문제해결기법 11주차 과제

202033762 장민호

-Code

```
#include <stdio.h>
#include <stdlib.h> // malloc, rand, atoi ..동적메모리, 난수, 문자열 변환
#include <string.h> //strcpy.. 문자열함수. memcpy.. 메모리블럭 함수
#define TRUE 1
#define FALSE 0
#define Number_Of_Conditions 6
#define Condition_Len 20
/* List 의 데이터 */
typedef struct Data
{
     int back_number;
     char name[30];
     int age;
     int A_matches;
     int goals;
} Data;
// 노드 타입 정의
typedef struct _node
{
     Data data;           // 노드 데이터
     struct _node *pNext; // 다음 노드 포인터
} Node;
// Linked List 타입 정의
typedef struct _linkedList
{
     Node *pHead;
     Node *pTail;
     Node *pCurrent; // iteration 용
     int numData;
} LinkedList;
typedef LinkedList List;
/* List 의 동작 */
// 리스트 초기화
void list_init(List *pList);
// 데이터 추가
int list_add(List *pList, Data data);
// 데이터 조회, iteration 초기화
void list_init_iter(List *pList);
// 데이터 조회, iteration 다음 데이터 추출
Data list_next(List *pList);
// 데이터 조회, iteration 다음 데이터가 있나?
int list_hasNext(List *pList);
// 리스트 초기화
void list_init(List *pList)
{
     // head 용 dummy node 방식
     pList->pHead = (Node *)malloc(sizeof(Node));
     pList->pHead->pNext = NULL;
     pList->pTail = pList->pHead;
     pList->numData = 0;
}
// 데이터 조회, iteration 다음 데이터 추출
Data list_next(List *pList)
{
     pList->pCurrent = pList->pCurrent->pNext; // 우선 current 한발 앞으로 이동
     Data result = pList->pCurrent->data;      // 데이터 추출
```

```
    return result;
}
// 데이터 추가
int list_add(List *pList, Data data)
{
    // 새로운 node 생성
    Node *pNewNode = (Node *)malloc(sizeof(Node));
    memset(pNewNode, 0, sizeof(Node));
    pNewNode->data = data;
    // tail이 가리키던 node의 next를 새로운 node에 연결
    pList->pTail->pNext = pNewNode;
    // tail 을 새로운 node 로 이동
    pList->pTail = pNewNode;
    // 데이터 개수 증가
    (pList->numData)++;
    return TRUE;
}
// 데이터 조회, iteration 초기화
void list_init_iter(List *pList)
{
    pList->pCurrent = pList->pHead;
}
// 데이터 조회, iteration 다음 데이터가 있나?
int list_hasNext(List *pList)
{
    if (pList->pCurrent->pNext == NULL) // '다음 노드' 존재 여부 체크.. 없으면 false
        return FALSE;
    return TRUE;
}
void list_sort(List *pList)
{
    // 링크드리스트를 백넘버순으로 정렬한다. Selection Sort 사용
    int len = pList->numData;
    for (int i = 0; i < len - 1; i++)
    {
        // pList가 헤드를 가리키도록
        // pCurrent = pHead
        list_init_iter(pList);
        // FirstNode는 계속 그 다음 노드가 되어야 하므로 pCurrent를 i번 만큼 뒤로 땡겨야 한다.
        for (int j = 0; j < i; j++)
        {
            if (list_hasNext(pList))
            {
                pList->pCurrent = pList->pCurrent->pNext;
            }
        }
        // 처음 노드와 선택될 노드를 생성
        Node *firstNode = (Node *)malloc(sizeof(Node));
        Node *selectNode = (Node *)malloc(sizeof(Node));
        Node *tempNode = (Node *)malloc(sizeof(Node));
        Node *tempNode2 = (Node *)malloc(sizeof(Node));
        Node *a = (Node *)malloc(sizeof(Node));
        Node *b = (Node *)malloc(sizeof(Node));
        Node *c = (Node *)malloc(sizeof(Node));
        Node *d = (Node *)malloc(sizeof(Node));
        firstNode = pList->pCurrent;
        selectNode = firstNode;
        // 그 다음 노드부터 차례로 찾기
        while (list_hasNext(pList))
        {
            if (pList->pCurrent->pNext->data.back_number < selectNode->pNext-
>data.back_number)
            {
                // 만약 더 작은 데이터를 찾았을 경우, 그 이전 노드를 select한다.
                selectNode = pList->pCurrent;
```

```
                }
                // 현재 가리키는 노드를 그 다음 노드로 변경해준다.
                pList->pCurrent = pList->pCurrent->pNext;
            }
            memcpy(tempNode, firstNode, sizeof(Node));
            memcpy(tempNode2, selectNode, sizeof(Node));
            // 두 노드의 위치를 변경한다.
            if (firstNode == selectNode)
            {
                continue;
            }
            else if (firstNode->pNext == selectNode)
            {
                a = tempNode2->pNext;
                b = tempNode->pNext;
                c = tempNode2->pNext->pNext;
                firstNode->pNext = a;
                firstNode->pNext->pNext = b;
                firstNode->pNext->pNext->pNext = c;
            }
            else
            { /*
             firstNode의 pNext와, selectNode의 pNext를 서로 변경해야 한다.
             서순 주의 꼬이면 망함
        */
                a = tempNode2->pNext;
                b = tempNode->pNext->pNext;
                c = tempNode->pNext;
                d = tempNode2->pNext->pNext;
                firstNode->pNext = a;
                firstNode->pNext->pNext = b;
                selectNode->pNext = c;
                selectNode->pNext->pNext = d;
            }
        }
}
void printList(List *pList)
{
    list_init_iter(pList);
    while (list_hasNext(pList))
    {
        Data data = list_next(pList);
        printf("%d/%s/%d/%d/%d\n", data.back_number, data.name, data.age, data.A_matches,
data.goals);
    }
}
Data list_search(List *pList, char str[Condition_Len])
{
    // 서치 조건 분석
    char newstr[Condition_Len];
    strcpy(newstr, str);
    char *temp = strtok(newstr, " ");
    char condition[2][Condition_Len];
    int idx = 0;
    while (temp != NULL)
    {
        strcpy(condition[idx], temp);
        temp = strtok(NULL, " ");
        idx++;
    }
    // 분석한 조건대로 서치
    list_init_iter(pList);
    Data compareData;
    Data currentData;
```

```
compareData = list_next(pList);
if (strcmp(condition[1], "age") == 0)
{
     if (strcmp(condition[0], "maximum") == 0)
     {
          while (list_hasNext(pList))
          {
               currentData = list_next(pList);
               if (compareData.age < currentData.age)
               {
                    compareData = currentData;
               }
          }
     }
     else if (strcmp(condition[0], "minimum") == 0)
     {
          while (list_hasNext(pList))
          {
               currentData = list_next(pList);
               if (compareData.age > currentData.age)
               {
                    compareData = currentData;
               }
          }
     }
}
else if (strcmp(condition[1], "goals") == 0)
{
     if (strcmp(condition[0], "maximum") == 0)
     {
          while (list_hasNext(pList))
          {
               currentData = list_next(pList);
               if (compareData.goals < currentData.goals)
               {
                    compareData = currentData;
               }
          }
     }
     else if (strcmp(condition[0], "minimum") == 0)
     {
          while (list_hasNext(pList))
          {
               currentData = list_next(pList);
               if (compareData.goals > currentData.goals)
               {
                    compareData = currentData;
               }
          }
     }
}
else if (strcmp(condition[1], "A-matches") == 0)
{
     if (strcmp(condition[0], "maximum") == 0)
     {
          while (list_hasNext(pList))
          {
               currentData = list_next(pList);
               if (compareData.A_matches < currentData.A_matches)
               {
                    compareData = currentData;
               }
          }
     }
```

```
        else if (strcmp(condition[0], "minimum") == 0)
        {
            while (list_hasNext(pList))
            {
                currentData = list_next(pList);
                if (compareData.A_matches > currentData.A_matches)
                {
                    compareData = currentData;
                }
            }
        }
    }
    return compareData;
}
void printData(Data data, char condition[])
{
    printf("%s player's info: %d/%s/%d/%d/%d\n", condition, data.back_number, data.name,
data.age, data.A_matches, data.goals);
}
int main()
{
    // 리스트 생성
    List pList;
    list_init(&pList);
    /*
    ppt 97p condition 1
    Read data about players (back number, name, age, goals, and A-matches), and store them in
nodes of a linked list, sorted by the “back number” in ascending order.
    */
    FILE *openFile;
    openFile = fopen("./stored.dat", "r");
    if (openFile != NULL)
    {
        while (!feof(openFile))
        {
            Data data;
            fscanf(openFile, "%d/%[^/]/%d/%d/%d", &data.back_number, data.name, &data.age,
&data.A_matches, &data.goals);
            list_add(&pList, data);
        }
        // 데이터를 다 입력하면 sort 한다.
        list_sort(&pList);
        printList(&pList);
    }
    /*
    ppt 97p condition 2
    Next, read search conditions.
        minimum age
        maximum age
        minimum goals
        maximum goals
        minimum A-matches
        maximum A-matches
    Then, print the player data for the players who satisfy the conditions. Print the player
data in the following format
        - back number, name, age, goals, A-matches
    */
    Data returnData[Number_Of_Conditions];
    char condition[Number_Of_Conditions][Condition_Len] = {
        "minimum age", "maximum age", "minimum goals", "maximum goals", "minimum A-matches",
"maximum A-matches"};
    for (int i = 0; i < Number_Of_Conditions; i++)
    {
        returnData[i] = list_search(&pList, condition[i]);
```

```
        printData(returnData[i], condition[i]);
    }
    return 0;
}
```

-Screenshot

```
mhj@mhj-IdeaPad:~/gitRepo/2022_second-semester/Problem solving techniques/week11$ ./"assignment(week
11)"
1/Jung-Sung-Ryong/25/22/0
7/Park-Ji-Sung/29/94/13
10/Park-Chu-Young/25/47/15
12/Lee-Young-Pyo/33/119/5
16/Ki-Sung-Yueng/21/28/4
17/Lee-Chung-Yong/22/27/4
22/Cha-Du-Ri/30/51/4
minimum age player's info: 16/Ki-Sung-Yueng/21/28/4
maximum age player's info: 12/Lee-Young-Pyo/33/119/5
minimum goals player's info: 1/Jung-Sung-Ryong/25/22/0
maximum goals player's info: 10/Park-Chu-Young/25/47/15
minimum A-matches player's info: 1/Jung-Sung-Ryong/25/22/0
maximum A-matches player's info: 12/Lee-Young-Pyo/33/119/5
```