프로그래밍 기초 with C#

03. 문자열과 입출력

학습목표

- string 타입의 객체를 다룰 수 있다.
- 콘솔 환경에서 입출력을 할 수 있다.

들어가며

프로그램에서 문자를 여러 개 모은 것을 문자열(String)이라고 한다. 문자열 데이터를 다루려면 string을 이용하면 된다.

이번 시간에는 문자열을 다루는 방법에 대해서 알아보도록 하자.

문자열 객체 생성

문자열 객체는 아래와 같이 생성한다.

```
// 정수, 실수 타입과는 다르게 기본값을 사용하고 싶다면
// default 리터럴 대신 string.Empty를 사용한다.
string message = string.Empty;
```

문자열 리터럴은 기본적으로 쌍따옴표(“”)로 감싸게 된다.

```
using System;

class Program
```

```
{
  static void Main()
  {
    string message = "Hello, World!";
    Console.WriteLine(message);
  }
}
```

특정 문자에 접근하기

문자열을 이루는 여러 문자 중 인덱서(Indexer)를 활용해 특정 문자에 접근할 수 있다. 아래를 보자.

```
string message = "Hello";

// 0부터 시작이라는 것에 주의하자
System.Console.WriteLine(message[0]); // H
System.Console.WriteLine(message[1]); // e
System.Console.WriteLine(message[2]); // l
System.Console.WriteLine(message[3]); // l
System.Console.WriteLine(message[4]); // o
```

이스케이프 시퀀스

소스 코드에서 몇몇 문자는 컴파일러가 해석하기 위해 사용하기 때문에 문자열에 그대로 사용할 수 없다. 예를 들어 아래를 보자.

```
// 프로젝트 경로를 저장하여 사용하려고 했다.
string myProjectPath = "C:\Project\Sandbox"; // 오류!
```

문자열 안에 역슬래시를 사용하려고 했으나, 컴파일러는 이를 다른 것으로 해석하기 때문에 우리가 원하는 대로 사용할 수 없었다. 우리가 의도한 대로 작성하고 싶다면 역슬래시(\)를 사용해 역슬래시를 그대로 사용하고 싶다고 컴파일러에게 알려줘야 한다. 이를 이스케이프 시퀀스(Escape Sequence)라고 한다.

```
// 이제는 경로를 올바르게 사용할 수 있다.
string myProjectPath = "C:\\Project\\Sandbox";
```

이스케이프 시퀀스를 일일이 적어주는 게 귀찮다면 축자 문자열 리터럴 (Verbatim String Literal)이라는 것도 이용할 수 있다.

```
// 축자 문자열 리터럴은 ""앞에 @을 붙인다.
// 더이상 \을 쓰지 않아도 된다.
string myProjectPath = @"C:\Project\Sandbox";
```

문자열 합치기

더하기 연산자로 문자열을 합칠 수 있다.

```
string firstName = "SeonMun";
string secondName = "Choi";
string fullName = firstName + secondName;
```

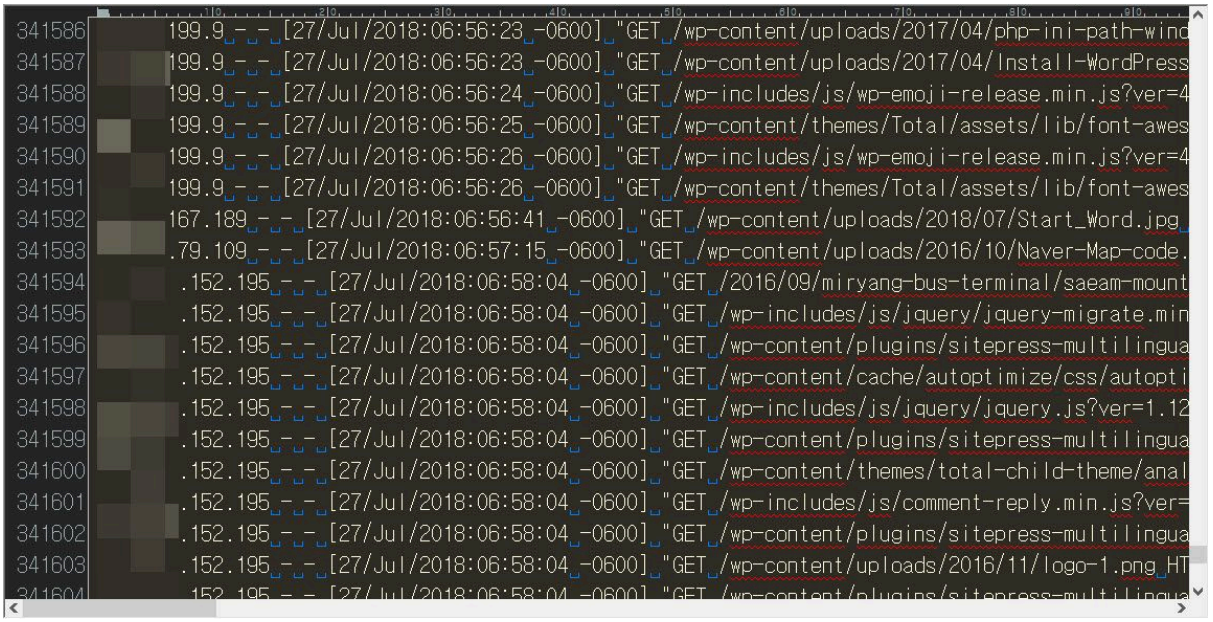
혹은 Concat()이라는 메소드를 사용할 수 있다.

```
string firstName = "SeonMun";
string secondName = "Choi";
string fullName = string.Concat(firstName, secondName);
```

Format()과 문자열 보간

문자열은 게임 내에서도 많이 사용되지만, 아래의 이미지처럼 로깅(Logging)*할 때도 사용된다.

* 컴퓨터 프로그램에서 어떤 동작이 일어났다는 데이터를 기록하는 것을 말한다.



로그를 보면 문자열이 일정한 형태를 가진다는 것을 알 수 있다. Format()은 이럴 때 유용하다. 아래는 기본적인 사용 방법이다.

```
// {} 안의 숫자는 넣어줄 데이터의 순서를 의미한다.
System.Console.WriteLine("{0} {1}!", "Hello", "World");

// 아래처럼도 사용 가능하다.
// 꼭 번호를 순서대로 넣어야 하는 것은 아니다.
System.Console.WriteLine("{1} {0}!", "Hello", "World");

// 같은 번호를 사용해도 된다.
System.Console.WriteLine("{0} {0} {0}!", "Hello");
```

콤마(,)를 이용해 출력할 문자열의 최소 길이를 지정할 수 있다.

```
// "Hello"는 실제 5의 길이지만 공백을 추가하여 12만큼의 길이로 출력하게 된다.
System.Console.WriteLine("{0, 12}", "Hello");
System.Console.WriteLine("{0, 11}", "Hello");
System.Console.WriteLine("{0, 10}", "Hello");
System.Console.WriteLine("{0, 9}", "Hello");
System.Console.WriteLine("{0, 8}", "Hello");
System.Console.WriteLine("{0, 7}", "Hello");
System.Console.WriteLine("{0, 6}", "Hello");
```

또, 정렬을 지정해줄 수 있는데 양수를 기입할 때는 오른쪽 정렬이다. 만일 왼쪽 정렬로 문자열을 출력하고 싶다면 음수를 기입하면 된다.

```
// 음수를 기입한 곳은 왼쪽 정렬로, 양수를 기입한 곳은 오른쪽 정렬로 표현된다는 것을 알 수 있다.
System.Console.WriteLine("{0, -12}", "Hello");
System.Console.WriteLine("{0, 12}", "Hello");
```

콜론(:)을 붙여 같은 데이터를 서로 다른 형식으로 출력해줄 수 있다.

```
int someValue = -27;
System.Console.WriteLine("{0, 10:G}, {0, 10:X}", someValue);

double someValue2 = 1234.5678;
System.Console.WriteLine("{0, 10:F} {0, 10:E2}", someValue2);
```

위의 예시는 숫자 데이터에 대해서만 진행한 것으로 이 외에도 여러 타입을 지원한다. 여기를 참고하자.

한편, 꼭 Format()을 써야하는 것은 아니다. 문자열 보간(String Interpolation)을 사용해 좀 더 편하게 만들 수 있다. Format()보다 훨씬 유연하고, 가독성이 높기 때문에 요즘은 아래처럼 쓰는 걸 권장하고 있다.

```
// 문자열 보간을 사용하고 싶다면 ""앞에 $를 붙여주도록 하자.
int someValue = -27;
System.Console.WriteLine($"{someValue, 10:G}, {someValue, 10:X}");
```

공백 제거하기

Trim() / TrimStart() / TrimEnd() 을 이용해 문자열의 앞 뒤 공백을 제거할 수 있다.

```
string someStringWithSpace = "     Hello World!      ";

// Trim()은 문자열 앞 뒤의 공백을 삭제한다.
System.Console.WriteLine(someStringWithSpace.Trim());

// TrimStart()는 문자열 앞의 공백을 삭제한다.
System.Console.WriteLine(someStringWithSpace.TrimStart());

// TrimEnd()는 문자열 뒤의 공백을 삭제한다.
System.Console.WriteLine(someStringWithSpace.TrimEnd());
```

대체하기

Replace()를 사용하면 특정 문자열을 다른 문자열로 대체할 수 있다.

```
string someString = "Hello World!";
System.Console.WriteLine(someString.Replace("World", "Programming"));
```

대소문자 변경하기

ToLower()나 ToUpper()를 이용하면 문자열의 모든 문자를 소문자로 변경하거나 대문자로 변경할 수 있다.

```
string someString = "Hello World!";
```

```
System.Console.WriteLine(someString.ToLower());
System.Console.WriteLine(someString.ToUpper());
```

문자열의 길이 구하기

Length를 이용하면 문자열의 길이를 알 수 있다.

```
string message = "Hello";
int messageLength = message.Length;
```

부분 문자열 추출하기

Substring()을 사용해 문자열에서 부분 문자열을 추출할 수 있다.

```
string message = "Hello World";
System.Console.WriteLine(message.Substring(3)); // 3번 인덱스부터 끝까지를 추출한다.
System.Console.WriteLine(message.Substring(5, 3)); // 5번 인덱스부터 3개의 문자를 추출한다.
```

콘솔 환경에서의 입력과 출력

문자열 객체를 다룰 수 있게 되었다면 콘솔 환경에서의 입력과 출력을 자유롭게 할 준비가 된 것이다. 콘솔 환경에서의 입력과 출력하는 것은 어렵지 않다. 각각 ReadLine()과 WriteLine()을 사용하면 된다.

```
string input = System.Console.ReadLine();
System.Console.WriteLine($"입력된 데이터 : {input}");
```

보면 알겠지만, 입력된 데이터의 타입은 문자열이다. 즉, 문자가 아닌 숫자 같은 데이터를 입력한다면 타입 변환이 필요하다.

비호환 타입의 변환

지난 시간에 암시적 변환과 명시적 변환에 대해서 배웠고, 막무가내로 아무 타입으로나 변환이 불가능한 것에 대해서도 배웠다. 둘 모두 컴파일 오류가 일어나는 경우라면 BitConverter 클래스나 Convert 클래스를 사용하면 된다. 특히, 문자열에서 변환하는 경우라면 각 타입에 내장되어 있는 메소드인 Parse()를 사용할 수 있다.

```
string input = System.Console.ReadLine();
int number = int.Parse(input);
```

더 나아가기

1. 인덱서를 이용해 특정 문자에 접근할 수 있었습니다. 인덱서를 사용할 때는 유효한 범위의 숫자를 넣어야 합니다. 위의 예제에서 음수나 5 이상의 숫자를 넣었을 때 어떤 일이 발생하는지 확인해봅시다.*

   * 이때 발생하는 오류를 **런타임 오류**(Runtime Error)라고 한다.

2. 아래 문자열의 유효한 인덱스의 범위는 무엇일까요?
   a. “Hello World!”
   b. “Programming”
3. `string message = "C-Sharp"`에서 `p`라는 문자를 추출하기 위해 사용해야 하는 인덱스 번호를 써 보세요.
4. 문자열 합치기 방법 중 더하기 연산자(+)와 `string.Concat()` 메소드의 차이점을 간략히 설명해 보세요.
5. Format()을 사용할 때도 숫자를 기입하는 것에 주의해야 합니다. 유효하지 않은 범위의 숫자를 넣어보고 어떤 일이 발생하는지 확인해 봅시다.
6. 문자열 형식 지정 시, 출력되는 문자열의 최소 길이를 15로 지정하고, 내용을 오른쪽 정렬로 표현하기 위해 `Format()` 혹은 문자열 보간에서 콤마(`,`) 뒤에 작성해야 할 값을 써 보세요.
7. 아래의 경우 어떤 일이 발생하는지 확인해 봅시다.

   ```
   int number = int.Parse("abc");
   ```

8. LLM으로 string의 유용한 메소드와 프로퍼티를 정리해 보세요.
9. 경로 문자열 `"C:\Project\Test"`를 C# 코드에서 오류 없이 선언하는 **두 가지 방법**을 코드로 작성해 보세요.
10. 입출력은 흔히 프로그램에서 병목이 되는데 왜 그럴까요? 이유를 기술하고 여기를 참고해서 병목을 완화하는 방법도 적어보세요.
11. 콘솔에서 사용자가 "456"을 입력했을 때, 이 값을 정수 타입 `number` 변수에 저장하는 코드를 `System.Console.ReadLine()`과 `int.Parse()`를 사용하여 작성해 보세요.

12. 백준 온라인 저지에서 아래의 문제를 풀어봅시다.
    a. 2557번: Hello World
    b. 10171번: 고양이
    c. 10172번: 개
    d. 25083번: 새싹
13. 프로그래머스에서 아래의 문제를 풀어봅시다.
    a. 코딩테스트 연습 - 특정 문자 제거하기 | 프로그래머스 스쿨
    b. 코딩테스트 연습 - 편지 | 프로그래머스 스쿨
    c. 코딩테스트 연습 - 모음 제거 | 프로그래머스 스쿨
    d. 코딩테스트 연습 - 인덱스 바꾸기 | 프로그래머스 스쿨

참고자료

- C# 교과서