Circular Double Link List

Nama : Muhammad Arifbillah Kamil

NIM : 1203230028

Kelas : IF-03-01

Code :

```c
#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;
}
```

Input 1 :

5

5

3

8

1

6

Hasil :

```
Masukkan jumlah data: 5
Masukkan data ke-1: 5
Masukkan data ke-2: 3
Masukkan data ke-3: 8
Masukkan data ke-4: 1
Masukkan data ke-5: 6
List sebelum pengurutan:
Alamat: 00781678, Data: 5
Alamat: 007804B0, Data: 3
Alamat: 007804C8, Data: 8
Alamat: 007804E0, Data: 1
Alamat: 007804F8, Data: 6

List setelah pengurutan:
Alamat: 00781678, Data: 1
Alamat: 007804B0, Data: 3
Alamat: 007804C8, Data: 5
Alamat: 007804E0, Data: 6
Alamat: 007804F8, Data: 8
```

Input 2 :

3

31

2

123

Hasil :

```
Masukkan jumlah data: 3
Masukkan data ke-1: 31
Masukkan data ke-2: 2
Masukkan data ke-3: 123
List sebelum pengurutan:
Alamat: 00AC1678, Data: 31
Alamat: 00AC04B0, Data: 2
Alamat: 00AC04C8, Data: 123

List setelah pengurutan:
Alamat: 00AC1678, Data: 2
Alamat: 00AC04B0, Data: 31
Alamat: 00AC04C8, Data: 123
```

Penjelasan code :

1. typedef struct Node {...} Node;: Ini mendefinisikan struktur Node yang memiliki tiga field, yaitu data (untuk menyimpan nilai), prev (untuk menunjuk ke node sebelumnya), dan next (untuk menunjuk ke node berikutnya).
2. Node* createNode(int data) {...}: Fungsi ini digunakan untuk membuat node baru. Node baru dibuat dengan menggunakan fungsi malloc dan nilai data diset sesuai dengan argumen yang diberikan. Pointer prev dan next diset menjadi node itu sendiri, karena pada awalnya, node ini adalah satu-satunya node dalam list.
3. void insertEnd(Node** head, int data) {...}: Fungsi ini digunakan untuk menambahkan node baru ke akhir list. Jika list kosong (head == NULL), maka node baru menjadi head. Jika tidak, fungsi ini akan mencari node terakhir dalam list (node yang next-nya menunjuk ke head), dan menambahkan node baru setelah node terakhir tersebut.
4. void sortList(Node** head) {...}: Fungsi ini digunakan untuk mengurutkan node-node dalam list secara ascending. Fungsi ini menggunakan algoritma bubble sort, di mana ia

membandingkan setiap pasangan node yang berdekatan dan menukar posisi mereka jika mereka tidak dalam urutan yang benar.

5. void displayList(Node* head) {...}: Fungsi ini digunakan untuk mencetak list dari awal (dari head) hingga akhir. Ini dilakukan dengan bergerak maju dari head, mencetak data dan alamat memori di setiap node, hingga kembali ke head lagi.
6. int main() {...}: Fungsi utama ini membuat list kosong dan menambahkan beberapa node ke dalamnya menggunakan fungsi insertEnd. Kemudian, ia mengurutkan node-node dalam list menggunakan fungsi sortList, dan mencetak isi list sebelum dan setelah pengurutan.