배열과 컬렉션, 그리고 인덱서

이것이 C#이다

Contents

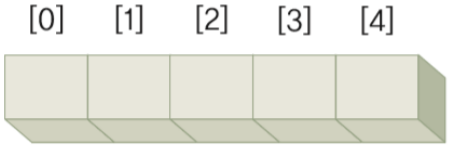
10.1 All for one, one for all

❖ 같은 성격을 띤 다수의 데이터를 한번에 다뤄야 하는 경우

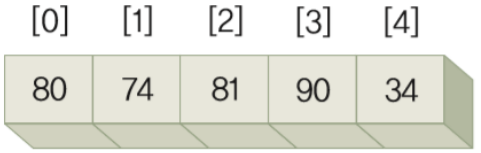
- 수 많은 변수 대신 그 만큼의 용량을 가진 변수 하나로 해결한다면?

❖ 배열 선언 형식

- **데이터형식[ ] 배열이름** = new **데이터형식[용량]**;
- int[] array = new int[5];




```
int[] scores = new int[5];
scores[0] = 80;
scores[1] = 74;
scores[2] = 81;
scores[3] = 90;
scores[4] = 34;
```



❖ for나 foreach 문을 이용해 코드를 훨씬 간결 만들 수 있다

변수	배열
Console.WriteLine (score_1); Console.WriteLine (score_2); Console.WriteLine (score_3); Console.WriteLine (score_4); Console.WriteLine (score_5);	foreach (int score in scores) Console.WriteLine(score);

변수	배열

int average = (score_1 + score_2 + score_3 + score_4 + score_5) / 5;	int sum = 0;
	foreach (int score in scores)
	  sum += score;
	
	int average = sum / scores.Length;

❖ 데모 예제 - ArraySample

10.2 배열을 초기화하는 방법 세 가지

❖ 배열 크기를 명시하고 컬렉션 초기자를 사용

배열의 용량을 명시

```
string[] array1 = new string[3]{ "안녕", "Hello", "Halo" };
```

❖ 배열 크기를 명시하지 않고 컬렉션 초기자 사용

배열의 용량을 생략

```
string[] array2 = new string[] { "안녕", "Hello", "Halo" };
```

❖ 컬렉션 초기자만 사용

```
string[] array3 = { "안녕", "Hello", "Halo" };
```

❖ 데모 예제 - InitializingArray

10.3 알아 두면 삶이 윤택해지는 System.Array

❖ 배열은 System.Array 형식에서 파생

```
Type Of array : System.Int32[]
Base type Of array : System.Array
```

❖ Array 클래스의 주요 메소드와 프로퍼티.

분류	이름	설명
정적 메소드	FindIndex<T>( )	배열에서 지정한 조건에 부합하는 첫 번째 요소의 인덱스를 반환합니다. IndexOf( ) 메소드가 특정 값을 찾는데 비해, FindIndex<T>( ) 메소드는 지정한 조건에 바탕하여 값을 찾습니다.
	Resize<T>( )	배열의 크기를 재조정합니다.
	Clear( )	배열의 모든 요소를 초기화합니다. 배열이 숫자 형식 기반이면 0으로, 논리 형식 기반이면 false로, 참조 형식 기반이면 null로 초기화합니다.
	ForEach<T>( )	배열의 모든 요소에 대해 동일한 작업을 수행하게 합니다.
인스턴스 메소드	GetLength( )	배열에서 지정한 차원의 길이를 반환합니다. 이 메소드는 나중에 설명하게 될 다차원 배열에서 유용하게 사용됩니다.
프로퍼티	Length	배열의 길이를 반환합니다.
	Rank	배열의 차원을 반환합니다.

❖ 데모 예제 - MoreOnArray

10.4 2차원 배열

- **2개의 차원(세로+가로)으로 원소 배치**
  - 1차원 배열을 원소로 갖는 배열
- **사용 형식**
  - **데이터형식**[,] **배열이름** = new **데이터형식**[2**차원길이**, 1**차원길이**];
- **배열 선언 예**

```
int[,] arr = new int[2, 3]{ {1, 2, 3}, {4, 5, 6} };     // 배열의 형식과 길이를 명시
int[,] arr2 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } }; // 배열의 길이를 생략
int[,] arr3 = { { 1, 2, 3 }, { 4, 5, 6 } };            // 형식과 길이를 모두 생략
```

- **데모 예제 – 2DArray**

10.5 다차원 배열

- 차원이 둘 이상인 배열 (2차원 이상)
- 3차원 배열
  - 2차원 배열을 요소로 갖는 배열
  - 배열의 각 차원 크기를 미리 정해주는 초기화 방법 권장
  - →컴파일타임에 초기화 코드와 선언문의 배열 차원 크기 비교

사용 예

```
int[, ,] array = new int[4, 3, 2]
            {
                { {1, 2}, {3, 4}, {5, 6} },
                { {1, 4}, {2, 5}, {3, 6} },
                { {6, 5}, {4, 3}, {2, 1} },
                { {6, 3}, {5, 2}, {4, 1} },
            };
```

데모 예제 – 3DArray

10.6 가변 배열

- 다양한 길이의 배열을 요소로 갖는 다차원 배열
- 선언 형식
  - **데이터형식**[ ][ ] **배열이름** = new **데이터형식**[**가변 배열의 용량**][ ];
- 선언 예

```
int[][] jagged = new int[3][];
jagged[0] = new int[5] { 1, 2, 3, 4, 5 };
jagged[1] = new int[] { 10, 20, 30 };
jagged[2] = new int[] { 100, 200 };
```

```
int[][] jagged2 = new int[2][] {
    new int[] { 1000, 2000 },
    new int[4] { 6, 7, 8, 9 } };
```

- 데모 예제 - JaggedArray

10.7 컬렉션 맛보기

- 컬렉션-같은 성격의 데이터 모음을 담는 자료 구조
- 배열(System.Array 클래스)도 컬렉션 자료 구조

```
public abstract class Array : ICloneable,
                              IList, ICollection, IEnumerable
```

- .NET 프레임워크의 컬렉션 클래스
  - ArrayList
  - Queue
  - Stack
  - Hashtable

10.7.1 ArrayList

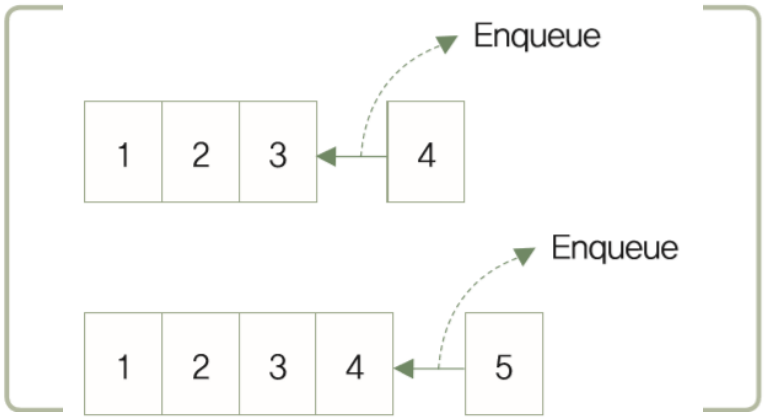
- 배열 처럼 [] 연산자 이용, 특정 위치 요소에 데이터 할당
- 용량을 미리 지정할 필요 없고 필요에 따라 용량 증가/감소
- 대표적인 메소드
  - Add(), RemoveAt(), Insert()
- 사용 예

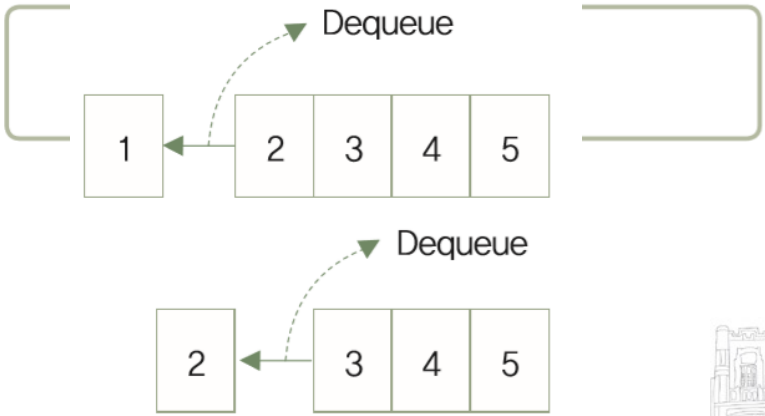
```
ArrayList list = new ArrayList();
list.Add( 10 );
list.Add( 20 );
list.Add( 30 );
list.RemoveAt( 1 ); // 20을 삭제
list.Insert( 1, 25 ); // 25를 1번 인덱스에 삽입. 즉, 10과 30 사이에 25를 삽입
```

- 데모 예제 - UsingList

10.7.2 Queue

- 데이터를 차례대로 넣고 입력한 순서로 하나씩 처리
  - CPU의 작업, 프린터의 여러 문서 출력, 스트리밍 서비스에서 콘텐츠 버퍼링
- 입력은 오직 뒤에서, 출력은 앞에서만
  - Enqueue()
  - Dequeue() 메소드





- 데모 예제 – UsingQueue

10.7.3 Stack

- First In - Last Out, Last In - First Out 컬렉션
- 데이터 입력은 Push(), 데이터 출력은 Pop()

```
Stack stack = new Stack();
stack.Push( 1 ); // 최상위 데이터는 1
stack.Push( 2 ); // 최상위 데이터는 2
stack.Push( 3 ); // 최상위 데이터는 3

int a = (int)stack.Pop(); // 최상위 데이터는 다시 2
```

- 데모 예제 - UsingStack

10.7.4 Hashtable

- 키와 값의 쌍으로 이루어진 데이터를 다룰 때 사용
  - 탐색 속도가 빠르고 사용하기 편리
- 배열과 비교한 장점
  - 데이터를 저장할 요소의 위치로 키 사용
  - 키로 사용할 수 있는 데이터 형식에 제한이 없음
  - 키를 이용해 단번에 데이터 저장 위치인 컬렉션 내의 주소 계산(해싱)
- 사용 방법

```
Hashtable ht = new Hashtable();
ht["book"]    = "책";
ht["cook"]    = "요리사";
ht["tweet"]   = "지저귀다";

Console.WriteLine( ht["book"] ) ;
Console.WriteLine( ht["cook"] );
Console.WriteLine( ht["tweet"] );
```

- 데모 예제- UsingHashtable

10.8 컬렉션을 초기화하는 방법

❖ 배열을 통한 초기화

```
int[] arr = { 123, 456, 789 };

ArrayList list = new ArrayList(arr); // 123, 456, 789
Stack stack = new Stack(arr); // 789, 456, 123
Queue queue = new Queue(arr); // 123, 456, 789
```

❖ 배열의 도움 없이 직접 컬렉션 초기자로 초기화

```
ArrayList list2 = new ArrayList() { 11, 22, 33 };
```

컬렉션 초기자는 생성자를 호출할 때, 생성자 뒤에 {와 } 사이에 컬렉션 요소의 목록을 입력하여 사용합니다.

❖ 딕셔너리 초기자를 이용한 초기화

```
Hashtable ht = new Hashtable()
{
    ["하나"] = 1, // ;가 아니라 ,를 이용하여 항목을 구분합니다.
    ["둘"] = 2,
    ["셋"] = 3
};
```

❖ 데모 예제 – InitializingCollections

10.9 인덱서

- 인덱스를 이용해 객체 내 데이터에 접근하게 하는 프로퍼티
- 인덱서 선언 형식과 예

```
public int this[int index]
{
    get
    {
        return array[index];
    }

    set
    {
        if (index >= array.Length)
        {
            Array.Resize<int>(ref array, index + 1);
            Console.WriteLine("Array Resized : {0}", array.Length);
        }

        array[index] = value;
    }
}
```

가 꼭 index일 필요는 없습
라 적당한 이름을 사용하세요.

- 데모 예제 - Indexer

10.10 foreach가 가능한 객체 만들기

- foreach 구문은 IEnumerable 과 IEnumerator를 상속하는 형식만 지원
- IEnumerable의 유일한 메소드 – GetEnumerator( )
  - 구현 시 yield return 문 필요
  - IEnumerator 형식의 객체 반환
- IEnumerator 인터페이스의 메소드 및 프로퍼티 목록

메소드 또는 프로퍼티	설명
boolean MoveNext( )	다음 요소로 이동합니다. 컬렉션의 끝을 지난 경우에는 false, 이동이 성공한 경우에는 true를 반환합니다.
void Reset( )	컬렉션의 첫 번째 위치의 "앞"로 이동합니다. 첫 번째 위치가 0번이라면, Reset( )을 호출하면 –1번으로 이동하는 것이죠. 첫 번째 위치로의 이동은 MoveNext( )를 호출한 다음에 이루어집니다.
Object Current { get; }	컬렉션의 현재 요소를 반환합니다.

- 데모 예제 – Yield
- 데모 예제 – Enumerable

Thank You !

이것이 C#이다