Nama : Rafif Muhammad

NIM : 1203230018

Kelas : IF 03-03

Laporan OTH Double Linked Lisk Circular

1. Source Code

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

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

typedef struct node node;

node *pHead = NULL;

node *nodeBaru() {
    node *pNew = (node*)malloc(sizeof(node));
    if (pNew == NULL) {
        printf("Error");
        exit(EXIT_FAILURE);
    }
    return pNew;
}

void insertLast(node *pNew) {
    printf("Masukkan Bilangan: ");
    scanf("%d", &pNew->data);

    if (pHead == NULL) {
        pNew->prev = pNew;
        pNew->next = pNew;
        pHead = pNew;
    } else {
        pNew->prev = pHead->prev;
        pNew->next = pHead;
        pHead->prev->next = pNew;
        pHead->prev = pNew;
    }
}

void View() {
    node *pWalker = pHead;

    if (pHead == NULL) {
        printf("Data kosong\n");
    } else {
        printf("\nLinked List:\n");
        do {
            printf("[%d] Alamat: %p\n", pWalker->data, (void*)pWalker);
            pWalker = pWalker->next;
        } while (pWalker != pHead);
    }
    printf("\n");
}
```

```c
void sortList() {
    if (pHead == NULL || pHead->next == pHead) {
        return;
    }

    int count = 0;
    node *temp = pHead;
    do {
        count++;
        temp = temp->next;
    } while (temp != pHead);

    node **array = (node **)malloc(count * sizeof(node *));
    temp = pHead;
    for (int i = 0; i < count; i++) {
        array[i] = temp;
        temp = temp->next;
    }

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (array[j]->data > array[j + 1]->data) {
                node *tempNode = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tempNode;
            }
        }
    }

    for (int i = 0; i < count; i++) {
        array[i]->next = array[(i + 1) % count];
        array[i]->prev = array[(i - 1 + count) % count];
    }

    pHead = array[0];

    free(array);
}

int main() {
    node *pNew;
    int pilih;

    do {
        View();
        printf("\n\n");
        printf("\n---------------------------------- \n\tDOUBLE LINKED LIST\n ----------------------------------\n");
        printf("\n1. masukkan data");
        printf("\n2. urutkan sesuai nilai data");
        printf("\n3. Exit");

        printf("\nPilihan: ");
        scanf("%d", &pilih);

        switch (pilih) {
            case 1:
                pNew = nodeBaru();
                insertLast(pNew);
                break;
            case 2:
                sortList();
                break;
            case 3:
                printf("\n");
                break;
            default:
                printf("\nPilihan tidak valid");
                break;
        }
    } while (pilih != 3);

    printf("\n");
    return 0;
}
```

2. Penjelasan

#include <stdio.h>

#include <stdlib.h>

Library yang digunakan pada Bahasa C

struct node {

struct node *prev;

int data;

struct node *next;

};

Sebuah struct dengan nama node yang memilki 3 pareameter yaitu *prev, data dan *next

typedef struct node node;

Memberikan nama alias pada struct node menjadi “node”.

node *pHead = NULL;

Menreset *pHead menjadi null

node *nodeBaru() {

Fungsi unutk mebuat node baru

node *pNew = (node*)malloc(sizeof(node));

Baris ini digunakan untuk mengalokasikan memori dinamis untuk sebuah node baru dalam linked list

if (pNew == NULL) {

printf("Error: Alokasi memori gagal.");

exit(EXIT_FAILURE);

}

Memeriksa apakah pNew sama dengan null, jika iya maka akan menampilkan output

return pNew;

Mengembalikan nilai pNew

void insertLast(node *pNew) {

Fungsi untuk menambahkan node pada paling akhir

printf("Masukkan Bilangan: ");

scanf("%d", &pNew->data);

Meminta user memasuukan bilangan

if (pHead == NULL) {

Kondisi untuk memeriksa apakah pHead adalah null, jika iya maka akan menjalankan fungsi dibawahnya

pNew->prev = pNew;

Pada prev bagian new akan mengarah pada dirinya sendiri

pNew->next = pNew;

Pada next bagian new akan mengarah pada dirinya sendiri

pHead = pNew;

pNew akan menajdi head

} else {

Apabila kondisi diatas tidak terpenuhi maka aakan menjalankan fungsi else ini

pNew->prev = pHead->prev;

Prev pada new aakan mengarah ke prev bagian head

pNew->next = pHead;

Next pada new akan mengarah ke head

pHead->prev->next = pNew;

Prev dan next pada head akan mengarah ke new

pHead->prev = pNew;

Prev pada head akan mengarah ke new

void View() {

Fungsi untuk menampilkan node yang sudah dimasukkan

node *pWalker = pHead;

Menggunakan variable tambahan yaitu walker yang berfungsi untuk menjelajahi linked list, dan walker akan diubah menjadi head

if (pHead == NULL) {

Kondisi if apabila head sama dengan null

printf("Data kosong\n");

jika kondisi diatas terpenuhi maka akan menampilkan “data kosong”

} else {

Jika kondisi diatas tidak terpenuhi maka akan menjalan fungsi else

printf("\nLinked List:\n");

do {

printf("[%d] Alamat: %p\n", pWalker->data, (void*)pWalker);

Fungsi menggunakan loop do-while untuk menampilkan data dan alamat setiap node dalam linked list.

pWalker = pWalker->next;

Walker akan berpindah ke walker next

} while (pWalker != pHead);

Perulangan ini akan berlangsung hingga walker Kembali ke head

void sortList() {

Fungsi untuk mengurutkan node

if (pHead == NULL || pHead->next == pHead)

return;

Memeriksa apakah head sama dengan null atau head next sama dengan head

int count = 0;

Menginisiasi variable count menajdi 0

node *temp = pHead;

Mengubah variabel temp menjadi head

do {

Perulangan yang berlangsung hingga temp Kembali ke phead

count++;

setiap perulangan yang terjadi count bertambah 1

temp = temp->next;

Mengarahkan temp ke temp next

} while (temp != pHead);

Perulangan yang berlangsung hingga temp Kembali ke phead

node **array = (node **)malloc(count * sizeof(node *));

temp = pHead;

for (int i = 0; i < count; i++) {

array[i] = temp;

temp = temp->next;

Membuat array pointer ke node (array) dan mengisinya dengan pointer ke setiap node dalam linked list. array dialokasikan dinamis sesuai dengan jumlah node (count). Loop for digunakan untuk mengisi array dengan node dari linked list

for (int i = 0; i < count - 1; i++) {

Perulangan luar yang akan berjalan dari i = 0 dan I < count -1

for (int j = 0; j < count - i - 1; j++) {

Perulangan dalam yang akan berjalan dari j = 0 dan j < count – i - 1

if (array[j]->data > array[j + 1]->data) {

Kondisi if jika data pada array[j] lebih besar daripada data pada array[j + 1] maka

node *tempNode = array[j];

Array J akan dipindahkan ke teampnode

array[j] = array[j + 1];

Memindahkan array [j + 1] ke dalam array [j]

array[j + 1] = tempNode;

Memindahkan tempnode kedalam array [j + 1] ke

for (int i = 0; i < count; i++) {

Loop untuk mengatur kembali pointer next dan prev dari setiap node sesuai dengan urutan baru dalam nodeArray

array[i]->next = array[(i + 1) % count];

Mengatur next untuk menunjuk ke node berikutnya dalam array

array[i]->prev = array[(i - 1 + count) % count];

Mengatur prev untuk menunjuk ke node sebelumnya dalam array

pHead = array[0];

Mengatur phead untuk menunjuk ke array

free(array);

Membebaskan memori yang dialokaasikan unutk array

}int main() {

Fungsi utama dalam sebuah perogram

node *pNew;

Pointer untuk menyimpan node baru yang akan ditambahkan ke linked list.

int pilih;

Variabel untuk menyimpan pilihan user dari menu

do {

Kondisi perulangan yang aka berjalan hingga user memilih untuk keluar

View();

Memanggil fungsi vies unutk menampilkan node yang sudah dimasukkan

printf("\n\n");

printf("\n====================================== \n\tDOUBLE LINKED LIST\n ======================================\n");

printf("\n1. masukkan data");

printf("\n2. urutkan sesuai alamat");

printf("\n3. Keluar");

printf("\nPilihan: ");

scanf("%d", &pilih);

Meminta user untuk memilih pilihan antara 1 sampai 3

switch (pilih) {

case 1:

pNew = nodeBaru();

Memanggil fungsi nodeBaru lalu menyimmpannya dalam pNew

insertLast(pNew);

Memanggil fungsi insertLast untuk memasukkan node

break;

Keluar dari switch setelah menjalankan perintah di case 1

case 2:

sortList();

Memanggil fungsi sortList diatas untuk mengurutkan node

break;

Keluar dari switch setelah menjalankan perintah di case 2

case 3:

printf("\n");

User keluar dari menu pilihan

break;

default:

printf("\nPilihan tidak valid");

Apabila user memilih pilihan di luar angak 1-3 maka akan menampilkan pesan pilihan tidak valid

break;

}while (pilih != 3);

Looping akan berjalan hingga user memilih opsi 3 untuk keluar

printf("\n");

return 0;

Program telah selesai dijalankan

3. Output

```
=====================================
        DOUBLE LINKED LIST
 =====================================

1. masukkan data
2. urutkan sesuai nilai data
3. Exit
Pilihan: 1
Masukkan Bilangan: 6

Linked List:
[5] Alamat: 00C12F60
[3] Alamat: 00C12F78
[8] Alamat: 00C12F90
[1] Alamat: 00C10C60
[6] Alamat: 00C10C78
```

```
=====================================
        DOUBLE LINKED LIST
 =====================================

1. masukkan data
2. urutkan sesuai nilai data
3. Exit
Pilihan: 2

Linked List:
[1] Alamat: 00C10C60
[3] Alamat: 00C12F78
[5] Alamat: 00C12F60
[6] Alamat: 00C10C78
[8] Alamat: 00C12F90
```