

# 컴퓨터정보과 C# 프로그래밍

---

3주차 클래스와 연산자/수식 (메소드)

# 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
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/Lambda/...

# 복습 및 2주차 추가요소

- 자료형
  - 기본자료형 => 리터럴이 존재하는 자료형
    - value type
      - 정수, 실수, 논리형, 문자형
    - reference type
      - 문자열
- 자료를 저장하는 공간
  - 변수
  - 상수
- 자료형 변환
  - 자동 형변환
  - 강제 형변환
    - () 연산자
    - 형변환 메소드
- 클래스(구조체) == 자료형 정의
  - 기본자료형 중 value type은 struct로 정의한 자료형
  - 기본자료형 중 reference type은 class로 정의한 자료형
  - 클래스의 기본 구성요소
    - 변수 & 메소드

# 연산자와 수식

---

데이터를 가공(계산)하는 기능

# 연산자

- 산술연산자
- 문자열연산자
- 비교연산자
- 논리연산자
- 대입연산자
- 증감연산자
- sizeof 연산자
- 비트연산자
- `&`, `^`, `|`, `~`
- `<<`, `>>`
- `as`, `is`, `()`, `typeof()`, `?:`, `??`, `=>`, `new`, `[]`, `.`, `,` ...

# 산술 연산자

연산자	설명	비교
+	덧셈	
-	뺄셈	
*	곱셈	
/	나눗셈	
%	나머지	실수에도 사용 가능

1. 정수간 연산 결과 : 정수
2. 실수간 연산 결과 : 실수
3. 실수와 정수간 연산 결과 : 실수

# 문자열 연산자

연산자	설명	비교
+	문자열 연결	
문자열[숫자]	문자 선택	

- 문자열 연결 연산자
  - "1" + "3" → "13"
  - "1" + 3 → "13"
  - 1 + 3 → 4
  - '가' + '힉' → 99235

(문자열 및 숫자 연산 결과 : 문자열 연결)

- 문자 선택 연산자
 

```
string name = "name";
char a = name[1];
```

## 조건 연산자 (삼항 연산)

연산자	설명	비교
?:	조건에 따른 선택	불_표현식 ? (true인 경우) : (false인 경우)

# 비교 연산자

연산자	설명	비교
==	같다	
!=	다르다	
>	왼쪽 피연산자가 크다	
>=	왼쪽 피연산자가 크거나 같다.	
<	오른쪽 피연산자가 같다.	
<=	오른쪽 피연산자가 크거나 같다.	

# 논리 연산자

연산자	설명	비교
!	논리 부정	!true
	논리 합	true    false
&&	논리 곱	True && false

- 두 연산자의 결과 값은 항상 bool 형으로 나온다.



# 대입 연산자

연산자	설명	비교
=	오른쪽 피연산자를 왼쪽 변수에 대입	$a = b$
+=	오른쪽 피연산자와 왼쪽 피연산자를 연산하고 왼쪽 변수에 넣는다.	$a += b$
-=		$a -= b$
*=		$a *= b$
/=		$a /= b$
%=		$a \% = b$

# 증감 연산자

연산자	설명	비교
[변수]++	기존 값에 1을 더함	후위 $a++$
++[변수]	기존 값에 1을 더함	전위 $++a$
[변수]--	기존 값에 1을 뺌	후위 $a--$
--[변수]	기존 값에 1을 뺌	전위 $--a$

# 메소드 (method)

---

실질적인 실행 단위

# 객체지향프로그래밍

- 객체지향 프로그래밍은 세상에 존재하는 모든 것을 객체로 보고 이들 객체가 서로 상호작용하는 것을 프로그래밍으로 옮긴 것.
  - 이전 챕터에는 데이터만 표현했음.
  - 객체 자체의 기능을 통해 객체 간의 상호작용
- 객체 : 데이터 + 기능
  - 사람 : (시력 + 보다) (체중+먹다/싸다/소화시키다) (다리 길이+걷다/뛰다)
- int가 표현할 수 있는 최대값? 최소값? int 값을 문자열로 바꾸고 싶어? 문자열을 int로 바꾸고 싶어?
  - `int.MaxValue` → property
  - `int.MinValue` → property
  - `(1).ToString()` → Method
  - `int.Parse("1")` → Method
- 객체지향의 int는 특정 값만을 표현하기만 하는 것이 아니라 특정 값과 그 값을 통해서 산출되는 다양한 특징이나 기능을 갖도록 설계 되어있다.
  - `float/string/bool/long/...` 모두
- C#에서 객체를 설계하는 도구 : `class, struct`
  - 객체의 기능을 설계하는 도구 : Method

# 다른 언어와 비교

## C언어

- 구조체(struct)
  - 멤버 : 변수 (variable)
- 함수(function)



## C#

- 클래스(class) / 구조체(struct)
  - 필드 (field) : 변수(variable)와 동일
  - 메소드(method) : 함수와 동일



# 변수

- 변수는 선언 위치에 따라서 의미와 생존시간(생성 시점과 종료 시점)이 다르다
  - class/struct 안에 선언한 변수
    - 필드 (field)
      - 인스턴스 필드 (Instance field)
      - 정적 필드 (static field) : static
        - 상수 필드 (constant field) : const
      - 읽기 전용 필드 (readonly field) : readonly
      - ...
  - method 안에 선언한 변수
    - 지역변수 (local variable)
      - 매개변수 (parameter variable)

# 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 (AddressBook)

```
AddressBook addrBook1 = new AddressBook();  
addrBook1.Name = "김인하";  
addrBook1.Address = "인천 미추홀";  
addrBook1.Phone = "010-1111-1111";  
addrBook1.Group = "";
```

```
AddressBook addrBook2 = new AddressBook();  
addrBook2.Name = "이인하";  
addrBook2.Address = "인천 남미추홀";  
addrBook2.Phone = "010-1111-1112";  
addrBook2.Group = "친구";
```

```
Console.WriteLine(addrBook1.Address);
```

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

# 국어, 영어, 수학 점수 관리 (Score)

```
Score score1 = new Score();
```

```
score1.Kor = 10;
```

```
score1.Eng = 20;
```

```
score1.Mat = 30;
```

```
Score score2 = new Score();
```

```
score2.Kor = 10;
```

```
score2.Eng = 20;
```

```
score2.Mat = 30;
```

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

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

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

```
class Score
```

```
{
```

```
//필드(field) : class안에 선언한 변수
```

```
public int Kor;
```

```
public int Eng;
```

```
public int Mat;
```

```
public int Average {
```

```
    get {
```

```
        return (Kor + Eng + Mat) / 3;
```

```
    }
```

```
}
```

```
}
```



# 면적 관리

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

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

```
rect1.ChangeWidth(20.0);  
rect1.ChangeHeight(-3);
```

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

# Rect

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

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

    public void ChangeWidth(double size)
    {
        Width += size;
    }

    public void ChangeHeight(double size)
    {
        Height += size;
    }
}
```

# 회원 관리

```
Member mem1 = new Member();
```

```
mem1.Name = "김인하";
```

```
mem1.Age = 27;
```

```
mem1.IsRegular = true;
```

```
Member mem2 = new Member();
```

```
mem2.Name = "이인하";
```

```
mem2.Age = 22;
```

```
mem2.IsRegular = false;
```

```
mem1.ChangeGrade();
```

```
mem2.ChangeGrrade();
```

```
Console.WriteLine(mem1.Name);
```

```
Console.WriteLine(mem1.Status);
```

```
Console.WriteLine(mem2.Status);
```

# Member

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

    public string Status
    {
        get {
            string type = IsRegular ? "정회원" : "준회원";
            return $"{Name} 회원은 {type} 입니다.";
        }
    }

    public void ChangeGrade()
    {
        IsRegular = !IsRegular;
    }
}
```

# 경기 팀 관리

```
Team team1 = new Team();  
team1.Name = "SSG";  
team1.Coach = "이승용";  
team1.Level = 9;  
team1.Home = "인천";
```

```
Team team2 = new Team();  
team2.Name = "삼성";  
team2.Coach = "박진만";  
team2.Level = 3;  
team2.Home = "대구";
```

```
team1.IncreaseLevel(2);  
team1.DecreaseLevel(2);
```

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

# Team

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

    public string CurrentStatus
    {
        get {
            string stt;
            if (Level >= 5) {
                stt = "가능";
            } else {
                stt = "불가능";
            }
            return $"{Name}은 현재 가을야구 {stt}";
        }
    }
}
```

```
public void IncreaseLevel(int value)
{
    if (Level - value >= 1) {
        Level -= value;
    } else {
        Level = 1;
    }
}

public void DecreaseLevel(int value)
{
    if (Level + value >= 10) {
        Level += value;
    } else {
        Level = 10;
    }
}
```

# 은행 계좌 관리

```
Account acc1 = new Account();  
acc1.Number = "111 - 1111 - 1";  
acc1.Owner = "김인하";  
acc1.Balance = 100000000;
```

```
Account acc2 = new Account();  
acc2.Number = "111 - 1111 - 2";  
acc2.Owner = "김인하";  
acc2.Balance = 100000000;
```

```
acc1.AddBalance(10000);  
acc2.SubBalance(100);  
Console.WriteLine(acc1.Balance);
```

# Account

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

    public bool AddBalance(decimal money)
    {
        Balance += money;
        return true;
    }
}
```

```
public bool SubBalance(decimal money)
{
    if (Balance - money >= 0) {
        Balance -= money;
        return true;
    } else {
        return false;
    }
}
```



# 직원 관리

```
Employee emp1 = new Employee();
```

```
emp1.Name = "김인하";
```

```
emp1.Number = "20240001";
```

```
emp1.Depart = "경영지원";
```

```
emp1.Salary = 3_600_0000;
```

```
Employee emp2 = new Employee();
```

```
emp2.Name = "이인하";
```

```
emp2.Number = "20200005";
```

```
emp2.Depart = "기술개발";
```

```
emp2.Salary = 4_5000_000;
```

```
emp1.ChangeSalary(5); //5% 상승
```

```
Console.WriteLine($"월급:{emp1.MonthlySalary}");
```

# Employee

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

    public decimal MonthlySalary
    {
        get {
            return Salary / 12;
        }
    }
}
```

```
public decimal ChangeSalary(double percent)
{
    Salary *= (decimal)(1.0 + percent);
    return Salary;
}
```