C#(1)

언어	C#
유형	정규_과제/스터디

1. 문자열 보간을 사용하여 형식이 지정된 문자열 생성

https://learn.microsoft.com/ko-kr/dotnet/csharp/tutorials/exploration/interpolated-strings

1-1. 보간된 문자열 만들기

```
var name = "<name>";
Console.WriteLine($"Hello, {name}. It's a pleasure to meet you!");

Hello, <name>. It's a pleasure to meet you!
```

다음과 같이 인사말에 사용자 이름이 포함된 문자열이 출력에 표시됨.

- $문자 뒤에 공백이 없어야 한다.
- 보간 식은 { } 처럼 중괄호로 표시된다. 중괄호 안에 값을 반환하는 C#식을 배치할 수 있다.

1-2. 다양한 데이터 형식 포함

```
var item = (Name: "eggplant", Price: 1.99m, perPackage: 3);
var date = DateTime.Now;
Console.WriteLine($"On {date}, the price of {item.Name} was {item.Price} per

On 2025. 4. 10. 오후 10:05:15, the price of eggplant was 1.99 per 3 items.
```

1-3. 보간 식의 서식 제어

```
var item = (Name: "eggplant", Price: 1.99m, perPackage: 3);
var date = DateTime.Now;
Console.WriteLine($"On {date:d}, the price of {item.Name} was {item.Price:C2
```

위의 코드에 형식 문자열을 지정함.

date : d에서 d를 t나 y, yyyy 등으로 바꾸면 해당 형식 문자열에 맞게 수정됨.

t : 짧은 시간 형식 표시

y : 연도 및 월 표시

yyyy : 연도를 4자리 숫자로 표시

1-4. 필드 너비와 보간 식의 맞춤 제어

```
var inventory = new Dictionary<string, int>()
 {
    ["hammer, ball pein"] = 18,
    ["hammer, cross pein"] = 5,
    ["screwdriver, Phillips #2"] = 14
 };

 Console.WriteLine($"Inventory on {DateTime.Now:d}");
 Console.WriteLine(" ");
 Console.WriteLine($"|{"Item",-25}|{"Quantity",10}|");
 foreach (var item in inventory)
   Console.WriteLine($"|{item.Key,-25}|{item.Value,10}|");
```

```
Inventory on 2025. 4. 10.

|Item                      |  Quantity|
|hammer, ball pein         |        18|
```

```
|hammer, cross pein       |        5|
|screwdriver, Phillips #2 |       14|
```

항목 이름을 왼쪽 맞춤, 해당 수량은 오른쪽 맞춤.
보간 식 뒤에 쉼표를 추가하고 최소 필드 너비를 지정해서 맞춤을 지정.
(만일 음수 기호를 제거하고 예제를 실행하면 항목 이름은 오른쪽으로 맞춤이 될 것.)

2. C#의 문자열 보간

https://learn.microsoft.com/ko-kr/dotnet/csharp/tutorials/string-interpolation#code-try-0

문자열 리터럴을 보간된 문자열로 식별하려면 $ 기호를 사용해서 추가하면 됨.

ex)

```
double a = 3;
double b = 4;
Console.WriteLine($"Area of the right triangle with legs of {a} and {b} is {0.5 *
Console.WriteLine($"Length of the hypotenuse of the right triangle with legs o
double CalculateHypotenuse(double leg1, double leg2) => Math.Sqrt(leg1 * leg
// Output:
// Area of the right triangle with legs of 3 and 4 is 6
// Length of the hypotenuse of the right triangle with legs of 3 and 4 is 5
```

2-1. 서식 문자열 지정하기

식 결과의 형식에서 지원되는 형식 문자열을 지정하려면 콜론 및 형식 문자열을 사용하여 보간 식을 따른다.

```
var date = new DateTime(1731, 11, 25);
Console.WriteLine($"On {date:dddd, MMMM dd, yyyy} L. Euler introduced the
// Output:
// On Sunday, November 25, 1731 L. Euler introduced the letter e to denote 2.7
```

다음과 같이 :F5를 통해 실수 5자리까지 나오는 것을 확인할 수 있다.

2-2. 필드 너비와 서식이 지정된 보간 식의 맞춤을 제어하는 방법

```
const int NameAlignment = -9;
const int ValueAlignment = 7;
double a = 3;
double b = 4;
Console.WriteLine($"Three classical Pythagorean means of {a} and {b}:");
Console.WriteLine($"|{"Arithmetic",NameAlignment}|{0.5 * (a + b),ValueAlignr
Console.WriteLine($"|{"Geometric",NameAlignment}|{Math.Sqrt(a * b),ValueA
Console.WriteLine($"|{"Harmonic",NameAlignment}|{2 / (1 / a + 1 / b),ValueAli
// Output:
// Three classical Pythagorean means of 3 and 4:
// |Arithmetic|  3.500|
// |Geometric|  3.464|
// |Harmonic |  3.429|
```

파이프 문자를 사용해서 텍스트 필드를 구분함.

2-3. 보간 식에서 3개로 구성된 ?: 조건부 연산자를 사용하는 방법

```
var rand = new Random();
for (int i = 0; i < 7; i++)
{
    Console.WriteLine($"Coin flip: {(rand.NextDouble() < 0.5 ? "heads" : "tails")
}

//output
//Coin flip: heads
//Coin flip: heads
//Coin flip: tails
//Coin flip: tails
```

```
//Coin flip: tails
//Coin flip: heads
//Coin flip: tails
```

보간 식에 콜론에 특별한 의미가 있으니까, 식에서 조건부 연산자를 활용하기 위해 해당 식을 괄호로 묶어버림.

3. 인덱스 및 범위

https://learn.microsoft.com/ko-kr/dotnet/csharp/tutorials/ranges-indexes

3-1. 인덱스 및 범위에 대한 언어 지원

- System.Index : 인덱스를 시퀀스로 표현.
- System.Range : 시퀀스의 하위 범위를 나타냄.

```
string[] words = [
                // index from start    index from end
    "The",      // 0                   ^9
    "quick",    // 1                   ^8
    "brown",    // 2                     ^7
    "fox",      // 3                   ^6
    "jumps",    // 4                     ^5
    "over",     // 5                   ^4
    "the",      // 6                   ^3
    "lazy",     // 7                   ^2
    "dog"       // 8                   ^1
];              // 9 (or words.Length) ^0
```

이런 인덱스가 있을 때, 마지막 단어를 가져오려면

```
Console.WriteLine($"The last word is {words[^1]}");
```

을 하면 됨.

- 인덱스 ^n은 sequence.Length -n과 동일함.
- [0..^0]은 전체 범위를 나타냄.

quick, brown, fox라는 단어를 포함하는 하위 범위를 만드려면

```
string[] quickBrownFox = words[1..4];
foreach (var word in quickBrownFox)
    Console.Write($"< {word} >");
Console.WriteLine();
```

여기에는 words[1]부터 words[3]까지가 포함됨. [1..4]니까.

또다른 예시.

```
string[] allWords = words[..]; // contains "The" through "dog".
//시작과 끝이 모두 열림.
string[] firstPhrase = words[..4]; // contains "The" through "fox"
//시작만 열림.
string[] lastPhrase = words[6..]; // contains "the", "lazy" and "dog"
//끝만 열림.
foreach (var word in allWords)
    Console.Write($"< {word} >");
Console.WriteLine();
foreach (var word in firstPhrase)
    Console.Write($"< {word} >");
Console.WriteLine();
foreach (var word in lastPhrase)
    Console.Write($"< {word} >");
Console.WriteLine();
```

범위나 인덱스를 변수로 선언할 수도 있고, 그때는 변수를 [ ] 사이에 사용하면 된다.

```
Index the = ^3;
Console.WriteLine(words[the]);
Range phrase = 1..4;
string[] text = words[phrase];
foreach (var word in text)
```

```
    Console.Write($"< {word} >");
Console.WriteLine();
```

3-2. 암시적 범위 연산자 식 변환

컴파일러가 암시적으로 시작 및 끝 값을 index로 변환하고 이 값에서 새 range 인스턴스를 만든다.

```
Range implicitRange = 3..^5;

Range explicitRange = new(
    start: new Index(value: 3, fromEnd: false),
    end: new Index(value: 5, fromEnd: true));

if (implicitRange.Equals(explicitRange))
{
    Console.WriteLine(
        $"The implicit range '{implicitRange}' equals the explicit range '{explicitR
}
// Sample output:
//     The implicit range '3..^5' equals the explicit range '3..^5'
```

3-3. 활용 : 인덱스 및 범위에 대한 시나리오

로컬 함수 MovingAverage는 range를 인수로 사용하는데, 그러면 메서드가 최소, 최대, 평균을 계산할 때 이 범위만 열거하게 됨.

```
int[] sequence = Sequence(1000);

for(int start = 0; start < sequence.Length; start += 100)
{
    Range r = start..(start+10);
    var (min, max, average) = MovingAverage(sequence, r);
    Console.WriteLine($"From {r.Start} to {r.End}:    \tMin: {min},\tMax: {max},\t
}
```

```
for (int start = 0; start < sequence.Length; start += 100)
{
    Range r = ^(start + 10)..^start;
    var (min, max, average) = MovingAverage(sequence, r);
    Console.WriteLine($"From {r.Start} to {r.End}:  \tMin: {min},\tMax: {max},\tA
}

(int min, int max, double average) MovingAverage(int[] subSequence, Range r
    (
        subSequence[range].Min(),
        subSequence[range].Max(),
        subSequence[range].Average()
    );

int[] Sequence(int count) ⇒ [..Enumerable.Range(0, count).Select(x ⇒ (int)(Ma
```

첫 번째 반복문은 순방향으로, 두 번째 반복문은 역방향으로 인덱스를 나타냄.

3-4. 범위 인덱스 및 배열 참고 사항

배열에서 범위를 가져오면 결과는 참조되지 않고 초기 배열에서 복사만 됨. 그래서 결과 배열의 값을 수정해도 초기 배열의 값에는 변화가 없다.

```
var arrayOfFiveItems = new[] { 1, 2, 3, 4, 5 };

var firstThreeItems = arrayOfFiveItems[..3]; // contains 1,2,3
firstThreeItems[0] =  11; // now contains 11,2,3

Console.WriteLine(string.Join(",", firstThreeItems));
Console.WriteLine(string.Join(",", arrayOfFiveItems));

// output:
// 11,2,3
// 1,2,3,4,5
```

4. 기본 인터페이스 메서드로 인터페이스를 업데이트

https://learn.microsoft.com/ko-kr/dotnet/csharp/advanced-topics/interface-implementation/default-interface-methods-versions

- 구현으로 메서드 추가해서 안전하게 인터페이스 확장하기
- 매개 변수가 있는 구현을 생성해서 향상된 유연성 제공하기
- 구현자가 재정의 형식으로 더 구체적인 구현을 제공하도록 지원하기

⇒에 대해 실습해보기.

ex) 고객에 대한 인터페이스 정의

```
public interface ICustomer
{
    IEnumerable<IOrder> PreviousOrders { get; }

    DateTime DateJoined { get; }
    DateTime? LastOrder { get; }
    string Name { get; }
    IDictionary<DateTime, string> Reminders { get; }
}
```

주문을 나타내는 두 번째 인터페이스

```
public interface IOrder
{
    DateTime Purchased { get; }
    decimal Cost { get; }
}
```

이 상태에서 각 customer마다 다르게 할인을 적용해야 한다면....

ICustomer 인터페이스를 개선해야 한다.

방법1 : 기본 인터페이스 메서드를 사용해서 업그레이드

```
// Version 1:
public decimal ComputeLoyaltyDiscount()
{
    DateTime TwoYearsAgo = DateTime.Now.AddYears(-2);
    if ((DateJoined < TwoYearsAgo) && (PreviousOrders.Count() > 10))
    {
        return 0.10m;
    }
    return 0;
}
```

그러나 이 구현은 너무 제한적이고, 만약 소비자가 구매 건수에 다른 임계값, 다른 멤버십 기간, 또는 다른 할인율을 선택하고 싶다고 하면 처리가 어렵다는 단점이 있음.

따라서 매개 변수를 설정하는 방법을 제공해서 향상된 업그레이드 환경 제공이 가능.

방법2 : 매개 변수화 제공

```
// Version 2:
public static void SetLoyaltyThresholds(
    TimeSpan ago,
    int minimumOrders = 10,
    decimal percentageDiscount = 0.10m)
{
    length = ago;
    orderCount = minimumOrders;
    discountPercent = percentageDiscount;
}
private static TimeSpan length = new TimeSpan(365 * 2, 0,0,0); // two years
private static int orderCount = 10;
private static decimal discountPercent = 0.10m;

public decimal ComputeLoyaltyDiscount()
{
    DateTime start = DateTime.Now - length;
```

```
    if ((DateJoined < start) && (PreviousOrders.Count() > orderCount))
    {
        return discountPercent;
    }
    return 0;
}
```