Chap5

값에 의한 호출
- 인자값을 복사해 매개변수에 넘겨줌
- 호출한 곳에서는 변화가 없음

주소에 의한 호출
- 인자의 주소를 포인터 타입의 매개변수에 전달
- 함수 호출 후 값이 변경되면 원본값도 변경

```
void swap(int *a){ ...}

int main( ){
    int m = 2;
    swap(&m); }
```

'값에 의한 호출'으로 객체 전달의 문제점
- 객체 복사 시간↑
- 값에 의한 호출 시, 매개변수에 생성자호출 없이 인자의 객체가 그대로 복사된다
  반면에 소멸자는 또 실행된다
  그래서 생성자는 실행X, 소멸자는 실행o 인 비대칭구조

*생성자x 이유 -> 복사된 내용 바뀔까봐...

'주소에 의한 호출'로 객체 전달
- 주소에 의한 호출은 매개변수에 인자의 주소값만 전달되기 때문에 시간적 효율도 좋고, 생성자/소멸자 비대칭 문제도X

객체 치환
- 객체의 모든 데이터가 비트 단위로 복사(복사본)
- 동일한 클래스 타입끼리만 가능

함수 객체 리턴
- 객체의 복사본 리턴

참조변수
- 이미 선언된 변수(원본 변수)에 대한 별명
- 참조자(&) 를 이용해 선언하고 바로 원본변수로 초기화 필수

```
int n = 3;
int &refn = n;
```

- 참조변수는 이름만 생성(별도공간 x)
- 원본변수의 공간만을 공유(같은 공간 가르킴)
- 참조변수에 = 로 값을 변경할 수 있다(참조변수는 포인터 X)
- 참조변수/원본 변수 값이 변하면 같이 다 변함(공간공유)

참조변수에 대한 포인터

```
int *p = &refn; (<- &n과 같음)
    *p = 20;
```

참조변수에 대한 참조 선언

```
int &r = refn;  (참조변수 r에 참조변수 refn의 값 들어감)
                (refn 에는 n의 주소가 저장)
```

참조에 의한 호출
- 매개변수에서 실인자를 참조하여 실인자와 공간 공유

```
void swap(int &a, int &){ ...}]

int main(){
  int m = 3;
  int n = 5;
  swap(m,n); }
```

참조리턴
- 현존하는 공간을 리턴하는 것

```
char c = 'a';          -> 문자형 변수 c는 a로 저장
   char& find( ) {.
        return c; }.     -> c에 대한 공간 반환

   char a = find( );      -> find()가 치환문 오른쪽에 오면
                            변수 c의 값이 a에 치환
   char &ref = find( );  -> ref도 c의 공간 가르킴
   ref = 'M';            -> ref가 가르키고 있는 공간이 M
   find( ) = 'b';        -> c의 공간에 b
```

* 얕은복사: 포인터의 경우 원본이 할당받은 메모리 공유
  (원본 변경 가능성...)
* 깊은복사: 모든 멤버가 원본이 가진 메모리 크기만큼 동적할당

복사 생성자
- 객체 생성시, 원본 객체를 복사하여 생성되는 경우 실행되는 생성자
- 매개변수는 오직 하나(자기 클래스에 대한 참조로 선언)

```
Circle(Circle &c) {
    this->radius = c.radius; }
int main( ){
    Circle src();
    Circle dest(scr); }
```

디폴트 복사 생성자

- 클래스 멤버로 복사 생성자를 갖고 있지 않는 경우 컴파일러는 얕은복사를 하게끔 디폴트 복사 생성자를 작성해 원본 객체의 각 멤버를 사본(this)에 복사하도록 만듦

- 포인터 타입의 멤버 변수의 경우, 메모리가 공유되면서 문제 발생

```
char *name;  (클래스에서 선언된 포인터멤버변수)

Person :: Person(Person &p) {
  this->name = p.name; (복사된 객체의 name의 주소)
}                       -> 메모리 공유
```

묵시적 복사 생성자

- 객체 생성 시, 생성자를 통해서 명시적으로 복사할 객체 적는것이 아닌 나도 모르게 복사생성자 호출되는 경우..

1) 객체로 초기화하여 객체 생성

```
Person son = father;
=> Person son(father); 으로 바뀜...

* Person son;
  son = father;   (치환문과는 다르다!!!!)
```

2) 값에 의한 호출로 객체가 전달될 때

생성자 실행 안되고, 디폴트 복사 생성자가 실행 (그럼 얕은 복사?...)되어 모든 멤버들의 복사본을 전달한다.....

3) 함수가 객체를 리턴시

객체 리턴도 복사본 전달이므로 복사 생성자 호출