03

클래스와 객체

학습 목표

1. 실세계의 객체와 C++ 객체에 대해 이해한다.
2. C++ 클래스를 작성할 수 있다.
3. 객체를 생성하고 활용할 수 있다.
4. 생성자와 소멸자를 알고 작성할 수 있다.
5. private, protected, public 접근 지정자를 이해한다.
6. 인라인 함수의 목적을 이해하고 활용할 수 있다.

함수 중복(function overloading) -> 6장

- 동일한 이름의 함수를 여러 개 만드는 기능
- 이름은 같지만 매개 변수 타입이 다르거나 개수가 달라야 함
- C언어는 이름만으로 함수를 구분하지만 C++는 이름과 인자로 구분

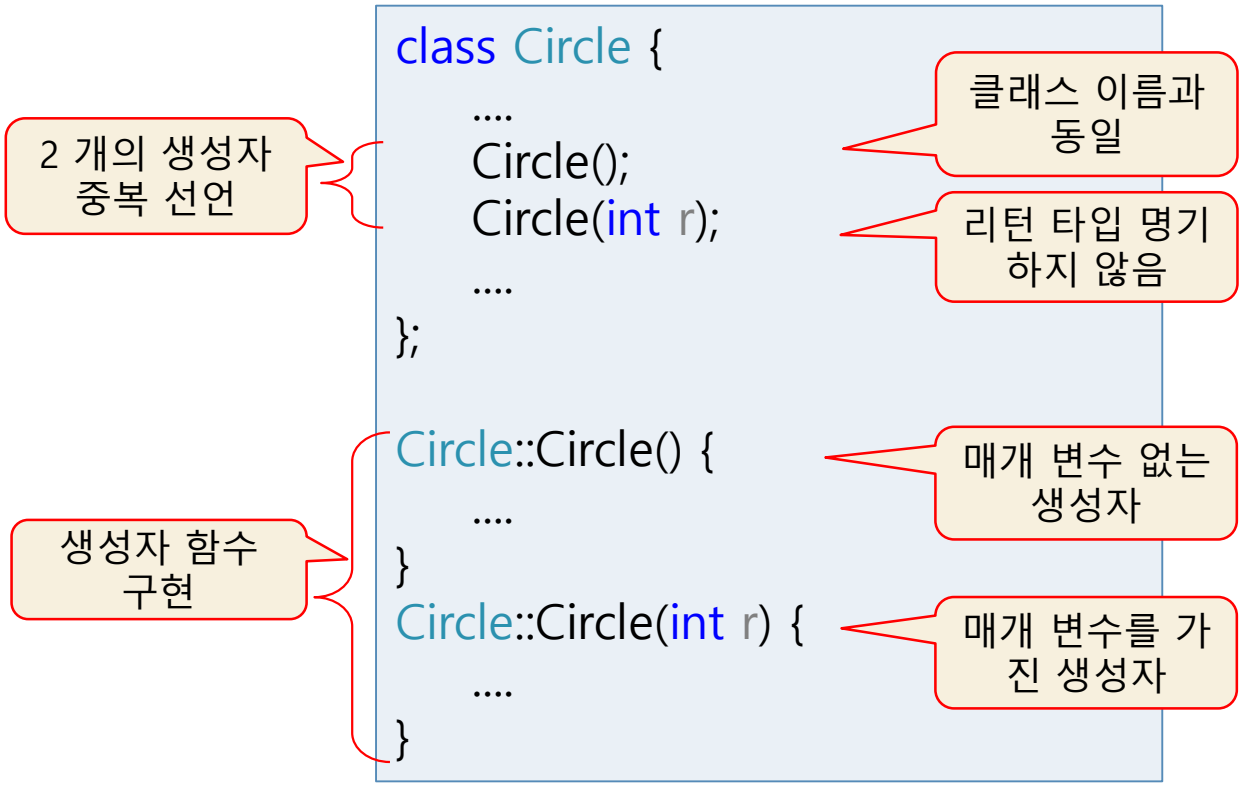
```
int sum1(int a, int b, int c) {
    return a + b + c;
}
double sum2(double a, double b) {
    return a + b;
}
int sum3(int a, int b) {
    return a + b;
}
cout << sum1(2, 5, 33);
cout << sum2(12.5, 33.6);
cout << sum3(2, 6);
```

```
int sum(int a, int b, int c) {
    return a + b + c;
}
double sum(double a, double b) {
    return a + b;
}
int sum(int a, int b) {
    return a + b;
}
cout << sum(2, 5, 33);
cout << sum(12.5, 33.6);
cout << sum(2, 6);
```

생성자(constructor)

- 객체가 생성될 때 필요한 초기화 작업을 수행하는 멤버함수
  - 멤버변수 초기화, 메모리 동적할당(4장), 파일 열기(12장), 등
- **객체가 생성되는 시점에서 자동으로 호출되는 멤버 함수**
- 생성자 이름은 반드시 클래스 이름과 동일
- 생성자는 리턴 타입을 선언하지 않음
  - 리턴 타입 없음. void 타입도 안됨
- 객체 생성 시 오직 한 번만 호출
  - 자동으로 호출되고 임의로 호출할 수 없음
  - 각 객체마다 생성자 실행
- 생성자는 중복 가능
  - 생성자는 한 클래스 내에 여러 개 선언 가능, 매개변수는 달라야 함
  - 중복된 생성자 중 하나만 실행

생성자(constructor)



생성자에 인자전달 방법

- 생성자는 직접 호출할 수 없고 자동으로 호출됨 -> 일반적인 함수 호출방식으로 생성자의 인자 전달 불가능
- 객체를 생성할 때 객체명 뒤에 소괄호안에 생성자의 인자를 전달
- 객체명 뒤의 인자의 자료형, 개수에 맞는 생성자를 호출함

```
class Circle {
      ....
      Circle();
      Circle(int x);
      Circle(int x, int y);
      ....
};
Circle::Circle(){ .... }
Circle::Circle(int x){ .... }
Circle::Circle(int x, int y){ .... }
```

```
Circle a;          //매개변수 없는 생성자 호출
Circle b(30);      //매개변수 1개 생성자 호출
Circle c(10, 20);  //매개변수 2개 생성자 호출
Circle d();        //error, 함수선언으로 해석
Circle e(1,2,3);   //error, 인자에 맞는 생성자x
```

예제 3–3: 2개의 생성자를 가진 Circle 클래스


```
#include <iostream>
using namespace std;
class Circle {
public:
	int radius;
	Circle();			// 매개 변수 없는 생성자
	Circle(int r);			// 매개 변수 1개 생성자
	double getArea();
};
Circle::Circle() {
	radius = 1;
	cout << "반지름 " << radius << " 원 생성" << endl;
}
Circle::Circle(int r) {
	radius = r;
	cout << "반지름 " << radius << " 원 생성" << endl;
}
double Circle::getArea() {
	return 3.14 * radius * radius;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

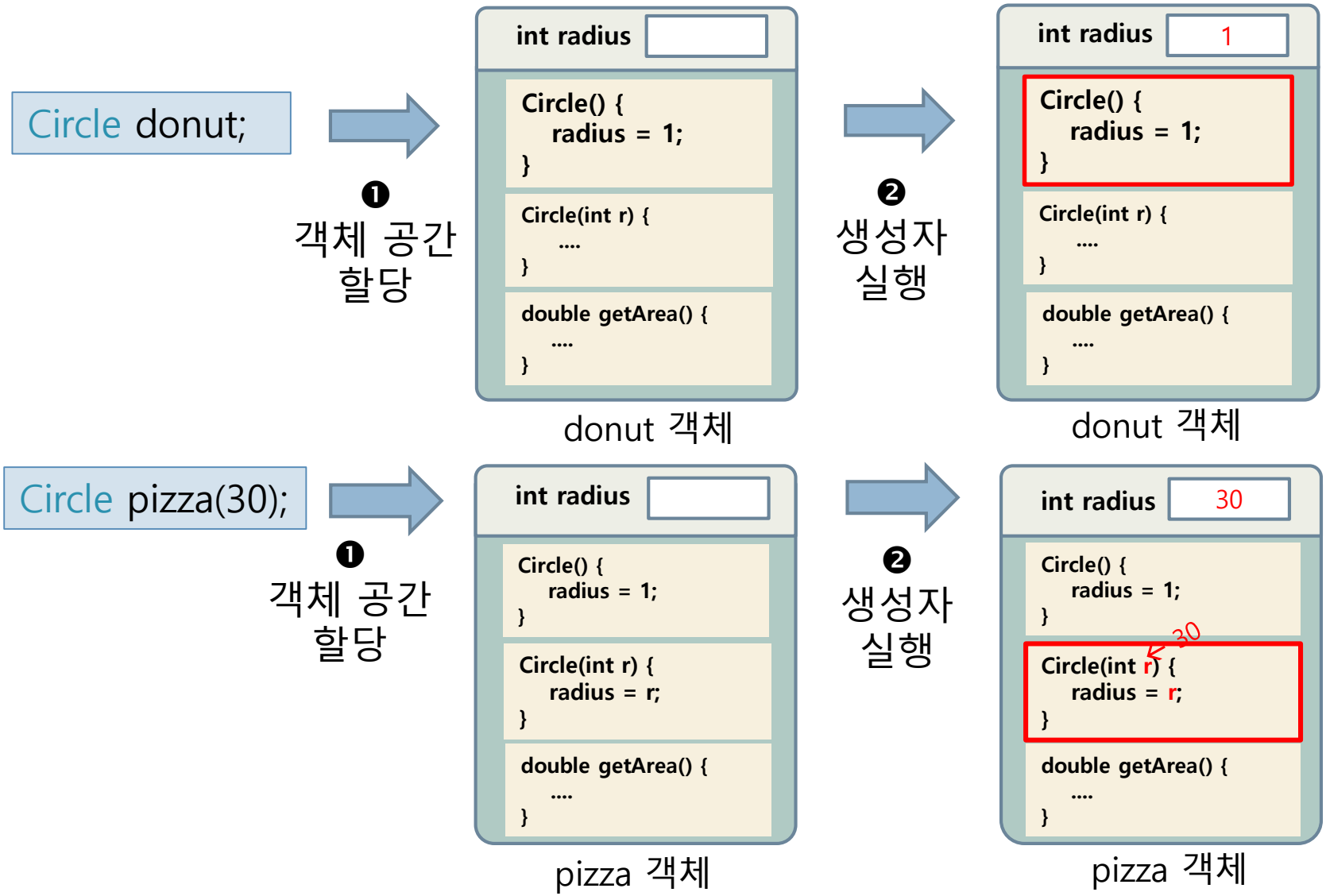
예제 3–3 2개의 생성자를 가진 Circle 클래스


```
int main() {
	Circle donut;			// 생성자 Circle(); 자동호출
	double area = donut.getArea();
	cout << "donut 면적은 " << area << endl;

	Circle pizza(30);		// 생성자 Circle(30); 자동호출
	area = pizza.getArea();
	cout << "pizza 면적은 " << area << endl;
	return 0;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

예제 3–3 2개의 생성자를 가진 Circle 클래스



생성자의 장점

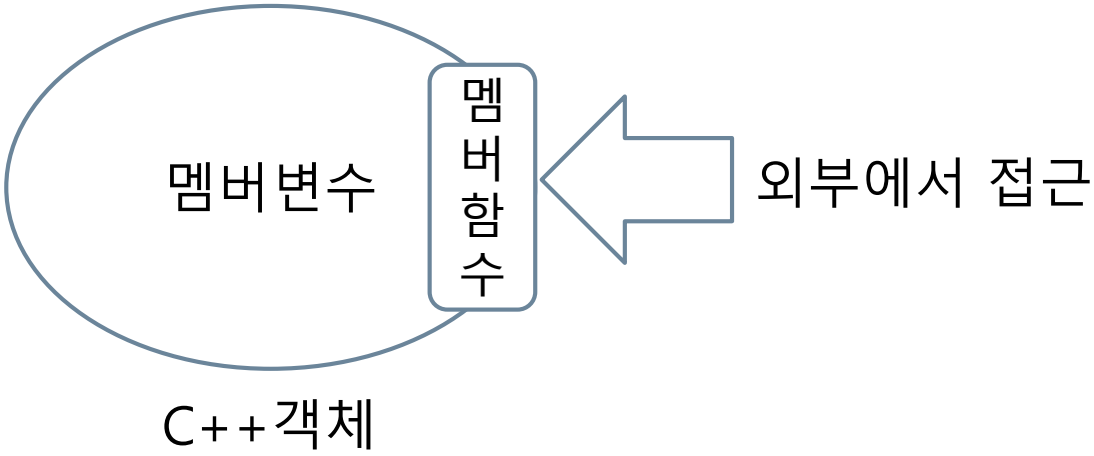
- 멤버변수에 직접 대입할 때 프로그래머의 실수를 막을 수 없으나 생성자를 이용한 초기화는 함수내에서 오류를 방지할 수 있음

error: 반지름은 양수만 가능!

```
// 전과 동일
Circle::Circle(int r) {
	if (r < 0)
	{
		cout << "error: 반지름은 양수만 가능!" << endl;
		radius = 1;
	}
	else radius = r;
}
int main() {
	Circle a;
	a.radius = -10;
	Circle b(-10);
	return 0;
}
```

생성자의 장점

- 객체 내부의 멤버변수에 대한 외부로부터의 직접적인 접근을 차단해줌 -> 캡슐화 또는 정보은닉
- 객체의 모든 멤버 변수는 멤버함수를 통해서만 접근하게 만듦 -> 멤버함수안에서 변수에 대한 잘못된 접근을 방지



위임 생성자와 타켓 생성자

- 여러 생성자에 중복 작성된 코드의 간소화 필요
- 타겟 생성자와 위임 생성자로 나누어 작성
  - 타겟 생성자 : 객체 초기화를 전담하는 생성자
  - 위임 생성자 : 타겟 생성자를 호출하는 생성자, 객체 초기화를 타겟 생성자에 위임

```
Circle::Circle() {
    radius = 1;
    cout << "반지름 " << radius << " 원 생성" << endl;
}

Circle::Circle(int r) {
    radius = r;
    cout << "반지름 " << radius << " 원 생성" << endl;
}
```

코드 중복

위임 생성자와 타켓 생성자

- 생성자중에서 다른 생성자의 기능을 포함하는 것을 타겟 생성자로 선정
- 나머지는 타겟 생성자를 이용하여 구현 -> 위임생성자

```
// 타겟 생성자
Circle::Circle(int r) {
      radius = r;
      cout << "반지름 " << radius << " 원 생성" << endl;
}
// 위임 생성자
Circle::Circle( ) {
      // 타겟 생성자를 호출하여 구현
}
```

위임 생성자와 타켓 생성자

- 위임 생성자는 멤버 초기화 리스트(member initializer list , 콜론부터 몸체 앞까지 영역)에 타겟 생성자를 호출하여 구현
- 생성자 호출되면 멤버 초기화 리스트 실행 -> 함수 몸체 실행

```
클래스명::생성자명(...) : /* 멤버 초기화 리스트 */ {
    /* 함수의 몸체*/
}

// 위임 생성자
Circle::Circle( ) : Circle(1) { }      // 타겟 생성자 호출, 세미콜론 X

// 잘못된 구현
Circle::Circle( ) {
    Circle(1);              // 함수 몸체에서 타겟 생성자 호출하면 안됨
 }
```

예제 3-4 위임, 타겟 생성자 작성


```
#include <iostream>
using namespace std;
class Circle {
public:
	int radius;
	Circle();                    // 위임 생성자
	Circle(int r);               // 타겟 생성자
	double getArea();
};
Circle::Circle() : Circle(1) {  }          // 위임 생성자
Circle::Circle(int r) {                    // 타겟 생성자
	radius = r;
	cout << "반지름 " << radius << " 원 생성" << endl;
}
double Circle::getArea() {
	return 3.14 * radius * radius;
}
```

예제 3-4 위임, 타겟 생성자 작성


```
int main() {
	Circle donut;			// Circle(); 호출
	double area = donut.getArea();
	cout << "donut 면적은 " << area << endl;

	Circle pizza(30);		// Circle(30); 호출
	area = pizza.getArea();
	cout << "pizza 면적은 " << area << endl;
	return 0;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```

다양한 생성자의 멤버 변수 초기화 방법

- 생성자 몸체에서 멤버 변수 초기화

```
class Point {
public:
	int x, y;
	Point();
	Point(int a, int b);
};
Point::Point() { x = 0; y = 0; }
Point::Point(int a, int b) { x = a; y = b; }
```

다양한 생성자의 멤버 변수 초기화 방법

- 멤버 초기화 리스트에서 멤버변수 초기화
  - : 멤버변수명(초기값), 멤버변수명(초기값), ...
  - 생성자 호출되면 멤버 초기화 리스트 -> 함수 몸체 순서로 실행됨
  - 멤버 초기화 리스트에 작성시 실행 속도 빠름

```
Point::Point() : x(0), y(0) {                    // x=0, y=0;
      ...
}
Point::Point(int a, int b) : x(a), y(b) {        // x=a, y=b;
      ...
}
```

예제 3-5 멤버변수의 초기화와 위임 생성자 활용

- 다음 Point 클래스의 멤버 x, y를 멤버 초기화 리스트에 초기값으로 초기화하고 위임 생성자를 이용하여 재작성하라.

```
class Point {
      int x, y;
public:
      Point();
      Point(int a, int b);
      void show() { cout << "(" << x << ", " << y << ")" << endl; }
};
Point::Point() { x = 0; y = 0; }
Point::Point(int a, int b) { x = a; y = b; }
```

예제 3-5 멤버변수의 초기화와 위임 생성자 활용


```
#include <iostream>
using namespace std;
class Point {
	int x, y;
public:
	Point();
	Point(int a, int b);
	void show() { cout << "(" << x << ", " << y << ")" << endl; }
};
Point::Point() : Point(0, 0) { }                    // 위임 생성자
Point::Point(int a, int b) : x(a), y(b) { }         // 타켓 생성자
int main() {
	Point origin;                                   // 생성자 Point() 호출
	Point target(10, 20);                           // 생성자 Point(10,20) 호출
	origin.show();
	target.show();
	return 0;
}
```

```
(0, 0)
(10, 20)
```

예제 3-5 멤버변수의 초기화와 위임 생성자 활용

- 멤버 초기화 리스트에 타겟 생성자 호출과 멤버변수 초기화를 동시에 하면 안됨 -> 위임했는데 또 초기화하는 건 모순

```
#include <iostream>
using namespace std;
class Point {
	int x, y;
public:
	Point();
	Point(int a, int b);
	void show() { cout << "(" << x << ", " << y << ")" << endl; }
};
Point::Point() : Point(0, 0),x(0),y(0) { }	// error
Point::Point(int a, int b) : x(a), y(b) { }	// 타켓 생성자
```

디폴트 생성자(default constructor)

- C++ 컴파일러는 객체가 생성될 때 반드시 하나의 생성자를 호출 -> 생성자가 1개 이상 구현해야 함
- 개발자가 클래스에 생성자를 작성하지 않으면 컴파일러에 의해 디폴트 생성자가 자동으로 추가됨
- 디폴트 생성자는 매개변수가 없고 아무 일도 하지 않는 생성자

```
class Circle {
        .....
        Circle();                   // 컴파일러에 의해 디폴트 생성자 자동생성
};

Circle::Circle() { }                // 매개변수 없고 하는 일 없음
```

디폴트 생성자가 자동으로 생성되는 경우

- 생성자가 하나도 작성되어 있지 않은 클래스의 경우

```
class Circle {
public:
        int radius;
        double getArea();
};
.....
int main() {
        Circle donut;
}
```

(a) 생성자를 작성하지 않음
-> 정상적으로 컴파일됨

```
class Circle {
public:
        int radius;
        double getArea();
        Circle( );
};
Circle::Circle( ){ }

int main() {
        Circle donut;       //디폴트 생성자 호출
        ....
}
```

컴파일러에 의해 자동으로 삽입됨

(b) 컴파일러에 의해 디폴트 생성자 자동 삽입

디폴트 생성자가 자동으로 생성되지 않는 경우

- 생성자가 하나라도 선언된 클래스의 경우
  - 컴파일러는 디폴트 생성자를 자동 생성하지 않음

```
class Circle {
public:
	int radius;
	Circle(int r);
};
Circle::Circle(int r) {
	radius = r;
}
int main() {
	Circle pizza(30);
	Circle donut;		//오류발생 -> 디폴트 생성자가 생성되지 않음
}
```

예제 3-6 Rectangle 클래스 만들기

- 다음 main() 함수가 잘 작동하도록 Rectangle 클래스를 작성하고 프로그램을 완성하라. Rectangle 클래스는 width와 height의 두 멤버 변수와 3 개의 생성자, 그리고 isSquare() 함수를 가진다.

```
int main() {
        Rectangle rect1;
        Rectangle rect2(3, 5);
        Rectangle rect3(3);

        if (rect1.isSquare()) cout << "rect1은 정사각형이다." << endl;
        if (rect2.isSquare()) cout << "rect2는 정사각형이다." << endl;
        if (rect3.isSquare()) cout << "rect3는 정사각형이다." << endl;
        return 0;
}
```

```
rect1은 정사각형이다.
rect3는 정사각형이다.
```

예제 3-6 Rectangle 클래스 만들기


```
#include <iostream>
using namespace std;
class Rectangle {
public:
	int width, height;

	Rectangle();
	Rectangle(int w, int h);
	Rectangle(int length);
	bool isSquare();
};
Rectangle::Rectangle() {
	width = height = 1;
}
Rectangle::Rectangle(int w, int h) {
	width = w; height = h;
}
```

예제 3-6 Rectangle 클래스 만들기


```
Rectangle::Rectangle(int length) {
	width = height = length;
}
// 정사각형이면 true를 리턴하는 멤버 함수
bool Rectangle::isSquare() {
	if (width == height) return true;
	else return false;
}
int main() {
	Rectangle rect1;
	Rectangle rect2(3, 5);
	Rectangle rect3(3);

	if (rect1.isSquare()) cout << "rect1은 정사각형이다." << endl;
	if (rect2.isSquare()) cout << "rect2는 정사각형이다." << endl;
	if (rect3.isSquare()) cout << "rect3는 정사각형이다." << endl;
	return 0;
}
```

```
rect1은 정사각형이다.
rect3는 정사각형이다.
```

실습과제1

- C++에서 생성자를 통하여 멤버변수의 초기화를 하는 이유를 설명하라.
- 생성자 작성시 생성자 몸체에서 멤버를 초기화하는 것 보다 멤버 초기화 리스트에서 초기화하는 이유를 설명하라.
- 위임, 타겟 생성자로 나누어 작성하는 이유를 설명하라
- 함수 몸체(body)에서 생성자를 직접 호출하면 어떻게 되는지 조사해보라

실습과제2

- 삼각형 클래스 Triangle 를 만들고 예제 3-5과 비슷하게 동작하는 프로그램을 작성하시오.
- 위임, 타켓생성자, 멤버 초기화 리스트 방법을 활용하여 작성할 것
- (힌트) 멤버변수 -> 밑변, 높이, 면적-> (1.0/2.0)*밑변*높이, 생성자 2개

```
//클래스 정의추가

int main() {
	Triangle tri1;			// 밑변,높이 1로 초기화
	cout << "삼각형의 면적은 " << tri1.getArea() << endl;

	Triangle tri2(2, 4);		// 밑변=2,높이=4로 초기화
	cout << "삼각형의 면적은 " << tri2.getArea() << endl;
	return 0;
}
```

삼각형의 면적은 0.5
삼각형의 면적은 4

실습과제3

- 구 클래스 Sphere 를 만들고 예제 3-5 와 비슷하게 동작하는 프로그램을 작성하시오.
- 위임, 타켓생성자, 멤버 초기화 리스트 방법을 활용하여 작성할 것
- (힌트) 멤버변수 -> 반지름, 부피-> (4.0/3.0)*3.14*반지름^3, 생성자 2개

```
//클래스 정의추가

int main() {
	Sphere sph1;		// 반지름 =1로 초기화
	cout << "구의 부피는" << sph1.getVolume() << endl;

	Sphere sph2(3);		// 반지름 =3로 초기화
	cout << "구의 부피는" << sph2.getVolume () << endl;
	return 0;
}
```

구의 부피는 3.45
구의 부피는 110.42

실습과제4

- 예제 3-6의 사각형 클래스에 다음 기능을 추가하라.
  - 좌측상단의 좌표 (x, y)를 멤버변수로 추가
  - 사각형의 둘레길이를 구해주는 함수, 사각형의 우측하단의 좌표를 구해주는 함수를 멤버함수로 추가
  - 아래 메인 함수가 실행되도록 적절한 생성자를 추가
  - 생성자는 위임, 타켓 생성자로 나누어 작성하고 멤버 초기화 리스트를 활용

```
//코드추가

int main() {
	Rectangle rect1;                  // 모든 멤버변수를 1로 초기화
	Rectangle rect2(3, 5);            // x=3,y=5,width=height=1로 초기화
	Rectangle rect3(3, 5, 2, 4);      // x=3,y=5,width=2, height=4로 초기화
	//코드추가
}
```

rect1의 면적은 1
rect2의 둘레길이는 4
rect3의 우측하단의 좌표는 (5, 1)

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ************************************************
//    제   목 :  정수 4개의 평균을 구하는 프로그램
//    날   짜 :  2023년 9월10일
//    작성자 :  15010101 홍길동
// ************************************************

// 소스코드 작성
```