9주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20192135 이름: 윤영인

**1. 실습**

**1.1 구조체 선언**

이중연결리스트를 구현하기 위해 연결리스트를 구성하는 노드 구조체를 선언하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이름과 점수가 저장되는 변수인 score, name 배열과 이전 노드를 가리킬 포인터인 prev, 다음 노드를 가리킬 포인터인 next를 정의하였다.

연결리스트의 구조체를 선언하였다.

텍스트, 폰트, 스크린샷, 시계이(가) 표시된 사진

자동 생성된 설명

연결리스트의 헤드를 나타낼 Node 구조체인 head를 정의하였다.

**1.2 createRankList**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

input파일인 rank.txt에서 랭킹 정보를 읽어 들이기 위해 선언한 함수이다. 파일을 열어서 createLinkedList 함수로 연결리스트를 생성하여 rankList 변수에 저장하였다. createLinkedList 함수는 다음과 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

malloc으로 LinkedList형 공간을 할당하고 head에 NULL을 넣어 생성한 연결리스트 포인터를 리턴하였다.

연결리스트를 생성하여 파일의 첫 줄에 써 있는 랭킹의 수를 rankCount 라는 전역변수에 저장하고, rankCount만큼 파일을 읽으면서 Node를 생성하여 이름과 점수 정보를 저장한다. Node를 생성하였으면 연결리스트에 삽입하는데, 만약 리스트가 비어 있으면 해당 노드를 넣고 연결 리스트의 head를 삽입한 노드가 가리키도록 한다. 만약 비어있지 않다면 마지막 노드를 찾아서 해당 노드의 다음 노드로 삽입한다.

함수의 시간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이다. 공간 복잡도는 O(연결리스트의 노드의 개수) = O(n)이다.

**1.3 rank()**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이번 주 실습에서 rank 함수의 1번 메뉴를 구현해야 했다. 따라서 x와 y를 사용자로부터 입력 받아 랭킹의 범위를 벗어날 경우 x는 1로 y는 rankCount로 초기화하였다. 만약 x가 y보다 크거나 연결리스트가 비어 있을 경우 에러 메시지를 출력하도록 하였다. 비어 있지 않다면, 연결리스트의 head부터 x 순위를 찾는다. x 순위를 찾으면 노드를 이동하는 것을 멈추고, 해당 노드부터 y 순위 노드까지 반복문을 돌면서 노드의 정보를 출력한다. 해당 함수의 시간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이고, 공간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이다.

**1.4 writeRankFile()**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

랭킹의 정보가 기록된 연결리스트가 갱신되면 rank.txt를 갱신하는 함수이다. 따라서 연결리스트의 head부터 마지막 노드부터 이동하면서 rank.txt 파일에 다시 적어준다. 시간 복잡도는 O(연결리스트의 노드의 개수)=O(n), 공간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이다.

**1.5 newRank()**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

게임이 종료되면 연결리스트의 score과 사용자의 이름을 업데이트하는 함수이다. 사용자로부터 이름을 입력 받고 사용자의 점수와 이름을 저장할 노드를 동적 할당한다. 연결리스트가 비어 있으면 해당 노드를 연결리스트의 헤드로 삽입하고, 그렇지 않으면 연결리스트를 탐색하면서 삽입하려는 노드의 점수보다 작은 점수를 가진 노드를 찾고 탐색을 멈춘다. 멈춘 지점의 노드의 다음에 노드를 삽입하고, 랭킹의 개수를 하나 증가한다. 그리고 writeRankFile 함수를 실행하여 rank.txt 파일을 새로 갱신한다. 이 함수의 시간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이고, 공간 복잡도는 O(연결리스트의 노드의 개수)=O(n)이다.

**1.6 연결리스트가 효율적인 이유**

위에서 설명한 함수들의 시간 복잡도와 공간 복잡도는 모두 O(연결리스트의 노드의 개수)=O(n)이다. 즉, 연결리스트의 노드의 개수만큼 시간과 공간을 소요한다는 뜻이다. 또한 이전 노드와 다음 노드를 서로 연결하고 있기 때문에 중간 노드를 삭제하거나 연결리스트 중간에 노드를 삽입할 때도 노드의 포인터들을 수정하면 되므로 굉장히 효율적이다. 노드를 삽입하거나 삭제할 때도 연결리스트의 노드를 탐색할 때의 시간과 공간이 소요될 뿐, 삽입과 삭제 자체의 연산의 시간 복잡도는 O(1)이므로 효율적이라고 판단하여 연결리스트를 사용하였다.

**2. 과제**

**2.1 모드 2**

우선 rank.txt는 다음과 같다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

NHN 사용자를 입력하였을 때 해당 사용자의 랭킹 정보가 출력되는지 확인하였다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

아래의 사진은 2번 메뉴를 실행하는 함수이다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이름을 입력 받아, 연결리스트의 head 노드부터 돌면서 노드 안의 저장된 이름과 사용자로부터 입력 받은 이름을 strcmp로 비교한다. 만약 같은 노드를 찾았다면 노드의 정보를 출력하고 check 변수를 1로 설정한다. 연결리스트 탐색이 끝난 후 check변수가 0이면 같은 노드를 찾지 못했단 뜻이므로 에러메시지를 출력한다. 따라서 시간 및 공간 복잡도는 O(연결리스트의 개수)=O(n)이다.

**2.1 모드 3**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

연결리스트를 돌면서 count를 증가시키며 사용자로부터 입력 받은 삭제 순위 노드를 찾으면 탐색을 종료하고 삭제 노드를 delNode 변수에 저장한다. 만약 삭제하려는 노드가 연결리스트의 head이면 head를 현재 노드의 다음으로 이동시키고, 이동시킨 head의 이전 포인터에 NULL로 저장한다. 그림을 그리면 다음과 같다.

원, 도표, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

만약 삭제하려는 노드가 중간 노드라면 삭제 노드의 이전 노드의 다음 노드를 삭제 노드의 다음 노드를 가리키게 하고, 삭제 노드의 다음 노드의 이전 노드를 삭제 노드의 이전 노드를 가리키도록 한다. 그림으로 표현하면 다음과 같다.

원, 도표, 폰트, 그래픽이(가) 표시된 사진

자동 생성된 설명

만약 삭제하려는 노드가 마지막 노드라면 삭제하려는 이전 노드의 다음 노드에 NULL을 저장하고 노드를 삭제한다. 그림을 그리면 다음과 같다.

원, 텍스트, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

**3.**

이번주 실습과 과제에서 연결리스트로 랭킹 정보를 구현하는 것을 진행하였다. txt 파일에 담긴 랭킹 정보를 노드에 담아 연결리스트를 생성하고, 사용자가 원하는 노드를 찾거나 삭제하는 것을 구현하면서 연결리스트의 시간 및 공간 복잡도를 분석하였다. 노드의 개수 만큼 시간 및 공간 복잡도가 필요하다는 것을 알 수 있었고 삭제 및 삽입 연산의 시간 복잡도가 O(1)이라는 것을 알게 되면서 효율적인 자료구조라는 것을 파악할 수 있었다.