C# 체크리스트

변수와 자료형

송지원 튜터(내일배움캠프 Unity 트랙)

체크리스트 주요 대상

C# 체크리스트를 진행하는 목적은 아래와 같습니다.

- 프로그래밍을 처음 배우시는 분🐤들의 이해를 도와드리는 것
- 프로그래밍 강의를 들어도 잘 모르겠는🤷‍♀️ 부분들 이해

심화된 내용에 대한 질문은 튜터링을 통한 해결을 추천해요!

← 처음 배우는 분들이 혼란스러워 해요..😅

체크리스트 진행 방식

- 매일매일 튜터들이 돌아가면서 1시간 내로 진행할 예정

- 7일동안 각 파트에 맞는 강의 짧게 진행 + 문제 풀이 진행 예정

C#의 특징

- 마이크로소프트에서 개발한 객체 지향 프로그래밍 언어.

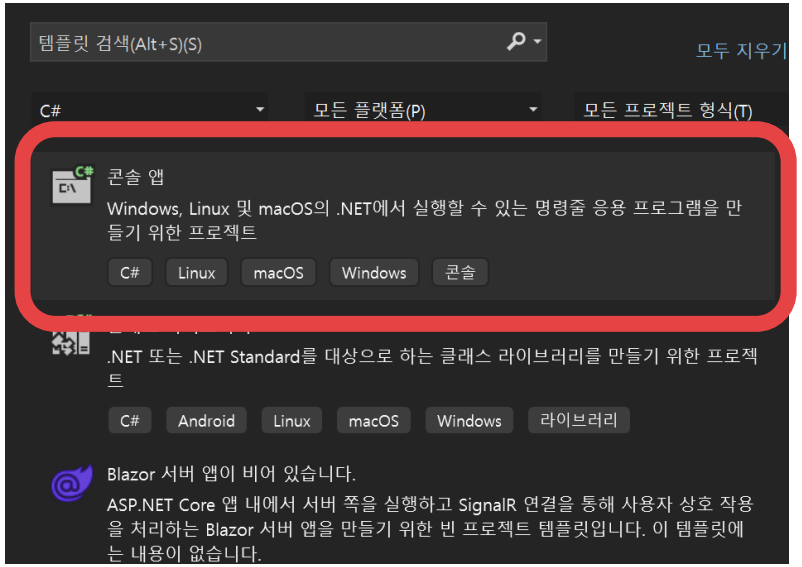
- C계열 개발자들에게 친숙함(문법, 강타입 언어 특성)

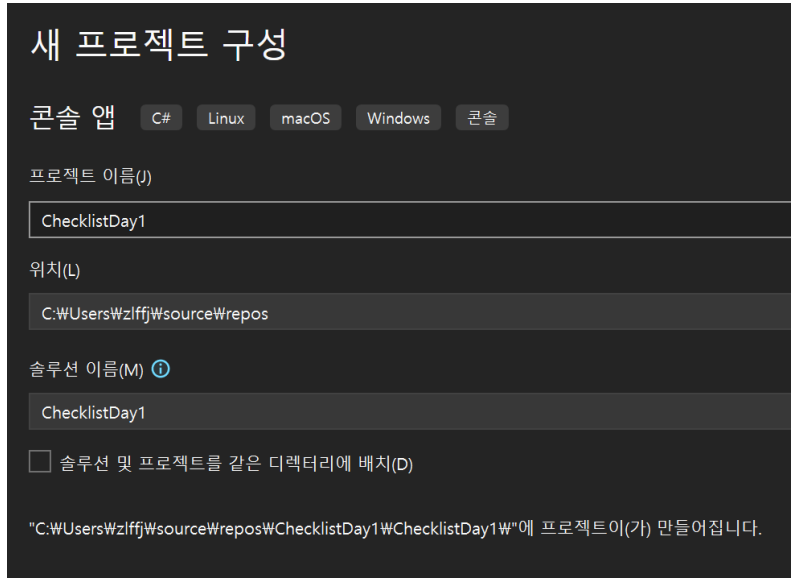
- 유니티에서 스크립팅에 C#을 활용함

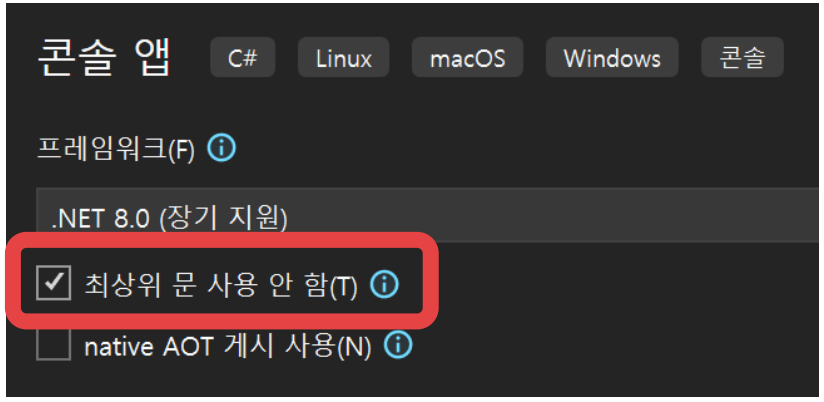
C #+

Visual Studio에서 프로젝트 시작하기

- 아래 이미지와 같이 프로젝트 생성
- 최상위문 기능은 강의에서처럼 사용하지 않음.



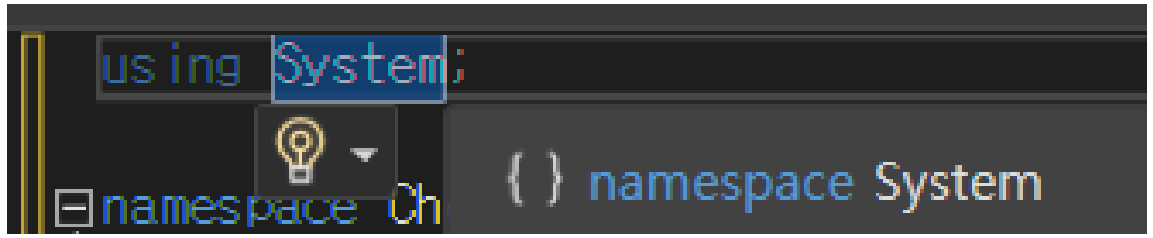


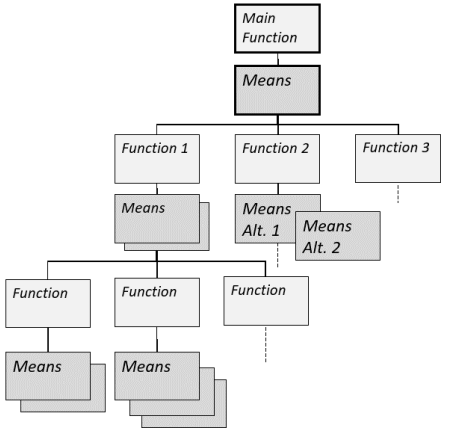


C# 프로그램 기본 구조

- Main 함수에서 시작해서 다른 기능들을 호출하면서 프로그램 실행

- 다른 사람이 만든 모듈을 끌고 오려면 네임스페이스를 추가해야 함 (using문)





C# 프로그램 기본 용어 정리

- **변수** 프로그램에서 데이터를 저장하는 메모리 공간에 대한 이름

```
int score = 100;      // 정수형 변수
float speed = 2.5f;   // 실수형 변수
string playerName = "John";  // 문자열 변수
bool isAlive = true; // 논리형 변수
```

- **함수** 특정한 작업을 수행하는 코드의 묶음.

```
public int Add(int a, int b) {
    return a + b;
}
```

- **필드** 클래스 내에 정의된 변수
- **메소드** 클래스 내에 정의된 함수

Q : 필드나 메소드가 클래스 밖에 있을 수도 있나요?
A : 거의 클래스안에 있기 때문에 함수/메소드처럼 혼용하기도 합니다.

C# 프로그램 기본 용어 정리

- **클래스** 다양한 필드와 메소드를 묶은 객체

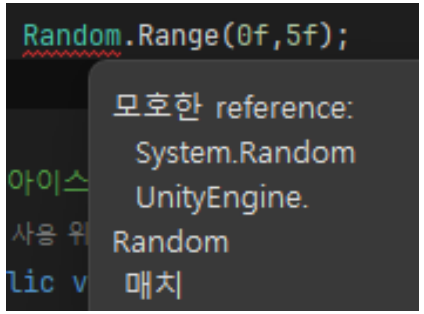
```
public class Player {
    public string name; // 필드
    public int health;  // 필드

    public void Move() { // 메소드
        // 이동 로직
    }
}
```

- **네임스페이스** 클래스 등을 담는 일종의 폴더 개념

  UnityEngine.Random / System.Random 사례

```
using UnityEngine;
using System;
```




```
UnityEngine.Random.Range(0f,5f);
```

주석

- **주석** 프로그램에서 실행되지 않는 설명이나 메모

단일행 주석 VS 다중행 주석

```
// 이 함수는 두 수를 더한 값을 반환합니다.
public int Add(int a, int b) {
    return a + b;
}

public void ExampleFunction() {
    /*
     이 블록 주석은 여러 줄에 걸쳐 작성할 수 있습니다.
     함수의 역할을 설명하거나, 구현 방법을 기록할 때 유용합니다.
    */
    Console.WriteLine("Hello!");
}
```

유의해야 할 주석 사례

```
// 이 함수는 첫 번째 값과 두 번째 값을 더합니다.
private int ChooseGreater(int a, int b){
  return a > b ? a : b;
}
```

변수

유저에게 몬스터의 HP값을 보여주고 싶다고 생각해봅시다.

일단.. HP를 저장해두어야하지 않을까요..?

보여주거나 처리하려면 일단 저장되어 있어야겠죠

다양한 변수

그런데, 변수가 다 똑같은 변수가 아닙니다.

캐릭터의 좌표

포탈 위에 있는지 여부

아이템 보유 개수

데미지 강화 비율

스킬 시전 시 나오는 멘트

광고 시청 횟수

변수에는 **'타입'**이 있다라고 생각해볼 수 있습니다.

변수와 자료형

변수 프로그램에서 데이터를 저장하는 메모리 공간에 대한 이름

자료형 변수나 값이 가질 수 있는 데이터의 종류. 어떻게 데이터가 저장되고 처리할 지를 결정

각각의 변수에는 그에 맞는 자료형이 적용되어야 한다는 것이 중요하며,

기본적인 자료형은 정수형(소수점 X), 실수형(소수점 O), 문자형, 논리형이 있습니다.

정수형 int

int 타입은 대표적인 정수형 타입 자료형입니다.
int 타입은 32개의 비트를 사용해서 데이터를 저장합니다.

[QUIZ] 32개의 비트를 사용하면 int가 저장할 수 있는 가장 큰 값은?
힌트 : 정상화되어 버린 구 풀메소..

정수형 int 써보기

변수를 정의하기 위해서는 아래와 같이 작성합니다.

[타입] [변수명]

ex) int count;

변수에 값을 적용하는 것을 대입 혹은 할당한다라고 하며, 처음 값을 대입하는 것을 초기화라고 합니다. 이때 =(대입 연산자)를 활용합니다.

ex) count = 0;

정의와 함께 초기화하는 문법도 있습니다.

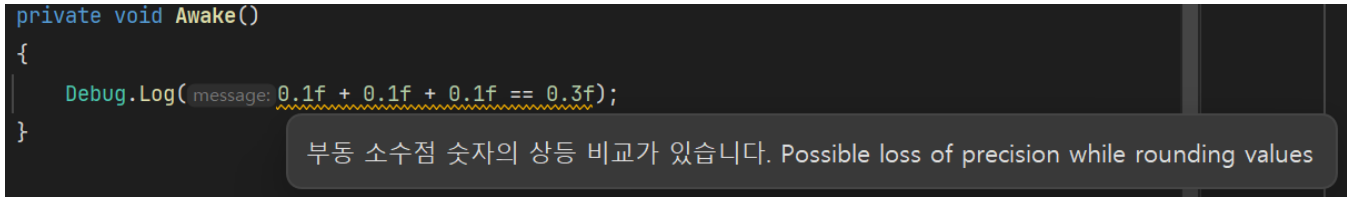
ex) int count = 0;

실수형 float

float 타입은 대표적인 실수형 타입입니다.
실수형이라는 것은 소수점이 있다는 의미입니다!

매우 큰 수를 저장하기 위해 지수를 통해 데이터를 저장합니다.
덕분에, int와 같이 32비트임에도 훨씬 넓은 범위를 표현할 수 있어요!

대신, 정밀도 문제가 발생합니다.



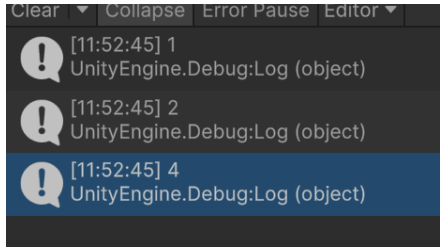
0.1f + 0.1f == 0.2f과 같은 동치 연산 시 매우 위험할 수 있다는 것

정밀도 실험

0부터 0.1 0.2 0.3 0.4 0.5 0.6 0.7... 이렇게 올라가는 숫자와
0.1 * n을 한 숫자의 동치 비교를 해보는 실험을 진행해보면
(심지어 유니티 실행결과도 달라짐)

```
internal class Program
{
    static void Main(string[] args)
    {
        float accumulative = 0f;
        for (int i = 0; i < 100; i++)
        {
            accumulative += 0.1f;
            Console.WriteLine(0.1f * (i + 1) == accumulative);
        }
        Console.ReadLine();
    }
}
```

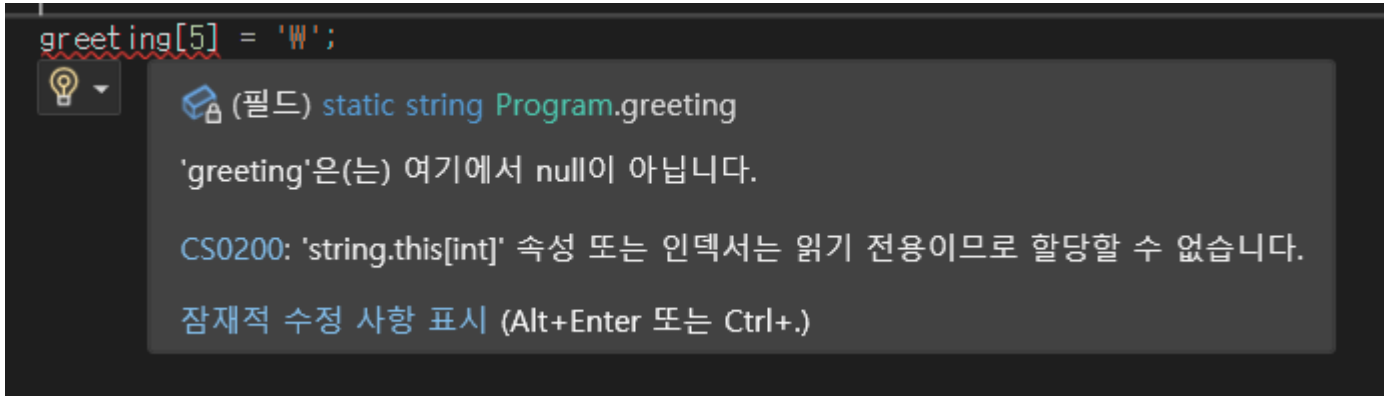
```
1
2
3
4
5
6
21
22
23
95
96
```



문자형 string

string 타입은 대표적인 문자형 타입 자료형입니다.
내부적으로는 글자들이 연결된 형태로 구현되어 있습니다.

문자 등을 특정한 데이터 형태로 변경하는 것을 인코딩이라고 하며, 대표적으로 ASCII가 있습니다.




```
greeting = "Bye, World!";
```

논리형 bool

bool 타입은 논리형 자료형입니다.
true/false의 두 가지 값을 가지며, 조건의 판단 결과 등에서 활용됩니다.

```
if (a >= 30) { // a 가 30이상이면 a >= 30이 true로, 그렇지 않으면 false로 판단된다.
  Console.WriteLine("a 는 30이상입니다.");
}
```

입력과 출력

콘솔 프로그램에서는 문자열로 모든 입력과 출력을 대신함.
유니티로 넘어가서는 마우스/키보드 등 다양한 입력의 사례를 배우겠습니다.

콘솔 프로그램에서, Console 클래스는 다양한 입출력 기능을 제공하며,
Console.WriteLine(string) 은 콘솔에 문자열을 출력하는 기능을 제공하며,
Console.ReadLine()은 유저가 문자열을 입력하도록 프롬프트를 띄운다.

문제은행 풀이 1 – 문자열 출력하기

시작하는 의미에서 간단한 문제를 풀어봅니다.

문제 링크

문제은행 풀이 2 – 문자열 붙여서 출력하기

문자열을 간단하게 처리하는 예제를 학습해봅니다.

문제 링크