**프로그램 보고서 #2**

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) 리스트에 새로 추가할 노드인 newnode를 새로 만들어 초기화시킨다.

(2) 만약 plist의 head가 null이면 newnode를 head로 만든다.

(3) 그렇지 않다면 plist의 tail에 newnode를 연결해준다.

(4) newnode를 새로운 tail로 지정해준다.

3. void print(List \*plist)

{

node \*temp = plist->head;

while (temp)

{

printf(" %d", temp->data);

temp = temp->next;

}

printf(" \n");

}

- Input: 출력할 리스트인 \*plist

- Output: void

- 알고리즘 설명:

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

(2) for문을 돌며 temp의 데이터를 출력한다. 그런 다음 temp를 temp의 다음 노드로 만들어준다.

4. void swap(node \*node1, node \*node2)

{

int tmp = node1->data;

node1->data = node2->data;

node2->data = tmp;

}

- Input: 서로 값을 바꿀 두 노드 \*node1, \*node2

- Output: void

- 알고리즘 설명:

(1) temp 변수를 새로 만들어 node1의 data로 초기화 시켜 준다.

(2) node1의 data를 node2의 data로 바꾼다.

(3) node2의 data를 temp로 바꾼다.

5. void sort(List \*plist) {

node \*min;

node \*temp = plist->head;

node \*savecur;

node \*curhead = plist->head;

while (curhead->next) {

min = curhead;

savecur = curhead->next;

while (savecur) {

if (savecur->data < min->data)

min = savecur;

savecur = savecur->next;

}

swap(curhead, min);

curhead = curhead->next;

printf("Sorted Array: \n");

print(plist);

}

}

- Input: 값을 정렬할 리스트 \*plist

- Output: void

- 알고리즘 설명:

(1) 현재 head 노드의 다음이 null이 아닌 동안 while문을 수행한다.

(1) min 노드를 현재 head 노드로 초기화하고, savecur 노드를 현재 head노드의 다음 노드로 초기화 시킨다.

(2) savecur이 null 아닌 동안 while문을 수행한다.

1. 만약 현재 노드의 data가 min의 data보다 작다면 min을 현재 노드로 다시 초기화 시킨다.

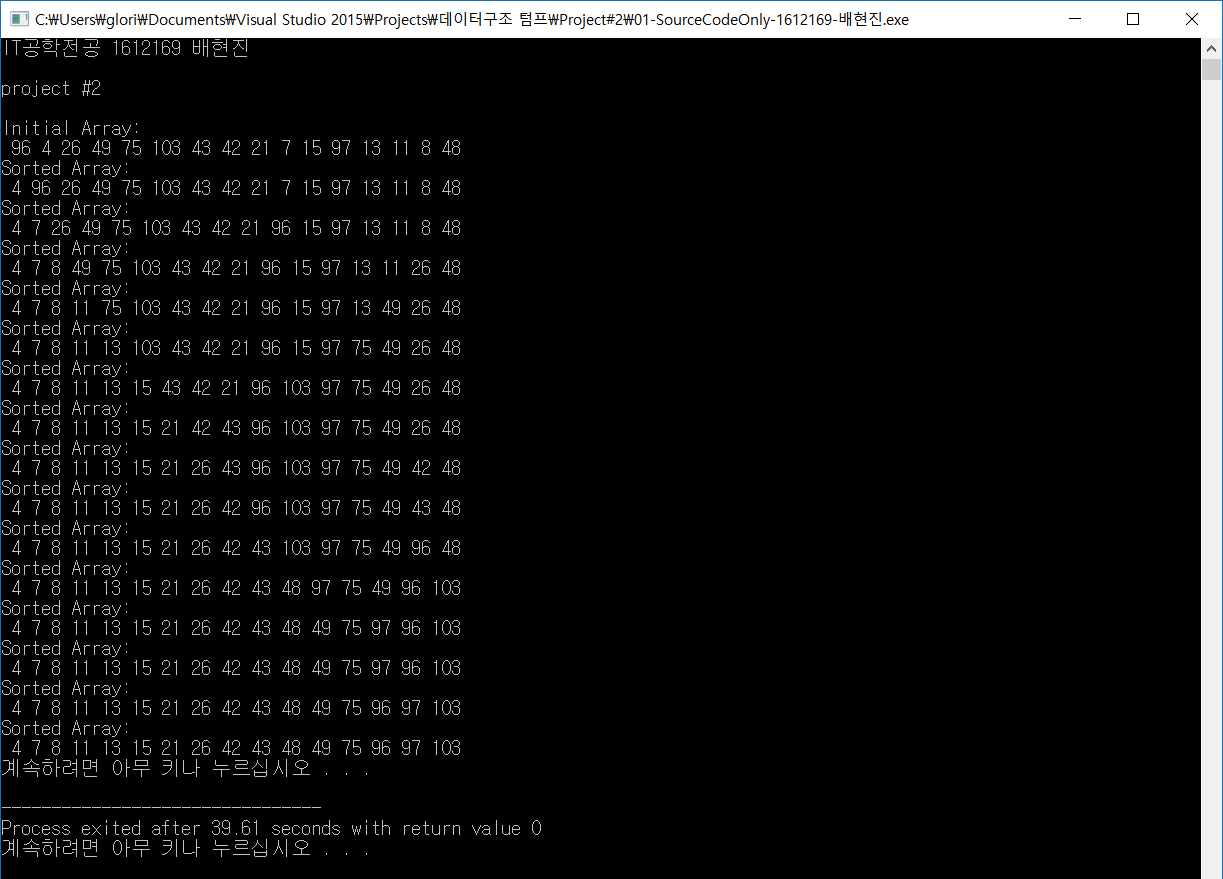
2. 그 다음 노드로 넘어간다.

(3) 현재 head 노드와 min 노드를 swap 한다.

(4) 현재 head 노드의 다음 노드로 현재 head 노드를 옮겨준다.

(5) 앞의 단계에서 정렬된 list를 출력한다.

(2) 실행 결과 화면 단계별 Capture



<전체화면>