**객체지향프로그래밍 :6차시**

중요 문단은 \*표시 남김.

**매개변수 (함수)**

1. 값에 의한 호출(call by value)

함수 호출 시, 매개변수가 스택에 생성됨.  
코드에서 연산된 값이 매개변수에 복사되어 리턴됨.

1. 주소에 의한 호출(call by adress)

함수 호출 시, 포인터 타입으로 매개변수가 스택에 생성됨.  
코드는 주소를 받게 되고, 주소 값에 연산된 값을 저장함.

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**값에 의한 호출 (함수)**

값에 의한 호출은, 함수를 호출하는 쪽(main)에서 객체(변수)의 값를 복사하여 전달합니다, 단, 원본 객체의 주소나 참조를 사용하는 것이 아니라, 객체의 값만 함수의 매개 변수로 복사하여 사용힙니다.

매개 변수 객체의 공간이 스택에 할당되어지고, 호출하는 쪽 객체의 값이 공간에 그대로 복사되어집니다.

복사된 값은 함수 내에서만 사용되는 임시적인 값이며, 원본 객체에는 영향을 미치지 않습니다.

매개 변수 객체의 생성자는 호출되지 않지만, 함수 종료 시 매개 변수 객체의 소멸자는 호출됩니다.

매개 변수 객체의 생성자가 호출되지 않는 이유는, 호출되는 순간의 실인자 객체 상태가 매개변수에 그대로 복사되어 사용되기 때문에, 함수 내에서 새롭게 객체를 생성하는 것이 아닌, 복사된 객체를 사용하기 때문입니다. (이미 생성된 객체이므로, 생성자가 호출되지 않음.)

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**주소에 의한 호출 (함수)**

주소에 의한 호출은, 함수 호출 시 객체의 주소만 전달합니다.

매개 변수는 사용할 객체 타입에 맞는 포인터 변수로 선언하여야 하며, 함수 호출 시 생성자와 소멸자가 실행되지 않고 사용하려는 객체의 주소 값에 그대로 접근해 연산합니다.

생성자와 소멸자가 실행되지 않는 이유는, 객체의 주소 값을 그대로 사용하여 값이 존재하는 주소에 접근 후 연산하므로, 객체가 생성되거나 소멸되지 않기 때문입니다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

**객체 치환**

동일한 클래스 타입의 객체끼리 가능한 방법.

객체의 모든 데이터가 비트 단위로 복사되어 객체에 치환됨.

Circle c1(5);

Circle c2(30);

c1 = c2; // c2 객체를 c1 객체에 비트 단위 복사. c1의 반지름 30됨

치환된 두 객체는 내용물만 같을 뿐 독립적인 공간을 유지함.

변수 할당과 비슷한, 단순 ‘복잡한 할당’

**참조**

참조 변수는, 일종의 ‘가리킴’, ‘변수의 또다른 이름’이다.

int a = 5;

int& ref = a; // ref는 a를 가리킴

int b = 10;

ref = b; // ref는 b를 가리키는 게 아니라, a의 값이 b의 값으로 바뀜 (a == 10)

포인터와 다른 점은, 포인터는 가리키는 대상을 수시로 변경할 수 있는 변수 같은 역할인 것과 반대로, 참조는 가리킨 대상을 변경할 수 없으며, 가리킨 대상에 바로 접근할 통로를 열어주는 요소과 같다는 것입니다.

또한, 참조는 메모리 주소를 직접 다루지 않기 때문에, 동적 메모리 할당 및 제거를 할 수 없습니다.  
참조는 이름만 생성되며, 참조 변수에 새로운 공간을 할당하지 않습니다.(포인터는 주소 값을 저장할 새로운 공간을 할당함.)

간단히 요약하여,

* **참조**는 한번 설정되면 변경할 수 없는 ‘변수의 또다른 이름’이고, 메모리 주소가 아닌 변수 그 자체를 다룹니다. 또한, 메모리 공간을 생성하지 않고, 가르키는 변수의 공간을 그대로 사용합니다.
* **포인터**는 가리키는 대상의 메모리 주소를 다루며, 언제든지 가르키는 대상을 변경할 수 있는 주소 변수입니다. 또한, 주소를 담을 메모리 공간을 생성합니다.

참조는 간단히 변수를 다룰 때 사용되며, 포인터는 주로 메모리 조작 및 관리를 위해 사용합니다.

+ 참조는 Null 을 가르킬 수 없지만, 포인터는 Null 포인터를 가르킬 수 있습니다.

int& ref = nullptr; // 불가능! 참조는 NULL을 가리킬 수 없음

int\* ptr = nullptr; // ptr은 아무 것도 가리키지 않음

사용은 위 같이 이루어집니다. (&붙임.)

**참조에 의한 호출 (함수)**

참조 매개 변수를 사용하는 함수. 참조 매개 변수의 이름만 생기고, 공간은 할당되지 않음.  
참조 매개 변수는 실인자 변수 공간을 공유함. -> 직접 실인자 변수를 조작하는 것과 같음.

**참조 리턴**

C의 함수는 반드시 값(기본 타입, 주소)만 리턴 가능했음. C++의 함수는 참조도 리턴이 가능함.

char c = 'a';

char& find() { // char 타입의 참조 리턴

return c; // 변수 c에 대한 참조 리턴

}

char a = find(); // a = 'a'가 됨

char &ref = find(); // ref는 c에 대한 참조

ref = 'M'; // c= 'M'

find() = 'b'; // c = 'b'가 됨

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**복사 생성자 : 얕은 복사와 깊은 복사**

**텍스트, 인간의 얼굴, 만화 영화, 사람이(가) 표시된 사진

자동 생성된 설명**

1. 얕은 복사

객체의 복사본을 만들 때, 객체의 멤버 변수를 1:1로 복사함. 즉, 표면적으로만 객체를 복사하는 방식.

객체에 포함된 멤버 변수가 동적 메모리에 할당된 경우, 얕은 복사는 그 메모리의 주소만 복사합니다. 따라서 복사된 객체와 원본 객체는 같은 동적 메모리를 공유하여, 하나를 수정하면 다른 객체에도 영향을 미칩니다.

**“동적 메모리를 공유하는 방식”**

1. 깊은 복사

객체의 복사본을 만들 때, 객체의 멤버 변수를 1:1로 복사하며 동적 메모리에 할당된 데이터도 별도의 메모리로 복사하는 방식. ‘완전한 복사’

별도의 메모리 공간에 복사되므로 원본 객체와 독립적으로 작동합니다. 따라서, 한 객체를 수정해도 다른 객체에 영향을 미치지 못합니다.

**“동적 메모리를 새롭게 할당하여 독립된 객체로 복사하는, 완전한 복사”**

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

**복사 생성자**

객체의 복사 생성 시 호출되는 특수한 생성자.

한 클래스에 오직 한 개의 클래스만 생성되며, 보통 생성자와 클래스 내에 동시 존재 가능.

클래스에 대한 참조 매개 변수를 가지는 생성자.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

**디폴트 복사 생성자**

복사 생성자가 생성되어 있지 않는 클래스는, 컴파일러가 자동으로 디폴트 복사 생성자를 삽입함.