chapter 06 배열

일차원배열(Array)

- 변수 선언이 많아지는 경우
- 동일한 데이터형
- 배열의 GetType()
- int[] arrNum(식별) = new int[5]; <- 자료형의 크기

일차원 배열의 선언과 초기화

- int[] arrNum = new int[3]
- int[] arrNum =new int[]{0,1,2}
- int[] arrNum = new int[3]{0,1,2}
- int[] arrNum = {0,1,2}
- 인덱스(index) 란?
  - 배열에 접근하는 위치 번호
  - 인덱스는 0부터 시작(0~(N-1))

foreach - 반복문

- 읽기 전용 : 배열의 값을 변경할 수 없다
- foreach(자료형 식별자 in 배열)

다차원 배열

- 행과 열과 구분
- int[,] arrNums = new int[3.2

	[0]	[1]
[0]]	[0,0]	[0,1]
[1]	[1,0]	[1,1]
[2]	[2,0]	[2,1]

이차원 배열 선언과 초기화

- int[,] arrNum = new int[3,2];
- int[,] arrNum = new int[,]{0,1},{2,3},{4,5}};
- int [,] arrNum = new int[3,2]{{0,1},{2,3},{4,5}};
- int[,] arrNum = {{0,1},{2,3},{4,5}};

```
static void Main(string[] args)
       {
           int[,] arrNum = new int[3, 2];

           arrNum[0, 0] = 1;
           arrNum[0, 1] = 2;

           arrNum[1, 0] = 3;
           arrNum[1, 1] = 4;

           arrNum[2, 0] = 5;
           arrNum[2, 1] = 6;

           foreach(int temp in arrNum) {
               Console.Write("  {0}", temp);
           }

           Console.WriteLine();
       }
```

- 각 배열의 위치에 값을 할당해준다.

다차원 배열 선언과 초기화

- 3차원이상은 잘 사용하지 않는다.
- int[,,] arrNum = new int[4,3,2];
- int[,,] arrNum = new int[,,]{
  {{0,1},{2,3},{4,5}},{{6,7},{8,9},{10,11}},{{12,13},{14,15},{16,17}},{{18,19},{20,21},{22,23}};

가변 배열

- 배열의 크기가 가변적
- int[][]arrNum = new int[3][]; -> 행은 정해져 있어야 한다.
- arrNum[0] = new int[2]{0,1};
- arrNum[1] = new int[4]{0,1,2,3};
- arrNum[2] = new int[]{0,1,2};
- int[][]arrNum = new int[2][]{
  new int[]{0,1},new int[]{0,1,2}};
- 상황에 따라 필요한 경우도 있지만, 잘 사용하지 않는다.

함수의 파라미터(매개변수)로 배열이용

- Call by reference
- 리턴형 함수명(`int[]` 파라미터명)
- void Func(int[] arr)
- 리턴형 함수명(`int[,]` 파라미터명)
- void Func(int[,] arr)

```
class Program
    {
        static void SwapArray(int oriIndex, int desIndex, int[] array) {
            int temp = array[oriIndex];
            array[oriIndex] = array[desIndex];
            array[desIndex] = temp;
        }

        static void Main(string[] args) {
            int[] arrNum = new int[]{1, 2, 3, 4};

            foreach(int temp in arrNum)
                Console.WriteLine("  {0}", temp);

            SwapArray(0, 1, arrNum);

            Console.WriteLine("");

            foreach(int temp in arrNum)
                Console.WriteLine("  {0}", temp);

        }
    }
```

- SwapArray(0,1,arrNum) : arrNum의 0번째 요소와 1번째 요소를 바꾸겠다(swap)
- output : 2 1 3 4
- why? 레퍼런스(reference) 여서 서로 영향을 주기 때문이다.

```
namespace _073_Array_Func2
{
    class Program
    {
        static int[] CreateIntArray(int size) {
            int[] creArray = new int[size];

            for(int i = 0; i < creArray.Length; i++) {
                creArray[i] = 0;
            }

            return creArray;
        }

        static string[] CreateStrArray(int size) {
            string[] creArray = new string[size];

            for(int i = 0; i < creArray.Length; i++) {
                creArray[i] = string.Empty;
            }

            return creArray;
        }

        static int[,] CreateIntArray() {
            int[,] array = new int[3, 2];

            for(int i = 0; i < 3; i++) {
                for(int j = 0; j < 2; j++) {
                    array[i, j] = 0;
                }
            }

            return array;
        }

        static void Main(string[] args)
        {
            int[] arrNum = CreateIntArray(3);
            string[] strName = CreateStrArray(5);
            int[,] array = CreateIntArray();

            foreach(int temp in arrNum) {
                Console.Write("  {0}", temp);
            }

            Console.WriteLine("\n-------------------------");

            arrNum[0] = 1000;
            foreach(int temp in arrNum) {
                Console.Write("  {0}", temp);
```

```
            }

            Console.WriteLine("\n-------------------------");

            foreach(string s in strName) {
                Console.WriteLine("strName: {0}", s);
            }

            Console.WriteLine("-------------------------");

            strName[0] = "Hello World";
            strName[1] = "!!!!";

            foreach(string s in strName) {
                Console.WriteLine("strName: {0}", s);
            }

            Console.WriteLine("-------------------------");

            foreach(int data in array) {
                Console.WriteLine("data: {0}", data);
            }

            array[0, 0] = 10;
            array[2, 0] = 10;

            Console.WriteLine("-------------------------");

            foreach(int data in array) {
                Console.WriteLine("data: {0}", data);
            }
        }
    }
}
```

함수의 리턴으로 배열 이용

- Call by reference
- int[] 함수(int[] 파라미터명)
- int[] Func(int[] arr)
- 리턴형 함수명(int[,] 파라미터명)
- int[,] Func(int[] arr)

배열을 관리하는 방법

오버로딩되어있는 경우 많다. 암기하는 것이 아니라 필요할 때마다 microsoft reference참고하기

- public static void Clear(Array array, int index, int length);

    - ref 써주지 않아도 되는 이유 : 이미 Array가 레퍼런스형이기 때문
- public int Length{get;}
- public int GetLength(int dimension); = 파라미터인 dimension의 length를 int형으로 알려준다.
- public object Clone();
    - 리턴형 : object -> 어떤 변수도 가
    - 바로 쓸 수 없고, 캐스팅연산이 필요하다(자료형을 바꿔야 하니까)
    - object를 쓴 이유? 몇 차원의 배열인지 알 수 없기 때문 -> int[] int[,] int[, ,]
    - ex) arrA를 Clone한 arrB에서 아무리 값을 변경해도 arrA의 값이 바뀌는 것은 아니다. -> 서로 영향을 주지 않는다!(reference 아님)

```
  namespace _074_Array_Clear
{
    class Program
    {
        static void Main(string[] args) {
            int[] array = new int[3];

            Array.Clear(array, 0, array.Length);
            for(int i = 0; i < array.Length; i++) {
                array[i] = i;
            }

            for(int i = 0; i < array.Length; i++) {
                Console.Write("  {0}  ", array[i]);
            }

            Console.WriteLine("\n------------------------------");

            int[,] arrNum = new int[3, 2];

            Console.WriteLine("\n-------  Array.Clear  -------");
            Array.Clear(arrNum, 0, arrNum.Length);

            for(int i = 0; i < arrNum.GetLength(0); i++) {
                for(int j = 0; j < arrNum.GetLength(1); j++) {
                    arrNum[i, j] = (i * arrNum.GetLength(1)) + j;
                }
            }

            for(int i = 0; i < arrNum.GetLength(0); i++) {
                for(int j = 0; j < arrNum.GetLength(1); j++) {
                        Console.Write("  " + arrNum[i, j]);
                }
                    Console.WriteLine();
            }

            Console.WriteLine("------------------------------");

            int[,,] arrMulti1 = new int[,,]{
                {{0, 1}, {2, 3}, {4, 5}},
                {{6, 7}, {8, 9}, {10, 11}},
                {{12, 13}, {14, 15}, {16, 17}},
                {{18, 19}, {20, 21}, {22, 23}}
            };

            Console.WriteLine("arrMulti1.Length: " + arrMulti1.Length);

            for(int i = 0; i < arrMulti1.GetLength(0); i++) {
                for(int j = 0; j < arrMulti1.GetLength(1); j++) {
                    for(int k = 0; k < arrMulti1.GetLength(2); k++) {
                        Console.Write("  " + arrMulti1[i, j, k]);
```

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

        Console.WriteLine("-----------------------------");

        int[,,] cloneArray = (int[,,])arrMulti1.Clone();

        for(int i = 0; i < cloneArray.GetLength(0); i++) {
            for(int j = 0; j < cloneArray.GetLength(1); j++) {
                for(int k = 0; k < cloneArray.GetLength(2); k++) {
                    Console.Write("  " + cloneArray[i, j, k]);
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }

        Console.WriteLine("-----------------------------");
        cloneArray[0, 0, 0] = 10000;
        Console.Write("\n arrMulti1[0, 0, 0]:  {0}  ", arrMulti1[0, 0, 0]);
        Console.Write("\n cloneArray[0, 0, 0]:  {0}  ", cloneArray[0, 0, 0]);
    }
  }
}
```

주요코드정리

- Array라는 배열을 처음부터(0) 끝까지(array.Length) 깨끗하게 지우겠다(Clear)

```
Array.Clear(array, 0, array.Length);
```

- 지운(Clear) 후 값을 인덱스와 동일하게 넣음

```
for(int i = 0; i < array.Length; i++) {
    array[i] = i;
}
```

- arrNum.GetLength(0)==3, arrNum.GetLength(1)==2 -> for문을 돌면서 배열의 요소들에게 접근할 수 있다.

```
for(int i = 0; i < arrNum.GetLength(0); i++) {
    for(int j = 0; j < arrNum.GetLength(1); j++) {
        arrNum[i, j] = (i * arrNum.GetLength(1)) + j;
    }
}
```

- 3차원 배열 생성

```
int[,,] arrMulti1 = new int[,,]{
    {{0, 1}, {2, 3}, {4, 5}},
    {{6, 7}, {8, 9}, {10, 11}},
    {{12, 13}, {14, 15}, {16, 17}},
    {{18, 19}, {20, 21}, {22, 23}}
};
```

- for문을 통한 반복으로 3차원 배열 요소 조사

```
for(int i = 0; i < arrMulti1.GetLength(0); i++) {
                for(int j = 0; j < arrMulti1.GetLength(1); j++) {
                    for(int k = 0; k < arrMulti1.GetLength(2); k++) {
                        Console.Write("  " + arrMulti1[i, j, k]);
                    }
                    Console.WriteLine();
                }
                Console.WriteLine();
            }
```