객체지향 프로그래밍

C# - 예제 프로그램

KYUNGSUNG UNIVERSITY SINCE 1955

```
class Hello
{
    static void Main()
    {
        Console.writeLine("Hello C# world");
    }
}
```

```
using System;

namespace A003_Console
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("Hello ");
      Console.WriteLine("World!");
      Console.Write("이름을 입력하세요: ");

      string name = Console.ReadLine();
      Console.Write("안녕하세요, ");
      Console.Write(name);
      Console.WriteLine("님!");
    }
  }
}
```

```
using System;

namespace A004_Variable
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("이름을 입력하세요: ");
      string name = Console.ReadLine();
      Console.Write("나이를 입력하세요: ");
      int age = int.Parse(Console.ReadLine());
      Console.Write("카를 입력하세요(cm): ");
      float height = float.Parse(Console.ReadLine());
      Console.Write("안녕하세요, ");
      Console.Write(name);
      Console.WriteLine("님!");

      Console.Write("나이는 ");
      Console.Write(age);
      Console.Write("세, 키는 ");
      Console.Write(height);
      Console.WriteLine("cm 이군요!");
    }
  }
}
```

```
using System;

namespace A005_string
{
  class Program
  {
    static void Main(string[] args)
    {
      string a = "hello";
      string b = "h";

      b += "ello";
      Console.WriteLine(a == b);
      Console.WriteLine("b = " + b);

      int x = 10;
      string c = b + '!' + " " + x;
      Console.WriteLine("c = " + c);
    }
  }
}
```

```
using System;

namespace A006_Assignment
{
  class Program
  {
    static void Main(string[] args)
    {
      int i;
      double x;

      i = 5;
      x = 3.141592;
      Console.WriteLine("i = " + i + ", x = " + x);

      x = i; // 암시적 형변환 from int to double
      i = (int)x; // 캐스트가 필요함
      Console.WriteLine("i = " + i + ", x = " + x);
    }
  }
}
```

```
using System;

namespace A007_ConsoleWriteLine
{
  class Program
  {
    static void Main(string[] args)
    {
      bool b = true;
      char c = 'A';
      decimal d = 1.234m;      // m은 decimal 형의 접미사
      double dd = 1.23456789;
      float f = 1.23456789f;  // f는 float 형의 접미사
      int i = 1234;
      string s = "Hello";
```

```
      Console.WriteLine(b);
      Console.WriteLine(c);
      Console.WriteLine(d);
      Console.WriteLine(dd);
      Console.WriteLine(f);
      Console.WriteLine(i);
      Console.WriteLine(s);
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace A008_ConsoleWriteMulti
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("10 이하의 소수 : {0}, {1}, {2}, {3}", 2, 3, 5, 7);

      string primes;
      primes = String.Format("10 이하의 소수 : {0}, {1}, {2}, {3}", 2, 3, 5, 7);
      Console.WriteLine(primes);
    }
  }
}
```

```
using System;

namespace A009_VariablesAndWrite
{
  class Program
  {
    static void Main(string[] args)
    {
      int v1 = 100;
      double v2 = 1.234;

      //Console.WriteLine(v1, v2);  // 에러가 발생합니다.
      Console.WriteLine(v1.ToString() + ", " + v2.ToString()  );
      Console.WriteLine("v1 = " + v1 + ", v2 = " + v2);
      Console.WriteLine("v1 = {0}, v2 = {1}", v1, v2);
      Console.WriteLine($"v1 = {v1}, v2 = {v2}");
    }
  }
}
```

C : 통화
D : 십진수(정수)
E : 지수(과학)
G : 일반
N : 천 단위 구분 기호 숫자
P : 퍼센트
R : 라운드트립
X : 16진수

```
using System;

namespace CurrencyFormat
{
  class Program
  {
    static void Main(string[] args)
    {
      decimal value = 123456.789m;
      Console.WriteLine("잔액은 {0:C2}원 입니다.", value);
      Console.WriteLine("{0,20:C2}원 입니다.", value);
      Console.WriteLine("{0,-20:C2}원 입니다.", value);
      Console.WriteLine("12345678901234567890123456789O");
    }
  }
}
```

형식지정자는 크게 두 가지로 나눌 수 있는데, 하나는 표준 형식 지정자이고 또 다른 하나는 커스텀 형식지정자입니다. 많이 사용하는 숫자 표준 형식지정자는 N(Number), D(Decimal), C(Currency), F(Fixed Point), E(Scientific)입니다.
또 다른 하나는 커스텀 형식지정자입니다.
: Digit placeholder(0이 앞에 붙지 않음)
0 : Zero placeholder(0이 앞에 붙음)
. : 소수점(Decimal Point)
, : 천 자리(Thousands operator)
; : 섹션 구분 기호(Section separator)

```
using System;

namespace A011_FormatSpecifier
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("{0:N2}", 1234.5678);   // 출력: 1,234.57
      Console.WriteLine("{0:D8}", 1234);  // 출력: 00001234
      Console.WriteLine("{0:F3}", 1234.56);     // 출력: 1234.560
      Console.WriteLine("{0,8}", 1234);   // 출력: ____1234
      Console.WriteLine("{0,-8}", 1234);   // 출력: 1234____

      string s;
      s = string.Format("{0:N2}", 1234.5678);
      Console.WriteLine(s);
      s = string.Format("{0:D8}", 1234);
      Console.WriteLine(s);
      s = string.Format("{0:F3}", 1234.56);
      Console.WriteLine(s);
```

```
        Console.WriteLine(1234.5678.ToString("N2"));
        Console.WriteLine(1234.ToString("D8"));
        Console.WriteLine(1234.56.ToString("F3"));

        Console.WriteLine("{0:#.##}", 1234.5678);
        Console.WriteLine("{0:0,0.00}", 1234.5678);
        Console.WriteLine("{0:#,#.##}", 1234.5678);
        Console.WriteLine("{0:000000.00}", 1234.5678);

        Console.WriteLine("{0:#,#.##;(#,#.##);zero}", 1234.567);
        Console.WriteLine("{0:#,#.##;(#,#.##);zero}", -1234.567);
        Console.WriteLine("{0:#,#.##;(#,#.##);zero}", 0);
    }
  }
}
```

```
using System;

namespace A012_FloatDoubleDecimal
{
  class Program
  {
    static void Main(string[] args)
    {
      float flt = 1F / 3;
      double dbl = 1D / 3;
      decimal dcm = 1M / 3;
      Console.WriteLine("float: {0}₩ndouble: {1}₩ndecimal: {2}", flt, dbl, dcm);
      Console.WriteLine("float: {0} bytes₩ndouble: {1} bytes₩ndecimal: {2} bytes",
        sizeof(float), sizeof(double), sizeof(decimal));
      Console.WriteLine("float : {0}~{1}", float.MinValue, float.MaxValue);
      Console.WriteLine("double : {0}~{1}", double.MinValue, double.MaxValue);
      Console.WriteLine("decimal : {0}~{1}", decimal.MinValue, decimal.MaxValue);
```

```
        Console.WriteLine("float : {0}", float.MaxValue+1);
        Console.WriteLine("double : {0}", double.MaxValue + 1);
        //Console.WriteLine("decimal : {0}", decimal.MaxValue + 1);

    }
  }
}
```

```
using System;

namespace A013_TypeConversion
{
  class Program
  {
    static void Main(string[] args)
    {
      int num = 2147483647;
      long bigNum = num;  // 암시적 형변환
      Console.WriteLine(bigNum);

      double x = 1234.7;
      int a;

      a = (int)x;        // Cast double to int.
      Console.WriteLine(a);
    }
  }
}
```

```
using System;

namespace A014_StringToNumber
{
  class Program
  {
    static void Main(string[] args)
    {
      //int num = Int32.Parse("-15");
      //Console.WriteLine(num);

      //int i;
      //if (int.TryParse("-1004", out i))
      //  Console.WriteLine(i);
      //else
      //  Console.WriteLine("숫자로 바꿀 수 없습니다.");

      string input;
      int value;

      Console.Write("1. int로 변환할 문자열을 입력하세요: ");
      input = Console.ReadLine();
      bool result = Int32.TryParse(input, out value);
```

문자열을 숫자로 바꾸는 방법은 두 가지가 있습니다. 하나는 숫자 형식(int, float, double등)에 있는 Parse()나 TryParse()메소드를 사용하는 것이고, 또 다른 하나는 Convert클래스의 메소드를 사용하는 것입니다.

Parse()와 TryParse() 두 메소드는 모두 문자열의 앞뒤에 있는 공백은 무시합니다. 다른 모든 문자들은 int, float, decimal 등의 숫자형에 맞는 문자들이어야 합니다.
문자열 중간에 공백이 있으면 에러가 발생합니다.

```
        if (!result)
          Console.WriteLine("'{0}'는 int로 변환될 수 없습니다.₩n", input);
        else
          Console.WriteLine("int '{0}'으로 변환되었습니다.₩n", value);

        Console.Write("2. double로 변환할 문자열을 입력하세요: ");
        input = Console.ReadLine();
        try
        {
          double m = Double.Parse(input);
          Console.WriteLine("double '{0}'으로 변환되었습니다.₩n", m);
        }
        catch (FormatException e)
        {
          Console.WriteLine(e.Message);
        }

        // Convert

    }
  }
}
```

Console.ReadLine()으로 입력받은 데이터는 모두 string입니다. 입력받은 데이터를 숫자로 변환할 때는 Convert클래스를 사용합니다.
Convert클래스는 ToInt32(), ToSingle(), ToDouble()과 같이 string을 숫자형으로 바꾸는 여러가지 메소드를 제공합니다.
Conver에는 string을 숫자로 바꾸는 메소드뿐 아니라 ToString(), ToByte(), ToBoolean(), ToChar(), ToDateTime()등과 같이 여러 자료형 사이의 변환도 지원합니다.

```
using System;

namespace A015_Convert
{
  class Program
  {
    static void Main(string[] args)
    {
      int x, y;

      Console.Write("첫번째 숫자를 입력하세요: ");
      x = Convert.ToInt32(Console.ReadLine());
      Console.Write("두번째 숫자를 입력하세요: ");
      y = Convert.ToInt32(Console.ReadLine());
      Console.WriteLine("{0} + {1} = {2}", x, y, x + y);

      // 2진수, 8진수, 10진수, 16진수로 출력하기
      short value = short.MaxValue; // Int16.MaxValue

      Console.WriteLine("₩n2진수, 8진수, 10진수, 16진수로 출력하기");
```

```
//for(int baseNum = 2; baseNum <= 16; baseNum*=4)
//{
//  s = Convert.ToString(value, baseNum);
//  int i = Convert.ToInt32(s, baseNum);
//  Console.WriteLine("value = {0}, {1,2}진수 = {2,16}, i = {3}", value, baseNum, s, i);
//}

int baseNum = 2;
string s = Convert.ToString(value, baseNum); // value를 string으로 변환(2진수로)
int i = Convert.ToInt32(s, baseNum);  // string으로 변환된 값을 2진수로 읽어서 int로 다시 변환
Console.WriteLine("i = {0}, {1,2}진수 = {2,16}", i, baseNum, s);

baseNum = 8;
s = Convert.ToString(value, baseNum);
i = Convert.ToInt32(s, baseNum);
Console.WriteLine("i = {0}, {1,2}진수 = {2,16}", i, baseNum, s);
```

```
baseNum = 10;
s = Convert.ToString(value, baseNum);
i = Convert.ToInt32(s, baseNum);
Console.WriteLine("i = {0}, {1,2}진수 = {2,16}", i, baseNum, s);

baseNum = 16;
s = Convert.ToString(value, baseNum);
i = Convert.ToInt32(s, baseNum);
Console.WriteLine("i = {0}, {1,2}진수 = {2,16}", i, baseNum, s);

//double dNum = 23.15;

//int iNum = Convert.ToInt32(dNum);
//bool bNum = Convert.ToBoolean(dNum);
//string sNum = Convert.ToString(dNum);
```

```
        Console.WriteLine(@"abc₩nabc");
        Console.WriteLine("123₩n123");

        string a = "₩₩₩u0066₩n";
        Console.WriteLine(a);

        string b = "C:₩₩Windows₩₩";
        Console.WriteLine(b);              // "C:₩Windows₩"를 출력
    }
  }
}
```

```
using System;

namespace A016_Operators
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine(3 + 4 * 5);
      Console.WriteLine((3 + 4) * 5);
      Console.WriteLine(3 * 4 / 5);
      Console.WriteLine(4 / 5 * 3);

      int a = 10, b = 20, c;
      Console.WriteLine(c = a + b);
    }
  }
}
```

```
using System;

namespace A017_ArithmeticOperators
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("정수의 계산");
      Console.WriteLine(123 + 45);
      Console.WriteLine(123 - 45);
      Console.WriteLine(123 * 45);
      Console.WriteLine(123 / 45);
      Console.WriteLine(123 % 45);

      Console.WriteLine("₩n실수의 계산");
      Console.WriteLine(123.45 + 67.89);
      Console.WriteLine(123.45 - 67.89);
      Console.WriteLine(123.45 * 67.89);
      Console.WriteLine(123.45 / 67.89);
      Console.WriteLine(123.45 % 67.89);
    }
  }
}
```

```
using System;

namespace A018_DivideByZero
{
  class Program
  {
    static void Main(string[] args)
    {
      int x = 10, y = 0;

      try
      {
        Console.WriteLine(x / y);
      }
      catch (Exception e)
      {
        Console.WriteLine(e.Message);
      }
    }
  }
}
```

C#에서는 실행 중에 나오는 에러를 예외(Exception)라고 합니다. 산술 연산에서 나올 수 있는 예외는 0으로 나누는 나눔 예외(DivideByZeroException)와 오버플로우 예외(OverflowException)입니다.

```
using System;

namespace A019_Overflow
{
  class Program
  {
    static void Main(string[] args)
    {
      //Console.WriteLine(int.MaxValue);
      //Console.WriteLine(float.MaxValue);
      //Console.WriteLine(double.MaxValue);

      //int x = 10, y = 0;

      //try
      //{
      //Console.WriteLine((double)x / y);
      //Console.WriteLine(x / y);
      //}
      //catch (Exception e)
      //{
      //  Console.WriteLine(e.Message);
      //}
```

```
//Console.Read();

//Console.WriteLine("int.MaxValue = {0}", int.MaxValue);



// y = int.MaxValue + 10;  // 이 문장은 컴파일시에 에러가 나옵니다.

int x = int.MaxValue, y = 0;

checked
{
  try
  {
    y = x + 10;
  }
  catch (Exception e)
  {
    Console.WriteLine(e.Message);
  }
}
```

```
Console.WriteLine("int.MaxVlaue + 10 = {0}", y);
Console.Read();

checked
{
  try
  {
    Console.WriteLine(y = x + 10);
    Console.WriteLine(y = x - 1);
    Console.WriteLine(y = x * 2);
    Console.WriteLine(y = x / 2);
    Console.WriteLine(y = x % 2);
  }
  catch (OverflowException)
  {
    Console.WriteLine("OverflowExcetion 발생 ");
  }
}
```

```
      int value = 780000000;
      checked
      {
        try
        {
          // Square the original value.
          int square = value * value;
          Console.WriteLine("{0} ^ 2 = {1}", value, square);
        }
        catch (OverflowException)
        {
          double square = Math.Pow(value, 2);
          Console.WriteLine("Exception: {0} > {1:E}.",
                            square, Int32.MaxValue);
        }
      }
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace A020_RelationalOperator
{
  class Program
  {
    static void Main(string[] args)
    {
      bool result;
      int first = 10, second = 20;

      result = (first == second);
      Console.WriteLine("{0} == {1} : {2}", first, second, result);

      result = (first > second);
      Console.WriteLine("{0} > {1} : {2}", first, second, result);

      result = (first < second);
      Console.WriteLine("{0} < {1} : {2}", first, second, result);
```

```
        result = (first >= second);
        Console.WriteLine("{0} >= {1} : {2}", first, second, result);

        result = (first <= second);
        Console.WriteLine("{0} <= {1} : {2}", first, second, result);

        result = (first != second);
        Console.WriteLine("{0} != {1} : {2}", first, second, result);
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace A021_LogicalOperators
{
  class Program
  {
    static void Main(string[] args)
    {
      bool result;
      int first = 10, second = 20;

      result = (first == second) || (first > 5);
      Console.WriteLine("{0} || {1} : {2}", first == second, first > 5, result);
```

```
        result = (first == second) && (first > 5);
        Console.WriteLine("{0} && {1} : {2}", first == second, first > 5, result);

        result = true ^ false;
        Console.WriteLine("{0} ^ {1} : {2}", true, false, result);

        result = !(first > second);
        Console.WriteLine("!{0} : {1}", first > second, result);

        int year = 2018;
        bool isLeapYear = year % 4 == 0 && (year % 100 != 0 || year % 400 == 0);
        Console.WriteLine("{0}년은 윤년이다 : {1}", year, isLeapYear);
    }
  }
}
```

```
using System;

namespace A022_BitwiseOperators
{
  class Program
  {
    static void Main(string[] args)
    {
      int x = 14, y = 11, result;

      result = x | y;
      Console.WriteLine("{0} | {1} = {2}", x, y, result);
      result = x & y;
      Console.WriteLine("{0} & {1} = {2}", x, y, result);
      result = x ^ y;
      Console.WriteLine("{0} ^ {1} = {2}", x, y, result);
      result = ~x;
      Console.WriteLine("~{0} = {1}", x, result);
```

```
        result = x << 2;
        Console.WriteLine("{0} << 2 = {1}", x, result);
        result = y >> 1;
        Console.WriteLine("{0} >> 1 = {1}", y, result);

    }
  }
}
```

```
using System;

namespace A023_ConditionalOperator
{
  class Program
  {
    static void Main(string[] args)
    {
      int input = Convert.ToInt32(Console.ReadLine());

      string result = (input > 0) ? "양수입니다." : "음수입니다.";
      Console.WriteLine("{0}는 {1}", input, result);
      Console.WriteLine("{0}는 {1}", input, (input % 2 == 0) ? "짝수입니다." : "홀수입니다.");

      for (int i = 1; i <= 50; i++)
      {
        Console.Write("{0,3}{1}", i, i % 10 != 0 ? "" : "₩n");
      }
    }
  }
}
```

```
using System;

namespace A024_CompoundAssignment
{
  class Program
  {
    static void Main(string[] args)
    {
      int x = 32;

      Console.WriteLine(x += 2);
      Console.WriteLine(x -= 8);
      Console.WriteLine(x *= 3);
      Console.WriteLine(x /= 2);
      Console.WriteLine(x++);
      Console.WriteLine(--x);
    }
  }
}
```

x += y * 3; 이라는 문장은 x = x + (y * 3)이라는 뜻입니다. (x + y) * 3이 아니라는 것에 주목합니다. 이와 같은 압축 표현은 모든 이항연산자에 적용되며 이것을 지정연산자라고도 합니다.

```
using System;

namespace A025_StringMethods
{
  class Program
  {
    static void Main(string[] args)
    {
      string s = " Hello, World! ";
      string t;

      Console.WriteLine(s.Length);
      Console.WriteLine(s[8]);

      Console.WriteLine(s.Insert(8, "C# "));
      Console.WriteLine(s.PadLeft(20, '.'));
      Console.WriteLine(s.PadRight(20, '.'));
      Console.WriteLine(s.Remove(6));
      Console.WriteLine(s.Remove(6, 7));
```

문자열은 프로그램에서 가장 많이 쓰는 자료형 중에 하나입니다. 키워드 string은 .NET의 System.String 클래스와 동일합니다. Int가 Int32의 별명인 것처럼 string은 String의 별명입니다.
C# string은 불변(immutable)입니다. 한 번 문자열이 설정되면 다시 변경할 수 없습니다.
String 클래스의 속성인 Length는 string 객체의 길이입니다. 또한 string은 문자 배열처럼 인덱스로 String의 특정 위치에 있는 문자를 가져올 수 있습니다.
String 클래스의 필드 중에는 Empty가 있습니다. 이 필드는 static readonly이고 값은 길이가 0일 문자열 즉, 빈 문자열입니다. Empty와 null은 다릅니다. Null은 선언되고 할당되지 않은 스트링을 의미하고 Empty도는 “”는 빈 문자열이라는 뜻입니다.

```
Console.WriteLine(s.Replace('l', 'm'));
Console.WriteLine(s.ToLower());
Console.WriteLine(s.ToUpper());
Console.WriteLine('/' + s.Trim() + '/');
Console.WriteLine('/' + s.TrimStart() + '/');
Console.WriteLine('/' + s.TrimEnd() + '/');

string[] a = s.Split(',');
foreach(var i in a)
  Console.WriteLine('/' + i + '/');

char[] destination = new char[10];
s.CopyTo(8, destination, 0, 6);
Console.WriteLine(destination);

Console.WriteLine('/' + s.Substring(8) + '/');
Console.WriteLine('/' + s.Substring(8, 5) + '/');
```

```
        Console.WriteLine(s.Contains("ll"));
        Console.WriteLine(s.IndexOf('o'));
        Console.WriteLine(s.LastIndexOf('o'));
        Console.WriteLine(s.CompareTo("abc"));

        Console.WriteLine(String.Concat("Hi~", s));
        Console.WriteLine(String.Compare("abc", s));
        Console.WriteLine(t = String.Copy(s));

        String[] val = { "apple", "orange", "grape", "pear" };
        String result = String.Join(", ", val);
        Console.WriteLine(result);
    }
  }
}
```

콘솔에서 값을 입력받으려면 string s = Console.ReadLine()을 사용하는데 엔터를 입력할 때까지 입력되는 한 줄을 하나의 스트링으로 저장합니다. 이때 s는 "10 50 60 32 45 77"의 값을 갖는 하나의 스트링입니다. 계산을 위해서는 이 문자열에서 숫자들을 추출해서 더해주어야 하는데 이 때 사용할 메소드가 Split()입니다.

구분하고자 하는 문자는 디폴트로 공백입니다. s.Split(',')와 같이 구분하고자 하는 문자를 Split()메소드에 매개변수로 전달할 수도 있습니다. 또한 Split()메소드에서 구분하고자 하는 문자를 여러 개 지정할 수도 있습니다.

```
using System;

namespace A026_SplitMethod
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("더하고자하는 숫자들을 입력하세요: ");
      string s = Console.ReadLine();
      Console.WriteLine(s);

      int sum = 0;
      char[] delimiters = { ',', ' ', '-' };
      string[] v = s.Split(delimiters);
      foreach (var i in v)
      {
        sum += int.Parse(i);
      }
      Console.WriteLine("결과는 {0}", sum);
    }
  }
}
```

문자열 연결하기

```
using System;

namespace StringConcat
{
  class Program
  {
    static void Main(string[] args)
    {
      string userName = "bikang";
      string date = DateTime.Today.ToShortDateString();

      string strPlus = "Hello " + userName + ". Today is " + date + ".";
      Console.WriteLine(strPlus);

      string strFormat = String.Format("Hello {0}. Today is {1}.", userName, date);
      Console.WriteLine(strFormat);

      string strInterpolation = $"Hello {userName}. Today is {date}.";
      System.Console.WriteLine(strInterpolation);
```

```
        //string name = "bikang";
        //int age = 12;
        //Console.WriteLine($"{name} is {age} year{(age == 1 ? "" : "s")} old.");
        //Console.WriteLine(name + " is " + age + " year" + (age == 1 ? "" : "s") + " old.");

        string strConcat = String.Concat("Hello ", userName, ". Today is ", date, ".");
        Console.WriteLine(strConcat);

        string[] animals = { "mouse", "cow", "tiger", "rabbit", "dragon"};
        string s = String.Concat(animals);
        Console.WriteLine(s);
        s = String.Join(", ", animals);
        Console.WriteLine(s);
    }
  }
}
```

String 클래스에는 문자열을 검색할 수 있는 여러가지 메소드들이 있습니다.

```
using System;

namespace A028_StringContains
{
  class Program
  {
    static void Main(string[] args)
    {
      string s1 = "mouse, cow, tiger, rabbit, dragon";
      string s2 = "cow";
      bool b = s1.Contains(s2);
      Console.WriteLine("'{0}' is in the string '{1}': {2}", s2, s1, b);

      if (b)
      {
        int index = s1.IndexOf(s2);
        if (index >= 0)
          Console.WriteLine("'{0} begins at index {1}", s2, index);
      }
```

```
        int i = s1.IndexOf(s2, StringComparison.CurrentCultureIgnoreCase);
        if (i >= 0)
        {
          Console.WriteLine("'{0}' is in the string '{1}' ", s2, s1);
          Console.WriteLine("at index {0} (case insensitive)", i);
        }
      }
    }
}
```

String.Format 메소드는 지정된 형식에 따라 객체, 변수, 수식의 값을 문자열로 변환하여 다른 문자열에 삽입합니다. 포맷 문자열에 {0}, {1}, ..과 같은 인덱스를 사용하여 표현하고자 하는 객체, 변수, 수식(이것을 포맷 아이템이라고 합니다)을 나타냅니다. 포맷 문자열 뒤에 나오는 파라미터는 0부터 순서대로 번호가 부여됩니다.
[0:C]와 같이 인덱스 뒤에 콜론과 함께 형식 지정자가 올 수 있습니다.

```
using System;

namespace A029_StringFormat
{
  class Program
  {
    static void Main(string[] args)
    {
      string max = String.Format("0x{0:X} {0:E} {0:N}", Int64.MaxValue);
      Console.WriteLine(max);

      Decimal exchangeRate = 1129.20m;

      string s = String.Format("현재 원달러 환율은 {0}입니다.", exchangeRate);
      Console.WriteLine(s);

      s = String.Format("현재 원달러 환율은 {0:C2}입니다.", exchangeRate);
      Console.WriteLine(s);
```

```
        s = String.Format("오늘 날짜는 {0:d}, 시간은 {0:t} 입니다.", DateTime.Now);
        Console.WriteLine(s);

        TimeSpan duration = new TimeSpan(1, 12, 23, 62);
        string output = String.Format("소요 시간: {0:c}", duration);
        Console.WriteLine(output);
    }
  }
}
```

숫자의 정수부를 표시할 때 세자리마다 콤마(,)를 넣는 것이 큰 수를 읽을 때 편리합니다. 이 콤마를 그룹 불리자(Group Separator)라고 합니다.
표준 형식 지정자 중에 N이 그룹 분리자를 표시해줍니다. N 형식 지정자는 디폴트로 소수점 아래 두 자리를 표시합니다. 소수점 아래 자릿수를 지정할 때는 N 뒤에 숫자로 표시합니다.

```
using System;

namespace A030_GroupSaparator
{
  class Program
  {
    static void Main(string[] args)
    {
      while (true)
      {
        Console.Write("표시할 숫자를 입력하세요(종료:-1): ");
        string s = Console.ReadLine();
        double v = double.Parse(s);
        if (v == -1)
          break;
        Console.WriteLine(NumberWithGroupSeparator(s));
      }
    }
```

```
    private static string NumberWithGroupSeparator(string s)
    {
      int pos = 0;
      double v = Double.Parse(s);

      if (s.Contains("."))
      {
        pos = s.Length - s.IndexOf('.');
        string formatStr = "{0:N" + (pos - 1) + "}";
        s = string.Format(formatStr, v);
      }
      else
        s = string.Format("{0:N0}", v);
      return s;
    }
  }
}
```

String 객체의 값은 한 번 만들면 변경할 수 없습니다. 이것을 불변(immutable)이라고 합니다. 객체가 변경될 때마다 새로운 string을 만들어서 변수에 할당해 주는 것입니다. 따라서 String의 값이 빈번하게 변경되는 경우에는 쓸데없이 스트링이 많이 만들어지고 그만큼 실행 속도와 메모리 사용이 많아집니다.

C#은 String 클래스와 비슷한 StringBuilder 클래스를 제공합니다. StringBuilder는 가변(mutable)입니다. 문자열이 변경되면 자동으로 필요한 메모리를 동적으로 조정하고 내용을 바꿀 수 있기 때문에 자주 변경되는 스트링을 다룰 때 효율적입니다.

```
using System;
using System.Diagnostics;
using System.Text;

namespace A031_StringBuilder
{
  class Program
  {
    static void Main(string[] args)
    {
      StringBuilder sb = new StringBuilder("This is a StringBuilder Test.");
      Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);

      sb.Clear();
      Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);

      sb.Append("This is a new string.");
      Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);
```

```
sb.Insert(5, "xyz ", 2);
Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);

sb.Remove(5, 4);
Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);

sb.Replace("xyz", "abc");
Console.WriteLine("{0} ({1} characters)", sb.ToString(), sb.Length);

Stopwatch time = new Stopwatch();
string test = string.Empty;
time.Start();
for (int i = 0; i < 100000; i++)
{
  test += i;
}
time.Stop();
Console.WriteLine("Using String concatenation: " + time.ElapsedMilliseconds + " milliseconds");
```

```
        StringBuilder test1 = new StringBuilder();
        time.Reset();
        time.Start();
        for (int i = 0; i < 100000; i++)
        {
          test1.Append(i);
        }
        time.Stop();
        Console.WriteLine("Using StringBuilder: " + time.ElapsedMilliseconds + " milliseconds");
      }
    }
}
```

열거형은 서로 관련있는 상수들의 집합을 정의한 것입니다. 숫자에 특정한 명칭을 붙여주어 의미를 쉽게 이해할 수 있게 하는 용도로 사용됩니다.

원소로 기술된 명칭을 기호상수라고 부르며 명시된 순서에 따라 디폴트로 0부터 순서대로 정수값을 갖게 됩니다.

```
using System;

namespace A032_Enum
{
  class Program
  {
    enum Size { Short, Tall, Grande, Venti };
    static int[] price = { 3300, 3800, 4300, 4800 };
    enum Colors { Red = 1, Green = 2, Blue = 4, Yellow = 8 };
    enum Coffee { Short = 3300, Tall = 3800, Grande = 4300, Venti = 4800 };

    static void Main(string[] args)
    {
      Console.WriteLine("커피 가격표");
```

```
for(int i=0; i<4; i++)
{
  if (i == (int)Size.Short)
    Console.WriteLine("{0,10} : {1:C}", Size.Short, price[i] );
  else if(i == (int)Size.Tall)
    Console.WriteLine("{0,10} : {1:C}", Size.Tall, price[i]);
  else if (i == (int)Size.Grande)
    Console.WriteLine("{0,10} : {1:C}", Size.Grande, price[i]);
  else if (i == (int)Size.Venti)
    Console.WriteLine("{0,10} : {1:C}", Size.Venti, price[i]);
}

Console.WriteLine("₩n커피 가격표(Enum iteration)");
foreach (var size in Enum.GetValues(typeof(Size)))
{
  Console.WriteLine("{0,10} : {1:C}", size, price[(int)size]);
}
```

```
        Console.WriteLine("₩nColors Enum iteration");
        foreach (var color in Enum.GetValues(typeof(Colors)))
        {
          Console.WriteLine("{0,10} : {1}", color, Convert.ToInt32(color));
        }

        Console.WriteLine("₩n커피 가격표(Enum iteration with value)");
        foreach (var coffee in Enum.GetValues(typeof(Coffee)))
        {
          Console.WriteLine("{0,10} : {1:C}", coffee, Convert.ToInt32(coffee));
        }
    }
  }
}
```

```
using System;

namespace A033_ConstAndReadonly
{
  class ConstEx
  {
    public const int number = 3;
  }

  class ReadonlyEx
  {
    public readonly int number = 10;
    public ReadonlyEx()
    {
      number = 20;
    }
    public ReadonlyEx(int n)
    {
      number = n;
    }
  }
```

C#에서는 "읽기 전용 " 이라는 뜻의 readonly라는 키워드도 있습니다. Readonly 키워드를 붙인 변수는 변수를 선언한 시점과 생성자 메소드에서만 값을 변경할 수 있고, 그 외의 경우에 변경하면 오류가 발생합니다..

const
선언될 때 값이 할당됩니다.
Classname,.VarialbeName으로 사용해야 합니다.
컴파일 시에 값이 결정됩니다.

readonly
실행될 때 또는 객체가 생성자에 의해 초기화될 때 값이 할당됩니다.
InstanceName.VariableName으로 사용해야 합니다.
런타임 시에 값이 결정됩니다.

```
class Program
{
  static void Main(string[] args)
  {
    // const 사용
    Console.WriteLine(ConstEx.number);

    // readonly 사용
    ReadonlyEx inst1 = new ReadonlyEx();
    Console.WriteLine(inst1.number);

    ReadonlyEx inst2 = new ReadonlyEx(100);
    Console.WriteLine(inst2.number);
  }
}
}
```

C#의 자료형에는 두 가지 타입이 있습니다. 값 형식(value type)과 참조 형식(reference type)입니다. 값 형식은 변수가 실제 데이터 값을 저장하는 형식이고, 참조 형식은 변수가 값이 저장되어 있는 곳의 위치(이것을 참조, reference라고 합니다.)를 저장하는 형식입니다. C 언어의 포인터와 비슷한 개념입니다.

String 이나 배열은 참조형식입니다. 굉장히 긴 문자열이나 굉장히 큰 배열도 있을 수 있기 때문에 문자열이나 배열의 데이터는 힙 영역에 저장하고 스택에는 참조만 저장합니다.

```
using System;

namespace A034_ValueAndReference
{
  class Program
  {
    static void Main(string[] args)
    {
      string s = "before passing";
      Console.WriteLine(s);

      Test(s);
      Console.WriteLine(s);

      Test(ref s);
      Console.WriteLine(s);
    }

    public static void Test(string s)
    {
      s = "after passing";
    }

    public static void Test(ref string s)
    {
      s = "after passing";
    }
  }
}
```

```
using System;

namespace A035_PassingArrayAndObject
{
  class Program
  {
    static void Main(string[] args)
    {
      int[] arr = { 10, 20, 30 };
      Console.WriteLine("Main() before: arr[0] = {0}", arr[0]);

      Change(arr);
      Console.WriteLine("Main() after: arr[0] =  {0}", arr[0]);

      Student s1 = new Student();
      s1.name = "Alpha";
      Console.WriteLine("Main() before: " + s1.name);
```

배열은 참조형이고 배열의 이름은 그 배열이 저장된 곳의 참조입니다. 따라서 메소드를 호출할 때 배열의 이름을 매개변수로 전달하면 배열의 참조가 전달됩니다. 이때 호출된 메소드 안에서 배열의 내용을 바꾸면 호출한 곳에서도 배열이 바뀌게 됩니다.
클래스의 객체도 참조형이므로 배열과 같이 호출된 메소드 안에서 내용을 바꾸면 호출한 곳에서도 바뀌게 됩니다.

```
      Change(s1);
      Console.WriteLine("Main() after: " + s1.name);
    }

    private static void Change(int[] arr)
    {
      arr[0] = -10;
    }

    private static void Change(Student s1)
    {
      s1.name = "Beta";
    }
  }

  class Student
  {
    public string name;
  }
}
```

```
using System;

namespace A036_NullCOnditionalOperator
{
  class Program
  {
    static void Main(string[] args)
    {
      string animal = null;

      Console.WriteLine("4글자 이상인 동물의 이름만 출력합니다.");
      do
      {
        LongNameAnimal(animal);
        Console.Write("동물이름: ");
      } while ((animal = Console.ReadLine()) != "");
    }
```

```
    private static void LongNameAnimal(string animal)
    {
      if(animal.Length >= 4)
        Console.WriteLine(animal + " : " + animal.Length);
    }
  }
}
```

C#프로그래밍에서 null이란 "어떤 객체도 참조하지 않는 참조형 변수 " 라는 뜻입니다. null은 참조형 변수의 디폴트 값입니다.
참조형 변수의 속성을 사용하려고 하면그 변수는 null이 아니어야 합니다.
C#6.0부터 null 조건 연산자 ?가 도입되었습니다.
다음과 같이 null 조건 연산자 ?를 사용하면 s가 null 일 때는 Length를 찾지 않습니다.
If(s?.Length > 0)

```
using System;
using System.Collections.Generic;

namespace A037_Default
{
  class Program
  {
    enum E { Red, Green, Blue };

    static void Main(string[] args)
    {
      int a = default;
      string s = default;
      Console.WriteLine("a = " + a);
      Console.WriteLine("s = " + s);
      Console.WriteLine("E = " + default(E));
      Console.WriteLine("E = " + (E)0);

      GenericList<int> iList = new GenericList<int>();
      Console.WriteLine("iList : " + iList.GetLast());

      GenericList<string> sList = new GenericList<string>();
      Console.WriteLine("sList : " + sList.GetLast());
```

C#에서는 초기화되지 않은 변수를 사용할 수 없습니다. 특정한 값을 할당해주거나 해당 형식의 기본값(default)를 사용하여 변수를 초기화하여야 합니다.

Deault식은 특히 제네릭 클래스와 메소드에서 유용합니다. 제네릭 사용으로 발생하는 문제는 매개변수의 형식 T에 기본값을 할당하는 방법입니다.

Default 식 T가 어떠한 타입이라도 문제없이 초기화를 해줍니다.

```
        //sList.PrintList();
        //sList.AddNode("Tiger");
        //sList.AddNode("Lion");
        //iList.PrintList();
        //iList.AddNode(10);
        //iList.AddNode(20);
        //Console.WriteLine(iList.GetLast());
    }
}

public class GenericList<T>
{
    private class Node
    {
        public T data;
        public Node next;
    }
    private Node head = default;
```

```
public void AddNode(T t)
{
  Node newNode = new Node();
  newNode.next = head;
  newNode.data = t;
  head = newNode;
}

public void PrintList()
{
  for(Node ptr=head; ptr!=default; ptr=ptr.next)
    Console.Write(ptr.data + " ");
  Console.WriteLine();
}
```

```
    public T GetLast()
    {
      // The value of temp is returned as the value of the method.
      // The following declaration initializes temp to the appropriate
      // default value for type T. The default value is returned if the
      // list is empty.
      T temp = default(T);

      Node current = head;
      while (current != null)
      {
        temp = current.data;
        current = current.next;
      }
      return temp;
    }
  }
}
```

```
using System;

namespace A038_Nullable
{
  class Program
  {
    static void Main(string[] args)
    {
      Nullable<int> i = null;
      Console.WriteLine("i="+i);
      Console.WriteLine(i.GetValueOrDefault());

      if (i.HasValue)
        Console.WriteLine(i.Value); // or Console.WriteLine(i)
      else
        Console.WriteLine("Null");
```

Nullable<int>는 보통의 int에 그 변수가 값을 가지고 있는지 아닌지를 표현하는 플래그를 가지고 있는 셈입니다. 즉, null인지 아닌지를 알 수 있습니다.
GetValueOrDefault() 메소드는 값이 있으면 그 값을 가져오고, null이면 디폴트 값을 가져옵니다.

값 형식의 변수에는 null 값을 할당할 수 없습니다.
C#2.0부터 nullable형을 도입해서 값 형식의 변수에 null을 할당할 수 있게 했습니다.
nullable형은 Nullable<T>를 사용해서 선언합니다.

```
        int? x = null;
        int j = x ?? 0;
        Console.WriteLine("x = {0}, j = {1}", x, j);

        Console.WriteLine("x >= 10 ? {0}", x >= 10);
        Console.WriteLine("x < 10 ? {0}", x < 10);

        if (Nullable.Compare<int>(i, j) < 0)
          Console.WriteLine("i < j");
        else if (Nullable.Compare<int>(i, j) > 0)
          Console.WriteLine("i > j");
        else
          Console.WriteLine("i = j");
    }
  }
}
```

Nullable<T>를 쓰는 대신 ? 연산자를 쓸 수 있습니다.
Nullable<int>, Nullable<double>대신에
int? i = null;
double? d = null;
로 쓸 수 있습니다.

Nullable 타입을 non-nullable 타입에 할당할 때는 ?? 연산자를 사용해야 합니다.
?? 연산자는 null일 때 0을 할당해 줍니다.

Nullable 변수를 비교할 때는 Nullable.Compare() 메소드를 사용합니다. 이때 주의할 점은 null 값은 모든 값보다 작다는 것입니다.

```
using System;

namespace A039_Object
{
  class Program
  {
    static void Main(string[] args)
    {
      int i = 123;

      object o = i; // i의 값을 박싱하여 o로 복사합니다

      i = i + 10; // i의 값을 바꿉니다. o는 변하지 않습니다.

      int j = (int)o; // o의 값을 언박싱하여 j로 복사합니다.
```

C#에서 모든 형식은 기본 형식이나 사용자가 정의한 형식이나, 값 형식이나, 참조 형식이나 상관없이 모두 Object로부터 상속됩니다.

```
        // i의 값이 변해도 o에 저장된 값은 영향받지 않습니다.
        Console.WriteLine("The value-type value i = {0}", i);
        Console.WriteLine("The object-type value o = {0}", o);
        Console.WriteLine("The value-type value j = {0}", j);

        object p = o;
        o = 100;
        Console.WriteLine("The object-type value o = {0}", o);
        Console.WriteLine("The object-type value p = {0}", p);
    }
  }
}
```

따라서 object 타입은 모든 데이터의 조상이고 object 타입의 변수에는 어떠한 값이라도 할당할 수가 있습니다.
Object와 object는 같은 말입니다.
값 형식의 변수가 object 타입으로 변환되는 것을 박싱(Boxing)이라고 하고 object 타입의 변수가 값 형식으로 변환되는 것을 언박싱(Unboxing)이라고 합니다.

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