


1단계_11. UCL #2 구조체와 Map

📅 수강일	@2023/04/03
➦ 이름	 전영재
🔍 멘토	Min-Kang Song, 현웅 최
👥 멘티	
✨ 작성 상태	Not started
📁 단계	
☑ 강의 시청 여부	<input checked="" type="checkbox"/>
☑ 이수 여부	<input type="checkbox"/>

Contents



[강의 제목](#)

[0n. UCL #2 구조체와 Map 강의 과제](#)

강의 제목



강의 목표

- 언리얼 구조체의 선언과 특징 이해
- UCL 중 TMap의 내부구조 이해
- TArray, TSet, TMap의 장단점을 파악하고 알맞게 활용하는 방법 학습

언리얼 구조체

Document보기

구조체(UStruct) 구성 방법과 커스터마이징 방법을 알려주겠다.

구현방법

구조체를 정의하려는 헤더(.h)를 열고, C++구조체 앞에 `USTRUCT` 매크로를 추가한다.

구조체 상단에 `GENERATED_BODY`를 추가하고, 필요한 멤버변수에 `UPROPERTY`를 태그하면 된다.

`USTRUCT(BlueprintType)` //에디터에서도 사용할 수 있게하는 메타데이터

구조체에 메모리관리를 위해서는 `UPROPERTY`를 꼭 쓰는게 좋다.

사실 C++문법상 구조체와 클래스는 큰차이가 없는데(기본접근자가 public이냐 private냐 정돈 데..) 언리얼에선 `UObject`와 `UStruct`는 사용용도가 완전히 다르다. **`UStruct`는 단순한 데이터 타입에 적합하므로, 복잡한 인터랙션을 위해선 차라리 `UObject`를 써라.**

`UStruct`는 리플리케이션용으로 간주되지 않는다. 그렇기에 리플리케이션 쓰고싶으면 `UPROPERTY`를 붙여야된다.

구조체를 위한 `Make`, `Break` 등 여러 함수자동생성 기능을 제공하는데, 후에 보여주겠다.

언리얼 구조체 `UStruct`

데이터 저장/전송에 특화된 가벼운 객체

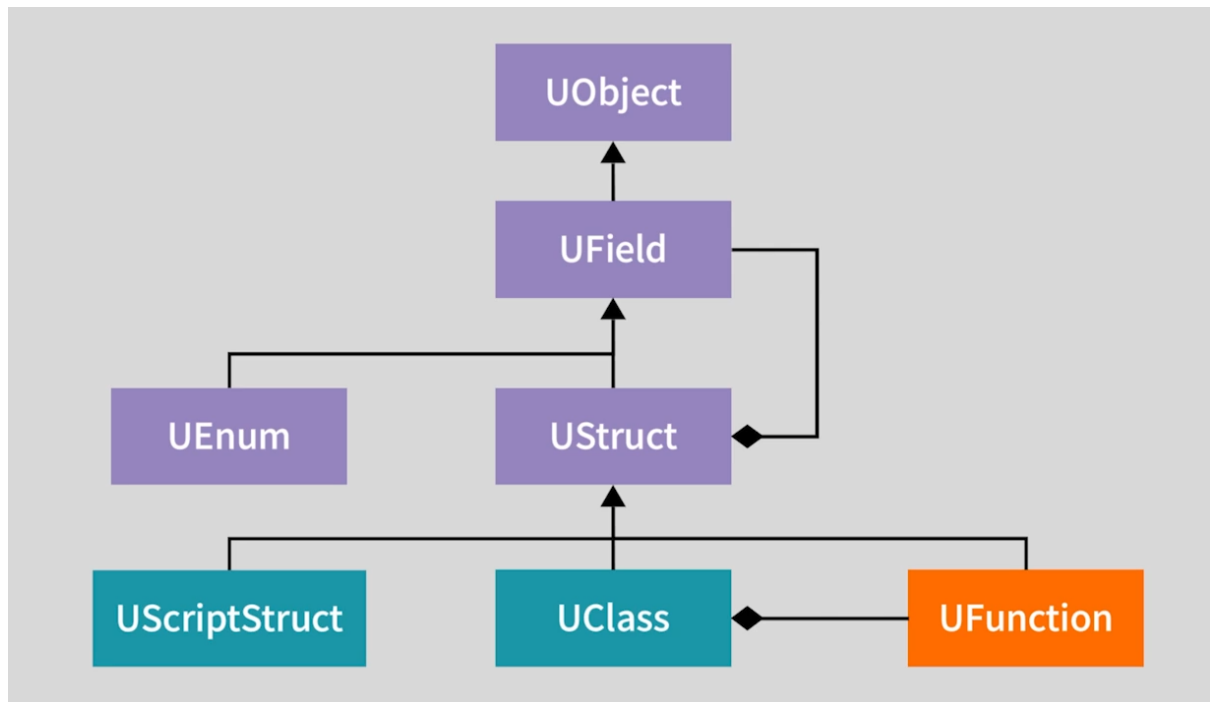
대부분 `GENERATED_BODY`매크로를 선언함.

- 하면 리플렉션, 직렬화 같은 기능 지원
- 엄밀히 말하면 `generated`를 선언한 구조체는 `UScriptStruct`클래스로 구현된다.
- 이 경우 제한적인 리플렉션을 지원함. (`UPROPERTY`는 되고 `UFUNCTION`은 안됨)

이렇게 만들어진 구조체는 F라는 접두사를 얻음 = 힙할당(포인터연산) 없이 스택내 데이터로 사용됨.

이런 구조체 특징을 이해하려면, 리플렉션을 위해 언리얼이 무슨 구조를 채택했는지 이해하는게 좋다.

언리얼 리플렉션 관련 계층 구조



//UClass는 UStruct를 상속받기에, UField를 컴포지션으로 갖고, 그렇기에 UScriptStruct나 UStruct와 달리 UFunction을 가질수 있따한다....무슨소리지? 아무튼 구조체와 클래스는 사용 용법이 다르다.

실습

구조체를 별도의 헤더파일을 만들지않고, GameInstance헤더에 넣어서 만들수도 있다.

```

MyGameInstance.h

USTRUCT()
struct FStudentData
{
    GENERATED_BODY()
    //기본적으로 Struct는 public접근자를 갖기에 따로 안써줘도되더라.
    FStudentData()
    {
        ...
    }

    //구조체는 UObject가 아니기에, New api를 통해 생성되지 않으므로, 인자를 가진 생성자를 만들어 자유롭게 사용하겠다.
    FStudentData(FString InName, int32 InOrder) : Name(InName), Order(InOrder) {}

    UPROPERTY() //안써도됨
    FString Name;
    UPROPERTY()
    int32 Order
}
  
```

//UMyGameInstance:: 이걸 안붙여줘도 cpp내에서 사용할 수 있는 함수는 만들 수 있더라. 그냥 클래스의 멤버함수가 아닐뿐임

TArray에 300개나 되는 이름 데이터를 집어넣는방법?

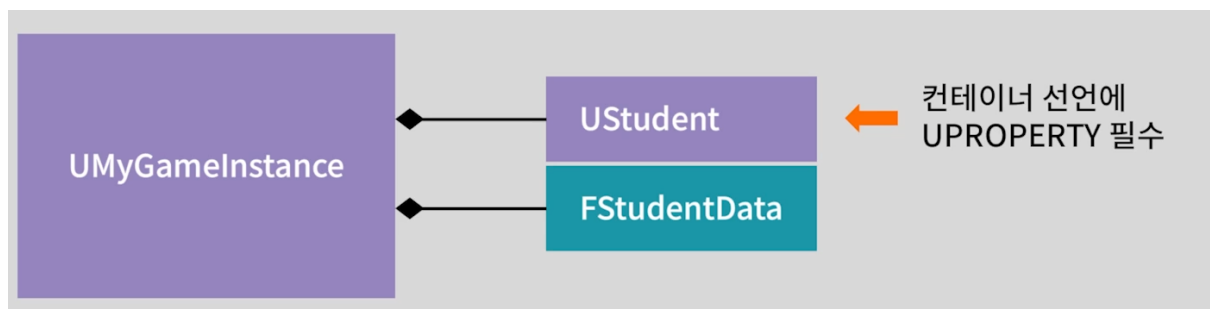
→ for문 돌리는게 아니라 Algo::Transform을 사용한다.

오브젝트를 만들고 이것을 TArray로 관리하는 실습도 해보자.

UPROPERTY() //메모리(포인터)를 관리할때는 무조건 UPROPERTY()를 붙여줘야한다.

TArray<TObjectPtr<class UStudent>> Students; //전방선언 말고 약포인터 해주기

- 언리얼 오브젝트 학생의 동적 배열 관리 방법
- 언리얼 구조체 학생 정보의 동적 배열 관리 방법



구조체는 단순히 데이터이기에 UPROPERTY를 안붙여도 되지만, UStudent는 오브젝트이기에 반드시 UPROPERTY해줘라 이말임

TMap의 구조와 활용

STL Map과 TMap의 비교

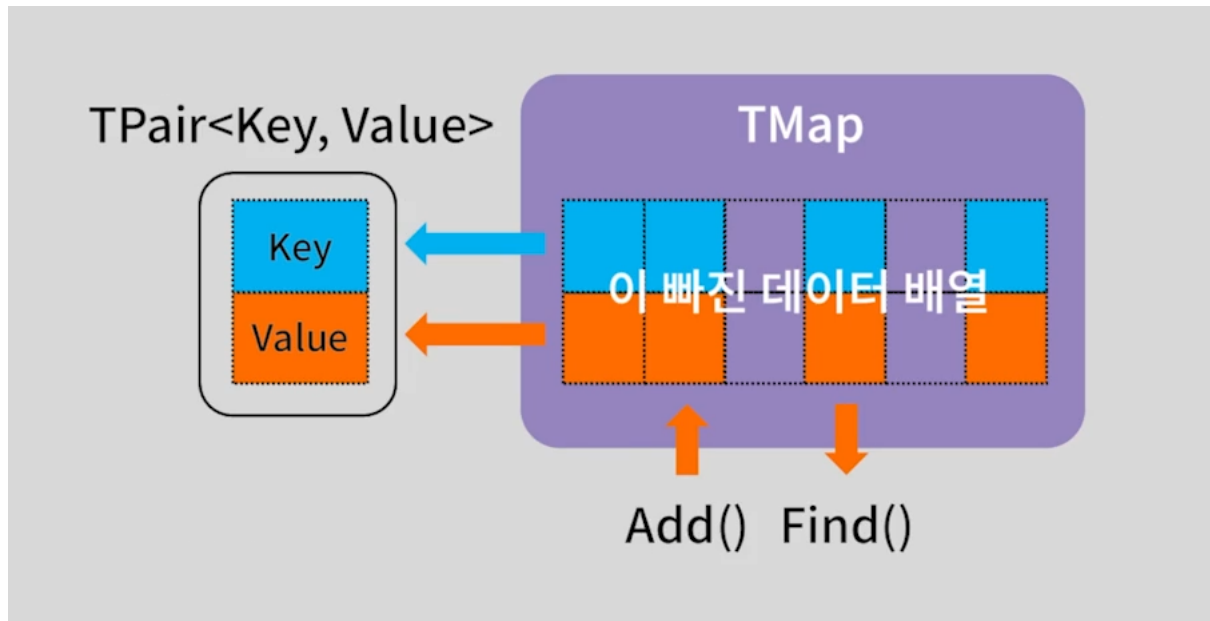
STL

- 이진트리 구성
- 정렬은 지원하지만 메모리 구성이 효율적이지 않음. 삭제시 재구축
- 자료 순회에 적합 X

TMap

- 키-밸류 구성의 Tuple데이터의 TSet구조임
- 그래서 내부구조랑 메카니즘이 TSet랑 똑같음

- 해시테이블 - 동적배열 - 빠른순회 - 빠른검색- 삭제후 재구축안함 - 비어있는데이터
- TMultiMap을 사용하면 중복데이터 관리할 수 있다.
- 오히려 STL unordered_map과 유사함



Document 보기

TArray다음으로 많이 쓰인다.

TMap에 중복된 키저장을 넣으면 기존값이 대체되고, TMultiMap에 넣으면 새로 추가한다.

키-값 쌍을 마치 오브젝트처럼 Map의 엘리먼트 유형으로 정의한다.

해시 컨테이너라 GetTypeHash 함수를 통해 키유형에서 해시를 뽑을 해시표를 만들어야한다.

KeyFuncs는 아래에 추가내용이 있어 거기서 다시설명

생성방법

```
TMap<int32, FString> FruitMap;
```

Add대신 Emplace 쓸수있다.

그냥 키-밸류 인 TSet인지라 진짜 거의 똑같다.

FindKey() 함수는 밸류를 통해 키를 역조회 하는방법임. 그런데 키에 해시가 있는거지 밸류에 해시가 있는게 아니라, 재수없으면 전체를 순회해야 될 수도 있다. 즉, 성능 안좋다. 어쩔수없으면 쓰자.

대부분 document에 이런 함수들이 있더라 그리고 TSet이랑 비슷하더라 이런내용들임.

KeyFuncs

이게 좀 중요하다. 저번에 TSet에서 안다뤘던 그것임.

해시테이블 구조이기에 커스텀 구조체의 'operator =='과 'GetTypeHash'를 만들어줘야 한다.

특이한 경우?

구조체 생성시 키값을 uniqueID를 만들게 설정한 경우, ==는 어떻게 만들것이나?

아무리 밸류가 같아도 키값이 다르면 ==를 사용해 정의하기 어렵기에, 이것을 위한 특수조치가 필요하다라는 것.

이를 위해서는 MapKeyFuncs를 구현해야함. 이걸 해결하려면 document를 참고해라. 사실 이런 경우가 얼마나 있을진 모르겠다.

실습

TMap 만들고, 기존 TArray로 만들었던 데이터를 TMap에 넣는다. 넣을때는 Algo::Transform을 사용해 람다 만들어줘서 넣었다.(튜플로 넣어야 되니까 람다 만들때 return값을 2개로 만들었다.)

이걸로 order-name이 아니라 name-order순의 TMap도 하나 만들어봤고, 중복키를 지원하는 TMultiMap도 만들어봤다. 그후 Num을 사용해 몇개가 들어갔는지 보기도 했고, TargetName을 만들어 해당 키를 갖는 order가 몇개인지 찾아보기도 했다.

좀 중요한 실습!

FStudentData를 TSet에서 쓰려면 어떤 작업을 거쳐야 하는지 살펴보자. (커스텀 구조체 사용하기)

```
TSet<FStudentData> StudentSet;
/** 그냥 하면 GetTypeHash가 없다고 에러가 엄청 나온다. 해결하려면 operator와 GetTypeHash를 만들어야함.*/
for (int32 ix=1; ix<=StudentNum; ++ix)
{
    StudentSet.Emplace(FStudentData(MakeRandomName(),ix));
}
```

해결하려면

FStudentData 구조체에 operator==와 GetTypeHash()를 추가한다.

```
struct FStudentData
{
    GENERATED_BODY()
    ...
    bool operator==(const FStudentData* InOther) const
    {
        return Order == InOther.Order; //여길 내맘대로 커스텀
    }

    //전역함수로 만들던가 friend로 만들어도 깔끔하다.
    friend FORCEINLINE uint32 GetTypeHash(const FStudentData& InStudentData)
    {
        return GetTypeHash(InStudentData.Order); //order를 곧 해쉬값으로 쓰겠다.
    }
}
```

자료구조의 시간복잡도 비교

	TArray	TSet	TMap	TMultiMap
접근	O(1)	O(1)	O(1)	O(1)
검색	O(N)	O(1)	O(1)	O(1)
삽입	O(N)	O(1)	O(1)	O(1)
삭제	O(N)	O(1)	O(1)	O(1)

빈틈없는 메모리
가장 높은 접근성
가장 높은 순회성

빠른 중복 감지

중복 불허
키 밸류 관리

중복 허용
키 밸류 관리

TArray - TMap - TSet 순으로 많이 쓰인다.

Summary

- TArray, TSet, TMap의 내부구조와 활용 방법 학습

- 언리얼 구조체의 선언 방법
- TSet과 TMap에서 언리얼 구조체를 사용하기 위해 필요한 함수의 선언과 구현 방법 학습

0n. UCL #2 구조체와 Map 강의 과제

? Q1. 현재 프로젝트에서 구조체, Array, Set, Map, MultiMap이 유용하게 활용될 수 있는 사례를 생각해보시오.

? Q2.

? Q3.



Reference