7주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

1. <자료구조의 효율성>

예비 레포트에서 힙 자료구조 또한 랭킹 시스템을 구현하기 괜찮은 자료구조라고 설명을 했었지만, 구간 랭킹 출력 기능을 사용할 때 힙 전체를 복사해야 하는 과정이 필요했고, 이러한 이유때문에 공간복잡도가 linked list보다 더 크게 계산된다. 따라서 전체 복사과정이 필요하지 않는 linked list를 사용하는 것이 더 효율적이다. 힙 자료구조는 삽입과 삭제 과정이 매우 빈번한 상황에서 사용하기 효율적인 자료구조이다. 그러나, 테트리스 프로젝트에서는 삽입과 삭제과정이 빈번하기 보다는 구간 랭킹을 확인하는 과정이 더 자주있을거라는 생각을 하였다. 이러한 측면에서 생각했을때 linked list가 더 효율적이다. 또한, 삽입, 삭제, 구간 랭킹 출력, 이름 탐색 기능 모두 linked list의 길이가 len이라면 시간복잡도 O(len)이 되어 선형 시간복잡도를 가진다. 사용자가 이용하는데는 어떤 상황에서는 아주 빠르고, 어떤 상황에서는 아주 느린 코드는 부적합하고, 너무 느리지 않으면서 정보의 양에 따라 시간복잡도가 아주 급격하게 변화하지 않는 알고리즘으로 구현하는 것이 중요하다. Linked list는 선형 시간복잡도를 가지므로, 이러한 특성을 모두 가지고 있는 자료구조이다. 또한, 공간복잡도는 각각의 기능에 대해서는 O(1)의 공간복잡도를 가지고, 활용하는 linked list의 공간복잡도가 O(len) 이므로 이 또한 충분히 효율적이라고 이야기할 수 있다.

<사용한 자료구조 설명>

실험시간에 랭킹 시스템을 구현하기 위한 자료구조로 Linked List를 선택하였다. 이 자료구조는 글로벌 변수로 정의되어 rank를 확인하는 메뉴를 들어갔을때 linked List 를 생성하고, 그 이후 구간 랭킹 출력이나 삽입, 및 삭제과정을 진행할때 생성된 linked list를 받아오는 과정으로 프로그램이 진행되도록 코드를 작성하였다.

<createRankList 함수>

Rank.txt 파일에 저장되어 있는 정보를 받아와서, 정보를 활용하기 편하도록 linked list 형태로 변환하는 함수이다. 먼저, 파일이 존재하지 않으면 score\_number을 0으로 설정해주고 함수를 종료한다. 파일이 존재한다면, 가장 위에 적힌 숫자는 score\_number를 의미하기 때문에 그 숫자를 fscanf함수를 통해 받아와서 score\_number로 저장한다. 그리고 sn =score\_number 라는 줄을 통해 writeRankFile 함수에서 파일을 새롭게 생성해야 하는지 아닌지를 판단한다. 다음 줄부터 score\_number 수만큼 줄이 있는데, 각 줄은 이름, 점수 순으로 정보가 저장되어 있다. 이 정보들을 반복문을 통해 접근하여 각각의 이름과 점수를 하나의 노드에 저장하여 linked list의 하나의 원소로서 끼워넣는다. 파일에서 EOF를 만나면 더 이상의 정보가 없는 것이므로 반복문을 빠져 나오고 fclose함수를 통해 파일을 닫아준다.

<rank 함수>

이 함수에서는 입력에 따라 하는 행동이 다르다. 1을 입력한다면 구간 랭킹 출력 기능, 2를 입력하면 이름 검색을 통한 정보 출력 기능, 3을 입력하면 정보 삭제 기능이 실행된다. 실습에서는 이 3가지 기능 중 구간 랭킹 출력 기능만 구현하였고, 숙제에서 이름 검색을 통한 정보 출력 기능, 정보 삭제 기능을 구현하였다.

함수를 시작하자마자 clear함수를 통해 화면을 깨끗이 한 다음, createRankList 함수를 불러와서, 현재 rank.txt 파일과 linked list의 정보를 동기화한다. 그 이후, 1을 입력하면 먼저 시작랭킹과 끝 랭킹을 입력할 수 있는 화면이 출력된다. 시작 랭킹과 끝 랭킹을 입력하면 rank.txt상에 저장되어 있는 랭킹 기준으로 시작랭킹부터 끝 랭킹까지 이름과 점수를 모두 출력한다. 시작 랭킹이 입력되지 않았으면 1등을 의미하고, 끝 랭킹이 입력되지 않으면 가장 끝 등수를 의미하도록 하였고, 만약 범위에 벗어나는 등수가 입력된 경우 입력된 랭킹의 범위와 rank.txt 상에 있는 등수들의 교집합을 출력하도록 하였다. 즉, 교집합이 존재하지 않게 입력이 된다면 “search failure : no rank in the list”라는 에러 메세지만 출력하였다. 또한, 시작 랭킹이 끝 랭킹보다 크거나 같아도 “search failure : no rank in the list” 라는 에러 메세지만을 출력하였다.

<writeRankFile 함수>

현재 linked list정보를 rank.txt에 업데이트 시켜주는 함수이다. 먼저, fopen함수를 w+모드로 열어준다. 만약 sn == score\_number이면 업데이트할 필요가 없다는 의미이므로 함수를 바로 종료한다. 그렇지 않다면, fprintf함수를 통해 파일에 정보를 입력해주는데, 가장 윗줄에 score\_number을 입력해주고, 그 아랫줄부터 차례로 linked list의 정보를 탐색하며 이름, 점수 순으로 정보를 fprintf 함수를 통해 입력해준다. 모든 과정이 끝났다면, fclose함수를 통해 파일을 닫아준다.

<newRank 함수>

게임이 종료되고 나면 새로운 정보를 입력할 수 있는 창이 뜨도록 구현하였다. 이 함수또한 rank 함수와 마찬가지로 createRankList함수를 호출해 현재 rank.txt 정보와 Linked list 정보를 일치시키는 과정을 먼저 진행한다. 그 이후, 이름을 입력하면 linked list 에 점수 내림차순으로 위치를 찾아서 새로운 정보를 삽입되게 하였다. 위치를 찾기 위해서는 linked list의 첫번째 노드부터 끝 노드까지 탐색하며 현재 새로운 정보의 점수가 탐색중인 정보의 점수보다 크거나 같게 되는 상황을 찾았다면, 새로운 정보의 링크를 현재 탐색중인 정보의 링크가 되게 한 후, 이전 노드의 링크를 새로운 정보가 되게 한다. 그런데 현재 노드가 가장 첫 노드인 상황은 이전 노드가 없으므로 새로운 정보의 링크가 첫 노드가 되게 한 후, 새로운 정보가 첫 노드가 되게 하였다. 즉, 삽입되는 위치가 첫번째 위치인지, 아닌 지에 따라서 두가지 경우로 나뉘어서 삽입과정을 진행하게 된다. 그리고, 만약 rank.txt 파일이 없는 상태에서 새로운 정보를 삽입하려고 할 때는, 먼저 rank.txt 파일을 생성한 후 정보를 위의 과정을 통해 삽입하였다. 삽입 과정을 마친다음 score\_number을 1 증가시키고, writeRankFile함수를 호출해 현재 linked list정보를 rank.txt에 업데이트한다.

1. <숙제 및 실습에서 배운 점>

이번 실험을 통해 linked list 자료구조의 삽입 및 삭제과정에 대한 개념을 더 확실히 잡을 수 있었다. 이전에는 삽입과 삭제과정이 상황에 따라 다르게 구현해야 한다는 정도만 알고 있었는데, 이번 실험에서 삽입 및 삭제과정의 대상이 첫번째 노드인지 아닌지에 따라 두가지 경우로 나뉜다는 것을 알게 되었다. 또한, 실습과 과제를 진행하면서 segmentation fault 오류를 가장 많이 맞닥뜨렸었는데, 항상 그런것은 아니지만 많은 경우 NULL 포인터의 멤버변수에 접근하거나 값을 할당할 때 발생한다는 것을 알게 되었다. 오류의 의미를 파악하는 것이 디버깅을 효율적으로 할 수 있는 최선의 방법이라는 것을 깨달았던 실험이었다.

<이름을 통한 랭킹 정보 출력 화면>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 1]

[사진 1]은 현재 rank.txt에 저장되어 있는 모든 사용자를 출력한 화면이다. 아무입력도 주지 않아서 1등부터 마지막등수까지 출력하도록 입력을 주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 2]

[사진 2]는 rank.txt상에 존재하는 이름을 입력하였을 때 일치하는 모든 정보를 출력해주는 화면이다. 현재, AA라는 이름을 가진 정보가 총 4개 있어서, 이들이 모두 출력되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 3]

[사진 3]은 rank.txt 상에 존재하지 않는 이름을 입력하였을 때, 에러 메세지가 출력되는 것을 확인할 수 있는 화면이다. 현재, FF라는 이름을 가진 정보는 하나도 없으므로, “search failure: no information in the list”라는 메세지가 출력되었다.

1. <이름 검색 기능의 시간 및 공간 복잡도>

이름을 입력해서 해당 이름을 가진 플레이어를 모두 찾아서 이름과 점수를 출력해주는 기능을 구현하였다. 이는, 첫번째 노드부터 끝 노드까지 순서대로 탐색하며, 현재 입력한 이름과 탐색중인 노드의 이름이 같다면 정보를 출력하고, 이름이 같지 않다면 아무 행동도 하지 않는다. 만약, 모든 노드를 탐색하였는데 입력한 이름과 일치하는 정보가 하나도 없다면 “search failure: no name in the list” 라는 에러메세지를 출력하도록 구현하였다. 모든 노드를 하나씩 탐색하는 과정이 있으므로 linked list의 길이를 len이라고 하면 시간복잡도는 O(len)이다. 또한, 이름 검색 함수 내부에서는 탐색을 위한 현재 노드와 이전 노드를 정의하므로, 이는 O(1)의 공간복잡도를 가진다. 글로벌 변수로 정의된 linked list의 공간복잡도는 O(len)이다.

1. <삭제 알고리즘에 대한 설명>

랭킹 정보를 입력하여서 해당 하는 랭킹이 존재한다면 삭제하는 기능이다. 따라서 현재 n개의 사용자 정보가 존재한다면, 입력가능 범위는 1부터 n까지이다. K를 입력하면 k등 정보를 삭제하라는 의미이다. 이는 첫번째 노드부터 차례로 링크를 따라가 K번째 노드를 찾는다. 그리고 이전 노드의 링크를 현재 노드의 링크가 되게 하고, 현재 노드의 메모리를 해제해준다. 만약 첫번째 노드를 삭제해야하는 입력이 주어진다면, temp 변수에 첫번째 노드를 저장하고, 첫번째 노드를 첫번째 노드의 링크가 되게 하고, temp의 메모리를 해제해준다. 삭제 기능또한, 삽입과정과 마찬가지로 삭제해야 하는 노드의 위치가 첫번째인지 아닌 지에 따라서 두가지 경우로 나뉘어서 진행되게 된다. 만약 입력가능 범위를 벗어난 입력이 주어진다면 “search failure: the rank not in the list” 라는 에러메세지를 출력하도록 구현하였다.

<삭제 알고리즘에 대한 그림>

파워포인트를 이용하여 그림을 그려서 캡처하여 첨부하였다.

