2022.06.08 바이너리 파일 vs 메모리 데이터 속도

int 1000만개를 바이너리 파일에서 읽어오는 속도와 ranges::copy를 통해 기존 메모리에 있는 데이터를 복사하는 속도를 비교함.



결과는 메모리에서 copy하는 것이 훨씬 빨랐다. 물론 백그라운드 프로그램이나 캐시의 영향이 있을 수 있더라도 유의미하게 차이가 있는것 같다. 같은 객체를 여러개 만들때 표준이 되는 객체를 미리 만들어 놓고 그것을 깊은복사하는 것이 파일로 부터 새로 읽어서 만드는 것보다 효율적일 것이라 예상됨. (대신 오브젝트를 저장하기 위한 메모리가 추가로 들 수 밖에 없음)

2022.06.20 다형성 복습

Animal <- Dog <- Retriever 의 상속관계를 갖는 클래스가 있다.

Animal\* retriever = new Retriever();

retriever->Print();

를 수행했을때 Animal\* 이지만 오버라이딩된 Retriever::Print()함수가 수행된다.

2022.06.27 이동할당을 정의하지 않았는데 R-value Reference를 넘겨준다면?

Dog a(10);

Dog b;

b = move(a);

\*b.pNum = 5;

cout << "a의 num" << \*a.pNum << "\n";

cout << "b의 num" << \*b.pNum << "\n";

다음과 같이 수행 하였을 때 b = move(a); 에서 이동 할당 연산자가 정의되어 있지 않다면 복사 할당 연산자가 호출된다. 만약 복사 할당 연산자마저 정의되어 있지 않다면 default(얕은 복사)가 호출된다.

즉, move함수를 통해 우측값 래퍼런스를 넘겨 줄때 " 이동 할당 > 복사 할당 > 얕은 복사(default) " 의 우선순위를 가진다.

2022.08.06 상속관계 클래스에서의 virtual 함수와 스마트 포인터

//shared\_ptr<Scene> spScene(new ShopScene);

Scene\* pScene(new ShopScene);

pScene->Animate();

pScene->Animate2();

delete pScene;

기본 생성자의 경우 virtual을 붙이지 않고 자식 클래스에서 알아서 조상 클래스들의 생성자도 자동으로 호출되고, 소멸자의 경우 일반 포인터 사용시 virtual을 붙이지 않을 경우 자식 클래스의 소멸자까지는 호출되지 않아 가상함수임을 명시해야 하지만, 스마트 포인터의 경우 virtual을 붙이지 않아도 자식 클래스까지 소멸자가 호출이 되었다.

그래도 스마트 포인터를 사용하더라도 우리는 소멸자에 virtual을 명시해주기로 결정했다. 멤버 함수의 경우에는 virtual을 붙이지 않을 경우 자식 클래스의 멤버 함수가 호출이 되지 않고, 붙일 경우 소멸자와 다르게 부모 클래스의 멤버함수는 호출 되지 않고, 자식 클래스의 멤버 함수만 호출 된다. 둘 다 호출하고 싶다면 자식 클래스 멤버함수 내에서 부모 클래스 함수도 따로 호출해주어야 한다.