컴퓨터정보과
C# 프로그래밍

2주차 클래스와 자료형

강의 순서

1. C# 환경설치 / C# 기본 구조
2. **클래스 기본(필드) + 변수,자료형**
3. 클래스 기본(메소드) + 연산자,수식
4. 클래스 기본(메소드) + 제어문 빠르게 보기
5. 배열/리스트/딕셔너리
6. 클래스 기본: 접근제한자 (한정자)
7. 클래스 심화:상속
8. 클래스 심화:인터페이스/추상 클래스
9. 프로퍼티/예외처리/일반화
10. 파일처리
11. UI (Winform or WPF)
12. LINQ/Delegate/Lamda/…

복습

- C# 코드 기본 구조

  namespace
  - ↳ **class**
    - ↳ filed
    - ↳ method
      - Main() 메소드

- C# desktop 프로그램 종류
  - CUI : Console
  - GUI: Winform / WPF
- .NET Framework
  - 실행환경(CLR) + 라이브러리
  - CLR
    - ↳ Garbage Collection
- Visual Studio
  - solution
    - ↳ project

자료형

프로그램에서 데이터를 표현하는 형식(크기, 해석방법, …)

기본 자료형

- **값형** (value type) : struct로 생성한 자료형
  - 숫자형
    - 정수형 (integer)
      - sbyte(1), **byte(1)**, short(2), ushort(2), **int(4)**, uint(4), **long(8)**, ulong(8)
    - 실수형
      - floating point number(부동소수점) - IEEE754
        - float(4), **double(8)**
      - fixed point number (고정소수점)
        - decimal(16)
  - 논리형 (boolean)
    - **bool**
  - 문자형 (character)
    - **char**

참고 : 값 형식 - C# 참조 - C# | Microsoft Learn

- **참조형** (reference type) : class로 생성한 자료형
  - 문자열형
    - **string**

참고 : 참조 형식 - C# 참조 - C# | Microsoft Learn

정수형 Integer Number

signed type	unsigned type	size (bytes)	리터럴 (접미사)
sbyte	**byte**	**1**	**없음, 없음**
short	ushort	2	없음, 없음
int	**uint**	**4**	**없음,100U**
long	ulong	8	100L, 100UL

기타: nint, nuint (시스템 가변 int)
진법 : 10진법, 16진법(0x, 0X 접두어), 2진법(0b, 0B 접두어) … 8진법 없음

실수형 Real Number

type	decimal place	size (bytes)	리터럴 (접미사)
float	6~9	4	3.14f
double	**15~17**	**8**	**3.14 ,3.14d**
decimal	28~29	16	3.14m

float, double : 부동소수점형 (IEEE754), decimal : 고정소수점형
실수계열은 unsigned가 없음

최대값: int.MaxValue, double.MaxVlaue
최소값: int.MinValue, double.MinValue

논리형 Boolean

type	size (bytes)	리터럴
bool	1	true / false

C언어처럼 0이나 1로 참과 거짓을 판단할 수 없음
주로 제어문과 함께 사용 됨

```
//C
while(1) { … }

//C#
while(true) { … }
```

문자형 Character

type	size (bytes)	리터럴	문자코드
char	2	‘1’, ‘a’, ‘A’, …	Unicode (ascii + α)

C언어는 ascii code를 기본으로 1바이트
내부적으로는 정수형

문자열형 String

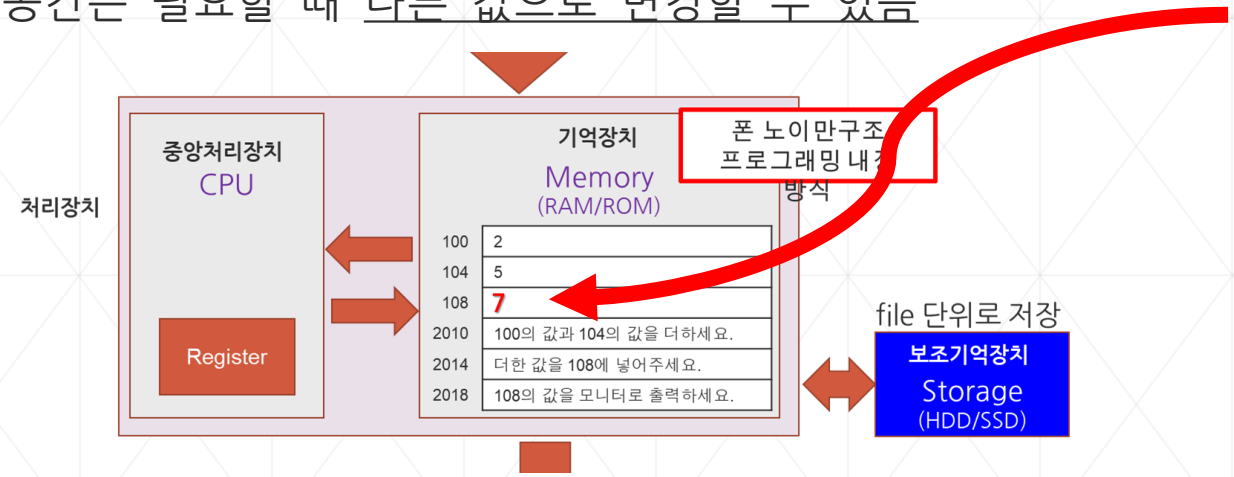
type	size (bytes)	리터럴
string	알 수 없음 (내부적으로 가변)	“ ”, ”a”, ”ABC”, …

C언어처럼 문자 n개가 나열되어 있다고 생각하면 됨.
앞의 다른 타입과 달리 단일 값이 여러 개가 모여있는 복합구조의 형태를 갖고 있음.

변수 & 상수

변수 Variable

- 메모리(RAM)에 프로그램 실행 시 필요한 값을 저장하기 위해 할당받은 공간
  해당 공간은 필요할 때 다른 값으로 변경할 수 있음



- 공간할당을 받기 위해서는 어떤 형태의 데이터를 저장할 것이며, 얼마만큼의 크기가 필요한지 정보를 제공해줘야 함.(자료형 정보가 필요함)

- 변수 선언 및 대입

```
double radius; //변수 공간 선언
radius = 45.5; //변수 공간에 값 대입
```

  - double 자료형 으로 변수 공간을 할당받고 해당 공간의 이름을 프로그램에서는 radius라고 부른다.
  - radius라는 변수 공간에 double 자료의 리터럴 45.5를 보관한다.

상수 Constant

- 메모리(RAM)에 프로그램 실행 시 필요한 값을 저장하기 위해 할당받은 공간
  해당 공간은 처음 초기화한 값을 다른 값으로 변경할 수 없음
- 기호(심볼릭, Symbolic) 상수 선언 및 대입

  const double radius = 45.5; //상수 공간 선언 후 동시에 초기화 해야함
  - double 자료형 으로 상수 공간을 할당받고 해당 공간의 이름을 프로그램에서는 radius라고 부른다.
  - radius라는 상수 공간에 double 자료의 리터럴 45.5를 보관한다.

  ~~radius = 55.5~~; //상수 공간에 초기값 대입 후 다른 값을 대입하면 에러가 발생함
  - 이후 radius의 공간에 있는 값은 변경이 불가능하다.

- 리터럴 (Literal) 상수
  - double radius = **50.5**;
  - 문자 그대로…
  - 50.5와 같이 ‘ 소스 코드에 고정된 값 ' 을 뜻하며, 해당 내용도 상수(constant)로 간주한다.

암시적 형식 지역 변수 var

- 자료형 키워드 : var
  - 가능 조건
    - 메소드 내의 지역변수만 가능
    - 변수 선언과 동시에 초기화
  - 예
    - `var i = 10ul;`
    - `var d = 10.2m;`
    - `var q = 20.1;`

자료형 변환

- 자동 (묵시적) 형변환
- 강제 (명시적) 형변환
  - 형변환 연산자 : ()
    - 값형 ↔ 값형, 참조형 ↔ 참조형
  - 형변환 메소드 : 자료형에 따라 다름
    - 값형 ↔ 참조형
    - 문자열 ↔ 정수, 실수, 불, …
    - 정수,실수, 불, … ↔ 문자열

강제 형변환 - 형변환 연산자 ()

- 큰 자료에서 작은 자료형으로 변환할 때
  - 강제 형변환

```
long a = 100000000;
byte b = (byte)a;
```

  - 자동 형변환

```
byte b = 10;
long a = b;
```

- 정수, 실수간 형변환

```
int a = 10;
double b = (double) a;
```

강제 형변환 - 형변환 메소드

- 문자열과 숫자간의 형변환
  - “ 문자열 ” → 숫자 : 자료형.Parse(“ 문자열값 “)
    - 예제

```
int a = int.Parse( “ 1 ” );
double b = double.Parse( “ 1.1 ” );
bool c = bool.Parse( “ true");
```

  - 숫자 → “ 문자열 ” : 숫자리터럴.ToString()
    - 예제

```
string d = a.ToString(); //1.ToString();  (1).ToString();
string e = b.ToString(); // (1.1).ToString();
string f = true.ToString(); //(true).ToString();
```

객체지향프로그래밍

- Object-Oriented Programming (OOP)
  - 개(객)체지향 프로그래밍

- 1,0,-1,120000000, 45, -230

  → 소수점이 없는 수 → 정수
  → 저장 장치에 따른 2진법 표현 필요
  → 효율적인 음수 표현법 필요 : 2의 보수
  → -230~120000000 사이를 프로그래밍 세계에서 표현할 수 있도록 연구
  → 4바이트 필요 (2진수 32개로 표현 가능한…)
  → int 자료형으로 이들을 묶어서 표현
  → int : 4바이트로 약-21억~+21억 표현할 수 있음.

  → int : 분류의 이름 , 자료형 (**class**)
  → 1, 0, -1, 120000000, 45, -230: int 분류를 따르는 **실제 존재하는 객체(object)**

class (자료형)	object (객체, instance, 인스턴스)
float	3.14F, 43.5F
char	‘a’, ‘A’, ‘B’, ‘\n’, \r’
string	“ ”, “a”, “A”, “ABC”, “\r\n”

객체지향 프로그래밍

- 정수형, 실수형, 논리형, 문자형, 문자열형은 단일한 해당 객체를 표현하기 위한 기본 자료형이고… 이외에 프로그래밍을 하기 위해 다른 자료형이 필요해진다.

- 성적처리
  - 학생, 과목, 성적, …
- 도서관
  - 책, 대출자, …
- 주차관리
  - 자동차, 주차 시간, 주차 장소,…
- 학교
  - 학생, 선생, 과목, …
- 회사
  - 고용주, 고용인, …
- 게임
  - 플레이어, 적, 아이템, …

주소록 관리

```
string name1 = “김인하”;
string addr1= “인천 미추홀구 용현동”;
string phone1 = “010-1111-2221”;
string group1 = “ ”;

string name2 = “이인하”;
string addr2= “인천 연수구 동춘동”;
string phone2 = “010-1111-2223”;
string group2 = “친구”;

Console.WriteLine(name1);
Console.WriteLine(name2);
```

```
AddressBook addrBook1= new  AddressBook ();
addrBook1.Name = “김인하”;
addrBook1.Address = “ “인천 미추홀구 용현동”;
addrBook1.Phone = “010-1111-2221”;
addrBook1.Group = “ ”;

AddressBook addrBook2= new  AddressBook ();
addrBook2.Name = “이인하”;
addrBook2.Address = “ “인천 연수구 동춘동”;
addrBook2.Phone = “010-1111-2223”;
addrBook2.Group = “친구”;

Console.WriteLine(team1.Name);
Console.WriteLine(team2.Name);
```

AddressBook

```
class AddressBook
{
        string Name;
        string Address;
        string Phone;
        string Group;
}
```

```
class AddressBook
{
        public string Name;
        public string Address;
        public string Phone;
        public string Group;
}
```

국어,영어,수학 점수 관리

```
int kor1 = 10;
int eng1 = 20;
int mat1= 20;

int kor2 = 20;
int eng2 = 20;
int mat2 = 30;

Console.WriteLine( kor1);
Console.WriteLine( (kor1+eng1 + mat1) / 3);
Console.WriteLine( (kor2+eng2 + mat2) / 3);
```

```
Score score1 = new Score();
score1.Kor = 10;
score1.Eng = 20;
score1.Mat = 20;

Score score2 = new Score();
score2.Kor = 10;
score2.Eng = 20;
score2.Mat = 20

Console.WriteLine( score1.Kor);
Console.WriteLine( score1.Average);
Console.WriteLine( score2.Average);
```

Score

```
class Score
{
        int kor;
        int eng;
        int math;
}
```

```
class Score
{
        public int kor;
        public int eng;
        public int math;
}
```

```
class Score
{
        public int Kor;
        public int Eng;
        public int Mat;
        public int Average
        {
             get
            {
               return (Kor + Eng+ Mat)/3;
             }
       }
}
```

면적 관리

```
double width1= 20.1;
double height1 = 30.5;

double width2 =2;
double height2=3;

Console.WriteLine(width1);
Console.WriteLine(width1 * height1);
Console.WriteLine(width2 * height2);
```

```
Rect rect1 = new Rect();
rect1.Width = 20.1;
rect1.Height = 30.5;

Rect rect2 = new Rect();
rect2.Width = 2;
rect2.Height = 3;

ConsoleWriteLine(rect1.Width);
Console.WriteLine( rect1.Area);
Console.WriteLine( rect2.Area);
```

Rect

```
class Rect
{
        double Width;
        double Height;
}
```

```
class Rect
{
        public double Width;
        public double Height;
}
```

```
class Rect
{
        public double Width;
        public double Height;

        public double Area
        {
             get
             {
               return Width * Height;
             }
        }
}
```

회원 관리

```
string name1 = “ 김인하 ” ;
int age1 = 27;
bool isRegular1 = true;

string name2 = “ 이인하 ” ;
int age2 = 22;
bool isRegular2 = false;

Console.WriteLine(name1);
Console.WriteLine(
    $ ” {name1}회원은 {isRegular1? ” 정회원 ” : ” 준회원 ” }입니다. ” }
Console.WriteLine(
    $ ” {name2}회원은 {isRegular2? ” 정회원 ” : ” 준회원 ” }입니다. ” }
```

수업시간에 직접 할 것

Member

```
class Member
{
        string Name;
        int Age;
        bool IsRegular;
}
```

수업시간에 직접 할 것

```
class Member
{
        public string Name;
        public int Age;
        public bool IsRegular;
}
```

경기 팀 관리

```
string name1 = “ SSG ”;
string coach1 = “ 이숭용 ”;
int level1 = 9;
string home1 = “ 인천 ”;

string teamName2 = “ 삼성 ”;
string coach2 = “ 박진만 ”;
int level2 = 3;
string home2 = “ 대구 ”;

Console.WriteLine(name1);
string stt = (level1 >= 5) ? “ 가능 ” : “ 불가능 ”;
Console.WriteLine($ ”{name1} ” 은 현재 가을야구 {stt} ” );
```

수업시간에 직접 할 것

Team

```
class Team
{
        public string Name;
        public string Coach;
        public int Level;
        public string Home;
}
```

수업시간에 직접 할 것

은행 계좌 관리

```
string accountNumber1 = “111-1111-1”;
string owner1= “김인하”;
decimal balance1 = 10000000;

string accountNumber2 = “111-1111-2”;
string owner2= “김인하”;
decimal balance2 = 10000000;

balance1 += 10000;
Console.WriteLine(balance1);
```

수업시간에 직접 할 것

Account

```
class Account
{
    public string Number;
    public string Owner;
    public decimal Balance;
}
```

수업시간에 직접 할 것

직원 관리

```
string name1 = “김인하”;
string number1= “20240001”;
string depart1 = “경영지원”;
decimal salary1 = 36000000;

string name2 = “이인하”;
string number2= “20200005”;
string depart2 = “기술개발”;
decimal salary2 = 45000000;

Console.WriteLine(“월급:{salary1/12}”);
```

수업시간에 직접 할 것

Employee

```
class Employee
{
    public string Name;
    public string Number;
    public string Depart;
    public decimal Salary;
}
```

수업시간에 직접 할 것