



고급프로그래밍

복사생성자
함수중복

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory
Kwangwoon University
(02-940-5625, mhjeong@kw.ac.kr)

실습 1

- 다음 결과가 나오도록 half() 함수를 작성하시오(단, 참조변수 활용).

```
int main( )  
{  
    double a=20.0;  
    half(a);  
    cout << a;  
}
```

10

실습 1 - 답

- 다음 결과가 나오도록 half() 함수를 작성하시오.

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

void half(double& a)
{
    a /= 2.0;
}

int main( )
{
    double a=20.0;
    half(a);
    cout << a;
}
```

실습 2

- 다음 결과가 나오도록 `increaseBy()`를 추가하시오.

```
class Circle
{
private:
    int radius;
public:
    Circle();
    Circle(int r);
    ~Circle() { }
    int  getRadius() { return radius; }
    void setRadius(int radius) { this->radius = radius; }
    void show( ) {cout << "radius = "<< radius<< endl; }
};
Circle::Circle(): Circle(1){ }
Circle::Circle(int r) {
    radius = r;
    cout << "Constructor : " << radius << endl;
}
```

```
int main()
{
    Circle x(10), y(5);
    increaseBy(x, y);
    x.show( );
}
```

```
radius = 15;
```

실습 2 - 답


- 다음 결과가 나오도록 `increaseBy()`를 추가하시오.

```
class Circle
{
private:
    int radius;
public:
    Circle();
    Circle(int r);
    ~Circle() { }
    int  getRadius() { return radius; }
    void setRadius(int radius) { this->radius = radius; }
    void show( ) { cout << "radius = " << radius << endl; }
};
Circle::Circle(): Circle(1){ }
Circle::Circle(int r) {
    radius = r;
    cout << "Constructor : " << radius << endl;
}
```

```
void increaseBy(Circle& x, Circle& y)
{
    x.setRadius(x.getRadius() +
                y.getRadius());
}

int main()
{
    Circle x(10), y(5);
    increaseBy(x, y);
    x.show( );
}
```

Schedule

week	Topics		Homework	Quiz
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++ 	C++ 프로그래밍의 기본(3, 3/12), 클래스와 객체(4, 3/14)	1	1
3		휴강(3/17), 객체생성과 사용(5, 3/19)	2	
4		함수와 참조(6, 3/26), 복사 생성자와 함수중복(7. 3/28)	3	2, 3
5		static, friend, 연산자 중복, 상속가상함수와 추상클래스	4	4
6		템플릿과 STL, 표준 입출력	5	5
7		파일 입출력		
8	중간고사			
9	C++	예외처리 및 C 사용, 람다식	6	6
10		멀티스레딩	7	7
11		멀티스레딩, 고급문법	8	8
12		고급문법	9	9
13	병렬 프로그래밍	병렬프로그래밍		
14		병렬프로그래밍		
15	기말고사			

오늘의 학습내용

- 복사 생성자
- 함수 중복

복사 생성자

C++에서 얕은 복사와 깊은 복사

■ 얕은 복사(shallow copy)

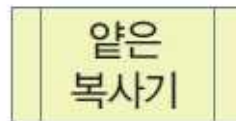
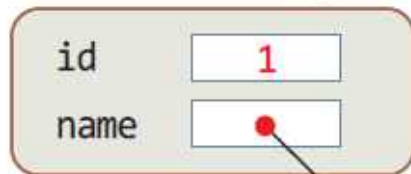
- 객체 복사 시, 객체의 멤버를 1:1로 복사
- 객체의 멤버 변수에 동적 메모리가 할당된 경우
 - 사본은 원본 객체가 할당 받은 메모리를 공유하는 문제 발생

■ 깊은 복사(deep copy)

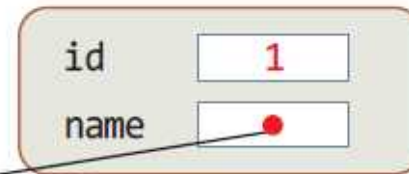
- 객체 복사 시, 객체의 멤버를 1:1대로 복사
- 객체의 멤버 변수에 동적 메모리가 할당된 경우
 - 사본은 원본이 가진 메모리 크기 만큼 별도로 동적 할당
 - 원본의 동적 메모리에 있는 내용을 사본에 복사

```
class Person {
    int id;
    char *name;
    .....
};
```

Person 타입 객체, 원본



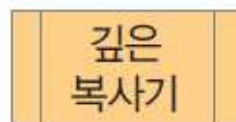
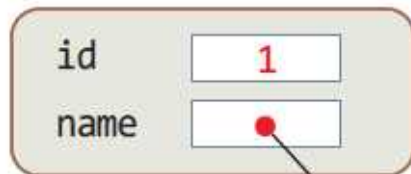
복사본 객체



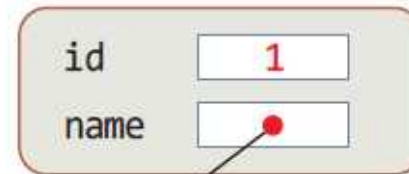
(a) 얕은 복사

name 포인터가 복사되었기 때문에 메모리 공유! - 문제 유발

Person 타입 객체, 원본



복사본 객체



(b) 깊은 복사

name 포인터의 메모리도 복사되었음

복사 생성자

- 복사 생성자(copy constructor)란?
 - 객체의 복사 생성시 호출되는 특별한 생성자
- 특징
 - 한 클래스에 오직 한 개만 선언 가능
 - 복사 생성자는 보통 생성자와 클래스 내에 중복 선언 가능
 - 클래스에 대한 참조 매개 변수를 가지는 독특한 생성자
- 복사 생성자 선언

```
class Circle {  
    .....  
    Circle(Circle& c); // 복사 생성자 선언  
    .....  
};  
  
Circle::Circle(Circle& c) { // 복사 생성자 구현  
    .....  
}
```

자기 클래스에 대한
참조 매개변수

예제 – Circle의 복사 생성자

```
class Circle {  
private:  
    int radius;  
public:  
    Circle(Circle& c); // 복사 생성자 선언  
    Circle() { radius = 1; }  
    Circle(int radius) { this->radius = radius; }  
    double getArea() { return 3.14*radius*radius; }  
};  
  
Circle::Circle(Circle& c) { // 복사 생성자 구현  
    this->radius = c.radius;  
    cout << "Copy constructor called; radius = " << radius << endl;  
}  
  
int main() {  
    Circle src(30); // src 객체의 보통 생성자 호출  
    Circle copy(src); // copy 객체의 복사 생성자 호출  
  
    cout << "Source area = " << src.getArea() << endl;  
    cout << "Copy area = " << dest.getArea() << endl;  
}
```

```
Copy constructor; radius = 30  
Source area = 2826  
Copy area = 2826
```

디폴트 복사 생성자

- 복사 생성자가 선언되어 있지 않는 클래스
 - 컴파일러는 자동으로 디폴트 복사 생성자 삽입

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

복사 생성자 없음

```
Circle src(10);  
Circle cpy(src);    // 복사 생성. Circle(Circle&) 호출  
Circle cpy2 = src; // 선언시 '='은 Circle(Circle&) 호출  
Circle(Circle&) 호출
```

```
Circle::Circle(Circle& c) {  
    this->radius = c.radius;  
}
```

디폴트 복사 생성자

디폴트 복사 생성자 사례

복사 생성자가 없는 Book 클래스

```
class Book {  
    double price; // 가격  
    int pages;    // 페이지수  
    char *title;  // 제목  
    char *author; // 저자이름  
public:  
    Book(double pr, int pa, char* t, char* a);  
    ~Book()  
};
```

컴파일러가 삽입하는
디폴트 복사 생성자

```
Book(Book& book) {  
    this->price = book.price;  
    this->pages = book.pages;  
    this->title = book.title;  
    this->author = book.author;  
}
```

예제 - 얇은 복사로 인한 비정상 종료

```
class Person { // Person 클래스 선언
    char* name;
    int id;
public:
    Person(int id, const char* name);
    ~Person();
    void changeName(const char *name);
    void show() { cout << id << ' ' << name << endl; }
};
```

```
Person::Person(int id, const char* name) { // 생성자
    this->id = id;
    int len = strlen(name);                // name의 문자 개수
    this->name = new char [len+1];         // name 문자열 공간 할당
    strcpy(this->name, name);              // name에 문자열 복사
}
Person::~~Person() { // 소멸자
    if(name)                               // 만일 name에 동적 할당된 배열이 있으면
        delete [] name;                   // 동적 할당 메모리 반환
}
void Person::changeName(const char* name) {
    if(strlen(name) > strlen(this->name))
        return;
    strcpy(this->name, name);
}
```

컴파일러에 의해
디폴트 복사 생성자 삽입

```
Person::Person(Person& p) {
    this->id = p.id;
    this->name = p.name;
}
```

예제 - 얇은 복사로 인한 비정상 종료

```
int main()
{
    Person father(1, "Kitae");    // father 객체 생성
    Person daughter(father);      // daughter 객체 복사 생성. 복사생성자호출

    cout << "daughter object created ----" << endl;
    father.show();
    daughter.show();

    daughter.changeName("Grace"); // daughter의 이름을 "Grace"로 변경
    cout << "name changed to Grace ----" << endl;
    father.show();
    daughter.show();

    return 0;                    //daughter, father 객체 소멸
}
```

컴파일러가 삽입한
디폴트 복사 생성자 호출

daughter, father 순으로 소멸.
father가 소멸할 때, 프로그램
비정상 종료됨

예제 - 깊은 복사 생성자로 수정

```
class Person { // Person 클래스 선언
    char* name;
    int id;
public:
    Person(int id, const char* name);
    Person(Person& person);          // 복사 생성자
    :
};

:

Person::Person(Person& person)
{
    this->id = person.id; // id 값 복사

    int len = strlen(person.name);    // name의 문자 개수
    this->name = new char [len+1];    // name을 위한 공간 할당
    strcpy(this->name, person.name); // name의 문자열 복사

    cout << "Copy constructor called " << this->name << endl;
}

:
```

예제 - 묵시적 복사 생성

```
void f(Person person) {  
    person.changeName("dummy");  
}
```

2. '값에 의한 호출'로 객체가 전달될 때.
person 객체의 복사 생성자 호출

```
Person g() {  
    Person mother(2, "Jane");  
    return mother;  
}
```

3. 함수에서 객체를 리턴할 때.mother 객체의
복사본 생성. 복사본의 복사 생성자 호출안됨

```
int main()  
{  
    Person father(1, "Kitae");  
    Person son = father;  
    f(father);  
    g();  
}
```

1. 객체로 초기화하여 객체가 생성될 때.
son 객체의 복사 생성자 호출

```
복사 생성자 실행 Kitae  
복사 생성자 실행 Kitae  
복사 생성자 실행 Jane
```

실습 1

- 프로그램이 아래와 같이 수행되도록 Book 클래스의 생성자, 소멸자, set() 함수를 작성하시오.

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

class Book
{
    char *title;
    int price;

public:
    Book(const char* title, int price);
    Book(Book& b);
    ~Book();
    void set(const char* title, int price);
    void show() { cout << title << ' ' << price << "Won" << endl; }
};
```

```
int main()
{
    Book cpp("C++", 10000);
    Book java = cpp; // java(cpp)
    java.set("Java", 12000);
    cpp.show();
    java.show();
}
```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe

```
C++ 10000Won
Java 12000Won
Press <RETURN> to close this window...
```

함수 중복

함수 중복

■ 함수 중복

- 동일한 이름의 함수가 공존
 - 다형성
 - C 언어에서는 불가능
- **function overloading**
- 함수 중복이 가능한 범위
 - 보통 함수들 사이
 - 클래스의 멤버 함수들 사이
 - 상속 관계에 있는 기본 클래스와 파생 클래스의 멤버 함수들 사이

■ 함수 중복 성공 조건

- 중복된 함수들의 이름 동일
- 중복된 함수들의 매개 변수 타입이 다르거나 개수가 달라야 함
- 리턴 타입은 함수 중복과 무관

함수 중복 성공 사례

```
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;  
}
```

```
int main(){  
    cout << sum(2, 5, 33);  
  
    cout << sum(12.5, 33.6);  
  
    cout << sum(2, 6);  
}
```

중복된 sum() 함수 호출.
컴파일러가 구분

성공적으로 중복된 sum() 함수들

함수 중복 실패 사례

- 리턴 타입은 함수 중복과 무관

```
int sum(int a, int b) {  
    return a + b;  
}  
  
double sum(int a, int b) {  
    return (double)(a + b);  
}
```

```
int main()  
{  
    cout << sum(2, 5);  
}
```

함수 중복 실패

?

컴파일러는 어떤 sum() 함수를 호출하는지 구분할 수 없음

함수 중복의 편리함

- 동일한 이름을 사용하면 함수 이름을 구분하여 기억할 필요 없고, 함수 호출을 잘못하는 실수를 줄일 수 있음

```
void msg1() {  
    cout << "Hello";  
}  
void msg2(string name) {  
    cout << "Hello, " << name;  
}  
void msg3(int id, string name) {  
    cout << "Hello, " << id << " " << name;  
}
```

(a) 함수 중복하지 않는 경우



```
void msg() {  
    cout << "Hello";  
}  
void msg(string name) {  
    cout << "Hello, " << name;  
}  
void msg(int id, string name) {  
    cout << "Hello, " << id << " " << name;  
}
```

(b) 함수 중복한 경우

예제 - big() 함수 중복 연습

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

int big(int a, int b) { // a와 b 중 큰 수 리턴
    if(a>b) return a;
    else return b;
}

int big(int a[], int size) { // 배열 a[]에서 가장 큰 수 리턴
    int res = a[0];
    for(int i=1; i<size; i++)
        if(res < a[i]) res = a[i];
    return res;
}

int main() {
    int array[5] = {1, 9, -2, 8, 6};
    cout << big(2,3) << endl;
    cout << big(array, 5) << endl;
}
```

예제 - sum() 함수 중복 연습

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

int sum(int a, int b) { // a에서 b까지 합하기
    int s = 0;
    for(int i=a; i<=b; i++)
        s += i;
    return s;
}

int sum(int a) { // 0에서 a까지 합하기
    int s = 0;
    for(int i=0; i<=a; i++)
        s += i;
    return s;
}

int main() {
    cout << sum(3, 5) << endl;
    cout << sum(3) << endl;
    cout << sum(100) << endl;
}
```

12
6
5050

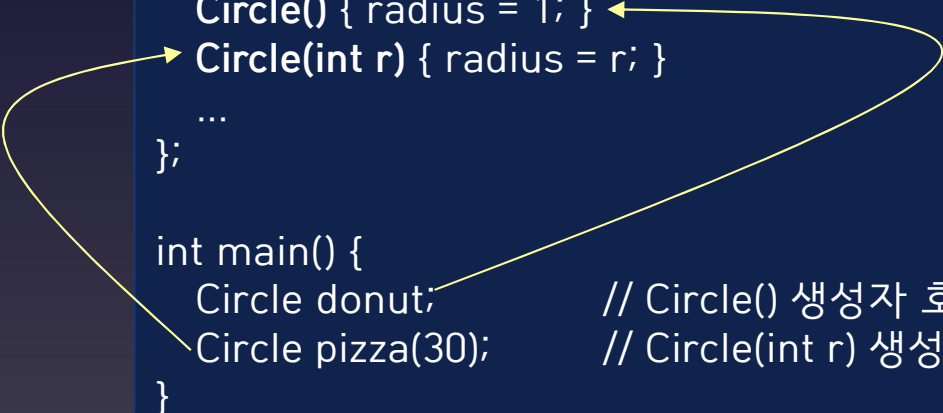
생성자 함수 중복

■ 생성자 함수 중복 가능

- 생성자 함수 중복 목적

- 객체 생성시, 매개 변수를 통해 다양한 형태의 초깃값 전달

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    ...  
};  
  
int main() {  
    Circle donut;           // Circle() 생성자 호출  
    Circle pizza(30);       // Circle(int r) 생성자 호출  
}
```



string 클래스의 생성자 중복 사례

```
class string {  
    .....  
public:  
    string();           // 빈 문자열을 가진 스트링 객체 생성  
    string(string& str); // str을 복사한 새로운 스트링 객체 생성  
    string(char* s);    // '\0'로 끝나는 C-스트링 s를 스트링 객체로 생성  
    .....  
};
```

```
string str;  
string address("서울시 성북구 삼선동 389");  
string copy(address); // address의 문자열을 복사한 별도의 copy 객체 생성
```

디폴트 매개 변수

- 디폴트 매개 변수(default parameter)

- 매개 변수에 값이 넘어오지 않는 경우, 디폴트 값을 받도록 선언된 매개 변수
 - '매개 변수 = 디폴트값' 형태로 선언

- 디폴트 매개 변수 선언 사례

```
void star(int a=5); // a의 디폴트 값은 5
```

- 디폴트 매개 변수를 가진 함수 호출

```
star(); // 매개 변수 a에 디폴트 값 5가 전달됨. star(5);와 동일  
star(10); // 매개 변수 a에 10을 넘겨줌
```

디폴트 매개 변수 사례

■ 사례 1

```
void msg(int id, string text="Hello"); // text의 디폴트 값은 "Hello"
```

```
msg(10); // msg(10, "Hello"); 호출과 동일. id에 10, text에 "Hello" 전달
```

```
msg(20, "Good Morning"); // id에 20, text에 "Good Morning" 전달
```

```
msg();      // 컴파일 오류. 첫 번째 매개 변수 id에 반드시 값을 전달하여야 함
```

```
msg("Hello"); // 컴파일 오류. 첫 번째 매개 변수 id에 값이 전달되지 않았음
```

호출 오류

디폴트 매개 변수에 관한 제약 조건

- 디폴트 매개 변수는 보통 매개 변수 앞에 선언될 수 없음
 - 디폴트 매개 변수는 끝 쪽에 몰려 선언되어야 함

컴파일 오류

```
void calc(int a, int b=5, int c, int d=0); // 컴파일 오류  
void sum(int a=0, int b, int c);           // 컴파일 오류
```

```
void calc(int a, int b=5, int c=0, int d=0); // 컴파일 성공
```

매개변수에 값을 정하는 규칙

■ 사례 2

```
void square(int width= 1, int height= 1);
```

square();	————	square(<u> </u> <u> </u>);	————	square(1, 1);
square(5);	————	square(5, <u> </u>);	————	square(5, 1);
square(3, 8);	————→	square(3, 8);	————→	square(3, 8);

컴파일러에 의해
변환되는 과정

디폴트 매개 변수 사례

■ 사례 3

```
void g(int a, int b= 0, int c= 0, int d= 0);
```

디폴트 매개 변수를
가진 함수

`g(10);`

`g(10, _ , _ , _);`

`g(10, 0, 0, 0);`

`g(10, 5);`

`g(10, 5 , _ , _);`

`g(10, 5, 0, 0);`

`g(10, 5, 20);`

`g(10, 5, 20, _);`

`g(10, 5, 20, 0);`

`g(10, 5, 20, 30);`

`g(10, 5, 20, 30);`

`g(10, 5, 20, 30);`

컴파일러에 의해
변환되는 과정

함수 중복 간소화

- 디폴트 매개 변수의 장점 - 함수 중복 간소화

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    .....  
};
```

```
class Circle {  
    .....  
public:  
    Circle(int r=1) { radius = r; }  
    .....  
};
```

2 개의 생성자 함수를
디폴트 매개 변수를 가진 하나
의 함수로 간소화

- 중복 함수들과 디폴트 매개 변수를 가진 함수를 함께 사용 불가

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    Circle(int r=1) { radius = r; }  
    .....  
};
```


중복된 함수와
동시 사용 불가

예제

- 다음 두 개의 중복 함수를 디폴트 매개 변수를 가진 하나의 함수로 작성하라

```
void fillLine() { // 25 개의 '*' 문자를 한 라인에 출력
    for(int i=0; i<25; i++) cout << '*';
    cout << endl;
}
```

```
void fillLine(int n, char c) { // n개의 c 문자를 한 라인에 출력
    for(int i=0; i<n; i++) cout << c;
    cout << endl;
}
```



```
void fillLine(int n=25, char c='*') { // n개의 c 문자를 한 라인에 출력
    for(int i=0; i<n; i++) cout << c;
    cout << endl;
}
```

함수 중복의 모호성

- 함수 중복이 모호하여 컴파일러가 어떤 함수를 호출하는지 판단하지 못하는 경우
 - 형 변환으로 인한 모호성
 - 참조 매개 변수로 인한 모호성
 - 디폴트 매개 변수로 인한 모호성

형 변환으로 인한 함수중복의 모호성

■ 매개 변수의 형 변환으로 인한 중복 함수 호출의 모호성

```
double square(double a) {  
    return a*a;  
}  
int main() {  
    cout << square(3);  
}
```

정상 컴파일

int 타입 3이
double 로 자동
형 변환

3.0은 double 타입이므로
모호하지 않음

```
float square(float a) {  
    return a*a;  
}  
double square(double a) {  
    return a*a;  
}  
int main() {  
    cout << square(3.0);  
    cout << square(3);  
}
```

모호한 호출, 컴파일 오류

예제

두 함수는 근본적으로
중복 시킬 수 없다.

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

int add(int a, int b) {
    return a + b;
}

int add(int a, int &b) {
    b = b + a;
    return b;
}

int main(){
    int s=10, t=20;
    cout << add(s, t); // 컴파일 오류
}
```

call by value인지
call by reference인지 모호

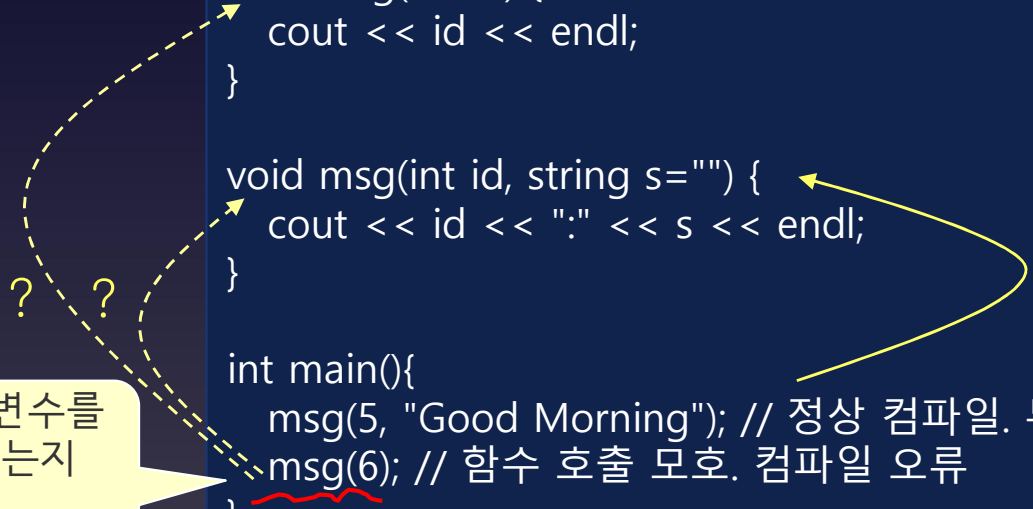
예제

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

void msg(int id) {
    cout << id << endl;
}

void msg(int id, string s="") {
    cout << id << ":" << s << endl;
}

int main(){
    msg(5, "Good Morning"); // 정상 컴파일. 두 번째 msg() 호출
    msg(6); // 함수 호출 모호. 컴파일 오류
}
```



디폴트 매개 변수를
이용하고 있는지
모호함

실습 2

- 다음에서 디폴트 매개변수를 가진 하나의 `add()` 함수를 작성하시오

```
int main()
{
    int a[] = {1,2,3,4,5};
    int b[] = {6,7,8,9,10};
    int c = add(a, 5); // 배열 a의 정수를 모두 더한 값 리턴
    int d = add(a, 5, b); // 배열 a와 b의 정수를 모두 더한 값 리턴
    cout << c << endl; // 15 출력
    cout << d << endl; // 55 출력
}
```


실습 3

- 주어진 클래스에서 디폴트 매개변수를 가진 하나의 생성자로 수정하시오

```
class MyVector
{
    int *mem;
    int size;
public:
    MyVector();
    MyVector(int n, int val);
    ~MyVector() { delete [] mem; }
};
```

```
MyVector::MyVector() {
    mem = new int [100];
    size = 100;
    for(int i=0; i<size; i++) mem[i] = 0;
}
```

```
MyVector::MyVector(int n, int val) {
    mem = new int [n];
    size = n;
    for(int i=0; i<size; i++) mem[i] = val;
}
```

```
int main() {
    MyVector a; // a(100, 0);과 동일
    MyVector b(10, 3);

    a.show(); // 100개의 0이 출력
    b.show(); // 10개의 3이 출력
}
```

