**STL 보고서**

**STL 월화 23/56**

**학번:** **2014182011**

**이름:**  **김태화**

**목 차**

1. **프로그램에 대한 설명......................................... 1**

1) 조작 방법

2) 메뉴

3) 게임 목표

1. **과제 해결방법.................................................. 2**

1) 리플레이를 만들 게임 수정

2) 컨테이너와 클래스 설계

3) 파일 입출력

**3. 과제를 마치고 느낀 점 ................................. 3**

1) Unordered\_Multimap은 Hash Table로 구성되어 저장하는

순서가 정해져 있지 않다.

2) Iterator를 사용하면서 루프를 돌릴 때 조건을 잘못 주거나,

삽입 삭제 동작으로 반복자를 무효화 시킬 수가 있다.

1. **과제 해결 방법**

1) 토론

- 같이 수업을 듣는 친구들과 기본적인 틀을 만드는 것부터 토론을

시작하였다. 메인에서 모든 일을 처리 할 것인지 아니면 핸들러를 이용해서 일을 하는 곳을 나눌 것인지를 이야기 하였다. 그렇게 해서 나는 데이터

관리를 해줄 CMainGame과 플레이어의 데이터를 담아줄 CPlayer

Class 두 개를 만들었다.

- 그런 다음 Container를 어떤 것을 사용할 지에 대하여 토론을 하였다. Vector는 임의적 접근이 가능하고, List는 삽입과 삭제가 빠르다. 하지만 임의의 요소의 접근이 느려 여기서는 플레이어의 데이터를 찾아 작업을

해야 하기에 Vector를 사용하기로 하였다.

- 이제 자료를 저장할 Container를 선택하였으니 이것을 어떠한 형태로

저장을 할지에 대하여 생각을 해보았다. 이를 그냥 Vector로 담아서 관리하기엔 비용이 커지기에 Pointer를 사용하기로 하였다. 하지만 Pointer로만 하기엔 해야 할 작업이 전부 해결 되는 것이 아니여서 smart\_ptr을 사용하기로 했고, 그 중에서도 shared\_ptr을 사용해

Playerlist 포인터를 여러 소유자에게 할당을 하여 참조 할 수 있게 하였다.

2) 관련된 자료 조사

- 토론을 하면서 그 주제에 대해서 stl교재, 인터넷 검색을 통해서

사용하고자 하는 것의 특징, 장ㆍ단점 등을 비교해보고 좀 더

프로그램에 적합한 모습을 찾기 위해 노력을 하였다.

3) 수업 내용 복습

- 이제 자료조사도 마치고 프로그램이 조금씩 틀을 갖춰가고 있었다. 하지만 계속 진행해 나가면서 원하는대로 진행이 안되어 이를 해결하기에 고민을 하던 교수님의 수업내용 중에 같은 문제를 다루는 것들이 있어 이를

찾아보고 코드에 적용을 시켜 문제를 해결 하였다.

-1-

**2. 요구사항 중 구현하지 못한 내용과 어려웠던 점**

1) 구현하지 못한 내용

- 없음.

2) 어려웠던 점

i. 프로그램 설계

- 이 과제를 시작하면서 아무런 설계도 하지 않고 과제를 진행하려다

진행이 불가능할 것 같아서 친구들과 토론을 통해서 설계를 시작하게

되었다. 하지만 사람들 간의 의견과 방식이 다르기 때문에 이를 조율하고

각자에게 도움이 될 만한 방법을 찾으려고 많은 노력을 하였다.

ii. 파일 입출력을 이용한 저장, 불러오기

- 가장 처음 테스트 코드로 코드가 문제가 없는지 확인을 하고 이를

수정하는 과정에서 클래스의 구조나 자료형 등이 바뀌는 경우가

발생하였는데 이를 입출력 부분을 수정하지 않고 바로 저장, 불러오기를

하였다가, 데이터가 이상하게 저장되고 데이터를 사용하는데 문제가

생겼었다. 그래서 코드를 다시 되돌아보고 원래 있던 데이터를 없애고

다시 생성을 하여 문제를 해결하였다.

iii. 다른 내용을 저장하는 자료구조 간에 데이터 공유

- 랭킹 시스템을 구현하면서 원본 데이터를 각 게임별 점수로 정렬을 해서

각각의 Vector에 자료를 담고 랭킹을 설정을 했는데, 이를 다시 원본

데이터 값에 랭크 값을 주기엔 10000번의 작업을 2번이나 해야하는

일이 발생해 버린다. 그래서 이를 해결하기 위해 shared\_ptr을 사용하여서

각각의 Vector들이 서로를 참조 할 수 있게 만들어 주었다.

-2-

**3. 과제를 하면서 느낀 점**

1) 많은 양의 데이터를 관리할 때는 포인터를 사용해

관리하는 것이 좋다.

- 데이터의 양이 많아지고 그 데이터의 크기가 크면 그것을

이용하는 비용이 커지게 되고 그렇게 되면 속도가 현저하게

떨어지게 된다. 그러나 Pointer를 사용하게 되면 그 크기가

확실히 작아지기 때문에 좋다.

2) 코딩을 할 때 무작정 코드를 치는 것 보다

전체적인 그림을 그려보고 시작하는 것이 좋다.

- 예전부터 강조 되어 왔던 이야기지만 의식의 흐름대로만

코드를 작성하는 것이 아니라 전체적인 틀을 잡고 그 틀에

살을 붙여나가는 식으로 해야 진행에도 도움이 많이 된

것 같다.

3) 하드코딩을 하지 않는 습관을 들여야 한다.

- 하드코딩을 사용하게 되면 stack overflow가 불리게 되거나

필요없는 작업을 하는 경우가 발생하게 될 수가 있다.

실제로 벡터의 Capacity를 100,000을 예약하고 for문을 0을

하나 더 붙인 1,000,000만큼을 돌리게 되어 계속 stack

overflow가 불리게 되었었다. 심지어 0하나 차이여서 코드

내에서 찾는데도 고생을 많이 하였다.

-3-