◇ 문제 및 요구하는 것

**6장 연습문제 22**

**- SortedList의 ADT에 포함된 모든 연산을 구현하라.**

**- 위의 연산을 이용하여 다음과 같이 주어진 배열을 연결리스트로 구성하고, 출력하라.**

// 1~100의 범위에 속한 난수 30개를 생성하고, 이들을 포함하는 배열 a를 구성하고, // 출력하라.

// 배열 a에 포함된 각 요소를 순서대로 가져와 노드를 생성하여 연결 리스트를 구성하라.

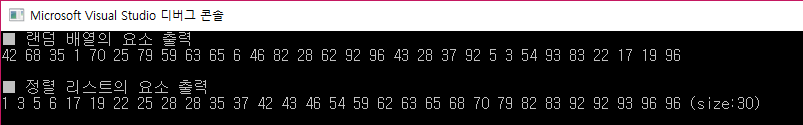
// 단, 리스트에 포함된 노드들은 항상 오름차순으로 정렬된 상태를 유지한다.

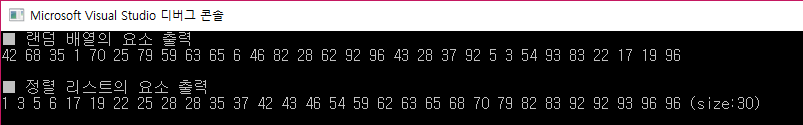
◇ 제목

◇ 제목

◇ 제목

◇ 프로그램 실행 결과



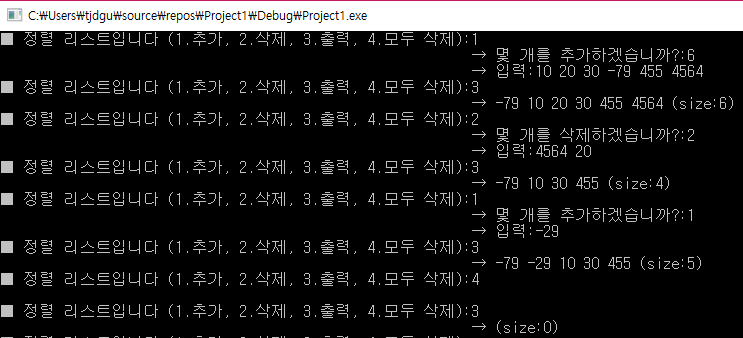


◇ 제목

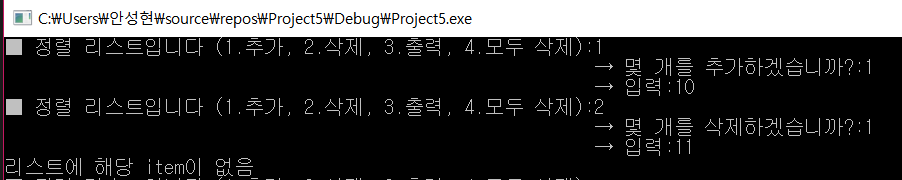
◇ 제목

◇ 제목

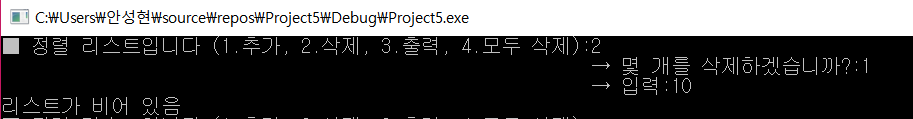
**1. 추가/ 삭제/ 출력/ 모두 삭제 < 정상 >**



**2. 추가되지 않은 요소 삭제 < 비정상 >**



**3. 빈 리스트에서 삭제 < 비정상 >**



◇ 부가적인 실행 결과

◆ IOPE 차트 만들기 (연결 리스트 추가 및 정렬 함수)

|  |  |
| --- | --- |
| 차트 |  |
| **INPUT** | a: a배열을 가리키는 포인터 변수, num: a배열의 크기 |
| **OUTPUT** | List: 연결 리스트로 구현된 정렬 리스트 |
| **PROCESSING** | // a배열의 요소 전부를 list(연결 리스트) 요소에 추가  // add함수는 insert\_first와 같은 연산  for (i<-0 to num-1) do  list <- add(list, a[i])  // Exchange Sort 알고리즘을 이용해서 리스트 정렬  for (i <- (list) to i->link->link=NULL)  for (j <- (i->link) to i->link=NULL)  if (i->data > j->data) then  tmp <- (i->data)  i->data <- (j->data)  j->data <- (tmp)  return list // 정렬된 리스트를 반환 |
| **EXAMPLE** | 만약 a배열이 [ 2 1 0 ]이면 list는 { [2 | p] → [1 | p] → [0 | p] → NULL } 형태가 된다. 여기서 선택 정렬 알고리즘을 이용하고, list->data만 정렬하면  2 1 0 → 1 2 0 → 0 2 1 / (다음 반복) 0 2 1-> 0 1 2 형태로 만들어 진다. |

◆ SPARKS 언어로 알고리즘 작성하기

Procedure gen\_sortedList (a:int 배열 포인터, num:integer) : SortedNode\_ptr

List <- NULL; < list: (SoltedNode\_ptr) >

for (i<-0 to num-1) do < i: (integer) >

list <- add(list, a[i])

repeat

// 교환 정렬(Exchange Sort) 알고리즘 이용

for (i <- (list) to i->link->link=NULL) < i,j: (SoltedNode\_ptr) >

for (j <- (i->link) to i->link=NULL)

if (i->data > j->data) then

tmp <- (i->data) < tmp: (integer) >

i->data <- (j->data)

j->data <- (tmp)

endif

repeat

repeat

return list

End Procedure

#include <stdio.h>

#include <stdlib.h>

#define max 100

typedef struct Node {

int data;

struct Node\* link;

}SortedNode;

typedef SortedNode \* SortedNode\_ptr;

int get\_length(SortedNode\_ptr list) {

SortedNode\_ptr p = list;

int num = 0;

while (p != NULL) {

p = p->link;

num++;

}

return num;

}

int is\_empty(SortedNode\_ptr list) {

if (list == NULL)

return 1;

else

return 0;

}

SortedNode\_ptr clear(SortedNode\_ptr list) {

if (!is\_empty(list)) {

free(list);

list = NULL;

}

else

printf("리스트가 비어 있음");

return list;

}

int is\_in\_list(SortedNode\_ptr list, int item) {

int idx = -1;

for (SortedNode\_ptr i = list; i; i = i->link) {

idx++;

if (i->data == item)

return idx; // item이 있는 인덱스 반환

else if (i->link == NULL) // 마지막 요소까지 Item을 못 찾음

return -1;

}

}

SortedNode\_ptr add(SortedNode\_ptr list, int item) {

SortedNode\_ptr p = (SortedNode\_ptr)malloc(sizeof(SortedNode));

p->data = item;

p->link = list;

list = p;

return list;

}

SortedNode\_ptr delete(SortedNode\_ptr list, int item) {

if (!is\_empty(list)) {

SortedNode\_ptr p;

int tmp; // item이 있으면, item이 있는 노드 인덱스

if ((tmp = is\_in\_list(list, item)) >= 0) {

if (tmp == 0) { // 첫 번째 노드에 item있으면, delete\_first

p = list;

list = p->link;

free(p);

}

else { // 다른 노드에 item있으면, delete

SortedNode\_ptr pre = list;

for (int i = 1; i < tmp; i++) {

pre = pre->link;

}

p = pre->link;

pre->link = p->link;

free(p);

}

}

else {

printf("리스트에 해당 item이 없음\n");

}

}

else

printf("리스트가 비어 있음\n");

return list;

}

void display(SortedNode\_ptr list) {

printf("\n■ 정렬 리스트의 요소 출력\n");

for (SortedNode\_ptr i = list; i; i = i->link)

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

printf("(size:%d)\n", get\_length(list));

}

void gen\_array(int a[], int num) {

for (int i = 0; i < num; i++)

a[i] = (rand() % 100) + 1;

}

void print\_array(int a[], int num) {

printf("■ 랜덤 배열의 요소 출력\n");

for (int i = 0; i < num; i++)

printf("%d ", a[i]);

printf("\n");

}

SortedNode\_ptr gen\_sortedList(int a[], int num) {

SortedNode\_ptr list=NULL;

for (int i = 0; i < num; i++)

list = add(list, a[i]);

for (SortedNode\_ptr i = list; i->link; i = i->link) {

for (SortedNode\_ptr j = i->link; j; j = j->link) {

if (i->data > j->data) {

int tmp = i->data;

i->data = j->data;

j->data = tmp;

}

}

}

return list;

}

int main(void) {

int a[30];

SortedNode\_ptr list = NULL;

// 1~100의 범위에 속한 난수 30개를 배열에 생성하고 출력

gen\_array(a, 30);

print\_array(a, 30);

// 배열 a에 포함한 요소를 순서대로 가져와 연결 리스트 구성

list = gen\_sortedList(a, 30);

// 리스트에 포함된 모든 항목을 순서대로 출력하라

display(list);

return 0;

}

◆ C언어로 코딩하기

// Sorted-List 연산 모두 구현 < add함수는 아래와 같이 수정 >

SortedNode\_ptr sort(SortedNode\_ptr list) {

for (SortedNode\_ptr i = list; i->link; i = i->link) {

for (SortedNode\_ptr j = i->link; j; j = j->link) {

if (i->data > j->data) {

int tmp = i->data;

i->data = j->data;

j->data = tmp;

}

}

}

return list;

}

SortedNode\_ptr add(SortedNode\_ptr list, int item) {

SortedNode\_ptr p = (SortedNode\_ptr)malloc(sizeof(SortedNode));

p->data = item;

p->link = list;

list = p;

// 오름차순 정렬

sort(list);

return list;

}

SortedNode\_ptr input\_processing(int choice, SortedNode\_ptr list) {

int a[max] = { 0 }, num = 0;

scanf\_s("%d", &num);

printf(" → 입력:");

for (int i = 0; i < num; i++) {

scanf\_s("%d", &a[i]);

}

// choice에 따른 함수 호출

switch (choice) {

case 1:

for (int i = 0; i < num; i++)

list = add(list, a[i]);

break;

case 2:

for (int i = 0; i < num; i++)

list = delete(list, a[i]);

break;

}

return list;

}

int main(void) {

SortedNode\_ptr list = NULL;

int choice, item;

while (1) {

printf("■ 정렬 리스트입니다 (1.추가, 2.삭제, 3.출력, 4.모두 삭제):");

scanf\_s("%d", &choice);

switch (choice) {

case 1: // 추가

printf(" ");

printf("→ 몇 개를 추가하겠습니까?:");

list = input\_processing(1, list);

break;

case 2: // 삭제

printf(" ");

printf("→ 몇 개를 삭제하겠습니까?:");

list = input\_processing(2, list);

break;

case 3: // 출력

printf(" → ");

display(list);

break;

case 4: // 모두 삭제

list = clear(list);

printf("\n");

break;

}

}

}

◆ 부가적인 실행 결과를 위한 코드

: delete, clear, is\_empty, gen\_length 함수가 잘 실행되는지 보려면 부가적인 실행 코드가 필요하다. 다음은 사용자가 직접 정렬 리스트를 추가/삭제 하는 프로그램 함수이다.

◇ 레포트 작성 후기

Delete함수를 만들 땐 노드가 한 개만 존재할 시, delete\_first를 이용해서 노드를 완전히 없앴고 노드가 여러 개 존재할 시, is\_in\_list함수에서 item이 존재한 노드 인덱스를 가져오고, for문을 통해 pre노드를 찾은 후, delete함수를 이용했습니다.

Add함수를 만들 땐 insert\_first를 해서 계속 아이템이 앞에서 채워지게 만들었습니다.

정렬 같은 경우, add를 하고 정렬을 하면 delete를 해도 오름차순 정렬이 유지되기 때문에 손쉽게 해결할 수 있었습니다.

리스트 자료구조를 활용함으로써, 함수 작성에 더욱 자신감이 생겼습니다.