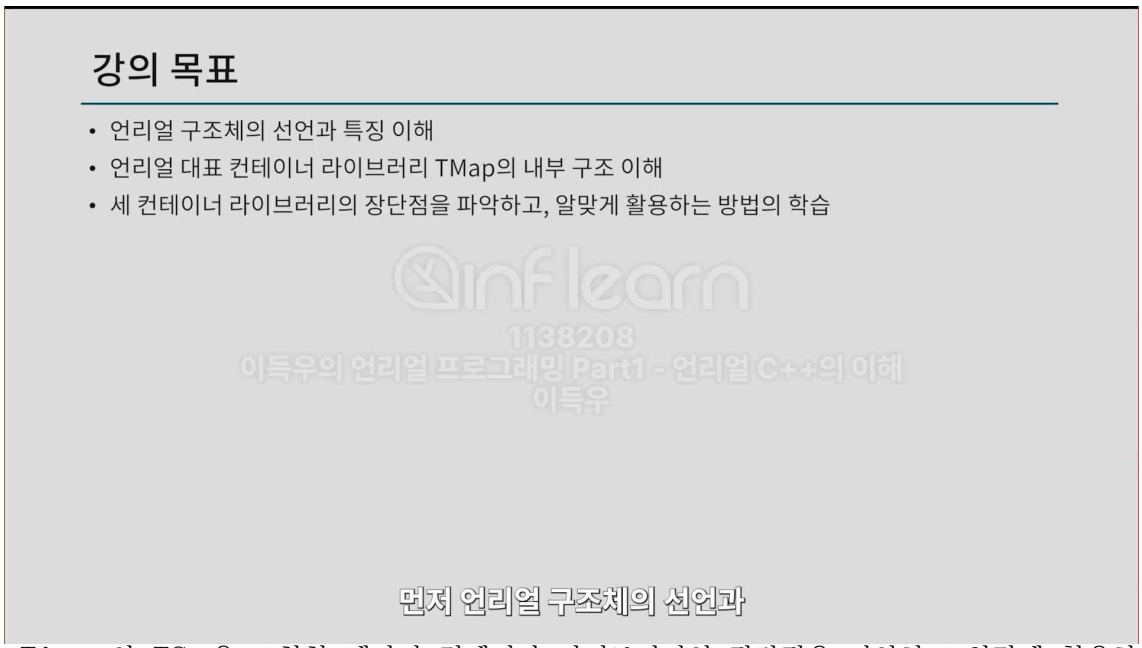
언리얼 프로그래밍 Part1-11

제목:언리얼 컨테이너 라이브러리 II - 구조체와 Map

**강의 내용 : 언리얼 구조체의 특징을 이해하고, 다양한 컨테이너 라이브러리에서 구조체를 활용하기

**강의 목표



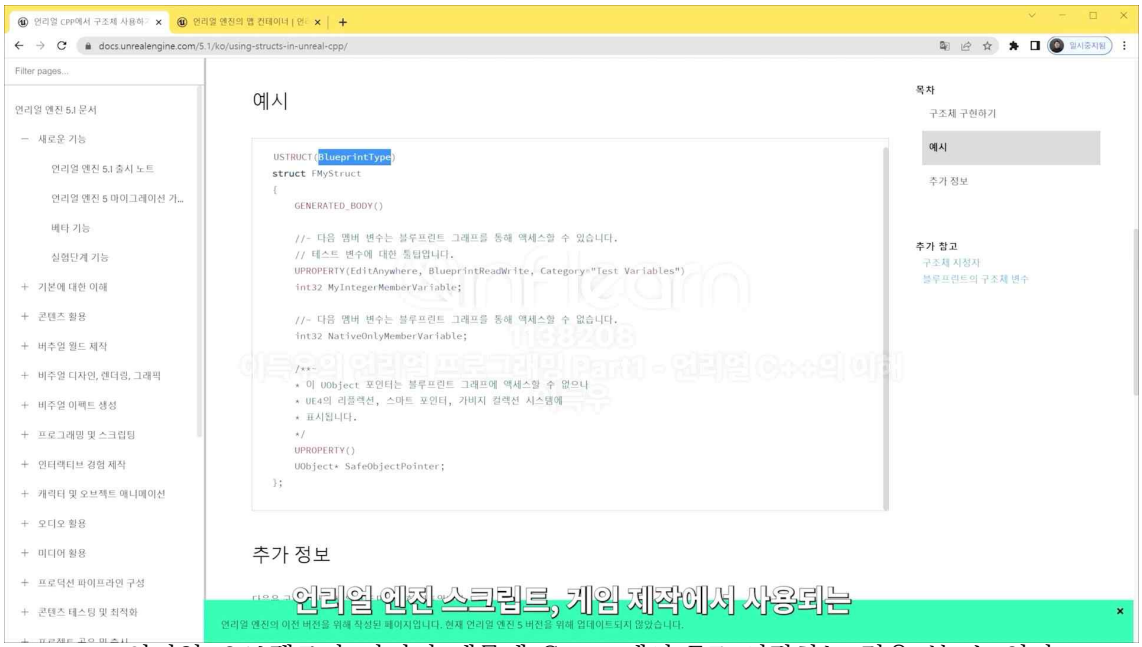
-TArray와 TSet을 포함한 세가지 컨테이나 라이브러리의 장단점을 파악하고 알맞게 활용하는 방법을 학습할 것임.

**공식문서
-링크(5.1 버전 링크가 막혀서 5.2버전으로 대체함) : https://dev.epicgames.com/documentation/ko-kr/unreal-engine/structs-in-unreal-engine?application_version=5.2

-구조체는 데이터에 특화된 자료 형태이다. 구조체를 사용하면 커스텀 변수타입을 생성할 수 있고, 프로젝트를 체계화할 수 있다. 즉 여러 가지 데이터들을 한데로 묶기 때문에 체계적인 관리가 가능하다.

-공식 문서는 구조체 구성 방법과 커스터마이징 방법을 알려준다.

-구조체 구현하기
(1) 구조체를 정의하려는 헤더 파일을 연다.
(2) USTRUCT 매크로를 구조체 구문 앞에 추가한다. 그리고 매크로 안에다가 필요하다면 지정자를 포함해준다.
(3) 구조체에서 GENERATED_BODY매크로를 추가한다.
(4) 구조체 멤버 변수에서 필요한 부분에 대해서는 UPROPERTY로 태그해 준다.
(5) 구조체 예시

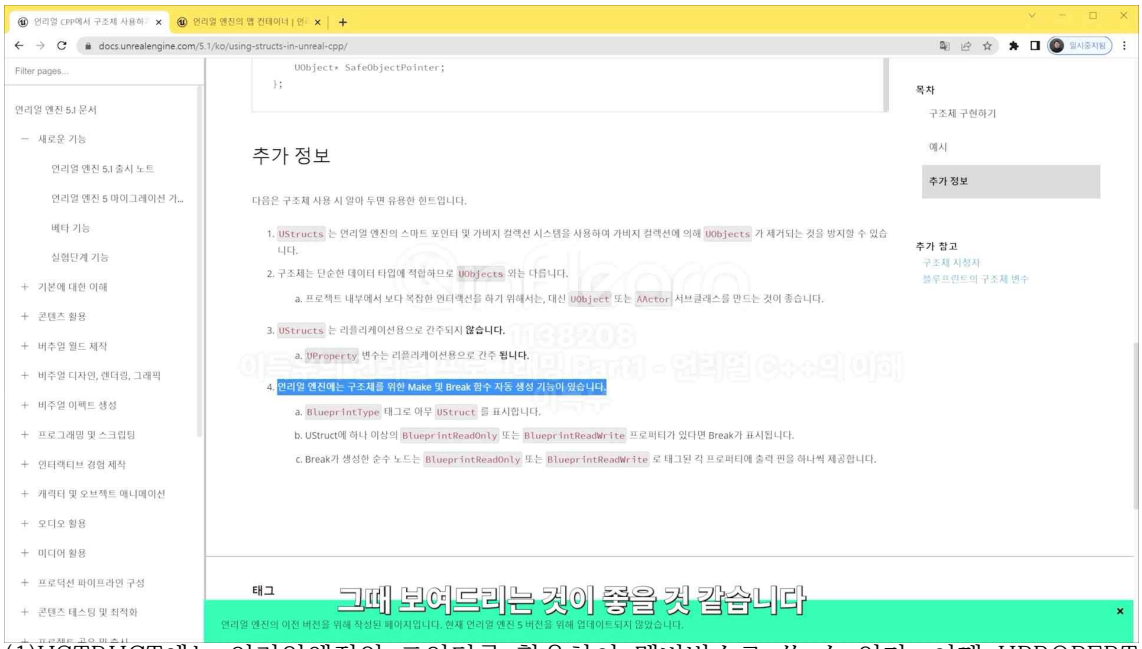


-언리얼 오브젝트가 아니기 때문에 Struct에서 F로 시작하는 것을 볼 수 있다.
-Struct선언위에 USTRUCT라는 언리얼 특유의 매크로를 선언 하였다.\
-BlueprintType이라는 키워드를 추가해주게 되면 해당 구조체는 언리얼 엔진 스크립트, 게임 제작에서 사용되는 블루 프린트와 호환되는 데이터 성격을 가지게 된다.
-GENERATED BODY() 매크로는 언리얼 오브젝트와 동일한데 관련하여 구조체가 가지고 있는 각종 속성들을 리플렉션 해서 활용할 수 있는 뼈대를 제공한다.
-그 다음에 구조체 맴버 변수들을 추가해주면 되는데 UPROPERTY라는 매크로를 달아주면 리플렉션 기능이들어가면서 앞으로 구조체에 멤버 변수를 조회한다던지 이후에 블루프린트라고 불리는 언리얼 에디터에서 동작하는 스크립트와 연동이 가능하다.

-UPROPERTY를 안붙일수도 있는데 이 경우에는 c++멤버변수로써 활용하면 된다.

-하단에 UPROPERTY()라고 단순히 선언한 경우에는 이 오브젝트의 포인터, 언리얼 오브젝트 포인터를 직접 관리할 수 있는데 구조체에 이렇게 언리얼 오브젝트 포인터를 선언할 땐 반드시 UPROPERTY()를 붙여야 자동으로 메모리 관리가 가능하다.

-추가정보



(1)USTRUCT에는 언리얼엔진의 포인터를 활용하여 멤버변수로 쓸 수 있다. 이때 UPROPERTY매크로를 반드시 지정해줘야지 UObject가 제거되는걸 방지한다던지 메모리 관리가 가능하다.

(2) c++문법상 struct라는 클래스는 큰 차이가 없고 기본 접근자 차이밖에 없다. 하지만 언리얼 엔진에서 언리얼 오브젝트와 이 Ustructs가 붙은 언리얼 구조체는 사용 용도가 완전히 다르다.

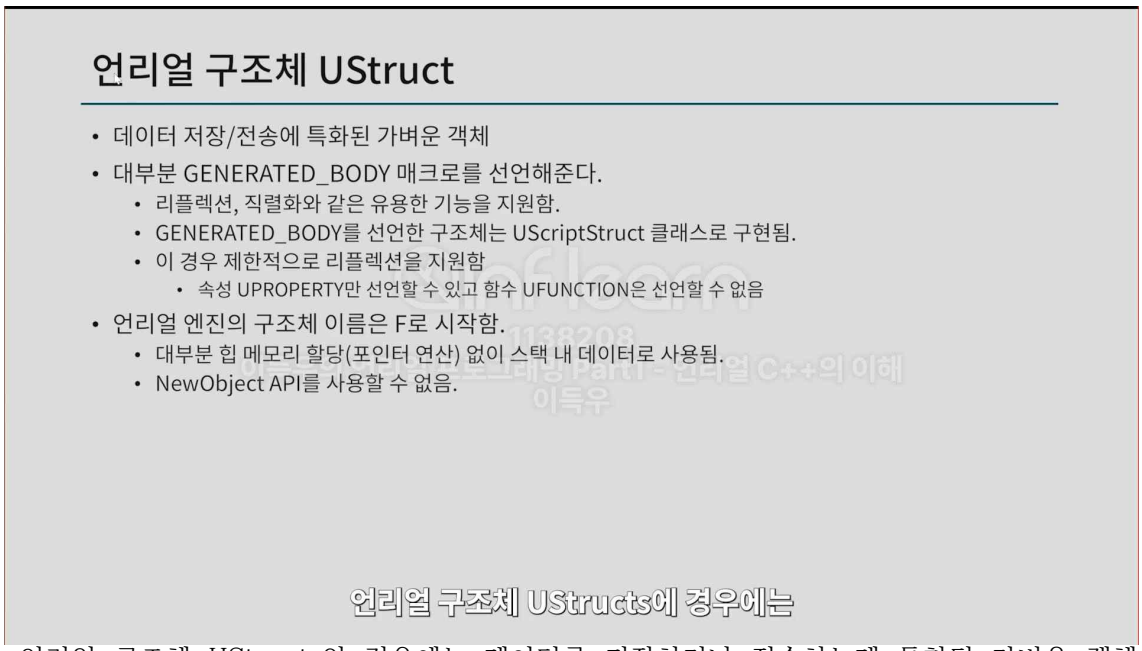
(3) 언리얼 구조체는 단순한 데이터 타입에 적합하다. 만약 보다 복잡한 인터렉션을 하기 위해서는 구조체 사용이 아닌 언리얼 오브젝트를 사용하자.

(4) 구조체 자체는 언리얼 엔진이 제공하는 굉장히 편리한 시스템에서는 제외가 된다. 즉 선언할 때 우리가 F로시작하는데 일반 객체처럼 언리얼 엔진이 취급한다는 것이다. 따라서 리플리케이션이라는 좋은 기능을 사용하지 못하지만, 구조체에 선언된 UPROPERTY() 변수들의 경우에는 언리얼 시스템의 혜택을 받기 때문에 사용이 가능하다.

(5) 언리얼 엔진에는 구조체를 위한 Make 및 Break 함수 자동 생성 기능이 있다.

(6) 에디터에서 사용하는 편리한 기능들을 언리얼 구조체 c++를 통해서 제공이 가능하다.

**언리얼 구조체 UStruct



-언리얼 구조체 UStructs의 경우에는 데이터를 저장하거나 전송하는데 특화된 가벼운 객체다.

-대부분 GENERATAED_BODY를 선언하는게 일반적이고 선언 안해도 되지만 선언 하게 되면 리플렉션이나 직렬화, 데이터 전송과 같은 유용한 기능을 지원하기 때문에 대부분의 경우 해당 매크로를 선언한다.

-GENERATAED_BODY를 선언한 구조체는 내부적으로는 UScriptStruct라는 클래스로 자동으로 구현되며 엄밀히 따지면 UStruct가 아닌 UScriptStruct클래스가 된다. 이 경우엔 제한적으로 리플렉션을 지원한다.

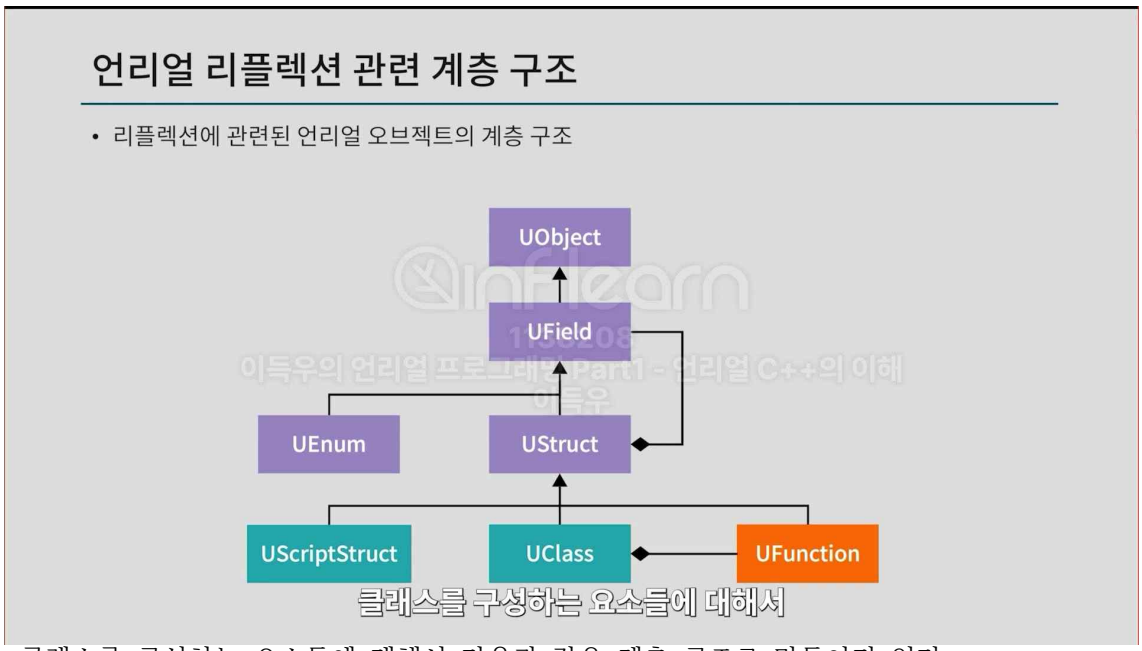
-구조체의 맴버변수에 UPROPERTY선언이 가능한데 구조체에서 함수(UFUNCTION)는 선언할 수가 없다.

-구조체 이름은 F로 시작하기 때문에 언리얼 엔진이 메모리를 관리해주지 않는다. 왜냐하면 데이터 저장과 전송에 특화된 가벼운 객체로만 사용하기 때문에 힙 메모리 할당을 하지 않고 스택 내 데이터로 사용하거나 클래스의 멤버변수로 그냥 들어가게 되는 것이 대부분의 활용 방법이다.

-언리얼 엔진도 이러한 부분 때문에 구조체에 대해서는 이런 NewObejct라는 API도 제공하지 않는다.

-이러한 구조체의 특징을 이해하려면 전체적으로 언리얼 엔진이 리플렉션 시스템을 사용하여 어떤 구조로 설계 했는지 알아보면 좋다.

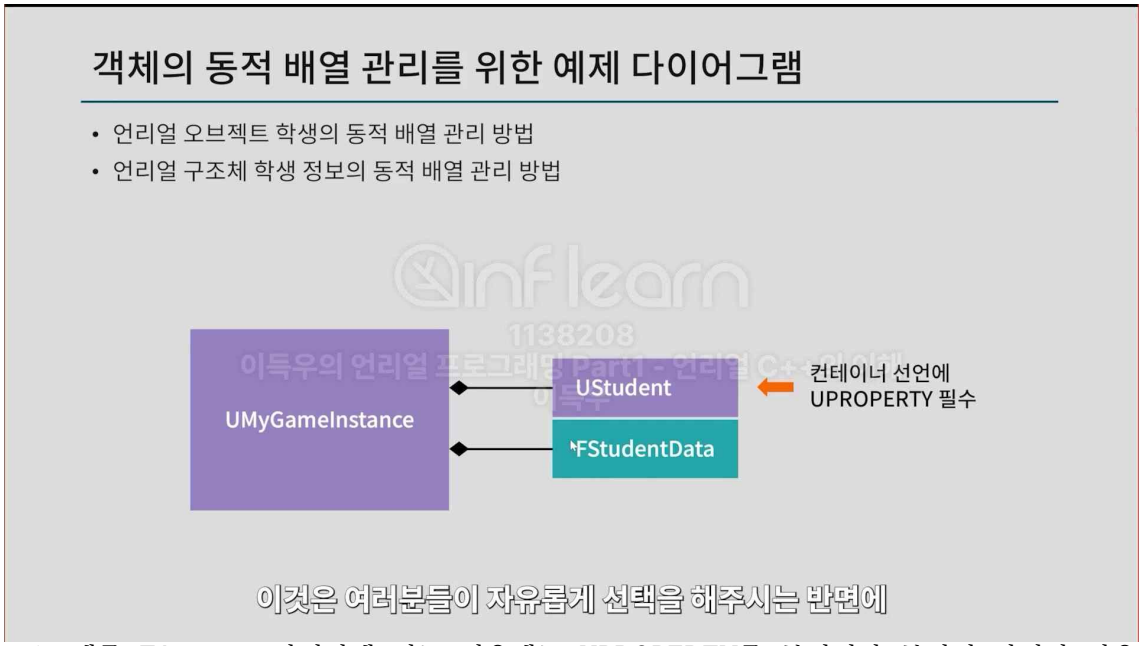
**언리얼 리플렉션 관련 계층 구조



-클래스를 구성하는 요소들에 대해서 다음과 같은 계층 구조로 만들어져 있다.
-가장 상위엔 UObject가 있고 UObject를 상속받은 UField가 있다. 하나의 멤버, 필드에 대한 값은 UField라는 클래스가 관리하도록 설정되어 있다.
-UField를 상속받아서 열거형이나 UStruct가 만들어지는데 UStruc는 다시 UField를 멤버변수로 받아 컴포지션(UField에서 UStruct로 이어지는 화살표)으로 관리하는 구조를 가지고 있다.
-UStruct를 상속받은 USctriptSturct가 있는데 GENERATED_BODY가 들어간 특별한 유형의 스트럭트(구조체) 이다.
-UStruct를 상속받은 것이 언리얼 오브젝트의 클래스를 담은 UClass가 있고 또 다시 UStruct를 상속받은 UFunction이 있다.
-UClass는 UStruct를 상속받았기 때문에 UField를 컴포지션으로 가지게 되고, UStruct나 USctriptSturct가 가지지 않은 UFunction 정보를 컴포지션으로 소유하게 된다.
-이러한 구조로 구조체와 클래스는 사용 용법과 구성이 다르다.

**실습(지난번 UnrealContainer 프로젝트를 이어서 진행함).
-우리가 UPROPERTY를 넣어도 되고 안넣어도 무방한데 UPROPERTY를 넣을때는 넣는 목적이 명확히 있어야 한다. 예를 들어 리플렉션을 통해 보여준다던지 스크립트 랭귀지인 블루프린트와 호환을 시킨다든지 또한. 언리얼 오브젝트 포인터를 멤버변수로 가진다면 UPROPERTY매크로를 넣어야 한다.

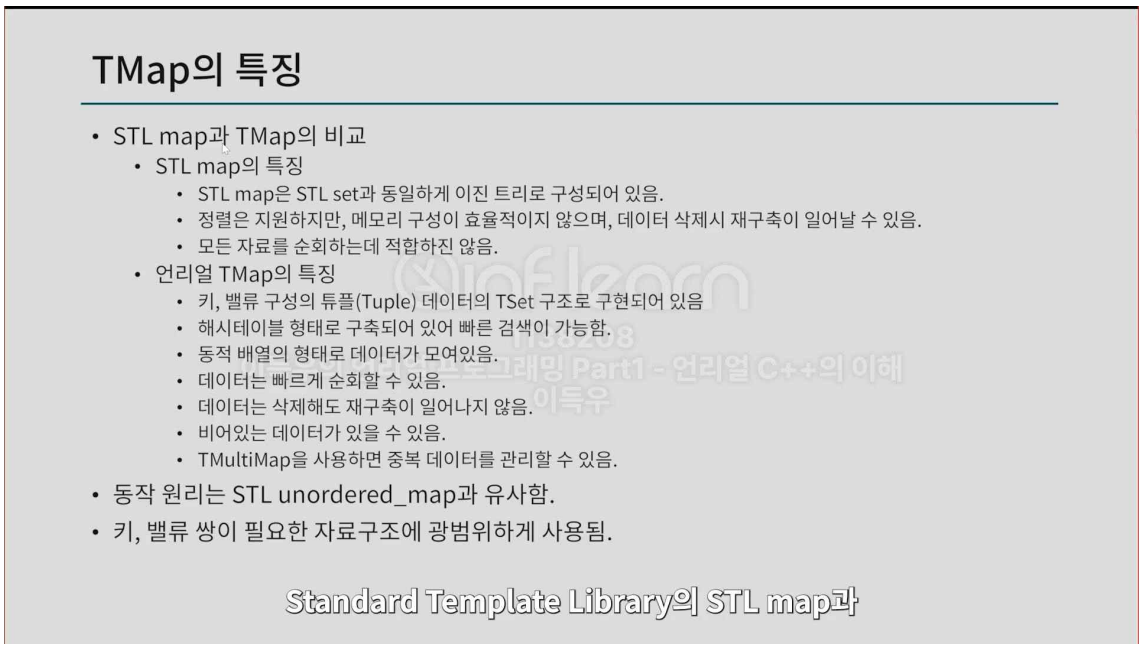
-객체의 동적 배열 관리를 위한 예제 다이어 그램



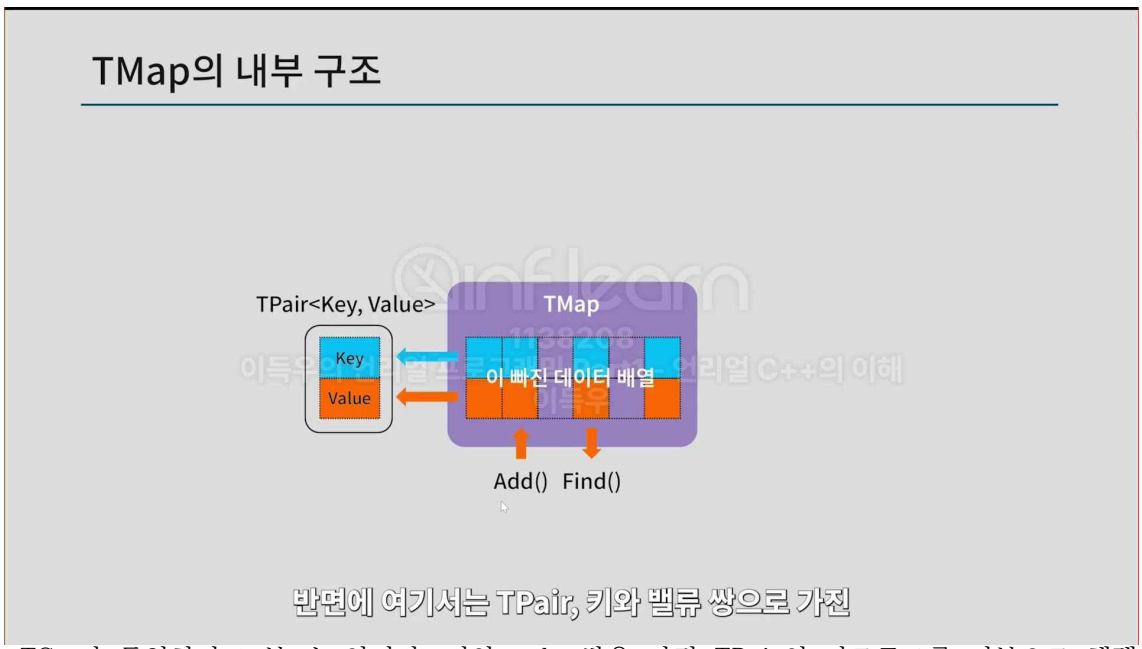
-구조체를 TArray로 관리하게 되는 경우에는 UPROPERTY를 붙이거나 붙이지 않거나 자유롭게 해준다.
-TArray에서 UStudent 언리얼 오브젝트를 관리해줄때는 반드시 컨테이너 선언에 UPROPERTY 선언을 부착해줘야 한다.

TMap의 구조와 활용

**TMap의 특징



**TMap의 내부구조



-TSet과 동일하다고 볼 수 있지만, 키와 value쌍을 까진 TPair의 자료구조를 기본으로 채택하고 있다는 점이 다르다.

**TMap 공식 문서

링크 : https://dev.epicgames.com/documentation/ko-kr/unreal-engine/map-containers-in-unreal-engine?application_version=5.1

-기본설명
(1)TMap은 TArray 다음으로 자주 사용되는 컨테이너다. TMap은 해시 기반이라 TSet과 거의 비슷하다. [성질은 동일함]

(2)TMap(중복 허용x)과 TMultiMap(중복 허용o) 두가지 유형이 있다. 따라서 중복이 들어올 때 TMap은 기존 것을 대체하고, TMultiMap은 새로 저장됨.

-'TMap'
(1) TMap은 키-값을 개별 오브젝트처럼 처리한다.
(2) 내부적으로 TPair< KeyType, ElementType >이러한 오브젝트가 있다고 볼 순 있지만, 실제로는 KeyType과 ElementType 두 가지만 사용해서 TMap을 선언해 준다. 그리고 모든 요소는 같은 두 개의 타입을 가진 TPair로 동질하게 구성이 되어 있어 빠르게 접근이 가능하다.
(3)TSet과 동일 해시 컨테이너라서 기본적으로 키 유형은 GetTypeHash로 해시값을 뽑아낼 수 있는 함수를 제공해줘야 한다. 또한 '이퀄리티 Operator=='도 함께 제공 되어야 한다.
==> 이것은 TSet에서도 설명함.
(4) 별도의 옵션 얼로케이터를 받기도 한다는 것은 모든 자료구조가 지원하는 옵션이기도 함.
(5) 템플릿 파라미터로 KeyFuncs 받는다.[하단에 자세히 설명함]
(6) TArray와 달리 TSet의 성격이라 이빨 빠진 동적 배열이란 것은 동일하다.

-맵 만들고 채우기
(1) 선언 : TMap<int32, FString> FruitMap;
보통 이렇게 선언 하고 TPair로 선언하는 경우는 거의 없다. 그리고 위처럼 선언하면 비어있는 자료구조가 만들어지고 표준 힙 할당이 되어가지고, 비교를 수행하게 된다.
(2) 대부분의 작업은 키 값을 사용해서 진행하도록 TMap이 설계가 되어 있다.
(3)Add함수는 동일하다
(4)TMultiMap의 경우는 중복 키를 시도하면 어떤 일이 벌어지는지 볼 수 있다.[예제를 통해 살펴볼거임]
(5)TArray처럼 Emplace 사용하는 것도 동일하다.(TSet도 동일함)
(6) 전반적인 함수 구성이 Set과 유사하다고 볼 수 있고, 마치 집합처럼 우리가 다른 TMap에 있는 요소들을 추가해서 넣을 수가 있다.
(7) 에디터와 연동해서 UPROPERTY키워드를 추가하면 에디터에서도 쉽게 편집 가능한 기능을 제공한다.

-반복처리
(1) ranged for문과 두가지 종류의 이터레이터를 제공한다.
(2) 이터레이터를 순회할 때는 key와 value타입을 가진 이터레이터로 순회하기 때문에 접근할땐 key와 value값으로 데이터를 가져와야 한다.

-쿼리
(1)TSet과 거의 성격이 유사하다.
(2)어떤 맵에 데이터를 찾을려면 contains로 확인하고 해당 인덱서를 가져와야 하는데 내부 구조상 동일한 작업을 두 번 하는거라 Find함수로 Null포인터를 체크하는 걸로 한번으로 줄일 수 있다는 것도 동일하다.
(3)일반적으로 맵 구조의 특성상 인덱스 오퍼레이터에 키 값을 넣었을 때 해당 키에 대한 자료가 없으면 새로운 데이터를 생성하는 매커니즘이 있다. 그 매커니즘을 쓰고 싶다면 FindOrAdd함수를 사용하면 된다. 하지만 강의자는 이 방법보단 Find면 Find, Add면 Add로 명확한 함수를 사용하기를 추천한다. 기존의 STL map에 익숙한 사람들을 위해 제공하는 함수일 것이라고 강의자는 생각한다.
(4)FindKey 함수는 value 값을 가지고 키를 역조회 하는 것인데 키는 해시값이 들어가 있지만 value값은 해시 값이 없기 때문에 TArray와 똑같이 최악의 경우 모든 요소를 돌게 된다. 따라서 성능은 안좋기 때문에 편리하게 사용할 수 있는 함수라고 생각하자.
(5)키에 대한 정보나 Value에 대한 정보를 TArray로 가져올 수 있는 함수가 있다.(사이트 참고)

-제거
(1)키가 있는지 없는지 확인하고 제거하는 함수도 있다.

-정렬
(1) 읽어보자

-연산자
(1) 기본적으로는 복사 생성을 통해 연산지 진행 되고, MoveTemp를 통해 기존에 있는 것을 이주가 가능하다.

-슬랙
(1) Set과 동일하고, 중간에 이빨 빠진 여유분이 생기거나 끝부분에 여유분이 생길수 있고 끄부분의 슬랙을 지우고 싶으면 Shrink 함수를 사용하면 된다.

-KeyFuncs
(1)KeyFuncs는 기본적으로 해시테이블 구조로 관리한다. 우리가 커스텀 데이터 자료구조를 만들 때는 이 구조체에 대한 해시값이 무엇인지 지정해줘야 한다. 이것을 위해서 이퀄리티오퍼레이터(operator==)와 우리가 선언한 구조체에 GetTypeHash를 선언해줘야 해시값을 추출할 수가 있는데 대부분은 TMap으로 사용하면 된다.

(2) 특정한 경우가 있는데 기본적으로 멤버변수에 UniqueID가 생성되도록 설계가 되어 있다. 공식 사이트의 예시를 보면 생성 할때 NewGuid를 통해 계속해서 다른 Unique 값을 만들어 지도록 설계를 하였다. 이 경우에는 이퀄리티 오퍼레이터를 어떻게 처리할지에 대한 딜레마가 빠지게 된다.

(3) (2)번의 경우 someFloat 값이 같다고 할지라도 어쨌든 UniqueID가 서로 다르면 이 구조체는 사실 같다고 볼 수 없다. 우리가 해시 값을 생성할땐someFloat을 가지고 생성을 할 수도 있지만, 실제적으로 내용물은 이 구조체가 가지고 있는 어떻게 활용 되는가에 따라서 ID는 다르지만 같은 구조체라고도 볼 수 있다. 이런 애메한 상황이 발생한다.

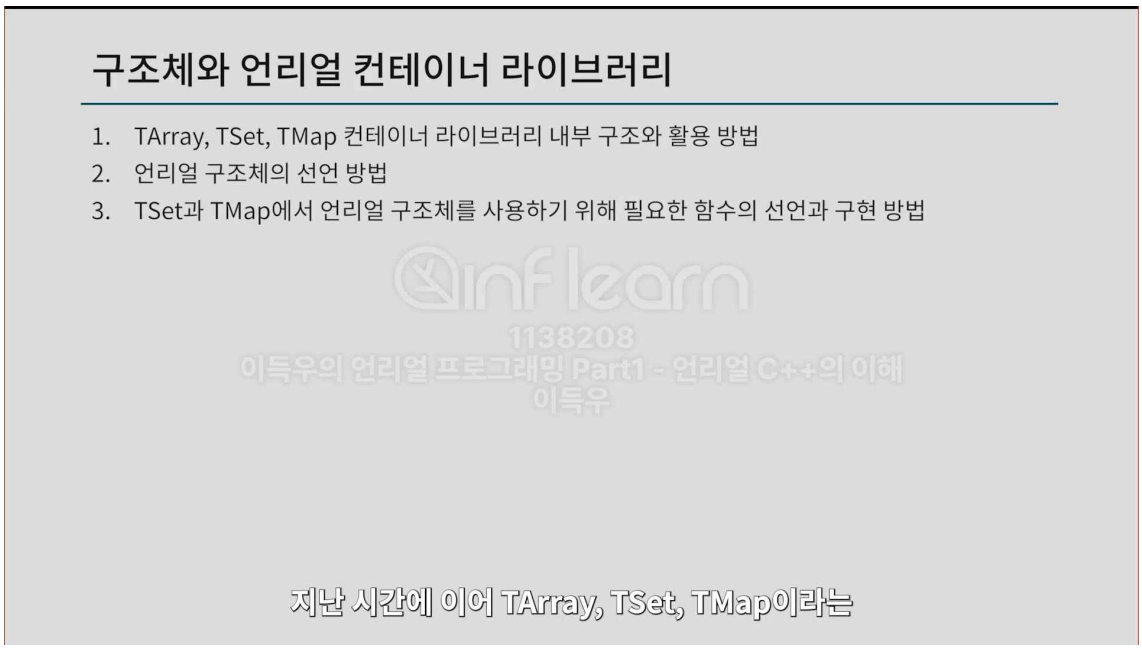
(4) (3)처럼 딜레마에 빠진 경우 해시값이라던지 같음에 대해 추가적으로 정의할 필요가 있는데 그런 경우 MapKetFuncs라는 스트럭트를 정의해주는데 이것을 어떻게 선언하고 사용할지에 대해서 어떤 함수를 구현할지에 대해서 쭉 구현을 해 주었다. 즉 이런 상황이 발생했을 때 이 문서를 참고해주면 된다. 그래서 여기에 지정된 함수들을 가이드라인에 맞춰서 구현해주면 된다.

-기타
(1) 메모리 통계에 대한 함수도 제공한다. 사이트 참고하자.

**자료구조의 시간 복잡도 비교

-TMap은 키와 벨류쌍을 가지고 있기 때문에 굉장히 편리하게 자료 검색이 가능하다.
-만약 키 벨류가 아닌 중복 처리 하지 않는 집한은 TSet을 사용하자.

**정리


구조체와 언리얼 컨테이너 라이브러리
1. TArray, TSet, TMap 컨테이너 라이브러리 내부 구조와 활용 방법
2. 언리얼 구조체의 선언 방법
3. TSet과 TMap에서 언리얼 구조체를 사용하기 위해 필요한 함수의 선언과 구현 방법
inflearn
1138208
이득우의 언리얼 프로그래밍 Part1 - 언리얼 C++의 이해
이득우
지난 시간에 이어 TArray, TSet, TMap이라는