7주차 TETRIS2 결과 보고서

전공 : 수학 학년 : 4 학번 : 20151378 이름 : 진상우

1. 실험 시간에 작성한 랭킹 시스템의 실자료구조와 랭킹 시스템의 각 기능에 대한 알고리즘을 요약하여 기술하시오. 본인이 선택한 랭킹 시스템을 구현하기 위한 자료구조가 왜 효율적인지 시간 및 공간복잡도를 통해 보이고, 설명하시오.

Linked list

검은색, 하얀색이(가) 표시된 사진

자동 생성된 설명

이름과 점수 정보를 가지는 구조체 노드를 생성하고 해당 노드의 포인터를 내부에 넣어주어서 linked list를 구현하였다. 전역변수로는 HEAD를 선언하여 테트리스를 시작하였을때 HEAD에 사용자 정보를 가지는 노드를 next를 이용하여 연결하고, 종료되면 새로운 profile 노드를 생성하여 정보를 저장 한 뒤, 다시 rank.txt 파일을 킴으로서 새로운 HEAD를 txt에 덮어씌운다.

createRankList()

스크린샷이(가) 표시된 사진

자동 생성된 설명

목적은 테트리스를 실행하였을 때 input 파일인 rank.txt에서 랭킹 정보를 읽여들여, 읽어들인 정보로 랭킹 목록을 생성하는 것으로, 먼저 랭킹의 개수를 읽어 rank\_c에 저장하고, 그 뒤로는 한 줄씩 읽으며 Linked List를 만들어주는 것이다

1. 먼저 HEAD를 profile 포인터로 할당해주고, fp를 이용하여 rank.txt 파일을 연다
2. 우선 랭킹정보의 개수를 읽어내어 rank\_c에 저장한다.
3. 긱 줄의 랭킹(이름, 정보)를 저장할 newprofile을 만든 뒤에 이름과 점수 정보를 입력하고 next를 null로 맞춘다.
4. 노드를 리스트의 마지막에 넣기 위하여 선언한 TAIL의 next에 newprofile을 연결하고 TAIL을 newprofile로 고친다. 이 때, 첫 TAIL은 HEAD로 맞추어 HEAD 의 next부터 넣는다.
5. 3,4의 과정을 rank\_c번 실행한다.
6. 테트리스 프로그램을 종료한다.

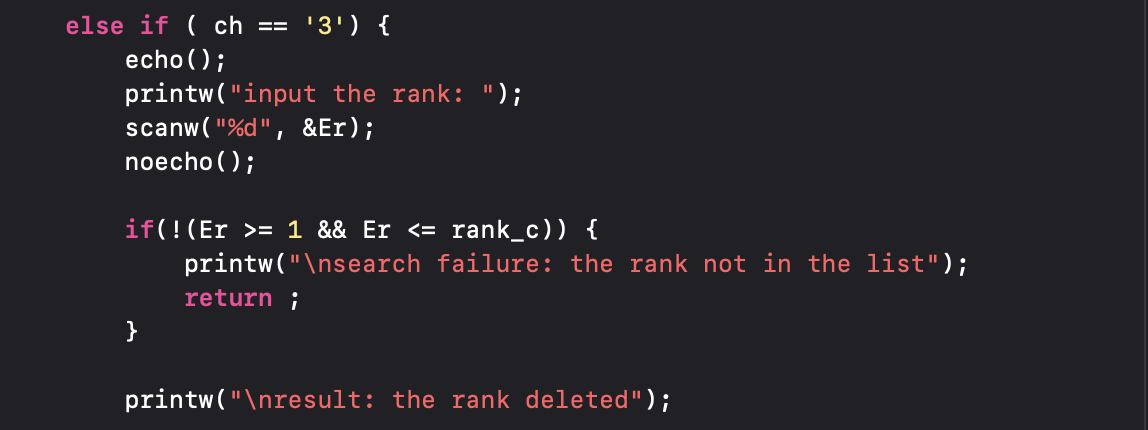
Rank()

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명



스크린샷, 검은색, 그리기이(가) 표시된 사진

자동 생성된 설명

Rank()는 원하는 랭킹정보를 읽고, 삭제해주는 함수이다.

1. 특정 범위의 랭킹 정보를 출력
2. X를 1로 초기화하고, Y를 랭킹 정보의 개수로 초기화해준다.
3. 시작 X와 끝 Y를 입력 받는다.
4. X>Y or 범위 안에 들어 가지 않으면 에러메세지를 출력한다. 입력이 없으면 X는 1이고, Y는 랭킹정보 개수를 갖는다
5. 리스트를 앞에서 읽을 temp를 HEAD로 설정하고 X만큼 next를 통해 이동한다.
6. temp가 출력할 처음 노드로 이동하므로 Y-X+1만큼 이름과 점수를 출력한다.
7. 입력하는 사용자의 랭킹 기록을 출력
8. 원하는 사용자의 랭킹기록이 있는지 확인하는 name\_s를 0으로 한다
9. 찾고 싶은 사용자의 이름을 name에 입력 받는다.
10. 리스트를 맨 처음부터 읽으며 원하는 사용자의 이름이 있는지 확인해줄 temp를 HEAD의 next로 설정하여 NULL이 될 때까지 temp의 name과 입력받은 name이 같을 경우엔 이름과 점수를 출력하고, name\_s를 1로 바꾼다. 다를 때는 temp를 다음으로 넘긴다.
11. 리스트를 다 돌고 난 뒤 원하는 사용자의 이름이 있었는지 확인하고, 없으면 에러 메세지를 출력한다.
12. 원하는 랭킹 정보를 삭제
13. 삭제하고 싶은 랭킹 번호를 입력받는다.
14. 입력 받은 번호가 1부터 rank\_c사이에 없으면 에러 메세지를 출력한다.
15. 입력 받은 번호 만큼 이동하여 삭제할 노드를 찾아간다.
16. 삭제할 노드의 앞뒤 노드를 연결해준다.
17. free를 통해 삭제할 노드의 메모리를 해제한 뒤, 삭제 문구를 출력한다.

writeRankFile()

스크린샷, 노트북이(가) 표시된 사진

자동 생성된 설명

writeRankFile() : 랭킹 정보가 추가되거나 삭제되 변경된 랭킹을 다시 rank.txt에 기록하는 함수

1. fp를 이용해 rank.txt를 w모드로 연다.
2. Temp를 처음 정보가 들어있는 HEAD의 next로 설정한다.
3. 파일에 랭킹 기록의 개수를 의미하는 rank\_c를 만든다
4. temp가 NULL이 될 때까지 temp내부의 이름과 점수를 기록하고, temp를 다음 노드로 보낸다.

newRank()

그리기이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

newRank() : Gameover 되었을때, 새로운 랭킹 정보를 작성하여 사용자의 이름을 입력받고, score와 함께 리스트의 적절한 위치에 저장

1. 화면을 지우고, 사용자의 이름을 입력받아 새로운 노드인 new\_profile의 name에 저장한다.
2. Score에는 게임이 끝났을 때 점수를 저장하고 next는 NULL로 설정한다.
3. 기존의 랭킹이 없을땐, HEAD의 뒤에 바로 new\_profile를 붙인다.
4. 새로운 랭킹 정보가 들어갈 위치를 찾기 위한 변수인 temp를 HEAD로 설정한다.
5. temp의 next가 존재하는지 체크한다.
6. 존재할때는 next의 score가 현재 게임의 score보다 작은지 확인하여 작지 않다면 temp를 다음 노드로 넘기고, 작다면 next앞에 현재 랭킹 정보가 들어가야 하기에 멈춘다.
7. 5,6을 새로운 랭킹이 들어가야 할 곳을 찾았거나 next의 정보가 없어 NULL까지 진행한다.
8. New\_profile의 next로 temp의 next를 설정하고, temp의 next는 new\_profile로 한다.
9. 랭킹이 늘어낫으므로 rank\_c를 1늘리고 wirteRankFile함수를 통해 rank.txt.파일에 새로운 랭킹을 기록한다.

링크드 리스트가 더 효율적인 이유

- 링크드 리스트를 이용하지 않으면 배열을 이용해야 하는데 이것은 새로운 랭킹을 추가하거나 삭제할 때, 기존의 랭킹 정보들을 한칸씩 뒤로 밀거나 앞으로 미는 과정이 필요하다. 만약 점수가 가장 높은 정보를 추가하면 모든 원소들을 하나씩 뒤로 밀어야 하고, 점수가 가장 높은 정보를 삭제하면 한칸씩 앞으로 밀어야 한다. 따라서 배열을 이용하면 시간복잡도가 O(n)으로 크다. 반면 링크드 리스트는 해당 위치에 링크만 연결해주면 되고 삭제할 경우도 링크만 바꾸면 되므로 O(1) 의 시간복잡도를 가저 더 효율적이다. 또한, 공간복잡도도 배열로 하면 랭킹의 정보의 개수를 알 수 없기에 랭킹이 하나씩 추가되어 범위를 넘을때마다 새로 선언해주어야 하므로 링크드리스트보다 메모리를 더 많이 차지하기에 LinkedList가 더 효율적이라 할 수 있다.

2. 본 실험 및 숙제를 통해 습득한 내용을 기술하시오.

- 이번 실험과 숙제를 통하여 여러가지의 자료구조 중 어떠한 것이 더 효율적인지 고려하는 중, 배열과 연결리스트의 차이를 잘 알게 되었고, 각각의 자료구조들을 어떠한 상황에서 이용해야 그 장점을 잘 활용할 수 있는지 알게되었던 것 같다. 새로운 printw와 같은 함수들도 알게 되었고, 이번 테트리스와 같은 상황에서는 배열의 단점을 보완하기 위해 힙을 이용하여 시간복잡도를 낮춰 더욱 더 효율적인 코드를 작성하는 방법을 배운 것 같다. 또한 LinkedList로 할 수 있는 많은 방식의 알고리즘 구현에 익숙해져 적응 하게 될 수 있었던 좋은 기회였던 것 같다.