7주차 결과보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

1-1.

Tetris.h 파일에서 추가 및 수정 사항들이다.

#define MENU\_RANK '2' //menu rank를 2로 정의한다

Define을 통해 menu rank 2를 사용할 수 있게 해주었다.

typedef struct \_scorenode{ //scorenode linkedlist 만들기 위해

char rankname [NAMELEN]; //랭킹 이름을 저장

int rankscore; //랭킴 점수를 저장

struct \_scorenode\* link; //link도 추가하여 다음 노드를 이어질 수 있도록

}scorenode;

scorenode\* top=NULL; //linkedlist top 설정 (가장 Rank 높은 곳)

int ranknumber=0; //초기 Ranknumber를 글로벌 변수로 둠

본 실험은 linkedlist 를 이용하여 랭킹시스템을 구현하였다. scorenode struct 구조체를 만들어 그 안에 rankname, rankscore 와 다음 노드와 연결해주고 또 연결을 끊어줘 해당 랭킹시스템이 구현되도록 link 를 추가하였다. 이후 scorenode 중에서 가장 rankscore가 가장 높은 노드를 지칭하는 top을 초기값 null로 두었다. rank.txt의 출력, 입력, 삭제 등을 할 때는 탐색이 필수적으로 필요한데 앞으로 이 탐색을 시작할 때 top부터 시작할 것이다. 또한 ranknumber는 node의 개수를 의미하는 변수이다. 이 역시 초기값 0으로 두었다.

다음은 rank.txt 를 열어 해당 랭킹 정보를 가져와 linkedlist의 각 node에 저장하는 createRankList함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

주석에 달린 것처럼 우선 rank.txt를 fopen을 통해 열어 온다. 이때 fp==NULL 인지 체크한다. 만약 NULL 이라면 rank.txt 라는 파일이 존재하지 않는 것이다. 때문에 이를 해결하기 위해 fopen(“rank.txt”,”w+”)를 통해 rank.txt 파일을 새로 생성해준다. 이때 새로 생성된 파일은 아무 것도 없기 때문에 첫 줄에 ranknumber라고 할 수 있는 ‘0’을 넣어준다. 그리고 rank.txt에서 받아올 정보인 name, score 라는 변수, tetris.h 에서 선언했던 top을 다시 NULL로 초기화 시켜주고(createRankList를 불러올때마다 초기화를 시키기 위해서이다.) 새로 prev라는 노드를 만들어 linkedlist 탐색에 사용할 것이다. 이후 fscanf(fp, “%d”, &ranknumber) 를 통해 ranknumber에 rank.txt의 가장 첫 줄인 랭킹시스템에 몇명이 있는지 가져온다. 만약 이게 받아지지 않고 비었다면 move(50,0)에 printw를 통해 데이터가 없다고(“No data”) 에러 메세지를 출력했다. 있다면 저장된 ranknumber 보다 1작은 수만큼 for문을 돌며(i=0 부터 시작) 이제 그 rank.txt의 다음 줄 인 1등부터 순차적으로 rank와 score를 fscanf로 받아오고 새로 생성한 newnode에 저장한다. 이름은 strcpy(newnode->rankname,name)로 붙여 넣고 score는 newnode->rankscore=score로 넣는다. 처음 이 함수가 실행될 때는 당연히 top이 NULL 일 것이다. 그래서 우선 첫 for문에서 받아온 newnode를 top으로 선언하고 linkedlist 탐색 및 추가에 사용되는 prev도 top으로 둔다. 그리고 top->link는 NULL로 초기화해준다. 2등이 들어올 시 이제 top이 null이 아니다. 때문에 top(prev)->link=newnode로 설정하여 2등으로 만들어주고 prev를 newnode로 설정하여 top은 그대로 있고 prev는 2등이 된다. 이 과정을 ranknumber만큼 반복하면 정확히 ranknumber 개수 만큼의 linkedlist의 node가 rank 순으로 정렬되어 있을 것이다. (top은 가장 상위 랭커이다.) 이후 fopen으로 열었던 rank.txt를 fclose로 닫았다.

실험 시간에 작성한 두 정수를 입력해 X~Y 순위를 출력하기 위한 rank 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 rank 함수는 처음에 createRankList()를 호출하여 해당 rank.txt에 들어있는 정보들을 linkedlist의 node들에 넣고 연결시킨다. 변수 X와 Y는 ch==’1’ 에서 범위를 나타낸다. 아무것도 받아지지 않았다면 그냥 X=1, Y=ranknumber로 X는 1등을 의미하고 Y는 마지막 등수를 의미한다. 이후 clear()를 한번 해줘 화면을 지워준다. 그리고 랭킹 시스템에서 실험에서 진행한 1번, 과제로 진행한 2,3 번에 대한 내용을 printw를 통해 화면에 출력할 수 있도록 해준다. 이후 wgetch(stdscr)를 통해 숫자 하나를 입력 받는다. 이 숫자를 통해 세부 랭킹 시스템의 내용이 달라진다.

1번은 실험에 진행한 X,Y 값을 입력받고 범위를 벗어나지 않았다면 X등부터 Y등 까지 이름, 점수를 출력하게 된다. echo() 는 사용자가 문자나 숫자를 입력했을 때 화면에 보이도록 해주는 것이다. 이 기능을 켜고 scanw를 통해 X,Y 를 입력 받는다. 그리고 다시 noecho()로 화면에서 안보이도록 원래대로 해주고 실험에서 요구하는 화면 디자인을 printw를 통해 출력해주었다.

그 다음 X,Y 값의 범위를 확인해주는 과정이 필요하다. 우선 X가 Y보다 크면 당연히 점수가 더 높은 사람이 더 낮은 숫자에 위치해야 되는데 오류이다. 또한 새로 rank.txt를 생성했을 때 ranknumber가 0일 경우가 있다. 이때는 X,Y가 어떤 수가 오든 오류이다. 그리고 X 또한 ranknumber 보다 커버리면 출력할 것이 없기 때문에 오류이다. 그래서 printw를 통해 “search failure”를 출력해주었다.

범위가 적절히 들어왔다면 처음에 X나 Y가 아무 값도 받아지지 않았는지 확인해보고 값이 받아지지 않았다면 X=1, Y=ranknumber로 그대로 둔다. 또한 X 값이 1보다 작은 경우에도 1위부터 출력해야 하고 Y값이 ranknumber 보다 큰 경우에도 결국 최대 마지막 순위인 ranknumber 까지 출력해야 되기 때문에 이때도 X=1, Y=ranknumber로 설정 하였다. 이후 currentnode를 새로 만들어 linkedlist를 탐색할 수 있게 한다. top부터 순차적으로 node를 확인하고 적절한 위치에 왼쪽으로 붙여서 출력할 것이다. 탐색에 필요한 반복문인 for문은 최대 Y번(i=1부터 시작)까지만 하면 될 것이다. 만약 i가 X 보다 작다면 해당 랭킹은 출력할 필요가 없다. 그래서 currentnode=currentnode->link로 다음 노드로 설정을 한다. 만약 i가 X보다 크거나 같다면 그때부터 Y까지 printw를 통해 currentnode->rankname, currentnode->rankscore를 출력한다. 시간복잡도의 최악의 경우는 X=1, Y=ranknumber로 총 ranknumber만큼 탐색 및 출력할 때일 것이다.

테트리스가 끝날 때 linkedlist에 들어있는 node 내용들을 rank.txt에 다시 기입하는 writeRankFile함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 createRankList()와 달리 여기서는 rank.txt에 읽어 들이는게 아니라 다시 기입해야 하기 때문에 fopen( “rank.txt”,”w”)로 열어준다. 그리고 가장 첫 줄에 적힌 ranknumber를 읽어들여 sn에 저장한다. 만약 sn이랑 ranknumber가 일치하다면 추가적으로 기입할 필요가 없다. 만약 다르다면 다시 기입을 해줘야 한다. 이때 탐색에 사용될 currentnode를 top으로 두었다. 이미 newRank같은 곳에서 새로 rank가 들어왔다면 최소한 노드는 1개가 있어야 하기에 top은 무조건 존재 해야 한다. 만약 top이 NULL 이면 에러이기 때문에 exit(-1)를 해주었다. 아니면 이제 기입을 시작한다. 우선 바뀐 ranknumber를 rank.txt 첫 줄에 fprintf로 입력을 해주고 While문에서 currentnode가 null이 아니라면 currentnode->rankname, currentnode->rankscore를 순서대로 기입을 해준다. 그리고 currentnode=currentnode->link로 하여 다음 노드로 설정 해준다. Currentnode가 null이 되었다면 거기서 break를 한다.

이후 이미 새로운 정보를 rank.txt에 저장했기 때문에 linkedlist의 노드들이 필요 없어진다. 그래서메모리 절약을 하기 위해 top, currentnode를 이용하여 while문에서 top이 NULL이 아닐때 까지 처음 currentnode=top으로 두고 top을 다음 노드로 순차적으로 이동시키며 currentnode를 free 해준다.

게임이 끝나고 사용자의 이름, 점수를 받아 이를 node에 넣어 linkedlist에 점수에 따라 적절한 위치에 넣어주고 writeRankFile을 호출 하여 rank.txt에 기입까지 해주는 역할을 하는 newRank 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 처음 createRankList()를 호출하여 현재 linkedlist의 정보를 가져온다. 또한 echo()로 scanw(“%s”,str) 에서 사용자가 입력한 이름이 보이도록 해준다. 이후 다시 입력이 안보이도록 noecho()로 설정한다. 여기서는 currentnode와 prevnode를 통해 linked list 탐색을 할 것이다. 우선 currentnode=top으로 설정해두고 top부터 순차적으로 탐색한다. 다음은 newnode를 만들어서 여기에 input으로 받아오는 score, scanw 입력으로 받아오는 name 을 strcpy(newnode->rankname, str), newnode->rankscore=score 로 newnnode에 기입을 한다.

첫 if문은 top이 비었는지 비어있지 않은지 확인하는 곳이다. 만약 비었다면 ranknumber=0이라는 의미이다. 때문에 새로운 newnode가 top이 된다. 또한 top->link는 NULL로 설정해준다. Top이 비어있지 않다면 newnode의 rankscore가 top의 rankscore 보다 큰지 확인해주었다. 만약 크다면 newnode가 top이 되어야 한다. 그러면 newnode->link=top으로 하고 top을 newnode로 해 top을 변경해준다. 그렇지 않다면 top을 변경할 일이 없어진다. 그래서 while 반복문에서 Linkedlist 탐색을 시작하는데 처음은 가장 아래 else문으로 간다. 여기서 prev=currentnode로 두고 currentnode를 하나 앞의 Node로 설정한다. 즉 prev는 currentnode 이전의 node라고 생각하면 된다.

이때는 경우의 수가 2개가 있는데 우선 currentnode가 NULL일 경우이다. 즉 새로운 newnode의 rankscore가 가장 작아 마지막 node와 비교했을 때도 더 작을 때이다. Newnode가 마지막 노드가 되야하므로 prevnode가 현재 Linkedlist의 마지막 node인데 이 link를 newnode로 연결하고 newnode->link를 NULL로 설정한다. (top->…->prev->new->null)

남은 경우의 수는 중간에 newnode를 삽입하는 경우인데 newnode의 rankscore 가 currentnode의 rankscore가 보다 크거나 같을 때이다. 이때는 현재의 newnode의 link 를 currentnode로 설정해주고 하나 이전 node 인 prevnode의 link 를 newnode로 설정하면 될것이다. (top->…->prev->new->current)

이후 ranknumber 를 1 증가시켜주고 writeRankFile()을 호출하여 현재 추가된 newnode를 rank.txt에 기입 할 수 있도록 해준다.

1-2.

본 실험은 예비보고서에서 생각했던 array와 linkedlist 중 linkedlist를 선택했다. 그 이유는 우선 array는 linkedlist와 달리 모든 원소들이 continuous 하게 붙어 있다. 때문에 정보를 추가 하거나 삭제할 때 linkedlist는 해당 위치에 노드를 추가하거나 삭제만 하면 되지만 array는 그 이후 원소들을 모두 위치를 이동시켜야 한다. 만약 k ranking에 새로운 rank를 추가 한다면 linkedlist는 기존의 k node를 현재의 newnode에 link하고 newnode를 k-1 node에 새로 link하면 끝이 난다. Linkedlist 자체의 공간복잡도는 O(ranknumber) (ranknumber는 linked list 길이이다.) 이지만 rank 추가에 대한 공간복잡도는 newnode를 제외한 그 이외의 메모리 할당이 없기에 O(1)이고 최악의 경우 마지막 노드까지 순차적으로 찾는다면 노드 추가에 대한 시간복잡도는 O(ranknumber)이다. 하지만 array는 새로 array 배열 크기를 재할당하고 k 부터(index 0부터 시작) ranknumber-1 까지 모든 원소들이 다 한 칸 씩 뒤로 가야 하는 연산이 필요하다. 재할당에 대한 공간복잡도는 O(ranknumber), 원소 추가에 대한 시간 복잡도 역시 O(ranknumber)가 된다. K 번째 ranking을 삭제한다고 했을 때 역시 linkedlist는 rank 삭제에 대해 탐색에 필요한 변수들을 제외한 추가적인 메모리 할당은 없어 O(1) 이고 시간복잡도는 top부터 탐색을 해서 최악의 경우 마지막 node를 삭제할 때인 O(ranknumber)이다. 하지만 array는 배열 크기가 1 줄어들기 때문에 새로 메모리 재할당을 해줘야 해 공간복잡도는 O(ranknumber)이고 원소 삭제 역시 a[k-1]로 바로 찾을 수는 있겠지만 continuous 하게 다시 이어 붙어야해서 k부터(index 0부터 시작) ranknumber-1의 원소들이 앞으로 한 칸 씩 움직어야 한다. 시간복잡도는 O(ranknumber)이다.

즉 array는 삭제, 추가 시 앞 뒤로 대부분의 원소들을 움직이는 불필요한 연산들을 추가해야하고 계속해서 메모리를 재할당 해야 하는 비효율성이 생긴다.

Array는 장점은 검색이다. 원하는 ranking이 k 라면 바로 Array[k-1] 하면 찾을 수 있기 때문이다. 하지만 우리의 실험 및 과제에서는 검색 기능을 사용할 때 ranking 을 가지고 검색하지 않고 이름을 통해 같은지 다른지 비교하여 출력을 보였다. 때문에 Array도 결국 Array[0] 부터 [ranknumber-1]까지 순차적으로 모든 원소들을 탐색해야 해서 Array의 장점이 없다.

2-1.

본 실험자는 linkedlist에 대해 알고는 있었지만 자세하게 구현하고 코드를 짜보는 과정을 해보지는 않았다. 때문에 처음 실습을 할 때 segmentation fault, core dump 와 같은 메모리에 관련한 오류들이 많이 발생하였다. 이전 코딩은 포인터로 메모리를 많이 건드리지 않아 알지 못했던 오류였다. 또한 writeRankFile()에서 다 기입 후 모든 linkedlist를 free 시키는 것도 2중 free를 시키지 않았다고 생각했는데 linkedlist 특성 상 rank()에서 해당 Node를 delete할 때 그 전에 node를 NULL로 바꾸지 않으면 2중 free 오류가 나오는 것을 확인할 수 있었다. PL(프로그래밍 언어)에서 포인터로 메모리를 건드린다는 것은 imperative programming, high language 측면에서 좋지 않다고 배웠다. 그럼에도 link를 사용하는 linkedlist의 장점은 array에 비해 시간 복잡도, 공간 복잡도를 고려해보니 확실히 알 수 있었다. 메모리 할당만 잘 해주고 코드만 정확하다면 top node 부터 시작하는 linkedlist는 link만으로 모든 node를 탐색할 수 있었고 추가 및 삭제 역시 간단히 해당 node와 그 이전, 다음 node만 건드리면 해결되었다. 앞으로 코드를 구현할 때 단순히 코드만 돌아가는지에 집중하지 않고 시간, 공간 복잡도 역시 매우 중요한 고려 사항임을 인지하였다.

또한 ncurses library 역시 처음 접해보았는데 linux 상에서 keypad, scanw, getch 와 같은 함수로 어떻게 입력 받고 또 printw, clear 와 같은 함수들로 어떻게 출력하는지 테트리스 과제를 통해 실습을 할 수 있었다. (하지만 ncurses 는 아직 익숙하지 않은 것 같다. 강의 자료 pdf를 참고하면서 했다.)

3-1.

과제1. 이름 입력 후 해당 이름이 rank.txt에 있으면 모든 정보들을 출력하는 함수 코드이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음 사진은 현재 rank.txt의 정보이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

NHN을 검색해보았다. rank.txt에 있는 NHN과 관련한 모든 정보가 출력된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

해당 rank.txt에 없는 이름을 입력해보았다. Search failure 메세지가 정상적으로 출력되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

해당 함수는 처음 rank 실행 시 2를 누를 때 실행된다. 우선 printw를 통해 input the name을 출력하여 scanw로 str에 이름을 입력 받는다. 이후 과제의 요구사항대로 출력양식을 그대로 출력하고 currentnode를 top으로 두고 탐색을 시작한다. 이때 for문은 1부터 ranknumber까지 최대로 돌면 될 것이다. 반복문을 돌 때마다 currentnode=currentnode->link로 순차적으로 다음node로 이동시킨다. strcmp(currentnode->rankname,str) string compare(비교)를 하여 일치한다면 currentname->rankname, currentnode->rankscore를 출력한다. 이때 일치하면 check flag를 1 씩 증가 시켜 마지막에 check가 0이라면 일치하는 이름이 없기에 search failure 메세지를 출력한다. 이 과제에는 없지만 check를 통해 총 몇 개의 rank 정보를 출력했는지 역시 출력 가능할 것이다.

여기서는 for문이 ranknumber 만큼 돌면서 탐색을 하기 때문에 시간복잡도는 O(ranknumber)가 될 것이다. 또한 검색에 필요한 공간 복잡도는 반복문에 필요한 변수나 flag check 변수, 탐색에 사용되는 currentnode 등을 제외하면 추가적인 메모리 할당이 없어 O(1)이 된다.

(global 변수인 linkedlist 자체의 공간복잡도는 O(ranknumber) 이다.)

3-2.

rank 숫자를 입력받은 후 그 숫자에 해당하는 rank를 삭제하는 함수이다.

삭제알고리즘이 설명은 코드 사진상에 주석으로 표시를 하여 추가적인 설명은 생략하고 그림으로 표현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1번 즉 1위 rank를 삭제해 보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

rank.txt와 비교하여 ‘sangkeuny’가 정상적으로 삭제되었음을 확인할 수 있었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ranknumber보다 초과하는 숫자를 입력 시 search failure 메세지가 정상 출력되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ch==’3’ 이면 start

Num==i (loop 변수 i와 Num이 같을때 삭제)

true

false

false

true

false

true

end

Num==1(top 삭제)

prev=currentnode

currentnode=currentonde->link

Prev와 currentnode를 한 node씩 이동시킴

Printw(“search failure”) 메시지 출력 후 종료

Prev->link=currentnode->link(current node 삭제)

currentnode=NULL

Free(currentnode) (node 삭제)

Ranknumber=rankumber-1(ranknumber 1 줄임)

writeRankFile() 다시 rank.txt 작성

printw(“deleted”) 정상 삭제 알림 후 종료

top=currentnode->link(top 이동)

currentnode=NULL

Free(currentnode) (node 삭제)

Ranknumber=rankumber-1(ranknumber 1 줄임)

writeRankFile() 다시 rank.txt 작성

printw(“deleted”) 정상 삭제 알림 후 종료

Currentnode=top

Prev=NULL

“input the rank: “ 출력, num으로 입력 받기

num<1||num>ranknumber

Linked list (3번째 노드 삭제 시 예시)

Top node

Free 시킴

curruent

prev

Fourth node

Third node

Second node

Final node

Top 노드 삭제 시

Free 시킴

curruent

Top node

Top 재설정

Second가 다음부터 top

Fourth node

Third node

Second node

Final node