

Nama : Hafizhah Nur Zahira  
NIM : 1203230024  
Kelas : IF 03-02

## OTH Circular Double Linked List

### ❖ Source Code

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int data;
    struct Node *next;
    struct Node *prev;
} Node;

Node *head = NULL;
Node *tail = NULL;

Node *createNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

void insertNode(int data) {
    Node *newNode = createNode(data);

    if (head == NULL) {
        head = newNode;
        tail = newNode;
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = head;
        head->prev = newNode;
        tail = newNode;
    }
}

void printList() {
    Node *curr = head;
    if (curr == NULL) {
        printf("List is empty\n");
    }
}
```

```

        return;
    }

    do {
        printf("Address: %p, Data: %d\n", (unsigned long)curr, curr->data);
        curr = curr->next;
    } while (curr != head);
}

void swapNodes(Node *a, Node *b) {
    if (a->next == b) {
        a->next = b->next;
        b->prev = a->prev;
        a->prev->next = b;
        b->next->prev = a;
        b->next = a;
        a->prev = b;
    } else {
        Node *tempNext = a->next;
        Node *tempPrev = a->prev;
        a->next = b->next;
        a->prev = b->prev;
        b->next = tempNext;
        b->prev = tempPrev;
        a->next->prev = a;
        a->prev->next = a;
        b->next->prev = b;
        b->prev->next = b;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}

void sortList() {
    if (head == NULL) return;

    int swapped;
    Node *curr;

```

```

do {
    swapped = 0;
    curr = head;

    do {
        Node *nextNode = curr->next;
        if (curr->data > nextNode->data) {
            swapNodes(curr, nextNode);
            swapped = 1;
        } else {
            curr = nextNode;
        }
    } while (curr != tail);
} while (swapped);
}

int main() {
    int n;
    printf("Masukkan jumlah data: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int data;
        printf("Masukkan data ke-%d: ", i + 1);
        scanf("%d", &data);
        insertNode(data);
    }

    printf("\nList sebelum pengurutan:\n");
    printList();

    sortList();

    printf("\nList setelah pengurutan:\n");
    printList();

    return 0;
}

```

#### ❖ Penjelasan

```

#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int data;
    struct Node *next;
}

```

```
    struct Node *prev;  
} Node;
```

Pada 'typedef struct Node' mendefinisikan struktur 'Node' yang memiliki tiga elemen yaitu 'data' untuk menyimpan nilai data, 'next' untuk menunjuk ke node berikutnya, dan 'prev' untuk menunjuk ke node sebelumnya.

```
Node *head = NULL;  
Node *tail = NULL;
```

Node \*head dan Node \*tail merupakan pointer yang menunjuk ke objek bertipe 'node'. '\*head' = NULL dan '\*tail' = NULL berarti pointer 'head' dan 'tail' diinisialisasikan tidak menunjuk ke node manapun.

```
Node *createNode(int data) {  
    Node *newNode = (Node *)malloc(sizeof(Node));  
    newNode->data = data;  
    newNode->next = NULL;  
    newNode->prev = NULL;  
  
    return newNode;  
}
```

Fungsi ini membuat node baru dengan nilai 'data' yang diberikan dan mengalokasi memori untuk node baru serta menginisialisasikan pointer 'next' dan 'prev' sama dengan NULL.

```
void insertNode(int data) {  
    Node *newNode = createNode(data);  
  
    if (head == NULL) {  
        head = newNode;  
        tail = newNode;  
        newNode->next = newNode;  
        newNode->prev = newNode;  
    } else {  
        tail->next = newNode;  
        newNode->prev = tail;  
        newNode->next = head;  
        head->prev = newNode;  
        tail = newNode;  
    }  
}
```

Fungsi ini menyisipkan node baru ke dalam daftar. Jika daftar kosong ('head' = NULL) maka node baru menjadi 'head' dan 'tail'. Tetapi jika tidak, node baru ditambahkan setelah 'tail' dan dihubungkan kembali ke 'head'.

```

void printList() {
    Node *curr = head;
    if (curr == NULL) {
        printf("List is empty\n");
        return;
    }

    do {
        printf("Address: %p, Data: %d\n", (unsigned long)curr, curr->data);
        curr = curr->next;
    } while (curr != head);
}

```

Fungsi ini digunakan untuk mencetak setiap elemennya, tetapi jika daftar kosong maka akan menampilkan bahwa "List is empty".

```

void swapNodes(Node *a, Node *b) {
    if (a->next == b) {
        a->next = b->next;
        b->prev = a->prev;
        a->prev->next = b;
        b->next->prev = a;
        b->next = a;
        a->prev = b;
    } else {
        Node *tempNext = a->next;
        Node *tempPrev = a->prev;
        a->next = b->next;
        a->prev = b->prev;
        b->next = tempNext;
        b->prev = tempPrev;
        a->next->prev = a;
        a->prev->next = a;
        b->next->prev = b;
        b->prev->next = b;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}

```

```
}
```

Fungsi ini menukar posisi dua node 'a' dan 'b' dalam daftar. Penukaran dilakukan dengan memperbarui pointer 'next' dan 'prev' dari node dan node sekitarnya. Jika salah satu node adalah 'head' atau 'tail' maka pointer 'head' atau 'tail' akan diperbarui.

```
void sortList() {
    if (head == NULL) return;

    int swapped;
    Node *curr;

    do {
        swapped = 0;
        curr = head;

        do {
            Node *nextNode = curr->next;
            if (curr->data > nextNode->data) {
                swapNodes(curr, nextNode);
                swapped = 1;
            } else {
                curr = nextNode;
            }
        } while (curr != tail);
    } while (swapped);
}
```

Fungsi ini digunakan untuk mengurutkan daftar menggunakan metode bubble sort. Bubble sort akan dilakukan jika nilai 'data' dari node saat ini lebih besar daripada node berikutnya dan proses akan terus diulangi hingga tidak ada lagi node yang perlu ditukar (swapped == 0).

```
int main() {
    int n;
    printf("Masukkan jumlah data: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int data;
        printf("Masukkan data ke-%d: ", i + 1);
        scanf("%d", &data);
        insertNode(data);
    }

    printf("\nList sebelum pengurutan:\n");
    printList();
}
```

```

    sortList();

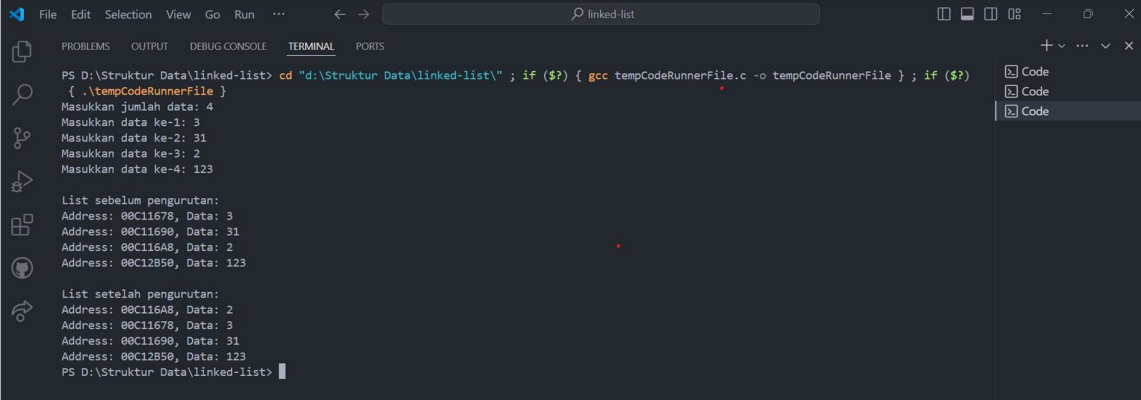
    printf("\nList setelah pengurutan:\n");
    printList();

    return 0;
}

```

Fungsi ini merupakan fungsi utama dari program ini. Pada fungsi ini program meminta jumlah data yang akan dimasukkan (n). Kemudian 'n' akan dimasukkan kedalam daftar dengan memanggil fungsi 'insertNode'. Selanjutnya program akan mencetak daftar sebelum dilakukan pengurutan. Program melakukan pengurutan dengan memanggil fungsi 'sortList'. Lalu program akan mencetak kembali daftar yang sudah diurutkan.

### ❖ Output



```

PS D:\Struktur Data\linked-list> cd "d:\Struktur Data\linked-list\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) {
{ .\tempCodeRunnerFile }
Masukkan jumlah data: 4
Masukkan data ke-1: 3
Masukkan data ke-2: 31
Masukkan data ke-3: 2
Masukkan data ke-4: 123

List sebelum pengurutan:
Address: 00C11678, Data: 3
Address: 00C11690, Data: 31
Address: 00C116A8, Data: 2
Address: 00C12850, Data: 123

List setelah pengurutan:
Address: 00C116A8, Data: 2
Address: 00C11678, Data: 3
Address: 00C11690, Data: 31
Address: 00C12850, Data: 123
PS D:\Struktur Data\linked-list>

```