C++

가상 함수

사람이라는 클래스를 부모 클래스로 두고 선생님과 학생 코스를 상속으로 구현한다고 생각해보자. 이 경우, addCourse라는 학생과 부모를 위한 함수를 정의해야 한다고 생각해보자.

일반적으로 사람 클래스에서 addCourse 함수는 의미가 없다. 이 경우, virtual bool addCourse()= 0; 라는 식으로 정의하지 않고 선언만 해둔다.

이 경우, 각각의 상속받은 클래스에서 함수를 정의해주면 된다.

단, 이 경우 사람 클래스는 추상 클래스가 되어, 객체를 만들어 낼 수 없는 클래스가 된다.

가상 소멸자

Person \* p = new Student( );

delete p;

를 실행할 경우, p에대한 소멸자만 실행된다. 그러나 student에 배정된 메모리 또한 반환해야 하므로 상위 클래스인 Person의 소멸자를 virtual로 선언하고 Student 클래스 내부에서 ~Student( )를 정의해주면 된다.

연산자 오버로딩

* 객체 간 연산에 활용할 수 있는 기능.
* ex) BookShelf BookShelf::operator+(BookShelf &other){ … }

포인터와 참조

* 포인터 변수의 크기는 아키텍처에 따라 다르다. 32bit는 32bit, 64bit는 64bit다.
* 포인터는 주소를 가리키므로 같은 변수를 여러 포인터로 받을 수 있다.
* 참조는 별도의 메모리를 갖지 않는다. 기존에 존재하는 객체에 붙는 또 다른 이름이기에

null이 될 수 없으며 다른 메모리에 재할당될 수 없다.

템플릿

* 하나의 클래스를 서로 다른 여러 타입에 재사용할 수 있도록 하는 방법

해시 테이블과 STL map 각각의 장단점

* 트리가 아닌 해시 자료구조를 이용한, 검색이 빠른 장점이 있다.
* STL map은 정렬된 상태로 자료를 저장, 범위 검색에 유용하다
* 해시 테이블을 검색속도가 빠르다

깊은 복사와 얕은 복사의 차이

* 얕은 복사란 값을 복사하는 것이 아닌 참조하는 것. 참조하는 객체가 사라지면 에러
* 깊은 복사는 메모리를 할당해서 값을 복사하는 것. 디폴트 복사 생성자를 오버로딩 해주면 된다.

C언어의 volatile 의 중요성

쓰레드와 프로세스의 차이

* 프로세스 : 운영체제로부터 자원을 할당받는 작업의 단위
* 쓰레드 : 프로세스가 할당받은 자원을 이용하는 실행의 단위.

사용자수준 쓰레드 vs 커널수준 쓰레드

* 운영체제에서 관리하면 커널 수준 쓰레드

모니터와 세마포어

* 스레드와 공유 자원 사이의 상호작용을 제어해야 한다( 스레드 동기화 )
* 모니터 : 상호 배제 락. 한번에 한 스레드만 실행 가능

데드락

* 두개의 스레드가 서로가 필요로 하는 락을 가지고 있는 경우.

동기화

* Busy waiting : 스레드가 활성 상태이지만 실제로는 아무 일도 하지 않는 상태

쓰레드와 락

* 동기화와 락
* 동기화된 블록
* 교착상태와 교착상태 방지
* 문맥 전환에 소요되는 시간을 측정하려면?

오버로딩 vs 오버라이딩

* 오버로딩 : 이름이 같은 메서드가 인사의 수나 자료형이 다른 경우
* 오버라이딩 : 상위 클래스의 메서드를 재정의 하는 것

DB

정규화된 DB와 비정규 DB

* 정규화된 DB : 중복이 최소가 되도록 설계된 DB
* 비정규 DB : 읽기에 소요되는 시간 줄이는게 목적

데이터베이스 트랜잭션

* 트랜잭션이란 한 단위를 이루는 일련의 연관된 데이터베이스 조작을 의미

이 트랜재션에 실패하면 롤백(과거로 복원), 성공하면 반영(커밋)

* 특징 네가지(ACID)
  + 원자성 : 모든 작업이 성공적으로 처리되지 않으면 어떤 작업도 처리되지 않는다
  + 일관성 : 트랜잭션 전후로 참조 무결성은 깨져서 않된다.
  + 고립성 : 트랜잭션에서 DB를 변경한 내용은 트랜잭션이 커밋될 때까지 고립되어야 한다.
  + 영속성 : 커밋 후에는 변경된 내용이 영구적이어야 한다.