dongguk UNIVERSITY

심화프로그래밍

실습강의 9주차

실습강의 소개

- **실습 진행 방법**
  - 간단한 이론 복습 및 해당주차 실습과제 설명
  - 실습 후 보고서와 소스코드를 압축하여 **수요일 자정(23:59)까지**

    꼭!! 이클래스 제출(**이메일 제출 불가, 반드시 이클래스를 통해 제출**)
  - 실습과제 제출기한 엄수(제출기한 이후로는 0점 처리)
- **Q & A**
  - 이클래스 및 실습조교 이메일을 통해 질의응답
  - **이메일 제목 : [심화프로그래밍_홍길동] *본인 과목명과 성명 꼭 작성!!**
  - **실습조교 메일 주소 : 0hae@dgu.ac.kr**

실습 보고서 작성 방법 [1/2]

- **실습 보고서**
  - **문제 분석: 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술**
  - **프로그램 설계 및 알고리즘**
    - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
    - e.g.) 문제 해결 과정 및 핵심 알고리즘 기술
  - **소스코드 및 주석**
    - 소스코드와 그에 해당하는 주석 첨부
    - 각각의 함수가 수행하는 작업, 매개변수, 반환 값 등을 명시
    - 소스코드 전체 첨부**(소스코드 화면 캡처X, 소스코드는 복사/붙여넣기로 첨부)**
  - **결과 및 결과 분석**
    - 결과 화면을 캡쳐 하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
  - **소감**
    - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

실습 보고서 작성 방법 [2/2]

- **제출 방법**
  - 보고서, 소스코드를 1개의 파일로 압축하여 e-class "과제" 메뉴를 통해 제출
    - **"이름학번실습주차.zip" 형태로 제출(e.g. :김동국19919876실습9.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
- **유의 사항**
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한내 제출
    - 기한 넘기면 **0점** 처리
    - 이클래스가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일전에 제출
    - **과제 제출 당일 이클래스 오류로 인한 미제출은 불인정**
  - 소스코드, 보고서를 자신이 작성하지 않은 경우 **실습 전체 점수 0점 처리**
  - **Visual Studio 2019 또는 Sharstra 웹 IDE 기반 학습 프로그램 사용하여 실습 진행**

프렌드 함수(friend function)

- **프렌드**
  - 접근지정자(private, protected)를 접근해야 하는 경우가 생길 때를 대비해 제공하는 함수임

```
1. friend 클래스명 or 함수명(필요시 매개 변수 목록);
```

- **주요 특징**
  - 프렌드를 클래스나 함수에 사용할때 정의에서는 사용하지 않고 **원형에서만 사용하여 접근할수 있는 권한는 주는 역할**을 함
  - 프렌드 키워드를 적용한 클래스나 함수에게 데이터를 공개하도록 해당 클래스에 알려주는 것이므로 private, public, protected 어느곳에서 프렌드를 적용하여 선언해도 동일한 기능을 수행함
- **프렌드 함수를 선언할 수 있는 3가지 경우**
  - 클래스 외부에 작성된 함수를 프렌드로 선언
  - 다른 클래스의 멤버 함수를 프렌드로 선언
  - 다른 클래스의 모든 멤버 함수를 프렌드로 선언

static 멤버 [1/5]

- **정적 멤버 변수(static member variable)**
  - 클래스에는 속하지만 객체 별로 할당되지 않고 **클래스의 모든 객체가 공유하는 멤버**를 의미한다
  - 정적 멤버 변수는 클래스의 멤버 함수나 나중에 학습하는 프렌드를 통해 접근 가능하다
    - 외부에서도 접근할 수 있게 하고 싶으면 정적 멤버 변수를 public 영역에 선언하면 된다

```
1.  class Person {
2.  private:
3.      string name_;
4.      int age_;
5.  public:
6.      static int person_count_;          // 정적 멤버 변수의 선언
7.      Person(string& name, int age);     // 생성자
8.      ~Person() { person_count_--; }     // 소멸자
9.      void ShowPersonInfo();
10. };
11. ...
12. int Person::person_count_ = 0;         // 정적 멤버 변수의 정의 및 초기화
```

```
1 번째 사람이 생성되었습니다.
이 사람의 이름은 길동이고, 나이는 29살입니다.
2 번째 사람이 생성되었습니다.
이 사람의 이름은 순신이고, 나이는 35살입니다.
```

static 멤버 [2/5]

- **정적 멤버 함수(static member function)**
  - 클래스의 멤버 함수도 정적(static) 키워드를 사용하여 선언할 수 있다
  - 해당 클래스의 객체를 생성하지 않고도(this 포인터 없이), **클래스 이름만으로 호출 가능**하다

```
객체이름.멤버함수이름();    // 일반 멤버 함수의 호출
클래스이름.멤버함수이름(); // 정적 멤버 함수의 호출
```

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <random>

using namespace std;

// 오름차순 정렬 함수
void sortAscending(vector<int>& nums) {
    sort(nums.begin(), nums.end());
}

// 내림차순 정렬 함수
void sortDescending(vector<int>& nums) {
    sort(nums.begin(), nums.end(), greater<int>());
}
```

static 멤버 [3/5]

```
(앞 코드 계속)

int main() {
    // 난수 생성 엔진 초기화
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> dis(1, 100);

    // 난수 생성 및 파일 쓰기
    ofstream outfile("input.txt");
    if (!outfile.is_open()) {
        cout << "파일 열기 실패" << endl;
        return 1;
    }

    const int num_numbers = 500;
    for (int i = 0; i < num_numbers; i++) {
        outfile << dis(gen) << endl;
    }

    outfile.close();

    // 입력 파일 열기
    ifstream infile("input.txt");
    if (!infile.is_open()) {
        cout << "파일 열기 실패" << endl;
        return 1;
    }
```

static 멤버 [4/5]

```
(앞 코드 계속)


// 파일 내용 읽어들이기
vector<int> numbers;
int number;
while (infile >> number) {
    numbers.push_back(number);
}

// 정렬 방식 선택하기
int option;
cout << "1. 오름차순 정렬, 2. 내림차순 정렬: ";
cin >> option;

// 선택한 정렬 방식에 따라 정렬하기
if (option == 1) {
    sortAscending(numbers);
}
else if (option == 2) {
    sortDescending(numbers);
}
else {
    cout << "잘못된 옵션" << endl;
    return 1;
}
```

static 멤버 [5/5]

```
    (앞 코드 계속)

    // 정렬된 결과 출력하기
    for (int num : numbers) {
        cout << num << " ";
    }

    // 파일 닫기
    infile.close();

    return 0;
}
```

```
1 번째 사람이 생성되었습니다.
2 번째 사람이 생성되었습니다.
현재까지 생성된 총 인원 수는 2명입니다.
```

static 멤버 vs non-static 멤버

- **static 멤버**
  - 공간 특성: 멤버는 클래스 당 **하나** 생성
    - 멤버는 객체 내부가 아닌 별도의 공간에 생성
    - 클래스 멤버라고 부름
  - 시간적 특성: **프로그램**과 생명을 같이 함.
    - 프로그램 시작 시 멤버 생성, 객체가 생기기 전에 이미 존재
    - 객체가 사라져도 여전히 존재, 프로그램이 종료될 때 함께 소멸
  - **공유의 특성: 동일한 클래스의 모든 객체들에 의해 공유됨**
- **not-static 멤버**
  - 공간 특성: 멤버는 객체마다 **별도** 생성
    - 인스턴스 멤버라고 부름 (<=> 클래스 멤버)
  - 시간적 특성: **객체**와 생명을 같이 함
    - 객체 생성 시에 멤버 생성, 객체 소멸 시 함께 소멸, 객체 생성 후 객체 사용 가능
  - **공유의 특성: 객체끼리 공유되지 않음**
    - 멤버는 객체 별로 따로 공간 유지

연산자 중복(operator overloading) [1/2]

- **연산자 중복**
  - 피연산자에 따라 서로 다른 연산을 하도록 **동일한 연산자를 중복해서 작성하는 것**
  - C++에 원래 있는 연산자만 중복 가능, **정수/실수가 아닌 객체나 값**
  - 연산자 중복으로 연산자의 우선 순위를 바꿀 수 없음
  - 모든 연산자(. Or ::)가 중복 가능하지 않음
  - 피연산자의 개수를 바꿀 수 없음
- **연산자 함수 작성 방법**
  - 클래스의 멤버 함수로 구현 가능함
  - 외부 함수로 구현하고 클래스의 프렌드 함수로 선언 가능함

```
1.  리턴타입 Operator 연산자(parameter);
```

- **이항 연산자 중복(binary operator)**
  - 피연산자 2개 → 덧셈(+), 뺄셈(-) 등
- **단항 연산자 중복(unary operator)**
  - 피연산자 1개 → 전위(Prefix), 후위(Postfix) 연산자

연산자 중복(operator overloading) [2/2]

- **이항 연산자 중복(Binary operator)**
  - 피연산자 2개 → 덧셈(+), 뺄셈(-) 등

Binary Operator	Description	Binary Operator	Description	Binary Operator	Description	Binary Operator	Description
+	addition	==	equal to	>=	greater than or equal to	^	bit-wise XOR
-	subtraction	!=	not equal to	&&	logical and	<<	bit-wise left-shift
*	multiplication	<	less than	||	logical or	>>	bit-wise right-shift
/	division	>	greater than	&	bit-wise AND	=	assignment
%	remainder	<=	less than or equal to	|	bit-wise OR	...	

- **단항 연산자 중복(Unary operator)**
  - 피연산자 1개 → 전위(Prefix), 후위(Postfix) 연산자

Unary Operator	Description	Unary Operator	Description	Unary Operator	Description	Unary Operator	Description
+	make positive	!	NOT	++	auto-increment	*	indirection
-	negate	~	Bit-wise not	--	auto-decrement	&	address of

연습문제(7장)

이론 문제

1. 프렌드에 대한 설명 중 틀린 것은?
   ① 한 클래스의 전체 멤버 함수를 프렌드로 선언할 수 없다.
   ② 프렌드 함수는 클래스의 멤버 함수가 아니다.
   ③ 프렌드 함수는 클래스의 private 멤버에 대한 접근 권한을 가진다.
   ④ 프렌드 함수는 friend 키워드로 클래스 내에 선언된다.

2. 프렌드 함수가 필요한 경우가 아닌 것은?
   ① 멤버는 아니지만 클래스의 private 멤버에 접근해야만 하는 함수를 작성하는 경우
   ② 두 개 이상의 클래스에 대해 private 멤버를 동시에 접근하는 함수를 작성하는 경우
   ③ 연산자 중복 시에
   ④ 함수 중복 시에

연습문제(7장)

3. 다음 클래스 Sample에서 클래스 SampleManager의 모든 멤버 함수를 프렌드로 초대하도록 선언하라.

```
class Sample {
...
};
```

4. 클래스 SampleManager에는 다음의 멤버 함수가 있다.

```
bool compare(Sample &a, Sample &b);
```

이 멤버 함수를 다음 클래스 Sample에 프렌드로 초대하도록 선언하라.

```
class Sample {
...
};
```

연습문제(7장)

5. 다음 프로그램은 컴파일 오류가 발생한다. 오류의 원인은 무엇인가? 오류를 바람직하게 수정하라.

```
class Student {
   int id;
public:
   Student(int id) { this->id = id;}
};
bool isValid(Student s) {
   if(s.id > 0) return true;
   else return false;
}
```

연습문제(7장)

6. 다음 프로그램은 컴파일 오류가 발생한다. 오류의 원인은 무엇인가? 오류를 바람직하게 수정하라.

```
class Student {.
   int id;
public:
   Student(int id) { this->id = id;}
};
class Professor {
   string name;
public:
   Professor(string name) { this->name = name;}
};
void show(Student s, Professor p) {
   cout << s.id << p.name;
}
```

힌트

문제를 푸는 와중에 전방 참조 문제를 해결하기 위해, 코드의 맨 앞에 다음 두 줄의 전방 선언문을 삽입하는 것을 잊지 마라.

```
class Student;
class Professor;
```

연습문제(7장)

7. 다음 프로그램은 컴파일 오류를 가지고 있다. 오류를 지적하고 바람직하게 수정하라.

```
class Food {
   int price;
   string name;
public:
   Food(string name, int price);
   void buy();
};
class Person {
   int id;
public:
   void shopping(Food food) {
      if(food.price < 1000)
         food.buy();
   }
   int get() { return id; }
};
```

연습문제(7장)

8. 프렌드 선언의 위치에 대한 설명 중 옳은 것은?
   ① 클래스 내의 private 영역에 선언되어야 한다.
   ② 클래스 내의 protected 영역에 선언되어야 한다.
   ③ 클래스 내의 public 영역에 선언되어야 한다.
   ④ 클래스 내의 아무 영역에 선언되어도 상관없다.

9. 다음의 friend 선언과 main() 함수의 isZero()의 호출이 옳은지 설명하라.

```
class Sample {
public:
   int x;
   Sample(int x) {this->x = x;}
   friend bool isZero(Sample &a) { // 이곳에 주목
      if(a.x == 0) return true;
      else return false;
   }
};
int main() {
   Sample a(5), b(6);
   bool ret = a.isZero(b); // 이곳에 주목
}
```

연습문제(7장)

10. 다음 코드에서 프렌드 선언이 필요한지 설명하라.

```
class Sample {
public:
   int x;
   Sample(int x) {this->x = x;}
   friend bool isZero(Sample &a); // 이곳에 주목
};
bool isZero(Sample& a) {
   if(a.x == 0) return true;
   else return false;
}
```

11. 다음은 어떤 다형성을 보여주고 있는가?

```
int a = 4 << 2;
cout << 'a';
```

12. 다음에 보이는 다형성에 일치하는 다른 사례를 하나만 들어라.

2 + 3 = 5

빨강 + 파랑 = 보라

남자 + 여자 = 결혼

연습문제(7장)

13. 연산자 중복의 특징이 아닌 것은?

① 모든 연산자를 중복할 수 있는 것은 아니다.
② 연산자 중복을 통해 연산자의 근본 우선순위를 바꿀 수 없다.
③ 연산자 중복은 반드시 클래스와 관계를 가진다.
④ 연산자 중복은 어떤 기호를 사용하든지 가능하다.

14. 다음에서 멤버 함수로 + 연산자 함수를 작성할 수 없는 경우는?

```
Power a, b; // Power 클래스에 대해
```

① a = a + b　② b = a + 3　③ b = a += 3　④ b = 3 + a

15. 다음에서 a, b는 Power 클래스의 객체이다. 연산자 함수를 Power 클래스의 멤버 함수로 작성한다고 할 때, 왼쪽의 연산과 오른쪽의 연산자 함수의 선언이 잘못된 것은?

① a + b	Power operator + (Power b);
② a == b	bool operator == (Power &b);
③ a += b	void operator += (Power b);
④ !a	bool operator !();

연습문제(7장)

16. 다음에서 a, b는 Power 클래스의 객체이다. 연산자 함수를 Power 클래스의 프렌드 함수로 작성한다고 할 때, 왼쪽의 연산과 오른쪽의 연산자 함수의 선언이 잘못된 것은?

① a + b — Power operator + (Power& a, Power &b);
② a == b — bool operator == (Power a, Power b);
③ a++ — Power operator ++ (Power a, int b);
④ a = b — Power& operator = (Power &a, Power b);

17. 다음과 같은 Circle 클래스가 있다.

```
class Circle {
   int radius;
public:
   Circle(int radius=0) { this->radius = radius; }
   int getRadius() { return radius; }
};
```

다음과 같이 Circle 객체 a, b에 대한 치환 연산(=)이 제대로 이루어지도록 할 때 치환연산자를 작성할 필요가 있는지 설명하라.

```
Circle a(20), b(30);
a = b;
```

실습과제(7장)

* 실습 과제는 짝수번만 제출하시오.

실습과제(7장)

*문제 1~4까지 사용될 Book 클래스는 다음과 같다.

```
class Book {
    string title;
    int price, pages;
public:
    Book(string title="", int price=0, int pages=0) {
        this->title = title; this->price = price; this->pages = pages;
    }
    void show() {
    cout << title << ' ' << price << "원 " <<pages << " 페이지 "  << endl;
    }
    string getTitle() { return title; }
};
```

실습과제(7장)

1. Book 객체에 대해 다음 연산을 하고자 한다.

```
Book a("청춘", 20000, 300), b("미래", 30000, 500);
a += 500; // 책 a의 가격 500원 증가
b -= 500; // 책 b의 가격 500원 감소
a.show();
b.show();
```

```
청춘 20500원 300 페이지
미래 29500원 500 페이지
```

1) +=, -= 연산자 함수를 Book 클래스의 멤버 함수로 구현하라.
2) +=, -= 연산자 함수를 Book 클래스의 외부 함수로 구현하라.

실습과제(7장)

2. Book 객체를 활용하는 사례이다.

```
Book a("명품 C++", 30000, 500), b("고품 C++" , 30000, 500);
if(a == 30000) cout << "정가 30000원 " << endl; // price 비교
if(a == " 명품 C++") cout << "명품 C++ 입니다." << endl; // 책 title 비교
if(a == b) cout << "두 책이 같은 책입니다." << endl; // title, price, pages 모두 비교
```

```
정가 30000원
명품 C++ 입니다.
```

1) 세 개의 == 연산자 함수를 가진 Book 클래스를 작성하라.
2) 세 개의 == 연산자를 프렌드 함수로 작성하라.

3. 다음 연산을 통해 공짜 책인지를 판별하도록 ! 연산자를 작성하라.

```
Book book("벼룩시장", 0, 50); // 가격은 0
if(!book) cout << "공짜다" << endl;
```

```
공짜다
```

실습과제(7장)

4. 다음 연산을 통해 책의 제목을 사전 순으로 비교하고자 한다.< 연산자를 작성하라.

```
Book a("청춘"", 20000, 300);
string b;
cout << "책 이름을 입력하세요>>";
getline(cin, b); // 키보드로부터 문자열로 책 이름을 입력 받음
if(b < a)
    cout << a.getTitle() << "이 " << b << "보다 뒤에 있구나!" << endl;
```

```
책 이름을 입력하세요>>바람과 함께 사라지다
청춘이 바람과 함께 사라지다보다 뒤에 있구나!
```

실습과제(7장)

5. 다음 main()에서 Color 클래스는 3요소(빨강, 초록, 파랑)로 하나의 색을 나타내는 클래스이다.(4장 실습 문제 1번 참고). + 연산자로 색을 더하고, == 연산자로 색을 비교하고자 한다. 실행 결과를 참고하여 Color 클래스와 연산자, 그리고 프로그램을 완성하라.

```
int main () {
    Color red (255, 0, 0), blue (0, 0, 255), c;
    c = red + blue;
    c.show(); // 색 값 출력

    Color fuchsia (255, 0, 255);
    if (c == fuchsia)
        cout << "보라색 맞음";
    else
        cout << "보라색 아님";
}
```

```
255 0  255
보라색 맞음
```

   1) +와 == 연산자를 Color 클래스의 멤버 함수로 구현하라.
   2) +와 == 연산자를 Color 클래스의 프렌드 함수로 구현하라.

실습과제(7장)

6. 2차원 행렬을 추상화한 Matrix 클래스를 작성하고, show() 멤버 함수와 다음 연산이 가능하도록 연산자를 모두 구현하라.

```
Matrix a(1,2,3,4), b(2,3,4,5), c;
c = a + b;
a += b;
a. show(); b.show(); c.show();
if(a == c)
    cout << "a and c are the same" << endl;
```

```
Matrix = { 3 5 7 9 }
Matrix = { 2 3 4 5 }
Matrix = { 3 5 7 9 }
a and c are the same
```

1) 연산자 함수를 Matrix의 멤버 함수로 구현하라.
2) 연산자 함수를 Matrix의 프렌드 함수로 구현하라.

실습과제(7장)

7. 2차원 행렬을 추상화한 Matrix 클래스를 활용하는 다음 코드가 있다.

```
Matrix a (4, 3, 2, 1), b;
int x[4], y[4] = {1,2,3,4}; // 2차원 행렬의 4 개의 원소 값
a >> x; // a의 각 원소를 배열 x에 복사. x[]는 (4,3,2,1}
b << y; // 배열 y의 원소 값을 b의 각 원소에 설정

for(int i=0; i<4; i++) cout << x[i] << ' '; // x[] 출력
cout << endl;
b.show();
```

```
4  3  2  1
Matrix = { 1 2 3 4 }
```

1) <<, >> 연산자 함수를 Matrix의 멤버 함수로 구현하라.
2) <<, >> 연산자 함수를 Matrix의 프렌드 함수로 구현하라.

실습과제(7장)

8. 원을 추상화한 Circle 클래스는 간단히 아래와 같다.

```
class Circle {
    int radius;
public:
    Circle(int radius=0) { this->radius = radius; }
    void show() { cout << "radius = " << radius << " 인 원 "  << endl; }
};
```

다음 연산이 가능하도록 연산자를 프렌드 함수로 작성하라.

```
Circle a (5), b(4);
++a; // 반지름을 1 증가 시킨다.
b = a++; // 반지름을 1 증가 시킨다.
a.show();
b.show();
```

```
radius = 7 인원
radius = 6 인원
```

실습과제(7장)

9. 문제 8번의 Circle 객체에 대해 다음 연산이 가능하도록 연산자를 구현하라.

```
Circle a(5), b(4);
B= 1 + a; // b의 반지름을 a의 반지름에 1을 더한 것으로 변경
a.show();
b.show();
```

```
radius = 5 인원
radius = 6 인원
```

실습과제(7장)

10. 통계를 내는 Statistics 클래스를 만들려고 한다. 데이터는 Statistics 클래스 내부에 int 배열을 동적으로 할당받아 유지한다. 다음과 같은 연산이 잘 이루어지도록 Statistics 클래스와 !, >>, <<, ~ 연산자 함수를 작성하라.

```
Statistics stat;
if(!stat) cout << "현재 통계 데이타가 없습니다." << endl;

int x[5];
cout << "5 개의 정수를 입력하라>>";
for (int i=0; i<5; i++) cin >> x[i]; // x[i]에 정수 입력

for(int i=; i<5; i++) stat << x[i]; // x[i] 값을 통계 객체에 삽입한다.
stat << 100 << 200; // 100, 200을 통계 객체에 삽입한다.
~stat; // 통계 데이타를 모두 출력한다.

int avg;
stat >> avg; // 통계 객체로부터 평균을 받는다.
cout << "avg=" << avg << endl; // 평균을 출력한다
```

```
현재 통계 데이타가 없습니다.
5 개의 정수를 입력하라>> 1 2 3 4 5
1 2 3 4 5 100 200
avg=45
```

실습과제(7장)

11. 스택 클래스 Stack을 만들고 푸시(push)용으로 << 연산자를, 팝(pop)을 위해 >> 연산자를, 비어 있는 스택인지를 알기 위해 ! 연산자를 작성하라. 다음 코드를 main()으로 작성하라.

```
Stack stack;
stack << 3 << 5 << 10; // 3, 5, 10을 순서대로 푸시
while (true) {
    if(!stack) break; // 스택 empty
    int x;
    stack >> x; // 스택의 탑에 있는 정수 팝
    cout << x << ' ';
}
Cout << endl;
```

```
10  5  3
```

실습과제(7장)

12. 정수 배열을 항상 증가 순으로 유지하는 SortedArray 클래스를 작성하려고 한다. 아래의 main() 함수가 작동할 만큼만 SortedArray 클래스를 작성하고 +와 = 연산자도 작성하라.

```
class SortedArray {
    int size; // 현재 배열의 크기
    int *p; // 정수 배열에 대한 포인터
    void sort(); // 정수 배열을 오름차순으로 정렬
public:
    SortedArray(); // p는 NULL로 size는 0으로 초기화
    SortedArray(SortedArray& src); // 복사 생성자
    SortedArray(int p[], int size); // 생성자. 정수 배열과 크기를 전달받음
    ~SortedArray(); // 소멸자
    SortedArray operator + (SortedArray& op2); // 현재 배열에 op2 배열 추가
    SortedArray& operator = (const SortedArray& op2); // 현재 배열에 op2 배열 복사
    void show(); // 배열의 원소 출력
};
```

실습과제(7장)

12. (앞 슬라이드 계속)

```
int main () {
    int n[] = { 2, 20, 6 };
    int m[] = { 10, 7, 8, 30 };
    SortedArray a(n, 3), b(m, 4), c;

    c=a+b; // +, = 연산자 작성 필요
    // +연산자가 SortedArray 객체를 리턴하므로 복사 생성자 필요

    a.show();
    b.show();
    c.show();
}
```

```
배열 출력 : 2 6 20
배열 출력 : 7 8 10 30
배열 출력 : 2 6 7 8 10 20 30
```

힌트: + 연산자는 SortedArray 객체를 리턴하므로 복사 생성자가 반드시 필요하다. a=b; 연산에서 = 연산자는 객체 a의 배열 메모리를 모두 delete 시키고 객체 b의 크기만큼 다시 할당받은 후 객체 b의 배열 내용을 복사하도록 작성되어야 한다.