2020년도 2학기 컴퓨터공학설계및실험I

8주차 결과 보고서

20170175 김태안

1. 실습 목적

일반적인 게임은 사용자의 기록을 등록하여, 확인할 수 있는 기능을 갖는다. 테트리스 프로젝트에서도 이러한 기능을 하는 랭킹 시스템을 구현한다. 이 시스템은 테트리스 플레이가 종료되면, 사용자의 이름과 점수(score)를 기록하고, 기록된 랭킹 정보들을 확인할 수 있는 기능을 갖는다. 이 랭킹 정보들은 테트리스 프로그램 내에서 효율적인 자료구조를 사용하여 관리되고, 테트리스 프로그램을 종료할 때는 rank.txt 파일에 기록되어, 지속적으로 유지된다.

1. 실습 구현 내용
2. createRankList();

rank.txt 파일을 읽어 연결 리스트에 저장한다. 파일이 없다면 새 파일을 생성한다.

input : rank.txt.

output: ScoreList

1. rank();

rank는 랭킹 시스템의 전반을 관리하는 함수이다. menu에서 2번을 눌렀을 때 사용할 수 있는 모든 기능이 구현되어 있다. 숫자 1을 입력 받을 경우, 연결 리스트를 탐색하며 X부터 Y까지의 등수를 출력한다. 이때, 아무것도 입력하지 않으면 X는 1, Y는 마지막 랭크로 자동 설정된다.

input: command

output: 탐색한 rank 결과

1. void writeRankFile();

rank가 변경되었을 경우 1인 ScoreModifiedFlag를 확인한 뒤, 변경되었다면 rank.txt를 다시 쓴다.

input: ScoreModifiedFlag

output: rank.txt

1. void newRank(int score);

게임이 끝나면 사용자 이름을 입력 받아 새 등수를 기록한다. 내림차순인 연결 리스트를 탐색하며 점수가 들어갈 위치를 찾은 뒤, 알맞은 위치에 새 노드를 삽입한다. 그리고 ScoreModifiedFlag와 ScoreListLength 변수를 알맞게 수정한다. 새 Rank가 저장된 후, writeRankFile 함수를 호출해 rank.txt 파일과 synchronize한다.

input: score

output: ScoreList

1. 실습 환경

cspro.sogang.ac.kr

OS: Ubuntu 16.04.2 LTS (GNU/Linux 4/4/0-184-generic x86\_64)

Compiler: gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.12)

1. 과제
2. rank();

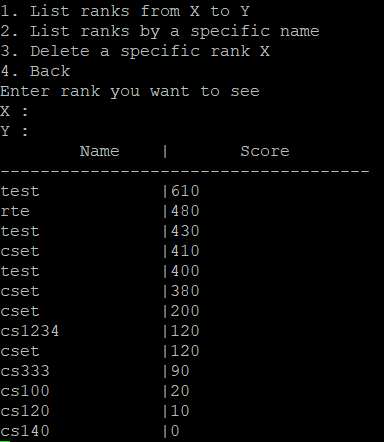
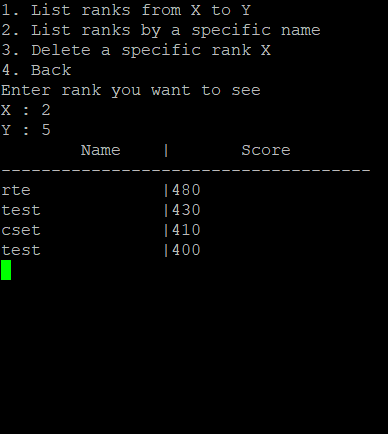
rank의 입력 2와 3에 대한 기능을 구현한다. 2를 입력 받을 경우, 사용자 명을 입력받고, 연결 리스트에서 해당 이름을 탐색한다. 결과가 있다면 결과를 출력하고, 없다면 오류 메시지를 출력한다. 이때, 같은 사용자 명이 여러 개 있을 경우 모두 출력한다.

3을 입력 받을 경우, 지우고 싶은 랭크의 숫자를 입력받는다. 입력 받은 숫자가 랭킹 범위에 있으면 해당 랭크를 찾아 노드를 삭제한다. 범위에 없으면 오류 메시지를 출력한다.

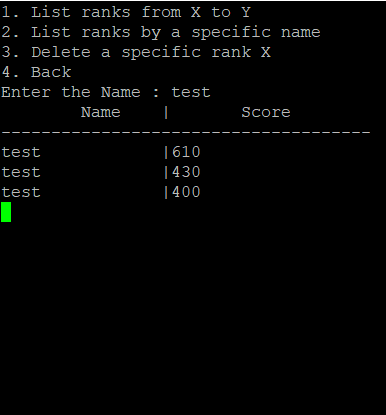
input: Command

output: Searched Result, ScoreList

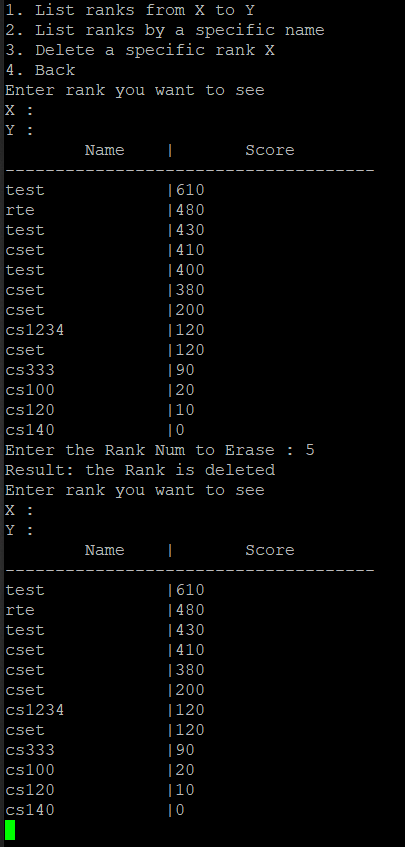
1. 실습 결과 및 분석

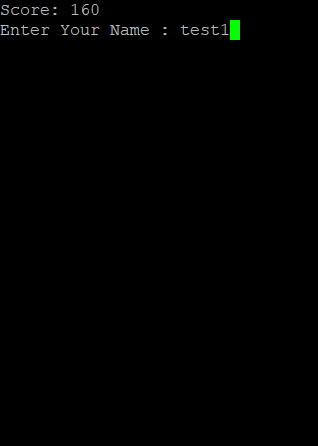
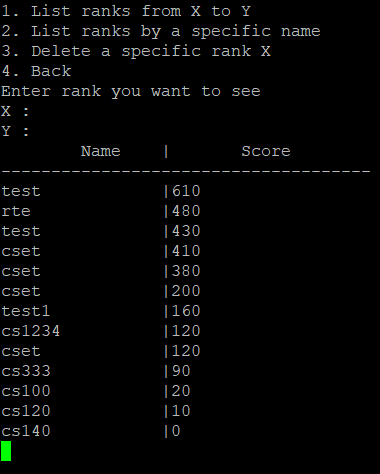
X, Y에 아무것도 입력하지 않으면 모든 rank를 볼 수 있다. 숫자를 넣어 보고 싶은 범위를 지정할 수 있다.



이름을 입력해 이름으로 결과를 탐색할 수 있다.



Delete 기능을 사용해 삭제하고 싶은 등수를 삭제할 수 있다.

게임이 끝나면 왼쪽과 같은 화면이 나타나고, 이름을 입력해 점수를 등록할 수 있다.

**함수 분석**

1. 공간 복잡도

연결 리스트는 노드의 개수만큼의 메모리를 차지한다. 따라서 rank가 차지하는 공간의 복잡도는 O(ScoreListLength)이다. 여러 상황에서 연결 리스트의 탐색을 진행할 때, 노드의 포인터를 통해 연결 리스트에 접근하기 때문에 추가적인 메모리는 포인터를 저장하는 메모리만 필요하다.

1. 시간 복잡도

rank()

rank는 총 3가지의 기능으로 이루어져 있다. 그 중 1, 2번 기능인 탐색은 ScoreListLength만큼의 노드를 모두 방문하며 값을 출력해야하므로 O(ScoreListLength)의 시간 복잡도를 가진다. 삭제의 경우, 삭제를 위해 전체 연결 리스트의 탐색이 이루어지기 때문에 마찬가지로 O(ScoreListLength)의 시간 복잡도를 가진다.

createRankList();

createRankList는 파일의 모든 점수를 읽어 이를 저장해야한다. 즉, 노드의 생성과 연결이 점수의 개수 ScoreListLength번 만큼 이루어지므로 O(ScoreListLength)만큼의 시간 복잡도를 가진다.

writeRankFile();

writeRankFile은 연결 리스트의 모든 원소를 탐색하며 파일을 쓴다. 따라서 O(ScoreListLength)만큼의 시간 복잡도를 가진다.

1. 결론

이번 실습에서는 연결 리스트를 활용해 rank 시스템을 구현하였다. 연결 리스트는 포인터를 통해 원소를 연결하여 리스트를 구현하는 방식이다. 연결 리스트를 사용하면 새로운 원소를 쉽게 삽입하고 삭제할 수 있다. 이는 몇 개의 점수를 저장할 지 정해지지 않은 테트리스의 rank 시스템을 구현하기에 알맞았다. rank의 탐색은 O(n)의 시간 복잡도를 가지며, 추가적인 메모리가 필요하지 않은 효율적인 방법이라고 할 수 있다.