# 2. 객체지향 프로그래밍 (OOP)

## 객체 지향 프로그래밍(Object-Oriented Programming)

### 객체지향 프로그래밍이 개발된 이유

초창기 프로그램은 컴퓨터 계기판의 스위치를 조작하여 만들어 졌음으로 아주 작은 프로그램만이 가능했다. 그러다 어셈블리 언어가 개발되어 조금 더 긴 프로그래밍이 가능해졌고 이후에는 고급언어 PORTRAN의 출현으로 수천 라인의 프로그래밍이 가능해 졌지만 읽기도 어렵고 관리도 어려워 1960년대에 구조적 프로그래밍 언어가 나타나게 되었다. 구조적 프로그래밍 언어의 대표주자가 C언어 이다. 하지만 이러한 구조적 프로그래밍 언어도 5만라인 이상의 복잡한 프로그램이 작성되면 관리가 어려워 한계점에 다다르게 된다. 이에 대해 새로운 프로그래밍 방법론이 요구되면서 객체지향의 개념이 나타나게 되었다.

### 객체

객체는 각자가 자신의 명령과 데이터를 포함하는 독립적인 존재 이다.

### 객체 지향 프로그래밍이란? (Object-Oriented Programming)

어떠한 문제가 주어지면 그 문제를 여러개의 객체로 나누어 구성한다. 이렇게 구성된 객체들의 상호 관계를 구현하여 프로그래밍을 하는 것이 객체 지향 프로그래밍이다.

### 객체지향 프로그래밍 과정

① 어떠한 객체를 구현할 것인지를 식별한다.
② 각 객체들의 특징을 결정해 모양을 변수로 동작을 함수로 구현하여 객체를 완성한다.
③ 완성된 객체들 간의 메시지 전달을 구현하여 프로그램을 완성시킨다.

### 모델링

어떠한 실제 세계의 사물을 가지고 프로그램의 대상으로 이끌어 내는 과정

### 추상화 (모델링의 원리)

사물을 정의할 때  특정 상태나 동작만을 추출하는 것, 모델링에 필요한 상태나 동작만을 추출

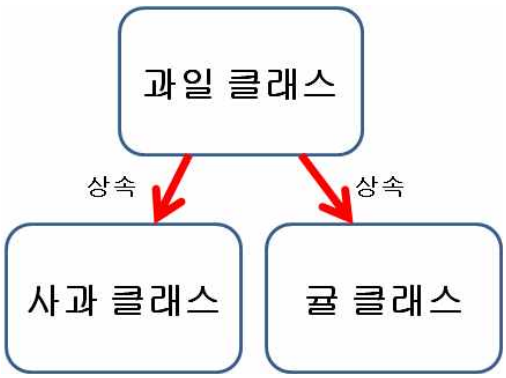
## 클래스의 개념

데이터 추상화를 사용자 정의 데이터 형으로 바꿀 수 있는 C++의 도구이며 C의 구조체에 비해 의미가 확장된 C++만의 구조체라 할 수 있다. 클래스는 멤버함수를 통해 이형 타입 변수의 집합인 구조체가 스스로의 동작을 정의 할 수 있도록 기능이 추가된 것이다. 처음의 이름은 구조체 였지만 C++의 창시자인 비얀 스트로스트룹이 부여한 새로운 이름이 클래스이다

## 클래스의 특징

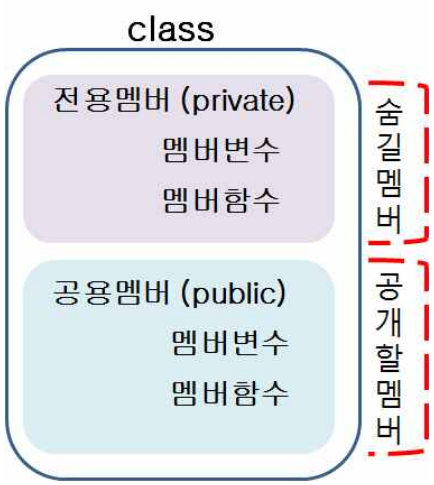
상속성 (Inheritance)

- 하나 이상의 클래스로부터 그 클래스의 다른 클래스가 물려받는 것을 말한다.
- 클래스를 디자인 할 때 중복되는 특성을 매번 정의 하지 않고 한 번의 정의로 재사용할 수 있게 해준다.
- 한번 정의 된 내용을 다시 사용할 수 있음으로 경제적이며 검증된 내용을 물려받아 사용함으로 코드의 신뢰성을 높일 수 있다.




캡슐화 (Encapsulation)

- 공개할 멤버는 공개하고 숨길 멤버는 숨기면서 관련된 데이터 멤버(데이터)와 멤버함수(처리)를 하나의 단위로 묶는 것을 의미
- 데이터 멤버와 데이터 멤버를 처리하는 멤버함수를 함께 묶어 공개할 멤버와 공개하지 않을 멤버를 구분하여 내부의 복잡한 내용은 숨기고 추상화 시킨 것




다형성 (Polymorphism)

- 하나의 인터페이스로 서로 다른 반응을 얻어내는 것을 의미
- 함수의 이름이 같더라도 전달인자의 데이터 형이나 수에 따라 다른 함수로 인식을 하는 중복함수
- 기존의 연산자에 새로운 기능을 부여하여 문맥에 따라 다르게 연산하게 하는 연산자 중복기능
- 클래스 상속관계의 파생클래스가 자신의 멤버함수를 정확하게 호출하게 하는 가상함수
- 데이터 형에 관계없이 동일한 기능을 수행할 수 있도록 하는 템플릿 함수

## 클래스의 정의

① C++에서는 구조체의 태그가 타입으로 승격되어 태그로부터 바로 구조체 변수를 선언할 수 있다.

② 구조체가 하나의 타입으로 인정되는 것과 마찬가지로 클래스도 하나의 타입으로 취급된다.

③ 클래스의 이름은 int, double, char 같은 기본형 타입과 동등한 자격을 가지며 사용 방법도 똑같다.

④ C++은 클래스가 완전한 타입이 되기 위한 여러 가지 언어적 장치(생성자, 연산자 오버로딩 등)를 제공 한다.

## 클래스의 멤버 변수의 키워드

| | |
|---|---|
| private | 전용부분, 이 부분에 존재하는 멤버는 오직 멤버함수들에 의해서만 접근이 가능하고 클래스 외부에서의 접근이 불가능하다 |
| protected | 클래스의 상속을 위하여 존재하는 키워드, 클래스 외부에서는 접근할 수 없으나 파생클래스에서는 접근이 가능 |
| public | 공용부분, 이 부분에 존재하는 멤버는 멤버함수 뿐만이 아니라 클래스 외부에서의 접근이 가능하다. |

## 클래스의 구현

```
class 클래스명{
        private:
            :           // 배타적으로 사용할 멤버들을 정의
        protected:
            :           // 상속시 공용으로 사용할 멤버들을 정의
        public:
            :           // 공용으로 사용할 멤버들을 정의
};
```

## 클래스의 선언 예

```
calss Circle {
        double Redius;
public :
        double Cicle_Area();
}
```

## Lx1) 클래스 Circle의 객체 Cir1을 생성하여 원의 면적을 구하는 프로그램

```
#include <iostream.h>

class Circle {
	double Radius;
public :
	double Circle_Area() {
		Radius = 5.0;
		return (3.14 * 5.0 * 5.0);
	}
};

int main()
{
	Circle Cir1;
	// Cir1.Radius = 6.0; 에러
	cout << "원의 면적은 " << Cir1.Circle_Area()
		 << "입니다" << endl;
	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| 원의 면적은 78.5입니다<br><br>Press any key to continue... | 선언한 객체를 가지고 멤버 함수를 호출할 때에는 점( . ) 연산자를 사용하여 Cir1.Circle_Area()처럼 기술한다.<br>Cir1.Radius = 6.0 은 private 멤버를 클래스 외부에서 접근했기 때문에 에러가 발생한다. |

## Lx2) 클래스 Counter의 멤버 변수 Value에 대한 멤버 함수들의 선언

```
#include <iostream.h>

class Counter {
	int value;
public :
	void set(int n)  { value = n; }
	void inc() {++value; }
	void dec() {--value; }
	int val() { return value; }
};

int main()
{
	Counter cnt1, cnt2;
	cnt1.set(10);
	cnt1.inc();
	cout << "cnt1 value : " << cnt1.val() << endl;

	cnt2.set(0);
	cnt2.dec();
	cout << "cnt2 value : " << cnt2.val() << endl;

	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
| --- | --- |
| cnt1 value : 11<br>cnt2 value : -1<br><br>Press any key to continue... | 멤버함수를 통해 private영역의 변수 value 값을 변화시켰다. |

## Ex3) 클래스 외부에서 멤버함수 정의하기

```
#include <iostream.h>

class Circle {
	double Radius;
public :
	double Circle_Area();
};

double Circle::Circle_Area()
{
	Radius = 5.0;
	return (3.14 * 5.0 * 5.0);
}

int main()
{
	Circle Cir1;
	cout << "원의 면적은 " << Cir1.Circle_Area()
			<< "입니다" << endl;
	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
| --- | --- |
| 원의 면적은 78.5입니다.<br><br>Press any key to continue... | 클래스외부에서 멤버함수를 기술할 때는 :: 연산자를 이용하여 함수이름 옆에 소속클래스 이름을 적어주어야 한다. |

## Ex4) 디폴트 매개변수를 이용한 값의 생략

```
#include <iostream.h>

class Date {
	int year, month, day;
public:
	void set_date(int y, int m, int d);
	int get_year()  { return year; }
	int get_month() { return month; }
	int get_day()   { return day; }
};

void Date::set_date(int m, int d, int y = 2005) {
	year = y;
	month = m;
	day = d;
}
int main()
{
	Date dt;
	dt.set_date(3, 25);
	cout << dt.get_year() << "."
	     << dt.get_month() << "."
			<< dt.get_day() << endl;

	dt.set_date(1, 1, 2000);
	cout << dt.get_year() << "."
			<< dt.get_month() << "."
			<< dt.get_day() << endl;
	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| 2005.3.25<br>2000.1.1<br><br>Press any key to continue... | void set_date() 함수는 인수가 세 개지만 y에 대한 값을 디폴트 값으로 지정하였음으로 y에 대한 인수의 생략이 가능하다 |

## Lx5) 대입연산자를 이용한 객체 복사 프로그램

```
#include <iostream.h>
#include <string.h>

class String {
	char *str;
	int len;
public:
	void set(char *cp);
	void prn();
};

void String::set(char *cp) {
	len = strlen(cp);
	str = new char[len+1];
	strcpy(str, cp);
}

void String::prn() {
	cout << "문자열 : " << str
		<< "(" << len << ")" << endl;
}

int main()
{
	String str1, str2;
	str1.set("The first string");
	str2.set("This is the second string");
	str1.prn();
	str2.prn();
	cout << "₩n=========================₩n";
	str1 = str2;  //객체 복사
	str1.prn();
	str2.prn();
	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| 문자열 : The first string(16)<br>문자열 : This is the second string(25)<br><br>=========================<br>문자열 : This is the second string(25)<br>문자열 : This is the second string(25)<br><br>Press any key to continue... | 같은 클래스형인 str1과 str2는 할당 연산자를 이용하여 객체 복사가 가능하다. |

## Lx6) 함수의 인남인시로 객제 신남이기

```
#include <iostream.h>

class XY {
	int x, y;
public:
	void set_xy(int a, int b) { x = a, y = b; }
	int get_x()	{ return x; }
	int get_y()	{ return y; }
};

void swap(XY ob) {
	ob.set_xy(ob.get_y(), ob.get_x());
	cout << "== swap() ==Wn";
	cout << "x = " << ob.get_x() << ", y = " << ob.get_y() << endl;
}

int main()
{
	XY ob;
	ob.set_xy(10, 20);
	swap(ob);
	cout << "Wn== Main() ==Wn";
	cout << "x = " << ob.get_x() << ", y = " << ob.get_y() << endl;

	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| == swap() ==<br>x = 20 , y = 10<br>== Main() ==<br>x = 10 , y = 20<br><br>Press any key to continue... | 객체는 일반 변수와 동일한 형식으로 일반 함수의 매개 변수로 전달이 가능하며 위 예제에서는 call by value 로 함수를 호출하였음으로 swap함수내의 객체 ob는 swap함수의 소멸과 동시에 사라진다. |

## Lx7) 객체를 반환하는 함수에 대한 프로그램

```
#include <iostream.h>
#include <string.h>

class String {
	char *str;
	int len;
public:
	void set(char *cp);
	void prn();
};

void String::set(char *cp) {
	len = strlen(cp);
	str = new char[len+1];
	strcpy(str, cp);
}

void String::prn() {
	cout << "문자열 : " << str << "(" << len << ")₩n";
}

String input() {
	char text[80];
	String temp;
	cout << "Input String : ";
	cin >> text;
	temp.set(text);
	return temp;
}

int main()
{
	String str1;
	str1 = input();
	str1.prn();
	return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
| --- | --- |
| Input String : Object-oriented<br>문자열 : Object-oriented(15)<br><br>Press any key to continue... | 객체를 일반 변수와 동일한 형식으로 일반 함수의 return data로 돌려 줄 수 있다. |

## Lx8) 객체 배열을 이용한 프로그램

```
#include <iostream.h>

class Circle {
	int Radius;
public :
	void set(int R) { Radius = R; }
	 double Area() {  return (3.14 * Radius * Radius); }
};

int main()
{
	Circle Cir[5];  // 객체 배열

	cout << "== 원의 넓이 구하기 ==₩n";
	for(int i=0; i<5; i++)  {
		Cir[i].set(i+1);
		cout << "반지름(" << i+1 << ")=> " << Cir[i].Area() << endl;
	}
return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| == 원의 넓이 구하기 ==<br>반지름(1)=> 3.14<br>반지름(2)=> 12.56<br>반지름(3)=> 28.26<br>반지름(4)=> 40.24<br>반지름(5)=> 78.5<br><br>Press any key to continue... | C++에서는 class가 하나의 완벽한 data type로 인정됨으로 객체의 배열도 생성 가능하다. |

## Lx9) 객체 포인터를 이용하여 함수 호출하기

```
#include <iostream.h>

class Point {
	int xp, yp;
public :
	void setpt(int x, int y) { xp = x, yp = y; }
	void offset(int x, int y) {xp += x, yp +=y; }
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

int main()
{
	Point p, *ptr;    // Point 형 객체 포인터 *ptr
	p.setpt(10, 20);
	ptr = &p;
	ptr->offset(100, 200);

	cout << "== 직접 참조 ==₩n";
	cout << "x 좌표 : " << p.get_x()
		<< ", y 좌표 : " << p.get_y() << endl;

	cout << "₩n== 간접 참조 ==₩n";
	cout << "x 좌표 : " << ptr->get_x()
		<< ", y 좌표 : " << ptr->get_y() << endl;
return 0;
}
```

| 실 행 결 과 | 결 과 분 석 |
|---|---|
| == 직접 참조 ==<br>x 좌표 : 110, y 좌표 : 220<br>== 간접 참조 ==<br>x 좌표 : 110, y 좌표 : 220<br><br>Press any key to continue... | C에서는 모든 data type에 대한 포인터가 존재함으로 하나의 data type으로 인정받는 클래스 역시 포인터의 생성이 가능하며 객체 포인터로 객체를 참조 할 경우 대입된 객체의 멤버를 호출하기 위해서는 -> 연산자를 사용한다. |