

Nama : Nugraha Suryapratama

NIM : 1203230113

Kelas : IF 03-03

Tugas OTH ASD Week 13

SOURCE CODE

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

struct DataCilcular {
    struct DataCilcular *prev;
    int data;
    struct DataCilcular *next;
};

typedef struct DataCilcular *node;

node createNode(int data) {
    node new_node = (node)malloc(sizeof(struct DataCilcular));
    new_node->data = data;
    new_node->next = new_node->prev = new_node;
    return new_node;
}

node insertLast(node head, node new_node) {
    if (head == NULL) {
        head = new_node;
    } else {
        node last = head->prev;
        new_node->next = head;
        new_node->prev = last;
        head->prev = new_node;
        last->next = new_node;
    }
    return head;
}

node sortingAscending(node head, node new_node) {
    if (head == NULL) {
        head = new_node;
    } else if (head->data >= new_node->data) {
```

```

        node last = head->prev;
        new_node->next = head;
        new_node->prev = last;
        last->next = new_node;
        head->prev = new_node;
        head = new_node;
    } else {
        node curr = head;
        while (curr->next != head && curr->next->data < new_node->data) {
            curr = curr->next;
        }
        new_node->next = curr->next;
        new_node->prev = curr;
        curr->next->prev = new_node;
        curr->next = new_node;
    }
    return head;
}

node ascendingData(node head) {
    node new_head = NULL;
    node curr = head, next;
    do {
        next = curr->next;
        curr->next = curr->prev = curr;
        new_head = sortingAscending(new_head, curr);
        curr = next;
    } while (curr != head);
    return new_head;
}

void viewData(node head) {
    if (head == NULL) {
        printf("Daftar kosong\n");
        return;
    }
    node curr = head;
    do {
        printf("(%p, %d)\n", curr, curr->data);
        curr = curr->next;
    } while (curr != head);
    printf("\n");
}

int main() {
    node head = NULL;
    int N;

```

```

printf("Masukkan jumlah elemen: ");
scanf("%d", &N);

if (N < 1 || N > 10) {
    printf("Jumlah elemen tidak valid. Harus antara 1 dan 10.\n");
    return 1;
}

for (int i = 0; i < N; i++) {
    int data;
    printf("Masukkan data %d: ", i + 1);
    scanf("%d", &data);
    if (data < 1 || data > 10) {
        printf("Data tidak valid. Harus antara 1 dan 10.\n");
        return 1;
    }
    head = insertLast(head, createNode(data));
}

printf("Daftar sebelum pengurutan:\n");
viewData(head);

head = ascendingData(head);

printf("Daftar setelah pengurutan:\n");
viewData(head);

return 0;
}

```

PENJELASAN CODE

- Struktur Data

1. Struct DataCircular:

```

struct DataCircular {
    struct DataCircular *prev;
    int data;
    struct DataCircular *next;
};

```

Struktur ini mendefinisikan node dari linked list dengan pointer prev yang menunjuk ke node sebelumnya, data yang menyimpan nilai data, dan pointer next yang menunjuk ke node berikutnya.

2. Alias node:

```
typedef struct DataCircular *node;
```

node adalah alias untuk pointer ke struct **DataCircular**, sehingga lebih mudah digunakan dalam fungsi-fungsi berikutnya.

- Fungsi

1. createNode:

```
node createNode(int data) {  
    node new_node = (node)malloc(sizeof(struct DataCircular));  
    new_node->data = data;  
    new_node->next = new_node->prev = new_node;  
    return new_node;  
}
```

Fungsi ini membuat node baru dengan data yang diberikan dan menginisialisasi next dan prev menunjuk ke node itu sendiri (karena ini adalah circular linked list).

2. insertLast

```
node insertLast(node head, node new_node) {  
    if (head == NULL) {  
        head = new_node;  
    } else {  
        node last = head->prev;  
        new_node->next = head;  
        new_node->prev = last;  
        head->prev = new_node;  
        last->next = new_node;  
    }  
    return head;  
}
```

Fungsi ini mengurutkan linked list secara ascending dengan menyisipkan node baru pada posisi yang tepat. Jika head adalah NULL, node baru menjadi head. Jika data node baru lebih kecil atau sama dengan data head, node baru menjadi head. Jika tidak, node baru disisipkan pada posisi yang tepat dalam list.

3. sortingAscending

```
node sortingAscending(node head, node new_node) {  
    if (head == NULL) {  
        head = new_node;  
    } else if (head->data >= new_node->data) {  
        node last = head->prev;  
        new_node->next = head;  
        new_node->prev = last;  
        last->next = new_node;  
    }  
}
```

```

        head->prev = new_node;
        head = new_node;
    } else {
        node curr = head;
        while (curr->next != head && curr->next->data < new_node->data) {
            curr = curr->next;
        }
        new_node->next = curr->next;
        new_node->prev = curr;
        curr->next->prev = new_node;
        curr->next = new_node;
    }
    return head;
}

```

Fungsi ini mengurutkan linked list secara ascending dengan menyisipkan node baru pada posisi yang tepat. Jika head adalah NULL, node baru menjadi head. Jika data node baru lebih kecil atau sama dengan data head, node baru menjadi head. Jika tidak, node baru disisipkan pada posisi yang tepat dalam list.

4. ascendingData

```

node ascendingData(node head) {
    node new_head = NULL;
    node curr = head, next;
    do {
        next = curr->next;
        curr->next = curr->prev = curr;
        new_head = sortingAscending(new_head, curr);
        curr = next;
    } while (curr != head);
    return new_head;
}

```

Fungsi ini mengurutkan seluruh linked list dengan membuat linked list baru yang sudah terurut. Setiap node di list asli dipindahkan ke linked list baru dengan memanggil sortingAscending.

5. viewData

```

void viewData(node head) {
    if (head == NULL) {
        printf("Daftar kosong\n");
        return;
    }
    node curr = head;
    do {
        printf("(%p, %d)\n", curr, curr->data);
        curr = curr->next;
    } while (curr != head);
    printf("\n");
}

```

```
}
```

Fungsi ini mencetak data dari semua node dalam linked list, mulai dari head hingga kembali ke head.

- Fungsi main

1. Inisialisasi:

```
node head = NULL;
int N;

printf("Masukkan jumlah elemen: ");
scanf("%d", &N);

if (N < 1 || N > 10) {
    printf("Jumlah elemen tidak valid. Harus antara 1 dan 10.\n");
    return 1;
}
```

Bagian ini meminta jumlah elemen yang akan dimasukkan ke dalam linked list. Jumlah harus antara 1 dan 10.

2. Memasukkan Data:

```
for (int i = 0; i < N; i++) {
    int data;
    printf("Masukkan data %d: ", i + 1);
    scanf("%d", &data);
    if (data < 1 || data > 10) {
        printf("Data tidak valid. Harus antara 1 dan 10.\n");
        return 1;
    }
    head = insertLast(head, createNode(data));
}
```

Bagian ini memasukkan data ke dalam linked list dengan memanggil **insertLast**.

3. Menampilkan dan Mengurutkan Data:

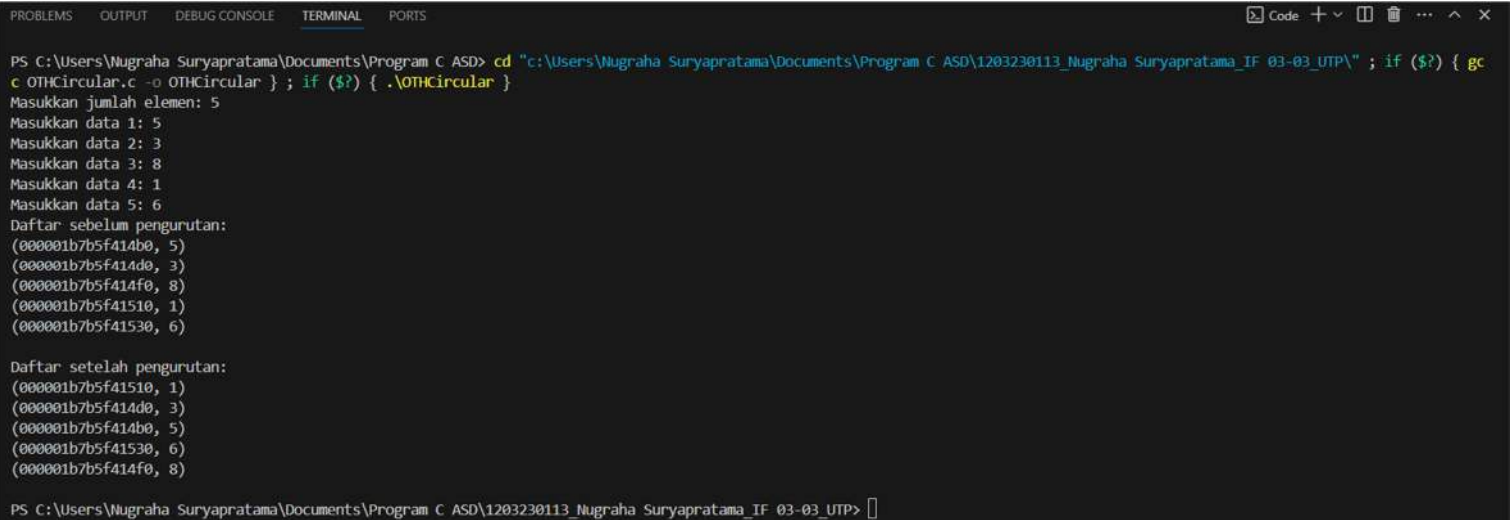
```
printf("Daftar sebelum pengurutan:\n");
viewData(head);

head = ascendingData(head);

printf("Daftar setelah pengurutan:\n");
viewData(head);
```

Bagian ini menampilkan linked list sebelum dan sesudah pengurutan dengan memanggil **viewData** dan **ascendingData**.

SCREENSHOT OUTPUT



```
PS C:\Users\Nugraha Suryapratama\Documents\Program C ASD> cd "c:\Users\Nugraha Suryapratama\Documents\Program C ASD\1203230113_Nugraha Suryapratama_IF 03-03_UTP\" ; if ($?) { gcc OTHCircular.c -o OTHCircular } ; if ($?) { .\OTHCircular }
Masukkan jumlah elemen: 5
Masukkan data 1: 5
Masukkan data 2: 3
Masukkan data 3: 8
Masukkan data 4: 1
Masukkan data 5: 6
Daftar sebelum pengurutan:
(000001b7b5f414b0, 5)
(000001b7b5f414d0, 3)
(000001b7b5f414f0, 8)
(000001b7b5f41510, 1)
(000001b7b5f41530, 6)

Daftar setelah pengurutan:
(000001b7b5f41510, 1)
(000001b7b5f414d0, 3)
(000001b7b5f414b0, 5)
(000001b7b5f41530, 6)
(000001b7b5f414f0, 8)

PS C:\Users\Nugraha Suryapratama\Documents\Program C ASD\1203230113_Nugraha Suryapratama_IF 03-03_UTP> 
```