1단계_10. UCL #1 Array와 Set

수강일	@2023/04/03
이름	전영재
멘토	Min-Kang Song,현웅 최
작성 상태	In progress
단계	1단계
강의 시청 여부	☑

Contents

▼

UCL #1 Array와 Set

💡 **강의 목표**

- 언리얼 대표 컨테이너 라이브러리 TArray, TSet의 내부 구조 이해
- 각 컨테이너 라이브러리의 장단점을 파악하고, 알맞게 활용하는 방법의 학습

---

[1] 언리얼 컨테이너 라이브러리

언리얼엔진이 자체 제작해 제공하는 **자료구조 라이브러리로 UCL**(Unreal Container Libarary)이라고도 한다.

- 유용한 라이브러리는 TArray, TMap, TSet 3가지가 있다

> 💡 여기서 접두사 T는 Template 라이브러리를 의미한다.

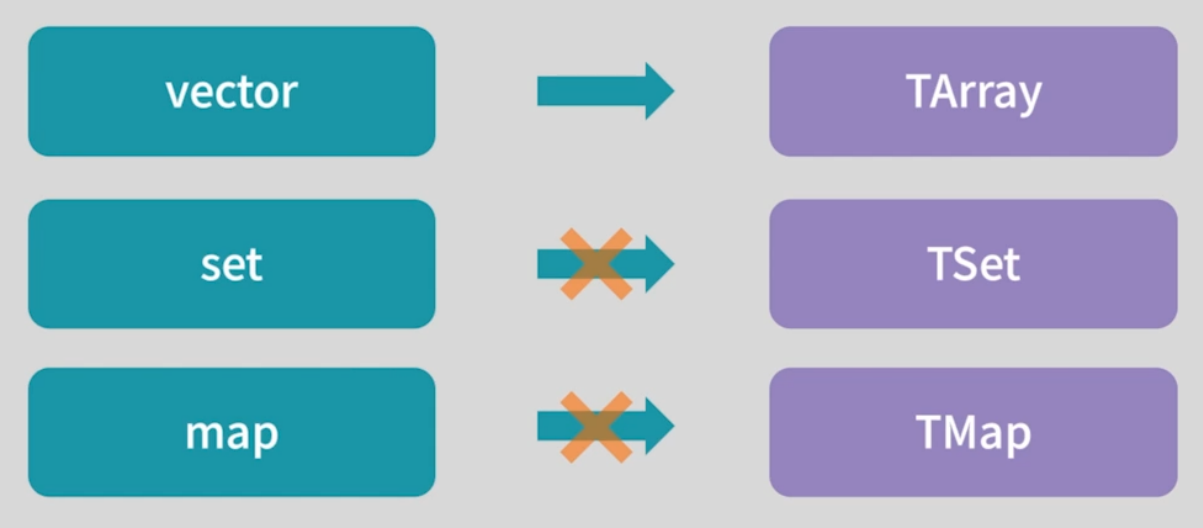
C++STL과 차이점

C++STL은 범용적이고 표준이기에 호환성이 높다. 그러나 너무 **많은 기능이 있기에 컴파일 시간이 오래**걸린다.

⇒ UCL은 **언리얼에 특화**되어 있어 가볍고, 게임제작에 최적화 되어있으며, 안정적이다.

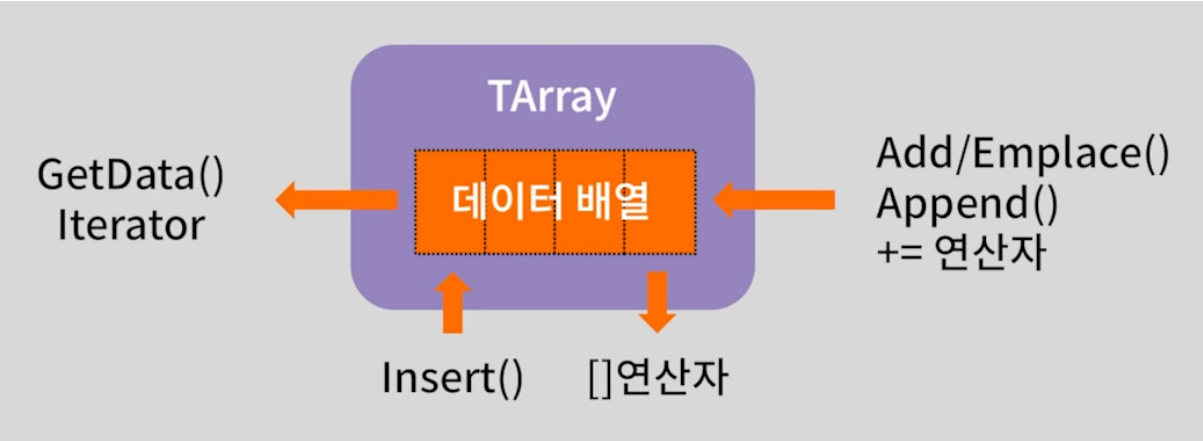
UCL의 주요 컨테이너 라이브러리

TArray	오브젝트를 순서대로 담아 관리
TSet	중복되지 않는 요소로 구성된 집합을 만드는데 용이
TMap	키, 밸류 조합의 레코드를 관리



Set와 Map은 이름과 용도는 유사하지만, 내부적으로는 다르게 구현되어 있다.

[2] TArray 구조와 활용



insert와 remove는 비용이 많이들지만, 데이터가 같은유형이기에 **[]로 중간선택**은 빠르게 작동한다.

맘대로 크기를 변경할 수 있는 **가변배열**(Dynamic Array) 자료구조로 STL의 **Vector**와 동작원리가 유사하다.

가장 간단하면서도 빠르기에 가장 자주 쓰인다.

TAray의 유형은 두 프로퍼티로 정의되는데 '엘리먼트 타입'과 '얼로케이터'다.

> 얼로케이터는 메모리를 어떻게 관리하는지에 대한 설정인데, 자주 생략되는데, 특수하게 메모리 관리가 필요할 때 직접 작성해서 사용한다.

<장점>

- 데이터가 순차적으로 모여있기에 메모리를 효과적으로 사용할 수 있고 **캐시 효율**이 높다.
    - 컴퓨터 사양이 발전하면서, **캐시 지역성**으로 인항 성능향상의 중요성이 높아졌기에, 게임제작에서 위 특징은 어마어마한 장점이라 할수 있다.
- 임의데이터 **접근이 빠르고**, 요소의 **고속순회**가 가능하다.

<단점>

- 맨끝에 데이터를 추가하기는 쉽지만, **중간 삽입/삭제는 비용**이 많이 든다.

⇒ 데이터가 많아질 수록 **검색, 삭제, 수정 작업이 느려**지므로, 많은 데이터의 검색 작업이 빈번하게 일어난다면 대신 **TSet**을 사용하는 것을 권장한다.

정의 방법

```
TArray<int32> Array;

//채우기는 여러 방식으로 가능하다.
Array.Init(10, 5);
//Array == [10, 10, 10, 10, 10]
```

- `Add` / `Push` : 추가할 데이터를 밖에서 생성하고 TArray안에 복사해서 붙여넣는다.
- `Emplace` : TArray안에서 즉시 요소를 생성한다. 위보다 좀 더 효율적이다.

> 💡 그러니 사소한건 `Add`, 그외에는 `Emplace`를 쓰자. `Add`는 가독성이 좋아서 사용된다.
>
> 절대 `Emplace`가 `Add`보다 효율이 떨어지는 일은 없다.

- `Append` : 한번에 여러 엘리먼트를 넣을때 사용한다.
- `AddUnique` : 컨테이너에 동일한 엘리먼트가 존재하지 않으면(검색) 추가한다. 별로 효율적이진 않기에. 차라리 `TSet`를 사용하는 것이 좋다.
- `Insert` : 해당 인덱스에 집어넣기
- `SetNum` : 배열의 크기를 인위적으로 조정가능

💡 만약 엘리먼트가 제거된다면, 그 뒤의 엘리먼트가 **낮은 인덱스로 정리**된다. 그렇기에 배열에는 절대 구멍이 생기지 않는다.

연산자

- 할당연산자(`=`)로 값을 넣을수있는데, 이때 **값을 복사**해서 넣는다는게 중요하다. 즉, Emplace가 아니라 Add함수처럼 작동한다는 것.

```
auto ValArr2 = ValArr1;
//이러면 ValArr1이 복사되서 2에게 들어간다는것
```

아예 이주하고 싶다면 `MoveTemp()` 함수를 사용하자

- 비교연산자(`==`)의 경우 엘리먼트 수와 순서 모두 같아야 `true`를 반환한다. 이때 대소문자는 구분하지 않는다.

슬랙

메모리가 확보되야, 즉 정보를 넣을 자리가 있어야 넣을수있는건데 배열의 크기 변경을 하나하나씩 하면 **메모리 오버헤드**가 생긴다. 그렇기에 **요청한것보다 좀더 널널하게 데이터를 잡는다**.

⇒ 즉, **여유분(slack)**을 잡는다는것. Vector의 큰 특징

💡 그러나 여유분도 너무 많이 남는다면 메모리 낭비가 될 것이다. TArray에는 이조차 깔끔하게 정리해주는 기능이 존재하며 공식다큐멘트에 이에 대한 설명이 있다.

TArray: 언리얼 엔진의 배열

Ⓤ https://docs.unrealengine.com/4.27/ko/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/TArrays/

원시메모리

TArray는 궁극적으로 할당된 **메모리를 둘러싼 포장 단위**일 뿐이다. 그렇기에 내부의 데이터를 다룰때에는 **직접 메모리에 접근**하는 것이 유용할수도 있으며 이를 위한 API들을 설명한

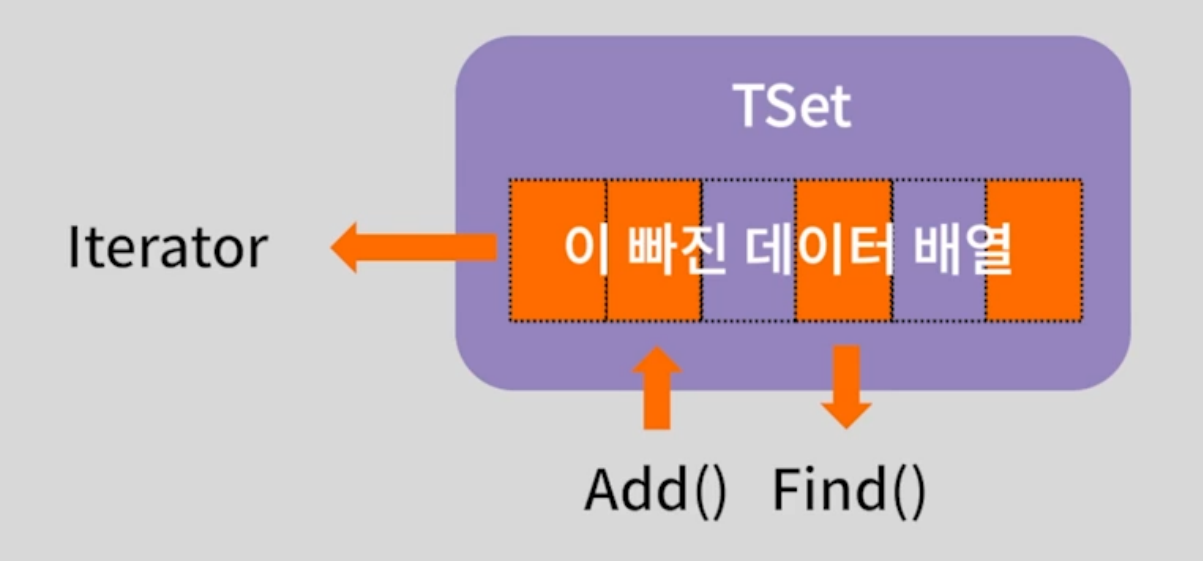
다.

> `GetData()`를 통해 포인터를 얻어와서 C스타일로 다루는 것이다.

---

[3] TSet 구조와 활용

STL Set 와 언리얼TSet의 특징



- **STL**은 내부적으로 **이진트리**로 구현되어 **정렬을 지원**함. 그런데 정렬때문에 **효율적이진 않다**(삭제하면 **균형**맞추려고 재구축하는것처럼). 그리고 트리라서 **모든자료**를 순회하는데 적합하지 않다.
- 그러나 **언리얼 TSet**는 **해시테이블** 형태로 키데이터가 구축되어서 **빠르게 검색 가능**함.
  - 동적배열 형태로 데이터가 모여있어 빠른순회가 가능하고, 데이터를 삭제해도 재구축이 일어나지 않는다.
  - 또한 Map과도 차이점이 있는데, 키-밸류의 쌍이 아니라, 데이터값(value) 자체를 키로 사용한다. 즉, 밸류를 키로 사용하는 해시테이블이다.
  - 엘리먼트 추가, 검색, 제거가 진짜 매우 빠르다. 단, 중복키는 지원하지 않는다.

> ⇒ TSet은 **중복 없는 데이터 집합**을 구축하는데 유용하며, **순서가 중요치 않은** 상황에 **고유 엘리먼트**를 저장하는데 사용하는 **고속 컨테이너 클래스**다.

💡 STL set와 언리얼 TSet은 활용방법이 다르기에 착각해선 안된다.
(차라리 unordered_set과 유사하다.)

TSet 생성 및 채우기

```
TSet<FString> FruitSet; //단순하게 타입을 지정해주는 것만으로 생성이 가능하다.
```

`TArray`와 유사하게, 선언만 한다면 비어있는 자료구조가 생성되며 `Add` / `Emplace`로 채울 수 있다.

데이터를 넣으면 `GetTypeHash`를 통해서 **해싱**을 진행하는데, `FString`, `int32` 같이 기본적으로 제공하는 타입은 그런 해싱표가 준비되어있다. 그러나 우리가 만든 클래스를 넣으려면 **직접 해싱표를 만들어**줘야 된다.

쿼리

- `Contains`로 자료가 있는지 확인이 가능한데, `TArray`와 다르게 **해시테이블**로 되어있어서 찾는데 훨씬 빠르다.
    - 내부적으로 해시테이블 ID는 `FSetElementId` 구조체로 저장되는데, 이걸 잘 안다면 `FSetElementId` 를 직접 지정해서 찾을 수도 있다.
- `Find` : 세트에 키 포함여부를 확인할때 `contains`를 쓰고 `operator[]`를 써도된다. 하지만 이 경우, 2가지 과정을 하기에 비효율적이므로 `Find`함수를 쓰는게 좋다.

💡 `TSet`도 동적배열 구조라서 `TArray`로 반환할 수 있다. `TArray`는 빈틈이 없기때문에 `TSet`의 빈틈이 사라진 모습을 얻을수있다.
⇒빈틈을 없애준다는 것에 집중하면 다양하게 활용할 수 도 있겠다!

제거

내부순서가 안정적이지 않다한 것처럼, `Remove`함수에 인덱스를 붙여 제거하기 힘들다. (할 수는 있다.)

```
FruitSet.Remove(3); //3번 인덱스에 머가 들어있을지 알기어렵다. 그래서 잘못지울수도있다.
```

슬랙

여유분 메모리를 말하는건데, TSet에 중간중간 **빈틈**이 있다고 했던것이 이것에 해당한다.

그래서 앞,뒤,중간 **어디든** 슬랙이 있을수 있다. 그리고 그값은 invalid로 할당된다.

- `Shrink` : 뒤쪽의 슬랙만 제거한다.

DefaultKeyFuncs

커스텀 구조체로 TSet만드려면, **'==오퍼레이터'**랑 **'해당 타입에 대한 GetTypeHash함수'** 두가지를 구현해야한다.

다음 강좌에서 set과map에서 직접 함수를 사용하면서, 어떻게 선언해야 하는지 보여주겠다.

> 💡 그런데, 이걸 오버로드 하지 않을 거면 document에 나온 **DefaultKeyFuncs을 보면된다.**
>
> (정말 특이한 경우라 사용할진 모르겠다.)

---

[4] TArray와 TSet의 시간복잡도 비교

지금까지 계속 **TArray는 메모리 효율성 및 접근/순회성능**을 강조하였고, **TSet는 빠른 속도와 중복비허용**을 강조해왔다.

시간복잡도의 비교를 살펴보고 얼마나 차이가 나는지 확인해보자.

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

📌 **Summary**

- TArry, TSet의 내부구조와 특징

TArray	빈틈없는 메모리, 최고의접근/순회 성능
TSet	모든 경우에 빠름, 중복을 허용하지 않는다.

0n. UCL #1 Array와 Set 강의 과제

❓ **Q1. 자료구조에서 해시테이블이 어떻게 동작하는지 설명하고, Bucket의 개념에 대해 설명하시오.**