**객체지향프로그래밍 : 7차시**

중요 문단은 \*표시 남김.

**함수 중복**

동일한 함수의 매개 변수의 타입과 개수를 다르게 하여 다형성을 만드는 함수 사용법.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

대충 이런거.

단, 리턴 타입만 다르고 매개변수는 같을 경우 구분할 수 없기에 불가능함.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

함수 중복을 사용하면 함수의 이름을 개별적으로 암기하거나 호출을 잘못하는 실수를 줄일 수 있고, 한 개의 함수를 여러 매개변수로 이용할 수 있기에 다형성이 증폭됨.

**생성자 함수 중복**

생성자도 이전에 배웠듯이 매개 변수로 다양한 초기값을 전달할 수 있음.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**소멸자 함수 중복**

소멸자는 매개변수를 가질 수 없기에, 중복 함수로 사용할 수 없음.

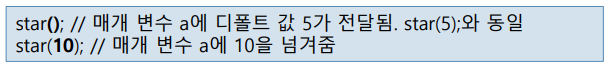
소멸자는 메모리 공간을 반환하는 역할을 하기에,   
한 클래스 내에 반드시 하나만 존재할 수 있기 때문임.

**디폴트 매개 변수**

매개 변수에 값이 입력되지 않았을 때, 할당될 값.



디폴트 매개 변수를 가진 함수의 매개변수를 사용할 때는, 해당 매개변수의 값을 입력하지 않더라도 사용할 수 있음. 때문에, 함수 중복을 간소화해 줄 수 있음.



단, 디폴트 매개 변수는 보통 매개 변수 앞에 선언될 수 없음.  
디폴트 매개 변수는 값을 입력하지 않아도 사용 가능한 매개변수이기에, 후열에 반드시 값을 입력해야 하는 보통 매개 변수가 존재한다면, 의미가 없어짐.

C++는 Function( , ,3)과 같은 형식을 지원하지 않기에, 충돌이 발생함.

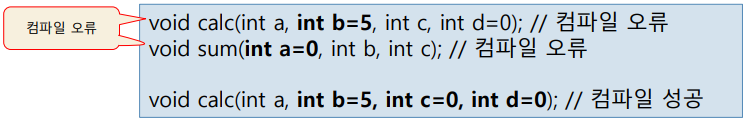
중간에 비어있는 값이 발생할 경우, 어느 매개변수를 생략하고, 어느 매개변수에 값을 넣어야 하는지 알 수 없다.

void addNum(int a = 10, int b);

의 경우,

addNum(100)

라고 입력하였을 때,   
a의 디폴트 매개변수를 사용한 채 b에 100을 넣겠다는 건지,   
a에 100을 넣겠다는 건지 알 수 없기 때문이다.



+ 컴파일러는 이러한 함수의 호출 방식을 정의될 때 미리 결정해 둡니다. 하지만 addNum(100)과 같은 경우 컴파일러가 해석하기에 모호하기에 불가능합니다. (그냥 맨 앞 매개변수로 보면 될 것 같지만, 안됨.) 그냥 단순히 말하자면, **C++ 설계자가, 헷갈리지 말라고 에러로 막아둔 것.**

**함수 중복의 모호성**

1. **형 변환으로 인한 모호성**

만약, float과 double 반환값을 중복하여 가지는 중복 함수가 존재한다면, int 타입을 매개변수로 입력했을 때 정수를 float으로 형변환할지, double로 형변환할지 모호하기에 에러가 발생함.

텍스트, 폰트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

1. **참조 매개 변수로 인한 모호성**

Int, int&의 매개변수를 동시에 가지는 중복 함수는 매개변수가 입력될 경우 값 기반인지 참조 기반인지 모호하기에 에러가 발생함.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

1. **디폴트 매개 변수로 인한 모호성**

하나의 보통 매개 변수를 가지는 중복함수와, 같은 보통 매개 변수와 하나의 디폴트 매개 변수를 가지는 중복함수가 존재할 경우, 둘 중 어떤 함수를 사용하려 하는지 모호해 에러가 발생함.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

아까 앞서 말한 디폴트 매개변수를 앞에 두는 경우도 모호하기에 에러가 발생함.

**Remind** : C++는 Function( , ,3)과 같은 형식을 지원하지 않기에, 충돌이 발생함.

중간에 비어있는 값이 발생할 경우, 어느 매개변수를 생략하고, 어느 매개변수에 값을 넣어야 하는지 알 수 없다.

void addNum(int a = 10, int b);

의 경우,

addNum(100)

라고 입력하였을 때,   
a의 디폴트 매개변수를 사용한 채 b에 100을 넣겠다는 건지,   
a에 100을 넣겠다는 건지 알 수 없기 때문이다.

**static과 non-static**

**인간의 얼굴, 텍스트, 만화 영화, 사람이(가) 표시된 사진

자동 생성된 설명**

위 예시는, “사람”이라는 클래스의 객체들은 모두 “눈”이라는 static 멤버를 가지고 있다는 말이다.  
눈은 여기서 static 멤버로 비유될 수 있다.

static : 프로그램이 시작될 때 생성되어, 종료 시 소멸하는, **클래스 당 하나만 존재할 수 있는 클래스 공유 멤버** (이게 진짜 다다.)

non-ststic : 스태틱 아닌거

**static의 활용**

전역 변수나 전역 함수를 클래스에 캡슐화하여 사용함.

싱글톤 같은 디자인 패턴을 알고 있다면, 싱글톤이 static 메서드의 대표적인 활용법이다.

**public class Singleton { // 싱글톤 예제**

**private static Singleton instance;**

**private Singleton() {**

**// 프라이빗 생성자**

**}**

**public static Singleton getInstance() {**

**if (instance == null) {**

**instance = new Singleton();**

**}**

**return instance;**

**}**

**}**