STL 과제 보고서 - 2021184018

한국공학대학교 게임공학과 3학년 STL 과제 보고서

1. 파일 읽기 및 컨테이너 저장 방식

std::ifstream을 통해 바이너리 파일 "2025 STL 과제 파일 - 2021184018"을 열고, Player::read() 멤버 함수를 이용하여 std::array<Player, 2'500'000>에 모든 데이터를 한 번에 읽었다. 바이너리 파일 구조에 맞게 Player 객체의 기본 정보(name, score, id, num)를 먼저 읽고, 그 이후에 unique\_ptr<char[]> p로 관리되는 메모리 블록을 num 바이트만큼 추가로 읽는다. 이 과정은 반복문 하나로 250만 개 객체에 대해 실행되며, 메모리에 모두 적재된 후 추가적인 파일 접근 없이 문제들을 해결하였다. 마지막 Player 객체는 Show() 함수를 통해 다음과 같이 출력되었다.



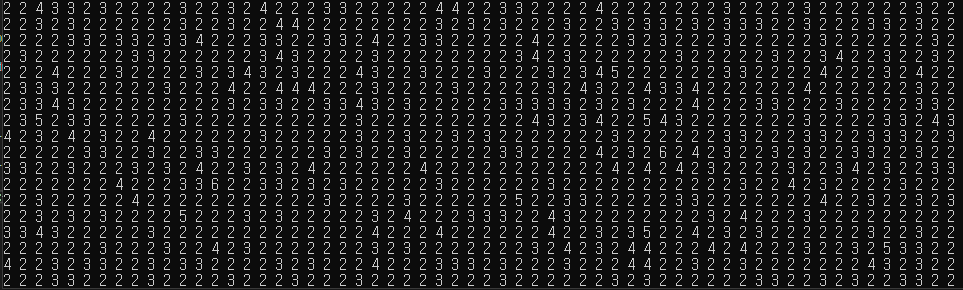
2. 최고 점수와 평균 점수 계산

모든 Player의 점수를 scoreMap이라는 std::map<int, vector<ref>>에 score를 key로 하여 저장하고, rbegin()을 통해 가장 큰 점수의 Player를 찾았다. 평균 점수는 std::accumulate를 사용하여 operator+= 오버로드로 모든 Player의 score를 더해 계산하였다. 출력 예시는 다음과 같다:



3. 같은 id를 가진 Player 탐색 및 저장

모든 Player를 id 기준으로 정렬한 후, 인접한 두 객체의 id가 같은 경우를 탐색하여 "같은아이디.txt"에 출력했다. 중복되는 id의 개수는 2개부터 전부 화면에 한 칸 간격으로 계속 출력되게 하였다.



4. p가 가리키는 문자열 정렬 및 'a' 개수 기준 필터링

Player::getBuffer()를 이용해 unique\_ptr<char[]>가 가리키는 문자열을 std::string\_view로 받아 정렬에 사용하였다. 이후 std::count를 통해 'a'가 10개 이상 포함된 Player 객체의 수를 집계하였다. 결과적으로 해당 조건을 만족하는 객체 수는 아래처럼 출력되었다:



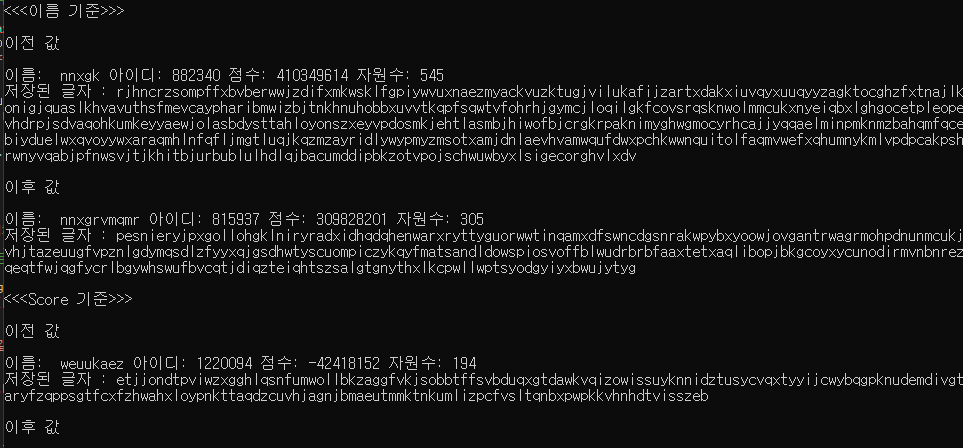
5. id 기반 검색 및 3가지 기준 탐색

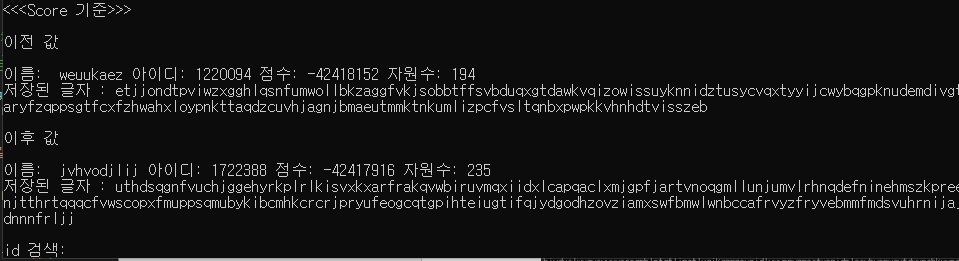
사용자로부터 id를 입력받아 idmap에서 참조를 검색한다. 일치하는 id가 있으면:

- id 기준으로 앞뒤 Player 출력 (std::map으로 이루어진 idMap)  
- name 기준으로 앞뒤 출력 (players[]를 name 기준으로 사전에 정렬)  
- score 기준으로 앞뒤 출력 (scoreMap을 활용)

특히, 배열 내 포인터 차를 이용해 &player - players.data()를 통해 반복자 없이 인덱스를 계산하였고, STL 정렬 함수 및 std::map, std::vector<ref>를 적절히 혼합하여 탐색 성능을 확보하였다. id 입력은 std::cin.fail() 확인과 std::cin.clear() 및 ignore() 없이 구현되어 있어, 이는 개선할 수 있는 지점이다.









효율성 평가 및 개선점

전체 데이터는 고정 크기의 std::array로 관리되므로, 반복적인 할당과정이 없고 메모리 오버헤드도 낮다. map<int, vector<ref>>를 활용한 그룹핑은 중복 탐색을 효율적으로 수행할 수 있게 해주며, 정렬 후 탐색에서는 std::prev, std::next를 적극 활용하여 포인터 및 반복자 연산을 잘 사용하였다.

개선 가능한 점:  
- std::cin 입력 실패 처리 추가 필요  
- Answer5에서 ++++itr처럼 비직관적인 표현보다는 std::next(itr, 2)가 더 가독성이 좋음  
- Player의 read()에서 name.data()에 바로 읽는 건 std::string의 정의상 안전하지 않음 (UB 가능)

과제를 하면서 느낀 점

이번 STL 과제를 통해 std::array, std::map, std::vector, unique\_ptr, string\_view, accumulate, sort, count 등 다양한 STL 기능을 실제 대규모 데이터에 적용해보는 경험을 할 수 있었다. 특히 바이너리 파일 입출력과 메모리 할당을 직접 다뤄보면서 C++의 자원 관리 기법과 STL의 반복자 기반 설계의 장점을 체감할 수 있었다. 250만 개 객체를 대상으로 수행된 고효율 데이터 처리 경험은 향후 게임 개발이나 실시간 시스템에서 큰 도움이 될 것이라 생각한다.