NAMA : SURYA ADI ARGA WIDANA

NIM : 1203230101

KELAS : IF 03 – 03

TUGAS : OTH CIRCULAR DOUBLE LINKED LIST

**SOURCE CODE :**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

    int data;

    struct Node\* prev;

    struct Node\* next;

} Node;

Node\* createNode(int data) {

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

    newNode->data = data;

    newNode->prev = newNode;

    newNode->next = newNode;

    return newNode;

}

void insertEnd(Node\*\* head, int data) {

    Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

    } else {

        Node\* last = (\*head)->prev;

        newNode->next = \*head;

        (\*head)->prev = newNode;

        newNode->prev = last;

        last->next = newNode;

    }

}

void sortList(Node\*\* head) {

    if (\*head == NULL || (\*head)->next == \*head) return;

    Node\* current = \*head;

    do {

        Node\* nextNode = current->next;

        while (nextNode != \*head) {

            if (current->data > nextNode->data) {

                int temp = current->data;

                current->data = nextNode->data;

                nextNode->data = temp;

            }

            nextNode = nextNode->next;

        }

        current = current->next;

    } while (current->next != \*head);

}

void displayList(Node\* head) {

    if (head == NULL) {

        printf("List kosong.\n");

        return;

    }

    Node\* temp = head;

    do {

        printf("Alamat: %p, Data: %d\n", temp, temp->data);

        temp = temp->next;

    } while (temp != head);

    printf("\n");

}

int main() {

    Node\* head = NULL;

    int N, data;

    printf("Masukkan jumlah data: ");

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {

        printf("Masukkan data ke-%d: ", i+1);

        scanf("%d", &data);

        insertEnd(&head, data);

    }

    printf("List sebelum pengurutan:\n");

    displayList(head);

    sortList(&head);

    printf("List setelah pengurutan:\n");

    displayList(head);

    return 0;

}

**PENJELASAN:**

*Struktur Node:*

typedef struct Node {

    int data;

    struct Node\* prev;

    struct Node\* next;

} Node;

Kode ini mendefinisikan sebuah **struct** bernama **Node** yang mewakili sebuah node pada linked list. Setiap **Node** memiliki tiga atribut:

* **data**: menyimpan nilai integer.
* **prev**: pointer ke node sebelumnya.
* **next**: pointer ke node berikutnya.

*Fungsi createNode:*

Node\* createNode(int data) {

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

    newNode->data = data;

    newNode->prev = newNode;

    newNode->next = newNode;

    return newNode;

}

Fungsi ini membuat node baru. Fungsi ini menerima satu argumen **data**, mengalokasikan memori untuk node baru, menginisialisasi data node, dan mengatur pointer **prev** dan **next** untuk menunjuk ke node itu sendiri (membuat node menjadi node tunggal dalam list melingkar).

*Fungsi insertEnd:*

void insertEnd(Node\*\* head, int data) {

    Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

    } else {

        Node\* last = (\*head)->prev;

        newNode->next = \*head;

        (\*head)->prev = newNode;

        newNode->prev = last;

        last->next = newNode;

    }

}

Fungsi ini menyisipkan node baru di akhir circular doubly linked list:

1. Membuat node baru dengan **createNode**.
2. Jika list kosong (**\*head == NULL**), node baru menjadi head.
3. Jika list tidak kosong, menghubungkan node baru ke akhir list dan mengatur pointer **prev** dan **next** dari node terkait.

*Fungsi sortList:*

void sortList(Node\*\* head) {

    if (\*head == NULL || (\*head)->next == \*head) return;

    Node\* current = \*head;

    do {

        Node\* nextNode = current->next;

        while (nextNode != \*head) {

            if (current->data > nextNode->data) {

                int temp = current->data;

                current->data = nextNode->data;

                nextNode->data = temp;

            }

            nextNode = nextNode->next;

        }

        current = current->next;

    } while (current->next != \*head);

}

Fungsi ini mengurutkan linked list menggunakan bubble sort:

1. Jika list kosong atau hanya berisi satu node, fungsi langsung keluar.
2. Melakukan perulangan ganda untuk membandingkan dan menukar data antar node sampai list terurut.

*Fungsi displayList:*

void displayList(Node\* head) {

    if (head == NULL) {

        printf("List kosong.\n");

        return;

    }

    Node\* temp = head;

    do {

        printf("Alamat: %p, Data: %d\n", temp, temp->data);

        temp = temp->next;

    } while (temp != head);

    printf("\n");

}

Fungsi ini menampilkan data dari setiap node dalam linked list:

1. Jika list kosong, menampilkan pesan "List kosong".
2. Menggunakan loop untuk menampilkan alamat dan data dari setiap node sampai kembali ke head.

*Fungsi main:*

int main() {

    Node\* head = NULL;

    int N, data;

    printf("Masukkan jumlah data: ");

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {

        printf("Masukkan data ke-%d: ", i+1);

        scanf("%d", &data);

        insertEnd(&head, data);

    }

    printf("List sebelum pengurutan:\n");

    displayList(head);

    sortList(&head);

    printf("List setelah pengurutan:\n");

    displayList(head);

    return 0;

}

Fungsi **main** adalah fungsi utama yang:

1. Mendeklarasikan head sebagai **NULL** dan variabel **N** serta **data**.
2. Menerima input jumlah data yang akan dimasukkan.
3. Menggunakan loop untuk menerima input data dan menyisipkannya ke dalam list dengan **insertEnd**.
4. Menampilkan list sebelum pengurutan menggunakan **displayList**.
5. Mengurutkan list menggunakan **sortList**.
6. Menampilkan list setelah pengurutan menggunakan **displayList**.

**HASIL RUNNING PROGRAM:**

**A screenshot of a computer screen

Description automatically generated**

Gambar 1Running-1

A screenshot of a computer screen

Description automatically generated

Gambar 2 Output Running-1

A screenshot of a computer program

Description automatically generated

Gambar 3 Running-2

A screen shot of a computer

Description automatically generated

Gambar 4 Output Running-2