9주차 결과보고서

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

랭킹 시스템을 작성하기 위해 사용한 자료구조는 linked list이다. 점수를 저장하는 rank\_score, 사용자의 이름을 저장하는 rank\_name[NAMELEN], 그리고 구조체를 가리키는 \*link 변수를 list라는 구조체에 생성한다. 그리고 linked list를 가리키는 포인터 top을 생성한다. 이후 createRankList() 함수에서 rank.txt라는 파일을 열고, 파일을 성공적으로 열었을 때 가장 먼저 랭킹의 개수를 score\_number에 저장한다. 개별 점수와 이름을 저장할 때는 temp라는 list를 만들어 사용자의 이름과 점수를 저장하고, top이 가리키는 list의 점수보다 temp의 점수가 더 크면 temp의 link가 top을 가리키고, top이 temp를 가리키도록 한다. 그렇지 않은 경우에는 list\* current = top, list\* previous = NULL로 초기화 한 뒤 temp의 점수가 current의 점수보다 작으면 current와 previous를 다음 list로 이동시키고, temp의 점수가 더 크면 그 위치에 temp를 삽입한다. 이 과정을 score\_number의 크기만큼 반복하고 파일을 닫는다.

rank() 함수에서는 wgetch로 메뉴를 선택하고, wgetstr로 x와 y의 값을 입력 받는다. x, y 값이 string이기 때문에 이를 정수로 바꿔주는 작업을 수행하고, x가 입력되지 않으면 x를 1로, y가 입력되지 않으면 y를 score\_number로 설정한다. x>y인 경우에는 에러 메시지를 출력하고, 그렇지 않다면 x, y에 해당하는 랭킹을 출력한다. 이 때 list\* now = top, count = 1으로 초기화 한 뒤 now가 NULL이 아니고, count가 y보다 작거나 같고, count가 x보다 크거나 같으면 now의 rank\_name과 rank\_score를 출력하고 count의 값을 증가시키고, now를 다음으로 이동시킨다.

writeRankFile() 함수는 변경된 랭킹을 rank.txt 파일에 저장하는 함수이다. 먼저 파일을 열고, score\_number를 파일에 작성한다. list\* current = top으로 초기화 한 뒤 current가 NULL이 아닌 동안에 current의 rank\_name과 rank\_score를 작성하고 current를 current의 link로 이동시킨다. 반복문이 끝나면 파일을 닫는다.

newRank() 함수는 게임이 종료되면 사용자의 이름을 입력받아 랭킹에 저장하는 함수이다. 먼저 사용자의 이름을 wgetstr로 입력받는다. 이후 temp라는 list를 생성하고, 여기에 사용자의 이름과 점수를 저장한다. score\_number가 0이 아니면 list \*current = top, list \*previous = NULL로 초기화 하고, createRankList() 함수와 동일한 작업을 반복한다. temp가 삽입될 위치를 current, previous를 통해 찾고, 적절한 위치에 temp를 삽입하는 작업이다. score\_number가 존재하지 않으면 top이 temp를 가리키고 score\_number는 1이 된다.

linked list를 사용하면 랭킹을 삭제, 삽입하거나 랭킹을 불러오고 저장할 때의 시간 복잡도와 공간 복잡도는 O(n)이다. 하지만 예비보고서에서 작성했던 바와 같이 구조체 배열을 사용한다면 특정 랭킹을 삭제하거나 삽입할 때 공간복잡도는 비슷하겠지만, O(n^2)의 시간 복잡도가 발생하기 때문에 linked list를 사용하는 것이 더욱 효과적이다. 이 때 구조체를 사용하여 랭킹을 삭제하거나 삽입하기 위해서는 이중 for loop이 실행되어야 하기 때문에 시간 복잡도가 O(n^2)가 된다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

사용자의 이름을 검색하는 경우에 코드는 다음과 같다.

list \*now = top;

*while* (now != NULL)

        {

*if* (strcmp(now->rank\_name, str) == 0)

            {

                printw("%-16s| %d\n", now->rank\_name, now->rank\_score);

                check++;

            }

            now = now->link;

        }

*if* (check == 0)

        {

            printw("\nsearch failure: no name in the list\n");

        }

위 코드에서 while문이 사용되고 있고, 이 while문은 linked list의 크기에 따라 결정되기 때문에 시간 복잡도는 O(n)이며, 공간 복잡도 역시 linked list 크기에 따라 결정되므로 O(n)이다.

linked list를 사용한 삭제 알고리즘을 그림으로 표현하면 다음과 같다.

텍스트, 스크린샷, 그래픽 디자인, 폰트이(가) 표시된 사진

자동 생성된 설명

**2.**

이번주차 실습과 과제를 통해 linked list 사용해 익숙해졌다. 이번 주차 실습을 처음 진행했을 때 linked list를 사용하면서 segmentation fault가 자주 발생하였는데, 실습을 진행하면서 segmentation fault가 발생하지 않도록 linked list를 수정할 수 있었다. 특히 temp를 생성하고 temp를 적절한 위치에 삽입하는 데 어려움이 있었는데, current와 previous를 이용하여 이를 해결할 수 있었고, current와 previous를 어떤 방식으로 초기화해야 하는지 배울 수 있었다. 또한 ncurses 라이브러리를 사용하는 데 익숙해질 수 있었다. wgetch, wgetstr, scanw, echo, noecho 등 ncurses 라이브러리의 함수를 적절하게 사용하는 법을 배울 수 있었다. 마지막으로 기본적인 c언어 문법에 대해 복습할 수 있었던 시간이었다. 파일을 열고 닫는 fopen과 fclose, printf 포맷 등에 대해 복습할 수 있었고, 특히 파일을 여는 과정에서 파일이 정상적으로 열리지 않는 경우 예외 처리하는 것에 대해 다시 공부할 수 있었다.