**과제2 연결리스트/트리**



|  |  |
| --- | --- |
| **과 목 :** | **자료구조** |
| **제출 일자 :** |  |
| **담당 교수 :** | **김승태** |
| **학 과 :** | **사회복지학부** |
| **학 번 :** | **20170299** |
| **이 름 :** | **김유빈** |

**(1) 연결 리스트 – 전화번호부 만들기**

<조건>

자료형 => 연결리스트

데이터 – 이름(영어대문자 3글자로), 전화번호

기능(5가지)

①이름으로 찾기

②추가하기

③삭제하기

④정렬하기

⑤목록보기

**1. 연결리스트 종류 선택**

: 뒤로 돌아갈 일이 없으므로 이중 연결리스트 말고 ‘단순 연결리스트’로 선택

typedef struct ListNode {

char name[4];

char phone[14];

struct ListNode\*link;

}ListNode;

**2. 헤드 노드 선택**

: 헤드 노드에도 데이터 넣기(더미 노드 안 만들기)

**3. 연결리스트 초기화**

ListNode\* head = NULL;

**4. 10명의 랜덤한 데이터 생성하기**

: ①새로운 노드(new\_node)를 생성하여 ListNode\* new\_node;

②데이터를 채우고

③새로운 노드를 연결리스트에 삽입하는 방식

★★데이터 생성 주의점

문자 하나하나 입력할 때 문자열 마지막에 NULL 추가하기!!!

* 안 그러면 이상한 문자열(쓰레기)

④데이터 중복 검사하기

: 연결리스트의 탐색함수(search함수)로 새로운 노드의 데이터를 탐색

ListNode\* search(ListNode\* head, char\* data) -> 순차탐색

있으면 다시 반복문 실시(i--)

없으면 새로운 노드를 연결리스트에 삽입하기(insert\_node함수)

**5. ‘이름으로 찾기’, ‘추가하기’ 기능**

if (strcmp(key, "목록") != 0 && strcmp(key, "정렬") != 0 && strcmp(key, "삭제") != 0)

: 이름을 입력했다면

해당 이름을 연결리스트에서 찾기 => ListNode\* target = search(head, key);

해당 이름이 연결리스트에 없다면 => if (target == NULL)

새로운 노드를 생성하여 해당 이름과 입력한 전화번호 넣기

new\_node = (ListNode\*)malloc(sizeof(ListNode));

strcpy(new\_node->name, key);

strcpy(new\_node->phone, number);

새로운 노드를 연결리스트에 삽입하기

* void insert\_node(ListNode \*\*phead, ListNode\* p, ListNode\* new\_node)
* 주의점: p == NULL인 경우는 첫 칸에 삽입

만약, 해당 이름이 연결리스트에 있다면 target->phone(해당 전화번호 출력)

**6. ‘정렬하기’ 기능**

연결리스트는 ‘버블정렬’(양옆 비교)

head

B

Q

A

S

tail

cur next

ListNode\* pre = NULL; //이전 위치

ListNode\* cur = \*phead; //현재 위치

ListNode\* next = cur->link; //다음 위치(현재와 비교할 데이터)

ListNode\* tail = NULL; //정렬된 구간의 맨 처음 위치

버블정렬을 tail(정렬된 구간의 맨 처음)이 head값이 되기전까지 반복 => while (tail != \*phead)

버블정렬 한 회는 next가 tail이 되기 전까지 반복 => while (next != tail)

head

버블정렬로 가장 큰 문자를 null쪽으로 보내기

if (cur->name[i] > next->name[i]) : cur이 next보다 클 때

Q

A[next]

S[cur]

cur->link = next->link;

next->link = cur;

if (cur == \*phead) //헤드값이 바뀌는 경우

phead = next;

else

pre->link = next;

head

A[cur]

S[next]

Q

B

교환이 이루어지면 cur과 next가 가리키는 곳도 바꿔줘야 한다!!!

SWAP(cur, next, temp);

tail

break; // 비교하는 단어 중 하나라도 커서 교환이 이루어지면 반복문 나가기!!!

다음 위치로 이동하기 =>

pre = cur;

cur = cur->link;

next = next->link;

★★버블정렬 한 회가 끝날 때마다 초기화 시키기!

tail = cur; //정렬된 맨 처음 구간(while반복문에서 나온 cur)

pre = NULL;

cur = \*phead;

next = cur->link;

head

A

Q

B

S

tail

next

cur

pre

**7. ‘삭제하기’ 기능**

key(명령어)에 “삭제’라는 단어가 있다면 => while (strcmp(key,"exit") != 0)

key변수에 “삭제” 넣기

strcpy(pkey, key);

token = strtok(pkey, " ");

strcpy(key, token);

삭제할 이름은 변수removed\_name에 저장

token = strtok(NULL, " ");

strcpy(removed\_name, token);

삭제할 노드 찾기 => ListNode\* removed = search(head, removed\_name);

삭제할 노드의 이전 노드(before) 찾기

* ListNode\* psearch(ListNode\* head, char\*data)

ListNode\* p = head;

if (!strcmp(head->name, data)) return NULL; //**헤드노드가 찾는 노드인 경우**(이전 노드는 null반환)

while (**p->link** != NULL) { //마지막 노드 볼 필요x

if (!strcmp(p->link->name, data)) return p; //두 문자열(이름)비교 성공

p = p->link;

}

마지막으로 삭제 실행 => remove\_node(&head, before, removed);

**8. “목록보기” 기능 과 데이터 개수 함수**

연결리스트 요소 출력 함수 void display(ListNode\* head)

연결리스트 요소의 개수 함수 int get\_length(ListNode\* head)

if (head != NULL){ //빈 리스트가 아니라면

while (p != NULL) {

length++;

p = p->link;

}

}

->length 출력

텍스트이(가) 표시된 사진

자동 생성된 설명<결과화면>

텍스트이(가) 표시된 사진

자동 생성된 설명

**(2) 연결 리스트 - 응용**

<문제>

노드에는 학번, 이름, 전화번호 3가지 데이터 저장

하나의 연결리스트에 데이터를 추가할 때마다 학번순, 이름순으로 모든 데이터가 정렬된 상태가 유지되도록 추가해야한다!

->데이터를 저장할 때 미리 정렬해서 저장하는 것

**1. 연결리스트 종류 선택**

->노드 변형

head

typedef struct DlistNode {

int id;

char name[11];

char phone[12];

struct DlistNode\* link;

struct DlistNode\* id\_link;

struct DlistNode\* name\_link;

}DlistNode;

link: 단순 연결리스트

id\_link: 학번순으로 연결된 연결리스트

name\_link: 이름순으로 연결된 연결리스트

헤드 노드: 더미 노드(데이터 x)

**2. 변형된 연결리스트에 새로운 노드 삽입 함수★**

삽입위치의 이전 노드(DlistNode\* pre)

새로운 노드와 비교할 노드(DlistNode\* next)

head

pre

next

new\_node

new\_node의 학번/이름이 next보다 작은 경우

: pre와 next 사이에 새로운 노드 삽입하기

if (new\_node->id < next->id) {

new\_node->id\_link = next;

pre->id\_link = new\_node;

break; }

그것이 아니면(즉, 새로운 노드의 데이터가 next보다 클 경우)

pre, next를 이동하기

이 과정을 while (next != NULL)까지 반복하기

단, 주의할 점이 있다. 밑에 그림 같은 상황에 와도 새로운 노드가 next보다 클 경우

next

pre

즉, 새로운 노드를 마지막에 삽입하는 경우

(이 과정을 while (next != NULL)까지 반복하므로 마지막에 삽입하지 못하고 반복문이 끝나 버림)

else if (next->id\_link == NULL) {

new\_node->id\_link = NULL;

next->id\_link = new\_node;

break;}

next 노드의 다음이 null인 경우 새로운 노드를 마지막에 삽입하기!!

**2. 1만명분 랜덤 데이터 생성**

학번(int형), 이름(문자열), 전화번호(문자열) 를 새로운 노드(new\_node)에 삽입

**3. 데이터 중복 검사**

연결리스트 탐색함수(학번, 이름&전화번호를 기준으로)를 통해

데이터가 이미 연결리스트에 있는지를 검사

if (search\_id(head, new\_node->id) != NULL)

i--;

else if (search\_namephone(head, new\_node->name) != NULL)

i--;

else if (search\_namephone(head, new\_node->phone) != NULL)

i--;

중복이지 않으면 삽입함수

else

insert\_node(head, new\_node);

**4. 연결리스트 출력 함수**

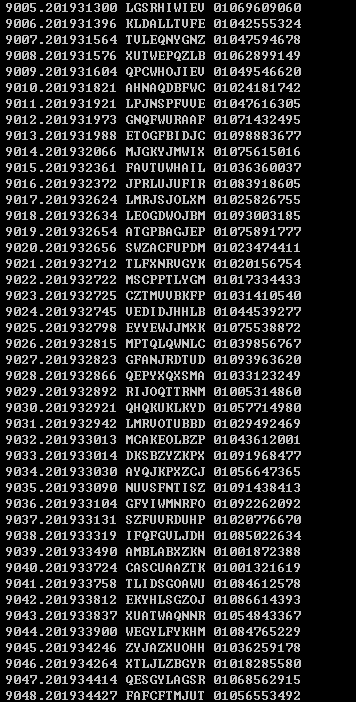
void display(DlistNode\* head, int choice)

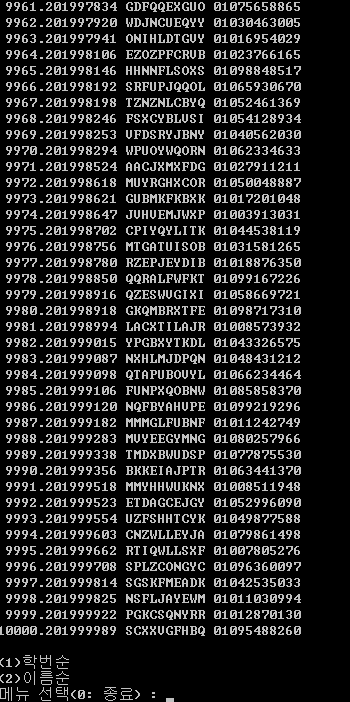
choice == 1 이면 p = p->id\_link;

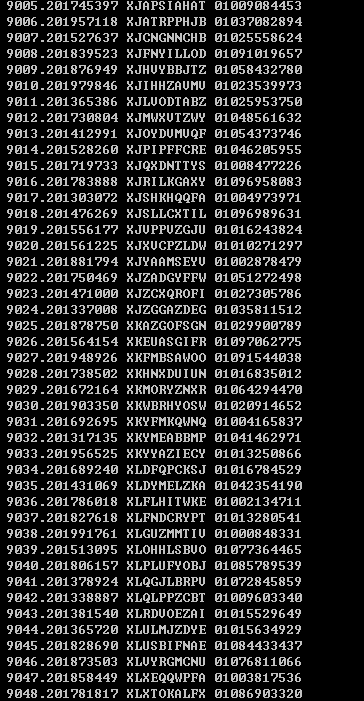
choice == 2 이면 p = p->name\_link;

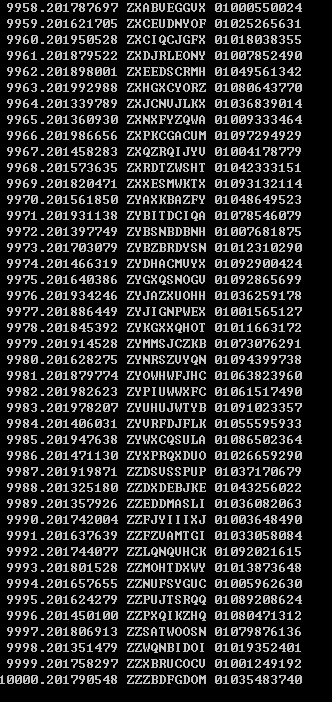
화면에 학번 또는 이름순의 연결리스트를 출력

<결과화면>

1. 1를 입력할 때(학번순)



2. 2를 입력할 때(이름순)



**(3) 이진 탐색 트리 구성하기**

<문제>

1. 데이터 5만개 생성하여 이진 탐색 트리에 저장

->단, 마지막 1000개의 학번은 따로 배열에 저장

2. 이진 탐색 트리 삽입 함수

3. 이진 탐색 트리 중위 순회 함수 -> 스택으로 구현!

+ 나중의 노드가 먼저 노드에 비해 학번이 뒤쪽이어야 한다

4. 이진 탐색 트리 탐색 함수

<이진 탐색 트리의 노드 구조>

|  |  |  |
| --- | --- | --- |
| left | data | right |

**1. 데이터 5만개 생성하여 이진 탐색 트리에 저장**

학번(int형), 이름, 전화번호(문자열)을 생성한다.

단, 학번 생성시 이진 탐색 트리 탐색함수를 이용하여 학번 중복 검사 실시

->중복이면 다시 학번 생성

->중복되지 않으면 new\_node 생성

마지막 1000개의 학번을 배열(myid)에 따로 저장

if (i >= MAX\_NUM - 1000) //가장 마지막 1000명의 학번 저장

myid[k++] = new\_node->key;

**2. 이진 탐색 트리 삽입 함수:** void insert\_node(TreeNode\*\* root, TreeNode\* new\_node)

데이터가 넣진 new\_node를 이진 탐색 트리에 삽입

이진 탐색 트리는 왼쪽 트리 <= 루트 <= 오른쪽 트리

리스트나 트리에 삽입하기 위해서는 이전 노드 위치(p)가 필요

new\_node의 키값(학번)을 먼저 탐색한다

탐색하고 정지한 위치(null)의 이전(p)이 new\_node를 가리키면 삽입 끝!

단, 빈 트리인 경우( p == null 즉, root == null) new\_node가 루트노드가 된다!

**★3. 이진 탐색 트리 중위 순회 함수 -> 스택으로 구현!**

**+ 나중의 노드가 먼저 노드에 비해 학번이 뒤쪽이어야 한다**

중위 순회 방법은 1)재귀 함수 2) 스택

하지만 나중에 방문한 노드(위쪽 노드)가 먼저 방문한 노드(아래쪽 노드)에 비해 학번이 뒤쪽임을 검사하기 위해서는 재귀 함수보다 스택이 구현하기 쉽다.

스택 구현을 위해 -> 스택 만들기, 스택이 비었는지 검사, 스택이 차 있는지 검사, 스택에 push, 스택에서 pop하는 함수 필요

중위 순회 함수: void inorder(TreeNode\* root)

node

<p>

0

TreeNode\* node = root

int p = 0 -> 먼저 방문한 노드의 키 값

일단 왼쪽 노드를 방문하여 스택에 집어 넣기

for (;;) {

for ( ; node ; node = node->left) //node가 null이면 종료

Push(pstack, node);

일단 왼쪽 노드 꺼내서 출력하기

node = Pop(pstack); -> ‘1’

if (!node) break; //node == null 즉, 스택이 비었음(끝)

스택이 비지 않았다면 pop한거 출력하기

printf("%d ", node->key);

먼저 키값(p)가 나중의 키값(node->key)보다 작아야 함

if (node->key > p)

printf(“(OK)\n");

else

printf("(ER)\n");

이전 키 값을 올려준다 -> p = node->key; ->p = ‘1’

루트 출력한 다음은 오른쪽 노드 방문하기

node = node->right;

하지만 null이므로(‘1’의 오른쪽 서브노드) 스택에서 끄집어내기(pop) -> ‘4’

이를 화면에 출력하고 오른쪽 노드 방문

또 오른쪽 노드가 null이므로 스택에서 pop -> ‘15’

15 출력하고 오른쪽 노드로 이동 -> node = ‘20’노드

스택에 ‘20’노드부터 시작해서 왼쪽 노드 넣기

node

‘16’노드 pop되어 출력 -> ‘16’

‘20’노드 pop되어 출력 이때 p = 16 -> ‘20’

‘20’노드 오른쪽으로 이동하여 ‘25’노드 스택에 집어넣기

‘25’노드 출력하여 끝!! -> ‘25’

**4. 이진 탐색 트리 탐색 함수**

: TreeNode\* search(TreeNode\* root, int key)

트리의 node의 key보다 key가 작으면 왼쪽으로

key가 더 크면 오른쪽으로 노드를 이동한다

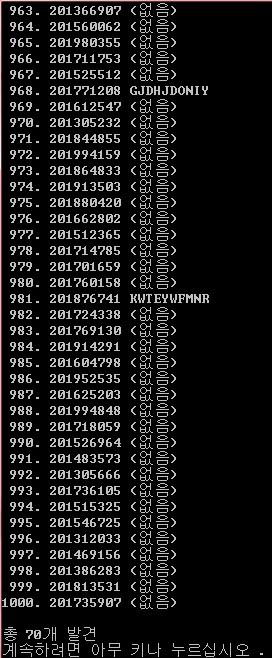
둘이 같아지면 반환

<결과화면>

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명