[KING] C++ 공부하기

생성자	날돌이
생성 일시	@2025년 4월 9일 오후 1:08
카테고리	개발스터디
최종 편집자:	날돌이
최종 업데이트 시간	@2025년 4월 10일 오후 5:36

객체와 클래스, 기본구조

클래스는 객체를 만드는 틀과 같고, 객체는 클래스를 통해 생성된 결과물에 비유할 수 있다.

→ 객체는 클래스를 통해 실제로 메모리에 만들어진 것

클래스를 정의할 때

- 클래스 이름과 그 안에 들어가는 멤버(데이터나 함수 등)
- 필수는 아니지만 멤버를 어떻게 다룰 것인지에 따라 접근 지정자가 포함되기도
- **접근 지정자**
  - **Private →** 같은 클래스 안의 다른 멤버만 접근 가능, 객체 데이터 보호를 위해 사용
  - **Protected** → 같은 클래스 안의 다른 멤버 또는 파생 클래스의 멤버들 접근 가능
  - **Public** → 객체가 visible한 범위라면 어디서든 접근 가능
- 다른 접근 지정자보다 앞서서 그냥 들어있는 멤버들은 기본적으로 Private로 취급
- 변수 선언과 클래스/객체 관계는 유사하고, 그래서 클래스를 새로운 Type으로 봐도 됨 (int + num 이렇게 정수형 변수를 선언하듯 class_name + object_name)
- **정의 키워드 정리**

키워드	공통점	차이점	기본 접근 제한자
class	멤버 변수, 멤버 함수 가짐	일반 클래스 정의 시 사용	private

struct	class 와 문법 동일	기본이 public	public
union	멤버 변수, 멤버 함수 가짐	메모리 공유 (하나만 저장)	public

클래스를 활용할 때

- 클래스의 public 멤버는 객체 이름 뒤에 .을 찍고 멤버 이름을 쓰면 일반 변수나 함수처럼 사용할 수 있다. 이 방식은 구조체의 멤버에 접근하는 방식과 완전히 동일하다.
- **스코프 연산자 (:)**
  - 클래스 외부에서 멤버 함수를 정의할 때 사용 (먼저 선언만 하고 외부에서 정의하는 경우)
- 같은 클래스로부터 생성된 객체여도 각각 변수는 따로 가짐
- 클래스 멤버인 함수는 변수 전달하지 않아도 자동으로 .뒤에 호출한 객체의 변수를 활용

생성자(Constructor)

- 객체가 생성될 때 자동으로 호출되는 특별한 멤버 함수로, 멤버 변수 초기화를 위해 사용
- 생성자는 클래스 이름과 동일하게 정의하며, 리턴값 없음
- 객체를 생성할 때 괄호 안에 값을 넣으면 그 값들이 생성자의 매개변수로 전달되어 멤버 변수가 자동 초기화됨
- 생성자는 객체를 만들 때 딱 1번만 자동 호출, 일반 함수처럼 직접 호출 불가능
- **생성자 오버로딩 (Constructor Overloading)**
  - **오버로딩**
    - 같은 이름의 함수를 매개변수를 다르게 해서 여러 개 정의하는 것
    - 다양한 전달 인자가 있을 때 알맞은 매개변수 타입의 함수를 호출하기 위해 사용
  - 생성자도 매개변수 개수나 타입에 따라 여러 개 정의 가능
  - 객체 생성 시 전달한 값에 맞는 생성자가 자동 호출됨
  - 기본 생성자를 호출할 때는 괄호를 사용하지X
- **다양한 생성자 호출 방법**

방식	예시	특징	비고

함수형 호출	`Rectangle r(1, 2);`	전통적 방식	가장 많이 쓰임
복사 초기화	`Rectangle r = Rectangle(1, 2);`	매개변수 1개일 때 자연스러움	복사처럼 보임
중괄호 초기화	`Rectangle r{1, 2};`	최신 문법 / 가장 추천	에러 방지 효과 있음

- **생성자의 멤버변수 초기화 (리스트 이용)**
  - 생성자에서 멤버 변수를 초기화할 때, 생성자 본문 { } 안에서 대입하는 대신 콜론(:) 뒤에 초기화할 멤버 변수와 값을 나열하는 방식이 있음

구분	생성자 본문에서 대입	멤버 초기화 리스트
기본형 타입 변수(int, double 등)	가능	추천
클래스 타입 멤버	가능할 때도 있지만 비효율	거의 필수
const 멤버, 참조형 멤버	불가능	무조건 사용

객체와 포인터

- 클래스가 정의되면 그 클래스 이름은 새로운 데이터 타입이 되므로('클래스를 정의할 때' 참고) 따라서 해당 클래스 타입의 포인터도 선언 가능
- 선언 → class_name *Pointer_name;
- **(참고) 기본 포인터 표현**

표현	읽는 방법	설명
`*x`	포인터 x가 가리키는 객체	객체 자체
`&x`	객체 x의 주소	메모리 주소
`x.y`	객체 x의 멤버 y	일반 접근
`x→y`	포인터 x가 가리키는 객체의 멤버 y	포인터 접근 전용
`(*x).y`	포인터 x가 가리키는 객체의 멤버 y	`x→y` 와 동일
`x[n]`	포인터 x가 가리키는 n+1번째 객체	배열처럼 접근 가능