데이터 구조

프로젝트 01

Contents

❖ 단일 연결 리스트(Singly Linked list)를 이용한 학생 성적 관리 시스템

- 제출 파일 : student.h, student.c (main함수는 교수가 따로 사용할 예정)
- 제출 기한 : 11/17~ 23:59, 11/20 수업시간 전까지 지각제출
- 조건
  - 제공되는 파일을 수정하여 다음 시스템을 만족하여라.(main 함수 제외)
  - 다음 함수들을 만족하는 학생 성적 관리 시스템을 완성하여라.
    - **학생과 점수를 입력받거나 랜덤으로 생성하여 저장하는데, 점수는 동점이 있을 수 있으나, 동명이인은 없다고 가정한다.**
    - **학생 이름은 영어이름으로 입력하고 모두 소문자로 입력되며, 9글자(char [10])이하로 가정한다.**
    - **성적의 경우 0점~100점 사이로 입력된다.**
    - **점수의 크기대로 나열한다.(이 때 동점자의 경우 사전적으로 앞에 있는 사람이 먼저 저장된다.)**
    - **함수 하나씩 구현이 안될 때 마다 1점씩 감점(10점 만점인 상태에서)**

Contents

❖ 단일 연결 리스트(Singly Linked list)를 이용한 학생 성적 관리 시스템

- 조건
  - 다음 함수들을 만족하는 학생 성적 관리 시스템을 완성하여라.
    - **입력에 한해서는 예외처리를 하지 않아도 된다.(이름의 길이가 10글자 이상인 경우 X, 동명이인 입력 X, 점수가 0~100사이로 입력되지 않는 경우 X)**
    - **Listhead 구조체에 type이라는 int형 변수를 하나 만들고 이 값이 0 인경우는 현재 저장된 데이터가 오름차순으로 1인 경우는 내림차순으로 저장된다. 저장되어 있는 형태가 내림차순인 상태일 때 type 이 0으로 바뀌면 오름차순으로 바꿔 저장한다. (100-90-80 : 내림, 80-90-100 : 오름)**
    - ✓ 이름의 경우도 마찬가지 조건을 따른다. 100-90(A)-90(B)-90(c) // 80-90(c)-90(B)-90(A)-100
    - **함수를 추가하는 것은 자유이며, 함수의 파라미터나 리턴 형태를 변경은 불가하다.**
    - **내부가 채워져 있는 함수를 바꾸는 것은 자유이다.**
    - **Main 함수는 학생들이 사용하기 편하도록 만들어 놓은 것이고, main 함수는 자유롭게 변경하여 사용해도 상관 없다.**
    - **전역 변수 사용 금지!**

Student.h

❖ **노드 형태(제공)**

```
typedef struct listNode {
        char name[10];                  // 이름
        int score;                      // 성적(0~100)
        struct listNode* link;
}listNode;
```

❖ **헤드 형태(제공)**

```
typedef struct {
        struct listNode* head;
        int type;                       //1이면 내림 차순, 0이면 올림 차순
}listHead;
```

❖ **필요 함수들(일부 제공)**

```
listHead* CreateNode();                                     //제공(헤드 생성 초기값은 head에 NULL, type = 1)
void ClearNode(listHead* h);                                //제공(노드 전체삭제)
listNode* AddRandomNode(listHead* h);                       //수정필요
listNode* AddDirectNode(listHead* h, int s, char* n);       //수정필요
listNode* DeleteSelectNode(listHead* h, char* n);           //작성필요
listNode* DeleteLastNode(listHead* h);                      //제공(마지막 노드 삭제 및 리턴)
listNode* ChangeNode(listHead* h, int s, char* n);          //작성필요
listNode* FindbyName(listHead* h, char* n);                 //작성필요
listNode* FindbyScore(listHead* h, int s);                  //작성필요
listNode* FindMiddleNode(listHead* h);                      //작성필요
listNode* FindLastNode(listHead* h);                        //수정필요(항상 꼴등 출력)
double CalAverage(listHead* h);                             //작성필요
void PrintAllNode(listHead* h);                             //제공(전체 출력)
void ReverseList(listHead* h);                              //작성필요
void PrintRangedNode(listHead* h, int min, int max);        //작성필요
```

Student.c

listHead* CreateNode();

>> list 생성 함수 head를 동적할당 하고, 첫 노드를 가르키는 포인터 변수에 NULL을 넣고 리스트의 type을 1로 초기화 한다. 이 때 1의 의미는 데이터들이 내림차순으로 정렬된다는 뜻이고 0으로 바뀌는 경우 데이터들이 오름차순으로 정렬된다는 뜻이다.

>> 프로그램 실행 시 1회 불린다.

void ClearNode(listHead* h);

>>노드가 전체 삭제된다. 프로그램 종료 시 1회 불린다.

listNode* AddRandomNode(listHead* h);

>> h 리스트에 랜덤으로 노드를 생성하여 추가하고, 추가된 노드를 리턴한다.

>>랜덤으로 생성되어 첫 위치에 저장되도록 구현되어 있다.

>> 실제 구현의 경우 랜덤으로 생성된 데이터의 정보 형태에 따라서 위치를 찾아서 저장한다. 이 때 type이 1인지 0인지에 따라서 자기 위치를 잘 찾아야한다.

listNode* AddDirectNode(listHead* h, int s, char* n);

>> 입력된 점수 s와 이름 n을 노드에 저장하고 리스트에 추가하고 그 노드를 리턴한다.

>> 현재는 첫 노드에 저장되도록 구현되어 있다.

>> 실제 구현의 경우 데이터의 정보 형태에 따라서 위치를 찾아서 저장한다. 이 때 type이 1인지 0인지에 따라서 자기 위치를 잘 찾아야한다.

listNode* DeleteSelectNode(listHead* h, char* n);

>> 입력된 이름을 기반으로 리스트내에서 노드를 찾아 삭제하고 그 노드를 리턴한다.

>> 만약 삭제에 실패하는 경우 NULL을 리턴한다.

listNode* DeleteLastNode(listHead* h);

>> 마지막 노드를 삭제하고 그 노드를 리턴한다.

Student.c

listNode* ChangeNode(listHead* h, int s, char* n);

>>입력된 n 이름을 기반으로 노드를 찾아 그 점수를 입력된 s값으로 수정하고 수정한 노드를 리턴한다.

>> 점수가 바뀌게 될 경우 그 위치에 맞게 노드의 위치를 재조정한다.

listNode* FindbyName(listHead* h, char* n);

>> 입력된 n 이름을 기반으로 노드를 찾아 그 노드를 리턴한다.

listNode* FindbyScore(listHead* h, int s);

>> 입력된 s점수를 기반으로 노드를 찾아 그 노드를 리턴한다.

>> 동점자가 있기 때문에, 가장 마지막에 저장되어 있는 노드를 리턴한다.

>> type1인 경우는 사전적으로 제일 뒤에, type 0 인 경우는 사전적으로 제일 앞에 오는 사람이 리턴된다.

listNode* FindMiddleNode(listHead* h);

>> 현재 저장된 노드중 중간 위치에 있는 노드를 찾아 리턴한다.

>> 저장된 개수가 홀수인 경우는 정확히 중간을 리턴하고, 짝수인 경우는 가운데 둘 중 하나를 리턴하면 된다. Ex) 10명인 경우 5/6등 중 아무나 리턴이 가능하다.

listNode* FindLastNode(listHead* h);

>> 가장 마지막에 저장된 노드를 리턴한다.

>> type1 의 경우 꼴등을 type 0의 경우는 일등이 리턴된다.

>> 현재는 type 1 만 구현되어 있다.

Student.c

double CalAverage(listHead* h);
>> 전체 학생의 평균을 계산해서 그 값을 리턴한다.

void PrintAllNode(listHead* h);
>> 전체 학생의 이름 및 점수를 출력한다.

void ReverseList(listHead* h);
>> type 의 값을 바꾼다. 0 <->1
>> 그에 맞게 리스트의 형태도 정반대로 바꿔 정렬한다.

void PrintRangedNode(listHead* h, int min, int max);
>> 입력받은 min~max 사이의 점수안에 들어간 학생들을 모두 출력한다.