STL 과제

게임공학과 2020182028 이규원

1. 파일에 저장된 모든 Player 정보를 컨테이너에 저장하라. 제일 마지막 Player의 정보를 다음과 같은 형식으로 화면에 출력하라.

결과 :

마지막 플레이어 정보:이름: bgpdeifnqylh, 아이디: 530402, 점수: -170253909, 자원수: 21

저장된 글자: imvsocqmohbjwvtpjpegb

* 파일 입출력

파일을 열기 위해 Ifstream로 열었고, 바이너리 파일이었기에 ios::binary로 열었다. 혹시 열리지 않았을 때를 대비해 열리지 않았을 시 에러코드 444로 탈출할 수 있게 만들었다.

* 컨테이너 사용

컨테이너는 5번 문제의 전후 데이터의 접근을 빠르게 접근할 수 있도록 vector 사용했다.

* read함수

Player를 읽어오기 위해 class에 read함수를 만들었다.

우선 유니크 포인터를 제외한 나머지 부분 name, score, id, num을 먼저 읽어오고, 유니크 포인터의 크기를 읽은 후 유니크 포인터가 가리키는 실제 메모리를 할당하고 데이터를 읽어왔다.

* Player 정보 출력 함수

다음은 Player의 정보를 출력하는 함수를 만들었다.

Private 내부 데이터에 접근하기 위해 friend 선언을 하고 name, id, score, num, 글자를 출력하는 코드를 작성했다.

* 1번문제 가장 마지막 Player의 정보 출력

while문으로 vector 컨테이너에 players에 push\_back을 통해 넣었고 마지막 플레이어 정보를 꺼내기 위해 players.back()으로 맨 뒤의 정보를 꺼냈다.

* 해결되지 못한 버그

이 때부터, 해결할 수 없는 알 수 없는 버그가 생겼는데, 테스트나 정보를 쓰기 위해 주석들을 적으면 ‘{‘, ‘}’ 를 식별할 수 없다고 하거나, 전방 선언되어 있는 함수를 읽지 못하는 버그가 생겼다. 주석을 달 수 없어 코드의 정보 전달 능력이 떨어져 많은 애로 사항이 있었는데 끝내 해결하지 못해 가장 답답했던 버그였다. 코드 옆에 주석처리를 하면 괜찮은 경우도 있었다.

1. 점수가 가장 큰 Player를 찾아 화면에 출력하라.

Player의 평균 점수를 계산하여 화면에 출력하라.

결과 :

플레이어의 평균 점수: 513.298

플레이어의 최고 점수: 이름: sgmijivbbzs, 아이디: 2096573, 점수: 423456642, 자원수: 80

저장된 글자: qnmlufkiwkmcerwzwdrwjanogrmzqrgyosmoeviupksdaqgdvribjhxzioxjhmkysezmjuqtgufjdqhj

* 플레이어의 점수만 입력받기

Player의 점수만 계산하기 위해 player Class에 getScore함수를 추가해 점수를 리턴 받을 수 있도록 만든다.

* 최대 점수, 평균 점수 탐색

총 점수를 얻기 위해 algorithm - accumulate를 통해 players의 점수를 계산하고, 평균으로 나누기위해 players.size()로 나눠 평균 점수를 구했다.

Algorithm - Max\_element를 이용해 플레이어의 최고 점수를 출력했다.

1. id가 서로 같은 객체를 찾아 "같은아이디.txt"에 기록하라. id가 같은 객체는 모두 몇 개인지 화면에 출력하라. - "같은아이디.txt" 일에는 Player 마다 이름과 아이디 정보만을 한 줄씩 기록한다

결과 : 같은아이디 개수: 1146331

* Ofstream

ofstream out을 통해 “같은아이디.txt”에 파일을 쓸 수 있도록 준비해 뒀다.

* Map을 활용한 중복자료 입력

Player – id를 중복 데이터를 찾는데 있어 가장 효율적인 자료구조는 map이라고 생각했고, map에 player 정보를 넣고, pair.second.size() > 1 동일 ID를 가진 플레이어가 1명 초과일 때 out에 중복 player 정보들을 쓰도록 구현했다.

* 중복자료 개수

그 뒤, int 값으로 중복자료를 찾을 때 마다 int값을 올려 총 몇 개의 자료가 있는지 셈했다.

* 개선된 / 개선 할 부분

중복자료 입력과 개수를 세는 것을 동시에 할 수 있도록 코드를 수정했고, 그럼에도 중복자료를 찾거나 개수를 세주는 STL이 있을 것 같은데 아쉽게도 찾지 못해 직접 구현했다.

1. 멤버 p가 소유한 메모리로 읽어온 char를 오름차순으로 정렬하라. 'a'부터 ‘z’까지 모든 문자가 메모리에 있는 Player의 개수를 화면에 출력하라

결과 : a to z : 1632855

* sortP, p가 소유한 메모리를 오름차순으로 정렬하라.

1.p.get()을 호출해 문자열을 얻어 String 객체 str에 저장한다,

2.Sort를 통해 문자열의 문자를 오름차순으로 정렬한다.

3.NULL값을 포함해 1을 더해 p에 새로운 char배열을 동적 할당한다.

4.Strcpy로 원본 내용에도 적용한다.

* IsAtoZ, A부터 Z까지 모든 문자가 메모리에 있는지 확인

1.String 객체 str에 문자를 입력 받는다.

2.해당 문자가 알파벳인지 확인한다.

3.tolower로 우선 문자를 소문자로 변경하고 해당 값을 ‘a’를 빼 인덱스 0~25를 얻는다.

4.alphabetMask에 비트OR 연산을 통해 각각의 인덱스를 입력한다.

5.모든 알파벳 비트가 활성화 되었는지 확인하고 결과에 따라 값을 리턴 한다.

* 위 함수들을 이용해 IsAtoZ가 true를 리턴 할 때 마다 count를 세 그 값을 출력했다.

1. [ LOOP ] id를 입력 받아 존재하는 id라면 다음 내용을 한 번에 화면 출력하라

결과 :

텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명

* 입력 처리

While을 true로 돌려 반복문을 만들고 size\_t인 targetID를 cin했다.

* ID 정렬 및 검색

Algorithm - Sort를 통해 ID기반 정렬을 수행한다.

targetID에 입력했던 id의 첫번쨰 위치를 찾아 search에 저장한다. 이때 player 내에 존재하는 id가 아니면 처음으로 돌아간다.

ID가 존재하면, vector을 이용해 맨 앞의 함수가 아니면 prev을 통해 이전의 정보를 출력하고 함수 본인을 출력, next를 이용해 이후의 정보를 출력한다.

* Name, Score 출력

위와 마찬가지로 sort를 Name, Score을 재정렬한다. 이 때, name, score 검사할 때 잘못된 멤버를 가리키지 않도록 두번째 정렬 전에 찾아낸 정보를 저장해둔 search의 id, name, score을 미리 저장한다.

Name, score에선 Find 대신 조건을 검사할 수 있도록 Find\_if를 통해 target의 ID, name, Score 모두 동일한 player를 구한다.

출력은 동일하게 name, score가 존재하면, vector을 이용해 맨 앞의 함수가 아니면 Prev을 통해 이전의 정보를 출력하고 함수 본인을 출력, next를 이용해 이후의 정보를 출력한다.

* 개선할 부분

반복되는 작업이 많아 따로 함수를 빼서 할 수 있었을 것 같지만 시도했을 때의 에러를 잡지 못해 일단은 각각 쓰는 방식으로 구현했다.

매 탐색마다 정렬을 해주고 있는데, 따로 정렬한 데이터를 담아 놓으면 2번째 검사부턴 데이터를 더 정렬하지 않아도 되겠지만, 동시에 그만큼 저장공간을 3배는 더 차지하게 될 테니 어떤 방식이 더 효율적인지는 아직 고민중이다. 아마 상황에 따라 더 나은 방식이 있는 부분인 것 같다.

* 과제를 하며 느낀 점

본문에는 서술되지 않았지만, 과제를 진행하면서 컨테이너를 꽤 자주 바꿨다. 과제 전까지는 array, vector의 쓸모를 크게 못 느끼고 메모리를 효율적으로 관리할 수 있는 deque나 list쪽에 계속 눈이 갔었는데, 5번 문제를 진행하면서, 특정 변수에 접근하는 데 있어, vector가 얼마나 큰 효용 가치를 가지고 이번 과제를 통해 깨달을 수 있었다. 동시에, 최고의 컨테이너는 존재하지 않고, 단지 상황에 따라 더 나은 선택지가 있을 뿐이며, 프로그래머는 그 사이에서 득실을 생각하며 선택을 하는 게 프로그래머의 임무이자 책임이라고 느끼게 됐다.

문제는 정말 단순한 것처럼 몇 줄로 전부 설명할 수 있는 내용이었지만, 실제로 구현할 땐 많은 어려움과 시행착오를 겪어야만 했다. 문제 하나하나가 구조에 대한 이해와 그 상황에 알맞은 템플릿을 가져오는 능력을 요구했으며, 코드를 쓰면서도 “아, 이건 템플릿으로 있을 것 같은데..”라는 생각을 계속 들게 만들었다. 다행히 잘 찾아서 구현된 내용도 있지만, 있을 것 같은데 구현 못한 내용도 꽤 있었다. 대표적으로 3번의 중복 아이디 찾을 때 템플릿으로 구현할 수 있을 것 같은데 그러지 못해 직접 손코딩으로 구현해 많이 답답했다.

당연한 말이지만, 코딩에 정답이 없다는 듯, 가장 최선에 가까운 답을 찾기 위한 여정이 얼마나 어려운지 말하는 듯했고, 내 코드를 다시 보고 있으면 최적화면으로 괜찮을 지 많이 의문이 들게 만들었다. 그럴수록 코딩의 길은 마치 안개가 낀 산을 등반하듯, 꼭대기는 어딘지 모르지만, 더 효율적이고 나은 방법이 존재함을 내게 일러주는 것 같았다. 이번 과제를 하면서 그런 부분을 많이 느꼈다.

동시에 STL을 잘 활용하는 게 얼마나 중요한 능력인지도 느꼈다. STL이 아니라 직접 구현하면서 많은 노력도 들어가는 것도 그렇지만, 동시에 가독성, 효율성 측면에서도 얼마나 좋지 않은 방법인지 뼈저리게 느낄 수 있었다. 만일 STL이 없었다면 아마 대부분의 소프트웨어들을 개발하는데 몇배의 시간과 버그들을 마주치지 않을까 싶다.

정말 어려운 과제였지만, STL의 중요성과 활용, 뿐만 아니라 코딩의 최적화엔 끝이 없다는 경험을 얻게 됐다.