7주차 cpp

프렌드함수

- 논의한 점 : 프렌드 함수는 다른곳에서 선언된 함수를 내가 원하는 클래스에 친구로 가져온다고 생각하기로 했다. 약간 참조와 비슷한 것 같다는 생각을 했다.
- 클래스 외부, 다른 클래스의 멤버함수, 다른클래스의 모든 멤버함수.
- 전역변수를 내 클래스에 가져올때는 friend (반환값) (함수); 로 가져옴
- 다른클래스의 멤버함수를 내 클래스로 가져올때는 friend (클래스이름)::(함수); 로 가져옴
- 다른클래스에 있는 모든 멤버함수를 내 클래스로 가져올때는 friend (클래스이름); 로 가져옴

논의한점 : 프렌드 함수는 약간 클래스의 private 및 protected 멤버에 접근하도록 하는, 그 클래스 외부에 정의된 함수? 라고 생각하자,

논의한점 : 프렌드 함수는 뭔가 클래스 외부에서 선언된 함수가 클래스 내부의 멤버에 접근할 수 있다는 점에서 참조(reference)와 비슷하다고 생각한다.

연산자중복 <- 연산자 오버로딩이라고도 한다!

- 연산자 함수는 2가지 선언방식이 있다, 클래스의 멤버함수로 구현하는 방법, 외부함수로 구현한 다음 클래스에 프렌드 함수로 구현하는 방법
- 피 연산자 타입이 다른 새로운 연산 정의
- 연산자는 함수 형태로 구현 - 연산자 함수
- 반드시 클래스와 관계를 가짐
- 피연산자의 개수를 바꿀 수 없음
- 연산의 우선 순위 변경 안됨
- 모든 연산자가 중복 가능하지 않음
- (리턴타입) operator (연산자) (매개변수);
- 실제로는 a. +(b) 로 멤버함수를 부르는 .으로 처리
- +, ==, +=

논의한점 : 연산자 중복시에 우리가 알고있는 연산자의 기본 의미를 크게 벗어나지 않는 방식으로 사용하는 것이 좋겠다고 생각을 했다, 예를들어 + 연산자를 제곱의 연산으로 사용하지는 말자

<단항 연산자>

- 피연산자가 하나 뿐인 연산자
  연산자 중복 방식은 이항 연산자의 경우와 거의 유사함
- 단항 연산자 종류
  전위 연산자(prefix operator)
  !op, ~op, ++op, --op
  후위 연산자(postfix operator)
  op++, op--

논의한점 : 전위연산자는 변수 앞에 오는거라고 생각하고 후위연산자는 뒤에오는거라고 생각하도록 하자.

포인터 개념

- * 는 값을 가리킨다
- this 는 객체의 주소값을 가리킨다. 0x000000
- 5번은 Power 는 객체 this 는 주소값 이기때문에 error
- 3번 Power는 참조값, this는 주소값 error
- 2번은 나를 복사한 애를 넘겨주는 복사생성자 Power
- 1번은 나 자신을 전달해준다 Power&
  call by reference 는 나 자신을 참조한 걸 바꾸는거니까 둘 다 바뀐다.

클래스의 멤버 함수가 아닌 외부 함수
전역 함수
다른 클래스의 멤버 함수

friend 키워드로 클래스 내에 선언된 함수
클래스의 모든 멤버를 접근할 수 있는 권한 부여
프렌드 함수라고 부름

프렌드 선언의 필요성
클래스의 멤버로 선언하기에는 무리가 있고, 클래스의 모든 멤버를 자유롭게 접근
할 수 있는 일부 외부 함수 작성 할때

프렌즈 함수가 되는 3가지
전역함수, 다른 클래스 멤버 함수, 다른 클래스 전체

프렌드 이용 연산자 중복

- 원래 연산자 함수 모두 클래스의 멤버 함수로 작성, 연산자 함수는 클래스 바깥의
  외부 전역 함수로도 작성 가능
  →이 경우 연산자 함수를 클래스에서 friend로 취해 클래스의 멤버를 자유롭게 접근 가능
  +연산자를 외부 함수로 작성