

테트리스 2주차 결과보고서

전공: 철학과

학년: 3학년

학번: 20180032

이름: 남기동

1. 실습에 작성한 랭킹 시스템의 자료구조와 알고리즘

1) 실습에 작성한 랭킹 시스템의 자료구조와 알고리즘을 요약하여 기술

실습에서 작성한 랭킹 시스템의 자료구조는 '연결 리스트'이다. 파일로부터 정보를 읽어와서 새로 생성한 노드에 값을 대입하고 노드와 노드끼리 연결을 지어 연결 리스트로서 데이터를 저장한다. 실습에서 작성한 코드들 외에도 기능2, 기능3까지도 연결 리스트를 자료구조로 택하여 구현하였다. 실습에서 작성한 함수는 총 4개이다.

① void createRankList()

파일을 열어서 첫 줄에 있는 정보들의 수를 읽어서 score_number에 저장한다

그 다음 줄부터 이름과 스코어를 파일이 끝날 때까지 입력 받아서 각각의 노드에 저장한다.

이 때 위에서부터 랭킹이 높은 순으로 정렬되어 있기 때문에, 즉 스코어가 높은 사람 순으로 정렬되어 있기 때문에 연결 리스트에서도 새로 입력 받는 정보를 저장한 노드를 기존 연결 리스트의 끝에 잇는 방식을 택한다.

정보를 다 읽어 왔으면 파일을 닫고 함수를 종료한다.

② void rank()

X의 값은 디폴트값을 1로, Y의 값은 디폴트값을 score_number로 초기화하고 화면에 3가지 rank 옵션을 출력한다.

출력된 옵션 중 해당하는 번호를 입력 받아 기능을 구현한다

실습에서 구현한 기능은 3가지 중 첫 번째 기능이었는데, 이는 바로 아랫줄에 X와 Y의 값을 입력 받아 X나 Y에 대입한다. 이 때 값을 입력하지 않는 경우에는 scanw함수의 반환값이 -1이 되므로 이를 이용하여 디폴트값 그대로 유지되도록 한다.

X와 Y의 값이 적절한 값이면(X가 Y이하이고 X는 1이상이며 Y는 score_number이하라면)

노드 포인터 ptr을 head에서 이동시켜서 X와 Y 사이의 노드들의 이름과 스코어를 출력한다

만약 X와 Y의 값이 적절한 값이 아니면, 오류 메시지를 출력한다.

③ void writeRankFile()

파일을 읽기 모드로 연다

맨 첫 줄에 있는 파일의 정보 개수를 읽어와서 현재의 프로그램에서 저장하고 있는 연결 리스트의 노드 개수와 동일한지 체크한다

프로그램의 연결 리스트에 있는 노드의 개수와 같으면 파일을 종료하고 함수를 벗어난다

프로그램의 연결 리스트에 있는 노드의 개수와 다르면,

파일을 닫고 이번에는 읽기 모드로 열어서 현재 프로그램에 저장된 연결 리스트의 정보를 fprintf 함수를 이용하여 파일에 저장한다.

이 때 노드 포인터 ptr을 헤드를 가리키게 한 후 score_number만큼 ptr을 이동시키면서 노드를 조사하여 해당하는 노드의 이름과 스코어를 파일에 적으면 된다.

작업이 완료되면 파일을 종료하고 함수를 벗어난다.

④ void newRank()

Gameover이 되면 newRank함수를 호출하게 된다.

사용자의 이름을 입력 받아서, 새로운 노드를 만들고 사용자의 이름과 스코어를 저장한 다음

노드 포인터 ptr을 head를 가리키게 한 뒤, ptr이 연결 리스트의 마지막 노드의 링크인 NULL에 도달할 때까지, 즉 모든 연결 리스트의 노드를 조사하도록 이동시키면서, 사용자의 스코어가 위치할 부분의 노드를 찾아낸다.

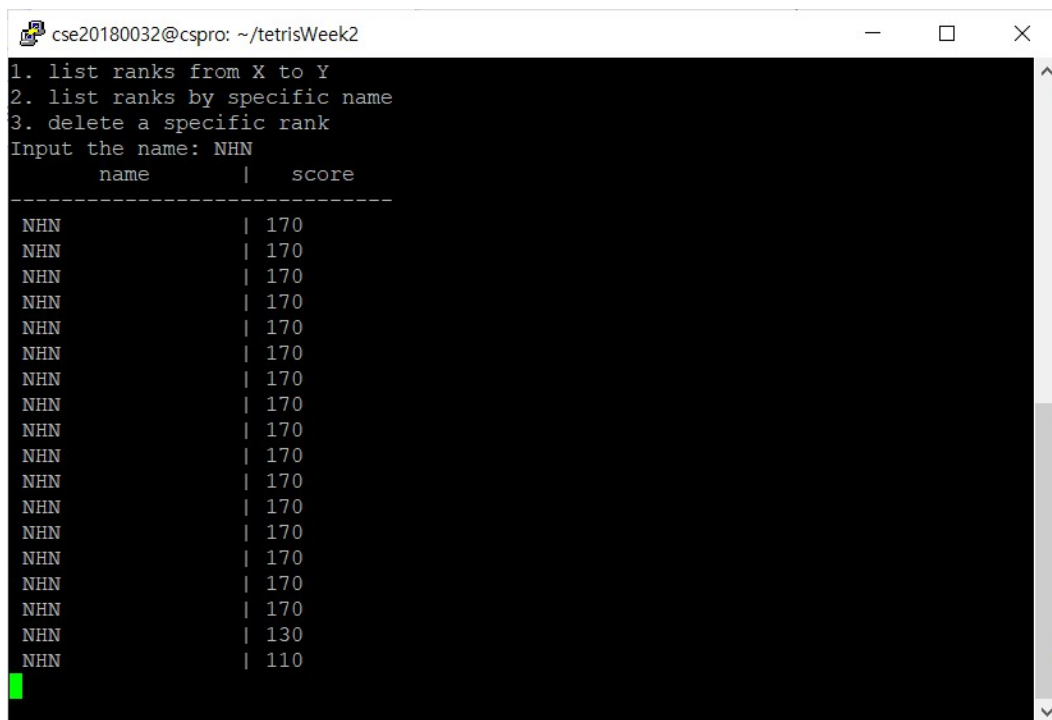
해당 부분의 노드 이전 노드를 사용자의 정보를 저장한 노드에 연결시키고 사용자의 정보를 저장한 노드를 해당 부분의 노드에 연결시켜서 기존의 연결리스트에서 랭킹에 맞게 정렬된 채로 새로운 노드를 기입하게 된다.

새로 저장된 노드는 writeRankFile 함수를 호출하여 rank.txt 파일에도 반영될 수 있도록 한다.

2) 본인이 선택한 자료구조의 효율성에 대한 평가

연결 리스트를 통해 랭킹 시스템을 구현했다. 연결 리스트를 자료구조로 선택하면 노드에 값을 저장하고 각 노드들끼리 연결 지어놓은 구조이기 때문에 값을 찾거나 값을 삭제하고 추가하는데 용이하다. 왜냐하면 값을 추가한다면 새로운 노드를 만들어서 링크를 연결시키면 되고 값을 삭제한다면 해당 노드를 빼고 주위 노드들의 링크만 손보면 되기 때문이다. 이를 시간 복잡도로 계산해보면 효율성에 대해 확실히 파악할 수 있다. n 개의 정보를 저장하고 있는 노드가 있을 때, 검색이나 추가, 삭제 어떤 기능이든 head에서부터 시작하여 NULL이라는 끝부분까지 n 개의 노드만을 조사하면 되기 때문에 시간 복잡도는 $O(n)$ 에 해당한다. 공간 복잡도 역시 n 개의 노드를 저장하는 공간 이외에는 상수이기 때문에 공간 복잡도도 $O(n)$ 에 해당한다.

3) 사용자의 이름을 입력하고 해당하는 랭킹 정보 출력 화면 첨부



```
cse20180032@cspro: ~/tetrisWeek2
1. list ranks from X to Y
2. list ranks by specific name
3. delete a specific rank
Input the name: NHN
  name | score
-----|-----
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 170
NHN    | 130
NHN    | 110
```

4) 사용자 이름으로 원하는 사용자의 이름을 검색할 때의, 시간 공간 복잡도

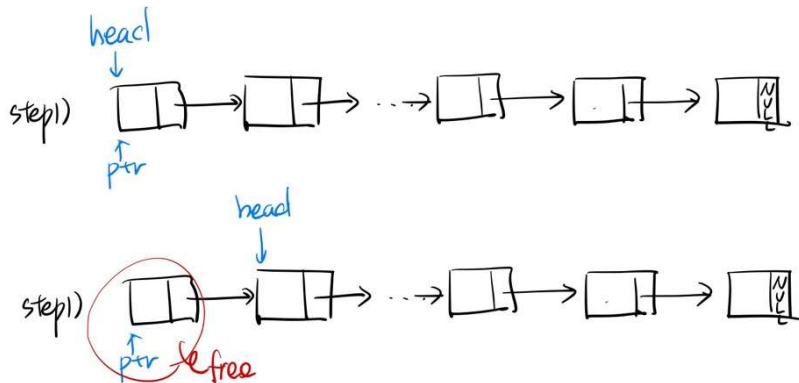
n 개의 정보, 즉 노드가 있다고 가정할 때 원하는 사용자의 이름을 검색하는 것은 연결 리스트에서 노드 포인터로 ptr을 설정하여 head에서부터 NULL에 도달하기까지 n 개의 노드를 조사하는 것이다. 그때 strcmp를 이용하여 입력받은 이름과 ptr이 가리키고 있는 노드의 이름이 동일하다면 화면에 출력하게 된다. 따라서 시간 복잡도는 n 개의 노드를 탐색하기 때문에 $O(n)$ 이다. 공간 복잡

도는 n개의 노드의 정보만이 변수로 활용되고 나머지는 상수이기 때문에 $O(n)$ 이다.

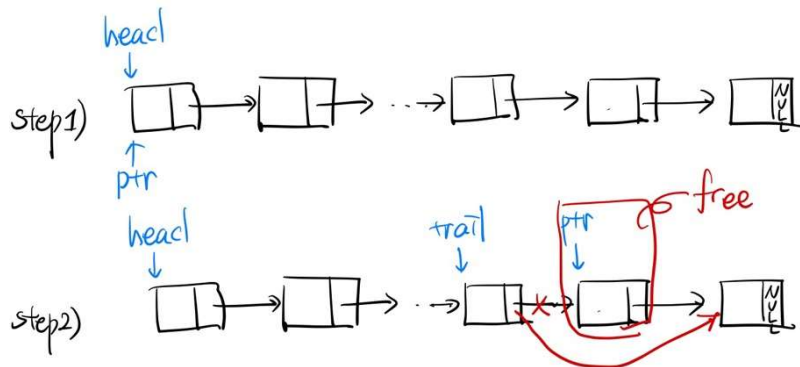
5) 선택한 자료구조에 맞춰 삭제 알고리즘을 그림으로 표현

① 입력받은 'rank'가 범위 ($1 \leq \text{rank} \leq \text{score_number}$)가 없을 때
`printf("search failure: no name in the list")`

② 입력받은 rank가 1일 때



③ 입력받은 rank가 2 이상일 때



2. 본 실험 및 숙제를 통해 습득한 내용 기술

이번 실험 및 숙제를 통해서 파일 입출력에 대해서 학습할 수 있었다. 기존의 실습들이 하나의 코드 안에서 내부적으로 기능을 구현한 것이었다면, 이번 실습에서 주로 했던 사항은 코드 외부의 파일의 값을 읽어오고 반대로 파일에 변화된 사항들을 기입하는 등 파일을 활용한 작업에 대해서 학습할 수 있었다.

또한 랭킹 시스템을 구현함에 있어서 어떤 자료구조가 효율적일지를 고민하는 과정을 가졌다. 단순히 정해진 자료구조를 통해 기계적으로 구현하는 것보다 시간 복잡도, 공간 복잡도라는 객관적인 지표를 통해 코드를 어떤 자료구조를 통해 구현했을 때 효율적일 수 있는지 탐구해볼 수 있었다. 이러한 학습은 코드를 기능적으로 작동시키는 것을 넘어서 코드가 시간적으로 빠른 시간 안에 작동할 수 있도록 기능을 구현하고 공간 복잡도 측면에서는 보다 적은 공간을 활용하여 기능을 구현할 수 있음을 몸소 체험할 수 있는 기회를 제공한 것 같다.

이외에는 ncurses 라이브러리라는 기존에 사용하지 않았던 라이브러리를 사용하여 새로운 공부, 학습이 되었고 연결 리스트를 자료구조로 선택하여 기능들을 구현하기 위해 고민한 결과, 연결 리스트의 삽입, 삭제, 검색 등의 기능을 자유자재로 사용할 수 있게 되었다.