자료구조 프로그래밍 과제 3

제출일: 2022-11-29
학번/이름: 2016121150 / 윤준영

문제 1. 일반적인 힙 정렬 함수를 사용하여 학생들의 정보를 정렬하여 보여주는 프로그램을 작성하시오. 학생의 정보는 파일에서 읽어서 구조체의 배열에 저장하며, 학과와 학번의 오름차순으로 정렬하되 학과를 우선으로 한다. 이름과 학과명은 빈칸 없는 한 단어이며, 학과명의 정렬 시에는 대소문자를 고려하지 않으며, 결과를 <입력파일명>.out에 저장한다. (편의상 파일 내 학생 정보의 개수는 최대 50개로 가정한다.)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 50
#define NAME_SIZE 12
#define ID_SIZE 12
#define MAJ_SIZE 12
#define LINE_SIZE 36
#define SENTINEL "ZZZZZZZZZZZ 999999999 ZZZZZZZZZZZ"

typedef struct {
    char name[NAME_SIZE];
    char id[ID_SIZE];
    char maj[MAJ_SIZE];
} student;

int up(int i) {
    // make upper letter
    if (i >=97 && i <= 122) i -=32;
    return i;
}
int cmp(student* a, student* b) {
    // return +: a > b
    // return 0: a = b
    // return -: a < b
    // continue until different letter or until end of string
    // if different letter return a-b(major)
    // if same end of string return a-b(id)
    int i = 0;
    while(up(a->maj[i]) == up(b->maj[i]) && up(a->maj[i]) != 0 && up(b->maj[i]) != 0) i++;
    if (up(a->maj[i]) == up(b->maj[i])) {
        i = 0;
        while (up(a->id[i]) == up(b->id[i]) && up(a->id[i]) != 0 && up(b->id[i]) != 0) i++;
        return up(a->id[i]) - up(b->id[i]);
    }
    else return up(a->maj[i]) - up(b->maj[i]);
    printf("%s", up(a->maj[i]));
}
```

1) student : student는 이름(name), 학번(id), 학과(maj)를 포함하는 구조체이다.
2) up : 문자를 정수로 읽어 만약 소문자이면 대문자를 정수로 반환하는 함수이다.
3) cmp : 두 student 구조체를 받아 비교하는 함수이다. 먼저 학과를 한 글자씩 받아 다른 부분이 나올

때까지 다음 글자를 받아 반복하며 두 문자 중 null 문자가 나오면 반복을 종료한다. 만약 두 문자 모두 null 문자일 학번으로 넘어가 학번을 같은 방법으로 비교한다. 두 문자 모두 null이 아닐 경우 두 문자의 ASCII 코드 차이를 반환한다. 만약 첫 student가 우선순위가 높을 경우 -, 같을 경우 0, 반대일 경우 +의 정수를 반환하게 된다.

```
49  void upheap(student* S[], int k) {
50      // sort along ancestors going up, k/=2
51      student* v = (student *)malloc(sizeof(student));
52      memcpy(v, S[k], sizeof(student));
53      while (cmp(S[k/2], v) <= 0) {
54          memcpy(S[k], S[k/2], sizeof(student));
55          k /= 2;
56      }
57      memcpy(S[k], v, sizeof(student));
58      free(v);
59  }
60  void downheap(student* S[], int n, int k) {
61      // sort along descendants going down, k*=2
62      student* v = (student *)malloc(sizeof(student));
63      int i;
64      memcpy(v, S[k], sizeof(student));
65      while (k <= n/2) {
66          i = 2*k;
67          if (i < n && cmp(S[i], S[i+1]) < 0) i++;
68          if (cmp(v, S[i]) >= 0) break;
69          memcpy(S[k], S[i], sizeof(student));
70          k = i;
71      }
72      memcpy(S[k], v, sizeof(student));
73      free(v);
74  }
75  void heap_sort(student* S[], int n) {
76      // heap sort
77      // upheap -> extract -> downheap
78      student* tmp = (student *)malloc(sizeof(student));
79      for (int i = 2; i < n; i++) upheap(S, i);
80      for (int i = n-1; i >= 2; i--) {
81          //extract
82          memcpy(tmp, S[1], sizeof(student));
83          memcpy(S[1], S[i], sizeof(student));
84          memcpy(S[i], tmp, sizeof(student));
85          downheap(S, i-1, 1);
86      }
87      free(tmp);
88  }
```

4) upheap : student 구조체 배열과 upheap 대상이 되는 요소의 index를 받는다. 위의 cmp함수를 이용하여 해당 요소와 해당 요소의 조상들을 내림차순으로 정렬한다.

5) downheap : student 구조체 배열과 downheap 대상이 되는 요소 개수와 뿌리 노드의 index를 받는다. 위의 cmp함수를 이용하여 뿌리 노드와 자손 노드 중 가장 큰 노드를 가장 마지막 노드로 이동하여 downheap을 수행한다.

5) heap_sort : student 구조체 배열과 요소의 개수를 받아 downheap, extract, downheap을 차례로 수행하여 힙 정렬을 하는 함수이다.

```c
int loadfile(char* fn, student* S[MAX]) {
    // input: file name, student struct array
    // output: number of student
    FILE* fp;
    student* s;
    char line[LINE_SIZE] = SENTINEL;
    char *lp;
    int i = 0;
    if ((fopen_s(&fp, fn, "r")) != NULL) {
        printf("\nWrong File!");
        return i;
    } // return 0 no file
    while(1) {
        // remove \n if exist
        if (line[strlen(line)-1] == 10) line[strlen(line)-1] = 0;
        s = (student *)malloc(sizeof(student));
        //printf("%s\n", &line);
        lp = strtok(line, " ");     // get name
        strcpy(s->name, lp);
        lp = strtok(NULL, " ");     // get id
        strcpy(s->id, lp);
        lp = strtok(NULL, " ");     // get major
        strcpy(s->maj, lp);
        S[i++] = s;
        if (feof(fp)) break;
        fgets(line, sizeof(line), fp);
    }
    fclose(fp);
    return i;
}

void data_save(char* fn, student* S[], int n) {
    FILE* fp;
    char fno[MAX];
    strcpy(fno, fn);
    strcat(fno, ".out");
    char line[LINE_SIZE];
    fopen_s(&fp, fno, "w");
    // put data 1 ~ n-1, 0 is sentinel value
    for (int i = 1; i < n; i++){
        strcpy(line, S[i]->name);
        strcat(line, " ");
        strcat(line, S[i]->id);
        strcat(line, " ");
        strcat(line, S[i]->maj);
        if (i < n-1) strcat(line, "\n");
        fputs(line, fp);
    }
    fclose(fp);
    printf("Save complete: %s\n", fno);
}
```

6) loadfile : 파일의 이름과 student 구조체 배열을 받아 파일을 열어 구조체 배열에 student 구조체를 저장하는 함수이다. 저장 후 불러온 student 구조체의 개수를 반환한다. 파일을 한 줄씩 line 문자열로 불러와 개행문자를 삭제한 뒤, strtok 함수를 이용하여 띄어쓰기로 구분한 뒤 차례로 구조체 s의 name, id, maj에 저장한다. 첫 line은 매크로로 지정되어 있는 SENTINEL로 지정하여 S[0]는 보초의 역할을 하게 된다.

7) data_save : 저장할 파일의 이름(.out 제외), 구조체의 배열 S, 구조체의 개수 n을 받아 S에 저장된 학생의 정보를 파일로 저장하는 함수이다. strcpy와 strcat을 이용하여 i번째 학생의 name, " ", id, " ", maj, 개행문자를 차례로 line에 합친 뒤 fputs함수를 이용하여 파일에 작성한다. 1부터 n-1까지 반복한다. 만약 i가 마지막 학생이라면 개행문자를 삽입하지 않는다.

```
142  int main(int argc, char* argv[]) {
143      student* S[MAX];
144      int n = loadfile(argv[1], S);
145      heap_sort(S, n);
146      // for (int i = 1; i < n; i++) {
147      //     printf("%s %s %s\n", S[i]->name, S[i]->id, S[i]->maj);
148      // }
149      data_save(argv[1], S, n);
150      return 0;
151  }
```

8) main : main 함수는 위와 같다. 먼저 최대길이 50의 구조체 배열을 만든 후, loadfile 함수를 함수를 실행할 때 받은 argv인 파일 이름으로 구조체 배열을 채워넣을 후 n에 배열의 길이를 저장한다. heap_sort 함수를 이용하여 정렬한 후, data_save함수를 이용하여 정렬된 구조체를 .out 파일로 저장하게 된다.

9) 실행 결과 : 함수의 실행 화면과 저장된 student.txt.out은 다음과 같다.

student.txt.out

```
1   jsPark 90302003 Archi
2   Park 99333939 Archi
3   Choi 92323033 Comm
4   Kim 88302032 CS
5   Choe 91323303 CS
6   Yi 99303092 CS
7   Lee 23223232 Elec
8   Jang 01303009 Math
9   kim 99303099 Math
10  Ahn 88302025 Physics
```

```
C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW3\hw3_2016121150_1.exe student.txt
Save complete: student.txt.out
```

학과의 ABC 순서로 정렬된 후 같은 학과일 경우 학번 순으로 잘 정렬된 것을 확인할 수 있다.

문제 2. 학생 이름과 학번, 전공으로 이루어진 학생 레코드를 이진검색나무에 저장하고 관리하는 프로그램을 작성하시오. 이진검색나무에 저장시에는 학번을 기준으로 저장하게 되며, save를 선택하여 결과를 파일에 저장하면 학번 순서대로 저장된다.

```
10  #include <stdio.h>
11  #include <stdlib.h>
12  #include <string.h>
13  #define MAX 50
14  #define NAME_SIZE 12
15  #define ID_SIZE 12
16  #define MAJ_SIZE 12
17  #define LINE_SIZE 36
18
19  typedef struct _stdt {
20      char name[NAME_SIZE];
21      int id;
22      char maj[MAJ_SIZE];
23      struct _stdt* left;
24      struct _stdt* right;
25  } stdt;
```

```
27  int digit(int i) {
28      int digit = 0;
29      while (i != 0) {
30          i = i/10;
31          ++digit;
32      }
33      return digit;
34  }
35  int up(int i) {
36      // upper letter
37      if (i >=97 && i <= 122) i -=32;
38      return i;
39  }
40  int idcmp(stdt* a, stdt* b) {
41      return a->id - b->id;
42  }
43  void input(char* s){
44      /* get string and delete new_line */
45      fgets(s, MAX, stdin);
46      if (s[strlen(s)-1] == '\n') s[strlen(s)-1] = '\0';
47  }
48  int str2int(void){
49      /* get integer using input function */
50      char s[ID_SIZE];
51      input(s);
52      return atoi(s);
53  }
```

1) stdt : stdt 구조체는 학생의 이름(name, 문자열), 학번(id, 정수), 전공(maj, 문자열)과 이진나무에서 사용할 다음 left, right 노드의 포인터를 담고 있다.

2) digit : 학번을 정수로 저장하기 때문에 0으로 시작하는 학번은 앞의 0이 생략되게 된다. 이때 학번의 자리수를 계산하여 앞에 0을 추가하기 위한 함수이다. 정수를 받아 10을 반복해서 나누어 계산한 후 자리수를 정수로 반환한다.

3) up : 문자를 정수로 읽어 만약 소문자이면 대문자를 정수로 반환하는 함수이다.

4) idcmp : 두 stdt 구조체를 받아 학번의 차이를 반환하는 함수이다. 첫 학번이 크면 +, 같으면 0, 작으면 –를 반환한다.

5) input, str2int : 과제 2번에서 사용한 함수와 같은 함수로, input 함수를 fgets를 이용하여 문자열을 받은 뒤, 마지막 개행문자를 null문자로 바꾼다. str2int는 input을 통해 입력받은 문자열을 숫자로 바꿔주는 함수이다. 정수를 입력받을 때 사용하는 함수이다.

```
54  void bintree_init(stdt** s) {
55      *s = (stdt *)malloc(sizeof(stdt));
56      (*s)->id = INT_MAX;
57      (*s)->left = NULL;
58      (*s)->right = NULL;
59  }
60
61  int data_insert(stdt* S, stdt* s, int n) {
62      stdt* c = S;    // child
63      stdt* p;        // parent
64      while (c != NULL) {
65          // go left if c>s, go right if c<s
66          p = c;
67          if (idcmp(c, s) > 0) c = c->left;
68          else c = c->right;
69      }
```

```
70      s->left = NULL;
71      s->right = NULL;
72      if (idcmp(p, s) > 0) p->left = s;
73      else p->right = s;
74      return ++n;
75  }
```

6) bintree_init : 이진검색나무를 초기화하는 함수이다. 첫 노드를 초기화할 때 id는 보초값으로 사용하기 위하여 INT_MAX로 최대값으로 설정하였다.

7) data_insert : 이진검색나무에 새로운 노드를 삽입하는 함수이다. 노드들의 id를 비교하는 idcmp함수를 이용하여 만약 현재 노드보다 크면 우측으로, 작으면 좌측으로 이동하며 현재 노드가 없을(null) 경우 그 위치에 새로운 노드를 추가한다. 요소의 개수를 받아 추가된 경우 +1하여 반환한다.

```
76  int data_load(char* fn, stdt* S) {
77      // input: file name, student struct array
78      // output: number of student
79      FILE* fp;
80      stdt* s;
81      char line[LINE_SIZE];
82      char *lp;
83      int i = 0;
84      if ((fopen_s(&fp, fn, "r")) != NULL) {
85          printf("\nWrong File!");
86          return i;
87      } // return 0 no file
88      while(!feof(fp)) {
89          fgets(line, sizeof(line), fp);
90          // remove \n if exist
91          if (line[strlen(line)-1] == 10) line[strlen(line)-1] = 0;
92          s = (stdt *)malloc(sizeof(stdt));
93          //printf("%s\n", &line);
94          lp = strtok(line, " ");     // get name
95          strcpy(s->name, lp);
96          lp = strtok(NULL, " ");     // get id
97          s->id = atoi(lp);
98          lp = strtok(NULL, " ");     // get major
99          strcpy(s->maj, lp);
100         s->left = NULL;
101         s->right = NULL;
102         i = data_insert(S, s, i);
103     }
104     fclose(fp);
105     printf("Records ard loaded to binary tree !");
106     return i;
107 }
```

8) data_load : 파일 이름과 head 노드를 받아 head 노드 아래(왼쪽)에 이진검색나무를 생성하는 함수이다. 위 문제1)과 마찬가지로 파일을 한 줄씩 line으로 불러와 개행문자를 null 문자로 교체한 뒤 strtok, strcpy를 이용하여 띄어쓰기로 구분된 자료를 name(문자열), id(정수), major(문자열)로 구분하여 새 노드 s에 저장한다. 저장할 때에는 data_insert 함수를 이용하여 새로운 노드를 이진나무에 삽입한다.

```
108 void inorder_print(stdt* s) {
109     while (s != NULL) {
110         inorder_print(s->left);
111         printf("%s %d %s\n", s->name, s->id, s->maj);
112         inorder_print(s->right);
113         return;
114     }
115 }
```

9) inorder_print : 프로그램을 제작할 때 확인하기 위하여 중위순회로 데이터를 print하는 함수이다.

```
int inorder_save(FILE* fp, stdt* s, int n) {
    char l1[LINE_SIZE];
    char l2[LINE_SIZE];
    char zero[ID_SIZE] = "00000000";
    while (s != NULL) {
        n = inorder_save(fp, s->left, n);
        sprintf(l1, "%s ", s->name);
        strncat(l1, zero, 8-digit(s->id));
        sprintf(l2, "%d %s", s->id, s->maj);
        strcat(l1, l2);
        if (n!=1) strcat(l1, "\n");
        fputs(l1, fp);
        n = n-1;
        n = inorder_save(fp, s->right, n);
        return n;
    }
    return n;
}
void data_save(char* fn, stdt* S, int n) {
    FILE* fp;
    char line[LINE_SIZE];
    fopen_s(&fp, fn, "w");
    inorder_save(fp, S, n);
    fclose(fp);
    printf("%d records are saved in %s!", n, fn);
}
void data_find(stdt* S, int id) {
    stdt* s = S->left;
    while (s != NULL) {
        if (s->id == id) {
            printf("found! : [%s %d %s]", s->name, s->id, s->maj);
            return;
        }
        if (s->id > id) s = s->left;
        else s = s->right;
    }
    printf("[%d] is not found.", id);
}
```

10) inorder_save, data_save : 중위순회를 이용하여 저장된 이진검색나무를 파일로 저장하는 함수이다. 재귀함수로 사용하기 위하여, inorder_save 함수가 중위순회로 파일로 저장하고, data_save 함수가 파일 포인터를 불러와준다.

11) data_find : 이진검색나무를 이용하여 찾을 id의 값이 현재 노드보다 작으면 왼쪽, 크면 오른쪽, 같으면 해당 노드의 정보를 출력하는 함수이다. 만약 진행하다 null 노드를 만나게 되면 not found 문구를 출력한다.

```
int data_delete(stdt* S, int id, int n) {
    stdt* p = S;
    stdt* s = S->left;
    stdt* des;
    int d = -1;
    while (s != NULL) {
        if (s->id == id) {
            printf("[%s %d %s] is deleted!", s->name, s->id, s->maj);
            if (s->right == NULL) {
                if (d == 1) p->right = s->left;
                else p->left = s->left;
            }
```

```
166             else if (s->right->left == NULL) {
167                 s->right->left = s->left;
168                 if (d == 1) p->right = s->right;
169                 else p->left = s->right;
170             }
171             else {
172                 des = s->right;
173                 while (des->left->left != NULL) des = des->left;
174                 if (d == 1) p->right = des->left;
175                 else p->left = des->left;
176                 des->left->left = s->left;
177                 des->left = des->left->right;
178             }
179             free(s);
180             return --n;
181         }
182         else if (s->id > id) {
183             p = s;
184             s = s->left;
185             d = -1;
186         }
187         else {
188             p = s;
189             s = s->right;
190             d = 1;
191         }
192     }
193     printf("[%d] is not found", id);
194     return n;
195 }
```

12) data_delete : 중위순회를 이용하여 삭제할 노드를 찾은 뒤, 해당 노드의 우측 노드가 없을 경우 좌측 노드로 대체한다. 만약 우측 노드의 좌측 노드가 없을 경우 해당 노드의 좌측 노드를 우측노드의 좌측 노드로 지정한 뒤, 우측 노드로 대체한다. 만약 있을 경우 우측노드의 최좌측 노드를 해당 노드로 대체한다.

13) 실행 결과 : 프로그램의 main 함수와 실행 결과는 다음과 같다. 우측 out.txt를 보면 학번 순으로 정렬이 잘 된 것을 확인할 수 있다.

```c
int main(void) {
    stdt* S;
    stdt* s = (stdt *)malloc(sizeof(stdt));
    bintree_init(&S);
    int c = 0;
    int n = 0;
    int id;
    char fn[MAX];
    do {
        printf("\n Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : ");
        c = str2int();
        switch (c) {
            case 1: // load
                printf("Input file name : ");
                input(fn);
                n = data_load(fn, S);
                break;
            case 2: // insert
                printf("Input Name : ");
                input(s->name);
                printf("Input Id : ");
                s->id = str2int();
                printf("Input Major : ");
                input(s->maj);
                n = data_insert(S, s, n);
                printf("Data Inserted!");
                break;
            case 3: // find
                printf("Input Id : ");
                id = str2int();
                data_find(S, id);
                break;
            case 4: // delete
                printf("Input Id : ");
                id = str2int();
                n = data_delete(S, id, n);
                break;
            case 5: // Save
                printf("Input file name : ");
                input(fn);
                data_save(fn, S->left, n);
                break;
            case 9:
                inorder_print(S->left);
                printf("number of id: %d", n);
                break;
            default:
                break;
        }
    } while (c != 6);
    printf("Good bye~\n");
    return 0;
}
```

```
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 1
Input file name : infile.txt
Records ard loaded to binary tree !
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 2
Input Name : Baek
Input Id : 91323203
Input Major : CS
Data Inserted!
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 3
Input Id : 91323203
found! : [Baek 91323203 CS]
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 3
Input Id : 91323303
found! : [Choe 91323303 CS]
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 4
Input Id : 91323377
[91323377] is not found
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 4
Input Id : 91323303
[Choe 91323303 CS] is deleted!
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 5
Input file name : out.txt
15 records are saved in out.txt!
 Select one (1:load, 2:insert, 3:find, 4:delete, 5:Save, 6:Quit) : 6
Good bye~
```

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.

out.txt

```
1   Jang 01303003 Comm
2   Jang 01303009 Math
3   Ahn 02303077 CS
4   Park 03333932 Comm
5   Jang 05303033 Archi
6   Lee 23223232 Elec
7   Ahn 88302025 Physics
8   Kim 88302032 CS
9   Park 90302003 Archi
10  Baek 91323203 CS
11  Choi 92323033 Comm
12  Yi 99303092 CS
13  Ahn 99303099 Math
14  Park 99333901 Elec
15  Park 99333939 Archi
```

새로 삽입된 Baek 학생의 정보가 추가된 것을 확인할 수 있으며, 삭제된 Choe의 정보는 잘 삭제된 것을 확인할 수 있다.
파일의 개수도 첫 15개에서 insert후 16개, delete후 15개인 것을 확인할 수 있다.

입력한 수식에 대해 함수값 계산과 중위식 출력이 잘 되는 것을 확인할 수 있다.