Main(string[] args)
    {
      Product p = new Product();
      p.name = "시계";
      p.price = 100000;

      Console.WriteLine("{0} : {1:C}", p.name, p.price);
      Console.WriteLine("원주율 : {0}", MyMath.PI);

      Console.WriteLine("한달은 평균 {0:F3}일", MyCalendar.daysPerMonth);
    }
  }
}
```

어떤 날이 1월 1일부터 며칠째 되는 날인지를 알아보자

```
using System;

namespace A067_InstanceMethod
{
  struct Date
  {
    public int year, month, day;

    public static bool IsLeapYear(int year)
    {
      return year % 4 == 0 && (year % 100 != 0 || year % 400 == 0);
    }

    static int[] days = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 };
    public int DayOfYear()
    {
      return days[month - 1] + day +
        (month > 2 && IsLeapYear(year) ? 1 : 0);
    }
  }
```

```
  class InstanceMethod
  {
    static void Main()
    {
      Date xmas = new Date();

      xmas.year = 2018;
      xmas.month = 12;
      xmas.day = 25;

      Console.WriteLine("xmas: {0}/{1}/{2}는 {3}일째 되는 날입니다",
        xmas.year, xmas.month, xmas.day, xmas.DayOfYear());

      if(Date.IsLeapYear(2018) == true)
        Console.WriteLine("2018년은 윤년입니다");
      else
        Console.WriteLine("2018년은 평년입니다");
    }
  }
}
```

static 키워드를 갖는 메소드를 스태틱 메소드 혹은 클래스 메소드라고 한다. static이 아닌 메소드를 인스턴스 메소드라고 한다. 스태틱 메소드는 객체를 만들지 않고 클래스 명.메소드()의 형태로 사용할 수 있다. stati 키워드가 없는 인스턴스 메소드라면 인스턴스를 만들고 인스턴스의 멤버 메소드로 호출해야 한다.

```
using System;

namespace A068_Constructor
{
  class Date
  {
    private int year, month, day;

    public Date()
    {
      year = 1;
      month = 1;
      day = 1;
    }

    public Date(int y, int m, int d)
    {
      year = y;
      month = m;
      day = d;
    }

    public void PrintDate()
    {
      Console.WriteLine("{0}/{1}/{2}", year, month, day);
    }
  }
  class Program
  {
    static void Main(string[] args)
    {
      Date birthday = new Date(2000, 11, 22);
      Date christmas = new Date(2018, 12, 25);
      Date theDay = new Date();

      birthday.PrintDate();
      christmas.PrintDate();
      theDay.PrintDate();
    }
  }
}
```

객체가 만들어지면서 수행해야 하는 작업을 생성자라고 하는 특별한 메소드에 코딩한다. 생성자는 말 그대로 객체가 생성될 때 처리되는 메소드라는 뜻이다. 생성자 메소드의 이름은 클래스의 이름과 같고 리턴 값이 없으며 중복해서 정의할 수 있다.

구조체도 생성자를 가질 수 있다. 다만 구조체에서는 매개변수가 없는 생생성자는 사용할 수 없다. 구조체에서는 자동으로 필드의 값을 0 또는 null로 만들어 주기 때문에 매개변수 없는 명시적 생성자를 포함할 수 없다.

```
using System;

namespace A069_Property
{
  class Rectangle
  {
    private double width;
    private double height;

    public double Area()
    {
      return width*height;
    }

    //public Rectangle(double width, double height) // 생성자
    //{
    //  this.width = width;  this.height = height;
    //}

    // Getter
    public double GetWidth()
    {
      return width;
    }

    public double GetHeight()
    {
      return height;
    }
```

필드를 public으로 선언하면 원치 않게 클래스 외부에서 값을 바꿀 수 있기 때문에 캡슐화 원칙에 위배된다. 따라서 일반적으로 private로 선언하는데, 이러면 외부에서 접근할 수 없으므로 public 메소드를 사용해서 값을 가져오거나 바꿀 수 있게 한다.

속성(Property)은 겟터와 셋터를 손쉽게 만들 수 있게 하는 방법이다.

VS에서 prop또는 propfull을 입력하고 Tab를 두 번 누르면 자동으로 코드가 입력된다.

```
    // Setter
    public void SetWidth(double width)
    {
      this.width = width;
    }

    public void SetHeight(double height)
    {
      this.height = height;
    }
  }
class RectWithProp
  {
    //private double width;
    //private double height;

    //public RectWithProp(double width, double height) // 생성자
    //{
    //  this.width = width;
    //  this.height = height;
    //}

    public double Area()
    {
      return Width * Height;
    }
    public double Width { get; set; } // Width 속성
    public double Height { get; set; } // Height 속성
}


class RectWithPropFull
{
  private double width;

  public double Width
  {
    get { return width; }
    set { if(value > 0 ) width = value; }
  }

  private double height;
  public double Height
  {
    get { return height; }
    set { if (value >= 0) height = value; }
  }
}
```

```
using System;

namespace A070_StaticMethod
{
  class Methods
  {
    // using instance method
    /*
    static void Main(string[] args)
    {
      int a = 10, b = 30, c = 20;
      Methods x = new Methods();
      Console.WriteLine("가장 큰 수는{0}", x.Larger(x.Larger(a, b), c));
    }

    private int Larger(int a, int b)  // static이 아닙니다.
    {
      return (a >= b) ? a : b;
    }
    */
```

세 개의 숫자 중 가장 큰 수를 찾는 정적 메소드

```
    // using static method
    static void Main(string[] args)
    {
      int a = 10, b = 30, c = 20;
      Console.WriteLine("가장 큰 수는{0}", Larger(Larger(a, b), c));
    }

    private static int Larger(int a, int b)
    {
      return (a >= b) ? a : b;
    }
  }
}
```

클래스, 구조체에 포함되어 있는 " 함수"를 메소드라고 부른다. C#은 클래스 바깥에 선언되는 함수를 허용하지 않기 때문에 모든 함수는 메소드가 된다.

메소드는 클래스 또는 구조체에서 public 또는 private 등의 액세스 수준, 자료형, 메소드 이름 및 메소드 매개변수를 지정하여 선언한다. 메소드 선언에서 매개변수가 한 개 이상 있을 때는 괄호 안에 매개변수를 쉼표로 구분하여 표시한다. 빈 괄호는 메소드에 매개변수가 필요하지 않음을 나타낸다. 일반적으로 C#에서 메소드는 대문자로 시작한다.

메소드는 호출한 곳에 값을 반환할 수 있다. 메소드 이름 앞에 나열된 자료형이 void가 아니면 메소드는 return 키워드를 사용하여 값을 반환할 수 있다. void란 return 값이 없는 메소드란 뜻이다.

객체의 메소드 호출은 필드와 같이 객체 이름 뒤에 마침표, 메소드 이름 및 괄호를 추가한다. 매개변수는 괄호 안에 나열되고 쉼표로 구분한다.

메소드를 호출할 때 객체를 만들지 않으려면 static으로 만들어야 한다 지금까지 작성한 모든 프로그램에는 stati void Main(string[] args)라는 메소드가 반드시 하나 있었다.

Main()은 특수한 메소드로써 모든 C# 응용 프로그램의 진입점이므로 반드시 있어야 하며 유일해야 하고 static으로 선언되어야 한다.

소수인지를 알아내는 정적 메소드

```
using System;

namespace A071_IsPrime
{
  class Program
  {
    static void Main(string[] args)
    {
      // 2~100까지의 소수를 찾는 프로그램
      int count = 0;
      for (int i = 2; i <= 100; i++)
        if (IsPrime(i))
        {
          Console.Write("{0} ", i);
          count++;
        }
      Console.WriteLine("₩n2~100까지 소수는 모두 {0}개 있습니다.", count);
    }

    private static bool IsPrime(int x)
    {
      for (int i = 2; i < x; i++)
      {
        if (x % i == 0)
          return false;
      }
      return true;
    }
  }
}
```

윤년인지 알아내는 정적 메소드

```
using System;

namespace A072_IsLeapYear
{
  class Program
  {
    static void Main(string[] args)
    {
      // 2001~2100년 사이의 윤년을 찾는 프로그램
      for (int year = 2001; year <= 2100; year++)
        if (IsLeapYear(year))
          Console.Write("{0} ", year);
      Console.WriteLine();
    }

    private static bool IsLeapYear(int year)
    {
      return year % 4 == 0 && (year % 100 != 0 || year % 400 == 0);
    }
  }
}
```

생애 계산하기

```
using System;

namespace A073_AgeCalculator
{
  class AgeCalculator
  {
    static void Main(string[] args)
    {
       Console.Write("생일을 입력하세요(yyyy/mm/dd) : ");
      string birth = Console.ReadLine();
      string[] bArr = birth.Split('/');
      int bYear = int.Parse(bArr[0]);
      int bMonth = int.Parse(bArr[1]);
      int bDay = int.Parse(bArr[2]);

      int tYear = DateTime.Today.Year;
      int tMonth = DateTime.Today.Month;
      int tDay = DateTime.Today.Day;

      int totalDays = 0;
      for (int year = bYear + 1; year < tYear; year++)
      {
        if (IsLeapYear(year))
          totalDays += 366;
        else
          totalDays += 365;
      }
```

```
 // 올해의 1월 1일부터 오늘까지의 날짜 수
      totalDays += DayOfYear(tYear, tMonth, tDay);

      // 생년의 생일부터 마지막날까지의 날짜 수 = 365(366) - DayOfYear(bYear, bMonth, bDay)
      int yearDays = IsLeapYear(bYear) ? 366 : 365;
      totalDays += yearDays - DayOfYear(bYear, bMonth, bDay);

      Console.WriteLine("total days from birth day : {0}일", totalDays);
    }

    static int[] days = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 }; // 평년

    public static int DayOfYear(int year, int month, int day)
    {
      return days[month - 1] + day +
        (month > 2 && IsLeapYear(year) ? 1 : 0);
    }

    private static bool IsLeapYear(int year)
    {
      return year % 4 == 0 && (year % 100 != 0 || year % 400 == 0);
    }
  }
}
```

피라미드 메소드

```
using System;

namespace A074_PyramidMethod
{
  class Program
  {
    static void Main(string[] args)
    {
      DrawPyramid(3);
      DrawPyramid(5);
      DrawPyramid(7);
    }

    static void DrawPyramid(int n)
    {
      for (int i = 1; i <= n; i++)
      {
        for (int j = i; j < n; j++)
          Console.Write(" ");
        for (int k = 1; k <= 2 * i - 1; k++)
          Console.Write("*");
        Console.WriteLine();
      }
    }
  }
}
```

```
using System;
namespace A075_Factorial
{
  class Program
  {
    static void Main(string[] args)
    {
      int sum = 0;
      for (int i = 1; i <= 10; i++)
      {
        sum += Factorial(i);
        Console.WriteLine("{0,2}! : {1,10:N0}", i, Factorial(i));
      }
      Console.WriteLine("1!~10!의합= {0,8:N0}", sum);
    }

    private static int Factorial(int n)
    {
      int fact = 1;
      for (int i = 1; i <= n; i++)
        fact *= i;
      return fact;
    }
  }
}
```

팩토리얼을 계산하는 메소드

두 숫자 사이의 모든 정수 값을 더하는 메소드

```
using System;

namespace A076_Add
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("  1~100까지의 합: {0,8}", Add(1, 100));
      Console.WriteLine("101~200까지의 합: {0,8}", Add(101, 200));
    }

    private static int Add(int v1, int v2)
    {
      int sum = 0;
      for (int i = v1; i <= v2; i++)
        sum += i;
      return sum;
    }
  }
}
```

```
using System;

namespace A077_Power
{
  class Program
  {
    static void Main(string[] args)
    {
      for (int i = 0; i <= 20; i++)
        Console.WriteLine("2 ^ {0,2} = {1, 7}", i, Power(2, i));
    }

    private static int Power(int v, int i)
    {
      int p = 1;
      for (int i = 1; i <= m; i++)
        p *= n;
      return p;
    }
  }
}
```

n의 m승을 계산하는 메소드

원의 면적을 계산하는 메소드

```
using System;

namespace A078_AreaOfCircle
{
  class Program
  {
    static void Main(string[] args)
    {
      for (double r = 1; r <= 10; r++)
        Console.WriteLine("Area of circle with radius {0,2} = {1,7:F2}",
          r, AreaOfCircle(r));
    }

    private static double AreaOfCircle(double r)
    {
      return Math.PI * r * r;
    }
  }
}
```

```
using System;

namespace A079_RecursivePower
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Power(x,y)를 계산합니다.");
      Console.Write(" x를 입력하세요: ");
      double x = double.Parse(Console.ReadLine());
      Console.Write(" y를 입력하세요: ");
      double y = double.Parse(Console.ReadLine());
      Console.WriteLine(" {0}^{1} = {2}", x, y, Power(x, y));
    }

    private static double Power(double x, double y)
    {
      if (y == 0)
        return 1;
      else
        return x * Power(x, y - 1);
    }
  }
}
```

재귀 메소드 Power(x, y)

재귀메소드로 택토리얼 계산

```
using System;

namespace A080_RecursiveFactorial
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("m!을 계산합니다. m를 입력하세요: ");
      double m = double.Parse(Console.ReadLine());
      Console.WriteLine("{0}! = {1}", m, Fact(m));
    }

    private static double Fact(double x)
    {
      // 1! = 1, n! = n*(n-1)!
      if (x == 1)
        return 1;
      else
        return x * Fact(x - 1);
    }
  }
}
```

재귀메소드로 역수의 합 계산

```
using System;

namespace A081_RecursiveSumOfReciprocal
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("1~n 까지의 역수의 합을 구합니다. n을 입력하세요: ");
      int n = int.Parse(Console.ReadLine());
      Console.WriteLine("1~{0}까지의 역수의 합: {1}", n, SumOfReci(n));
    }

    private static double SumOfReci(int n)
    {
      // n==1 return 1, else return 1/n + Sum(n-1)
      if (n == 1)
        return 1;
      else
        return 1.0 / n + SumOfReci(n - 1);
    }
  }
}
```

하노이 탑과 메르센 수

```
using System;

namespace A082_HanoiTower
{
  class Program
  {
    static void Main(string[] args)
    {
      // 메르센 수 2^n - 1
      for (int i = 1; i <= 70; i++)
      {
        double m = Mersenne(i);
        Console.WriteLine("메르센 수({0}) = {1:N0} = {2:N1}일 = {3:N1}년", i, m, m / 3600 / 24, m / 3600 / 24 / 365);
      }

      //   하노이탑 문제...
      Console.WriteLine("₩nHanoi Tower: {0}, {1}->{2}->{3}", 4, 'A', 'B', 'C');
      Hanoi(4, 'A', 'C', 'B');
    }

    private static double Mersenne(int n)
    {
      return Math.Pow(2, n) - 1;
    }

    private static void Hanoi(int n, char from, char to, char by)
    {
      if (n == 1)
        Console.WriteLine("Move : {0} -> {1}", from, to);
      else
      {
        Hanoi(n - 1, from, by, to);
        Console.WriteLine("Move : {0} -> {1}", from, to);
        Hanoi(n - 1, by, to, from);
      }
    }
  }
}
```

재귀 이진 탐색

```
using System;

namespace A083_RecursiveBinarySearch
{
  class Program
  {
    static void Main(string[] args)
    {
      Random r = new Random();
      int[] v = new int[30];

      for (int i = 0; i < 30; i++)
        v[i] = r.Next(1000);
      PrintArray("정렬 전", v);

      Array.Sort(v);  // 정렬, 이진탐색은 배열이 정렬되어야 합니다

      PrintArray("정렬 후", v);

      Console.Write("=> 검색할 숫자를 입력하세요: ");
      int key = int.Parse(Console.ReadLine());

      int index = RecBinSearch(v, 0, v.Length - 1, key);
      if (index == -1)
        Console.WriteLine("찾는 값이 배열에 없습니다.");
      else
        Console.WriteLine("v[{0}] = {1}", index, key);
    }
```

```
    private static int RecBinSearch(int[] v, int low, int high, int key)
    {
      if (low <= high)
      {
        int mid = (low + high) / 2;
        if (key == v[mid])
          return mid;
        else if (key > v[mid])
          return RecBinSearch(v, mid + 1, high, key);
        else
          return RecBinSearch(v, low, mid - 1, key);
      }

      return -1;
    }

    private static void PrintArray(string s, int[] v)
    {
      Console.WriteLine(s);
      for (int i = 0; i < v.Length; i++)
        Console.Write("{0,5}{1}", v[i], (i % 10 == 9) ? "₩n" : "");
    }
  }
}
```

Stopwatch로 피보나치 수열의 실행 시간 측정

```
using System;

namespace A084_ExecutionTime
{
  class Program
  {
    static int[] f = new int[50];

    static void Main(string[] args)
    {
      Console.Write("피보나치 수열의 n항까지를 구합니다. n를 입력하세요: ");
      int n = int.Parse(Console.ReadLine());

      var watch = System.Diagnostics.Stopwatch.StartNew();
      f[1] = f[2] = 1;
      for (int i = 3; i <= n; i++)
        f[i] = f[i - 1] + f[i - 2];
      for (int i = 1; i <= n; i++)
        Console.Write("{0} ", f[i]);
      Console.WriteLine();
      watch.Stop();
      var elapsedMs = watch.ElapsedMilliseconds;
      Console.WriteLine("실행시간은 {0}ms₩n", elapsedMs);
```

```
      watch = System.Diagnostics.Stopwatch.StartNew();
      for (int i = 1; i <= n; i++)
        Console.Write("{0} ", FiboRecursive(i));
      Console.WriteLine();
      watch.Stop();
      elapsedMs = watch.ElapsedMilliseconds;
      Console.WriteLine("실행시간은 {0}ms", elapsedMs);
    }

    private static int FiboRecursive(int n)
    {
      if (n == 1 || n == 2)
        return 1;
      else
        return FiboRecursive(n - 1) + FiboRecursive(n - 2);
    }
  }
}
```

DateTime 구조체

```
using System;

namespace A085_DateTime
{
  class Program
  {
    static void Main(string[] args)
    {
      DateTime date1 = new DateTime(1992, 7, 4, 8, 44, 0);
      DateTime date2 = new DateTime(1990, 1, 27, 12, 6, 0);
      Console.WriteLine(date1);
      Console.WriteLine(date2);
      Console.WriteLine("{0}과 {1}의 차이는 {2}일입니다",
        date1.ToString("yyyy년 M월 d일"),
        date2.ToString("yyyy년 M월 d일"),
        date1.Subtract(date2).Days);

      Console.WriteLine("₩n오늘: {0}", DateTime.Today);

      DateTime y = DateTime.Today.AddDays(-1);  // 어제
      Console.WriteLine("어제: {0}", y.ToShortDateString());

      DateTime t  = DateTime.Today.AddDays(1);  // 내일
      Console.WriteLine("내일: {0}", t.ToShortDateString());

      Console.WriteLine("₩n2020년은 {0}입니다",
DateTime.IsLeapYear(2020) ? "윤년" : "평년");
      Console.WriteLine("2020년 2월은 {0}일입니다.₩n",
DateTime.DaysInMonth(2020, 2));
```

```
        // Parse and TryParse
        string date = "1990-1-27 12:6";
        DateTime aDay = DateTime.Parse(date);
        Console.WriteLine(aDay);

        string input = "1992/7/4 8:44";
        DateTime bDay;
        if (DateTime.TryParse(input, out bDay))
        {
          Console.WriteLine(bDay);
        }
        Console.WriteLine();

        DateTime d1 = DateTime.Now;
        DateTime d2 = DateTime.UtcNow;

        Console.WriteLine(d1);
        Console.WriteLine(d2);
    }
  }
}
```

TimeSpan

```
using System;

namespace A086_TimeSpan
{
  class Program
  {
    static void Main(string[] args)
    {
      // TimeSpan
      DateTime christmas = new DateTime(2018, 12, 25);
      DateTime newYearsDay = new DateTime(2019, 1, 1);
      //TimeSpan span = newYearsDay.Subtract(christmas);
      TimeSpan span = newYearsDay - christmas;
      Console.WriteLine("크리스마스와 1월 1일의 시간 간격");
      Console.WriteLine("{0,14}", span);
      Console.WriteLine("{0,14} days", span.Days);
      Console.WriteLine("{0,14} hours", span.Hours);
      Console.WriteLine("{0,14} minutes", span.Minutes);
      Console.WriteLine("{0,14} seconds", span.Seconds);
      Console.WriteLine("{0,14} milliseconds", span.Milliseconds);
      //Console.WriteLine("{0,14} ticks", span.Ticks);
```

```
Console.WriteLine("₩n또는");

Console.WriteLine("{0,14}", span);
Console.WriteLine("{0,14} days", span.TotalDays);
Console.WriteLine("{0,14} hours", span.TotalHours);
Console.WriteLine("{0,14} minutes", span.TotalMinutes);
Console.WriteLine("{0,14} seconds", span.TotalSeconds);
Console.WriteLine("{0,14} milliseconds", span.TotalMilliseconds);
Console.WriteLine("{0,14} ticks", span.Ticks);

// 생애 계산기
Console.Write("생년월일 시분초를 입력하세요: ");
DateTime date1 = DateTime.Parse(Console.ReadLine());
DateTime date2 = DateTime.Now;
// Calculate the interval between the two dates.
TimeSpan interval = date2 - date1;
Console.WriteLine("탄생 시간: {0}", date1);
Console.WriteLine("현재 시간: {0}", date2);
Console.WriteLine("생존 시간: {0}", interval.ToString());
```

```
// Display individual properties of the resulting TimeSpan object.
      Console.WriteLine("당신은 지금 이 순간까지 {0}일 {1}시간"
        + " {2}분 {3}초를 살았습니다.",
        interval.Days, interval.Hours,
        interval.Minutes, interval.Seconds);
      Console.WriteLine("   {0,-35} {1,20}", "Value of Days Component:", interval.Days);
      Console.WriteLine("   {0,-35} {1,20}", "Total Number of Days:", interval.TotalDays);
      Console.WriteLine("   {0,-35} {1,20}", "Value of Hours Component:", interval.Hours);
      Console.WriteLine("   {0,-35} {1,20}", "Total Number of Hours:", interval.TotalHours);
      Console.WriteLine("   {0,-35} {1,20}", "Value of Minutes Component:", interval.Minutes);
      Console.WriteLine("   {0,-35} {1,20}", "Total Number of Minutes:", interval.TotalMinutes);
      Console.WriteLine("   {0,-35} {1,20:N0}", "Value of Seconds Component:", interval.Seconds);
      Console.WriteLine("   {0,-35} {1,20:N0}", "Total Number of Seconds:", interval.TotalSeconds);
      Console.WriteLine("   {0,-35} {1,20:N0}", "Value of Milliseconds Component:", interval.Milliseconds);
      Console.WriteLine("   {0,-35} {1,20:N0}", "Total Number of Milliseconds:", interval.TotalMilliseconds);
      Console.WriteLine("   {0,-35} {1,20:N0}", "Ticks:", interval.Ticks);
    }
  }
}
```

TimeSpan을 이용한 생애계산기

```
using System;

namespace A087_LifeTimeCalc
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("생년월일 시분을 입력하세요(0000-00-00 00:00): ");
      DateTime date1 = DateTime.Parse(Console.ReadLine());
      DateTime date2 = DateTime.Now;

      TimeSpan interval = date2 - date1;
      Console.WriteLine("탄생 시간: {0}", date1);
      Console.WriteLine("현재 시간: {0}", date2);
      Console.WriteLine("생존 시간: {0}", interval.ToString());
      Console.WriteLine("당신은 지금 이 순간까지 {0}일 {1}시간"
        + " {2}분 {3}초를 살았습니다.",
        interval.Days, interval.Hours,
        interval.Minutes, interval.Seconds);
    }
  }
}
```

DateTime Format

```
using System;
using System.Globalization;

namespace A088_DateTimeFormat
{
  class Program
  {
    static void Main(string[] args)
    {
      DateTime today = DateTime.Now;

      Console.WriteLine(today.ToString("yyyy년 MM월 dd일"));
      Console.WriteLine(string.Format("{0:yyyy년 MM월 dd일}", today));
      Console.WriteLine(today.ToString("MMMM dd, yyyy ddd", CultureInfo.CreateSpecificCulture("en-US")));
      Console.WriteLine(today.ToString("MMMM dd, yyyy ddd", new CultureInfo("en-US")));

      // 오전, 오후
      Console.WriteLine(today.ToString("tt"));
      today = today.AddHours(12);
      Console.WriteLine(today.ToString("tt"));
      today = today.AddHours(-12);

      Console.WriteLine("₩n표준 포맷 지정자");

      // d : 축약된 날짜 형식
      Console.WriteLine("d : " + today.ToString("d"));
      Console.WriteLine("d : " + today.ToString("d", new CultureInfo("en-US")));
```

메소드에 인수를 전달하는 세 가지 방법

```
using System;

namespace A089_MethodArguments
{
  class Program
  {
    static void Main(string[] args)
    {
      int a = 3;
      Sqr(a);
      Console.WriteLine("Value: {0}", a); // 3이 출력됩니다.

      int b = 3;
      Sqr(ref b);
      Console.WriteLine("ref: {0}", b); // 9가 출력됩니다.

      string name;
      int id;
      GetName(out name, out id);
      Console.WriteLine("out: {0} {1}", name, id);
    }
```

```
    static void Sqr(int x)
    {
      x = x * x;
    }
    static void Sqr(ref int x)
    {
      x = x * x;
    }
    static void GetName(out string name, out int id)
    {
      Console.Write("Enter Name: ");
      name = Console.ReadLine();
      Console.Write("Enter Id: ");
      id = int.Parse(Console.ReadLine());
    }
  }
}
```

```
using System;

namespace A090_params
{
  class Program
  {
    public static void PrintIntParams(params int[] arr)
    {
      for (int i = 0; i < arr.Length; i++)
      {
        Console.Write(arr[i] + " ");
      }
      Console.WriteLine();
    }

    public static void PrintObjectParams(params object[] arr)
    {
      for (int i = 0; i < arr.Length; i++)
      {
        Console.Write(arr[i] + " ");
      }
      Console.WriteLine();
    }
```

가변 길이 매개변수 params의 사용 방법

```
    static void Main(string[] args)
    {
      PrintIntParams(1, 2, 3, 4);
      PrintObjectParams(1, 1.234, 'a', "test");
      PrintObjectParams();

      int[] myIntArray = { 5, 6, 7, 8, 9 };
      PrintIntParams(myIntArray);

      object[] myObjArray = { 2, 2.345, 'b', "test", "again" };
      PrintObjectParams(myObjArray);

      PrintObjectParams(myIntArray);
    }
  }
}
```

선택적 인수와 명명된 인수

```
using System;

namespace A091_OptionalNamedArguments
{
  class Program
  {
    static int MyPower(int x, int y = 2)
    {
      int result = 1;
      for (int i = 0; i < y; i++)
        result *= x;
      return result;
    }

    static int Area(int h, int w)
    {
      return h * w;
    }
    static void Main(string[] args)
    {
      Console.WriteLine(MyPower(4, 2));
      Console.WriteLine(MyPower(4));
      Console.WriteLine(MyPower(3, 4));

      Console.WriteLine(Area(w: 5, h: 6));
      Console.WriteLine(Area(h: 6, w: 5));
    }
  }
}
```

```
using System;

namespace A092_MethodOverloading
{
  class Program
  {
    static void Main(string[] args)
    {
      Print(10);
      Print(0.123);
      Print("Sum = ", 123.4);
    }

    private static void Print(double x)
    {
      Console.WriteLine(x);
    }

    private static void Print(string s, double x)
    {
      Console.WriteLine(s + x);
    }

    private static void Print(int x)
    {
      Console.WriteLine(x);
    }
  }
}
```

메소드 오버로딩

Reference

- ✓ C# 프로그래밍 입문, 오세만 외4, 생능출판
- ✓ 초보자를 위한 C# 200제, 강병익, 정보문화사
- ✓ 프랙티컬 C#, 이데이 히데유키, 김범준, 위키북스
- ✓ C#언어 프로그래밍 바이블, 김명렬 외1, 홍릉과학출판사
- ✓ C# and the .NET Platform, Andrew Troelsen, 장시혁, 사이텍미디어