**No.1 typedef 와 #define의 차이점은 무엇인가?**

- typedef : 자료형의 재정의, 컴파일러가 인식 가능한 자료형

- #define : 코드의 치환, 컴파일 전에 수행되는 선행 처리기

“typedef는 자료형의 재정의이며 컴파일러가 인식이 가능한 자료형이고 #define은 코드의 치환이며 전처리기에서 수행됩니다.”

**No.2 Call by Value 와 Call by Reference은 무엇이고, 장단점은 무엇인가?**

- Value : 값을 직접 넘기기 때문에 원본이 변경될 가능성이 없지만, 메모리 공간을 사용한다.

- Reference : 메모리 공간 문제는 없지만 값이 변경될 가능성이 있다.

“Call by Value는 값을 복사하여 직접 넘기기 때문에 원본이 변경되지 않지만 메모리 공간을 사용하여 낭비가 있고, Call by Reference는 메모리를 아끼고 속도가 빠르지만 값이 변경될 위험성을 가지고 있습니다.”

**No.3 포인터와 레퍼런스의 차이점**

- 포인터는 널 값을 가질 수 있지만 참조자는 널 값을 가질 수 없다.

따라서 참조자는 대상이 반드시 존재할 때 사용하고 포인터는 널 검사를 할 때 사용한다.

- 참조자는 선언 시 반드시 초기화를 해야한다.

- 초기화 시, 참조자는 객체를 직접 받고 포인터는 주소를 받는다.

- 참조자는 가리킨 대상을 변경할 수 없지만 포인터는 언제든 변경 가능하다.

“포인터와 레퍼런스의 차이점은 주소를 담을 공간이 있고 없고 입니다. 포인터는 가리킬 대상이 없어도 NULL값을 담을 공간이 있고, 레퍼런스는 그렇지 않기에 선언과 동시에 초기화를 해야 합니다. 이런 특성 때문에 포인터는 가리킬 대상을 변경할 수 있지만 레퍼런스는 불가능합니다.”

**No.4 C언와 C++언어의 차이점이란?**

C : 절차지향적 언어(Procedural) / C++ : 객체지향 언어(Multi Paradigm)

C : 하향식 접근 방법(개괄->세분) / C++ : 상향식 접근 방법(작은 부분 개발 후 조합)

C++의 Class와 C의 Struct의 차이점 : 접근 지정자, 상속, 템플릿, 오버로딩, 레퍼런스 변수 유무

Decay(부식) : C++에서 배열이나 함수가 포인터로 변환되는 것(정보가 온전히 담겨있지 못한 상태)

“C와 C++의 차이는 하향식 접근 방법과 상향식 접근 방식에 있습니다. C는 전체적은 개요를 먼저 구성한 뒤에 세부적으로 작업하며, C++은 세부적인 것을 쌓아 올리면서 전체적인 구조를 짜 올립니다.”

**No.4-1 Class의 특징**

○ 캡슐화

- 관련있는 데이터와 함수를 하나의 단위로 묶는 것.

- 사용자는 내부 부분을 모르더라도 약속된 부분을 통하여 기능을 사용할 수 있다.

○ 정보 은닉

- 클래스 외부에서 알 필요가 없는 정보에 대해 접근을 제어하여 정보를 은닉하는 것.

- 접근 지정자를 통해 은닉하며, 객체의 안정성을 높인다.

○ 다형성

- 부품으로 사용하는 객체를 다른 객체로 교체 가능하게 하는 것.

- 부품 객체에서 약속된 인터페이스가 같다면 교체가 가능하다.(같은 함수로 다른 기능)

○ 상속성

- 기존의 클래스를 토대로 새로운 클래스를 만드는 것. 기능의 확장과 세분화 가능.

- 공통의 부모를 같은 계층을 만들어 객체의 집합에 다형성을 부여한다.

○ 재사용성

- 인터페이스만 동일하면 어디에서든 사용할 수 있게 만드는 것.

○ 추상화

- 추상적인 개념을 구체화 시키는 것(함수)

- 구체적으로 보여지지 않은 동작(처리)을 만들어내는 것

“첫째로 캡슐화가 있는데, 관련있는 데이터와 함수들을 하나의 단위로 묶는 것을 의미하고 둘째론 은닉성입니다. 접근 지정자를 통해 외부에서 알 필요 없는 정보에 대한 접근을 제한하여 안정성을 높입니다. 셋째는 상속성으로 기존 클래스를 토대로 새로운 클래스를 만들어 기능의 확장과 세분화를 가능하게 합니다. 그리고 마지막으로 다형성이 있는데 서로 다른 객체가 같은 메시지에 다른 기능을 하는 특성입니다.”

**No.5 프랜드 클래스(Friend Class)란?**

접근 지정자에 대한 예외처리가 가능한 클래스. 당하는 쪽에서 friend 선언.

friend 지정은 단방향이며 명시적으로 지정한 대상만 friend가 된다.

friend 지정은 전이되지 않으면 friend의 friend 관계는 인정하지 않는다.

복수의 대상에 대해 동시에 friend를 지정할 수 있지만 한 번에 하나씩만 가능하다.

friend 관계는 상속되지 않는다.

friend 클래스, friend 맴버함수, friend 전역함수로 사용할 수 있다.

“접근 지정자에 대한 예외처리가 가능한 클래스를 의미하는데 선언된 클래스는 무조건적으로 접근이 가능하도록 됩니다. 특이한 점은 당하는 쪽에서 friend를 선언해야 한다는 점입니다.”

**No.6 다중상속은 무엇이고, 단점은 무엇인가?**

한 클래스가 동시에 여러 클래스를 상속받는 구조.

다중 상속 관계에서 부모들의 같은 이름의 함수나 맴버변수를 사용할 경우 모호성을 유발한다.

가상 상속으로 해결 가능하나 객체의 크기도 크고 맴버 데이터 접근 비용도 크게 발생한다.

“한 클래스가 동시에 여러 클래스를 상속받는 구조입니다. 단점으로는 이러한 관계에서 동시에 상속한 부모 클래스들의 동일한 이름의 함수나 멤버변수를 사용할 경우 모호성을 유발하는 단점이 있습니다.”

**\*역질문 예상 – “단점을 극복할 수 있는가?”**

“가상 상속으로 해결이 가능하지만 객체의 크기도 크기고 멤버 데이터 접근 비용도 크게 발생하게 되는 우려가 있어서 애당초 가상 상속을 하지 않는 것이 가장 좋습니다.”

**No.7 랜더링 파이프라인에 대해 설명하시오.**

3D 장면을 2D화면으로 전환하기 위해 처리되는 일련의 과정

1. 로컬 스페이스 – 모델링 스페이스라고도 하며 물체를 구성하는 프리미티브의 좌표계.

2. 월드 스페이스 – 신을 구성하기 위해 모델을 이동, 회전, 크기에 따라 변환시킨 전역 좌표계.

3. 뷰 스페이스 – 카메라의 상대 위치로 기하물체를 옮기는 과정. 신을 찍기 위한 중간 좌표계.

4. 후면 추려내기 – 폴리곤의 와인딩 순서에 따라 렌더링하지 않을 대상을 제거하는 과정.

5. 조명 – 광원, 시선, 법선 등을 이용하여 조명 계산식을 통해 정점의 색상을 계산하는 과정.

ex) 테셀레이션 - GPU에서 정점을 자유자재로 증가, 감소시키는 과정.

6. 클리핑 – 시야 볼륨 위부의 기하물체를 추려내는 과정.

7. 투영 – 3D 장면을 2D로 표현하기 위한 과정. 원근 투영과 직교 투영이 있다.

8. 뷰포트 변환 – 투영된 삼각형들을 윈도우 좌표계로 변환하는 과정. z축 버퍼링 사용.

9. 래스터라이즈 – 정점 정보를 통해 픽셀 정보를 얻어내는 과정.

ex) 텍스처 적용 - UV좌표계를 이용하여 텍스쳐를 입히는 과정.

ex) 랜더 백엔드 – 알파 테스트, 스텐실 테스트, 깊이 테스트 등의 검사를 통해

알파 블랜딩, 포그, 안티앨리어싱 등의 작업을 수행하는 과정.

**No.7-1 랜더링 파이프라인과정에서 정점이 어떻게 변하는지 설명하시오.**

로컬 –> 월드 –> 뷰 –> 투영 –> 클립 –> 스크린

**No.8 HLSL(셰이더)에 대해서 설명하시오.**

다이렉트 3D API에서 사용되는 셰이딩 언어. 버텍스 셰이더, 지오메트리 셰이더, 픽셀 셰이더, 헐 셰이더, 도메인 셰이더 등의 프로그래머블 랜더링 파이프라인을 작성할 때 사용된다.

“Direct 3D API에서 사용되는 고급 셰이딩 언어로, 프로그래머블 렌더링 파이프 라인을 작성할 때 사용됩니다.”

**\*역질문 예상 – “써봤는가?”**

경험을 솔직히 말하는 것이 좋을 것 같다.

**No.9 컨텍스트 스위칭에 대해 설명하시오.**

멀티 프로세스 운영체제에서 Running중인 프로세스와 Ready중인 프로세스가 서로 상태전이를 통해 데이터가 스위칭 되는 작업을 컨텍스트 스위칭이라고 한다.

CPU에서 Running중인 프로세스를 작업하며 Ready상태중인 프로세스는 메모리에 백업된다. 스위칭 될 때는 Running중인 프로세스가 Ready상태로 전이되면서 메모리에 백업된다.

즉, 프로세스를 이것저것 우선순위에 따라 변경하기 위해서는 프로세스 데이터를 레지스터와 메모리 사이를 왔다갔다 하며 복사해야 한다. 이런 일련의 과정을 context switching(컨텍스트 스위칭)이라고 한다.

“Running중인 프로세스와 Ready중인 프로세스가 서로 상태 전이를 통해서 데이터가 스위칭 되는 작업을 말합니다. Running 중인 프로세스를 CPU에서 작업하고 Ready중인 프로세스는 메모리에 백업되어 이 두 데이터를 스위칭 합니다.”

**No.10 컨테이너의 종류와 설명을 하시오.**

**(컨테이너 3가지와 각컨테이너에 있는 STL에 대해 설명하시오)**

1. 시퀀스 컨테이너 : 자료 삽입 삭제에 규칙이 없는 컨테이너

ex) 백터, 리스트, 데크(deque)

2. 연관 컨테이너 : 자료 삽입 삭제에 규칙이 있는 컨테이너, 주로 균형 이진 트리 사용.

ex) 맵, 셋, 멀티맵, 멀티셋

3. 어댑터 컨테이너 : 시퀀스 컨테이너에 사용자 인터페이스를 추가한 컨테이너

ex) 스택, 큐, 우선순위 큐

“첫째로 시퀀스 컨테이너로, 자료의 삽입, 삭제에 규칙이 없는 컨테이너로 벡터, 리스트가 대표적입니다. 둘째론 연관 컨테이너입니다. 삽입과 삭제에 규칙이 있고 맵과 셋이 있습니다. 마지막으로 어댑터 컨테이너가 있는데, 이는 시퀀스 컨테이너에 인터페이스를 추가한 것으로 스택, 큐가 있습니다.”

**No.11 벡터와 리스트의 차이점을 설명하시오.**

**(메모리할당과 해제와 특성)**

* 벡터
  + 선형구조
  + 데이터 추가 시 카파시티를 넘어서면 오버헤드가 발생
  + 중간에 있는 데이터를 삭제 시 메모리의 복사현상 발생
  + 검색이 용이
* 리스트
  + 비선형구조
  + 데이터 추가 시 메모리 오버헤드 발생
  + 검색 시 순회를 해야 하므로 시간적 소모가 심하다.
  + 추가 및 삭제가 용이하다.

“둘의 차이는 메모리의 구조가 가장 큽니다. 벡터는 선형구조이고 리스트는 비선형구조이기 때문에 벡터는 검색은 빠르나 데이터 추가, 삭제가 느리고 리스트는 추가, 삭제는 용이하지만 검색에 있어서 느립니다.”

**No.12 스택프레임이란 무엇인가?**

* 메모리 영역 스택에 저장되는 함수 호출 정보를 스택 프레임이라고 하며 스택 프레임에는 함수로 전달되는 매개변수와 함수 실행을 마치면 돌아갈 복귀 주소 등의 정보가 들어간다.

“스택 메모리 영역에 저장되는 함수 호출 정보를 말합니다. 함수로 전달되는 매개변수, 함수 종료시 돌아갈 복귀 주소 등의 정보가 들어갑니다.”

**No.13 Transform matrix를 3 \* 3이 아닌 4 \* 4를 사용하는 이유는 무엇인가?**

* 동차 좌표계를 사용하여 3차원 정보를 4차원 자료형으로 저장하기 위함(동차 좌표계를 알 시)
* 3차원 상의 각 방향과 위치정보, 총 4가지 정보를 저장하기 위해(동차 좌표계를 모를 시)

“3차원 상의 각각의 위치정보와 방향, 총 4가지 정보를 저장하기 위해서 입니다. 이를 위해 동차 좌표계를 사용해 4차원 자료형으로 저장합니다.

**\*역질문 예상 : “동차좌표계가 뭔가?”**

**No.14 스케줄링이란 무엇인가?**

* 스케줄링은 다중 프로그래밍을 가능하게 하는 운영체제의 동작 기법.
* CPU 스케줄링은 CPU를 사용하려고 하는 프로세스들 사이의 우선순위를 관리하는 일.
* (완벽한 설명이 아니고 시스템 쪽으로 심화되는 문제이므로 완벽한 준비를 필요로 하는 사람은 더 깊은 조사가 필요함.)
* 프로세스의 CPU 할당 순서 및 방법을 결정짓는 일, 이때 사용되는 알고리즘을 스케줄링 알고리즘이라 한다.
* 스케줄링 알고리즘을 적용하여 실제로 프로세스를 관리하는 운영체제 요소(모듈)을 가리켜 스케줄러(Scheduler)라 한다.

“여러 프로세스를 하나의 CPU로 동시에 실행하는 경우, 프로세스의 CPU 할당 순서 및 방법을 결정하는 일을 말합니다.”

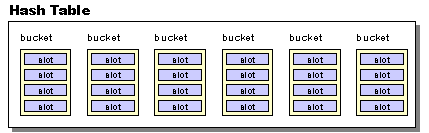
**No.15 해싱, 해시테이블, 해싱의 시간복잡도에 대해 설명해보시오.**

* 해싱
  + 하나의 문자열을 원래의 것을 상징하는 더 짧은 길이의 값이나 길이로 변경하는 것.
  + 짧은 해시 키를 사용하는 것은 원래 값을 이용하여 찾는 것 보다 빠르기 때문에 자료구조 내의 항목들을 색인 검색하는데 용이.
* 해시테이블
  + 데이터를 담을 테이블을 미리 할당한 후 입력받은 데이터를 해싱하여 테이블 내의 주소를 계산하고 이 주소에 데이터를 담는 것.
  + 공간을 팔아 시간을 얻는 것.
  + 해시 테이블은 해시함수를 사용하여 버킷이나 슬롯의 배열로 인덱스를 계산.
* 해싱의 시간복잡도
  + 검색은 O(1)이고 자료의 추가 삭제는 해싱 함수의 성능에 따라 다르다.

“해싱은 하나의 문자열을 본래의 것을 상징하는 더 짧은 길이의 값이나 길이로 변경하는 것을 의미합니다. 헤시 테이블은 데이터를 담을 테이블을 미리 할당한 후에 입력 받은 데이터를 해싱해 테이블 내의 주소를 계산하고 이 주소에 데이터를 담는 것입니다. 마지막으로 해싱의 시간복잡도는 빅오 표기법으로 검색은 O(1)이며 추가, 삭제에 있어선 해싱 함수의 성능에 따라 다릅니다.”

**No.16 버켓( bucket )이란 무엇일까?**

* 해시테이블의 구성요소로 해시테이블은 버킷들의 연속적인 주소를 갖는다.
* 각 버킷은 연결리스트나 트리로 구성되어있다.



“하나 이상의 슬롯으로 구성된 데이터들의 저장 장소로 연결리스트나 트리로 구성되어있습니다.”

**\*역질문 예상 : “슬롯은?”**

하나의 주소에 여러 데이터가 저장 가능하도록 버킷을 나누는 데이터 저장 단위.

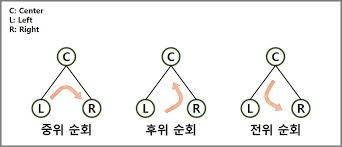
**No.17 해시와 트리의 장단점을 쓰시오.**

* 해시는 트리에 비해 메모리를 많이 사용하지만 검색 속도가 더 빠르다.
* 해시는 해싱함수에 따라 삽입에 시간이 걸리지만 트리는 해시에 비해 걸리는 시간이 적다.
* 해싱을 함으로써 키값이 원래의 정보를 다 담을 수 없기 때문에 충돌이나 클러스터 같은 문제들이 발생 할 수 있기 때문에 이것을 해결하는 작업이 필요한 반면 트리는 그럴 위험이 없다.

“둘의 차이는 검색 속도와 메모리 저장 방식에 있습니다. 해시는 트리에 보다 검색 속도가 빠르지만 메모리를 많이 사용하며 해싱 함수에 따라서 데이터 저장에 있어 충돌이 일어날 수 있고 트리는 삽입, 삭제가 수월하지만 검색이 있어선 좀 뒤쳐집니다.”

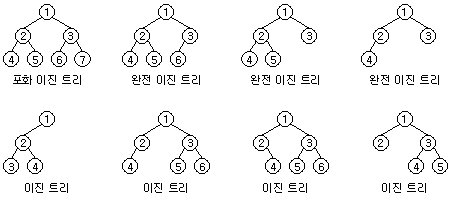
**No.17-1 트리 순회방식을 말해 보시오.**

* 전위, 중위, 후위가 존재하는데 루트 노드를 언제 순회 하느냐에 따라 나뉜다.
* (전위는 루트노드를 맨 처음 순회하고 중위는 루트노드를 중간에 순회하고 후위는 루트노드를 마지막에 순회한다.)



**No.18 레드블랙트리, 포화2진트리에 대해 설명하세요.**

* 레드블랙트리는 균형이진트리 중 하나로 노드를 검은색 노드와 빨간색 노드로 나눠서 정리하는데 루트노드부터 리프노드까지 모든 경로에 대한 검은색의 높이를 같게 하여 최대 검색시간을 보장하는 자료구조이다.
* 포화이진트리는 리프노드를 제외한 모든 노드가 두 개의 자식노드를 가지는 트리이다. 때문에 루트노드에서 리프노드까지의 거리가 모두 같다.



**No.19 벡터의 내적과 외적에 대해 설명해보세요.**

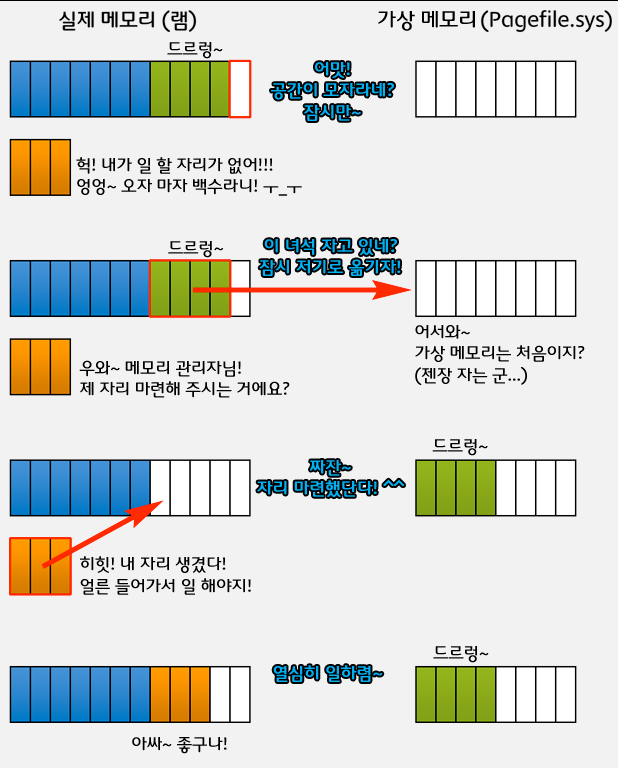
* 벡터의 내적은 스칼라 곱으로 결과가 두 벡터의 관계를 나타내는 스칼라 값으로 나오게 되고 외적은 벡터 곱으로 두 벡터에 동시에 직교하는 벡터가 결과로 나오게 된다.
* 내적은 교환법칙이 성립하지만 외적은 성립하지 않는다.

**No.20 virtual 키워드란? / 가상함수테이블(vtbl)이란? / 소멸자에 virtual붙이는 이유?**

* Virtual 키워드란
  + 함수를 가상함수로 만들어 주는 키워드로써 가상함수를 사용하게 되면 객체에 다형성을 부여하여 동적 바인딩을 사용하게 된다.
  + 컴파일러가 상속된 클래스보다 오리지널 클래스의 타입을 먼저 판단하여 함수를 호출하는 겁니다,,, 그래서 그걸 방지하기 위해서 상속된 클래스에서 원하는 함수가 호출될수 있도록 virtual 를 붙여줍니다,,,그러면 상속된 객체에서도 함수를 재정의하여 호출 할 수 있는거죠,
* 가상함수 테이블이란?
  + 다형성 객체에 한해 가지는 함수의 주소들을 담은 배열.
  + 각 객체는 vptr을 가지며 이 포인터는 가상함수테이블에서 객체마다 갖는 고유의 시작주소를 가리킨다.
* 소멸자에 virtual 붙이는 이유
  + 자식객체를 삭제 할 때 부모객체의 소멸자도 함께 호출하기 위해.

**No.21 가상메모리란 무엇인가?**

* 실제 물리적인 메모리 공간(RAM)이 아니라 하드디스크 파일 형태로 따로 준비 해 놓은 가상 메모리 공간으로 부족한 시스템 메모리를 보조 해 주는 역할.
* 가상메모리는 실제 메모리를 보조 해 주는 역할.
* 절대 실제메모리처럼 사용 해서도 안되고 착각을 하면 안된다.



**No.22 RTTI란 무엇인가?**

* Run-Time Type Information의 약자로써 가상함수테이블에 저장되는 객체의 실체 정보이다.
* RTTI는 TypeId 또는 DynamicCast 등에 사용되며 런타임 시에 실체를 판단하거나 안정적인 캐스팅에 사용된다.

**No.23 메모리 공간 구조에 대해 설명해보시오.**

* 코드
  + 프로그램 명령이 위치하는 곳으로 기계어로 제어되는 메모리 영역이다.
* 데이터
  + 전역변수, 정적변수, 구조체 등이 저장된다.
  + 초기화 된 데이터는 data영역에 저장되고 초기화되지 않은 데이터는 BSS(Block Stated Symbol) 영역에 저장된다.
* 스택
  + 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역.
  + 컴파일 타임에 크기가 결정된다.
* 힙
  + 프로그래머가 직접 제어하는 메모리 공간.
  + 런타임에 크기가 결정된다.

**No.24 바인딩, 정적바인딩, 동적바인딩 설명 해보시오.**

* 바인딩
  + 변수나 함수가 사용될 메모리 공간을 결정하는 것.
  + 함수의 경우
* 정적 바인딩
  + 실행 시간 전에 일어나고, 실행 중 변하지 않는 상태로 유지되는 바인딩.
* 동적 바인딩
  + 실행 시간 중에 일어나거나 프로그램 실행 과정에서 변경되는 바인딩.
* **오버로딩(Overloading) - 함수 중복 정의**
* 오버로딩은 같은 이름의 함수에 매개변수를 다르게 사용하여 매개 변수에 따라 다른 함수가 실행되는 것.
* **오버라이딩(Overriding) - 함수 재정의**오버라이딩은 상속받았을때 부모클래스의 함수를 사용하지 않고 다른 기능을 실행할때 함수를 자식클래스에 같은 이름, 매개변수로 재정의 해서 사용하는 것.

**No.25 struct 와 class의 차이점은 무엇인가?**

* C++에서는 Struct는 기본 접근 지정자가 public이고 class는 private이다.
* C에서는 Template, 오버라이딩 불가능.

**No.26 인터페이스(Interface), 상속(Inheritance), 다형성(Polymorphism)에 대해 설명하시오.**

* 인터페이스는 사용방법을 추상화 시켜놓은 추상클래스로써 이를 상속하여 구체화된 클래스를 만들 수 있고 그 클래스를 사용하여 같은 이름의 함수여도 여러가지 동작을 가능하게 하는 것이 다형성이다.

**No.27 타입캐스팅의 종류와 설명을 하시오.(6가지)**

* Static cast
  + 컴파일에서 허용하는 한도 내에서의 형 변환을 하는 캐스팅.
  + 형 변환에 대한 타입 체크를 run-time에 하지 않고, compile 타임에 정적으로 수행한다.
* Dynamic cast
  + 다형성을 가진 객체에 대해 안전한 형 변환만을 허용하는 캐스팅.
  + 실패 시 NULL반환.
* Const cast
  + 포인터의 상수성만 변경하고 싶을 때 사용.
* Reinterpret cast
  + 연관성이 없는 포인터 타입을 변환하기 위한 캐스팅.

**No.29 프로토콜이란?**

* 복수의 컴퓨터 사이나 중앙 컴퓨터와 단말기 사이에서 데이터 통신을 원활하게 하기 위해 필요한 통신 규약.

**No.30 인라인함수와 매크로함수의 차이점은 무엇인가?**

* 인라인 함수는 컴파일 단계에서 함수를 통째로 복사하여 함수호출을 하지 않고 함수의 기능을 수행하는 것으로 컴파일러가 오류를 체크하여 자료형에 종속적이며 디버깅이 가능하지만 매크로 함수는 기계적인 텍스트 치환으로 자료형에 독립적이며 전처리기에서 처리하므로 디버깅이 힘들다.

**No.31 가변인수함수란 무엇인가?**

* 매개변수의 타입과 개수에 종속적이지 않은 함수.
* Ex) printf()

**No.32 new와 malloc()의 차이점은 무엇인가?**

* new는 연산자 malloc은 함수
* new는 생성자를 호출하며 포인터를 반환하고 malloc은 객체크기만큼 메모리만 할당.

**No.33 const 란 무엇인가?**

* 상수화 키워드로 값의 변경이 불가능하다.

**No.33-1 const 포인터란 무엇인가?**

* 담은 주소를 변경할 수 없는 포인터.

**No.33-2 const 가 뒤에 붙는 함수?**

* Const 객체만이 호출할 수 있는 함수.(포인터 상수)

**No.33-3 const 객체란?**

* 객체를 포인터에 담을 때 객체를 보호하기 위해 const 키워드를 자료형 앞에 붙여 지정된 객체.

**No.34 static이란?**

* 정적 키워드로써 프로그램이 실행 시부터 종료 시까지 메모리 유지.
* 정적 변수는 변수가 선언된 파일이나 함수 내에서만 접근이 가능하다.(전역과의 차이)

**No.35 #define 과 const의 차이점은?**

* #define은 기계적 코드 치환이고 const는 변수의 상수화이다.

**No.36 템플릿은 무엇인가?**

* 다른 자료형에 대해 같은 동작을 수행하기 위해 만든 기능.

**No.37 선형구조와 비선형구조의 종류와 설명을 하시오.**

* 선형구조 : 배열
* 비선형구조 : 리스트, 트리

**No.38 벡터 카파시티보다 사이즈가 커질 때 재 할당 되는 과정을 설명해 보시오.**

* 원본 데이터를 벡터 크기의 1.5배 크기의 메모리를 할당 한 후 복사한다. 그 후 원래 있던 벡터를 삭제하고 새로운 벡터에 데이터를 삽입한다.

**No.38-1 벡터에서 사용하지 않는 메모리를 해제 시켜주기 위해서는 어떻게 해야하는가?**

* 스왑 : 임시벡터를 선언하여 삭제하려는 벡터와 스왑을 통해 메모리를 완전 삭제 할 수 있다.

**No.39 깊이우선탐색과 너비우선탐색에 대해 설명하시오.**

* 깊이우선탐색은 스택을 사용하는 탐색기법으로 더 이상 참조할 자식 노드가 없을 때 까지 찾아가며 탐색하는 기법이고 너비우선탐색은 큐를 사용하는 탐색기법으로 검사해야 할 노드가 가지고 있는 모든 자식 노드에 대해서 탐색을 하는 기법이다.

**No.40 정렬알고리즘에 대해 설명해보시오.(7~8개)**

**No.41 동기화기법에 대해 설명하시오.**

**No.41-1 임계영역이란?**

**No.42 멀티스레드를 사용하는것이 항상 효율적인가?**

**/ 그렇지않다면 어떤경우에 스레드를 사용하는것이 효율적인가?**

**No.43-1 적정 스레드 수는?**

**No.44 쓰레드의 상태 변화에 대해 설명하시오.**

**No.45 쓰레드 메모리 구조에 대해 설명하시오.**

**No.46 메모리 영역중 커널영역과 유저영역란 무엇인가?**

**No.47 커널모드와 유저모드란 무엇인지 설명하시오.**

**No.48 쓰레드의 종류에는 무엇이 있는지 말해보시오.[커널레벨쓰레드, 유저레벨쓰레드설명하고 장단점을 말하시오.]**

**No.49 쓰레드 동기화에 대해 설명하시오.[동기화 기법이 아님. :: 유저 모드 동기화, 커널 모드 동기화]**

**No.50 데드락과 라이브락의 차이점을 설명해보시오.**

* 데드락은 스레드들이 서로 상대방이 가진 자원을 기다리느라 전혀 진도를 나가지 못하는 상황이다.
* 라이브락은 두 스레드가 락의 해제와 획득을 무한 반복하는 상태이다.

**--------------------------------------트리 정렬 쓰레드 다시공부----------------------------------------**

**No.51 OBB(Oriented Bounding Box)란 무엇인지 설명하시오.**

* 축이 정렬되지 않은 박스들 간에 충돌을 검사하는 기법.

**No.52 AABB(Axis - aligned bounding box)란 무엇인지 설명하시오.**

* 축이 정렬된 박스들 간에 충돌을 검사하는 기법.

**No.53 짐벌락은 무엇인가?**

* 오일러각을 이용하여 회전을 할 때 축의 회전이 다른 축에 종속되어 하나의 축에 대한 자유도를 상실하는 현상.

**No.55 BSP tree(Binary space partitioning tree)은 무엇인가?**

* 복잡한 공간을 사용자가 다루기 쉬운 공간이 될 때까지 분할한 트리.

**No.59 Quadtree란?**

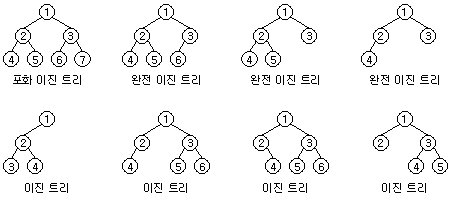
* 자식노드가 4개인 트리.

**No.60 Octree란?**

* 자식노드가 8개인 트리.

**No.64 완전이진트리란?**

* 포화이진트리에서 중간에 빠진 노드가 없는 트리.



**No.65 아스키코드와 유니코드의 차이점을 말하시오.**

* 아스키 코드는 1바이트를 사용하고 유니코드는 2바이트를 사용한다.

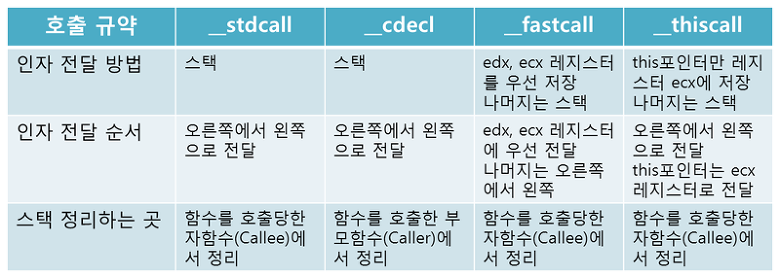
**No.66 문자셋중 SBCS, MBCS, WBCS의 특징을 말하시오.**

* WBCS : 모든 문자를 2바이트로 사용하는 문자셋
* MBCS : 싱글바이트 캐릭터셋으로 표현할 수 있는 것은 1바이트로, 다른 것은 2바이트로 사용하는 문자셋
* SBCS : 1바이트를 사용하는 문자셋

**No.67 함수 호출 규약이란?**

* 함수들간의 대화방식
* 파라메터 전달 방법, 전달 순서, 함수값 리턴 전달방법, 함수 호출간 사용했던 스택프레임을 정리하는 방법 등을 정한 약속.

**No.68 C++ 함수호출규약 : \_\_cdecl, \_\_fastcall, \_\_stdcall, \_\_thiscall 설명하시오.**



리턴 값이 8바이트 이하면 범용 레지스터를 사용, 그렇지 않으면 메모리 사용하여 그 포인터를 레지스터에 리턴.

**No.69 ecx와 edx는 무엇인가?**

* 범용 레지스터의 하나로 fastcall에서 인자를 전달할 때 빠르게 전달하기 위해 사용하는 레지스터.

**No.70 커널 오브젝트란?**

* 운영체제에서 리소스(프로세스, 쓰레드, 파일)들을 관리하기 위한 데이터를 저장하는 메모리 블록.

**No.71 핸들이란?**

* 운영체제에서 오브젝트의 구분을 위해 생성하고 발급하는 넘버링.

**No.72 인터락 함수란?**

* 여러 쓰레드에서 동시에 참조할 수 없게 만든 함수.

**No.73 volatile키워드에 대해 설명하시오.**

* 컴파일러가 코드를 컴파일 하는 과정에서 코드 최적화를 수행하는 것을 막고 값이 캐쉬에 저장되는 것을 막고 메모리에 직접 저장하는 키워드.

**No.74 쓰레드 풀에 대해 설명하고 사용이유에 대해 말하시오.**

* 쓰레드를 미리 여러 개 생성한 뒤 처리 요청이 왔을 때 활성화 되지 않은 쓰레드에게 작업을 시키는 방법.
* 컴퓨터가 성능에 맞는 최적의 쓰레드 개수만큼만 생성하기 위해.

**No.75 반복자란 무엇이며, 사용하는 이유는 무엇인가?**

* 자료구조에 효율적인 순회 및 참조를 위해 만든 객체
* 서로 다른 컨테이너에 대해 같은 인터페이스로 내부 자료를 참조하기 위해.

**No.76 함수자는 무엇인가?**

* 함수처럼 동작하는 객체.
* 객체의 함수가 inline 된다.
* 함수를 매개변수로써 사용할 수 있다.

**No.77 STL 함수자중 일반함수자에 대해 설명하시오.**

* 산술 연산, 비교 연산, 논리 연산 등의 특정 기능을 수행하는 함수자.

**No.78 STL 함수자중 함수 어댑터에 대해 설명하시오.**

* 함수, 함수 포인터 및 함수자를 인자로 받아 새로 만든 함수자.

**No.80 explicit 키워드란 무엇인가?**

* 묵시적인 생성자의 호출을 막아주는 키워드.

**No.81 mutable 키워드란 무엇인가?**

* Const로 선언된 함수 내에서 멤버 변수를 조작 가능한 변수로 선언하는 것.

**No.82 volatile 키워드란 무엇인가?**

**중복**

**No.83 정적 변수란 무엇인가?**

* 정적변수는 프로그램 실행 후 한번만 초기화가 되며 프로그램이 종료될 때까지 유지되는 변수로 선언된 영역 내에서만 접근이 가능하다.

**No.84 extern "C" 의 의미는 무엇인가?**

* 링크 규칙을 C언어의 링크 규칙으로 사용하겠다는 의미.

**No.85 특수화 템플릿이란 무엇인가?**

* 특정 자료형에 대해서만 특수한 처리를 해주기 위한 템플릿.

**No.86 스마트 포인터란 무엇인가?**

* 객체의 할당과 삭제 등 메모리를 편하게 관리하기 위해 사용하는 포인터.
* 어느 시점에서 생성하고 삭제하는지 판단해주는 기능을 내포하고 있다.

**No.88 STL에서 erase와 remove의 차이점은 무엇인가?**

* Erase는 자료구조 내의 메모리까지 삭제하는 것이고 remove는 조건에 맞는 노드를 자료구조 맨 뒤로 몰아만 주는 것. 메모리는 삭제되지 않으므로 프로그래머가 삭제해야 한다.

**No.96 TCP / UDP에 대해 설명하시오.**

* TCP
  + 연결지향
  + 데이터가 스트림 형태로 들어오게 된다.
  + 안정성이 매우 높다.

UDP

* + 비 연결지향
  + 데이터가 패키지 형태로 들어온다.
  + 안정성이 낮다.(데이터가 깨져있는지 알 수 없다.)

**No.100 다음 디자인 패턴에 대해 간략하게 설명하시오.**

**1.팩토리 메소드**

**2.추상 팩토리**

**3.템플릿 메소드**

**4.싱글톤**

**5.브릿지**

**6.메디에이터**

**7.프로토타입**

**8.옵저버**

**9.데코레이터**

**10.스테이트**

**11.스트래티지**

**12.퍼사드**

**13.컴포지트**

**IOCP는 오버랩 I/O에 스레드 풀링과 큐 라는 매커니즘을 동시에 접목시킨 기술이다.**

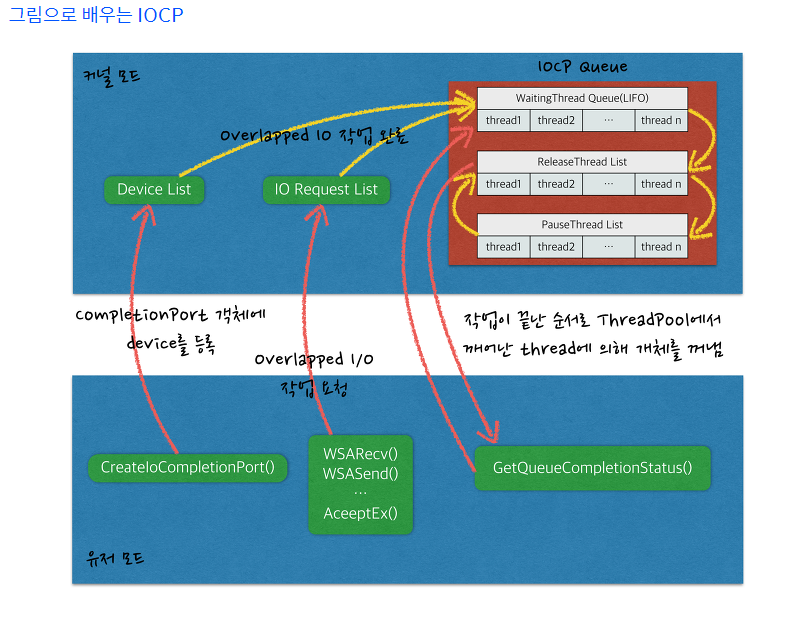
**스레드 풀링**

**스레드를 생성하고 사용 후 이 스레드를 소멸시키지 않고 대기 상태로 바꾸어 다시 풀에 넣어준다. 사용 시점에는 생성할 필요 없이 다시 대기 상태인 큐를 활성화하는 기법이다.**

**스레드 풀링이 왜 좋나요?**

**스레드를 자주 생성하고 파괴하지 않기 때문에 할당된 CPU의 시간을 낭비하지 않고 다른 일을 할 수 있다.**

**실행중인 스레드에서 다른 스레드로 작업전환을 할 때는 컨텍스트 스위칭이 발생한다. CPU 소모가 많은 작업이기 때문에 성능이 떨어진다. 스레드 풀은 이 컨텍스트 스위칭을 피할 수 있어 성능 향상에 도움이 된다.**



**Pragma pack(push N)**

**바이트 패딩.**

**32비트, 64비트 운영체제에 따라서 달라진다. Pragma pack(1)을 하면 바이트의 중복제거와 동시에 1바이트씩 데이터에 정렬이 된다.**

1. **정규화**

**-관계 데이터베이스 설계에 있어 가장 중복을 최소화 하는 것을 말한다.**

**-데이터 무결성 유지 및 안정성 최대화.**

**-1,2,3정규형 BCNF등**

**1정규형 : 테이블에 있는 모든 속성의 도메인이 원자값만으로 되어있는 정규형.**

**2정규형 : 기본키 2개 이상으로 구성되는 테이블에서 부분함수 종속성을 제거해준다.(기본키가 하나면 하지않음.)**

**3정규형 : 이행적 종속관계, A->B, B->C ==== A->C**

1. **뮤 공부…?;;; ee**
2. **커널모드 유저모드**

**유저모드 : - 사용자어플리케이션 코드가 실행됨.**

* **시스템 데이터에 제한된 접근만 허용, 하드웨어 직접적인 접근 불가능.**
* **시스템 서비스 호출 시 유저모드에서 커널모드로 잠시 전환됨.**
* **스레드는 자신만의 유저모드 스택을 가짐.**

**커널모드 : -모든 시스템 메모리에 접근 가능하고, 모든 CPU명령 실행 가능.**

* **운영체제 코드나 디바이스 드라이버 같은 커널모드 코드를 실행한다.**
* **유저모드와 커널모드를 제공하는 것은 윈도우 운영체제가 아니라 프로세서이다.**

1. **커널 유저 영역**

**커널영역 : 운영체제라는 하나의 소프트웨어를 실행시키기 위해서 필요한 메모리 공간.**

**유저영역 : 프로그램이 동작하기 위해 사용되는 메모리공간.(코드, 데이터, 스택, 힙 영역)**

1. **ODBC**
2. **레드블랙트리**

레드블랙트리는 균형이진트리 중 하나로 노드를 검은색 노드와 빨간색 노드로 나눠서 정리하는데 루트노드부터 리프노드까지 모든 경로에 대한 검은색의 높이를 같게 하여 최대 검색시간을 보장하는 자료구조이다.

작은 값노드는 왼쪽, 큰값을 가지는 노드는 오른쪽.

* 레드 및 블랙중 하나의 색을 가진다.
* 루트노드는 반드시 블랙.
* 모든 리프 노드는 블랙이다.
* 레드노드의 자식노드는 블랙, 블랙의 자식은 어떤게 와도 상관없다.
* 좌회전, 우회전.

1. **해시테이블**

**-** key와 value를 갖는 자료 구조이다. 주요 동작은 효율적인 검색(주어진 키(예를 들어, 사람 이름)로 적합한 값을 찾는(전화번호)) 이다. 해쉬 함수를 이용해서 주어진 키를 해쉬값으로 변환하고 해쉬값을 인덱스로 하여 원하는 값이 있는 버켓(bucket)을 찾아내는 것이다.

1. **시간복잡도 공간복잡도구하기.**

**-공간복잡도 : 얼마나 메모리를 많이 차지하느냐…………..**

1. **비-트리, 비+트리**

**-B 트리 :** 데이터베이스와 파일 시스템에서 널리 사용되는 **트리** 자료구조의 일종으로, 이진 **트리**를 확장해 하나의 노드가 가질 수 있는 자식 노드의 최대 숫자가 2보다 큰 **트리** 구조이다.

**루트노드는 최소한 2개의 자식을 가져야 한다.**

**모든 리프노드들은 같은 수준이어야 한다.**

**-B+트리 :** B트리와 대조적으로 B+트리는, 모든 레코드들이 트리의 가장 하위 레벨에 정렬되어있다. 오직 키들만이 내부 블록에 저장된다.

**데이터와 인덱스를 분리하여 삭제 조작이 용이. 단말노드는 정렬되어 있다.**

1. **라운드로빈 :** 프로세스들 사이에 우선순위를 두지 않고, 순서대로 시간단위(Time Quantum/Slice)로 CPU를 할당하는 방식의 CPU 스케줄링 알고리즘.

**만랩 720**