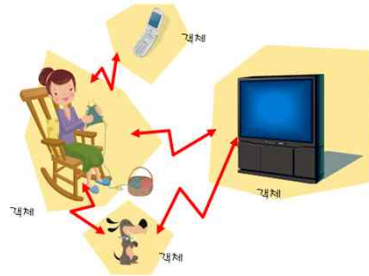


## 제6장 생성자와 소멸자



1/40

## 이번 장에서 학습할 내용

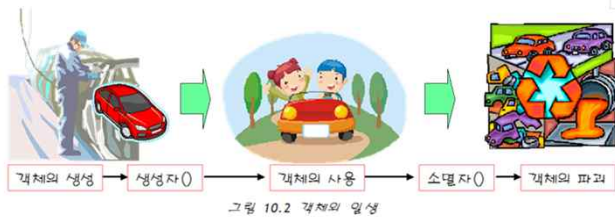
- 생성자
- 소멸자
- 초기화 리스트
- 복사 생성자 (개념만)
- 디폴트 멤버 함수
- 예제: 이차방정식
- 예제: 구구단 게임 및 랭킹

객체가 생성될 때 초기화를 담당하는 생성자에 대하여 살펴봅니다.

2/40

## 생성자

- 생성자(contructor)
  - 객체가 생성될 때에 필드에게 초기값을 제공하고 필요한 초기화 절차를 실행하는 멤버 함수



3/40

## 생성자의 특징

- 클래스 이름과 동일하다
- 반환값이 없다.
- 반드시 **public** 이어야 한다.
- 중복 정의할 수 있다.

```
class Car
{
    ...
public:
    Car()
    {
        ...
    }
};
```

생성자

4/40

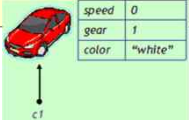
## 디폴트 생성자

```
#include <iostream>
#include <string>
using namespace std;
class Car {
private:
    int speed;    // 속도
    int gear;
    string color; // 색상
public:
    Car() {
        cout << "디폴트 생성자 호출" << endl;
        speed = 0;
        gear = 1;
        color = "white";
    }
};

int main() {
    Car c1; // 디폴트생성자호출
    return 0;
}
```

// 외부 정의: Car.cpp에서 생성자를 구현

```
Car::Car() {
    cout << "디폴트 생성자 호출" << endl;
    speed = 0;
    gear = 1;
    color = "white";
}
```



speed	0
gear	1
color	"white"

5/40

## 매개 변수를 가지는 생성자

```
#include <iostream>
#include <string>
using namespace std;


class Car {
private:
    int speed;    // 속도
    int gear;      // 기어
    string color; // 색상
public:
    Car(int s, int g, string c) {
        speed = s;
        gear = g;
        color = c;
    }
    void print()
    {
        cout << "===== " << endl;
        cout << "속도: " << speed << endl;
        cout << "기어: " << gear << endl;
        cout << "색상: " << color << endl;
        cout << "===== " << endl;
    }
};
```

6/40

## 매개 변수를 가지는 생성자

```
int main()
{
    Car c1(0, 1, "red"); // 생성자 호출
    Car c2(0, 1, "blue"); // 생성자 호출
    c1.print();
    c2.print();
    return 0;
}
```

```
=====
속도: 0
기어: 1
색상: red
=====
속도: 0
기어: 1
색상: blue
=====
```



speed	0
gear	1
color	"red"

speed	0
gear	1
color	"blue"

7/40

## 생성자의 중복 정의

- 생성자도 메소드이므로 중복 정의가 가능하다.

```
#include <iostream>
#include <string>
using namespace std;

class Car {
private:
    int speed;    // 속도
    int gear;      // 기어
    string color; // 색상
public:
    Car();
    Car(int s, int g, string c) ;
};
```

8/40

## 생성자의 중복 정의

```
Car::Car() {
    cout << "디폴트 생성자 호출<< endl;
    speed = 0;
    gear = 1;
    color = "white"
}
Car::Car(int s, int g, string c) {
    cout << "매개변수가 있는 생성자 호출<< endl;
    speed = s;
    gear = g;
    color = c;
}
void main() {
    Car c1;           // 디폴트 생성자 호출
    Car c2(100, 0, "blue"); // 매개변수가 있는 생성자 호출
}
```

디폴트 생성자 호출  
매개 변수가 있는 생성자 호출  
계속하려면 아무 키나 누르십시오...

9/40

## 생성자 호출의 다양한 방법

```
void main()
{
    Car c1;           // ①디폴트 생성자 호출
    Car c2();          // ②생성자 호출이 아니라 c2()라는 함수원형 선언
    Car c3(100, 3, "white"); // ③생성자 호출
    Car c4 = Car(0, 1, "blue"); // ④이것은 먼저 임시 객체를 만들고 이것을 c4에 복사
}
```

10/40

## 생성자를 하나도 정의하지 않으면?

```
class Car {
    int speed; // 속도
    int gear;  // 기어
    string color; // 색상
};
```

컴파일러가 비어있는 디폴트 생성자를 자동으로 추가한다.

```
class Car {
    int speed; // 속도
    int gear;  // 기어
    string color; // 색상
public:
    Car() { }
}
```

11/40

## 디폴트 매개 변수

```
Car(int s=0, int g=1, string c="red")
{
    speed = s;
    gear = g;
    color = c;
}
```

디폴트 생성자를 정의한 것과 같은 효과를 낸다.

12/40

## 생성자에서 다른 생성자 호출하기

```
...
class Car {
    int speed; // 속도
    int gear; // 기어
    string color; // 색상
public:
    // 첫 번째 생성자
    Car(int s, int g, string c) {
        speed = s;
        gear = g;
        color = c;
    }
    // 색상만 주어진 생성자
    Car(string c) {
        Car(0, 0, c); // 첫 번째 생성자를 호출한다.
    }
};
int main()
{
    Car c1("white");
    return 0;
}
```

13/40

## 중간 점검 문제

1. 만약 클래스 이름이 **MyClass**라면 생성자의 이름은 무엇이어야 하는가?
2. 생성자의 반환형은 무엇인가?
3. 생성자는 중복 정의가 가능한가?
4. 클래스 안에 생성자를 하나도 정의하지 않으면 어떻게 되는가?

14/40

## 복소수 클래스의 생성자

- `Complex()` { `set(0.0, 0.0);` }
- `Complex(double r, double i)` { `set(r, i);` }
- `Complex(double r=0, double i=0)` { `real = r; imag = i;` }
- 멤버 초기화 리스트
  - `Complex(double r=0, double i=0): real(r), imag(i) { }`

15/40

## 이런 프로그램이 가능한가?

```
#include "MyGameEx.h"

void main()
{
    MyGameEx x;
}
```

```
#include "MyGameEx.h"
MyGameEx x;

void main()
{
}
```

16/40

## 소멸자

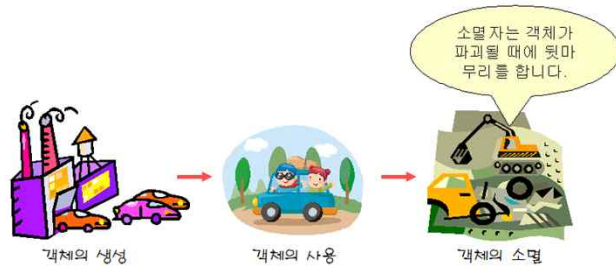


그림 10.4 소멸자의 개념

17/40

## 소멸자의 특징

- 소멸자는 클래스 이름에 ~가 붙는다.
- 값을 반환하지 않는다.
- **public** 멤버 함수로 선언된다.
- 소멸자는 매개 변수를 받지 않는다.
- 중복 정의도 불가능하다.

```
class Car
{
    ...
public:
    ~Car()
    {
        ...
    }
};
```

소멸자

18/40

## 소멸자

```
class Car {
private:
    int speed;           // 속도
    int gear;            // 주행거리
    string color;        // 색상
public:
    Car() {
        cout << "생성자 호출" << endl;
        speed = 0;
        gear = 1;
        color = "white";
    }

    ~Car() {
        cout << "소멸자 호출" << endl;
    }
};

int main() {
    Car c1;
    return 0;
}
```

생성자 (Callout for Car constructor)

소멸자 (Callout for Car destructor)

생성자 호출 (Callout for main creating c1)

소멸자 호출 (Callout for main returning 0)

19/40

## 디폴트 소멸자

- 만약 프로그래머가 소멸자를 정의하지 않았다면 어떻게 되는가?
- 디폴트 소멸자가 자동으로 삽입되어서 호출된다

```
class Time {
    int hour, minute, second;
public:
    print() { ... }
}
```

~Time()을 넣어준다.

20/40

## 중간 점검 문제

1. 만약 클래스 이름이 **MyClass**라면 소멸자의 이름은 무엇이어야 하는가?
2. 소멸자의 반환형은 무엇인가?
3. 소멸자는 중복 정의가 가능한가?
4. 복소수 클래스에서 소멸자 정의가 필요한가?
5. 이차방정식 클래스에서 소멸자 정의가 필요한가?
6. 랭킹 클래스에서 소멸자 정의는?
  - 랭킹 리스트의 파일 저장?

21/40

## 멤버 초기화 목록

- 멤버 변수를 간단히 초기화할 수 있는 형식

```
Car(int s, int g, string c) : speed(s), gear(g), color(c) {
    ...// 만약 더 하고 싶은 초기화가 있다면 여기에
}
```

22/40

## 상수 멤버의 초기화

- 멤버가 상수인 경우에는 어떻게 초기화하여야 하는가?

```
class Car
{
    const int MAX_SPEED = 300;
    int speed;
    ...
}
```

← 아직 생성이 안됐음!

```
class Car
{
    const int MAX_SPEED;
    int speed;    // 속도
public:
    Car()
    {
        MAX_SPEED = 300;
    }
}
```

← 상수를 변경할 수 없음!

23/40

## 상수 멤버의 초기화

```
class Car
{
    const int MAX_SPEED;
    int speed;    // 속도
public:
    Car() : MAX_SPEED(300)
    {
    }
};
```

상수 멤버의 초기화는 이렇게.

24/40

## 참조자 멤버의 초기화

```
#include <iostream>
#include <string>
using namespace std;
class Car {
    string& alias;
    int speed;    // 속도
public:
    Car(string s) : alias(s) {
        cout << alias << endl;
    }
};

int main() {
    Car c1("꿈의 자동차");
    return 0;
}
```

꿈의 자동차  
계속하려면 아무 키나 누르십시오 ...

25/40

## 객체 멤버의 초기화

```
#include <iostream>
#include <string>
using namespace std;

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

class Rectangle
{
    Point p1, p2;
public:
    Rectangle(int x1, int y1, int x2, int y2) : p1(x1, y2), p2(x2, y2) {}
};
```

생성자 호출

26/40

## 중간 점검 문제

1. 초기화 리스트를 반드시 사용하여서 초기화해야 되는 멤버의 타입은?
2. 클래스 MyClass의 상수 limit를 초기화 리스트를 사용하여서 초기화 하여 보라.
3. Complex의 데이터 멤버를 초기화 리스트를 사용하여서 초기화 하여 보라.
4. QuadFn의 데이터 멤버를 초기화 리스트를 사용하여서 초기화 하여 보라.  
r1과 r2는 0으로, nSol도 0으로 초기화 할 것

27/40

## 이차방정식 클래스의 생성자

- 디폴트 매개변수를 이용한 생성자

```
// QuadFn클래스의 다양한 생성자
// QuadFn( )
// QuadFn( double aa )
// QuadFn( double aa, double bb )
// QuadFn( double aa, double bb, double cc )
QuadFn( double aa=0.0, double bb=0.0, double cc=0.0 ) {
    a = aa;
    b = bb;
    c = cc;
    nSol = 0;
    r1.set(0.0, 0.0);
    r2.set(0.0, 0.0);
}
```

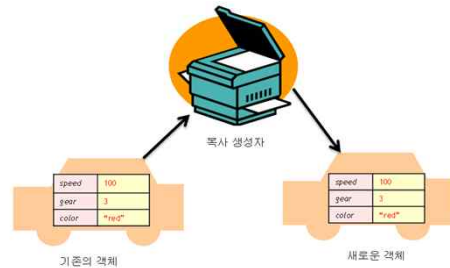
- 멤버 초기화 리스트 사용

```
QuadFn( double aa=0.0, double bb=0.0, double cc=0.0 )
    : a(aa), b(bb), c(cc), nSol(0),
      r1(0.0, 0.0), r2(0.0, 0.0) {}
```

28/40

## 복사 생성자

- 한 객체의 내용을 다른 객체로 복사하여서 생성
- 포인터 이후에 다시 공부할 예정



29/40

## 디폴트 멤버 함수

- 디폴트 생성자
- 디폴트 소멸자
- 디폴트 복사 생성자
- 디폴트 대입 연산자

자동으로 추가된다.  
만약 생성자가 하나라도 정의되어 있다면???

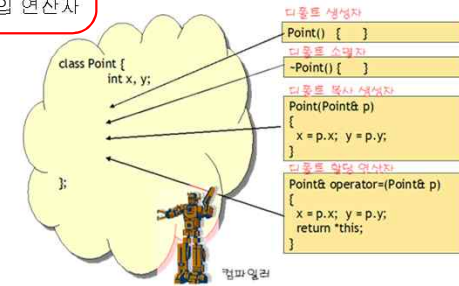
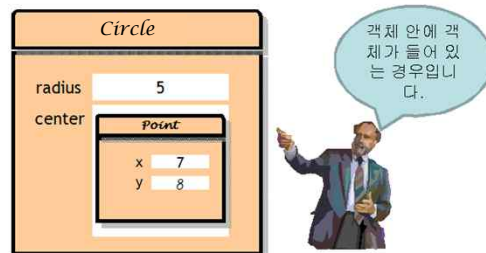


그림 10.7 디폴트 멤버 함수의 추가

30/40

## 객체 안의 객체

- Circle 객체 안에 Point 객체가 들어 있는 경우



31/40

## 예제: Point 와 Circle

```
#include <iostream>
#include <string>
using namespace std;

class Point {
private:
    int x;
    int y;
public:
    Point() {}
    Point(int a, int b) : x(a), y(b) {}
    void print() {
        cout << "(" << x << ", " << y << " )\n";
    }
};
```

32/40



```

class Circle {
private:
    int radius;
    Point center; // Point 객체가 멤버변수로 선언되어 있다.
public:
    Circle() : radius(0), center(0, 0) {}
    Circle(int r) : radius(r), center(0, 0) {}
    Circle(Point p, int r) : radius(r), center(p) {}
    Circle(int x, int y, int r) : radius(r), center(x, y) {}

    void print() {
        cout << "중심: ";
        center.print();
        cout << "반지름: " << radius << endl << endl;
    }
};

```

33/40

```

int main()
{
    Point p(5, 3);

    Circle c1;
    Circle c2(3);
    Circle c3(p, 4);
    Circle c4(9, 7, 5);

    c1.print();
    c2.print();
    c3.print();
    c4.print();
    return 0;
}

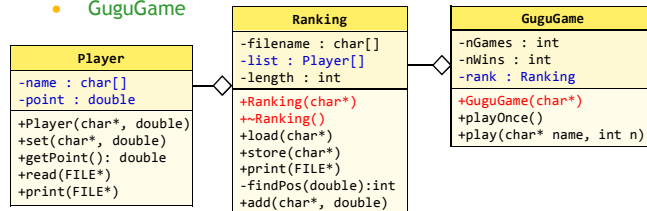
```

중심: (0, 0)  
반지름: 0  
중심: (0, 0)  
반지름: 3  
중심: (5, 3)  
반지름: 4  
중심: (9, 7)  
반지름: 5

34/40

## 예제: 구구단 게임과 랭킹

- UML 클래스 다이어그램
  - Player
  - Ranking
  - GuguGame



35/40

## 구구단 게임: 선수 클래스

```

class Player
{
    char name[80]; // 선수 이름
    double point; // 선수의 포인트
public:
    // 생성자: 매개변수 있음. 디폴트 매개변수 사용
    Player(char* na="none", double pt=0.0) {
        // 소멸자: 잘 알 없음
        ~Player() {}

        double getPoint() { return point; }
        void set(char* na, double pt) {
            ...
        }
        void print(FILE* fp=stdout) {
            ...
        }
        void read(FILE* fp) {
            ...
        }
};

```

RANKING	NAME	AGE	POINTS	TOURN
1	David Salazar	28	10.88	17
2	Andy Murray	28	10.75	18
3	Keigo Nishikori			
4	Stan Wawrinka			
5	Grigor Dimitrov			
6	Sam Querrey			
7	Tommy Haas			
8	David Ferrer			
9	Julien Benneteau	30	11.00	23
10	Nicola Pietrangeli	29	10.80	20

강감찬 2.789  
태양의후예 3.887  
홍길동 4.045

객체의 복사? c = a; → OK !! (디폴트 대입 연산자)  
배열의 복사? arr = list; → NO !! (반복문 또는 memcpy등)  
문자열의 복사? name = na; → NO !! (반복문 또는 strcpy등)

36/40

## 구구단 게임: 랭킹 클래스

```
#define MaxTopPlayer 10
// const int MaxTopPlayer = 10;

class Ranking
{
    char    filename[80]; // 랭킹 파일 이름
    Player  list[MaxTopPlayer]; // 선수 리스트
    int     length; // 등록된 선수수

public:
    Ranking(char* fname) { load(fname); }
    ~Ranking(void) { store(filename); }

    void load( char *fname ) { ... }
    void store( char *fname = NULL ) { ... }
    void print( FILE *fp=stdout ) { ... }

    int findPosition( double point ) { ... }
    void add( char* name, double pt ) { ... }
};
```

Ranking
-filename : char[]
-list : Player[]
-length : int
+Ranking(char*)
+~Ranking()
+load(char*)
+store(char*)
+print(FILE*)
-findPos(double):int
+add(char*, double)

디폴트 생성자가 있는가?

Ranking rank1; → NO!!!  
Ranking rank2("GoMasters.txt"); → Yes

37/40

## 구구단 게임: 게임 클래스

```
#include <ctime>
#include "Ranking.h"

inline int getNumber(int from=1, int to=9) {
    return rand()%(to-from+1) + from;
}

class GuguGame
{
    int    nGames;
    int    nWins;
    Ranking rank;

    void reset() { nGames = 0; nWins=0; }

public:
    GuguGame(char* rankfile = "rank.txt") : rank(rankfile) { reset(); }
    ~GuguGame(void) {}

    // 한번의 구구단 게임을 하는 함수
    void playOnce( ) { ... }

    // 선수 이름과 횟수를 입력으로 받음
    void play( char* player, int n=1 ) { ... }
};
```

GuguGame
-nGames : int
-nWins : int
-rank : Ranking
+GuguGame(char*)
+playOnce()
+play(char* name, int n)

멤버 초기화 리스트를 사용하지 않으면???

Ranking rank; → 초기화 불가능!!!

38/40

## 구구단 게임: 주 프로그램

```
#include <conio.h> // getch()
#include "GuguGame.h"

void main()
{
    GuguGame gugu;
    int n;
    char name[80];

    srand( (unsigned)time(NULL) ); // 실행마다 다른 무작위 숫자 발생

    printf("*** 구구단 게임에 오신것을 환영합니다!!! ***\n\n");
    printf("*** 선수의 이름을 입력하세요 ==> ");
    scanf("%s", &name);

    printf("몇 번 테스트하시겠습니까? ");
    scanf("%d", &n);
    fflush(stdin);

    while (true) {
        gugu.play( name, n );

        printf("다시 하시겠습니까? (y/n) ");
        if( getch() != 'y' )
            break;
    }
    printf("\n\n구구단 게임을 종료합니다!!!\n\n");
}
```

39/40

## 예제: 숫자 맞추기 게임

```
void main()
{
    NumberGame game1();
    NumberGame game2(50, 10);

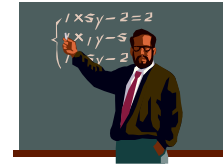
    game1.play();
    game2.play();
}
```

40/40

예제: 윗놀이

41/40

Q & A



42/40