자료 구조
연습 #08
리스트 연산 3 (이진탐색트리)

4분반 마감 시간: 5월 15일 오후 11시 59분
5분반 마감 시간: 5월 16일 오후 11시 59분

2019년 1학기

컴퓨터과학과
민경하

내용	easy	moderate	hard
(3) 배열	배열 생성/삽입/삭제 연결리스트 생성	다항식 Set/Map 연결리스트 삽입/삭제	희소 행렬 합, 곱
(4) 연결리스트			
(5) 스택	스택 구현	괄호 매칭	미로찾기 후위 표기
(5) 큐	큐 구현 원형 큐	요제푸스 문제	스케쥴러
(6) 정렬	버블 선택 삽입	합병 쾌속	
(7) 트리	트리 생성 이진 트리 순회	트리 깊이 트리 넓이	동적 트리 조선왕조
(8) 탐색 트리	**BST 생성/탐색**	**BST 제거** Set/Map	AVL tree 2-3 tree Red-black tree B+ tree
(9)우선순위 큐		Heap 생성/제거	
(10) 탐색	이진 탐색 보간 탐색	정적 해쉬	동적 해쉬
(11-13) 그래프	Adjacency matrix Adjacency list DFS BFS	Dijkstra ($O(n^2)$) Floyd Prim/Kruskal Network	Biconnected Strongly connected Dijkstra (O(n log n))

분류	연습문제	Line 수
연습	(1) 파일 입출력	65
연습	(2) 파일 입출력 + 클래스	65
배열	(3) 리스트 연산	156
연결 리스트	(4) 리스트 연산 + 연결 리스트	219
스택	(5) 팰린드롬	192
큐	(6) 요제푸스	126
트리	(7) 트리 생성	244
	(8) 이진 탐색 트리	283
		1,350

문제

- 첨부하는 input.txt로부터 명령어를 입력 받아 이에 대한 결과를 출력하는 프로그램을 작성하시오.
  - CREATE : 학생들의 정보를 저장하는 **연결 리스트**를 할당할 것. (이때, 최대 입력 받는 학생의 수는 **가정하지 않는다**.)
  - LOAD : 다음 txt파일에서 학생들의 정보를 입력 받아 이름순으로 정렬할 것
  - PRINT : 모든 학생들의 정보를 출력할 것
  - INSERT : 주어진 학생의 정보를 입력 받아 이름순으로 이진 탐색 트리에 삽입할 것
  - DELETE : 주어진 이름을 갖는 학생의 정보를 삭제할 것
  - SEARCH : 주어진 이름을 갖는 학생의 정보를 찾아 출력할 것

input.txt

```
CREATE
LOAD list.txt
PRINT
INSERT 김동규 M 서울 컴퓨터과학과 3.10 181 75
INSERT 조원석 M 인천 전기공학과 3.33 175 69
INSERT 김동규 M 인천 전기공학과 3.33 175 69
SEARCH 김동규
SEARCH 오영
PRINT
```

list.txt

```
김준규 M 서울 컴퓨터과학과 4.00 180 80
강태훈 M 서울 전자공학과 3.50 182 90
방승환 F 일산 컴퓨터과학과 3.70 178 78
이상원 M 서울 전기공학과 4.30 167 70
이아현 M 서울 전자공학과 4.40 165 60
김훈 F 분당 컴퓨터과학과 3.80 180 85
신동규 M 남양주 컴퓨터과학과 4.10 179 80
최유성 M 구리 전기공학과 3.00 185 81
허나영 M 서울 컴퓨터과학과 3.70 167 56
남소리 F 분당 전자공학과 3.80 165 60
임종현 M 서울 컴퓨터과학과 3.50 175 75
김혜라 F 일산 컴퓨터과학과 3.60 165 50
이재현 F 파주 전기공학과 3.20 180 77
이충완 F 서울 컴퓨터과학과 4.10 177 69
```

자료구조 (sinfo)

```
class sinfo {
    char name[8];
    char sex;
    char city[8];
    char dept[16];
    float gpa;
    int height;
    int weight;
public:
    void print() { ... }

    char *get_name() { ... }
    void load(char *str) { ... }
    void set(char *tok2, char *tok3, char *tok4, char *tok5, char *tok6, char *tok7, char *tok8) { ... }
};
```

자료구조 (node)

```
class node {
    sinfo data;
    nptr lchild, rchild;
public:
    void init();
    void insert(sinfo tdata);
    int remove(char *dname);
    void search(char *sname);
    void inorder();
    sinfo get_max();

    void process_create();
    void process_load(char *fn);
    void process_print();
    void process_insert(sinfo idata);
    void process_delete(char *dname);
    void process_search(char *sname);
};
```

Main.cpp

```cpp
int main()
{
    FILE *fp = fopen("input.txt", "r+t");
    char input[512];
    char tok1[32], tok2[32], tok3[32], tok4[32], tok5[32], tok6[32], tok7[32], tok8[32], tok9[32];

    node *root = (node *)malloc(sizeof(node));

    while (fgets(input, 512, fp) != NULL) {
        sscanf(input, "%s%s%s%s%s%s%s%s%s", tok1, tok2, tok3, tok4, tok5, tok6, tok7, tok8, tok9);
        if (strcmp(tok1, "CREATE") == 0) {
            root->process_create();
            printf(" %s done ==========\n\n", tok1);
        }
        else if (strcmp(tok1, "LOAD") == 0) {
            root->process_load(tok2);
            printf(" %s done ==========\n\n", tok1);
        }
        else if (strcmp(tok1, "PRINT") == 0) {
            root->process_print();
            printf(" %s done ==========\n\n", tok1);
        }
        else if (strcmp(tok1, "INSERT") == 0) {
            sinfo nitem;
            nitem.set(tok2, tok3, tok4, tok5, tok6, tok7, tok8);
            root->process_insert(nitem);
            printf(" %s done ==========\n\n", tok1);
        }
        else if (strcmp(tok1, "DELETE") == 0) {
            root->process_delete(tok2);
            printf(" %s done ==========\n\n", tok1);
        }
        else if (strcmp(tok1, "SEARCH") == 0) {
            root->process_search(tok2);
            printf(" %s done ==========\n\n", tok1);
        }
        else {
            printf("%s is not a keyword.\n", tok1);
        }
    }

    fclose(fp);

    system("Pause");
    return 0;
}
```

process_create ( )

```
void node::process_create()
{
   this->init();
}
```

process_load ( )

```
void node::process_load(char *fn)
{
   FILE *fp2 = fopen(fn, "r+t");
   sinfo nitem;
   char str[512];
   while (fgets(str, 512, fp2) != NULL) {
      nitem.load(str);
      this->process_insert(nitem);
   }

   fclose(fp2);
}
```

process_print ( )

```
void node::process_print()
{
   this->inorder();
}
```

process_insert (sinfo idata)

```
void node::process_insert(sinfo idata)
{
   this->insert(idata);
}
```

process_delete (char *dname)

```
void node::process_delete(char *dname)
{
   this->remove(dname);
}
```

process_search (char *sname)

```
void node::process_search(char *sname)
{
   this->search(sname);
}
```

search (char *sname)

```
void node::search(char *sname)
{
   char *name = this->data.get_name();
   if (strcmp(name, "-1") == 0) {// degenerate case:
                                //  search to an empty BST
      return;
   }
   if (strcmp(name, sname) == 0) {  // 같은 name을 찾음
      printf("Found: ");
      this->data.print();
   }
```

search (char *sname)

```
    if (strcmp(name, sname) > 0) { // 찾는 이름 (sname)이 작음
        if (this->lchild == NULL) {
            printf("Not found:  %s\n", sname);
        }
        else
            this->lchild->search(sname);
    }
    else if (strcmp(name, sname) < 0) { // sname이 큼
        if (this->rchild == NULL) {
            printf("Not found:  %s\n", sname);
        }
        else
            this->rchild->search(sname);
    }
}
```

remove (char *dname)

```
int node::remove(char *dname)
{
    char *name = this->data.get_name();
    if (strcmp(name, "-1") == 0) {    //  degenerate case:
                                      //  remove from an empty BST
        return 0;
    }
```

```
sinfo node::get_max()
{
    if (this->rchild == NULL)
        return this->data;
    else
        return this->rchild->get_max();
}
```

```
    if (strcmp(name, dname) == 0) {
        printf("Deleting  %s\n", dname);
        if (this->lchild == NULL and this->rchild == NULL) {    // Leaf
            return 1;
        }
        if (this->lchild == NULL and this->rchild != NULL) {    //  right
            this->data = this->rchild->data;
            this->lchild = this->rchild->lchild;
            this->rchild = this->rchild->rchild;
            return 0;
        }
        if (this->lchild != NULL and this->rchild == NULL) {    //  left
            this->data = this->lchild->data;
            this->rchild = this->lchild->rchild;
            this->lchild = this->lchild->lchild;
            return 0;
        }
        if (this->lchild != NULL and this->rchild != NULL) {    //  both
            this->data = this->lchild->get_max();
            char *new_dname = this->data.get_name();
            if (this->lchild->remove(new_dname) == 1)
                this->lchild = NULL;
            return 0;
        }
    }
```

```
    if (strcmp(name, dname) > 0) {
        if (this->lchild != NULL) {
            if (this->lchild->remove(dname) == 1)
                this->lchild = NULL;
        }
        else {
            printf("Not found:  %s\n", dname);
        }
    }
    else if (strcmp(name, dname) < 0) {
        if (this->rchild != NULL) {
            if (this->rchild->remove(dname) == 1)
                this->rchild = NULL;
        }
        else {
            printf("Not found:  %s\n", dname);
        }
    }

    return 0;
}
```

insert (sinfo idata)

```
void node::insert(sinfo idata)
{
    char *name = this->data.get_name();
    if (strcmp(name, "-1") == 0) {    //  degenerate case:
                                      //  insert to an empty BST
        this->data = idata;

        return;
    }
```

```
    char *iname = idata.get_name();
    if (strcmp(name, iname) > 0) {
        if (this->lchild == NULL) {
            this->lchild = (node *)malloc(sizeof(node));
            this->lchild->data = idata;
            this->lchild->lchild = this->lchild->rchild = NULL;
        }
        else
            this->lchild->insert(idata);
    }
    else if (strcmp(name, iname) < 0) {
        if (this->rchild == NULL) {
            this->rchild = (node *)malloc(sizeof(node));
            this->rchild->data = idata;
            this->rchild->lchild = this->rchild->rchild = NULL;
        }
        else
            this->rchild->insert(idata);
    }
    else { // strcmp(name, iname) == 0
        printf("Duplicate data insertion: %s\n", iname);
    }
}
```

주의할 점

- 연습은 MS Word나 hwp를 사용해서 작성하지 말고 반드시 VisualStudio에서 작성해서 컴파일하고 디버그할 것
- Source code (*.cpp) 만 제출하지 말고 반드시 프로젝트 전체를 zip해서 제출할 것
- 가능하면 VisualStudio 2017을 이용할 것