​

1. 대략적으로 1달 정도의 시간이 흘렀습니다.

1달의 시간 동안 느낀점과 간략한 소감을 작성해주세요.

부가적으로 앞으로의 포부 또한 같이 작성해주시면 감사하겠습니다.

**1**. 1달의 시간 동안 C/C++을 공부하면서 처음 코딩을 배우고 싶다는 생각으로

인터넷과 주변인들에게 물어보며 이 학원을 찾아왔고 집을 오가는 시간과

집에서 나름 코딩을 배우기 위해 공부했지만

아직은 너무 어렵기만 한 기술 같습니다.

하지만 만들지는 못하지만 프로그램이 만들어지는

과정과 만들어진 결과물을 보면서

저란 사람은 일을 하면서 서로의 생각을 공유하고 회의하며 설득과 이해가

필요한 그런 직업을 제가 좋아했던 것 같습니다.

그래서 저도 빨리 타인과 의견을 공유하면서 결과물을 만들고 싶은 생각뿐입니다.

생각보다 빠르게 시간이 흘러 이제는 수료 전까지 얼마나 많은 발전을 할지

조급하고 불안하지만 그래도 강사님이 말씀하신 "역치"를 감내하며 성장해

보도록 하겠습니다

​

2. 언어를 가리지 않고 Domain Driven Design을 수행하여 얻을 수 있는 이점들에 대해 기술하세요.

1. 각각의 개발자마다 한 가지를 책임지고 개발할 수 있다.

2. 아직 구현하지 못한 부분이 있어도 다음 작업을 먼저 개발할 수 있다.

3. 처음 구상한 아젠다를 유지하며 개발할 수 있다.

4. 추가적인 시스템을 도입할 때 유기적인 관계가 적어 시스템 추가 시 쉽다

5. 새로운 팀원이 합류했을 때 자신의 업무를 파악할 때 적응 속도가 빠르다.

6. 새로운 프로젝트를 진행할 때 기존에 사용한 부분과 유사한 점이 있다면

재사용이 쉬워 결과물을 더욱 빠르게 개발할 수 있어

불필요한 지출이 없어진다.

​

3. C/C++ 에서 Test 목적으로 GoogleTest를 사용하였습니다.

TDD(Test Driven Development)를 통해 얻을 수 있는 이점들에 대해 기술하세요.

1. 아직 구현하지 못한 부분이 있어도 다음 작업을 먼저 개발할 수 있다.

2. 코드가 길어지면 어느 부분에서 충돌이 생기는지 찾기가 어려워 새로 작성하기 전

먼저 만들고자 하는 코드를 GoogleTest에서 구동을 해본 뒤 문제점을 찾고

개발이 가능하다.

3. 이미 만들어둔 코드를 유기적 관계를 끊어 버리기 위한 시도할 때 사용할 수 있다.

4. 이미 사용 중이라면 DB를 손상 없이 개발도 가능해진다.

5. 오류가 발생할 만한 상황들을 먼저 만들어보면서 발생할 문제점들을 미연에 방지할 수 있다.

​

4. DDD(Domain Driven Design)과 TDD(Test Driven Development)의 관계에 대해 기술하세요.

1. 아젠다를 확실하게 유지하도록 도와준다.

2. 새로운 부분을 추가할 때 기존의 데이터를 유지하며 개발할 수 있다

3. 오류가 발생할 만한 상황들을 먼저 만들어보면서 발생할 문제점들을 미연에 방지할 수 있다.

​

5. 팀원들과 협업을 진행 할 때 중요하다 생각하는 요소들을 작성하고

각 요소들에 대해 조금 더 상세하게 기술하세요.

1. 이해 - 팀 단위를 작업을 진행 할 때 그 어떤 헛소리같은 말이라도 일단 공유하며 서로의 생각을 맞춘다.

2. 설득 - 팀원과 충분한 이야기를 했다면 목표로 하는 작업물을 위해 팀원들과 소통하며 초안을 구성한다.

3. 저장 - 회의중 내용을 저장하고 초안을 작성하면서 나온 내용을 backlog단위로 저장한다.

-초안, 아이디어 회의중 다른 아이디어로 삭제된 내용, 나중에 추가 할만한 기능등-

4. 리더 - 팀원들의 기량을 알고 적절한 인원 분배, 지시가 가능한 리더가 정말 필요하다.

꼭 리더가 아니여도 설득 부분에서 윗사람들과 최대한...........노력해본다. <이상적인>

​

6. 함수 포인터 테이블을 사용하는 이유에 대해 IoC 관점에서 기술하세요.

-함수 포인터 테이블은 의존성 제어, 동적 실행 로직 변경, 테스트 용이성, 확장성

1. 의존성 제어: 함수 포인터 테이블은 의존성을 제어하는 데 사용될 수 있습니다.

IoC는 의존성을 역전시켜 의존하는 객체가 직접 의존성을 관리하는 대신 외부에서 의존성을 주입받는 방식입니다.

함수 포인터 테이블은 이러한 외부 주입을 가능하게 하며, 의존하는 객체가 특정 함수를 호출할 때 실제로 실행되는

함수를 외부에서 결정할 수 있습니다. DDD

2. 동적으로 실행 로직 변경: 함수 포인터 테이블은 실행 로직을 동적으로 변경할 수 있는 유연성을 제공합니다.

객체의 동작을 변경하기 위해 객체를 수정하거나 파생 클래스를 만들 필요 없이,

함수 포인터 테이블을 업데이트하여 다른 로직을 실행할 수 있습니다.

이는 실행 로직의 변경이나 확장이 필요한 경우에 유용합니다. DDD

3. 테스트 용이성: 함수 포인터 테이블은 테스트 용이성을 높일 수 있습니다.

테스트 중에 특정 함수를 가짜 또는 목 객체로 대체하여 원하는 동작을 시뮬레이션할 수 있습니다.

이를 통해 의존성을 가진 객체를 격리시키고, 테스트의 안정성과 예측 가능성을 향상시킬 수 있습니다.

GoogleTest

4. 확장성: 함수 포인터 테이블은 애플리케이션의 확장성을 높일 수 있습니다.

새로운 기능이나 동작을 추가할 때, 함수 포인터 테이블에 새로운 함수를 등록하여

기존 코드 수정 없이 새로운 동작을 수행할 수 있습니다.

이는 애플리케이션의 유지 보수성과 재사용성을 향상시키는 데 도움이 됩니다. DDD

​

7. virtual method를 사용하는 이유에 대해 기술하시오.

재사용과 유지보수를 위해

1. 런타임 다형성: virtual method는 런타임 다형성을 제공합니다.

프로그램은 실행 중에 실제로 호출되는 메서드를 동적으로 결정할 수 있습니다.

2. 확장성과 유연성: virtual method를 사용하면 새로운 서브클래스를 추가하거나

기존 클래스의 동작을 변경할 수 있습니다.

코드의 확장성과 유연성을 높이는 데 도움이 됩니다.

3. 인터페이스 정의: virtual method는 추상적인 개념을 구체화하고, 인터페이스를 정의하여

다양한 클래스가 동일한 메서드를 구현할 수 있도록 합니다.

이는 코드의 모듈화와 재사용성을 높이는 데 기여합니다.

4. 다중 상속의 문제 해결: virtual method는 다중 상속에서 발생하는 메서드 충돌 문제를 해결하는 데 도움이 됩니다.

다중 상속에서는 여러 부모 클래스에서 동일한 메서드를 상속받을 수 있습니다.

이때 virtual method를 사용하면 서브클래스에서 해당 메서드를 재정의하여 충돌을 해결할 수 있습니다.

​

8. 협력형 과제를 진행하는 상황 혹은 회사에서 업무를 진행하는 상황에서

갑자기 대응하기 어려운 문제를 마주하게 되었습니다.

이런 경우 여러분들은 어떤 형식으로 문제에 대응 할 것인가요 ?

어떻게 대응 할 것인지 상세하게 기술하세요.

1. 선택 - 지금 발견한 문제점이 현재 진행 중인 과제를 멈추고 투입해야 하는가?

YES - 즉시 안건으로 올림과 동시에 회의를 진행하여 가용 인원을 구성한다.

NO - 진행 중인 과제를 이행 후 문제점을 수정한다.

NO - 개인의 자질 부족으로 문제점을 파기하지 못하는 중이라면 일을 해결하기 위해

함께 논의할 만한 대상을 찾아 논의한다.

문제를 발견하고 혼자 해결하지도 못하는데 혼자서 해결해 본다고 시간 투자해도 남들은 모른다.

2. 불가항력(클라이언트, 상사의 지시)

일단 진행한다는 기술적으로 불가능해도 일단 진행하는 모습을 보인 후 다시 한번 설득, 이해를 진행한다.

​

9. Backlog를 작성함으로서 얻을 수 있는 이점들에 대해 기술하세요.

1. 목표를 잡고 구현할 수 있다.

2. 프로젝트가 산으로 갈 때 다시 가르마를 탈 수 있다

3. 파기한 아이디어가 나중에 다시 사용하고자 할 때 기록이 되어있어 살리기 편하다.

4. 현재 팀원들의 업무 진행도를 볼 수 있다.

5. 버그 발생 시 진행 중인 업무를 보며 인원을 유동적으로 가용할 수 있다.

​

10. 여러분들은 이미 Board (게시판) 을 Console UI와 함께 구현해봤습니다.

이 때 여러분들이 각자 분업하여 만들었던 작업들이 존재합니다.

여러분들이 각자 분업하여 만든 내용물들이

다른 사람들의 결과물과 원활하게 결합하지 못하여 폐기처분하는 경우도 있었습니다.

혹은 끔찍한 혼종이 탄생하는 경우도 있었습니다.

이런 현상이 나타나게 된 근본적인 이유에 대해 기술해봅시다.

1. googleTest를 잘못 이용했습니다 - 먼저 구현 후 동작 확인용으로 사용하려 했습니다.

1-1 각 기능마다 객체 지향적으로 구현했어야 했는데 너무 물려있었다.

2. 개인 간의 능력치의 갭?

​

11. 기능 단위로 Backlog 를 작성하는 경우 어떤 문제들이 발생 할 수 있는지 기술해봅시다.

1. 각기 다른 도메인에서 기능을 구현할 때 기능 단위로 BackLog를 작정하면 나와 다른 도메인을 맞은 개발자가

이미 구현한 줄 알고 유기할 가능성이 크다

2. 구동 중 문제(기능추가, 유지보수)점을 찾으려 할 때 한눈에 보기 힘들다

​

[ 복합 문제 12 ~ 20 ]

12. 속도 Entity를 작성합니다. (복잡도를 낮추기 위해 2D 로 제한)

​

13. DB에 속도 정보와 등록 시간을 저장 할 수 있는 테이블을 작성합니다.

​

14. DB에 접속 할 수 있는 싱글톤을 생성합니다.

​

15. 소수점 형태의 랜덤 sleep 기능을 구현합니다.

​

16. 10번 정도 루프를 돌면서 0.3 ~ 1.0초 사이의 시간 이후 랜덤 속도 정보를 DB에 저장합니다.

​

17. 가속도(Acceleration) Entity를 작성합니다 (마찬가지로 2D)

​

18. 속도 정보를 토대로 가속도 정보를 산출합니다.

​

19. 속도 정보를 토대로 전체 이동 거리를 계산합니다.

​

20. 속도 정보를 토대로 현재의 변위 벡터를 계산합니다 (초기 위치 [0, 0] 으로 가정)

​

21. 현재 실시간으로 분석되는 영상 시스템을 사용하고 있습니다.

이 영상 시스템은 촬영하는 구간에 사람이 몇 명 있는지 확인 할 수 있습니다.

이러한 시스템을 특정 시간에 사람들이 붐비는 구간에서 사용하고자 합니다.

영상 시스템의 목적은 버스 정류장에 사람이 몇 명 있는지 판정하려고 합니다.

어떻게 DB에 데이터를 저장해야 스토리지를 효율적으로 사용 할 수 있을까요 ?

1. 특정 라인에서 걸음이 수초 이상 멈춘 사람

라인1 - 정류장 인근 범위를 표현

라인2 - 정류장에 그려진 선을 지칭

2. 정류장 부근에 도착 후 버스 시간표를 바라보는 사람

3. 정류장 의자에 앉아있는 사람

예외 처리- 앉아있는 사람이 정류장에 도착후 가장 오래 걸리는 버스가 도착했음에도

정류장에 잔류 중이라면

4. 위 모든 내용은 새로운 버스가 도착 후 출발한다면 데이터를 삭제한다.

​

22. 포인터가 필요한 이유가 무엇인지 기술하세요.

포인터는 메모리 접근과 관리, 데이터 공유와 전달, 동적 메모리 할당, 데이터 구조 구현

프로그래밍의 유연성과 효율성을 높이는 데 사용합니다.

​

23. 모든 함수들이 자신만의 개별적 공간인 Stack을 사용합니다.

함수들끼리 서로 Stack 을 공유하지 않는 이유에 대해 기술하세요.

1. 지역 변수와 매개 변수의 독립성 - 변수에 영향을 주거나 받지 않기위해

2. 호출 순서와 실행 상태관리

3. 지역성과 보호

개별적인 스택을 사용함으로써 함수들은 독립적으로 작동하고, 안정성과 데이터 무결성을 유지할 수 있다.

[ 복합 문제 24 ~ 32 ]

24. C언어로 Product (고유번호, 가격, 상품명, 상품 상세 설명) Entity를 작성합니다.

​

25. 해당 Entity와 관련한 사항들이 잘 동작하는지 확인하기 위해

GoogleTest를 통해 테스트를 진행합니다.

​

[ 복합형 DDD 문제 26 ~ 30 ]

S(변위) = V\_0(초기 속도) \* t(시간) + 0.5 \* a(가속도) \* t^2(시간 제곱)

V(속도) = V\_0(초기 속도) + a(가속도) \* t(시간)

​

위 공식은 고등학교 물리학 시간에 배우는 등가속도 법칙입니다.

시간에 따라 값들이 변하는 시간 함수 형태라면 위의 식들에 미적분이 필요합니다.

(일단 우리는 미적분을 적용하는 문제를 풀지는 않을 것입니다)

​

우주에서 레일건을 쏴서 적군의 전함을 공격하는 시나리오를 가정하겠습니다.

아주 이상적인 환경이라 가정하고 계산을 진행합니다.

​

전자기력을 이용하여 물체를 가속시키기 때문에 우주에서도 사격이 가능합니다.

레일건 함포의 길이는 110m 입니다.

탄환은 10m 구간에서 가속하여 발사됩니다.

고도의 과학 기술로 인해 전자기력에 의한 가속도는 500만 m/s^2 로 일정하다 가정합니다.

​

26. 문제에서 거론된 함포가 쏘는 탄환의 종단 속도를 계산 할 수 있도록 Backlog를 작성합니다.

​

27. 이에 대한 테스트 코드를 작성합니다.

​

28. 작성한 Backlog를 토대로 탄환의 종단 속도를 계산합니다.

​

29. 탄환이 포신의 종단에 도달하기까지 걸린 시간을 계산 할 수 있도록 Backlog를 작성합니다.

​

30. 이에 대한 테스트 코드를 작성합니.

​

31. 작성한 Backlog를 토대로 탄환이 포신의 종단까지 걸린 시간 값을 계산합니다.

​

32. 적군의 전함까지 떨어져 있는 거리는 대략 10000 km 입니다.

함포를 발사했을 때 적군의 함정에 착탄되기까지 걸린 시간을 계산해 봅시다.

​

33. 처음에는 Domain 분리가 필요 없다 생각하였던

특정 Entity 내부의 어떤 멤버 변수 (필드) 가 있습니다.

시간이 지남에 따라 시스템이 커졌고 위 내부 변수가 수행하는 작업의 복잡도가 높아지고 있습니다.

여러분들이라면 이 녀석을 어떻게 처리하겠습니까 ?

​

[ 복합 문제 34 ~ 35 ]

이 문제를 통해서 C언어로 Request, Response 객체의 free()를 크게 신경쓰지 않아도 됩니다.

​

34. C언어로 Request 객체를 전송하고 이 객체를 받아서 변환 이후

알아서 자동으로 객체가 free 되도록 코드를 작성해봅시다.

​

35. C언어로 응답 객체를 만들고 이를 요청했던 부분에 전달합니다.

Response 객체를 사용하면 자동으로 객체가 free 되도록 코드를 작성합시다.

​

36. Valgrind를 구동하여 Memory Leak이 발생하지 않는지 최종적으로 체크합니다.