C# 교실

CLASS 2
– C# Basic (Part2)

고주형
2019/05/03

C# Programming

Part 2

오늘 할 것

- Array (다차원 배열, 가변 배열)
- List
- Selection (if , switch, 관계, 논리)
- Loop (for, while, foreach)
- Jump (break, continue, goto)
- 실습!!

Array – 1차원 배열

- int[] myArray1 = new int[5];
- myArray1[1] = 1;
- int[] array1 = new int[] { 1, 3, 5, 7, 9 };

- string[] myArray2 = new string[5];
- string[] weekDays2 =

{ "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

Multi Array - 다차원 배열

- int[,] array = new int[4, 2];
- int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

- int[, ,] array1 = new int[4, 2, 3];
- int[, ,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } } };

Jagged Array(가변배열)

- 가변배열이란 길이가 가변 가능한 배열!
- 그래서 배열의 배열이라고도 부른다.

- 3x3 배열을 만들기 위해서는 어떻게 할까?

Jagged Array로 3x3 Array 만들기

- 3x3 가변 배열의 선언

```
int[][] jaggedArray = new int[3][];
jaggedArray[0] = new int[3];
jaggedArray[1] = new int[3];
jaggedArray[2] = new int[3];
```

Jagged Array - 길이를 바꿀 수 있다!

- 1. 차원마다 길이를 바꿀 수 있다!
- 가변 배열의 선언

```
int[][] jaggedArray = new int[3][];
jaggedArray[0] = new int[5];
jaggedArray[1] = new int[4];
jaggedArray[2] = new int[2];
```

Jagged Array

- 2. 배열의 선언 및 초기화

```
int[][] jaggedArray = new int[3][];
jaggedArray[0] = new int[] { 1, 3, 5, 7, 9 };
jaggedArray[1] = new int[] { 0, 2, 4, 6 };
jaggedArray[2] = new int[] { 11, 22 };
```

Jagged Array

- 3. 위랑 동일한 선언과 동시에 초기화 방식

```
int[][] jaggedArray2 = new int[][]
{
    new int[] { 1, 3, 5, 7, 9 },
    new int[] { 0, 2, 4, 6 },
    new int[] { 11, 22 }
};
```

Array의 한계

- 일반적인 배열의 크기는 정적이다!

- 크기를 예측할 수 없는 경우는 어떻게 할까?
- Ex) 현질을 해서 슬롯 추가할 수 있다.
  그 사람이 얼마나 쓸까? 예측할 수 있나…?

- 해결책: List를 사용한다!
- List는 크기를 동적으로 조절할 수 있다.

Array vs List

- **Array**
  일반적인 배열은 크기를 동적으로 조절할 수 없다.
  그렇기 때문에 크기가 어느정도 예상 가능해야 한다.

- **List**
  List는 원하는 만큼 추가하고 삭제할 수 있다.
  List는 크기를 미리 안정해도 된다.

List

- List 만드는 방법
- int를 저장하기 위한 리스트

```
List<int> list = new List<int>();
```

- List에 값 저장하기

```
list.Add(1);
list.Add(2);
list.Add(3);
list.Add(4);
```

List

- List안에 몇 개의 요소가 있을까?
  Count 속성을 이용!

- List안의 모두 지우고 싶으면?
  Clear() 메소드 이용!

- List의 해당 index에 있는 요소를 삭제하고 싶으면?
  RemoveAt(index)

Selection을 들어가가 전에

- 복합대입 연산자
  += -= *= /=
- 증감 연산자
  ++ --
- 논리 연산자
  && || !
- 관계 연산자
  > < >= <= != ==

복합대입 연산자

그냥 단축키!

- a += x
- a -= x
- a *= x
- a /= x

a = a + x
a = a - x
a = a * x
a = a / x

증감연산자

단축키 + α

- i++ : 후위 1 증가, 먼저 실행 후 증가
- i-- : 후위 1 감소
- ++i : 전위 1 증가, 먼저 증가 후 실행
- --i : 전위 1 감소

논리 연산자

- && AND
- || OR
- ! NOT

!연산자 써 보기: 밑에꺼 해석 가능한가요?

x의 배수 == if(!(number%x))

관계 연산자

- `>` : 크다
- `<` : 작다
- `>=` : 크거나 같다
- `<=` : 작거나 같다
- `==` : 같다
- `!=` : 다르다

조건문 - if

```
if (조건1){
      //조건1 이면 실행
}else if(조건2){
      //조건1 아니고 조건2이면 실행
}else{
      //조건1,2 모두 아니면 실행
}
```

조건문 - switch~case

```
if (조건1){
        //조건1 이면 실행
} else if(조건2){
        //조건1 아니고 조건2이면 실행
} else if(조건3){
        //…
} else if(조건4){
        //…
} else if(조건5){
        //…
}       .
        .
        .
```

너무 한 눈에 안들어와! ㅠㅠ

조건문 - switch~case

```
switch (변수)
{
        case 숫자1:             //변수가 해당 숫자일 경우 진입!!
                실행문;
                break;          //밑에 케이스들로 이동하면 안되니 break!
        case 숫자2:
                실행문;
                break;
        default:
                실행문;
                break;
}
```

★ 변수는 문자형 정수형 불린형등 다양하게 가능해요!

Loop - for

- For문

```
for(초기식;조건식;증감식){
    //실행할 명령어
}
```

- 초기식: 초기화를 진행한다.
- 조건식: 조건을 검사하여 true면 계속 반복
- 증감식: 변수를 증가 or 감소시킨다.

Loop - while

- while문: 조건이 true일 동안 반복한다.

```
while(조건문){
    //실행할 명령어
}
```

Loop - foreach

- foreach문: 배열이나 리스트의 요소를 반복하는 단순하고 깔끔한 방법

```
foreach(var 식별자 in 컬렉션){
    Console.Write(식별자);
}
```

- 배열이나 리스트의 각 요소를 하나씩 가져와서 foreach 루프 내의 블럭을 실행할 때마다 사용된다.

점프문

점프? 스킵하다, 뛰어들다

- break - 여기까지 그만 두고 나가!
  반복문에서 빠져나간다.
- continue - 여기까지 그만 두고 계속해!
  반복문에서 1 회만 건너뛰고 반복을 계속한다.
- goto - 여기로 가!
  특정 위치로 이동한다.

Jump - break

```
//Array arr[100]에서 target이 어디에 저장되어 있는지 찾고 싶다.
int target = 32; int where;
for(int a = 0; a<100; a++){
    if(arr[a] == target){
        where = a;
        break;
    }
}
```

Jump - continue

```
//Array 1~100에서 target만 빼고 더하고 싶다.
int target = 32; int sum=0;
for(int a = 1; a<=100; a++){
    if(a == target){
        continue;
    }
    sum += a;
}
```

Jump - goto

```
// Target이 나오면 메시지를 출력하고 프로그램을 종료하고 싶다.
int target = 23;
for (int a =0; a<100; a++)
{
    if(a == target) goto END_Pgm;
    Console.WriteLine("Current Number: " + a);
}

END_Pgm :
Console.WriteLine(">> Target detected.  WnEND Program");
```

실습

- Foreach를 사용해서 target을 찾아보자!
  (Input은 ReadLine()함수를 써서 받으면 된다.)

출처

- 마이크로소프트 공식 문서