1. **LinkedListInsert**

#include <stdio.h>

#include <stdlib.h>

// Structure를 이용해서 node 정의

struct node {

    char data;

    struct node\* next;

};

void insertNode(struct node\* head) {

// 삽입하고자 하는 data E를 가진 newNode노드 생성

    struct node\* newNode = (struct node\*)malloc(sizeof(struct node));

    newNode->data = 'E';

    newNode->next = NULL;

// B를 검색해서 찾기 current에 찾은 B의 next에 newNode 삽입 후 계속 진행

    struct node\* current = head;

    while (current != NULL && current->data != 'B') {

        current = current->next;

    }

// B찾으면 뒤에 E를 넣고

    if (current != NULL) {

        newNode->next = current->next;

        current->next = newNode;

    }

}

int main() {

// 노드 만들기

    struct node\* head = NULL;

    struct node\* second = NULL;

    struct node\* third = NULL;

    struct node\* fourth = NULL;

// Memory 할당하기

    head = (struct node\*)malloc(sizeof(struct node));

    second = (struct node\*)malloc(sizeof(struct node));

    third = (struct node\*)malloc(sizeof(struct node));

    fourth = (struct node\*)malloc(sizeof(struct node));

// Data 지정하기

    head->data = 'A';

    second->data = 'B';

    third->data = 'C';

    fourth->data = 'D';

// 노드 연결하기

    head->next = second;

    second->next = third;

    third->next = fourth;

    fourth->next = NULL;

// E 삽입하기

    insertNode(head);

// 결과를 출력

    struct node\* current = head;

    while (current != NULL) {

        printf("%c ", current->data);

        current = current->next;

    }

// memory 반환 memory leaking 방지

    free(head);

    free(second);

    free(third);

    free(fourth);

    return 0;

}

**결과**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Step 1. Linked list 생성

Step 2. Insert 함수 + Step 3. B와 C 사이를 찾는 함수

Step 4. Main에서 실행

Feedback

// 결과를 출력하는 코드를 함수로 만들어서 처리하면 앞으로도 이용할 수 있을 것!

// malloc() 사용 시에는 memory allocation error 주의!

// E를 scanf로 받아서 좀 더 general function으로 만들면 다른 곳에도 활용가능!

// B가 없는 경우나 B와 C 사이에 간격이 있는 경우 등 고려하지 못한 변수가 많음!

1. **CircularLinkedListDelete**

#include <stdio.h>

#include <stdlib.h>

// Structure로 노드 정의하기 int data type

struct Node {

   int data;

   struct Node \*next;

   struct Node \*prev;

};

// 기준 설정

struct Node \*head = NULL;

// arr[] 형태로 int data와 크기를 받아서 넣어준다

void createList(int arr[], int n) {

   int i, data;

   struct Node \*newNode, \*temp;

   head = (struct Node\*)malloc(sizeof(struct Node));

// 시작 지점 설정

   head->data = arr[0];

   head->next = NULL;

   head->prev = NULL;

   temp = head;

// for loop을 통해서 배정

   for (i = 1; i < n; i++) {

      newNode = (struct Node\*)malloc(sizeof(struct Node));

      newNode->data = arr[i];

      newNode->next = NULL;

      newNode->prev = temp;

      temp->next = newNode;

      temp = temp->next;

   }

// Memory leaking 방지 circular 구조로!

   temp->next = head;

   head->prev = temp;

   printf("\nCircular doubly linked list created successfully.\n");

}

void displayList() {

   struct Node \*temp;

   if (head == NULL) {

      printf("List is empty.\n");

   } else {

      printf("Data in the list:\n");

      temp = head;

      do {

         printf("%d\n", temp->data);

         temp = temp->next;

      } while (temp != head);

   }

}

void deleteNode(int data) {

   struct Node \*temp;

   if (head == NULL) {

      printf("List is empty.\n");

      return;

   }

   if (head->data == data) {

      temp = head;

      head = head->next;

      head->prev = temp->prev;

      temp->prev->next = head;

      free(temp);

      printf("Node with data %d deleted successfully.\n", data);

      return;

   }

   temp = head->next;

   while (temp != head) {

      if (temp->data == data) {

         temp->prev->next = temp->next;

         temp->next->prev = temp->prev;

         free(temp);

         printf("Node with data %d deleted successfully.\n", data);

         return;

      }

      temp = temp->next;

   }

   printf("Node with data %d not found in the list.\n", data);

}

int main() {

   int n, arr[100], data;

   printf("Enter the number of nodes: ");

   scanf("%d", &n);

   printf("Enter %d data values: ", n);

   for (int i = 0; i < n; i++) {

      scanf("%d", &arr[i]);

   }

   createList(arr, n);

   displayList();

   printf("Enter data of node to delete: ");

   scanf("%d", &data);

   deleteNode(data);

   displayList();

   return 0;

}

**결과**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Step 1. Circular doubly linked list 생성

Step 2. Delete 함수 + Step 3. 60을 찾는 함수

Step 4. 출력 함수

Step 5. Main에서 실행

Feedback

// scanf로 받아준 data input 검증과정이 필요!

// 삭제할 노드가 없는 경우에는 error 메시지를 출력하는 코드로 바꾸자!

// input arr[100]를 do-while loop를 통해서 만드는데 이 값보다 큰 경우에 error 메시지를 출력하는 검증과정이 필요!

// Memory allocation 문제 head node의 경우! 전체 list를 free 시켜주는 함수를 통해서 개선!

1. **DoublyLinkedListAdd10toAll**

#include <stdio.h>

#include <stdlib.h>

// Doubly linked 이므로 next, prev 2개로

struct Node {

   int data;

   struct Node \*next;

   struct Node \*prev;

};

struct Node \*head = NULL;

void createList(int arr[], int n) {

   int i;

   struct Node \*newNode, \*temp;

   head = (struct Node\*)malloc(sizeof(struct Node));

// Error 방지 초기값 설정!

   head->data = arr[0];

   head->next = NULL;

   head->prev = NULL;

   temp = head;

// Loop를 통해서 연결

   for (i = 1; i < n; i++) {

      newNode = (struct Node\*)malloc(sizeof(struct Node));

      newNode->data = arr[i];

      newNode->next = NULL;

      newNode->prev = temp;

      temp->next = newNode;

      temp = temp->next;

   }

   printf("Doubly linked list created successfully.\n");

}

void displayList() {

   struct Node \*temp;

   if (head == NULL) {

      printf("List is empty.\n");

   } else {

      printf("Data in the list:\n");

      temp = head;

      while (temp != NULL) {

         printf("%d\n", temp->data);

         temp = temp->next;

      }

   }

}

void addValueToList(int val) {

   struct Node \*temp;

   if (head == NULL) {

      printf("List is empty.\n");

   } else {

      temp = head;

      while (temp != NULL) {

         temp->data += val;

         temp = temp->next;

      }

      printf("Added %d to all data values in the list.\n", val);

   }

}

int main() {

   int n, arr[100], val;;

   printf("Enter the number of nodes: ");

   scanf("%d", &n);

   printf("Enter %d data values: ", n);

   for (int i = 0; i < n; i++) {

      scanf("%d", &arr[i]);

   }

   createList(arr, n);

   displayList();

   printf("Enter a value to add to the list: ");

   scanf("%d", &val);

   addValueToList(val);

   displayList();

   return 0;

}

**결과**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Step 1. Doubly linked list 생성

Step 2. 10을 data에 더해주는 함수

Step 3. 출력 함수

Step 4. Main에서 실행

Feedback

// Error check memory allocation 검증!

// Magic number 상수로 정의하자! #define MaxSize 100로 arr[MaxSize]!

1. **CircularLinkedListCountZeros**

#include <stdio.h>

#include <stdlib.h>

// Circular linked list의 노드 구조체 typedef를 이용해보자 structure와 사실 큰 차이없음

typedef struct Node {

    int data;

    struct Node\* next;

} Node;

// Circular linked list의 head 노드를 생성하는 createList 함수

Node\* createList(int n) {

    Node \*head = NULL, \*prev = NULL, \*curr = NULL;

    int i, val;

// 함수가 loop를 통해서 n개의 값을 입력받음

    for (i = 0; i < n; i++) {

        printf("Enter the value of node %d: ", i + 1);

        scanf("%d", &val);

// 받은 data를 curr를 이용해서 새로운 생성한 노드에 할당

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

        curr->data = val;

        curr->next = NULL;

// 첫 번째 노드인 경우 head 노드로 지정

        if (head == NULL) {

            head = curr;

        }

// 이전 노드의 next 포인터를 현재 노드로 설정

        else {

            prev->next = curr;

        }

// 마지막 노드인 경우, next 포인터를 head 노드로 설정하여 Circular linked list 생성

        if (i == n - 1) {

            curr->next = head;

        }

        prev = curr;

    }

    return head;

}

// Circular linked list에서 0이 아닌 값의 개수를 세는 함수

int countNonZero(Node\* head) {

    int count = 0;

    Node\* curr = head;

// head 노드부터 시작하여 Circular linked list를 돌면서 0이 아닌 값을 찾음

    do {

        if (curr->data != 0) {

            count++;

        }

        curr = curr->next;

    } while (curr != head);

    return count;

}

int main() {

    int n;

    Node\* head;

// 노드의 개수 입력받음

    printf("Enter the number of nodes: ");

    scanf("%d", &n);

// Circular linked list 생성 countNonZero 함수에 넣기 위해서 head에 저장

    head = createList(n);

// 0이 아닌 값의 개수 출력

    printf("Number of non-zero values: %d\n", countNonZero(head));

    return 0;

}

**결과**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Step 1. Circular linked list 생성

Step 2. CountNonZeros 함수

Step 3. Main에서 실행

Feedback

// 잘못된 data 입력 시 검증하는 코드!