Nama : Maria Rosa Wahyuning Utami

NIM : 1203230123

Kelas : IF 03-03

OTH Circular Double Linked List

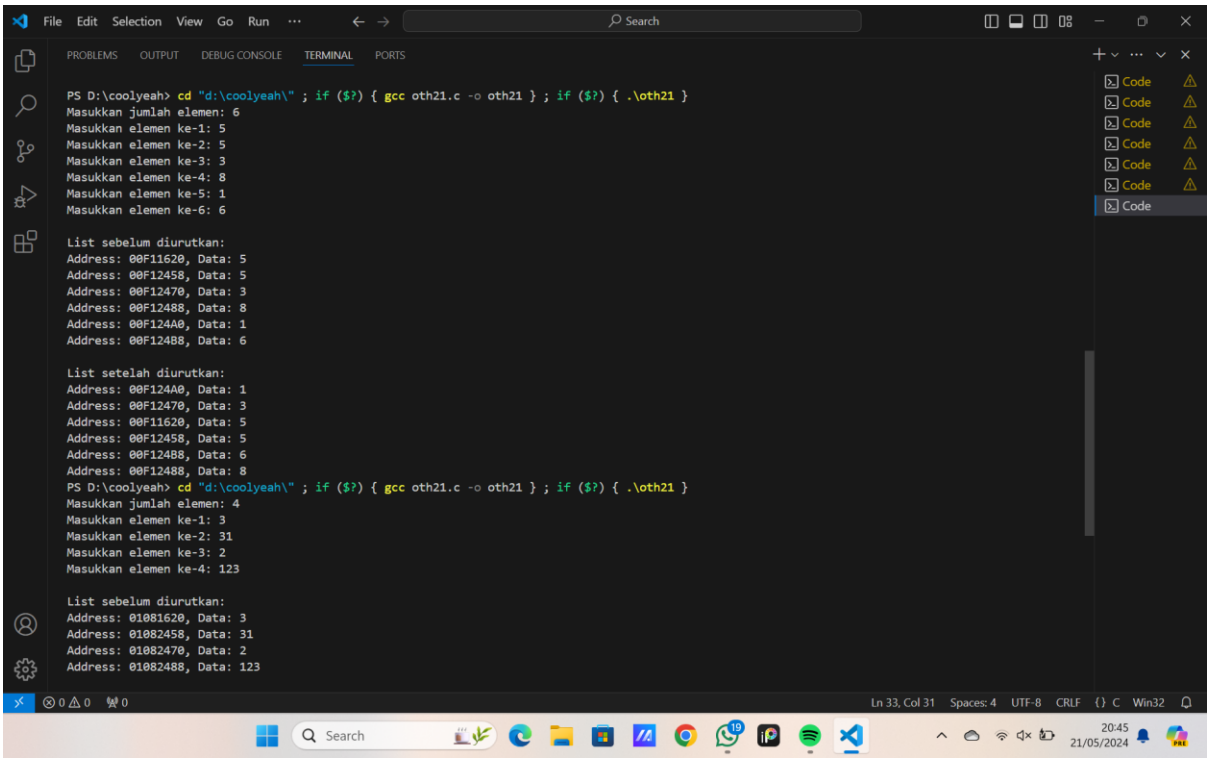
1. Source Code

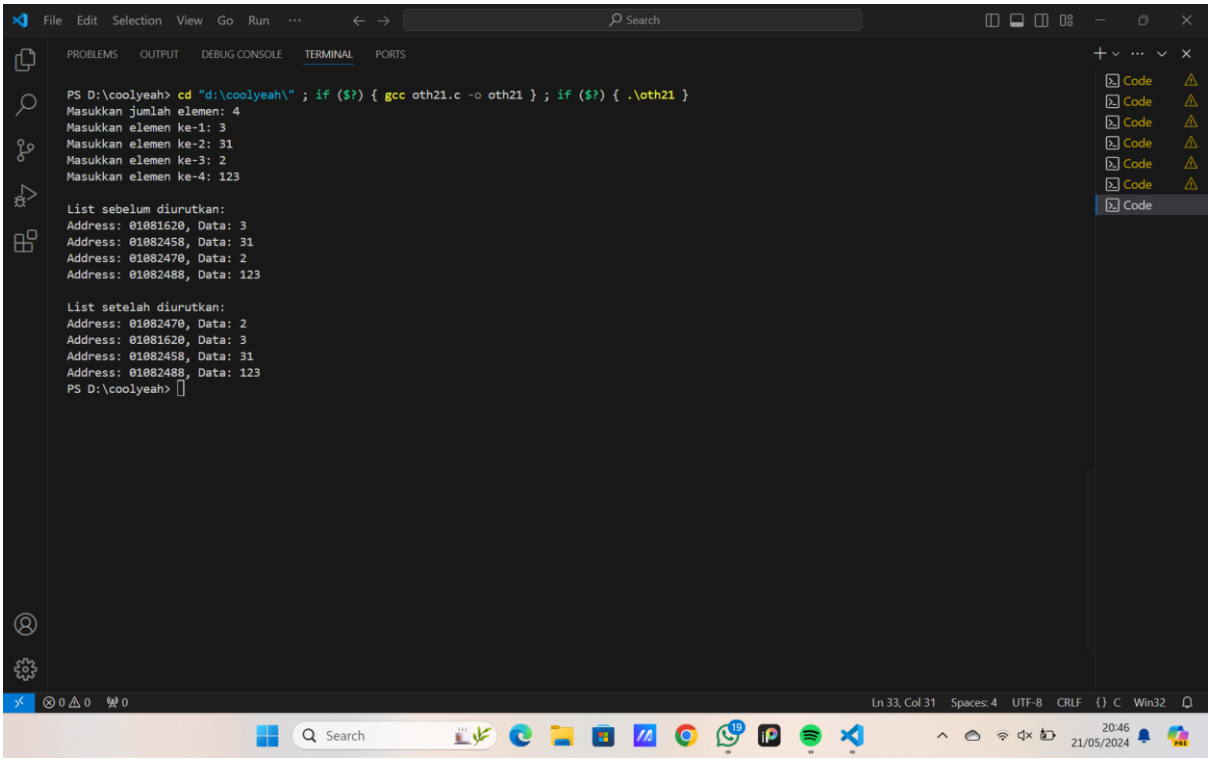
```c
#include <stdio.h>
#include <stdlib.h>

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;
    Node* current;
```

```
    int swapped;

    do {
        swapped = 0;
        current = head;
        do {
            Node *nextNode = current->next;
            if (current->data > nextNode->data) {
                swapNodes(current, nextNode);
                swapped = 1;
            } else {
                current = nextNode;
            }
        } while (current != tail);
    } while (swapped);
}

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

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

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

    sortList();

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

    return 0;
}
```

2. Output


File Edit Selection View Go Run
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc oth21.c -o oth21 } ; if ($?) { .\oth21 }
Masukkan jumlah elemen: 6
Masukkan elemen ke-1: 5
Masukkan elemen ke-2: 5
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 8
Masukkan elemen ke-5: 1
Masukkan elemen ke-6: 6
List sebelum diurutkan:
Address: 00F11620, Data: 5
Address: 00F12458, Data: 5
Address: 00F12470, Data: 3
Address: 00F12488, Data: 8
Address: 00F124A0, Data: 1
Address: 00F124B8, Data: 6
List setelah diurutkan:
Address: 00F124A0, Data: 1
Address: 00F12470, Data: 3
Address: 00F11620, Data: 5
Address: 00F12458, Data: 5
Address: 00F12488, Data: 6
Address: 00F12488, Data: 8
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc oth21.c -o oth21 } ; if ($?) { .\oth21 }
Masukkan jumlah elemen: 4
Masukkan elemen ke-1: 3
Masukkan elemen ke-2: 31
Masukkan elemen ke-3: 2
Masukkan elemen ke-4: 123
List sebelum diurutkan:
Address: 01081620, Data: 3
Address: 01082458, Data: 31
Address: 01082470, Data: 2
Address: 01082488, Data: 123
Code
Ln 33, Col 31 Spaces: 4 UTF-8 CRLF {} C Win32
Search
20:45
21/05/2024


File Edit Selection View Go Run
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc oth21.c -o oth21 } ; if ($?) { .\oth21 }
Masukkan jumlah elemen: 4
Masukkan elemen ke-1: 3
Masukkan elemen ke-2: 31
Masukkan elemen ke-3: 2
Masukkan elemen ke-4: 123
List sebelum diurutkan:
Address: 01081620, Data: 3
Address: 01082458, Data: 31
Address: 01082470, Data: 2
Address: 01082488, Data: 123
List setelah diurutkan:
Address: 01082470, Data: 2
Address: 01081620, Data: 3
Address: 01082458, Data: 31
Address: 01082488, Data: 123
PS D:\coolyeah>
Code
Ln 33, Col 31 Spaces: 4 UTF-8 CRLF {} C Win32
Search
20:46
21/05/2024

3. Penjelasan

```
#include <stdio.h> // Meng-include header file untuk input dan output standar (fungsi seperti printf dan scanf)

#include <stdlib.h> // Meng-include header file untuk fungsi utilitas standar (seperti malloc dan free)

typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
// Mendefinisikan tipe data Node yang merepresentasikan node dalam linked list sirkuler doubly. Setiap node memiliki data (integer), next (pointer ke node berikutnya), dan prev (pointer ke node sebelumnya)

Node *head = NULL;
Node *tail = NULL;
// Mendefinisikan dua pointer global head dan tail untuk menunjuk ke node pertama dan terakhir dalam linked list

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
// Fungsi createNode membuat node baru dengan data yang diberikan, mengalokasikan memori untuk node baru, dan menginisialisasi pointer next dan prev ke NULL. Mengembalikan pointer ke node yang baru dibuat

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 insertNode menambahkan node baru ke linked list. Jika linked list kosong (head == NULL), head dan tail menunjuk ke node baru yang membuatnya menjadi node tunggal yang mengarah pada dirinya sendiri (sirkuler). Jika tidak kosong, node baru ditambahkan setelah tail dan pointer di-update untuk menjaga linked list tetap sirkuler

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 printList mencetak seluruh node dalam linked list. Jika linked list kosong (curr == NULL), mencetak "List is empty". Jika tidak, iterasi dari head hingga kembali ke head lagi dan mencetak alamat dan data dari setiap node

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 swapNodes menukar posisi dua node a dan b dalam linked list. Ada dua kasus: jika a dan b bersebelahan, dan jika a dan b tidak bersebelahan. Pointer next dan prev di-update sesuai dengan posisi baru dari node yang ditukar. Jika salah satu dari a atau b adalah head atau tail, pointer head dan tail juga diperbarui

void sortList() {
    if (head == NULL) return;
    Node* current;
    int swapped;

    do {
        swapped = 0;
        current = head;
        do {
            Node *nextNode = current->next;
            if (current->data > nextNode->data) {
                swapNodes(current, nextNode);
                swapped = 1;
            } else {
                current = nextNode;
            }
        } while (current != tail);
    } while (swapped);
}
// Fungsi sortList mengurutkan linked list menggunakan metode bubble sort. Perulangan do-while dilakukan sampai tidak ada lagi node yang perlu ditukar (swapped menjadi 0). Setiap iterasi, node diperiksa dan ditukar jika data dari node saat ini lebih besar daripada nextNode

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

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

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

    sortList();

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

    return 0;
}
// Fungsi main mengelola alur program utama. Meminta pengguna untuk memasukkan jumlah elemen, membaca elemen-elemen tersebut, dan memasukkannya ke dalam linked list menggunakan insertNode. Sebelum dan sesudah pengurutan dengan sortList, linked list dicetak menggunakan printList. Program diakhiri dengan return 0
```