**프로그램 보고서 #4**

1612169 배현진

(1) 사용한 각 함수에 대한 설명

1. void NodeInit(List \*plist) {

plist->head = NULL;

plist->tail = NULL;

}

- Input: 노드를 삽입할 리스트인 \*plist

- Output: void

- 알고리즘 설명: 리스트를 사용하기 전 NULL과 0으로 초기화 시켜주는 함수이다.

2. void NodeInsert(List \*plist, int data) {

node \*newnode = (node\*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

if (plist->head == NULL)

plist->head = newnode;

else

plist->tail->next = newnode;

plist->tail = newnode;

}

- Input: 노드를 삽입할 \*plist와 노드에 들어갈 값인 data

- Output: void

- 알고리즘 설명:

(1) 새로은 newnode 노드를 만들고 초기화 시킨다.

(2) 만약 plist의 head가 null이라면 newnode를 head로 만들어준다.

(3) 그렇지 않다면 plist의 tail 다음에 newnode를 연결시켜준다.

(4) newnode를 tail로 지정한다.

3. int returnNode(List \*list, int middle) {

int i, searchData;

node \*temp;

temp = list->head;

for (i = 1; i <middle; i++) {

temp = temp->next;

}

searchData = temp->data;

return searchData;

}

- Input: 출력할 리스트인 \*plist, 리스트의 중심을 찾기 위한 위치인 middle

- Output: 중간에 위치한 data인 searchData

- 알고리즘 설명:

(1) temp 노드를 새로 만들어 매개 변수로 받은 리스트의 head 노드로 초기화한다.

(2) for문을 돌며 temp를 middle 위치까지 돌려준다.

(3) middle 위치에 도착을 하게 되면 searchData에 temp의 data를 넣어준다.

(4) searchData를 리턴한다.

4. int binsearch(List \*list, int searchnum, int left, int right) {

int middle, searchData;

while (left <= right) {

middle = (right + left) / 2;

printf("\ncurrent middle: %d", middle);

searchData = returnNode(list, middle);

if (searchData > searchnum)

right = middle - 1;

else if (searchData < searchnum)

left = middle + 1;

else

return middle;

}

return -1;

}

- Input: 출력할 리스트인 \*plist, 리스트에서 검색할 숫자인 searchnum, 리스트의 가장 왼쪽 위치, 리스트의 가장 오른쪽 위치

- Output: searchnum의 위치

- 알고리즘 설명:

(1) temp 노드를 새로 만들어 매개 변수로 받은 리스트의 head 노드로 초기화한다.

(2) 왼쪽 위치가 오른쪽 위치보다 작거나 같은 동안 while문을 수행한다.

1. 리스트의 중간 위치인 middle을 구한다. 그런 다음 현재 middle를 출력한다.

2. returnNode 함수로 현재 middle에 위치한 리스트의 값을 구해 searchData에 넣어준다.

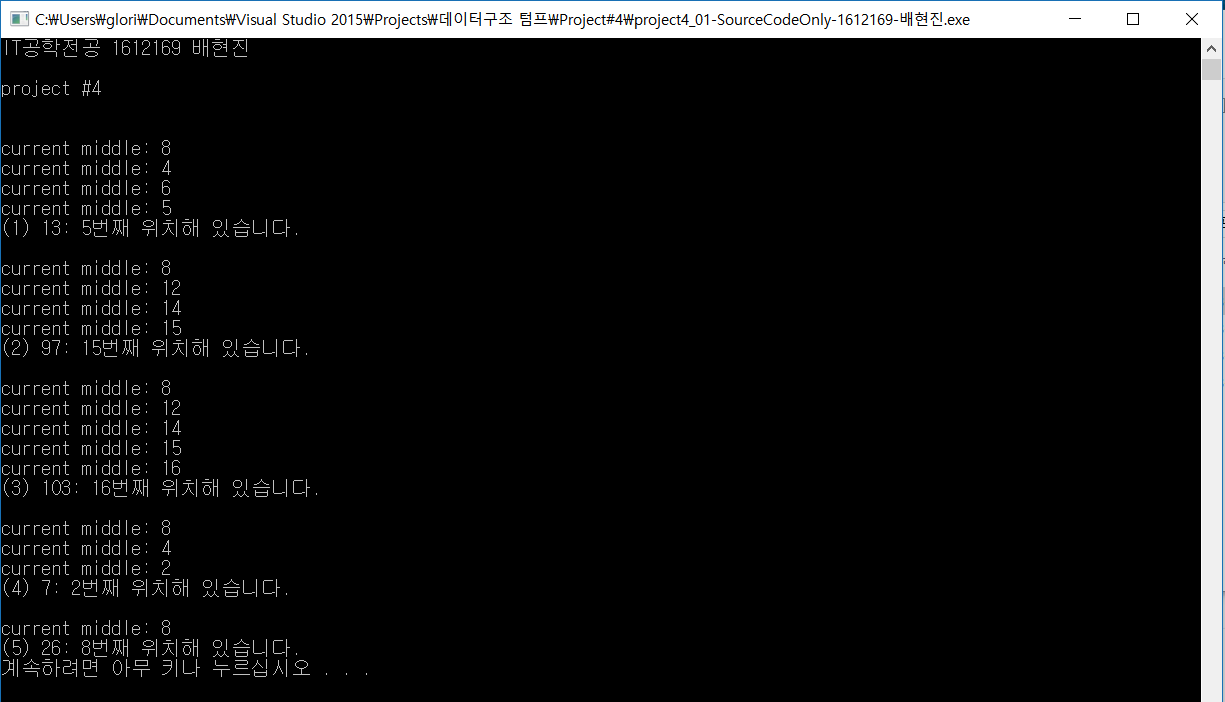
3. 만약 searchData가 searchnum보다 큰 경우, 왼쪽 절반으로 탐색 범위를 줄인다.

4. 만약 searchData가 searchnum보다 작은 경우, 오른쪽 절반으로 탐색 범위를 줄인다.

5. searchData와 searchnum이 같은 경우, 그 위치를 리턴한다.

(3) 어느 경우에도 속하지 않는 경우에는 -1을 출력한다.

(2) 실행 결과 화면 단계별 Capture



(3) 주의사항

- Binary Search 연산을 수행할 때 위치가 1부터 시작하여 37에서 끝난다.