

LAPORAN TUGAS PRAKTIKUM

OTH CIRCULAR DOUBLE

LINKED LIST

NAMA LENGKAP :MICHAEL VALENTINO SOGEN
KELAS :IF 03-02
NIM :1203230099
MATKUL :ALGORITMA STURUKTUR DATA (ASD)

Source Code

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

// Definisi struktur node untuk Circular Doubly Linked List
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode->prev = newNode;
    return newNode;
}

// Fungsi untuk menambahkan node ke akhir list
void append(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 untuk mencetak list (memory address & data)
void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("%p: %d\n", temp, temp->data);
        temp = temp->next;
    } while (temp != head);
}

// Fungsi untuk mengurutkan node pada Circular Doubly Linked List
void sortList(Node** head) {
    if (*head == NULL || (*head)->next == *head) return;
    Node *curr, *index;
    int temp;
    for (curr = *head; curr->next != *head; curr = curr->next) {
        for (index = curr->next; index != *head; index = index->next) {
            if (curr->data > index->data) {
                // Swap data antara curr dan index
                temp = curr->data;
                curr->data = index->data;
                index->data = temp;
            }
        }
    }
}

int main() {
    int N, data;
    Node* head = NULL;

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

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

```

```

}

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

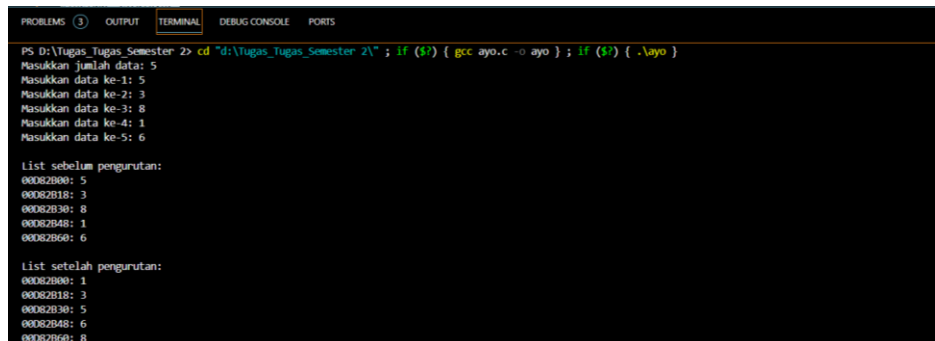
sortList(&head);

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

return 0;
}

```

Output Pertama



```

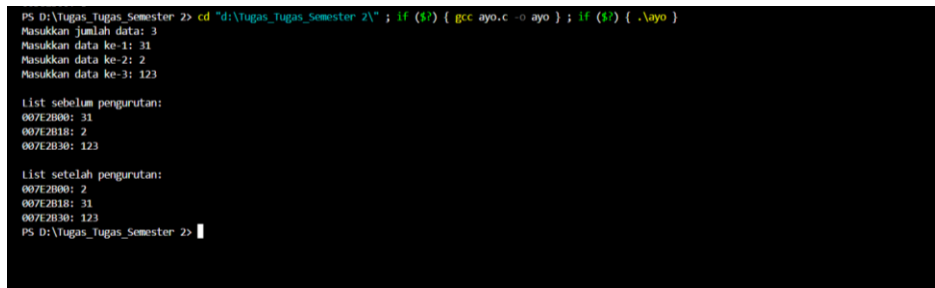
PROBLEMS ③ OUTPUT TERMINAL DEBUG CONSOLE PORTS
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc ayo.c -o ayo } ; if ($?) { .\ayo }
Masukkan jumlah data: 5
Masukkan data ke-1: 5
Masukkan data ke-2: 3
Masukkan data ke-3: 8
Masukkan data ke-4: 1
Masukkan data ke-5: 6

List sebelum pengurutan:
00082800: 5
00082818: 3
00082830: 8
00082848: 1
00082860: 6

List setelah pengurutan:
00082800: 1
00082818: 3
00082830: 5
00082848: 6
00082860: 8

```

Output Kedua



```

PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc ayo.c -o ayo } ; if ($?) { .\ayo }
Masukkan jumlah data: 3
Masukkan data ke-1: 31
Masukkan data ke-2: 2
Masukkan data ke-3: 123

List sebelum pengurutan:
007E2800: 31
007E2818: 2
007E2830: 123

List setelah pengurutan:
007E2800: 2
007E2818: 31
007E2830: 123
PS D:\Tugas_Tugas_Semester 2>

```

Penjelasan Program

Struktur **Node** didefinisikan dengan tiga anggota: **data** untuk menyimpan nilai, **next** untuk menunjuk ke node berikutnya, dan **prev** untuk menunjuk ke node sebelumnya. Fungsi **createNode** bertugas membuat node baru dengan nilai data yang diberikan. Node baru ini memiliki pointer **next** dan **prev** yang menunjuk ke dirinya sendiri, menandakan bahwa ia adalah node tunggal dalam list. Fungsi **append** menambahkan node baru ke akhir list. Jika list masih kosong, node baru menjadi head. Jika tidak, node baru dihubungkan ke node terakhir dan node pertama (head), memastikan bahwa CDLL tetap terhubung dengan benar di kedua arah. Fungsi **printList** mencetak semua elemen dalam list beserta alamat memorinya. Fungsi ini menggunakan loop **do-while** untuk menjelajahi setiap node, mulai dari head dan kembali ke head setelah mencetak semua node. Fungsi **sortList** mengurutkan elemen-elemen dalam list menggunakan metode Bubble Sort. Dua loop bersarang digunakan untuk membandingkan dan menukar nilai **data** antar node jika diperlukan. Fungsi ini memastikan bahwa setelah diurutkan, nilai-nilai data dalam list berada dalam urutan yang benar. Dalam fungsi **main**, pengguna diminta untuk memasukkan jumlah elemen yang akan ditambahkan ke list. Setiap elemen dibaca dan ditambahkan ke list menggunakan **append**. List kemudian dicetak sebelum dan sesudah diurutkan menggunakan **printList** dan **sortList**.