실험 PRJ-1 2주차 기본 테트리스 프로그램 결과보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**1. 목적**

실험시간에 작성한 랭킹 시스템 함수들의 자료구조와 각 기능의 알고리즘을 요약하라. 또한 작성한 자료구조가 왜 효율적인지 설명하라. 마지막으로 본 실험 및 숙제를 통해 습득한 내용을 한 가지 서술하라.

**2. 랭킹 시스템 함수들의 자료구조와 각 기능의 알고리즘**

이번 실습에서는 이전 실습의 연장으로, tetris 게임을 수행한 이후, 정보를 저장하고 랭킹 시스템을 만드는 실습 · 과제를 진행했다. 총 4가지의 함수를 제작했는데, createRankList(), rank(), writeRankFile(), newRank(int score) 함수이다.

첫 번째로 createRankList() 함수는 rank.txt file에 저장되어 있는 rank 정보를 불러오는 함수이다.(Linked List 형태로 구현해주는 함수이다.) 이 함수에서는 rank.txt의 첫 번째 줄을 읽어들여, 만약 첫 번째 줄에 저장되어 있는 숫자==0 이라면 별다른 동작을 수행하지 않지만, 0이 아닌 다른 수라면, 그 수만큼 파일을 읽어서 Linked list를 구현한다. 이때 파일을 읽음과 함께, malloc과 nextpoint로의 이동을 반복적으로 수행해야 한다.

두 번째로 rank() 함수는 크게 3가지의 역할을 수행한다. 첫 번째로는 X~Y만큼의 rank를 출력해주며, 두 번째로는 이름을 input으로 받아 동일한 이름에 대한 점수를 print 해준다. 마지막으로는 일정 ranking에 있는 정보를 제거해준다. 여기에서 첫 번째와 두 번째 역할은 이미 존재하는 linked list에 대해서 조건에 맞는 정보(x~y의 랭킹일 경우, 이름이 input과 동일할 경우)에 대해서 linear로 읽는 형태가 되며, 세 번째 역할 또한 linear로 읽어가며, 지워야 하는 랭킹의 이전 노드의 nextpoint를 지워야 하는 랭킹의 다음 노드로 연결하며, 지워야 하는 노드를 free 시켜주는 형태이다.

세 번째로 writeRankFile() 함수는 현재까지의 node의 변경사항을 rank.txt file에 저장시켜주는 함수이다. Linked list를 linear 형태로 현재 랭킹의 총 수만큼 읽어가며, 파일에 알맞은 형식으로 저장하면 된다. 여기에서 각 노드를 저장한 이후 free를 해주어야 한다.

마지막으로 newRank(int score) 함수는 각 tetris play가 종료된 이후, 사용자의 이름을 받아 데이터를 malloc을 통해서 하나의 node로 만들어준 이후, 알맞은 rank의 위치에 저장하는 함수이다. 이미 Linked list에 존재하는 score과 비교한 이후, score이 비교 node의 score보다 크거나, 같고 이름이 사전적으로 더 선행할 때 이미 존재하는 node의 앞에 위치하도록 코드를 만들었다. 이 함수에서 중요한 것은 node가 rank의 가장 위에 위치할 때에는 headptr(항상 첫 번째 node를 point하는 변수)를 새로운 node로 pointing 해주어야 하며, 이외의 경우에는 그대로 유지해야 한다는 것이다.

+추가적으로 데이터를 삽입, 삭제할 때 그 삽입, 삭제하는 위치의 이전, 이후 node의 연결을 시켜주는 형태로 구현하면 된다.

**3. 작성한 함수들의 시·공간 복잡도**

이번 실습에서 작성한 함수들의 복잡도는 아래와 같다.

-createRankList()

시간 복잡도: O(node\_num), input으로 받은 node의 개수만큼 file을 읽어야 하기 때문에 시간 복잡도는 위와 같다.

공간 복잡도: O(sizeof(Node)\*node\_num), malloc을 통해서 node들을 잡아준다.

-rank()

시간 복잡도: O(node\_num), 1, 2, 3번의 메뉴에서 모두 node\_num만큼 Linked list를 읽어내야 되기 때문에 위와 같다.

공간 복잡도: O(1), 함수에서 별다른 공간을 사용하지 않기 때문에, 상수가 된다.

-writeRankFile()

시간 복잡도: O(node\_num), Linked list를 linear로 읽으며, 파일에 저장한다.

공간 복잡도: O(1), 별다른 공간 할당이 변수 이외에는 존재하지 않기 때문에, 상수이다.

-newRank(int score)

시간 복잡도: O(node\_num), Linked list를 linear로 읽어낸다.

공간 복잡도: O(1), malloc(sizeof(Node))를 수행하지만, 고정된 크기이므로 상수가 된다.

**4. 실험 및 숙제를 통해서 습득한 내용**

이번 실험 · 실습에서 처음에 rank.txt file을 통해서 Linked list를 만들어낼 때, 처음에 구현을 하지 못했다. 그 이유로는 nowptr = (Node\*)malloc(sizeof(Node)); nowptr->nextpoint = NULL; 을 실행한 이후, 그 다음 node를 만들어낼 때, nowptr = nowptr->nextpoint; nowptr = (Node\*)malloc(sizeof(Node)); 형태로 진행했다. 하지만 이렇게 할 경우에 NULL에 동적할당을 하는 형태가 되어서 제대로 Linked list를 만들지 못했다. 따라서 nowptr->nextpoint 자체에 동적할당을 해주는 형식으로 문제를 해결할 수 있었다.

또한 ncurses 라이브러리의 작동에 대해서 이해할 수 있었다. noecho()와 echo()를 통해서 printw, getch이 바로 보이거나, refresh()를 사용해야 화면에 보일 수 있도록 하는 형태에 대해서 알 수 있었고, 이를 통해서 다양한 형태의 output을 구현할 수 있었다.