|  |
| --- |
|  |
| **컴퓨터공학실험1 (1반) 9주차 10주차 결과보고서 (테트리스(2))** |
|  |
|  |
|  |
| 컴퓨터공학과 20161620 이수연 |

**컴퓨터공학실험1 // 테트리스(2) 결과보고서**

전공: 컴퓨터공학과 학년: 2학년 학번: 20161620 이름: 이수연

**1. 실험시간에 작성한 랭킹 시스템의 자료구조와 랭킹 시스템의 각 기능에 대한 알고리즘을 요약하여 기술하시오. 본인이 선택한 랭킹 시스템을 구현하기 위한 자료구조가 왜 효율적인지 시간 및 공간 복잡도를 통해 보이고, 설명하시오.**

<createRankList> (rank.txt파일로부터 랭킹정보를 읽어오는 함수.)

1. FILE 포인터 변수 fp로 rank.txt 파일을 읽기 전용으로 연다.

2. fscanf를 통해서 파일의 첫줄 숫자를 읽어온다.

3. 새로운 node 를 할당하고 rank.txt파일로부터 이름 정보와 점수 정보를 넣는다.

4. node를 linkedlist의 맨 마지막 자리에 삽입한다.

5. 3~4번 과정을 아까 입력 받은 수 만큼 반복한다.

7. fclose를 통해 파일을 닫는다.

<rank>

세가지 기능을 구현한다. 이때 echo(); 와 noecho();, getch();를 사용하여 키보드 입력을 받을 수 있게끔 한다.

1) 랭킹정보를 출력해 주는 기능.

1. 사용자로부터 X 와 Y값을 입력받는다.

2. 만약 X와 Y가 둘 다 엔터일 경우에는 모든 랭킹을 , X가 엔터고 Y는 임의의 상수이면 1위부터 Y순위 까지, X가 임의의 상수이고 Y가 엔터일 경우에는 X순위부터 마지막 순위까지 랭킹을 출력한다.

3. 만약 X 값이 Y값보다 클 경우에는 에러메시지를 출력해 준다.

4. 임시변수에 linkedlist의 첫 번째 node를 가리키는 head변수 값을 넣는다.

5. 출력해야 하는 해당 랭킹 만큼 반복문으로 temp = temp->link;를 통해 노드를 이동하면서 temp의 정보를 출력한다.

2) 이름을 통해 랭킹정보를 가져와주는 기능.

1. 사용자로부터 찾을 이름을 입력 받는다.

2. 만약 입력 받은 이름이 랭킹정보에 등록되어 있지 않으면 에러메시지를 출력해준다.

3. 이름이 랭킹정보에 있을 경우에는 임시변수에 head값을 넣고 temp->name과 입력 받은 값을 strcmp 함수를 이용해 비교한다.

4. 만약 같을 경우에는 값을 출력하고. 그렇지 않으면 temp = temp->link를 이용해 다음 node로 넘어간다.

3) 입력 받은 랭킹을 지우는 기능.

1. 사용자로부터 지우고 싶은 랭킹순위를 입력 받는다.

2. 입력 받은 랭킹 순위가 현재 랭킹순위 밖이면 에러메시지를 출력한다.

3. 입력 받은 랭킹 순위만큼 반복문을 통해 해당 node를 찾아간다.

4. 해당 node의 이전 node와 다음 node를 연결한다.

5. 해당 node를 free를 통해 동정할당해제 시켜준다.

(연결리스트를 delete하는 코드를 이용한다)

6. writeRankFile 함수를 호출하여 rank.txt에 정보를 입력한다.

<writeRankFile> (현재 랭킹순위를 rank.txt파일에 입력해주는 함수.)

1. FILE포인터 변수 fp를 생성하여 쓰기전용으로 파일을 연다.

2. 임시변수에 head의 값을 넣고 정보를 rank.txt에 기록한다.

3. temp != NULL일 때까지 반복하면 한번 반복할 때마다 temp = temp->link를 통해 다음 node로 넘어간다.

<newRank> (새로운 랭킹정보를 추가하는 함수)

1. 사용자로부터 이름을 입력 받는다.

2. 새로운 node를 할당하여 입력 받은 이름과 score정보를 넣는다.

3. 만약 head 가 NULL이면 현재 랭킹정보가 없으므로 head에 node를 연결한다.

4. 그렇지 않으면 node의 score가 랭킹에 있는 score보다 커지는 순간까지 node를 이동시킨다. (점수를 내림차순으로 정렬하기 위함)

5. 찾은 node의 바로 이전 node를 하나 선언하여 랭킹정보에 연결한다.

6. rank\_num에 1을 더해준다. (rank\_num = 랭킹의 개수)

7. writeRankFile 함수를 호출하여 rank.txt에 정보를 입력한다.

8. fclose를 통해 파일을 닫는다.

**<연결리스트가 다른 자료구조보다 효율적인 이유>**

ranking을 구현하는데 있어서 linkedlist는 보통의 array, tree 등의 자료구제와 비교했을 때 공간복잡도는 비슷하거나 같다고 할 수 있다. 하지만 array를 예로 들면 array는 새로운 정보를 추가 삭제 하는데 있어서 총 n개의 배열이 있으면 n번 반복을 통해 추가 삭제를 해야 하지만, linkedlist는 O(1)만큼의 시간복잡도가 필요하다. 또한 array의 경우 sorting을 하면 이중포문이 사용되어 복잡도는 O(n^2)이 되지만 linkedlist는 rnum의 개수 즉 n만큼만 반복하면 되므로 O(n)이 된다. 따라서 시간복잡도에서는 현저하게 linkedlist가 효율적임을 알 수 있다.

**2. 본 실험 및 숙제를 통해 습득한 내용을 기술하시오.**

ranking에 관한 실험과 숙제를 하면서 여러 가지 자료구조들의 장단점을 파악할 수 있었다. 특히 이번 실습에서 사용한 linkedlist는 다른 자료구조들 보다 새로운 node를 연결하고 삭제하는데 있어서 시간복잡도가 훨씬 작다는 장점을 알 수 있었다. 자료구조 외에도 파일 입출력을 통한 자료의 저장을 연습하였고, node의 삽입 삭제 및 대소 비교를 통한 정렬을 습득하게 되었다. 또한 ncurses 라이브러리의 echo();, noecho();, clear();, getch(); 등의 함수의 역할을 배울 수 있었다.

**(끝)**