C++ Programming

02 - 1) C++ 프로그래밍 기본

2.1) C++ 프로그램의 기본 요소와 화면 출력

- 예제 2-1 : 기본적인 C++ 프로그램

```
#include <iostream>     // cout 과 << 연산자 포함

int main(){
    std::cout << "Hello\n";     // 화면에 Hello 를 출력하고 다음 줄로 넘어감
    std::cout << "첫 번째 맛보기 입니다.";
    return 0;
}
```

실행결과

```
Hello
첫 번째 맛보기 입니다.
```

- **iostream 헤더 파일** : 표준 입출력을 위한 클래스와 객체, 변수 등이 선언됨
  - cout, cin, <<, >> 등 연산자 선언

- **화면 출력**
  - cout 객체 : 출력 스트림 **객체**
  - << 연산자 : 스트림 삽입 **연산자**

CHECK TIME

1. 표준 C++에서 main() 함수의 리턴 타입은 무엇인가?

   int

2. 예제 2-1의 소스에서 #include iostream 을 제거하면 소스의 어떤 부분에서 컴파일 오류가 발생하는가? 그 이유는 무엇인가?

   std::cout 이 쓰여진 곳에서 오류가 발생한다. 왜냐하면 cout 객체와 << 연산자를 사용하려면 iostream 헤더 파일을 필요로 하기 때문이다.

3. cout은 무엇인가?

   객체

4. <<란 무엇인가?

   연산자

5. 자신의 이름을 출력하고, 다음 줄에 자신의 주소를 출력하는 한 줄의 C++ 코드를 작성하라.

```
#include <stdlib.h>
#include <iostream>

int main() {
    std::cout << "이상민\n";
    std::cout << "대전 서구 월평동\n";
    system("pause");
}
```

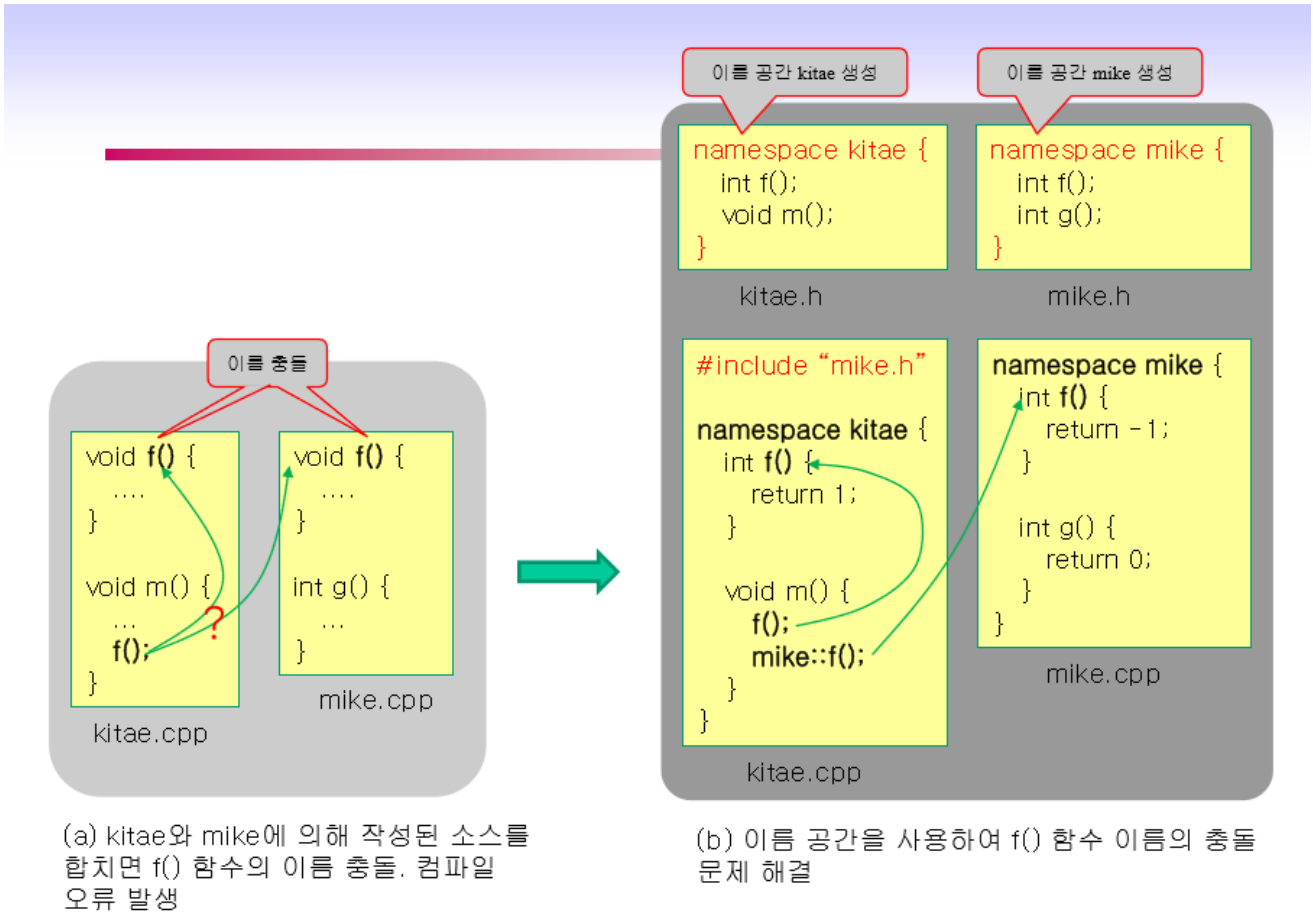
2.2) namespace 와 std::

namespace

- 이름 충돌 해결, 개발자가 자신만의 이름 공간을 생성할 수 있도록함
- 이름 공간 생성 및 사용

```
namespace kitae{    // kitae 라는 이름 공간 생성
    ... // 이 곳에 선언된 모든 이름은 kitae 이름 공간에 생성된 이름
}
```

> **이름 공간 사용** : 이름 공간 :: 이름



(a) kitae와 mike에 의해 작성된 소스를 합치면 f() 함수의 이름 충돌. 컴파일 오류 발생

(b) 이름 공간을 사용하여 f() 함수 이름의 충돌 문제 해결

std::

- ANSI C++ 표준에서 정의한 **namespace** 중 하나
- std 이름 공간에 선언된 이름을 접근하기 위해 사용
  - **ex)** std::cout, std::cin
- **std::생략**
  - **using 지시어 사용**

```
using std::cout;        // cout에 대해서만 std:: 생략
using namespace std;    // std 이름 공간에 선언된 모든 이름에 std:: 생략
```

CHECKTIME

1. C++ 에서 이름(identifier)에 속하지 않는 것은?

   파일명

2. 새로운 이름 공간을 선언할 때 사용하는 키워드는?

   namespace

3. C++ 표준 라이브러리에 선언된 모든 이름을 포함하는 C++ 표준 이름 공간은 무엇인가>

   std

4. std 이름 공간의 모든 이름에 std:: 를 생략하도록 지시하는 지시문을 쓰라.

```
using namespace std;
```

2.3) 키 입력 받기

예제 소스

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

int main() {
    cout << "너비를 입력하세요 >> ";

    int width;
    cin >> width;   // 키보드로부터 정수 값 너비를 읽어 width 변수에 저장

    cout << "높이를 입력하세요 >> ";

    int height;
    cin >> height;  // 키보드로 부터 정수 값 높이를 읽어 height 변수에 저장

    int area = width * height;          // 사각형의 면적 계산
    cout << "면적은 " << area << "\n"; // 면적을 출력하고 다음줄로
}
```

실행결과

```
너비를 입력하세요 >> 3
높이를 입력하세요 >> 5
면적은 15
```

cin과 >> 연산자를 이용한 키 입력

- **cin 객체** : C++ 입력 스트림 객체
- **>> 연산자** : 스트림 추출 연산자

Enter 키를 칠 때 변수에 값 전달

- **cin의 특징**
    - 입력 버퍼를 내장하고 있음
    - Enter 키를 입력받는다.

실행문 중간에 변수선언

- **C++ 의 변수선언** : C++ 에서 변수 선언은 아무 곳이나 가능

```
int width;
cin >> width;
int height;     // 실행문 중간에 변수 선언
cint >> height;
```

> **장점** : 번거로움 해소, 타이핑 오류 줄임
>
> **단점** : 변수를 찾기 힘들다.

CHECK TIME

1. 키보드 장치와 연결되어 C++ 응용프로그램에서 사용자가 입력한 키를 공급하는 객체는?

   cin

2. 다음 문에서 >> 연산자가 입력된 키 값을 정수형 변수 n에 저장하는 시점은?

   ```
   cin >> n;
   ```

   Enter 키가 입력될 때

3. 키보드로부터 int형의 radius 변수에 반지름 값을 읽어 들이고, 원의 면적을 계산하여 double 형의 area 변수에 저장한 후, 출력하는 프로그램을 작성하라.

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

int main() {
    int radius;

    cout << "반지름을 입력하세요 : ";
    cin >> radius;

    double area = radius * radius * 3.14;
    cout << "원의 면적은 " << area << " 입니다." << endl;
}
```

2.4) 키보드로 문자열 입력

C++의 문자열

- **C - 스트링 방식** : '\0' 로 끝나는 문자 배열 (C언어 문자열 표현 방식)

```
char name1[6] = {'G', 'r', 'a', 'c', 'e', '\0'};    // name1은 문자열 "Grace"
char name2[5] = {'G', 'r', 'a', 'c', 'e'};  // name2는 문자열이 아니고 단순 문자 배열
```

- **string 클래스 이용 방식** : 문자열을 객체로 다루는 방법
  - 헤더 파일에 선언됨

cin을 이용한 문자열 입력

- **문자열 입력**

```
char name[6];   // 5 개의 문자를 저장할 수 있는 char 배열
cin >> name;    // 키보드로부터 문자열을 읽어 name 배열에 저장한다.
```

- **ex) C-스트링을 이용하여 암호가 입력되면 프로그램을 종료하는 예**

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

int main(){
    char password[11];
    cout << "프로그램을 종료하려면 암호를 입력하세요." << endl;
    while(true){
        cout << "암호 >> ";
        cin >> password;    // 문자열을 입력받는다.
        if(strcmp(password, "C++") == 0){
            cout << "프로그램을 정상 종료합니다." << endl;
            break;
        }
        else
            cout << "암호가 틀립니다." << endl;
    }
}
```

실행결과

```
프로그램을 종료하려면 암호를 입력하세요.
암호>>Java
암호가 틀립니다.
암호>>C++
프로그램을 정상 종료합니다.
```

cin.getline() 을 이용한 문자열 입력

- **getline( )** 함수를 이용하면 **공백이 포함된 문자열**을 입력 받을 수 있다.
- **예제 2-6) cin.getline( )을 이용한 문자열 입력**

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

int main(){
    cout << "주소를 입력하세요>>";

    char address[100];
    cin.getline(address, 100, '\n');    // 키보드로부터 주소 읽기

    cout << "주소는 " << address << "입니다.\n";  // 주소 출력
}
```

실행결과

```
주소를 입력하세요>>대전 서구 월평동
주소는 대전 서구 월평동 입니다.
```

C++ 에서 문자열을 다루는 string 클래스

- **string 클래스**
  - 문자열 크기에 따른 제약없음
  - 다루기 쉽다
  - string 헤더 파일에 선언
- **ex) string 클래스를 이용한 문자열 입력 및 다루기**

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

int main(){
    string song("Falling in love with you");
    string elvis("Elvis Presley");
    string singer;  // 문자열들 선언

    cout << song + "를 부른 가수는";  // + 로 문자열 연결
    cout << "(힌트 : 첫글자는 " << elvis[0] << ")?";  // [] 연산자 사용

    getline(cin, singer);   // 문자열 입력
    if(singer == elvis)     // 문자열 비교
        cout << "맞았습니다.";
    else
        cout << "틀렸습니다." + elvis + "입니다." << endl;  // + 로 문자열 연결
}
```

실행결과

```
Falling in love with you를 부른 가수는(힌트:첫글자는 E)?Elvis Pride
틀렸습니다. Elvis Presley 입니다.
```

CHECK TIME

1. 다음 코드에 대해 잘못 설명한 것은?

```
char department[21];
cin >> department;
```

    1. 키보드로부터 문자열을 읽어 department[ ] 배열에 저장한다.
    2. 사용자는 영문자 20개로 구성된 문자열을 입력할 수 있다.
    3. **사용자는 한글 20 글자로 구성된 문자열을 입력할 수 있다. (X)** (한글은 1글자당 2바이트 이다.)
    4. 사용자는 반드시 공백 없이 문자열을 입력하여야 정상적으로 입력된다.
2. '.' 문자가 입력될 때까지 도시의 이름을 문자열로 입력받아 char city[21] 배열에 저장하는 cin.getline( ) 호출 코드를 보여라. 도시의 이름은 최대 20 글자이며, 영문자로 입력하는 것을 가정한다.

```
cin.getline(city, 20, '.');
```

비트 연산자

연산자	연산자의 의미	설명
&	비트 AND	비트가 모두 1이면 1, 아니면 0
|	비트 OR	비트가 하나만 1이면 1
^	비트 XOR	비트의 값이 같으면 0, 아니면 1
<<	왼쪽으로 이동	지정된 개수만큼 비트를 왼쪽으로 이동
>>	오른쪽으로 이동	지정된 개수만큼 모든 비트를 오른쪽으로 이동한다.
~	비트 NOT	0 -> 1 , 1 -> 0

연산자

- **Quiz**

```
x *= y + 1      // x = x * ( y + 1 )
x %= x + y      // x = x % ( x + y )
```

- **조건 연산자**

  max_value = (x > y) ? x : y ;

2장 연습문제

이론 문제

1. C++ 응용프로그램이 실행을 시작하는 함수의 원형은 무엇인가?

   int main( )

2. C++ 에서 main( ) 함수에 대한 설명 중 틀린 것은?

   1. C++ 표준에서 정한 main( ) 함수의 리턴 타입은 int 이다.
   2. void main( ) 으로 작성해도 대부분의 컴파일러에서는 처리된다.
   3. **main( ) 함수는 반드시 return 문을 가지고 있어야 한다. (X)**
   4. main( ) 함수는 반드시 정수 0을 리턴할 필요가 없다.

3. 다음 소스에서 생략해도 되는 라인은 어디인가?

```
#include <iostream>
int main(){
    std::cout << "I love C++\n";
    std::cout << "I love programming";
    return 0;   // 생략해도 된다.
}
```

   return 0

4. 다음 코드는 C 컴파일러로 컴파일하면 컴파일 오류가 발생하지만 C++ 컴파일러로 컴파일하면 정상적으로 컴파일 된다.

```
int a;
a = 4;
int square = a * a;
```

   (1) C 컴파일러로 컴파일할 때 어떤 컴파일 오류가 발생하는가?

   변수 square가 실행문 중간에 선언되어 있기 때문에 오류가 발생한다.

   (2) C++ 컴파일러로 컴파일할 때 정상적으로 컴파일되는 것은 C++ 언어의 어떤 특성 때문인가?

   실행문 중간에도 변수의 선언이 가능하다.

   (3) 이 특징이 가진 장단점은 무엇인가?

   이미 선언된 변수를 다른 변수로 잘못 타이핑하는 오류를 발견하기는 쉽지만 변수들이 함수 전체에 흩어져 있어 변수를 찾기가 어렵다는 단점이 있다.

5. 다음 프로그램의 실행 결과는 무엇인가?

```cpp
#include <iostream>
int main(){
    std::cout << "I love C++\n" << "I love programming";
    return 0;
}
```

실행결과

```
I love C++
I love programming
```

6. 다음 프로그램에 컴파일 오류가 발생하지 않도록 빈칸을 채워라.

(1)

```cpp
#include <iostream>
using std::cout;    // 밑줄 위치
int main(){
    int count = 0;
    std::cin >> count;
    cout << count + 1;
    return 0;
}
```

(2)

```cpp
#include <iostream>
using namespace std;    // 밑줄 위치
int main(){
    cout << "I love C++" << endl;
    cout << "I love programming";
    return 0;
}
```

7. 다음 C++ 프로그램 코드에서 틀린 부분을 수정하라.

```cpp
#include <iostream>;    // #include <iostream>
using namespace std     // using namespace std;
std::cin << name;       // cin >> name;
std:cout << 1 << 2 << 'a' << "hello" << '\n';   // std::cout ~
```

8. 다음 C++ 프로그램 코드에서 틀린 부분이 있으면 수정하라.

```
using std::cin;     // 틀린 부분 X
int year = 1;       // 틀린 부분 X
int n=1; cout >> n + 200;   // cout << n + 200;
int year = 2014; cout << 2014 + "년";    // cout << 2014 << "년";
```

9. 다음은 개발자가 작성한 myheader.h 파일의 소스이다.

```
#define MAX 100
#define MIN 0
```

다음 myprog.cpp 프로그램에서 빈칸에 적절한 라인을 삽입하라.

```
#include <iostream>
using namespace std;
#include "myheader.h"   // 빈칸 위치
int main(){
    cout << MAX << MIN;
    return 0;
}
```

10. C++ 문자열에 대한 다음 질문에 O, X 로 답하라.

(1) C-스트링이란 C 언어에서 문자열을 다루는 방식이다. **(O)**

(2) C++에서 C-스트링 방식의 문자열이 사용된다. **(O)**

(3) C++에서 문자열을 다루기 위해 string 클래스가 이용된다. **(O)**

(4) char name[ ] = "C++"; 이 컴파일 되면 name[ ] 배열의 크기가 3이 된다. **(X , 3이 아니라 4가 된다.)**

(5) char name[10]; cin >> name; 를 실행하면 공백 문자를 포함하여 키보드로 부터 최대 9개의 문자를 읽을 수 있다. **(O)**

11. C-스트링을 다루기 위해, strcmp( ), strlen( ) 등의 함수를 사용하고자 할 때 include 해야 하는 표준 헤더 파일은 무엇인가?

cstring 파일

12. 다음 프로그램이 있다.

```
#include <iostream>
int main(){
    char name[20];
    std::cout << "이름을 입력하세요?";
    std::cin >> name;
    std::cout << name << "님 환영합니다.";
    return 0;
}
```

(1) 프로그램을 실행하고 다음과 같이 키보드로 Kitae를 입력한 결과는 무엇인가?

```
이름을 입력하세요?Kitae
Kitae님 환영합니다.       // 빈칸 위치
```

(2) 프로그램을 실행하고 다음과 같이 키보드로 Kitae Hwang을 입력한 결과는 무엇인가?

```
이름을 입력하세요?Kitae Hwang
Kitae님 환영합니다.       // 빈칸 위치
```

13. cin.getline(buf, 100, ' ; ') 에 대한 설명으로 틀린 것은?
    1. buf는 아마 char buf[100]; 으로 선언되어 있을 것이다.
    2. 키보드로부터 최대 99개의 문자를 읽어 buf[ ] 배열에 저장한다.
    3. 키보드 입력 도중 ' ; ' 문자를 만나면 getline( ) 함수는 입력을 종료하고 끝에 '\0' 를 삽입하고 리턴한다.
    4. **cin.getline(buf, 100); 로 생략하여 써도 무관하다. (X , ' ; ' 를 생략하면 그 위치에 '\n' 이 들어가게 된다.)**

14. char buf[100]; 가 선언되어 있다고 가정하고, 다음과 같이 Enter 키가 입력될 때 까지 문자열을 읽는 코드로 잘못된 것은 무엇인가?

```
I love c++<Enter>
```

    1. cin >> buf;
    2. cin.getline(buf, 11);
    3. cin.getline(buf, 20, '\n');
    4. **cin.getline(buf, 11, '.'); (X , ' . ' 을 받을 때 입력을 종료시킨다.)**

15. C++ 에서 여러 사람들이 나누어 프로그램을 개발할 때 동일한 이름의 변수나 클래스, 함수 등이 충돌하는 것을 막기 위해, 개발자가 자신만의 이름 공간을 생성할 수 있도록 새로 도입한 키워드(혹은 개념)은 무엇인가?

    namespace

16. C++ 표준 라이브러리 모두 선언된 이름 공간은 무엇인가?

    std

17. C++ 표준에서 입출력을 위한 클래스, 함수, 객체들이 포함된 이름 공간은 무엇인가?

    std

18. C++ 표준에서 cin, cout 객체는 어떤 헤더 파일에 선언되어 있는가?

    iostream 헤더파일

19. 다음 화면에 나이와 학과를 출력하는 main.cpp 프로그램을 작성한 사례이다. 빈칸에 적절한 코드를 삽입하라.

```
#include <iostream>
using namespace std;
int main(){
    int age = 20;
    char *pDept = "컴퓨터 공학과";
    cout << age << " " << pDept;        // 빈칸 위치
}
```

    실행결과

```
20 컴퓨터 공학과
```

20. 다음 출력 결과와 같은 코드를 작성하고자 한다. 다음 C++ 프로그램을 완성하라.

    실행결과

```
*
**
***
****
```

    코딩

```
#include <iostream>
using namespace std;
int main() {
    for (int n = 0; n < 4; n++) {
        for (int i = 0; i <= n; i++) {  // 빈칸 위치
            cout << "*" ;                // 빈칸 위치
        }                               // 빈칸 위치
        cout << endl;                   // 빈칸 위치
    }
}
```

02 - 2) C언어 포인터와 구조체 복습

3주차 과제

- P = &a 와 P = *a 의 의미가 어떻게 다른지 설명하시오.

> P = &a 는 P에 a의 주소를 저장하는 것이고
>
> P = *a 는 P에 a의 주소의 저장된 값을 저장하는 것이다.

- 구조체 멤버 자료로 학번, 이름 및 점수를 가지는 구조체 배열을 정의하고, 10명의 학생에 대한 성적자료를 각각 키보드로 입력하여 정의된 구조체 배열에 저장하시오. 그리고 전체 총점과 평균 점수를 출력하는 프로그램을 작성하시오.

코딩

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

typedef struct Student {
    int num;
    char name[30];
    int grade;
}Student;

int main() {
    Student s[10];

    for (int i = 0; i < 10; i++) {
        cout << "이름 : ";
        cin >> s[i].name;
        cout << "학번 : ";
        cin >> s[i].num;
        cout << "점수 : ";
        cin >> s[i].grade;
    }

    int sum = 0;
    float average;

    for (int i = 0; i < 10; i++) {
        sum += s[i].grade;
    }
    average = sum / 10;

    cout << "전체 총점 : " << sum << endl;
    cout << "평균 : " << average << endl;
}
```

실행결과

```
이름 : a
학번 : 1
점수 : 10
이름 : b
학번 : 2
점수 : 20
이름 : c
학번 : 3
점수 : 30
이름 : d
학번 : 4
점수 : 40
이름 : e
학번 : 5
점수 : 50
이름 : f
학번 : 6
점수 : 60
이름 : g
학번 : 7
점수 : 70
이름 : h
학번 : 8
점수 : 80
이름 : i
학번 : 9
점수 : 90
이름 : j
학번 : 10
점수 : 100
전체 총점 : 550
평균 : 55
```

03) 클래스와 객체

3.1) 객체에 대한 이해

C++ 클래스와 C++ 객체

- **클래스(class)** : 객체(object)를 정의하는 틀 혹은 설계도, 클래스에 **멤버 변수** 와 **멤버 함수** 를 언선한다.
- **객체(object)** : 클래스의 모양을 그대로 가지고 메모리에 생성

CHECK TIME

1. C++ 에서 객체를 정의하는 틀을 무엇이라고 하는가?

   클래스

3.2) C++ 클래스 만들기

클래스 만들기

- **Class**

```
class Circle{   // class [클래스 이름]
public: // 멤버에 대한 접근 지정자
    int radius;         // 멤버 변수
    double getArea();   // 멤버 함수
};  // 클래스 선언부 (마지막에 반드시 세미콜론)

double Circle::getArea(){
    // [함수의 리턴 타입] [클래스 이름] :: [멤버 함수명]
    return 3.14 * radius * radius;
}
```

> Class 를 선언하고 마지막에 **세미콜론(;)** 을 반드시 써주자.

3.3) 객체 생성과 객체 활용

- **예제 3-1) Circle 클래스의 객체 생성 및 활용**

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

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

```
double Circle::getArea() {  // Circle 구현부
    return radius * radius * 3.14;
}

int main() {
    Circle donut;
    donut.radius = 1;   // 반지름 1로 설정
    double area = donut.getArea();  // donut 객체의 면적 계산
    cout << "donut의 면적은 " << area << " 입니다." << endl;
}
```

실행결과

```
donut 면적은 3.14 입니다.
```

3.4) 생성자

- **생성자** : 객체를 초기화 하기 위함.
- **생성자 특징**
  - 객체가 생성될 때 필요한 초기화를 한다.
  - 오직 한 번만 실행된다.
  - 함수의 이름은 클래스 이름과 동일하다
  - 리턴 타입을 선언하지 않으며 없다.
  - 중복 가능하다.
- **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 있는 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;
}

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

    Circle pizza(30);   // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
}
```

실행결과

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

기본 생성자

- **디폴트 생성자** : 매개 변수 없는 생성자
- **기본 생성자가 자동으로 생성되는 경우**

```
Circle donut;   // 기본 생성자 Circle() 호출
```

- **기본 생성자가 자동으로 생성되지 않는 경우**

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

Circle::Circle(int r){
    radius = r;
}

int main(){
    Circle pizza(3);    // 매개 변수가 있는 생성자가 있기 때문에 오류 x
    Circle donut;       // 기본 생성자가 없기 때문에 컴파일 오류!!!
}
```

- **예제 3-4) Rectangle 클래스 만들기**

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

class Rectangle {
public:
    int width, height;          // 너비, 높이
    Rectangle();                // 기본 생성자
    Rectangle(int w, int h);    // 2개의 매개변수 생성자
    Rectangle(int length);      // 1개의 매개변수 생성자
    bool isSquare();            // 멤버 함수
};

Rectangle::Rectangle() {
    width = height = 1;
}

Rectangle::Rectangle(int w, int h) {
    width = w;
    height = h;
}

Rectangle::Rectangle(int length) {
    width = height = length;
}

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 << "rect2은 정사각형이다." << endl;
    if (rect3.isSquare()) cout << "rect3은 정사각형이다." << endl;
}
```

실행결과

```
rect1은 정사각형이다.
rect3은 정사각형이다.
```

CHECK TIME

1. 다음 main( ) 함수에서 coin 객체 생성에 어떤 문제가 존재하는가? 문제를 해결해라.

```
class Circle{
public:
    int radius;
    double getArea();
};
int main(){
    Circle coin(3);
}
```

정답

```
class Circle{
public:
    int radius;
    Circle(int r);      // 매개변수가 1개인 생성자를 만들어준다.
    double getArea();
};
Circle::Circle(int r){
    radius = r;
}
int main(){
    Circle coint(3);
}
```

2. 다음 소스에서 컴파일 오류가 발생하는 라인은?

```
class Circle{
    int radius;
public:
    double getArea();
    Circle();
    Circle(int r);
    void Circle(short r);   // 오류 발생, 생성자는 반환형을 쓸수없다.
};

int main(){
    Circle waffle;
    Circle pizza(30);
    double d = pizza.getArea();
}
```

3.5) 소멸자

소멸자란?

- **소멸자** : 객체가 **소멸**되는 시점에서 자동으로 호출되는 클래스의 멤버 함수
- **소멸자의 특징**
  - 객체가 사라질 때 필요한 **마무리 작업**을 위함
  - 소멸자의 이름은 클래스 이름 앞에 ~를 붙인다.
  - 리턴 타입이 **없다**.
  - 오직 **한 개**만 존재
  - 선언되어 있지 않으면 **기본 소멸자** 자동 생성

소멸자 실행

- 생성된 **반대순으로** 객체가 **소멸**한다
- **ex) 실행결과**

```
반지름 1 원 생성
반지름 30원 생성
반지름 30원 소멸
반지름 1원 소멸
```

- **예제 3-6) 지역 객체와 전역 객체의 생성 및 소멸 순서**

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

class Circle {
public:
    int radius;
    Circle();
    ~Circle();
    Circle(int r);
};

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

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

Circle::~Circle() {
    cout << "반지름 " << radius << " 원 소멸" << endl;
}
```

```
Circle globalDonut(1000);   // 전역 객체 생성
Circle globalPizza(2000);   // 전역 객체 생성

void f() {
    Circle fDonut(100);     // 지역 객체 생성
    Circle fPizza(200);     // 지역 객체 생성
}

int main() {
    Circle mainDonut;       // 지역 객체 생성
    Circle mainPizza(30);   // 지역 객체 생성
    f();
}
```

실행결과

```
반지름 1000원 생성    // 전역
반지름 2000원 생성    // 전역
반지름 1 원 생성      // 지역 main
반지름 30 원 생성     // 지역 main
반지름 100 원 생성    // 지역 f()
반지름 200 원 생성    // 지역 f()
반지름 200 원 소멸    // 지역 f()
반지름 100 원 소멸    // 지역 f()
반지름 30 원 소멸     // 지역 main
반지름 1 원 소멸      // 지역 main
반지름 2000 원 소멸   // 전역
반지름 1000 원 소멸   // 전역
```

CHECK TIME

1. MyClass 클래스가 있다고 가정하면, 다음 코드에 의해 a, b, c, d 객체에 생성자와 소멸자가 실행되는 순서를 적어라.

```
MyClass a, b;
void f(){
    MyClass c;
}
int main(){
    f();
    MyClass d;
}
```

실행결과

```
생성 : a -> b -> c -> d
소멸 : d -> c -> b -> a
```

3.6) 접근 지정

접근 지정자

- **접근 지정자**

```
class Sample{
private:
    // private 멤버 선언. 클래스 내의 멤버 함수만 접근 가능
public:
    // public 멤버 선언. 클래스 내외의 모든 함수에게 접근 허용
protected:
    // protected 멤버 선언. 클래스 내의 멤버와 상속받은 파생 클래스에만 접근 허용
}
```

디폴트 접근 지정은 private

- **디폴트 접근 지정**

```
class Circle{
    int radius;     // 디폴트 접근 지정이기 때문에 private로 선언 된다.
public:
    Circle();
}
```

멤버 보호와 생성자

- **변수 멤버는 private 으로 지정하는 것이 바람직함** : 마음대로 접근하는 것을 막기 위함.
- **생성자는 public 으로 !!**

3.7) 인라인 함수

함수 호출에 따른 시간 오버헤드

: **짧은 코드**를 **함수**로 만들면, 함수 호출 오버헤드가 발생하여 실행 시간이 길어진다.

인라인 함수(inline function)

- **인라인 함수** : 짧은 코드로 구성된 함수에 대해, 함수 호출 오버헤드로 인한 프로그램의 실행 속도 저하를 막기 위한 방법이다.
- **제약 사항** : 재귀 함수, static 변수 등을 가진 함수는 인라인 함수로 **허용 X**

멤버 함수의 인라인 선언과 자동 인라인

- **ex) (a) 멤버 함수를 inline으로 선언하는 경우**

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

inline Circle::Circle(){    // inline 멤버 함수
    radius = 1;
}

Circle::Circle(int r){
    radius = r;
}

inline double Circle::getArea(){    // inline 멤버 함수
    return 3.14 * radius * radius;
}
```

- **ex) (b) 자동 inline으로 처리되는 경우**

```
class Circle{
private:
    int radius;
public:
    Circle(){   // 자동 인라인 함수
        radius = 1;
    }
    Circle(int r);
    double getArea(){   // 자동 인라인 함수
        return 3.14 * radius * radius;
    }
};

Circle::Circle(int r){
    radius = r;
}
```

CHECK TIME

1. 인라인 함수에 대해 잘못 설명한 것은?
    1. 인라인 함수를 사용하면 프로그램의 실행 속도가 증가한다.
    2. **인라인 함수를 사용하면 컴파일된 프로그램의 크기가 줄어들어 실행 속도를 증가시킨다. (X , 호출하는 곳이 여러 군데 있으면 그 만큼 전체 크기가 늘어나는 단점이 있다.)**

3. 컴파일러는 inline 으로 선언된 모든 함수를 인라인으로 처리하는 것은 아니다.
4. 생성자도 자동 인라인 함수로 만들 수 있다.

3.8) 클래스와 객체 (실습)

- **예제 3-4) Rectangle 클래스 만들기**

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

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

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 정사각형 이다." << endl;
    if (rect2.isSquare()) cout << "rect2은 정사각형 이다." << endl;
    if (rect3.isSquare()) cout << "rect3은 정사각형 이다." << endl;
}
```

실행결과

```

```

- **Tower 클래스는 height 멤버 변수와 2 개의 생성자, 그리고 getHeight( ) 함수를 가진다. (과제)**

```
#include <iostream>
```

```
using namespace std;

class Tower {
public:
    int height;
    Tower() {
        height = 1;
    }
    Tower(int length) {
        height = length;
    }
    int getHeight();
};

int Tower::getHeight() {
    return height;
}

int main() {
    Tower myTower;
    Tower seoulTower(100);

    cout << "높이는 " << myTower.getHeight() << " 미터" << endl;
    cout << "높이는 " << seoulTower.getHeight() << " 미터" << endl;
}
```

실행결과

```
높이는 1 미터
높이는 100 미터
```

연습 문제

이론 문제

1. 객체를 캡슐화하는 목적은 무엇인가?

   객체 외부의 접근으로부터 객체를 보호하기 위한 것이다.

2. 클래스와 객체에 관한 설명 중 틀린 것은?

   1. 객체를 실체 혹은 인스턴스(instance) 라고 부른다.
   2. 클래스는 객체를 생성하기 위한 설계도 혹은 틀과 같다.
   3. **클래스의 멤버들은 private 보다 public 접근 지정이 바람직하다. (X , pricate로 선언하는 것이 바람직하다.)**
   4. 클래스는 함수 멤버와 변수 멤버로 이루어진다.

3. 다음 C++ 코드가 객체 지향 언어의 캡슐화를 달성하고 있는지 설명하라.

```
int acc;
int add(int x){
    acc += x;
    return acc;
}
class Circle{
public:
    int radius;
    double getArea();
};
```

> acc 변수와 add( ) 함수는 **어떤 클래스에도 포함되어 있지 않아** 누구나 이들을 접근할 수 있기 때문에 캡슐화가 아니다. 또한 Circle 클래스의 멤버 변수 radius가 **public 속성**으로 되어 있으므로 적절하지 않다.

수정된 코드

```
class Count{        // acc 변수를 privat으로 선언하고
    int acc;        // add() 함수를 클래스 안에 선언하였다.
public:
    Count(int a){
        acc = a;
    }
    int add(int x){
        acc += x;
        return acc;
    }
    int getAcc(){
        return acc;
    }
};

class Circle{
    int radius;     //  radius 를 public 에서 private 으로 변경함.
```

```
public:
    double getArea();
};
```

4. 다음 C++ 프로그램에 캡슐화가 부족한 부분을 수정하여 캡슐화하라.

```
int age;
void older(){
    age++;
};
class Circle{
    int radius;
public:
    double getArea();
};
```

수정된 코드

```
class Age{          // 클래스 선언
    int age;        // age 변수를 private 선언
public:
    Age(int a){     // Age 생성자 선언
        age = a;
    }
    void older{     // 멤버 함수 선언
        age++;
    }
    int getAge(){   // age를 불러오기 위한 멤버 함수 선언
        return age;
    }
};
class Circle{
    int radius;
public:
    double getArea();
};
```

5. 다음 코드는 Circle 클래스의 선언부이다. 틀린 부분을 수정하라.

```
class Circle{
    int radius;
    double getArea();
}// 세미콜론 (;) 를 붙어야 한다.
```

6. 다음 코드는 Tower 클래스를 작성한 사례이다. 틀린 부분을 수정하라.

```
class Tower{
    int height = 20;    // 클래스의 선언부에서 변수를 초기화 할 수 없다.
public:
    Tower(){
        height = 10;
        return;
    }
};
```

7. 다음 코드에서 틀린 부분을 수정하라.

```
class Building{
private:
    int floor;
public:
    Building(int s){
        floor = s;
    }
    //  Building() { floor = 0; }
};
int main(){
    Building twin, star; // 이 줄은 기본 생성자가 없기 때문에 오류가 나므로 기본 생성자를 추가해야 한다.
    Building BlueHouse(5), JangMi(14);
}
```

8. 다음 코드는 Calendar 클래스의 선언부이다. year를 10으로 초기화하는 생성자와 year 값을 리턴하는 getYear()를 구현하라.

```
class Calender{
private:
    int year;
public:
    Calender();
    int getYear();
};
```

정답

```
Calender::Calender(){
    year = 10;
}
int Calender::getYear(){
    return year;
}
```

9. 생성자에 대한 설명 중 틀린 것은?

1. 생성자의 이름은 클래스 이름과 같다.
2. **생성자는 오직 하나만 작성 가능하다. (X , 중복가능하다)**
3. 생성자는 리턴 타입을 가지지 않는다.
4. 생성자가 선언되어 있지 않으면 컴파일러에 의해 기본 생성자가 삽입된다.

10. 소멸자에 대한 설명 중에 틀린 부분을 지적하라.

> 소멸자는 (1) 객체가 소멸되는 시점에 자동으로 호출되는 멤버함수 로서 (2) 클래스의 ~를 붙인 이름으로 선언 되어야 한다. (3) 매개 변수 있는 소멸자를 작성하여 소멸 시에 의미 있는 값을 전달할 수 있으며, 소멸자가 선언되어 있지 않으면 (4) 기본 소멸자가 자동으로 생성된다.

(3), 소멸자는 매개 변수를 받지 않는다.

11. 다음 프로그램에 대해 답하여라.

```
class House{
    int numOfRooms;
    int size;
public:
    House(int n, int s);    // n과 s로 numOfRooms, size를 각각 초기화
};

void f(){
    House a(2, 20);
}

House b(3, 30), c(4, 40);

int main(){
    f();
    House d(5, 50);
}
```

(1) n 과 s 로 numOfRooms, size를 각각 초기화하고 이들을 출력하는 생성자를 구현하라.

```
House::House(int n, int s){
    numOfRooms = n;
    size =s;
    cout << "방의 개수는 " << numOfRooms << " 이고 크기는 " << size << " 이다." <<
endl;
}
```

(2) size 와 numOfRooms 값을 출력하는 House 클래스의 소멸자를 작성하라.

```
~House();   // class 안에 선언

House::~House(){
    cout << size << " " << numOfRooms << endl;
}
```

(3) 객체 a, b, c, d 가 생성되는 순서와 소멸되는 순서는 무엇인가? (순서 햇갈리니 조심 !!!!!!) 중요!!!!!!

```
b생성 -> c생성 -> a생성 -> a소멸 -> d생성 -> d소멸 -> c소멸 -> b소멸
```

12. 다음 프로그램에서 객체 a, b, c 가 생성되고 소멸되는 순서는 무엇인가?

```
class House{
    int numOfRooms;
    int size;
public:
    House(int n, int s){
        numOfRooms = n;
        size = s;
    }
    void test(){
        House a(1, 10);
    }
    void f(){
        House b(2, 20);
        b.test();
    }
};
House c(3, 30);
int main(){
    f();
}
```

생성, 소멸 순서

```
c생성 -> b생성 -> a생성 -> a 소멸 -> b소멸 -> c소멸
```

13. 다음 프로그램의 오류를 지적하고 수정하라.

```
class TV{
    TV(){
        channels = 256;
    }
public:
    int channels;
    TV(int a){
        channels = a;
    }
};

int main(){
    TV LG;
    LG.channels = 200;
    TV Samsung(100);
}
```

수정한 코드

```
class TV{
public:
    int channels;   // public 으로 선언
    TV(){           // private 에서 public 으로 변경
        channels = 256;
    }
    TV(int a){
        channels = a;
    }
};

int main(){
    TV LG;
    LG.channels = 200;
    TV Samsung(100);
}
```

14. 다음 프로그램의 오류를 지적하고 수정하라.

```
class TV(){
    int channels;
public:
    int colors;
    TV(){
        channels = 256;
    }
    TV(int a, int b){
        channles = a;
        colors = b;
    }
};

int main(){
    TV LG;
    LG.channels = 200;
    LG.colors = 60000;
    TV Samsung(100, 50000);
}
```

> main( ) 함수에서 LG.channels = 200; 에서 컴파일 오류가 발생한다. channels 가 private 으로 선언되어 있기 때문에 main( ) 에서는 접근할 수 없다.

수정한 코드

```
class TV(){
    int channels;
    int colors;
public:
    TV(){
        channels = 256;
    }
    TV(int a, int b){
```

```
        channles = a;
        colors = b;
    }
    void setChannels(int n){
        channels = n;
    }
    void setColor(int n){
        colors = n;
    }
};

int main(){
    TV LG;
    LG.setChannels(200);
    LG.setColors(60000);
    TV Samsung(100, 50000);
}
```

15. 다음 코드에서 자동 인라인 함수를 찾아라.

```
class TV{
    int channels;
public:
    TV(){       // 자동 인라인 함수
        channels = 256;
    }
    TV(int a){  // 자동 인라인 함수
        channels = a;
    }
    int getChannels();
};
inline int TV::getChannels(){
    return channels;
}
```

16. 인라인 함수의 장단점을 설명한 것 중 옳은 것은?

    1. 인라인 함수를 사용하면 컴파일 속도가 향상된다. (X)
    2. **인라인 함수를 이용하면 프로그램의 실행 속도가 향상된다. (O)**
    3. 인라인 함수를 사용하면 프로그램 작성 시간이 향상된다. (X)
    4. 인라인 함수를 사용하면 프로그램의 크기가 작아져서 효과적이다. (X)

17. 인라인 함수에 대해 잘못 설명한 것은?

    1. **인라인 선언은 크기가 큰 함수의 경우 효과적이다. (X)**
    2. C++ 프로그램에는 크기가 작은 멤버 함수가 많기 때문에 이들을 인라인으로 선언하면 효과적이다.
    3. 컴파일러는 먼저 인라인 함수를 호출하는 곳에 코드를 확장시킨 후 컴파일 한다.
    4. 인라인 함수는 함수 호출에 따른 오버헤드를 줄이기 위한 방법이다.

18. inline 선언은 강제 사항이 아니다. 다음 함수 중에서 컴파일러가 인라인으로 처리하기에 가장 바람직한 것은?

1.

```
inline int big(int a, int b){
    return a > b ? a : b;
}
```

2.

```
inline int sum(int a, int b){
    if(a >= b)
        return a;
    else
        return a + sum(a+1, b);
}
```

3.

```
inline void add(int a, int b){
    int sum = 0;
    for(int n = a; n < b ; n++)
        sum += n;
}
```

4.

```
inline int add(int a){
    static int x = 0;
    x += a;
    return x;
}
```

> (1), (2) 는 재귀함수, (3) 은 반복문, (4) 는 static 변수를 가지고 있어서 컴파일러에 따라서는 인라인으로 처리하지 않을 가능성이 있다. 그러므로 답은 없다.

19. C++ 구조체(struct) 에 대해 잘못 설명한 것은?

    1. C++ 에서 구조체를 둔 이유는 C 언어와의 호환성 때문이다.
    2. C++ 에서 구조체는 멤버 함수와 멤버 변수를 둘 수 있다.
    3. C++ 에서 구조체는 생성자와 소멸자를 가진다.
    4. **C++ 에서 구조체는 상속을 지원하지 않는다. (X , 상속 가능)**

20. 다음 C++ 구조체를 동일한 의미를 가지는 클래스로 작성하라.

```
struct Family{
    int count;
    char address[20];
public:
    Family();
private:
    char tel[11];
};
```

답

```
class Family{
    char tel[11];
public:
    int count;          // 구조체의 디폴트는 모두 public 이다.
    char address[20];
    Family();
}
```

21. 다음 클래스를 구조체로 선언한다.

```
class Universe{
    char creator[10];
    int size;
private:
    char dateCreated[10];
public:
    Universe();
};
```

답

```
struct Universe{
    Universe();
private:
    char creator[10];       // 클래스의 디폴트는 private 이다.
    int size;
    char dateCreated[10];
};
```

04) 객체 포인터와 객체 배열, 객체의 동적 생성

4.1) 객체 포인터

- 객체에 대한 포인터 변수 선언

```
Circle *p;
```

- 포인터 변수에 객체 주소 지정

```
p = &donut; // p에 donut 객체의 주소 저장
Circle* p = &donut; // 포인터 변수 선언 시 객체 주소로 초기화
```

- 포인터를 이용한 객체 멤버 접근

```
d = donut.getArea();    // 객체 이름으로 (.) 연산자를 이용하여 멤버 함수 호출
d = p->getArea();   // 포인터로 객체 멤버 함수 호출
d = (*p).getArea(); // 위와 같다.
```

- 예제 4-1) 객체 포인터 선언 및 활용

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

class Circle {
    int radius;
public:
    Circle() {
        radius = 1;
    }
    Circle(int r) {
        radius = r;
    }
    double getArea();
};

double Circle::getArea() {
    return 3.14 * radius * radius;
}

int main() {
    Circle donut;
    Circle pizza(30);

    // 객체 이름으로 멤버 접근
    cout << donut.getArea() << endl;    // 3.14 출력
```

```
    // 객체 포인터로 멤버 접근
    Circle *p;
    p = &donut;
    cout << p->getArea() << endl;   // donut의 getArea() 호출
    cout << (*p).getArea() << endl; // donut의 getArea() 호출

    p = &pizza;
    cout << p->getArea() << endl;   // pizza의 getArea() 호출
    cout << (*p).getArea() << endl; // pizza의 getArea() 호출
}
```

> **p->getArea( )** 와 **(*p).getArea( )** 가 같은 의미이다.

CHECK TIME

1. public 멤버 함수 draw()를 가진 Polygon 클래스가 있을 때, 다음 두 선언문에 대해 물음에 답하여라.

```
Polygon poly;
Polygon *p;
```

(1) 포인터 p를 활용하여 poly 객체의 draw() 함수를 호출하는 코드를 두 줄로 작성하라.

```
p = &poly;
p->draw();
```

(2) 다음 중에서 다른 하나는 무엇인가?

1. poly.draw( );
2. p = &poly; p->draw( );
3. p = &poly; (*p).draw( );
4. **poly->draw( ); (X)**

4.2) 객체 배열

객체 배열 선언 및 활용

- **객체 배열 선언**
  - 배열의 각 원소 객체마다 기본 생성자 실행
  - **매개 변수 있는 생성자를 호출할 수 없음**

```
Circle circleArray[3](5);   // 오류
```

- **배열 소멸과 소멸자**
  - **원소 객체마다 소멸자가 호출**된다.

객체 배열 초기화

- **원소 객체를 초기화**

```
Circle circleArray[3] = { Circle(10), circle(20), Circle() };
```

다차원 객체 배열

- **다차원**

```
Circle circles[2][3];   // 2행 3열의 2차원 객체 배열 생성
```

CHECK TIME

1. 다음 클래스에 대해 물음에 답하여라.

```
class Sample{
    int a;
public:
    Sample(){
        a = 100;
        cout << a << ' ';
    }
    Sample(int x){
        a = x;
        cout << a << ' ';
    }
    Sample(int x, int y){
        a = x * y;
        cout << a << ' ';
    }
    int get(){
        return a;
    }
};
```

1. Sample arr[3]; 이 실행될 때 출력되는 결과는?

```
100 100 100
```

2. 다음과 같은 코드가 실행될 때 출력되는 결과는?

```
Sample arr2D[2][2] = {
    {Sample(2,3), Sample(2,4)},
    {Sample(5), Sample()};
};
```

실행결과

```
6 8 5 100
```

3. 객체 포인터를 이용하여 (1) 에서 선언된 arr의 모든 원소 (a) 의 합을 출력하는 for문을 작성하라.

```
int sum = 0;
for (int i = 0; i < 3; i++)
{
    sum += arr[i].get();
}
cout << sum << endl;
```

4. (2) 에서 선언된 arr2D 배열 이름을 이용하여 모든 원소 (a)의 합을 출력하는 for문을 작성하라.

```
int sum = 0;
for (int i = 0; i < 2; i++)
{
    for(int j = 0 ; j < 2 ; j++)
        sum += arr2D[i][j].get();
}
cout << sum << endl;
```

4.3) 동적 메모리 할당 및 반환

C++ 의 동적 메모리 할당/반환

- **new 연산자**
  - 메모리 할당
- **delete 연산자**
  - 메모리 반환
  - 동적으로 할당 받지 않는 메모리 반환 - **오류!!**
  - 동일한 메모리 두 번 반환 - **오류!!**

new와 delete 연산자

- **기본 형식**

```
데이터타입 *포인터변수 = new 데이터타입;
delete 포인터변수;
```

- **ex)**

```
int *pInt = new int;         // int 타입의 정수 공간 할당
char *pChar = new char;      // char 타입의 문자 공간 할당
Circle *pCircle = new Circle();     // Circle 클래스 타입의 객체 할당

delete pInt;    // 할당받은 정수 공간 반환
delete pChar;   // 할당받은 문자 공간 반환
delete pCircle; // 할당받은 객체 공간 반환
```

- **동적 할당 메모리 초기화**
  - **기본 형식**

    ```
    데이터타입 *포인터변수 = new 데이터타입(초깃값);
    ```

  - **ex)**

    ```
    int *pInt = new int(20);         // 20으로 초기화된 int 공간 할당
    char *pChar = new char('a');     // 'a'로 초기화된 char 공간 할당
    ```

배열의 동적 할당 및 반환

- **기본 형식**

```
데이터타입 *포인터변수 = new 데이터타입 [배열의 크기]; // 배열의 동적 할당
delete [] 포인터변수;    // 배열 메모리 반환
```

- **ex)**

```
int *p = new int [5];   // 크기가 5인 정수형 배열의 동적 할당
delete [] p;            // 배열 메모리 반환
```

CHECK TIME

1. 다음 물음에 대한 간단한 코드를 보여라.

   (1) 1개의 double 공간을 동적으로 할당받고 3.14 를 기록하라.

   ```
   double *p = new double(3.14);
   ```

   (2) 배열을 동적 할당받고 5개의 정수를 입력받아 저장한 후, 제일 큰 수를 출력하고 배열을 반환한다.

```
int *p = new int[5];
int sum = 0;
for (int i = 0; i < 5; i++) {
    cin >> p[i];
}
int max = p[0];
for (int i = 0; i < 5; i++) {
    if (max < p[i])
        max = p[i];
}
cout << max << endl;
delete[] p;
```

2. 다음 중 틀린 라인을 골라 수정하라.

    1.
    ```
    int *p = new int(3);
    int n = *p;
    delete[] p;     // p는 배열이 아니므로 delete p; 로 수정한다.
    ```

    2.
    ```
    char *p = new char[10];
    char *q = p;
    q[0] = 'a';
    delete[] q;
    delete[] p; // p를 가리키고 있는 q를 메모리 반환 해줬으므로 이 줄은 지우도록 한다.
    ```

4.4) 객체와 객체 배열의 동적 생성 및 반환

객체의 동적 생성 및 반환

- **기본 형식**

```
클래스이름 *포인터변수 = new 클래스이름;   // 기본 생성자 호출
클래스이름 *포인터변수 = new 클래스이름(생성자매개변수리스트);
// 매개 변수 있는 생성자 호출
delete 포인터변수;   // 객체 반환
```

- **ex)**

```
Circle *p = new Circle;     // 기본 생성자 Circle() 호출
Circle *q = new Circle(30); // 생성자 Circle(int r) 호출
delete p;   // Circle 객체 반환
```

객체 배열의 동적 생성 및 반환

- **기본 형식**

```
클래스이름 *포인터변수 = new 클래스이름 [배열 크기];   // 동적 생성
delete [] 포인터변수;    // 반환
```

- **ex)**

```
Circle *pArray = new Circle[3]; // new Circle[3](30) 컴파일 오류!!
delete[] pArray;    // 반환
```

- **예제 4-10) Circle 배열의 동적 생성 및 반환 응용**

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

class Circle {
    int radius;
public:
    Circle();
    ~Circle() {};
    void setRadius(int r) {
        radius = r;
    }
    double getArea() {
        return 3.14 * radius * radius;
    }
};

Circle::Circle() {
    radius = 1;
}

int main() {
    cout << "생성하고자 하는 원의 개수? ";
    int n, radius;
    cin >> n;
    if (n <= 0) return 0;
    Circle *pArray = new Circle[n];     // 객체 배열의 동적 생성
    for (int i = 0; i < n; i++) {
        cout << "원" << i + 1 << ":";
        cin >> radius;
        pArray[i].setRadius(radius);    // 반지름 초기화
    }
    int count = 0;
    Circle *p = pArray;     // 포인터 p에 배열의 주소값 설정
    for (int i = 0; i < n; i++) {
        cout << p->getArea() << ' ';
        if (p->getArea() >= 100 && p->getArea() <= 200)
            count++;
        p++;    // 다음 원소의 주소로 이동
    }
    cout << endl << "면적이 100에서 200 사이인 원의 개수는 " << count << endl;
```

```
    delete[] pArray;    // 객체 배열 반환
}
```

실행 결과

```
생성하고자 하는 원의 개수? 4
원1:5
원2:6
원3:7
원4:8
78.5 113.04 153.86 200.96
면적이 100에서 200 사이인 원의 개수는 2
```

4.5) this 포인터

this의 기본 개념

- **객체 자신에 대한 포인터**
- **멤버 함수 내에서만 사용**
- **ex)**

```
Circle(int radius){
    this->radius = radius;
}
```

- **제약 조건**
  - **정적 멤버 함수(static 멤버 함수)**는 this를 사용할 수 없다.
  - 멤버 함수가 아닌 함수에서 this 사용 불가

CHECK TIME

1. this에 대해 잘못 설명한 것은?
    1. this는 포인터이다.
    2. this는 static 타입을 제외한 객체의 모든 멤버 함수에서 사용할 수 있다.
    3. **this는 컴파일러가 삽입해주는 전역 변수로서 현재 실행 중인 객체에 대한 주소를 가진다. (X , 전역 변수가 아니다.)**
    4. 멤버 함수에서 this를 리턴할 수 있다.

4.6) string 클래스를 이용한 문자열 사용

string 클래스

- **ex)**

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

string str = "I love ";
str.append("C++");  // str은 "I love C++" 이 된다.
```

- **string 객체 생성**

```
string str;     // 빈 문자열을 가진 스트링 객체
string address("대전 월평");    // 문자열 리터럴로 초기화
string copyAddress(address);    // address를 복사한 copyAddress 생성
```

- **string 객체의 동적 생성**

```
string *p = new string("C++");  // 스트링 객체 동적 생성
cout << *p;     // "C++" 출력
p->append(" Great!!");  // p가 가리키는 스트링이 "C++ Great!!" 가 됨.
cout << *p;
delete p;   // 스트링 객체 반환
```

string 객체에 문자열 입력

- **string 객체에 문자열 입력**

```
string name;
cin >> name;    // 공백 문자를 포함하는 문자열을 읽어 들일 수 없다.
getline(cin, name, '\n');   // '\n'을 만날 때까지 문자열을 읽는다.
```

- **string 주요 멤버 함수**

멤버 함수	설명
insert(int pos, string& str)	문자열의 pos 위치에 str 삽입
replace(int pos, int n, string& str)	문자열 pos 위치부터 n개 문자를 str 문자열로 대치
int length( )	문자열 길이 리턴
int find(stirng& str)	str을 검색하여 발견한 처음 인덱스 리턴, 없으면 -1 리턴
swap(str1, str2)	str1과 str2를 서로 바꿔치기 함

- **string 클래스의 연산자**

연산자	설명
s1 = s2	s2를 s1에 치환
s [ ]	s의 [ ] 인덱스에 있는 문자
s1 + s2	s1 과 s2를 연결한 새로운 문자열
s1 += s2	s1에 s2 문자열 연결
s1 == s2	s1과 s2가 같은 문자열이면 true
s1 < s2	s1이 사전 순으로 s2 보다 앞에 오면 true

- **예제 4-13) 문자열을 입력 받고 회전시키기**

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

int main(){
    string s;

    cout << "아래에 문자열을 입력하세요. " << endl;
    getline(cin, s, '\n');  // 문자열 입력
    int len = s.length();   // 문자열의 길이

    for(int i = 0; i < len ; i++){
        string first = s.substr(0, 1);  // 맨 앞의 문자 1개를 문자열로 분리
        string sub = s.substr(1, len-1);    // 나머지 문자들을 문자열로 분리
        s = sub + first;    // 두 문자열을 연결하여 새로운 문자열 만듦
        cout << s << endl;
    }
}
```

CHECK TIME

1. 다음 프로그램의 실행 결과는 무엇인가?

```
string a("Hello C++");
cout << a.length() << endl;
a.append("!!");
cout << a << endl;
cout << a.at(6) << endl;
cout << a.find("C") << endl;
int n = a.find("+++");
cout << n << endl;
a.erase(1, 3);
cout << a << endl;
```

실행결과

```
9
Hello C++!!
C
6
-1
Ho C++!!
```

연습문제

이론 문제

1 ~ 5 번 문제에 사용되는 Rect 클래스. Rect 클래스는 폭과 높이로 사각형을 추상화 한다.

```
class Rect{
    int width, height;
public:
    Rect(int w, int h){
        width = w;
        height = h;
    }
    int getWidth(){
        return width;
    }
    int getHeight(){
        return height;
    }
    int getArea();
};

int Rect::getArea(){
    return width * height;
}
```

1. Rect의 객체를 다루는 다음 코드를 작성하려고 한다. 아래의 문제에 따라 빈칸에 적절한 코드를 삽입하라.

```
int main(){
    Rect r(2,3);
    // (1) Rect 클래스에 대한 포인터 변수 p를 선언하라.
    Rect *p;    // 빈칸위치
    // (2) 선언된 포인터 변수 p에 객체 r의 주소를 지정하라.
    p = &r;     // 빈칸위치
    // (3) 포인터 변수 p를 이용하여 객체 r의 폭과 높이를 출력하라.
    cout << p->getWidth() << p->getHeight();    // 빈칸위치
}
```

2. 사용자로부터 폭과 높이 값을 입력받아 동적으로 Rect 객체를 생성하고 면적을 구하여 출력하는 코드를 작성하고자 한다. 다음 물음에 따라 빈칸을 채워라.

```
int main(){
    Rect *q;
    int w, h;
    cin >> w >> h;  // 사용자로부터 사각형의 폭과 높이를 w, h에 각각 입력받는다.
    // (1) 포인터 변수 q에 wxh 크기의 사각형을 표현하는 Rect 객체를 동적으로 생성한다.
    q = new Rect(w, h);     // 빈칸 위치
    // (2) 포인터 q를 이용하여 사각형의 면적을 출력한다.
    cout << q->getArea() << endl;   // 빈칸 위치
    // (3) 생성한 객체를 반환한다.
    delete q;   // 빈칸 위치
}
```

3. Rect 객체나 배열을 생성하는 다음 코드 중 컴파일 오류가 발생하는 것은?
    1. **Rect a; (X , 기본 생성자 Rect( ) 가 없기 때문에)**
    2. Rect b(5, 6);
    3. Rect c[2] = { Rect(1, 1), Rect(2, 3) };
    4. Rect d[[2][3]][] = { {Rect(1, 2), Rect(2, 3), Rect(3, 4) } , { Rect(1, 1), Rect(2, 2), Rect(3, 3) } };

4. Rect 객체의 배열을 생성하는 다음 코드는 컴파일 오류가 발생한다. 컴파일 오류가 발생하지 않기 위해 Rect 클래스를 어떻게 수정하여야 하는가?

```
Rect *p = new Rect[10];
```

수정한 코드

```
// 클래스 안에 기본 생성자를 추가한다.
Rect(){
    width = height = 1;
}
```

5. Rect 클래스에 다음과 같은 기본 생성자를 삽입하고,

```
Rect(){ width = 1; height = 1;}
```

다음 배열 r 생성 후, 배열 r의 사각형 면적의 합을 출력하는 코드를 작성하라.

```
Rect r[5] ={
    Rect(), Rect(2, 3), Rect(3, 4), Rect(4, 5), Rect(5, 6)
};
```

코드

```
int sum = 0;
for(int i = 0 ; i < 5 ; i++){
    sum += r[i].getArea();
}
cout << sum << endl;
```

6. public 속성의 getVolume() 멤버 함수를 가진 Cube 클래스에 대해, 다음 코드가 있다.

```
Cube c;
Cube *p = &c;
```

다음 중 컴파일 오류가 발생하는 것은?

1. c.getVolume( );
2. p->getVolume( );
3. (*p).getVolume( );
4. **c->getVolume( ); (X , c는 포인터가 아니므로 (.) 연산자를 사용해야 한다)**

7. 다음 객체 배열에 관해 잘못 설명된 것은?

```
Cube c[4];
```

1. 배열 c가 생성될 때 c[0], c[1], c[2], c[3] 의 4개의 Cube 객체가 생성된다.
2. 기본 생성자 Cube( )가 4번 호출된다.
3. 배열 c가 소멸될 때 c[3], c[2], c[1], c[0]의 순서로 소멸자가 실행된다.
4. **delete c; 코드로 배열 c를 소멸한다. (X , delete[ ] c 로 반환해야한다.)**

8. 다음 프로그램이 실행될 때 출력되는 결과는 무엇인가?

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

class Color{
    string c;
public:
    Color(){
        c = "white";
        cout << "기본생성자" << endl;
    }
    Color(string c){
        this->c = c;
        cout << "매개변수생성자" << endl;
    }
    ~Color(){
        cout << "소멸자" << endl;
    }
};

class Palette{
    Color *p;
public:
    Palette(){
        p = new Color[3];
    }
```

```
    ~Palette(){
        delete[] p;
    }
};

int main(){
    Palette *p = new Palette();
    delete p;
}
```

실행 결과

```
기본생성자
기본생성자
기본생성자
소멸자
소멸자
소멸자
```

9. new와 delete는 무엇인가?
    1. **C++ 의 기본 연산자**
    2. C++ 표준 함수
    3. C++ 의 표준 객체
    4. C++ 의 특수 매크로

10. 다음 코드의 문제점은 무엇인가?

```
Cube *p = new Cube [4];
delete p;   // delete[] p 로 수정하여야 한다.
```

11. this에 대해 잘못 말한 것은?
    1. this 는 포인터 이다.
    2. this 는 컴파일러에 의해 묵시적으로 전달되는 매개 변수이다.
    3. **this 는 static 함수를 포함하여 멤버 함수 내에서만 다루어지는 객체 자신에 대한 포인터이다. (X , this 는 static 멤버 함수에서 사용할 수 없다.)**
    4. 연산자 중복에서 this 가 필요하다

12. this 의 활용에 대해 잘못 설명한 것은?
    1. this 는 클래스의 멤버 함수 외의 다른 함수에서는 사용할 수 없다.
    2. this 는 static 멤버 함수에는 사용할 수없다.
    3. **this 는 생성자에게 사용할 수 없다. (X , 사용가능)**
    4. 어떤 멤버 함수에서는 this 를 리턴하기도 한다.

13. this 를 최대한 많이 활용하여 다음 클래스를 가장 바람직하게 수정하라.

```
class Location{
    int width, height;
public:
    Location(){
        width = height = 0;
    }
    Location(int w, int h){
        width = w;
        height = h;
    }
    void show();
};
void Location::show(){
    cout << width << height << endl;
}
```

수정한 코딩

```
class Location{
    int width, height;
public:
    Location(){
        this->width = this->height = 0;
    }
    Location(int width, int height){
        this->width = width;
        this->height = height;
    }
    void show();
};
void Location::show(){
    cout << this->width << this->height << endl;
}
```

14. 메모리 누수란 어떤 상황에서 발생하는가?

    할당받은 동적 메모리에 대한 포인터를 잃어버리게 되어, 동적 메모리를 사용하지도 않고 반환할 수도 없게 된 상황에서 발생한다.

15. 함수 f( ) 가 실행되고 난 뒤 메모리 누수가 발생하는지 판단하고 메모리 누수가 발생하면 발생하지 않도록 수정하라.

    1.

    ```
    void f(){
        char *p = new char[10];
        strcpy(p, "abc");
        // delete[] p;  를 써주어 메모리를 반환한다.
    }
    ```

2.

```
void f(){
    int *p = new in;
    int *q = p;
    delete q;
}   // 메모리 누수 발생 X
```

3.

```
int f(){
    int n[10] = {0};
    return n[0];
}   // 메모리 누수 발생 X
```

4.

```
void f(){
    int *p;
    for(int i = 0 ; i < 5 ; i++){
        p = new int;    // 메모리 할당
        cin >> *p;
        if(*p % 2 == 1){
            delete p;   // 메모리 반환
            break;
        }
        delete p;   // 메모리 반환
    }
}
```

16. string 클래스를 사용하기 위해 필요한 헤더 파일은 무엇인가?

    string

17. string s1 = "123"; string s2 = "246"; 일 때 , a와 b의 문자열 속에 있는 수를 더하여 369를 출력하고자 한다. 아래 빈칸을 채워라.

```
int n = stoi(s1);   // 빈칸 stoi
int m = stoi(s2);   // 빈칸 stoi
cout << n + m;
```

18. 문자열을 다루고자 한다. C-스트링과 string 클래스에 대해 설명이 틀린 것은?

    1. C-스트링은 문자의 배열을 이용하여 문자열을 표현한다.
    2. string 클래스가 문자열을 객체화하므로 C-스트링보다 사용하기 쉽다.
    3. **string 클래스가 좋기는 하지만 C++의 표준이 아니므로 가급적 사용하지 않는 것이 좋다. (X , string 클래스는 C++의 표준이다.)**
    4. string 클래스는 문자열만 다루지 대문자를 소문자로 변환하는 등 문자를 조작하는 기능은 없다.

19. 다음 프로그램의 각 라인을 string 클래스에서 제공하는 연산자를 이용하여 고쳐라.

```
string a("My name is Jane.");
char ch = a.at(2);
if(a.compare("My name is John.") == 0 ) cout << "same";
a.append("~~");
a.replace(1, 1,"Y");
```

수정한 코딩

```
string a = "My name is Jane.";
char ch = a[2];
if(a == "My name is John.") cout << "same";
a += "~~";
a[1] = "Y";
```
