직군면접 Q&A

Q : 1분 자기소개

A :

안녕하십니까? 에이텍모빌리티 소프트웨어 개발에 지원한 이우상입니다. 저는 다양한 경험과 열정을 가진 개발자입니다. C++을 이용하여 안정적이고 효율적인 소프트웨어를 개발하는 데 관심이 많으며, 코드의 품질과 성능에 항상 신경 쓰고 있습니다. 또한 새로운 기술과 도구에 대한 열린 마음을 가지고 있어 항상 성장하고 발전하는 것을 추구하고 있습니다.

Q : 지원동기

A :

에이텍모빌리티에서 C++을 사용하여 다양한 분야에서 소프트웨어를 개발하고 있는 것을 알게 되어 흥미로웠습니다. 저는 C++을 주로 사용하며 개발해왔습니다. 특히 경력 뿐만 아니라 개인 프로젝트에서도 C++을 사용해서 게임 서버와 클라이언트를 개발했었습니다. 이러한 경험을 통해 C++의 다양한 활용 분야와 역량을 이해하고 있습니다. 그래서 저는 C++을 사용하여 에이텍모빌리티에서 소프트웨어를 개발하는 과정에 참여하고, 회사의 목표를 달성하는 데 기여하고자 지원했습니다.

Q : 퇴사 사유

A :

저는 이전 회사에서 다양한 프로젝트를 수행했었습니다. 처음에는 C++을 주로 사용하여 개발을 진행했었지만, 시간이 지남에 따라 프로젝트의 특성이 웹 개발 쪽으로 더욱 집중되었습니다. 웹 개발에 집중하다 보니 C++에 대한 실무 경험을 충분히 쌓지 못한 것을 느꼈습니다. 그래서 저는 이제까지의 C++ 학습과 경험을 활용하여 더 다양한 프로젝트에 참여하고, C++ 개발자로서의 커리어를 쌓고자 퇴사를 결정했습니다.

Q : 퇴사 후 공백기간에는 무엇을 했는지?

A :

퇴사 후 공백기간동안 c++을 사용하여 MMO게임 서버와 클라이언트를 만드는 개인 프로젝트를 진행하며 개발 역량을 길렀습니다.

Q : 자신의 장점과 단점

A :

성실함이 저의 가장 큰 장점입니다. 저의 일을 미루지 않고, 매일 계획한 부분까지 끝내려고 노력합니다. 프로젝트를 시작하면, 팀원에게 피해를 주지 않도록 최대한 저의 맡은 부분을 빠르게 구현하려고 노력합니다. 또한 팀 프로젝트에서 의사소통이 중요한 부분 중 하나입니다. 그래서 저는 다른 사람의 이야기를 잘 들어주는 성격입니다. 최대한 다른 사람의 이야기를 귀 기울이면서 그 사람과의 관계를 좋게 유지하려고 노력합니다.

단점은 걱정이 많은 것입니다. 프로젝트에서 저의 구현 파트가 발견하지 못한 버그가 생길까 걱정을 많이 합니다. 그래서 이러한 저의 단점을 극복하기 위해 끊임없이 테스트를 하여 저의 실수와 걱정을 줄이려고 노력합니다.

Q : 데드락이란?

A : 두개 이상의 프로세스 혹은 스레드가 서로 작업이 끝나기 만을 기다리고 있는 상태

Ex) 자원1이 프로세스A에게 할당되고, 자원2가 프로세스B에게 할당되었을 때, 프로세스A가 자원2를 요구하고, 프로세스B는 자원1을 요구할 때 발생한다.

Q : 데드락의 발생 조건

A : 아래 4가지 조건을 만족할 때 발생한다.

1. 상호 배제 : 오직 하나의 프로세스만이 자원을 사용할 수 있는 것 Ex) mutex
2. 점유와 대기 : 프로세스A가 자신에게 할당된 자원을 점유하면서 다른 프로세스가 보유하고 있는 자원을 추가로 요구하고 기다리는 상태
3. 비선점 : 어떤 프로세스에 할당된 자원을 다른 프로세스가 반환할 때 까지 강제로 선점할 수 없다. Ex) mutex lock( )을 걸고, unlock( )을 할 때까지 다시 lock( )을 할 수 없다.
4. 환형 대기 : 각 프로세스가 자신에게 할당된 자원을 보유하면서, 상대방의 자원을 요청하는 점에서 프로세스와 자원들이 원형을 이룬다.

Q : 데드락 해결 방안

A :

1. 예방 : 데드락 발생 조건 4가지중 한가지를 제거

* 자원 낭비 문제 발생

1. 회피 : 프로세스들의 자원 사용 정보를 파악하여 데드락이 발생하지 않는 안전한 상태에서만 자원 요청을 허용

Ex) 은행원 알고리즘

1. 탐지 : 데드락이 발생했는지 지속적으로 탐지하고, 데드락이 발생하면 회복 기법을 통해 복구

* 데드락을 계속 확인해야 하므로 성능 저하 발생

1. 회복 : 데드락을 해결하는 기법

* 데드락이 발생한 프로세스 중지
* 프로세스로부터 자원을 회수

Q : 프로세스와 스레드 차이

A : 프로세스는 운영체제로부터 자원을 할당받은 작업의 단위

스레드는 하나의 프로세스 내에서 실행되는 여러 흐름의 단위

Q : 스레드의 컨텍스트 스위칭

A : 동일한 프로세스 속에서 하나의 스레드A를 중지하고 다른 스레드B를 실행하는 것

* 멀티스레드 환경에서만 발생된다.
* 컨텐스트 스위칭 동안에는 CPU가 일을 하지 못하기에 너무 잦으면, 오버헤드가 발생

Q : 멀티 스레드에서의 동기화 방법

A :

1. Mutex : 공유된 자원의 데이터를 여러 스레드가 접근하는 것을 막고, 오직 1개의 스레드만 허용하고, 프로세스간 동기화에서 사용

* lock( ), unlock( ) 사용

1. 세마포어 : 임계 영역에 들어갈 수 있는 스레드가 1개이상

* 상호 배제를 보장할 수 없다.
* wait( ), signal( ) 사용

1. 스핀락 : 임계 영역에 진입이 불가능할 때, 진입이 가능할 때까지 루프를 돌면서 재시도

Q : 메모리의 구조

A : 커널, 스택, 힙, 데이터, 코드 영역으로 분류

Q : 변수의 저장 위치

A :

1. 스택 영역 : 지역 변수

* { }을 벗어날 때 자동으로 소멸

1. 힙 영역 : 동적할당한 변수(new)

* 사용자가 직접 해제, 스마트 포인터는 자동으로 해제

1. 데이터 영역 : 전역 변수, 정적 변수(static)

* 프로그램이 종종 될 때 메모리에서 해제

Intel TBB, VisualStudio PPL

Q : 스마트 포인터의 종류와 각 설명

A :

1. unique\_ptr : unique\_ptr 객체는 자신이 가리키는 리소스와 1:1로 대응되고, 객체의 복사나 대입 연산자가 허용되지 않는다.
2. shared\_ptr : 두개 이상의 객체가 한 개의 리소스 참조할 때 shared\_ptr을 사용하여 참조(reference count)개수를 알 수 있다
3. weak\_ptr : shared\_ptr 사용시 발생할 수 있는 순환 참조의 문제를 막기 위해 사용

* 참조 개수의 증감에 관여되지 않고, 포인터가 가리키는 리소스를 사용하기 위해

Q : 얕은 복사와 깊은 복사

A :

1. 얕은 복사 : 객체를 복사할 때 만약 객체의 멤버변수가 동적할당을 한다면, 복사된 객체의 멤버변수도 같은 주소를 가리키게 된다.

* 따라서 복사된 객체가 소멸될 때, 할당받은 공간을 해제하면서 원본 객체의 멤버가 해제된 주소를 가리키는 댕글링 포인터가 되면서 또 해제하려는 문제가 발생

1. 깊은 복사 : 객체가 복사될 때, 복사 생성자에서 새로 동적할당을 하여, 그곳에 원본 데이터를 복사하는 방식

Q : 퀵 정렬이란?

A :

1. 임의의 pivot 값을 설정한다.
2. pivot을 기준으로 i는 pivot보다 큰 값을 찾고, j는 pivot보다 작은 값을 찾는다.
3. 이에 해당하면 i와 j값을 서로 바꾼다.

* 해당하지 않으면 바꾸지 않는다.

1. i인덱스는 1개 증가시키고, j인덱스는 1개 감소를 반복
2. i인덱스가 j인덱스보다 크면, i의 값과 pivot의 값을 서로 바꾼다.
3. pivot값 좌우의 리스트에 대해 재귀적으로 수행

Q : 퀵 정렬의 장단점

A :

장점 : 평균적으로 시간복잡도가 O(nlogn)이 걸려 빠르다.

단점 : pivot을 설정한 값이 가장 작은 값 혹은 가장 큰 값을 설정하면, 최악의 시간복잡도인 O(n^2)이 걸린다.

Q : qsort 와 sort의 차이점

A :

qsort의 인자 :

1. void\*형 데이터
2. 개수
3. 원소의 크기
4. int compare(const void\* a, const void\* b) {}

sort의 인자

1. 시작 위치
2. 끝 위치
3. bool compare(const int& a, const int& b) {}

qsort는 퀵 정렬으로 구현되서 시간복잡도가 최악인 경우 O(n^2)

sort는 introsort(퀵 정렬 + 힙 정렬 + 삽입 정렬)로 구현되서 하이브리드 정렬 알고리즘이다. 그래서 시간이 O(nlogn)으로 일정

Q : std::sort와 list.sort의 차이점

A : std::sort는 랜덤 액세스가 필요하므로 list에서 사용할 수 없다. 또한 list의 요소 유형이 스왑 가능하지 않아도 된다.

Q : 객체지향의 5원칙

A :

1. SRP : 하나의 클래스는 하나의 목적을 위해 만들어져야 한다.
2. 개방폐쇄의 원칙 : 확장에는 열려있고, 수정에는 닫혀있어야한다.
3. 리스코프의 치환 원칙 : 자식클래스는 부모클래스를 대체할 수 있다.
4. 인터페이스 분리 원칙 : 사용하지 않는 인터페이스는 구현하지 않는다.
5. 의존역전원칙 : 의존관계를 맺을 때, 변화가 거의 없는 것에 의존.

* 클래스를 참조해야할 때, 추상클래스를 만들어 부모 클래스를 참조

Q : OOP(객체 지향 프로그래밍) 이란?

A : 절차 지향 프로그래밍과 반대되는 말로, 필요한 데이터를 추상화 시켜, 상태와 행위를 가진 객체로 만들고, 객체들 간의 상호작용을 통해 로직을 구성하는 방법

Q : OOP의 4가지 특성

A :

1. 캡슐화 : 클래스 내부에서 멤버 변수와 함수를 외부에 노출하지 않고 데이터와 알고리즘을 보호하는 것

* 정보 은닉

1. 상속 : 코드의 재사용을 목적

* 다중 상속 : 하나의 클래스가 두가지 이상 상속을 받는 경우
* 문제점 : Animal – 오리, 너구리 – 오리너구리(오리와 너구리 둘다 상속 받음)

1. 추상화 : 비슷한 개념의 데이터나 작업을 묶어 공통되게 관리하는 것
2. 다형성 : 서로 다른 객체들이 동일한 기능을 서로 다른 방법으로 처리할 수 있는 기능

* 부모의 함수를 호출하면, 상속받은 자식의 오버라이딩 된 함수가 호출된다.

Q : 가상함수, 순수 가상함수

A :

1. 가상함수 : virtual 키워드를 부모 클래스에서 선언하여 자식클래스에서 재정의할 것으로 기대하는 함수
2. 순수가상함수 : 자식클래스에서 반드시 재정의 해야 하는 멤버함수

Q : 동적 바인딩

A : 런타임에 호출될 함수가 결정되는 것

* 컴파일러가 가상 함수의 호출은 미리 알 수 없다. 왜냐하면 런타임에 객체가 결정되기 때문에

Q : 추상클래스

A :

1. 추상클래스 : 순수 가상함수를 포함하는 클래스

* 추상클래스는 객체로 만들 수 없으며, 상속을 목적으로 사용하는 클래스

Q : inline 함수

A : 전역함수와 다르게, 컴파일러가 함수의 호출을 거치지 않고, 함수의 모든 코드를 호출된 자리에 삽입하는 함수이다.

Q : 동기와 비동기

A :

동기 : 작업이 완료될 때 까지 기다려야한다.

비동기 : 작업이 완료될 때 까지 기다리지 않고 다른 일을 처리할 수 있다.

Q : 람다함수

A : 무명함수를 뜻하고 [캡처졀](인수){본문} 으로 사용

Q : RVO, NRVO

A :

RVO : 리턴 값을 임시객체를 생성하지 않고 바로 리턴해주는 것

NRVO : 리턴 값을 임시객체를 생성하여 리턴하는 것

Q : 프로그램 빌드 과정

A :

1. 전처리 : #include, #define과 같은 키워드에 따라 필요한 파일을 읽고, 실제 컴파일이 가능하도록 소스를 만든다.
2. 컴파일 : 컴파일러가 소스를 어셈블리 코드로 만든다.
3. 어셈블링 : 어셈블리 코드를 어셈블러에 의해 오브젝트 코드로 만든다.
4. 링크 : 오즈젝트 코드를 실행 가능한 프로그램으로 만든다.

Q : TCP와 UDP

A :

1. TCP :
2. 연결형 프로토콜
3. 신뢰성 있는 데이터 전송
4. 데이터를 재전송함
5. 1:1 통신 – 유니캐스팅
6. UDP :
7. 비연결형 프로토콜
8. 신뢰성 없는 데이터 전송
9. 데이터를 재전송하지 않음
10. 1:1 통신 - 유니캐스팅
11. 1:n 통신 – 브로드캐스팅, 멀티캐스팅

Q : volatile

A : volatile 변수를 참조할 경우 레지스터의 값을 사용하지 않고, 매번 메모리를 참조한다. – 최적화를 하지 않음

Q : data race

A : 여러 스레드가 공유 메모리를 읽고 쓴다. 읽고 쓰는 순서에 따라 실행 결과가 달라짐

Q : Transaction

A : 데이터베이스의 상태를 변화시키기 위해 수행하는 작업의 단위

* Commit : 트랜잭션이 성공적으로 끝났을 때 결과를 테이블에 반영한다.
* Rollback : 트랜잭션이 실패한 경우, 트랜잭션 수행전으로 되돌린다.

Q : SP(Stored Procedure)

A : 일련의 동작들을 SQL로 프로그래밍해서 서버에 저장해 놓은 것

* 쿼리문을 여러 번 실행하거나 복잡한 로직을 처리할 때 SQL 서버에 한단위로 저장해서 편리하게 호출할 수 있다.
* 네트워크 트래픽이 감소하고, 성능이 향상된다.

Q : DB 인덱스

A : 테이블에서 데이터의 검색 속도를 높이는 자료구조

Q : NoSQL

A : 비관계형 데이터베이스

Q : map vs unordered\_map

A :

1. map :
2. key, value 쌍으로 데이터를 저장
3. 이진 탐색 트리로 구현
4. key값에 의해 데이터를 정렬하여 관리한다.
5. 중복된 key를 허용하지 않음
6. 시간복잡도는 O(logn)
7. unordered\_map :
8. key, value 쌍으로 데이터를 저장
9. 해쉬테이블 기반으로 구현
10. 데이터들을 정렬하지 않고 관리한다.
11. 중복된 key를 허용하지 않음
12. 시간복잡도는 O(1)

Q : 싱글톤 패턴?

A : 하나의 객체를 생성하여 사용하는 디자인 패턴

* 클래스내에 자기 자신의 static 변수를 갖는다.
* static 함수를 사용하여, 객체를 얻는다.

Q. 인터프리터

A. 한 줄 단위로 즉시 실행시키는 프로그램

-> 생산성이 올라간다.