

CIRCULAR DOUBLE LINKED-LIST

Nama : I Gede Fender Adrea Veda

NIM : 1203230009

Kelas : IF 03-01

Soal : OTH

Circular Double Linked List

Deskripsi Tugas

Andi baru saja belajar tentang struktur data sirkular double linked list. Dia ingin melatih kemampuannya dengan membuat sebuah program untuk mengelola data dalam struktur tersebut.

Andi ingin membuat sebuah program C yang akan meminta pengguna untuk memasukkan sejumlah data ke dalam sirkular double linked list. Setelah itu, program akan mengurutkan list tersebut secara ascending dengan syarat "Never change the data. Change the position of the nodes." Artinya, Anda dilarang mengubah/memanipulasi data pada setiap node, namun Anda dapat mengubah posisi dari node-node tersebut.

Format Masukan

- Baris pertama berisi sebuah bilangan bulat N ($1 \leq N \leq 10$), menyatakan jumlah data yang akan dimasukkan ke dalam list.
- N baris berikutnya berisi sebuah bilangan bulat A_i ($1 \leq A_i \leq 10$), menyatakan data yang dimasukkan ke dalam list.

Format Keluaran

- Tampilkan list (memory address & data) sebelum pengurutan.
- Tampilkan list (memory address & data) setelah pengurutan.

1. Source Code

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

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

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

void addNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
```

```

    if (*head == NULL) {
        *head = newNode;
        (*head)->next = *head;
        (*head)->prev = *head;
    } else {
        struct Node* last = (*head)->prev;
        last->next = newNode;
        newNode->prev = last;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}

void displayList(struct Node* head) {
    struct Node* current = head;
    do {
        printf("Address: %p, Data: %d\n", current, current->data);
        current = current->next;
    } while (current != head);
}

void sortList(struct Node* head) {
}

int main() {
    struct Node* head = NULL;
    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);
        addNode(&head, data);
    }

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

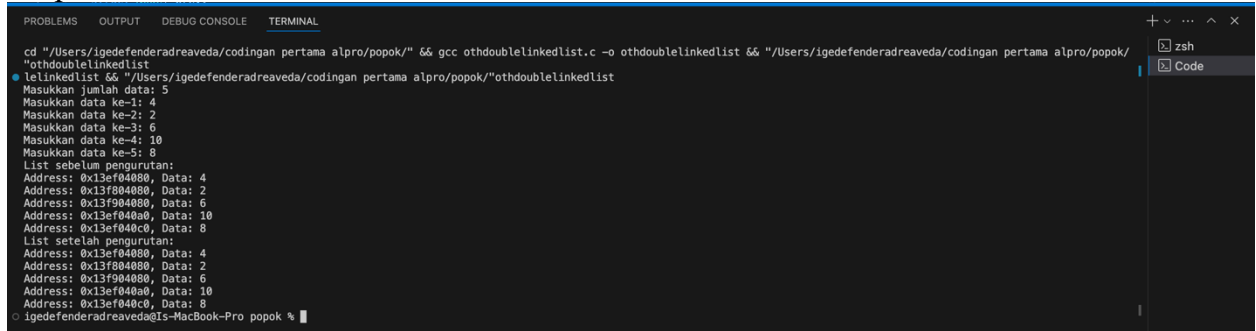
    sortList(head);

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

    return 0;
}

```

2. Output



```
cd "/Users/igedfenderadreaaveda/codingan pertama alpro/popok/" && gcc othdoublelinkedlist.c -o othdoublelinkedlist && "/Users/igedfenderadreaaveda/codingan pertama alpro/popok/" othdoublelinkedlist
ielinkedlist && "/Users/igedfenderadreaaveda/codingan pertama alpro/popok/" othdoublelinkedlist
Masukkan jumlah data: 5
Masukkan data ke-1: 4
Masukkan data ke-2: 2
Masukkan data ke-3: 6
Masukkan data ke-4: 10
Masukkan data ke-5: 8
List sebelum pengurutan:
Address: 0x13ef04080, Data: 4
Address: 0x13f804080, Data: 2
Address: 0x13f904080, Data: 6
Address: 0x13ef040a0, Data: 10
Address: 0x13ef040c0, Data: 8
List setelah pengurutan:
Address: 0x13ef04080, Data: 4
Address: 0x13f804080, Data: 2
Address: 0x13f904080, Data: 6
Address: 0x13ef040a0, Data: 10
Address: 0x13ef040c0, Data: 8
igedfenderadreaaveda@Is-MacBook-Pro popok %
```

3. Penjelasan

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

int data : variabel tipe int guna menyimpan data

struct Node* next : pointer yang menunjuk ke node selanjutnya dalam linked list. Jika node terakhir dalam list, maka next akan menunjuk ke node pertama.

struct Node* prev : pointer yang menunjuk ke node sebelumnya dalam linked list. Jika node pertama dalam list, maka prev akan menunjuk ke node terakhir.

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

Terdapat fungsi **createNode** guna membuat node baru dalam linked-list.

Fungsi **malloc** untuk menempatkan memori, dan **sizeof(struct Node)** menghitung berapa banyak memori yang diperlukan untuk struct Node. Hasilnya disimpan dalam pointer **newNode**.

newNode->data = data : menetapkan nilai dari parameter data yang diterima oleh fungsi ke dalam anggota data dari struct Node yang baru dibuat.

newNode->next = NULL dan **newNode->prev = NULL** menetapkan anggota next dan prev dari struct Node ke NULL, yang berarti node ini tidak menunjuk ke node lain di depan dan belakangnya.

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

```

    struct Node* last = (*head)->prev;
    last->next = newNode;
    newNode->prev = last;
    newNode->next = *head;
    (*head)->prev = newNode;
}
}

```

Terdapat fungsi **addNode** guna menambahkan node baru ke dalam circular double linked – list.

Pemanggilan fungsi **createNode** guna menambahkan node baru sesuai data yang diberikan.
if (*head == NULL) { ... } : apabila list masih kosong head ditunjuk ke node baru, kemudian next dan prev dari node baru untuk menunjuk ke node itu sendiri,
else { ... } : apabila list tidak kosong mencari node terakhir dalam list dengan mengikuti anggota prev dari head, anggota next dan prev dari node terakhir untuk menunjuk ke node baru, dan anggota prev dari head untuk menunjuk ke node baru, karena node baru sekarang menjadi node terakhir.

```

void displayList(struct Node* head) {
    struct Node* current = head;
    do {
        printf("Address: %p, Data: %d\n", current, current->data);
        current = current->next;
    } while (current != head);
}

```

Terdapat fungsi **displayList** guna menampilkan isi circular dalam linked-list.

struct Node* current = head : pointer current yang menunjuk ke node pertama dalam list.

do { ... } while (current != head) : perulangan selama current tidak sama dengan head.

printf("Address: %p, Data: %d\n", current, current->data) : mencetak alamat memori dan data yang ditunjuk oleh current.

current = current->next : fungsi memindahkan current ke node berikutnya dalam list.

```

void sortList(struct Node* head)

```

sortList mengurutkan list tanpa mengubah data yang disimpan dalam node. Artinya, Andi tidak diperkenankan untuk mengubah nilai data pada setiap node, tetapi dia dapat mengubah posisi dari node tersebut dalam list untuk mengurutkannya.

```

int main() {
    struct Node* head = NULL;
    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);
        addNode(&head, data);
    }
}

```

```
}
```

struct Node* head = NULL : menunjukan list awalnya kosong.

int N : menyimpan jumlah data yang akan dimasukkan.

printf("Masukkan jumlah data : mencetak pesan ke layar.

scanf("%d", &N) : membaca inputan dan disimpan dalam variabel N.

for (int i = 0; i < N; i++) { ... } : perulangan yang berjalan sebanyak N (jumlah data yang dimasukkan) kali.

int data : deklarasi variabel data yang akan dimasukkan ke dalam perulangan.

printf("Masukkan data ke-%d: ", i + 1) : mencetak pesan ke layar.

scanf("%d", &data) : membaca inputan dan disimpan ke variabel data.

addNode(&head, data) : memanggil fungsi addNode.

```
printf("List sebelum pengurutan:\n");  
displayList(head);
```

Penampilan isi dari sebuah sirkular double linked list sebelum melakukan pengurutan dan displayList akan mencetak alamat memori dan data dari setiap node dalam list.

```
sortList(head);
```

Mengurutkan list secara ascending tanpa mengubah atau memanipulasi data yang disimpan dalam node.

```
printf("List setelah pengurutan:\n");  
displayList(head);
```

Penampilan isi dari sebuah sirkular double linked list setelah melakukan pengurutan dan displayList akan mencetak alamat memori dan data dari setiap node dalam list.

```
return 0;
```

Program selesai dijalankan.