

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.

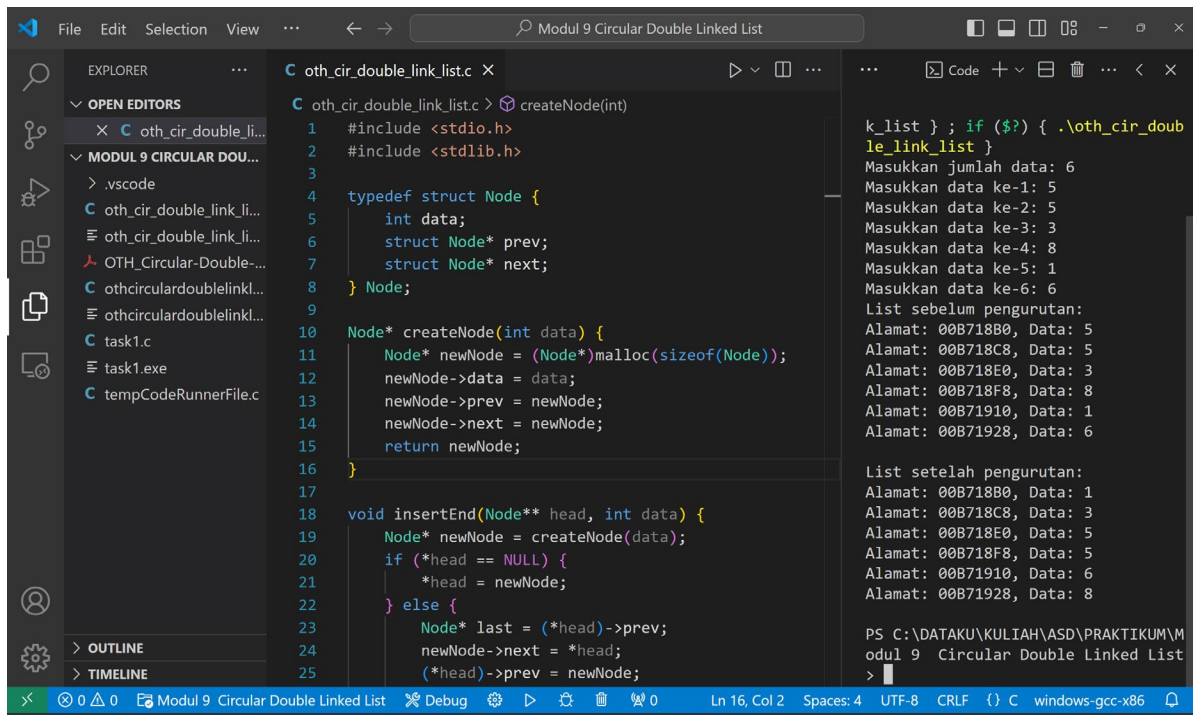
Fungsi main:

```
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:



The screenshot shows the Visual Studio Code interface with the file `oth_cir_double_link_list.c` open. The code defines a `Node` structure with `data`, `prev`, and `next` pointers. It includes functions `createNode` and `insertEnd`. The terminal on the right shows the program's execution output.

```
#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 = NULL;
    newNode->next = NULL;
    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;
    }
}

int main() {
    Node* head = NULL;
    int jumlah;
    printf("Masukkan jumlah data: ");
    scanf("%d", &jumlah);
    for (int i = 1; i <= jumlah; i++) {
        int data;
        printf("Masukkan data ke-%d: ", i);
        scanf("%d", &data);
        insertEnd(&head, data);
    }
    printf("List sebelum pengurutan:\n");
    Node* temp = head;
    while (temp != NULL) {
        printf("Alamat: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    }
    printf("List setelah pengurutan:\n");
    temp = head;
    while (temp != NULL) {
        printf("Alamat: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    }
    return 0;
}
```

Output from the terminal:

```
Masukkan jumlah data: 6
Masukkan data ke-1: 5
Masukkan data ke-2: 5
Masukkan data ke-3: 3
Masukkan data ke-4: 8
Masukkan data ke-5: 1
Masukkan data ke-6: 6
List sebelum pengurutan:
Alamat: 00B718B0, Data: 5
Alamat: 00B718C8, Data: 5
Alamat: 00B718E0, Data: 3
Alamat: 00B718F8, Data: 8
Alamat: 00B71910, Data: 1
Alamat: 00B71928, Data: 6
List setelah pengurutan:
Alamat: 00B718B0, Data: 1
Alamat: 00B718C8, Data: 3
Alamat: 00B718E0, Data: 5
Alamat: 00B718F8, Data: 5
Alamat: 00B71910, Data: 6
Alamat: 00B71928, Data: 8
```

Gambar 1Running-1

```
Masukkan jumlah data: 6
Masukkan data ke-1: 5
Masukkan data ke-2: 5
Masukkan data ke-3: 3
Masukkan data ke-4: 8
Masukkan data ke-5: 1
Masukkan data ke-6: 6
List sebelum pengurutan:
Alamat: 00B718B0, Data: 5
Alamat: 00B718C8, Data: 5
Alamat: 00B718E0, Data: 3
Alamat: 00B718F8, Data: 8
Alamat: 00B71910, Data: 1
Alamat: 00B71928, Data: 6
List setelah pengurutan:
Alamat: 00B718B0, Data: 1
Alamat: 00B718C8, Data: 3
Alamat: 00B718E0, Data: 5
Alamat: 00B718F8, Data: 5
Alamat: 00B71910, Data: 6
Alamat: 00B71928, Data: 8
```

Gambar 2 Output Running-1

```

oth_cir_double_link_list.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct Node {
5      int data;
6      struct Node* prev;
7      struct Node* next;
8  } Node;
9
10 Node* createNode(int data) {
11     Node* newNode = (Node*)malloc(sizeof(Node));
12     newNode->data = data;
13     newNode->prev = newNode;
14     newNode->next = newNode;
15     return newNode;
16 }
17
18 void insertEnd(Node** head, int data) {
19     Node* newNode = createNode(data);
20     if (*head == NULL) {
21         *head = newNode;
22     } else {
23         Node* last = (*head)->prev;
24         newNode->next = *head;
25         (*head)->prev = newNode;
26     }
27 }
28
29 int main() {
30     Node* head = NULL;
31     int n;
32     printf("Masukkan jumlah data: ");
33     scanf("%d", &n);
34     for (int i = 0; i < n; i++) {
35         int data;
36         printf("Masukkan data ke-%d: ", i+1);
37         scanf("%d", &data);
38         insertEnd(&head, data);
39     }
40     printf("List sebelum pengurutan:\n");
41     traverse(head);
42     printf("List setelah pengurutan:\n");
43     traverse(head);
44     return 0;
45 }
46
47 void traverse(Node* head) {
48     if (head == NULL) return;
49     Node* current = head;
50     do {
51         printf("Alamat: %08BE18B0, Data: %d\n", current->data);
52         current = current->next;
53     } while (current != head);
54 }

```

PS C:\DATAKU\KULIAH\ASD\PRAKTIKUM\M odul 9 Circular Double Linked List
> cd "c:\DATAKU\KULIAH\ASD\PRAKTIKU M\Modul 9 Circular Double Linked L ist\" ; if (\$?) { gcc oth_cir_doubl e_link_list.c -o oth_cir_double_lin k_list } ; if (\$?) { .\oth_cir_doub le_link_list }
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:
Alamat: 00BE18B0, Data: 3
Alamat: 00BE18C8, Data: 31
Alamat: 00BE18E0, Data: 2
Alamat: 00BE18F8, Data: 123
List setelah pengurutan:
Alamat: 00BE18B0, Data: 2
Alamat: 00BE18C8, Data: 3
Alamat: 00BE18E0, Data: 31
Alamat: 00BE18F8, Data: 123
PS C:\DATAKU\KULIAH\ASD\PRAKTIKUM\M odul 9 Circular Double Linked List
>

Gambar 3 Running-2

```

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:
Alamat: 00BE18B0, Data: 3
Alamat: 00BE18C8, Data: 31
Alamat: 00BE18E0, Data: 2
Alamat: 00BE18F8, Data: 123

List setelah pengurutan:
Alamat: 00BE18B0, Data: 2
Alamat: 00BE18C8, Data: 3
Alamat: 00BE18E0, Data: 31
Alamat: 00BE18F8, Data: 123

```

Gambar 4 Output Running-2