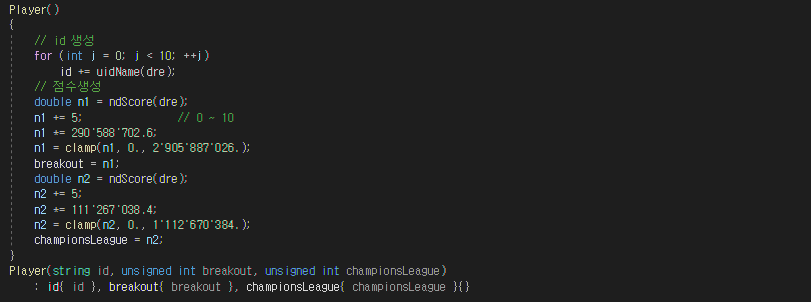
**STL 보고서**

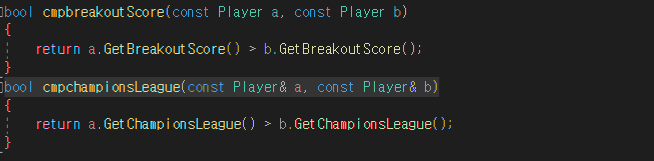
2016182027 이미륵

**[과제 해결 방법]**

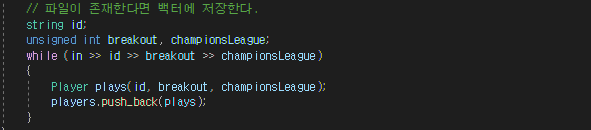
저는 Player클래스를 정의하여 private에 자료형이 string인 것과 unsigned int인 것을 각각 id와 점수를 저장할 수 있게 변수를 선언해 주었습니다. public에는 기본생성자에서 id와 점수를 받을 수 있게 하였습니다. default\_random\_engine과 uniform\_int\_distribution 을 사용하여 플레이어의 ID를 랜덤으로 생성하였고 normal\_distribution으로 점수는 정규분포사이에서 랜덤으로 값을 갖게 하여 생성하였습니다. 또한 매개 변수가 있는 기본생성자를 정의하여 텍스트 파일에서 불러왔을 때 벡터에 저장하기 위해 사용합니다.



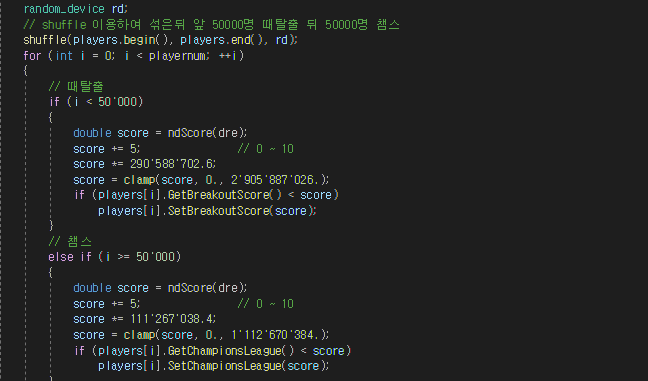
그 외에 외부에서 데이터를 읽을 수 있는 string GetName()과 unsigned int GetBreakoutScore(), unsigned int GetChampionsLeague() 함수를 정의하였습니다. 그리고 외부에서 값을 넣어서 점수를 갱신하기위해 void SetBreakoutScore(double score)와 void SetChampionsLeague(double score)를 정의하였습니다. 이렇게 클래스 Player가 정의 되어있고, 나중에 sort할 때 각 점수들을 비교해주기위해 bool cmpbreakoutScore(const Player a, const Player b)과 bool cmpchampionsLeague(const Player& a, const Player& b)를 선언하였습니다.



이제 main으로 가보겠습니다. 저는 100,000개의 Player를 vector에 저장하는 방법을 사용하였습니다. vector<Player> players를 선언하고 플레이어가 100,000명이니 reserve(100000) 선언하여 메모리를 할당하였습니다. 그 후에 ifstream으로 텍스트 파일을 열도록 하였습니다. “PlayerList.txt” 파일이 존재하지 않다면 100,000명의 플레이어를 생성하게끔 하였습니다. 만약 있다면 텍스트파일을 읽어와서 벡터에 하나씩 pushback으로 넣게 하였습니다.



저는 게임 진행과 플레이어 검색을 다음과 같이 했습니다. 이 과정을 반복 실행할 수 있어야 하므로 while문을 사용하였고 그 안에 switch-case로 1.게임 시작 2.플레이어 검색 3.종료 이렇게 구성하였습니다. 게임 시작을 하면 random\_device를 이용하여 난수를 만들어주었습니다. 플레이어가 100,000명이 있다면 그 중 50000명은 때탈출, 나머지 50000명은 챔피언스리그를 플레이해야 하는데 저는 shuffle함수를 이용하였습니다. shuffle함수를 사용하여 벡터 players를 뒤죽박죽 섞어 놓고 for문으로 벡터의 0번째부터 하나하나씩 읽어가면서 49999번째까지 때탈출을 플레이하고 나머지는 챔피언스리그를 플레이하도록 하였습니다. 똑같이 점수를 랜덤으로 넣어주고 기존의 점수보다 낮으면 그대로 높다면 높은 점수를 저장해주었습니다.



이렇게 게임 진행을 하게끔 하였습니다. 플레이어 검색은 sort를 사용하였습니다. 우선 검색할 플레이어를 받았고 sort함수를 이용하여 내림차순으로 정렬한 후 검색한 플레이어가 몇 등인지 몇 점인지 출력하고 그 위와 아래에 위치한 플레이어도 같이 출력하도록 하였습니다. 때탈출과 챔피언스리그의 등수와 점수를 출력하는 방식은 동일합니다. 마지막으로 3번을 누르면 while문을 빠져나오고 main문이 끝나기 전에 벡터 players를 “PlayerList.txt”를 ofstream으로 저장해줍니다.

**[자신의 해결방법이 옳다고 주장하는 근거]**

Player 객체를 왜 vector로 했는지에 대한 이유는 다음과 같습니다. 맨 처음에 객체를 어떤 컨테이너에 저장할지 고민 했었습니다. 고민 결과 array, map, set, vector으로 추려졌습니다. array는 vector보다 크기 조정에 있어서 비용이 많이 들어서 탈락되었고 map은 검색속도면에서 가장 유리하고 자동 정렬 기능이 있어서 효과적이라고 생각했지만 때탈출 점수와 챔피언스리그 점수를 같은 map에 넣을 수가 없어서 map를 두개나 선언해야 하는데 너무 비효율적이라 탈락되었습니다. set도 map과 비슷한 이유로 탈락되었습니다. vector는 연속적인 컨테이너이고, 순차 접근이 가능하며, 랜덤 엑세스가 가능합니다. 비록 vector가 다른 컨테이너에 비해 검색이 느리고 중간 위치의 삽입 삭제가 어렵지만 이 과제에서 삽입과 삭제할 필요가 없고 검색은 느리지만 다른 컨테이너를 모든 면에서 따지고 보면 비용적으로 훨씬 효율적이기 때문에 vector를 선택하였습니다. 무엇보다도 vector가 저에게 친숙하게 다가왔기에 사용하기 편했습니다. 게임 진행에서 shuffle를 사용한 이유는 맨 처음에는 난수를 50000개 뽑아서 선택된 수가 때탈출, 나머진 챔피언스리그를 플레이 하게끔 하려고 했었습니다. 하지만 최근에 교수님께서 강의 해주신 것중에 로또 추첨하는 코딩을 하면서 shuffle를 알려주셔서 이 과제에 써먹을 수 있다고 생각하여 사용했습니다. 또한 sort를 사용한 이유는 시간복잡도를 따져보자면 shuffle후 for문으로 한번만 돌려서 검색하면 되기 때문에 O(n)입니다. 그렇기에 수행시간으로 보면 다른 방법보다 훨씬 효율적이라고 생각해서 sort를 사용하였습니다.

**[요구사항 중 구현하지 못한 내용과 어려웠던 점]**

맨 처음에 Player 클래스를 만들면서 그 안에서 객체를 생성하고 그 객체를 벡터에 넣으려고 했으나 생각처럼 쉽게 되지 않았고 만약 클래스 내에서 벡터를 사용해서 반환한다면 메모리를 많이 사용하게 될 것 같았습니다. 또한 main문에서 처음에 플레이어를 벡터로 하나하나씩 저장할 때 한번에 넣는 방법이 있는지 고민해보았는데 아무리 생각해도 도저히 생각이 떠오르지 않았습니다. 게임 진행에 있어서도 어려운 점이 있었습니다. 보통 난수 생성을 해서 랜덤 값을 준다 해도 디버깅하고 종료했다가 다시 돌려도 같은 난수를 뽑아내기 때문에 게임 시작을 하고 계속 똑 같은 플레이어의 ID값을 찾고 그 플레이어의 점수를 살펴보면 매번 똑같이 진행되었습니다. 그래서 이것을 바꾸고 싶어서 고민하던 중 교수님께서 강의 시간에 random\_device를 사용했던 것이 기억이 나서 한번 시도해보니 매번 다르게 나와 해결하였습니다.

**[과제를 하면서 느낀 점]**

1학년 2학기에 C++이 너무 어려웠고 성적도 좋지 않았습니다. 그러고 2학년 때 윤정현 교수님 C++로 재수강을 하였는데 1학년때와 달리 머리 속에 속속 잘 들어왔고 좋은 결과를 가지고 완강 하였습니다. 그러고 이번에 STL를 듣고 있는데 C++기반이라고 해도 새로 배우는 것이라 아직 미숙하고 어렵게 느껴집니다. 하나의 과제나 무언가를 작성할 때 시간이 오래 걸리고 방향도 잘 안 잡힐 때도 있습니다. 역시 이번 과제를 수행할 때에도 처음에 어떻게 할까 막막하고 도저히 생각이 떠오르지 않았습니다. 저번 STL강의 때 교수님께서 코드를 무작정 치는 것 보다 먼저 어떻게 할지 한글로 적어보고 그 적은 것을 토대로 한 줄씩 코딩하면 된다고 하셨습니다. 지금껏 그냥 머리 속에서 생각만하고 그대로 코딩했던 제가 교수님이 말씀하신 것처럼 한줄 한줄 코딩을 하다 보니 잘되었고 재미있게 이번 과제를 수행했습니다. 저는 이번 과제가 어떻게 하면 효율적인 코드가 될지 생각해보는 과제라고 생각합니다. 그래서 하나하나 코딩할 때 과연 이 함수를 사용하는게 맞는지 수행하는 시간의 계산량을 생각하며 과제 하다 보니 어려웠습니다. 코딩을 다하고 돌려보았 때 성취감에 뿌듯했고 기분이 좋았습니다. 이번 과제를 통해 STL을 조금 더 알게 된 것 같고 저의 코딩 실력이 조금이라도 올랐다고 느껴졌습니다.

**[그 외 정리하고 싶은 내용]**

과제를 수행하면서 느꼇지만 난수값들을 얻는 부분에 있어서 많은 어려움이 있었습니다. 플레이어의 ID와 때탈출 점수, 챔피언스리그의 점수의 값을 랜덩으로 생성할 때 default\_random\_engine, random\_device, uniform\_int\_distribution, normal\_distribution을 사용하였는데 이 함수를 정확하게 알고 싶어서 구글에 검색해봤습니다. C언어의 경우 보통 rand() 함수와 time(NULL)을 통해 시드값을 지정하였습니다. 하지만 시드값이 너무 천천히 변한다는 점과 균등하게 난수를 생성하지 못한 점등 여러 문제가 있었고 C++에서는 좀더 양질의 시드값을 얻기 위해 random\_device 라는 것을 제공합니다. Default\_random\_engine은 의사 난수를 생성하는 클래스 입니다. 사용자가 직접 다룰 수 있으며 동기화할 필요가 없다면 rand()보다 빠릅니다. 가장 어렵게 느껴진 부분은 normal\_distribution인데 뜻은 정규 분포입니다. 평균값과 표준편차 값을 생성자를 통해 입력 받는데 정규 분포를 이해하지 못하면 사용하기 어렵게 느껴졌습니다. 그래서 구글에도 많이 검색을 해보았는데 완벽하게 이해하기가 어려웠고 이번 과제에서 사용할 때 교수님께서 강의시간에 알려준 것을 바탕으로 해서 응용하여 사용하였습니다. 기회가 된다면 랜덤값을 생성하는 방법을 다양하게 자세하게 알고 싶고 이해해보고 싶습니다.

또한 게임을 계속 진행하다 보니 결국 등수가 변하지 않고 플레이어의 점수들이 수렴하게 되었습니다. 결국 무한대로 게임을 진행하면 더 이상 값은 변하지 않을 것이라고 생각해서 플레이어의 때탈출 점수와 챔피언스리그 점수를 초기화 즉, 리셋을 시켜주는 기능을 추가하였습니다. 리셋 방법은 맨 처음 Player 클래스의 기본생성자에서 점수를 초기화해주는 방법과 동일하게 코딩하였습니다.