**1. 구조체와 노드 생성**

typedef struct Node {

int data;

struct Node\* next;

} Node;

링크드리스트의 각 원소를 나타내는 Node 구조체를 정의했다. 각 노드는 data와 next를 저장한다.

Node\* createNode(int data) {

Node\* nNode = (Node\*)malloc(sizeof(Node));

nNode->data = data;

nNode->next = NULL;

return nNode;

}

새로운 노드를 동적으로 할당하는 함수이다. 사용자가 입력한 숫자를 저장하고, next를 NULL로 초기화한다.

**2. 노드 삽입**

void sortedInsert(Node\*\* head, Node\* nNode) {

Node\* current;

if (\*head == NULL || (\*head)->data >= nNode->data) {

nNode->next = \*head;

\*head = nNode;

} else {

current = \*head;

while (current->next != NULL && current->next->data < nNode->data) {

current = current->next;

}

nNode->next = current->next;

current->next = nNode;

}

}

sortedInsert() 함수는 새로운 노드를 정렬된 위치에 삽입하는 역할을 한다. 리스트가 비어있거나 새로운 값이 최소값이면 맨 앞에 삽입한다. 최소값이 아니라면 current 포인터와 반복문을 통해 리스트를 돌며 현재 노드의 next값이 새로운 노드값보다 큰 노드를 찾을 때까지 이동한다.

**3. 삽입 정렬 알고리즘**

void insertionSort(Node\*\* a) {

Node\* sort = NULL;

Node\* current = \*a;

while (current != NULL) {

Node\* next = current->next;

sortedInsert(&sort, current);

current = next;

}

\*a = sort;

}

기존 리스트를 하나씩 확인하면서 정렬된 리스트에 삽입한다. 기존 리스트에서 current 노드를 하나씩 가져와 sortedInsert() 함수를 통해 정렬된 리스트(sorted)에 추가합니다. 마지막으로 정렬된 리스트의 헤드를 원래 리스트의 헤드로 변경한다.

**4. 리스트 출력 함수**

c

복사편집

void printList(Node\* head) {

while (head != NULL) {

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

head = head->next;

}

printf("/n");

}

연결 리스트의 모든 원소를 출력하는 함수이다.

**5. 메인 함수**

int main() {

Node\* head = NULL;

int n, num;

printf("정렬할 숫자의 개수: ");

scanf("%d", &n);

printf("정렬할 숫자: ");

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

scanf("%d", &num);

Node\* newNode = createNode(num);

newNode->next = head;

head = newNode;

}

printf("정렬 전:\n");

printList(head);

insertionSort(&head);

printf("정렬 후:\n");

printList(head);

return 0;

}

n을 입력받아 정렬할 숫자의 개수를 결정한다. 사용자가 n개의 숫자를 입력하면 createNode()를 이용해 연결 리스트에 추가한다. 입력받은 리스트를 출력한 후, insertionSort()를 호출하여 정렬한다. 정렬된 리스트를 다시 출력한다.