3) 변수와 자료형

매 강의 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

- C#에서 사용되는 기본 자료형에 대해 이해
- 변수를 선언하고 초기화하는 방법을 이해
- var 키워드를 사용하여 변수를 선언하는 방법을 이해
- 명시적 및 암시적 형변환을 사용하는 방법을 이해

[목차]

모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

01. 기본 자료형 소개

✔️ 자료형에 대해 알아보고 변수와 친해져 봅시다!

▼ 1) 자료형 (Data Type)

C#에서 사용되는 기본 자료형은 다음과 같습니다.

자료형	.NET 데이타 타입	크기 (바이트)	범위
sbyte	System.SByte	1	-128 ~ 127
byte	System.Byte	1	0 ~ 255
short	System.Int16	2	-32,768 ~ 32,767
ushort	System.UInt16	2	0 ~ 65,535
int	System.Int32	4	-2,147,483,648 ~ 2,147,483,647
uint	System.UInt32	4	0 ~ 4,294,967,295
long	System.Int64	8	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
ulong	System.UInt64	8	0 ~ 18,446,744,073,709,551,615
float	System.Single	4	±1.5 × 10^-45 ~ ±3.4 × 10^38
double	System.Double	8	±5.0 × 10^-324 ~ ±1.7 × 10^308
decimal	System.Decimal	16	±1.0 × 10^-28 ~ ±7.9 × 10^28
char	System.Char	2	유니코드 문자
string	System.String		유니코드 문자열
bool	System.Boolean	1	true 또는 false

이러한 기본 자료형을 사용하여 변수를 선언하고 값을 저장할 수 있습니다.

- 자료형의 크기, 표현 범위, 부호 여부 등의 특성에 따라 세분화된 자료형으로 구분됩니다.
- 예를 들어, C#에서 정수형 자료형으로는 sbyte, byte, short, ushort, int, uint, long, ulong이 있으며, 각각의 자료형은 **메모리의 크기와 표현 범위**가 다릅니다.

```
유저의 번호 저장한다면 127명 이하일때는 sbyte 자료형을 사용 가능
하지만, 128명 부터는 저장불가!
이때 우리가 음수의 번호를 사용하지 않는다면 byte 자료형으로 선언 함으로써 255번까지 저장 가능
```

다음은 변수를 세분화 해서 사용하는 이유

1. **메모리의 효율적인 사용**
   세분화된 자료형을 사용하면, 해당 자료형이 필요한 크기만큼의 메모리를 할당하여 메모리의 효율적인 사용을 가능하게 합니다.
2. **정확한 데이터 표현**
   세분화된 자료형을 사용하면, 데이터의 특성에 따라 정확한 표현이 가능합니다. 예를 들어, 부동소수점 자료형인 float과 double은 소수점 이하 자릿수가 다르며, 각각의 자료형은 다른 범위의 값까지 표현이 가능합니다.
3. **타입 안정성**
   세분화된 자료형을 사용하면, 코드의 타입 안정성을 유지할 수 있습니다. 예를 들어, 정수형 자료형으로 byte를 사용하면, 해당 자료형이 가질 수 있는 값의 범위를 벗어날 경우 오류가 발생하므로, 코드의 안정성을 보장할 수 있습니다.

▼ 2) 리터럴 (literal)

1. **리터럴의 개념과 역할**
   a. 프로그램에서 직접 사용되는 상수 값으로, 소스 코드에 직접 기록되어 있는 값
   b. C#에서 리터럴은 컴파일러에 의해 상수 값으로 처리되며, 변수나 상수에 할당되거나 연산에 사용
2. **리터럴의 종류와 예시**
   a. C#에서는 다양한 종류의 리터럴을 지원
   b. 각각의 리터럴은 다른 형식으로 표현되며, 다양한 값의 범위를 가지고 있습니다.

   아래는 C#에서 지원하는 리터럴의 종류와 예시입니다.

   - 정수형 리터럴
     - 10 (int)

    - 0x10 (16진수 int)
    - 0b10 (2진수 int)
    - 10L (long)
    - 10UL (unsigned long)
- 부동소수점형 리터럴
    - 3.14 (double)
    - 3.14f (float)
    - 3.14m (decimal)
- 문자형 리터럴
    - 'A' (char)
    - '\n' (개행 문자)
    - '\u0022' (유니코드 문자)
- 문자열 리터럴
    - "Hello, World!" (string)
    - "문자열 내 "따옴표" 사용하기"
    - @"문자열 내 개행 문자
      사용하기"

```
int num = 10;
float f = 3.14f;
char c = 'A';
string str = "Hello, World!";

int num1 = 0x10;
int num2 = 0b1010;
long num3 = 100000000000000L;
```

02. 변수 선언과 초기화 방법

✔️ 변수명를 만들어보고, 그 결과를 봅시다!

▼ 1) 변수란?

- 데이터(숫자, 문자 등) 를 저장하고 사용하기 위한 할당받은 공간
- 필요에 따라 데이터를 저장하거나 수정 가능
- 예를 들어 게임을 제작할때 플레이어의 재화나 공격력, HP 같은 정보를 저장하기 위해 사용

▼ 2) 변수 선언

C#에서 변수를 선언하는 방법은 다음과 같습니다.

```
자료형 변수이름;
```

예를 들어, int 자료형의 변수를 선언하는 방법은 다음과 같습니다.

```
int num;
```

다음은 한번에 변수 여러개를 선언하는 방법의 예시입니다.

```
int num1, num2, num3;
```

▼ 3) 변수 초기화

변수를 선언한 후에는 변수를 초기화하여 값을 저장해야 합니다. 변수를 초기화하는 방법은 다음과 같습니다.

```
변수이름 = 값;
```

예를 들어, 다음과 같이 변수를 선언하고 초기화할 수 있습니다.

```
int num;     // 변수 선언
num = 10;    // 변수 초기화
```

변수를 선언과 동시에 초기화할 수도 있습니다.

```
int num = 10;   // 변수 선언과 초기화를 한 번에 수행
```

한번에 여러개의 변수를 초기화 할 수도 있습니다.

```
int num1, num2, num3 = 10; (X)

num1 = num2 = num3 = 10;
```

03. 변수명

✔️ 아래의 규칙들을 지켜서 변수명을 만들어 주세요

▼ 1) 키워드 (Keywords)

- C#에서는 이미 예약된 단어들이 있기 때문에 변수, 메소드, 클래스 등의 이름으로 사용할 수 없다.
- 이러한 단어를 키워드(Keywords)라고 한다
- 다음과 같은 키워드들이 있습니다.

```
abstract as base bool break byte case catch char checked class const c
ontinue decimal default delegate do double else enum event explicit exter
n false finally fixed float for foreach goto if implicit in int interfa
ce internal is lock long namespace new null object operator out override
params private protected public readonly ref return sbyte sealed short si
zeof stackalloc static string struct switch this throw true try typeof u
int ulong unchecked unsafe ushort using virtual void volatile while
```

▼ 2) 식별자 (Identifiers)

식별자란 변수, 메서드, 클래스, 인터페이스 등에 사용되는 이름을 말합니다. 이 이름은 키워드와 동일하게 사용할 수 없습니다.

식별자를 사용할 때에는 다음과 같은 규칙을 따라야 합니다.

- 첫 문자는 알파벳, 언더스코어(_)가 올 수 있습니다.
- 두번째 문자부터는 알파벳, 언더스코어, 숫자가 올 수 있습니다.
- 대소문자를 구분합니다.
- 키워드와 같은 이름으로 사용할 수 없습니다.

```
// 좋은 예시
int playerScore;
string playerName;
float itemPrice;

// 나쁜 예시 (중요 의미 있는 변수명 짓기)
int x1;  // 변수명이 의미를 알기 어려움
string a; // 변수명이 명확하지 않음

// 오류 예시
int 1stNumber;  // 변수명은 숫자로 시작할 수 없음
string my-name; // 변수명에 하이픈(-)을 사용할 수 없음
float total$;   // 변수명에 특수문자($)를 사용할 수 없음
```

▼ 3) 코드 컨벤션(Code convention)

코드 컨벤션(Code convention)은 개발자들 사이에서 약속된 코드 작성 규칙으로, 코드의 가독성을 높이고 유지 보수를 쉽게 하기 위해 사용됩니다. 코드 컨벤션은 프로그래밍 언어마다 다를 수 있으며, 우리는 다음의 규칙을 따를 예정입니다.

1. 식별자 표기법
    - PascalCase: 클래스, 메서드, 프로퍼티 이름 등에 사용됩니다. 단어의 첫 글자는 대문자로 시작하며, 이후 단어의 첫 글자도 대문자로 표기합니다. 예를 들어, `ClassName`, `MethodName`, `PropertyName`과 같은 형태입니다.
    - camelCase: 변수, 매개변수, 로컬 변수 이름 등에 사용됩니다. 단어의 첫 글자는 소문자로 시작하며, 이후 단어의 첫 글자는 대문자로 표기합니다. 예를 들어, `variableName`, `parameterName`, `localVariableName`과 같은 형태입니다.
    - 대문자 약어: 예외적으로 전체 글자가 모두 대문자인 식별자도 있습니다. 대표적으로 `ID`, `HTTP`, `FTP` 등이 있습니다.

2. 들여쓰기
    - 탭(tab) 또는 스페이스(space) 4칸을 사용하여 코드 블록을 들여씁니다.
3. 중괄호 위치
    - 중괄호({})는 항상 새로운 줄에서 시작합니다.
4. 빈 줄 사용
    - 관련 없는 코드 사이에는 빈 줄을 사용하여 구분합니다.
    - 메서드, 클래스 등의 블록 사이에는 두 줄을 띄어씁니다.
5. 주석
    - // 한 줄 주석을 사용합니다.
    - /* / *여러 줄 주석을 사용할 때는* / 를 다음 줄에서 시작하고, */ 를 새로운 줄에서 끝내도록 합니다.

예를 들어, 다음과 같이 C# 코드를 작성하는 것이 일반적인 코드 컨벤션입니다.

```
class MyClass
{
    // 필드는 camelCase 표기법을 사용합니다.
    private int myField;

    // 프로퍼티는 PascalCase 표기법을 사용합니다.
    public int MyProperty { get; set; }

    // 메서드는 PascalCase 표기법을 사용합니다.
    public void MyMethod()
    {
        if (true)
        {
            // 중괄호는 새로운 줄에서 시작합니다.
        }

        // 코드 블록은 탭(tab) 또는 스페이스(space) 4칸으로 들여씁니다.

        // 관련 없는 코드 사이에는 빈 줄을 사용하여 구분합니다.
        // 블록 사이에는 두 줄을 띄어씁니다.

        /*
        여러 줄 주석을 사용할 때는
         / * 를 새로운 줄에서 시작하고,
         * / 를 새로운 줄에서 끝내도록 합니다.
        */

        // 한 줄 주석은 이렇게 사용합니다.
        int a = 10; // 코드 끝에도 한 줄 주석을 사용할 수 있습니다.
    }
}
```

코드 컨벤션을 지키면 코드의 가독성이 좋아지고 유지보수가 용이해집니다.
따라서 C# 개발을 할 때는 코드 컨벤션을 준수하는 것이 좋습니다.

Microsoft C# 코딩 규칙

04. 명시적 및 암시적 형변환

✔️ 명시적 형변환과 암시적 형변환을 사용해보고, 그 차이를 봅시다!

▼ 1) 형변환 이란?

C#에서는 자료형이 다른 변수 간에 값을 할당하거나 연산을 수행할 수 있습니다. 이때, 자료형이 다른 변수 간에 값을 할당하거나 연산을 수행하려면 명시적 형변환(explicit casting) 또는 암시적 형변환(implicit casting)을 해주어야 합니다.

▼ 2) 명시적 형변환

명시적 형변환은 다음과 같이 `(자료형)` 형식으로 수행할 수 있습니다.

```
int num1 = 10;
long num2 = (long)num1;   // int를 long으로 명시적 형변환
```

▼ 3) 암시적 형변환

1. 작은 데이터 타입에서 더 큰 데이터 타입으로 대입하는 경우
    - byte, short, char 등 작은 데이터 타입에서 int, long, float 등 더 큰 데이터 타입으로 대입할 때 암시적 형변환이 발생합니다.

```
byte num1 = 10;
int num2 = num1;  // byte형에서 int형으로 암시적 형변환
```

2. 리터럴 값이 대입되는 경우
    - C# 컴파일러는 리터럴 값의 데이터 타입을 판별하여 변수에 암시적으로 형변환합니다.

```
float result = 1;  // 1은 int형 리터럴 값이지만 float형으로 암시적 형변환
```

3. 정수형과 부동소수점형 간의 연산을 수행하는 경우
    - 정수형과 부동소수점형의 연산 결과는 부동소수점형으로 변환됩니다.

```
int num1 = 10;
float num2 = 3.14f;
float result = num1 + num2;  // int형과 float형의 덧셈에서 float형으로 암시적 형변환
```

- 암시적 형변환은 프로그래머가 직접 형변환 코드를 작성하지 않아도 되므로 코드를 간결하게 작성할 수 있습니다.
- 하지만, 암시적 형변환이 발생하는 경우 데이터 타입을 신중하게 고려하여 코드를 작성해야 합니다.

05. Console.ReadLine을 이용한 입력

✔️ 입력을 받기 위해 `Console.ReadLine`을 사용해봅시다!

▼ 1) Console.ReadLine

C#에서 콘솔 입력을 받을 때는 `Console.ReadLine` 메소드를 사용합니다. `ReadLine` 메소드는 사용자가 입력한 값을 문자열로 반환합니다.

`Console.ReadLine` 메소드는 다음과 같은 형식으로 사용할 수 있습니다.

```
string input = Console.ReadLine();
```

`input`은 사용자가 입력한 값을 저장하는 변수입니다. `Console.ReadLine` 메소드를 호출하면 사용자가 입력한 값이 문자열로 반환되어 `input` 변수에 저장됩니다.

▼ **[코드스니펫] 입력 기초**

```
Console.Write("Enter your name: ");
string name = Console.ReadLine();
Console.WriteLine("Hello, {0}!", name);
```

예를 들어, 다음과 같이 `Console.ReadLine` 메소드를 사용하여 값을 입력받을 수 있습니다.

```
Console.Write("Enter your name: ");
string name = Console.ReadLine();
Console.WriteLine("Hello, {0}!", name);

[출력]
Enter your name: Kero
Hello, Kero!
```

`Console.ReadLine` 메소드는 사용자가 입력한 값을 문자열로 반환하므로, 숫자나 논리값을 입력받을 때에는 적절한 형변환을 해주어야 합니다.

▼ 2) Split / 한줄에 여러 값 입력받기

사용자로부터 여러 개의 값을 한 줄에 입력받고 싶을 때에는 `Console.ReadLine` 메소드를 사용하여 입력받은 값을 문자열로 받은 후, `string.Split` 메소드를 사용하여 문자열을 나누어서 처리할 수 있습니다.

예를 들어, 사용자로부터 공백으로 구분된 두 개의 정수를 입력받아 더하는 코드를 작성해보겠습니다.

▼ **[코드스니펫] 다중 입력 기초**

```
Console.Write("Enter two numbers: ");
string input = Console.ReadLine();    // "10 20"과 같은 문자열을 입력받음

string[] numbers = input.Split(' ');  // 문자열을 공백으로 구분하여 배열로 만듦
int num1 = int.Parse(numbers[0]);     // 첫 번째 값을 정수로 변환하여 저장
int num2 = int.Parse(numbers[1]);     // 두 번째 값을 정수로 변환하여 저장

int sum = num1 + num2;                // 두 수를 더하여 결과를 계산

Console.WriteLine("The sum of {0} and {1} is {2}.", num1, num2, sum);
```

```
Console.Write("Enter two numbers: ");
string input = Console.ReadLine();    // "10 20"과 같은 문자열을 입력받음

string[] numbers = input.Split(' ');  // 문자열을 공백으로 구분하여 배열로 만듦
int num1 = int.Parse(numbers[0]);     // 첫 번째 값을 정수로 변환하여 저장
int num2 = int.Parse(numbers[1]);     // 두 번째 값을 정수로 변환하여 저장

int sum = num1 + num2;                // 두 수를 더하여 결과를 계산

Console.WriteLine("The sum of {0} and {1} is {2}.", num1, num2, sum);

[출력]
Enter two numbers: 10 20
The sum of 10 and 20 is 30.
```

이와 같이 `Console.ReadLine` 메소드를 사용하여 한 줄에 여러 개의 값을 입력받아 처리할 수 있습니다.

06. var 키워드 사용법

✔️ C#의 마법같은 키워드 'var'를 써보자

▼ 1) var 키워드

- C# 3.0부터는 var 키워드를 사용하여 변수를 선언할 수 있습니다.
- var 키워드를 사용하여 변수를 선언하면 변수의 자료형이 컴파일러에 의해 자동으로 결정됩니다.
- 즉, 초기화하는 값의 자료형에 따라 변수의 자료형이 결정됩니다.

예를 들어, 다음과 같이 var 키워드를 사용하여 변수를 선언할 수 있습니다.

```
var num = 10;          // int 자료형으로 결정됨
var name = "kero";   // string 자료형으로 결정됨
var pi = 3.141592;    // double 자료형으로 결정됨
```

- var 키워드를 사용하여 변수를 선언할 때에는 초기화하는 값의 자료형에 따라 변수의 자료형이 결정되므로, 변수를 선언하는 시점에서 변수의 자료형을 정확히 알 수

없는 경우에 유용하게 사용할 수 있습니다.

Tip

- 변수의 이름을 명확하게 지어주는 것이 코드의 가독성을 높이는 데에 큰 도움이 됩니다. 변수의 이름은 변수가 어떤 값을 저장하고 있는지 명확하게 드러나도록 지어주어야 합니다.
- 변수의 이름이 너무 짧거나 약어로 되어 있으면 코드를 읽는 사람이 변수가 어떤 값을 저장하고 있는지 파악하기 어려울 수 있습니다.

이전 강의

📕 2) 프로그래밍 기본 요소

다음 강의

4강 연산자와 문자열 처리