STL 과제 보고서

컴퓨터공학과

2015152025 이강우

1. 과제 해결방법
   1. 점수 구현 방법  
      점수는 normal\_distribution 정규 분포를 사용하였고,  
      평균 0.0 , 표준편차 3.0 으로 설정하였습니다.  
      clamp를 사용하여 초기 정규 분포 값을 -5 ~ +5로 설정하였고, 그 값에 +5를 해주어  
      0 ~ 10으로 정규 분포 값을 초기화 하였습니다.  
      초기화된 값에 떼탈출, 챔피언스리그 각각의 최대값을 곱해주어 각자의 범위에 맞게 지정해주었습니다.
   2. 파일 존재 유무
      1. 파일이 없을 때  
         파일이 존재하지 않는 경우 데이터를 새로 생성해주었습니다.  
         이 때는 임의의 5’0000명을 지정하지 않고 모든 플레이어에게 떼탈출, 챔피언스리그 점수를 모두 가지게 하였습니다.
      2. 파일이 있을 때  
         파일에 있는 데이터를 받아 vector 컨테이너에 적용하였습니다.
   3. 메뉴 선택  
      while문을 통해 플레이어가 종료하기 전까지 반복 실행할 수 있게 하였습니다.  
      1. 게임 진행  
      2. 나의 떼탈출 랭킹  
      3. 나의 챔피언스 리그 랭킹  
      4. 나가기  
      로 메뉴를 구성하였습니다.  
        
      반복되는 입력에 CMD 창이 복잡해 지는 것을 방지하기 위해  
      system(“cls”)를 통해 CMD 창을 정리하였고, sleep\_for() 함수를 통해  
      정리하기 전 1.5초 동안 결과 창을 보이게 하였습니다.
   4. 메뉴 입력 값 받기  
      1 ~ 4번까지 받기 때문에  
      int input;   
      cin >> input;   
      -> 정수형으로 입력을 받았으며,  
      문자 입력 시, cin.fail() 함수로 문자 입력을 예외 처리 하였습니다.  
      문자열로 입력 시, 입력 버퍼로 인해 생기는 에러를 방지하기 위해  
      에러 비트를 초기화 해주는 clear() 함수와   
      버퍼 값을 비워주는 ignore()함수를 사용하였습니다.  
      cin.ignore(INT\_MAX, ‘\n’) 을 통해 정수형의 최대값만큼 읽고,   
      개행 문자가 들어오면 종료시키는 것으로 하였습니다.  
      이를 통해 버퍼가 비워지지 않아 문자 입력 시, 반복되는 출력을 방지하였습니다.

==== 메뉴 별 기능 정리 ====

1. 게임 진행  
   임의의 5’0000명을 선택하기 위해 새로운 vector 컨테이너를 초기화 하였고,  
   컨테이너의 각 인덱스는 자기 인덱스 번호로 초기화하였습니다.  
   shuffle( ) 함수를 통해 vector 컨테이너를 섞었고,   
   10’0000명의 플레이어가 있는 vector 컨테이너에 shuffle( ) 함수로 섞은 vector 컨테이너의 각 인덱스 값을 인덱스로 주어 임의의 5’0000명을 선정하였습니다.  
     
   그리고 1.1 점수 구현 방법과 동일하게 정규분포를 통해 점수를 구현하여  
   10’0000명의 플레이어에게 새로운 점수를 초기화하였습니다.  
   점수 초기화 시, 기존 플레이어가 가지고 있는 점수보다 큰지를 비교하여 기존 점수보다 클 때만 점수가 반영되도록 하였습니다.
2. 나의 떼탈출 랭킹 / 3. 나의 챔피언스리그  
   유저에게 ID를 입력 받고, 해당 아이디가 있는지 검색합니다.
   1. ID 존재  
      10’0000명의 플레이어 데이터를 내림차순으로 정렬하고, 정렬한 vector 컨테이너(이하 A )에서 입력 받은 ID를 검색합니다.  
      distance 함수를 통해 몇 번 인덱스에 있는지 확인하고,  
      확인 된 값을 통해 상위 %를 계산하였습니다.  
        
      인덱스를 알고 있으므로, A 컨테이너에서 인덱스의 +1 과 -1에 있는 플레이어의 데이터를 같이 화면에 띄우게 하였습니다.
   2. ID 존재하지 않음  
      존재 하지 않을 시, “플레이어가 존재하지 않습니다.” 를 띄워주고 메뉴로 돌아가게 구성하였습니다.
3. 나가기  
   while 문을 빠져나가면서 빠져나가기 전 10’0000명의 데이터를 저장합니다.
4. 자신의 해결방법이 옳다고 주장하는 근거  
   저는 정규분포에서 평균 0 , 표준편차 3.0 을 주어 값의 차이를 크게 주었습니다.  
   각 상황의 예외처리를 지정해주어 런타임 에러를 방지하였고,   
   데이터를 정렬하거나 찾는 등 시간이 걸리는 상황에 대비하여 전과 후에 텍스트를 띄어 현재 진행 중임을 보였습니다.  
     
   vector 컨테이너를 쓴 이유는 상위 %를 구할 때 자기 자신과 자신의 앞과 뒤에 있는 플레이어의 정보도 접근해야 하는데, 데이터 정렬과 찾기를 빠르게 하기 위해 map과 set을 쓰기엔  
   앞과 뒤의 플레이어가 누구인지 찾는 것이 부정확하다고 판단하였고, 그러므로 vector 컨테이너를 사용하여 데이터를 관리하였습니다.  
     
   임의의 5’0000명을 구하는 방법이 옳다는 근거는 Shuffle 함수 입니다.  
   자신의 인덱스 번호를 값으로 가지는 10’0000 크기의 vector를 shuffle 함수로 섞어주었고,  
   이를 통해 매번 중복되지 않는 값을 가지는 vector 컨테이너를 구성할 수 있었습니다.  
   0 ~ 10’0000 값이 섞인 vector 컨테이너에서 차례대로 10’0000명의 플레이어를 가지고 있는 vector 컨테이너 인덱스로 주어 임의의 5’0000명을 선택하였기에,  
   가장 정확하고 빠른 방법이라 생각합니다.
5. 요구사항 중 구현하지 못한 내용과 어려웠던 점  
   정규 분포가 가장 어려웠습니다.   
   처음 정규분포를 적용할 때 평균 0 , 표준편차 1.0을 주어 확인을 하였는데, 값이  
   -1 ~ 1 사이로 나오게 되었습니다.   
   이것이 표준편차 때문임을 알았어야 했지만, 이를 파악하지 못하였습니다.  
   그리고 교수님께 질문을 통해 얻은 답을 분석하였습니다.  
   clamp 함수를 통해 데이터의 범위를 -5 ~ 5 사이로 정해두고, 거기서 +5를 더해  
   0 ~ 10을 만든 이유는  
   문제에서 떼탈출과 챔피언스리그 범위가 0 ~ n 까지 이기 때문이라 생각하였습니다.  
   그래서 저 또한 똑같이 설정하였고, 이를 통해 떼탈출, 챔피언스리그 별로 범위에 맞는 점수를 생성할 수 있었습니다.  
   어려웠던 이유는 정규 분포에 대한 개념이 부족하였고, 지식의 부족이 코드 접근에 대한 불안감을 키웠고, 이 불안감이 하고 있는 것에 확신을 주지 못하여 가장 어려웠던 것 같습니다.
6. 과제를 하면서 느낀 점  
   현재 저는 Unity3D를 통해 게임을 만들었고, 플레이스토어에 출시하여 관리를 하고 있습니다. ( “일렉트로드” 라는 게임입니다. )  
   이번 STL 수업을 들으면서 메모리 관리에 대해 다시 한번 생각하게 되었고, 이를 제가 만든 게임에도 적용하여 게임에 많은 최적화를 이루어냈습니다.  
   즉, 저는 제 게임 도움이 되기 위해 그리고 C++, STL를 제대로 다루기 위해 노력하였습니다.  
   하지만 이번 과제는 예상 외로 수학에서 벽을 느끼게 된 것 같습니다.  
   게임 클라이언트 프로그래머가 되기 위해서는 수학과 물리를 잘 해야한다고 생각했고,  
   이번 여름방학에 게임 수학을 독학해 실제로 어떻게 적용하는지 공부를 하려고 하였습니다.  
   이번 과제에서 정규분포, 상위 몇 %를 구하는 부분에서 막힘으로써 정확하게 푼 것이었지만 수학적인 부분에서 오는 불확실이 이번 과제에 많은 어려움을 느끼게 한 것 같습니다.  
   다시 한번 수학이 중요하다는 것을 깨우치게 되었고, 이번 여름방학에 게임 수학과 코딩 테스트를 더욱 공부하여 취업에 큰 도움이 될 수 있게 노력하겠습니다.
7. 그 외 정리하고 싶은 내용  
   정규분포를 사용하여 데이터를 생성하는 방법  
   shuffle 함수의 존재  
   -> 알고리즘 함수에는 거의 모든 알고리즘이 존재하는 것 같습니다. Shuffle도 처음에는 따로 함수가 없다고 생각하여 직접 구현하려고 했는데, 함수가 존재한다는 것을 알고 함수를 이용하였습니다.  
   알고리즘 헤더 파일이 제공해주는 기능들이 무엇이 있는지 더 파악해야 할 것 같고, 이를 저의 것으로 만들어서 필요할 때 쓸 수 있게 해야할 것 같습니다.