

[C++]

4주차: 객체, 클래스, 생성자, 소멸자

1
2
3
4
5

목차	내용
객체와 클래스	<ul style="list-style-type: none">- 클래스, 객체의 정의와 관계- 클래스 생성- 객체 생성
생성자	<ul style="list-style-type: none">- 생성자란?- 기본 생성자- 위임 생성자
소멸자	<ul style="list-style-type: none">- 소멸자란?- 생성, 소멸의 순서
접근지정자	<ul style="list-style-type: none">- 접근지정자란?- 접근지정자 특징
헤더파일과 cpp파일	<ul style="list-style-type: none">- 헤더 파일 과 cpp파일- 헤더 파일 중복과 include 문제- 헤더 파일과 cpp파일 분리

01

객체와 클래스

1. 클래스, 객체의 정의와 관계
2. 클래스 생성
3. 객체 생성

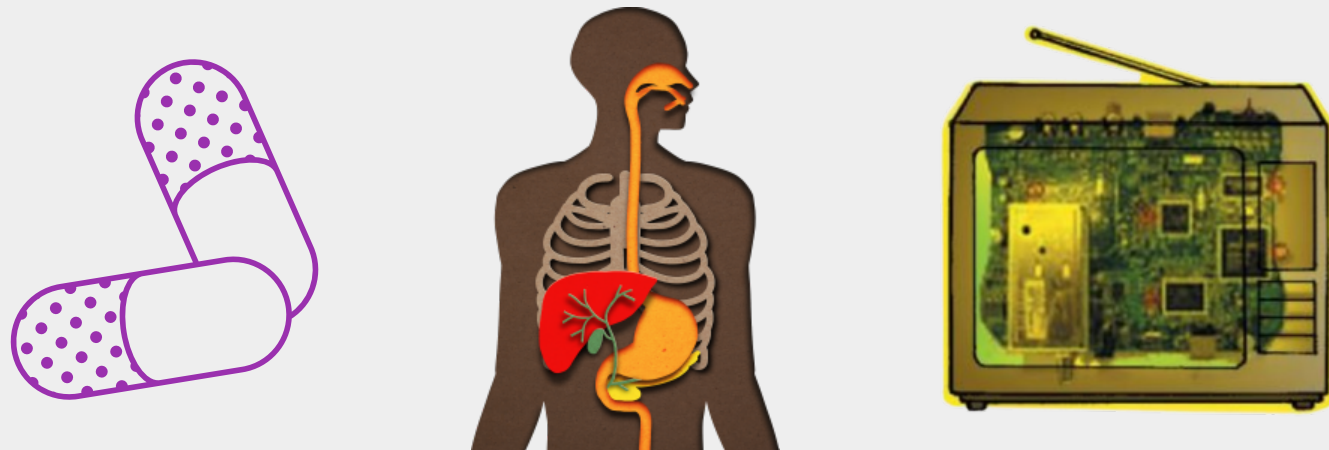
객체

세상의 모든 것 (TV, 카메라, 집, 컴퓨터 등...)

특징

캡슐화

- 객체의 본질적인 특성
 - 객체를 캡슐로 감싸서 내부를 보호하고 볼 수 없게 함
- 예시) 알약, 전자 기기, 인체 등...
- 객체 내, 데이터에 대한 보안, 보호, 외부 접근 제한



일부 공개

- 외부와 상호작용(정보 교환 & 통신)하기 위해 객체의 일부분을 공개

예시) TV 객체 - 리모컨 객체 통신

>>전원 버튼, 밝기/채널/음량 조절 버튼 노출

C++ 객체

- 객체는 **상태와 행동**으로 구성됨

예시)

TV 객체

- 상태

- on/off 속성 – 현재 전원 상태 표시
- 채널(channel) – 현재 방송중인 채널
- 음량(volume) – 현재 나오는 소리 크기

- 행동

- 전원 켜기
- 전원 끄기
- 채널 증가
- 채널 감소
- 음량 키우기
- 음량 줄이기

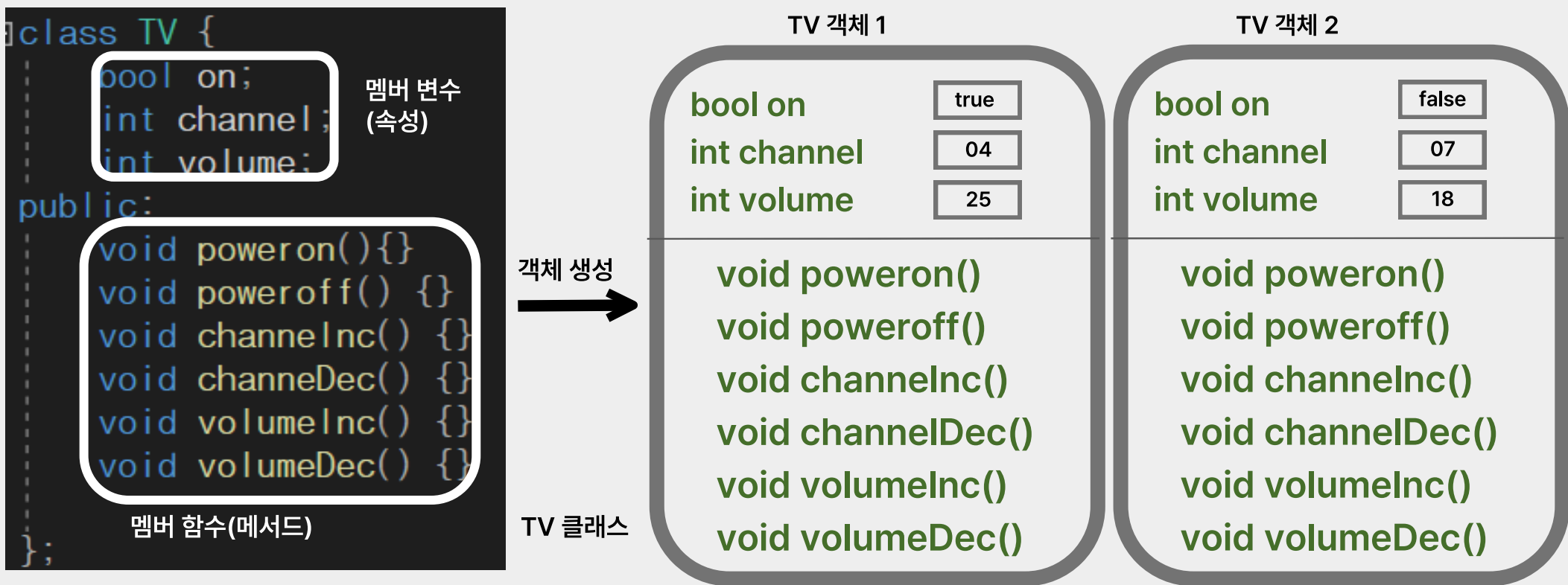
클래스

공통된 속성을 한데 모아 하나의 자료형으로 만든 것

특징

사용자 정의형

- 사용자가 정의한 자료형으로 기본 타입과 구별하기 위한 클래스의 다른 이름
- 멤버 변수와 멤버 함수 선언



- 새로운 타입일 뿐, 메모리에 저장된 변수가 아니다.
>>클래스를 통해 객체를 선언하고 초기화해줘야만 객체 사용 가능
- * int a;처럼 타입을 통해 변수를 선언하는 것처럼 클래스를 통해 객체 초기화 예시)

기본 타입 -> 변수		클래스 -> 객체	
int	a	TV	tv1
float	b	Car	car2

```
bool on = true;  
int channel = 05;  
int volume = 25;
```

```
bool on = false;  
int channel = 07;  
int volume = 18;
```

클래스 자료형: TV
객체 이름: sbs, kbs

```
int main(){  
    TV sbs(true, 05, 25);  
    TV kbs(false, 07, 18);  
}
```

클래스

붕어빵 틀

-
- 객체를 만들어내기 위해 정의된 틀/ 설계도
 - 객체도 실체도 아님
 - 멤버 변수와 멤버 함수를 선언
-

객체

붕어빵

-
- 클래스의 모양 그대로 가지고 생성됨
 - 멤버변수+멤버 함수로 구성
 - 하나의 클래스에서 여러개 생성 가능
 - 객체들은 상호 별도의 공간에서 각각 생성
-

클래스 생성

클래스 선언부

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

class Triangle {
public:
    int width;
    int height;
    //멤버 변수

    int area(); // 멤버 함수
};
```

class 키워드로
클래스 선언

클래스 이름

접근 지정자

세미콜론으로
마침

클래스 구현부

```
int Triangle::area() {
    return width * height/2;
}
```

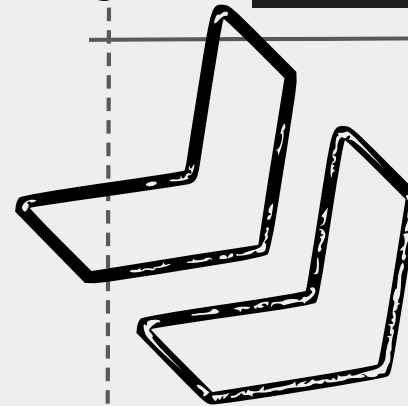
함수의 리턴 타입

클래스 이름

범위지정
연산자

멤버 함수명, 매개
변수

+



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

class Triangle {
public:
    int width;
    int height;
    //멤버 변수

    int area(); // 멤버 함수
};

int Triangle::area() {
    return width * height/2;
}
```

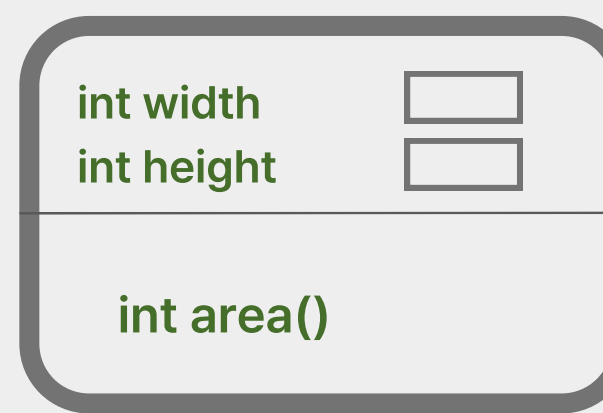
객체 생성

```
#include <iostream>
using namespace std;
class Triangle {
public:
    int width;
    int height;
    //멤버 변수

    int area(); // 멤버 함수
};

int Triangle::area() {
    return width * height/2;
}
```

클래스



첫 번째 삼각형의 넓이는 14
두 번째 삼각형의 넓이는 68

14

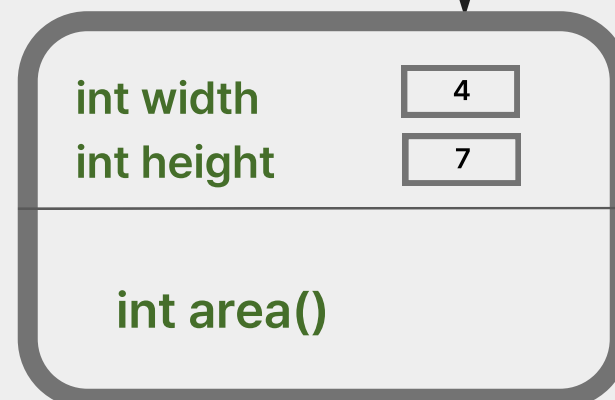
```
int main() {
    Triangle tri1;
    tri1.width = 4;
    tri1.height = 7;
    cout << "첫 번째 삼각형의 넓이는 " << tri1.area() << endl;
    Triangle tri2;
    tri2.width = 8;
    tri2.height = 17;
    cout << "두 번째 삼각형의 넓이는 " << tri2.area() << endl;
}
```

객체 tri1 생성

멤버변수
접근

객체 tri2
생성

멤버 함수 호출



객체 멤버 접근

- 객체 이름.멤버 변수
- 객체 이름.멤버 함수

02

생성자

1. 생성자란?

2. 기본 생성자

3. 위임 생성자

생성자

클래스 객체 생성 시, 객체의 멤버 변수를 초기화하는 함수

특징

- 객체 생성 시, 오직 한번만 자동 호출되는 멤버 함수
- >> 각 객체마다 생성자 실행
- 반드시 클래스 이름과 동일
- 리턴 타입을 선언하지 않음(void도 안됨)
- 한 클래스 내에서 중복 가능, 중복된 생성자 중 하나만 실행
- 생성자 선언이 없을 시, 기본 생성자가 자동 생성됨

기본 생성자

클래스에 생성자가 하나도 선언되지 않은 경우, 컴파일러가 대신 삽입해주는 생성자

= 디폴트 생성자

- 매개 변수가 없는 생성자

```
class Triangle {
public:
    int width;
    int height;
    int area();
};

int main() {
    Triangle tri1;
}
```

정상 컴파일

생성자 선언 X인
Triangle 클래스

```
class Triangle {
public:
    int width;
    int height;
    int area();
};

Triangle();

Triangle::Triangle() {
}

int main() {
    Triangle tri1;
}
```

컴파일러에 의해 자동 삽입

기본 생성자 호출

컴파일러에 의해 기본 생성자
자동 삽입

```
class Triangle {
public:
    int width;
    int height;
    int area();
    Triangle(int w, int h);
};
```

생성자의 매개 변수 부분 형식을 맞춰
줘야 인식함

Triangle 클래스
에 생성자가 선언
돼있어서 컴파일
러는 기본 생성자
자동 생성 X

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

호출

```
int main() {
    Triangle tri1;
    Triangle triangle(6, 7);
}
```

기본 생성자
없어서 오류남

abc E0291

"Triangle" 클래스의 기본 생성자
가 없습니다.

기본 생성자 자동 생성 예외 경우

- 생성자가 하나라도 선언된 클래스의 경우
<< 컴파일러는 기본 생성자를 자동 생성 X

위임 생성자

타겟 생성자를 호출하는 생성자
객체 초기화를 타겟 생성자에 위임

타겟 생성자

- 객체 초기화를 전담

특징

- 여러 생성자에 중복 작성된 코드 간소화

예시

```
Triangle::Triangle() {
    width = 6;
    height = 7;
}
Triangle::Triangle(int w, int h) {
    width = w;
    height = h;
}
```



위임
생성자

```
Triangle::Triangle() :Triangle(6, 7) {}
```

⇒ Triangle(int w, int h) 생성자 호출

w, h에 6, 7 전달

타겟
생성자

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

03

소멸자

1. 소멸자란?

2. 생성, 소멸의 순서

소멸자

객체 소멸시, 자동으로 호출되는 함수

특징

- 오직 한번만 호출됨
- <<임의로 호출 불가
- 객체 메모리 소멸 직전에 호출
- 객체 소멸 시, 마무리 작업이 목적
- 소멸자 함수 이름은 클래스 이름 앞에 ~를 붙임
- 리턴 타입 x, 리턴 값 x
- 변수 사용 불가
- 한 클래스 내에 오직 한개만 작성 가능
- 객체는 생성의 반대 순서로 소멸함
- 소멸자 선언이 없을 시, 기본 소멸자 코드 생성
- <<단순 리턴

예시

```
class Triangle {
public:
    int width;
    int height;
    int area();
    Triangle();
    ~Triangle();
    Triangle(int w, int h);
};
```

밑변1, 높이1 각각 4,9삼각형 생성
 밑변2, 높이2 각각 6,7삼각형 생성
 밑변, 높이 각각 6,7삼각형 소멸
 밑변, 높이 각각 4,9삼각형 소멸

```
int main() {
    Triangle tri1;
    Triangle triangle(6, 7);
}
```

```
Triangle::Triangle() {
    width = 4;
    height = 9;
    cout << "밑변1, 높이1 각각 " << width << "," << height << "삼각형 생성" << endl;
}

Triangle::Triangle(int w, int h) {
    width = w;
    height = h;
    cout << "밑변2, 높이2 각각 " << width << "," << height << "삼각형 생성" << endl;
}

Triangle::~~Triangle() {
    cout << "밑변, 높이 각각 " << width << "," << height << "삼각형 소멸" << endl;
}
```

생성자/소멸자 순서

지역 객체: 함수 내에서 선언된 객체, 함수 종료시 소멸

전역 객체: 함수 바깥에서 선언된 객체, 프로그램 종료시 소멸

객체 생성 순서

- 지역객체: 함수 호출 순간 순서대로
- 전역객체: 프로그램에 선언된 순서대로

객체 소멸 순서

- 지역객체, 전역객체: 생성된 순서의 역순으로

1. 전역 객체 생성
2. main 함수 객체 생성
3. 일반 함수 객체 생성
4. 일반함수 객체 소멸
5. main 함수 객체 소멸
6. 전역 객체 소멸

접근 지정자

1. 접근 지정자란?

2. 접근 지정자의 특징

접근 지정자

클래스 멤버 변수나 함수의 내부/외부에서의 사용을 제한

특징

- 객체 보호, 보안 목적
- 객체의 캡슐화 전략
- 멤버 변수에 대한 보호
- 주요 멤버는 외부 클래스나 객체에서 접근하지 못하게 보호
- 일부 멤버는 외부 접근 허용(외부와 인터페이스)
- 중복 접근 지정 가능
- 3가지 접근 지정자
- public
- protected(디폴트 접근 지정)
- private

public

다른 모든 클래스 접근을 허용

protected

클래스 자신과 상속받은 자식 클래스에만 허용

private

동일한 클래스의 멤버 함수에서만 접근 가능
+구현부 안에서 접근 가능

```
class AccessSpecifier {
private:
    //private 멤버 선언
public:
    //public 멤버 선언
protected:
    //protected 멤버 선언
};
```

헤더파일과 cpp파일

1. 헤더파일과 cpp파일

2. 헤더파일과 cpp파일
분리

3. 헤더파일 중복과
include 문제

C++ 프로그램 작성

목적: 클래스 재사용

- 클래스를 헤더 파일과 cpp 파일로 분리해서 작성
 - 클래스 선언부: 헤더파일(.h)에 저장
 - 클래스 구현부: cpp 파일에 저장
- ⇒헤더파일(클래스 선언부 포함)을 include해서 사용
- main() 함수: 전역함수/변수는 다른 cpp 파일에 분산 저장(필요 시, 헤더파일(클래스 선언부 포함) include)

전체 코드.cpp

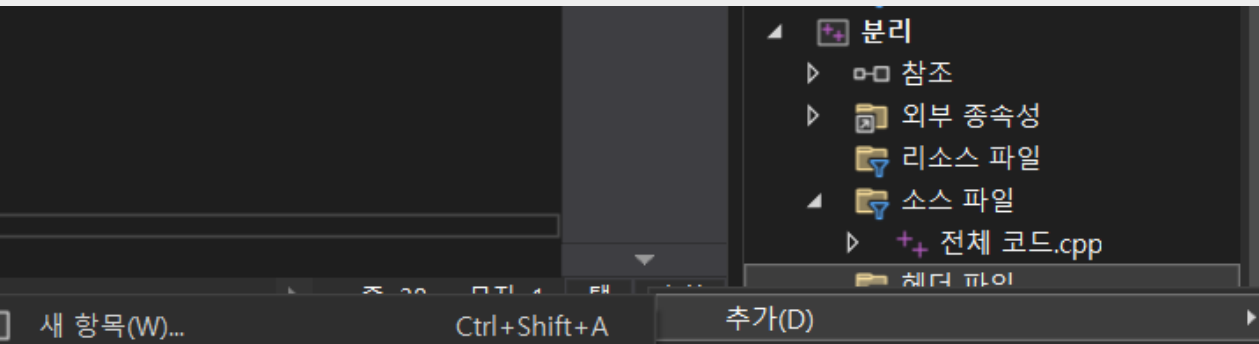
분리

(전역 범위)

```
#include <iostream>
using namespace std;
class Triangle {
private:
    int width;
    int height;
public:
    int Area() {
        return width * height / 2;
    };
    void setWH(int w, int h) {
        width = w;
        height = h;
    };

    Triangle() {
        width = 13;
        height = 8;
    };
    Triangle(int w, int h) {
        width = w;
        height = h;
    };
};

int main() {
    Triangle tri1;
    int area = tri1.Area();
    cout << "첫번째 삼각형의 넓이는 " << area << endl;
    Triangle tri2;
    tri2.setWH(8, 17);
    area = tri2.Area();
    cout << "두번째 삼각형의 넓이는 " << area << endl;
}
```



```
Triangle.h  X  전체 코드.cpp
+ 분리 (전역 분)
#include Triangle_H
#define Triangle_H

class Triangle{
public:
    int width;
    int height;
    int Area();
    void setWH(int w, int h);

    Triangle();
    Triangle(int w, int h);
};

#endif Triangle_H
```

```
#ifndef CIRCLE_H
#define CIRCLE_H

...

// 클래스 선언

...

#endif
```

헤더파일과 cpp파일 분리

main.cpp

triangle.cpp

Triangle.h

++ 분리

(전역 범위)

```

#include <iostream>
using namespace std;
#include "Triangle.h"

int main() {
    Triangle tri1;
    int area = tri1.Area();
    cout << "첫번째 삼각형의 넓이는 " << area << endl;
    Triangle tri2;
    tri2.setWH(8, 17);
    area = tri2.Area();
    cout << "두번째 삼각형의 넓이는 " << area << endl;
}

```

Triangle.h

main.cpp

triangle.cpp

Triangle.h

main

++ 분리

(전역 범위)

```

#ifndef Triangle_H
#define Triangle_H

class Triangle{
public:
    int width;
    int height;
    int Area();
    void setWH(int w, int h);

    Triangle();
    Triangle(int w, int h);
};

#endif Triangle_H

```

triangle.cpp

Triangle.h

main

++ 분리

(전역 범위)

```

#include <iostream>
using namespace std;
#include "Triangle.h"

Triangle::Triangle() {
    width = 13;
    height = 8;
};

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

int Triangle::Area() {
    return width * height / 2;
};

void Triangle::setWH(int w, int h) {
    width = w;
    height = h;
};

```

main.cpp

>>프로그램 동작

```

첫 번째 삼각형의 넓이는 52
두 번째 삼각형의 넓이는 68

```

Triangle.h

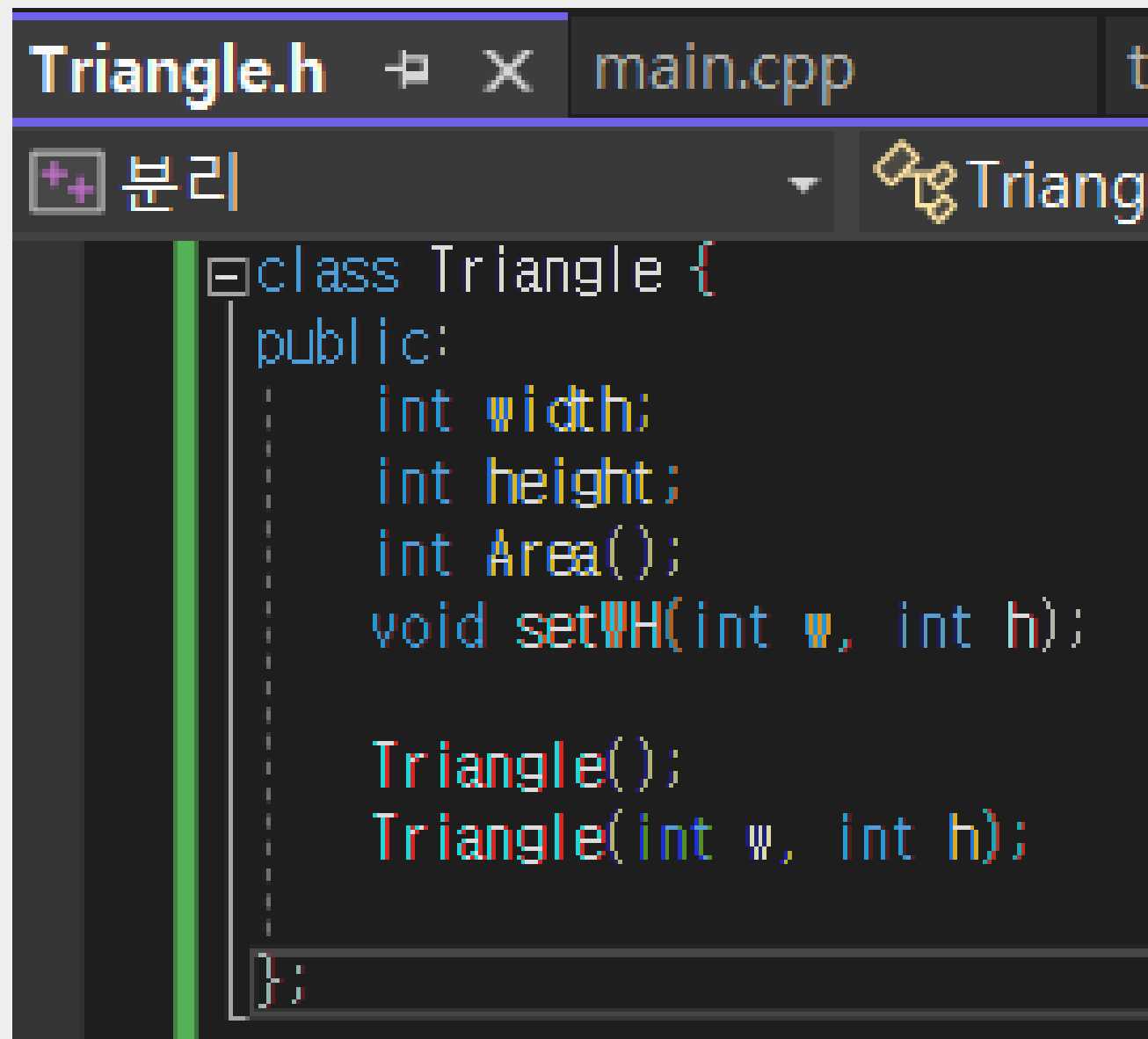
>>헤더파일(클래스 선언부)

triangle.cpp

>>클래스 구현부

헤더파일 중복과 include 문제

헤더 파일 중복 include 시, 발생하는 문제

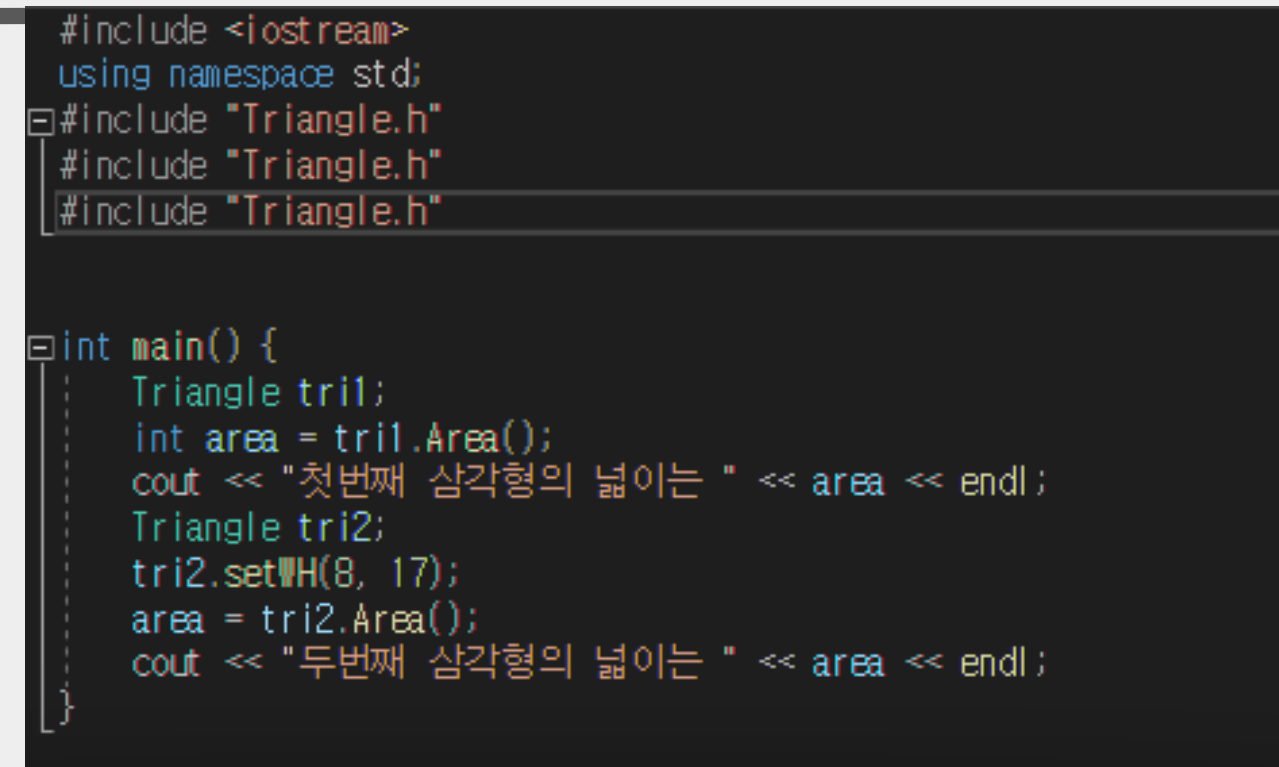


```

Triangle.h
main.cpp

class Triangle {
public:
    int width;
    int height;
    int Area();
    void setWH(int w, int h);

    Triangle();
    Triangle(int w, int h);
};
  
```



```

#include <iostream>
using namespace std;
#include "Triangle.h"
#include "Triangle.h"
#include "Triangle.h"

int main() {
    Triangle tri1;
    int area = tri1.Area();
    cout << "첫번째 삼각형의 넓이는 " << area << endl;
    Triangle tri2;
    tri2.setWH(8, 17);
    area = tri2.Area();
    cout << "두번째 삼각형의 넓이는 " << area << endl;
}
  
```

sual Studio

드 오류가 발생했습니다. 계속하고 마지막으로 성공한 빌드를 실행하겠습니까?

예(Y)

아니요(N)

상자를 다시 표시 안 함(D)

중복 오류 발생

```
Triangle.h  X 전체 코드.cpp
++ 분리 (전역 범위)
#include Triangle_H
#define Triangle_H

class Triangle{
public:
    int width;
    int height;
    int Area();
    void setWH(int w, int h);

    Triangle();
    Triangle(int w, int h);
};

#endif Triangle_H
```

```
Triangle.h  main.cpp  X triangle.cpp
++ 분리 (전역 범위)
#include <iostream>
using namespace std;
#include "Triangle.h"
#include "Triangle.h"
#include "Triangle.h"

int main() {
    Triangle tri1;
    int area = tri1.Area();
    cout << "첫번째 삼각형의 넓이는 " << area << endl;
    Triangle tri2;
    tri2.setWH(8, 17);
    area = tri2.Area();
    cout << "두번째 삼각형의 넓이는 " << area << endl;
}
```

```
Microsoft Visual Studio 디버그
첫 번째 삼각형의 넓이는 52
두 번째 삼각형의 넓이는 68
```

#ifndef CIRCLE_H

#define CIRCLE_H

...

// 클래스 선언

...

#endif

>> 조건 컴파일문을
사용해서 헤더의
중복사용도 문제 x

과제

1. 문서화
2. 원 계산기 만들기

1. 문서화

- 클래스 생성, 객체 생성, 생성자, 소멸자, 접근 지정자 내용 정리(실습은 선택)
- 헤더파일 분리 실습 진행 후 간단하게 진행해서 문서화

2. 원 계산기 만들기

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 초기 화면

선택 : 1

반지름 입력 : 6

평면 원 계산기 접속

평면원의 넓이 : 113.04

평면 원 계산기 종료...

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 2 원의 넓이, 둘레 구하는
계산은 같은

반지름 입력 : 7 [평면 원 클래스]에
넣어서 제작

평면 원 계산기 접속

원의 둘레 : 43.96

평면 원 계산기 종료...

선택 : 3

반지름 입력 : 6

높이 입력 : 7

원기둥 계산기 접속

원기둥의 부피 : 791.28

원기둥 계산기 종료...

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 4

반지름 입력 : 12

높이 입력 : 5

원기둥 계산기 접속

원기둥의 겉넓이 : 1281.12

원기둥 계산기 종료...

마찬가지로 원기둥의
겉넓이, 부피 구하는
계산은 같은
[원기둥 클래스]에
넣어서 제작

2. 원 계산기 만들기

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 5

반지름 입력 : 4 원뿔은 부피밖에 없기
높이 입력 : 6 때문에 [원뿔 부피 클래스]
 하나만 따로 만들어서 제작

원별 부피 계산기 접속

인플레이션율 : 100.48

원불 부피 계산기 종료...

선택 : 6

반지름 입력: 3

구 계 산 기 접 속

구의 부피: 113.04

구 계산기 종료...

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 7

반지름 입력 : 7

구 계 산 기 접 속

구의 겹넓이 : 615.44

구 계 산 기 종 료 . . .

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 종료

선택 : 9

1~8이 아닌 수를 입력하면 잘못됐다는 문장 출력

잘못된 숫자입니다.

8번 입력시 종료

원하는 계산을 선택하세요

1. 원의 넓이 계산
2. 원의 둘레 계산
3. 원기둥의 부피 계산
4. 원기둥의 겉넓이 계산
5. 원뿔의 부피 계산
6. 구의 부피 계산
7. 구의 겉넓이 계산
8. 총료

선택 : 8

프로그램을 종료합니다.

디버깅이 중지될 때 콘솔을 자동으로 닫으
이 창을 닫으려면 아무 키나 누르세요...

2. 원 계산기 만들기

평면 원 계산기 접속

원의 둘레: 43.96

평면 원 계산기 종료...

구 계산기 접속

구의 부피: 113.04

구 계산기 종료...

원기둥 계산기 접속

원기둥의 겉넓이: 1281.12

원기둥 계산기 종료...

'00계산기 접속', '00계산기
종료' 메시지 생성자, 소멸자
사용해서 출력

원뿔 부피 계산기 접속

원뿔의 부피: 100.48

원뿔 부피 계산기 종료...

2. 원 계산기 만들기

```
// 원 계산기를 위한 클래스
class [redacted] {
private:
    [redacted]

public:
    // 생성자
    [redacted] : [redacted] {
        cout << "***평면 원 계산기 접속***" << endl<<endl;
    }

    // 소멸자
    [redacted] {
        cout << "평면 원 계산기 종료..." << endl<<endl;
    }

    // 넓이
    double [redacted] {
        [redacted]
    }

    // 원의 둘레 계산기
    double [redacted] {
        [redacted]
    }
};
```

힌트 예시: 평면 원 클래스

THANK YOU

31기 육은서, 이시온, 황선영
질문은 카톡으로 보내주세요~