유니티 2D 게임개발 프로젝트

유석의 C# 핥아먹기 1

단국대 컴퓨터 그래픽 중앙동아리 CAGI

빠르게 배우는
C# 기초

튜토리얼

프로젝트 만들기

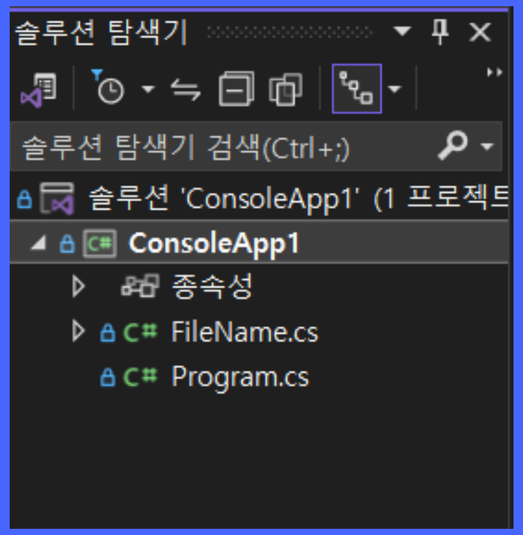
① Visual studio -> 새 프로젝트 -> 콘솔 앱 C#
이때 c#이 보이지 않는다면 visual studio installer에서 .NET 데스크톱 개발, 유니티 게임 개발 2개 설치되었는지 확인

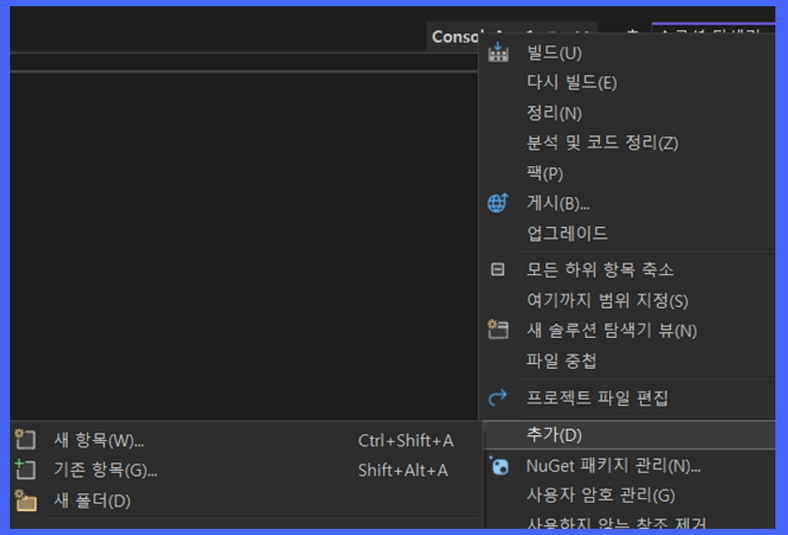
목차 페이지로 돌아가기 →

빠르게 배우는
C# 기초

튜토리얼

새 C#파일 만들기





② 우측의 솔루션 탐색기에서 프로젝트 이름 우클릭 ->
추가 -> 새 항목 -> 파일 이름 지정

목차 페이지로 돌아가기 →

C#

C#코딩의 특징

① 모든 문장은 세미콜론(;)으로 마무리
단, 구조체(보통 중괄호) 이후에는 넣지 않는다.

② c#의 모든 코드는 클래스 내부에 작성한다.

③ 코드 옆에 설명을 넣기 위한 주석은
단일 행은 // 로, 여러 행을 한번에 할때는 /* */ 로 표현한다.

④ 완성한 코드 콘솔에서 테스트하는 방법 : ctrl + f5
Console.WriteLine(“Hello World!”);

목차 페이지로 돌아가기 →

자료형

정수형

1. short - 2바이트(-32,768 ~ 32,767)
2. int - 4바이트(-2,147,483,648 ~ 2,147,483,647)
3. long - 8바이트(-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807)
4. 부호없는 정수형 ushort, uint, ulong, sbyte

목차 페이지로 돌아가기 →

자료형

실수형

1. float - 4바이트, 소수점 이하 7자리
2. double - 8바이트, 소수점 이하 15~16자리
3. decimal - 16바이트 고정소수점, 소수점 이하28~29자리
4. unsigned 형태가 없음

목차 페이지로 돌아가기 →

자료형

논리형, 문자형

① bool - true or false // bool hasJumped = true;

② char - 2바이트로 문자 하나를 저장 // char score = 'A';

③ String - 문자열을 저장 // String Name = "Yoo Seok";

④ NULL - 아무 것도 참조하고 있지 않음
//int? height = null;

목차 페이지로 돌아가기 →

자료형

기타 등등..

① enum - 열거형으로 값들의 집합을 나타냄
// enum Days { Sunday, Monday, ... }

② object - 자료형의 기본 타입으로 실수, 정수, 문자 모두 가능
// object data = 100; data = “Hello”;

목차 페이지로 돌아가기 →

함수

기본 함수

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

public - 접근 제한자 위치, 어디서든 접근이 가능함을 의미함
int - 반환되는 자료형을 나타냄. 반환 안할거면 void
Add- 함수의 이름
int a, int b - 매개변수, 함수 내에 전달할 변수
return - 반환할 값을 나타냄

목차 페이지로 돌아가기 →

클래스

기본 클래스

```
public class Person
{
    private string name;
    public int age;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
}
```

class - class 형임을 의미, c#의 모든 코드는 클래스 내에 작성해야 함
Person - 클래스의 이름
private - 같은 클래스 내에서만 사용 가능
// private, public 안쓰면 기본적으로 private!

get, set - 프로퍼티 접근자

```
// Person A = new Person();
A.Name = “SeokMin”;
Console.WriteLine(A.Name);
```

new는 새로운 객체를 생성, 나중에 자동으로 지워줌

목차 페이지로 돌아가기 →

클래스

```
public class Person {
private string name;
private int age;
Person() {
name = "Unknown";
age = 0; }

Person(string name,
int age) {
this.name = name;
this.age = age; }

public Person(string
name) : this(name, 0)
{ }
 ~Person() { }
```

생성자, 소멸자

생성자함수 이름은 클래스 이름과 동일

Person() - 매개변수 없는 생성자

Person(string name, int age) - 매개변수 2개인 생성자

Person(string name) : this(name,0) - 다른생성자를 호출하는 키워드 this

~Person() {} - 소멸자, 클래스 객체를 삭제함

목차 페이지로 돌아가기 →

클래스

상속

```
public class Student : Person
{
private string studentId;

public Student() : base()
    {
        studentId = "Unknown";
    }
    public Student(string name, int age, string
studentId) : base(name, age)
    {
        this.studentId = studentId;
    }
```

Student가 Person을 상속함
=> 부모 클래스의 메소드, 필드를 공유함
단 , 접근 제한자 public, protected까지

base()는 부모의 생성자에 접근

목차 페이지로 돌아가기 →

클래스

클래스 객체 생성

```
static void Main(string[] args) {
        Person person = new Person();
        person.Name = "John Doe";
        person.age = 30;
        Console.WriteLine("Person Name: " + person.Name);
        Console.WriteLine("Person Age: " + person.Age);
    }
```

Person person = new Person() :
만들어둔 Person 클래스 객체를 만들고, 이름을 person으로 설정
person.Name = “John Doe” :
Person 클래스의 메소드인 Name의 set 프로퍼티에 접근!
person.age = 30 :
Person 클래스의 필드 위 public 변수 age에 접근!

목차 페이지로 돌아가기 →

빠르게 배우는
C# 기초

연산자

산술/할당 연산자

① + - * : 덧셈, 뺄셈, 곱셉 // int a =7; int b = 4; a+b;

② / : 나누기(몫) // int a =8; int b = 4; a/b; 정답은 2에요~

③ %: 나누기(나머지) // int a =9; int b = 4; a/b; 정답은 ?

④ =, +=, -=, *=, /=, %= : =는 오른쪽을 계산해서 왼쪽으로 넘겨줌
// a = b : a에다가 b의 값을 넣어준다는 뜻!
// a += b 와 a = a+b는 같다

목차 페이지로 돌아가기 →

연산자

비교/논리 연산자

① ==, != : 앞 뒤가 서로 같은지/다른지 확인해서 bool값을 돌려줌
// bool calc = (a == a); 항상 calc는 true!

② >, <, >=, <= : 앞 뒤의 값을 비교해서 bool값을 돌려줌

③ && : and 연산자 // bool left = true; bool right = true;
left && right => true, 양쪽이 다 참일때만 true

④ || : or 연산자 // 둘 중 하나만 true면 true를 반환

목차 페이지로 돌아가기 →

연산자

단항/조건부 연산자

① ++, -- : 값을 1만큼 증가, 감소함 // int a = 8; a++; a는 9!

② ! : Not 연산, bool istrue = true; !istrue는 false!

③ ! : Not 연산, bool istrue = true; !istrue는 false!

④ ? : 조건부 연산, return a == b ? 1 : 0;
조건이 참이면 왼쪽을, 거짓이면 오른쪽을 반환함

목차 페이지로 돌아가기 →

조건문

if - else

```
int number = 10;
if (number > 0) {
Console.WriteLine(
"number는 양수입니다.");
}
else if (number < 0) {
Console.WriteLine(
"number는 음수입니다.");
}
else {
Console.WriteLine(
"number는 0입니다."); }
```

```
if (조건식1) {
// 조건식1이 참일 때 실행되는 코드
}
else if (조건식2) {
// 조건식1이 거짓이고 조건식2가 참
일 때 실행되는 코드 }
else {
// 위의 모든 조건이 거짓일 때 실행
되는 코드 }
```

목차 페이지로 돌아가기 →

빠르게 배우는
C# 기초

반복문

For

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("i의 값: " + i);
}


for (초기화; 조건; 반복 후 동작)
{
    // 반복할 코드
}
```

목차 페이지로 돌아가기 →

반복문

While

```
int count = 0;
while (count < 5) {
Console.WriteLine("count의 값: " + count);
count++;
}

while (조건)
{
    // 조건이 참일 때 반복할 코드
}
```

목차 페이지로 돌아가기 →