Agnes Destiny Sava S

IF-03-01/1203230092

*OTH Linked List Circular*

Source Code :

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

// Struktur node untuk double linked list circular
typedef struct alamat {
    int data;
    struct alamat* prev;
    struct alamat* next;
} alamat;

// Fungsi untuk membuat node baru
alamat* createNode(int data) {
    alamat* newNode = (alamat*)malloc(sizeof(alamat)); // Mengalokasikan
memori untuk node baru menggunakan malloc.
    newNode->data = data; // Mengatur anggota data dari node baru dengan
parameter data.
    newNode->prev = NULL; // Mengatur pointer prev menjadi NULL, menunjukkan
bahwa list kosong.
    newNode->next = NULL; // Mengatur pointer next menjadi NULL, menunjukkan
bahwa list kosong.
    return newNode; // Mengembalikan pointer ke node baru.
}

// Fungsi untuk memasukkan data ke dalam linked list
void insertNode(alamat** head, int data) {
    alamat* newNode = createNode(data); // Membuat node baru dengan data yang
diberikan.
    if (*head == NULL) { // Jika head bernilai NULL, berarti list kosong.
        newNode->next = newNode; // Mengatur pointer next dari node baru ke
dirinya sendiri.
        newNode->prev = newNode; // Mengatur pointer prev dari node baru ke
dirinya sendiri.
        *head = newNode; // Memperbarui pointer head untuk menunjuk ke node
baru.
    } else { // Jika list tidak kosong.
        alamat* tail = (*head)->prev; // Tail sebagai pointer ke node terakhir
dalam list.
        tail->next = newNode; // Mengatur pointer next dari node terakhir
untuk menunjuk ke node baru.
```

```
        newNode->prev = tail; // Mengatur pointer prev dari node baru untuk
menunjuk ke node terakhir.
        newNode->next = *head; // Mengatur pointer next dari node baru untuk
menunjuk ke node pertama (head).
        (*head)->prev = newNode; // Mengatur pointer prev dari node pertama
untuk menunjuk ke node baru.
    }
}

// Fungsi untuk menampilkan linked list
void displayList(alamat* head) {
    if (head == NULL) return; // Memeriksa apakah list kosong dengan melihat
pointer head bernilai NULL, jika iya maka return.
    alamat* temp = head; // Pointer sementara temp, diinisialisasi untuk
menunjuk ke head.
    do {
        printf("Memory address: %p, Data: %d\n", (void*)temp, temp->data); //
Mencetak alamat memori node saat ini (temp) dan data yang disimpan.
        temp = temp->next; // Memperbarui temp untuk menunjuk node berikutnya.
    } while (temp != head); // Berhenti ketika temp kembali ke head.
}

// Fungsi untuk menukar dua node dalam linked list
void swapNodes(alamat** head, alamat* node1, alamat* node2) {
    if (node1 == node2) return; // Jika node 1 dan 2 adalah node yang sama,
maka akan return.

    alamat* prev1 = node1->prev; // Prev1 menunjuk node sebelum node 1.
    alamat* next1 = node1->next; // Next1 menunjuk node setelah node 1.
    alamat* prev2 = node2->prev; // Prev2 menunjuk node sebelum node 2.
    alamat* next2 = node2->next; // Next2 menunjuk node setelah node 2.

    if (node1->next == node2) { // Jika node1 dan 2 berdampingan dengan node 2
setelah node 1.
        node1->next = next2; // Menunjuk ke next2 (node setelah node 2).
        node1->prev = node2; // Menunjuk ke node 2.
        node2->next = node1; // Menunjuk ke node 1.
        node2->prev = prev1; // Menunjuk ke prev1 (node sebelum node 1).
        if (next2 != NULL) next2->prev = node1; // Menunjuk ke node 1.
        if (prev1 != NULL) prev1->next = node2; // Menunjuk ke node 2.
    } else if (node2->next == node1) { // Jika node 2 dan 1 berdampingan
dengan node 1 setelah node 2.
        node2->next = next1; // Menunjuk ke next1 (node setelah node 1).
        node2->prev = node1; // Menunjuk ke node 1.
        node1->next = node2; // Menunjuk ke node 2.
        node1->prev = prev2; // Menunjuk ke prev2 (node sebelum node 2).
        if (next1 != NULL) next1->prev = node2; // Menunjuk node 2.
        if (prev2 != NULL) prev2->next = node1; // Menunjuk node 1.
```

```
    } else { // Jika node 1 dan 2 tidak berdampingan.
        if (prev1 != NULL) prev1->next = node2; // Menunjuk ke node 2.
        if (next1 != NULL) next1->prev = node2; // Menunjuk ke node 2.
        if (prev2 != NULL) prev2->next = node1; // Menunjuk ke node 1.
        if (next2 != NULL) next2->prev = node1; // Menunjuk ke node 1.

        node1->next = next2; // Menunjuk ke next2.
        node1->prev = prev2; // Menunjuk ke prev2.
        node2->next = next1; // Menunjuk ke next1.
        node2->prev = prev1; // Menunjuk ke prev1.
    }

    if (*head == node1) *head = node2; // Jika node 1 adalah head, maka head
diperbarui untuk menunjuk node 2.
    else if (*head == node2) *head = node1; // Jika node 2 adalah head, maka
head diperbarui untuk menunjuk node 1.
}

// Fungsi untuk mengurutkan linked list dengan menukar node
void sortList(alamat** head) {
    if (*head == NULL || (*head)->next == *head) return;
    // Memeriksa list kosong atau hanya memiliki satu node saja, jika iya maka
akan return.

    alamat* i;
    alamat* j;
    for (i = *head; i->next != *head; i = i->next) { // Loop untuk node i
mulai dari head ke node terakhir.
        for (j = i->next; j != *head; j = j->next) { // Loop untuk node j
mulai dari node setelah i ke head.
            if (i->data > j->data) { // Jika data node i lebih besar dari data
node j.
                swapNodes(head, i, j); // Menukar posisi node i dan j.
                alamat* temp = i;
                i = j;
                j = temp; // Menukar pointer untuk mempertahankan posisi loop.
            }
        }
    }
}

int main() {
    int N, data;
    alamat* head = NULL;
    // N untuk menyimpan jumlah data yang akan dimasukkan.
    // data untuk menyimpan data sementara yang akan dimasukkan.
    // head adalah pointer ke node pertama dari list, diinisialisasi dengan
NULL.
```

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

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

    // Tampilkan list sebelum pengurutan
    printf("\nList sebelum pengurutan:\n");
    displayList(head);

    // Mengurutkan list dengan menukar node
    sortList(&head);

    // Tampilkan list setelah pengurutan
    printf("\nList setelah pengurutan:\n");
    displayList(head);

    // Membersihkan memori yang dialokasikan
    alamat* current = head;
    alamat* nextNode;

    // Deklarasi dua pointer current dan nextNode
    do {
        nextNode = current->next;
        free(current); // Membebaskan memori yang dialokasikan untuk node saat
ini.
        current = nextNode; // Memperbarui current untuk menunjuk ke node
berikutnya.
    } while (current != head); // Berhenti ketika current kembali ke head.

    return 0; // Mengembalikan 0, menandakan program selesai dengan sukses.
}
```

Output:

```
PS C:\AlPro> cd "c:\AlPro\ASD\" ; if ($?)
Masukkan jumlah data: 6
Masukkan data ke-1: 5
Masukkan data ke-2: 5
Masukkan data ke-3: 3
Masukkan data ke-4: 8
Masukkan data ke-5: 1
Masukkan data ke-6: 6

List sebelum pengurutan:
Memory address: 00F61700, Data: 5
Memory address: 00F62500, Data: 5
Memory address: 00F62518, Data: 3
Memory address: 00F62530, Data: 8
Memory address: 00F62548, Data: 1
Memory address: 00F62560, Data: 6

List setelah pengurutan:
Memory address: 00F62548, Data: 1
Memory address: 00F62518, Data: 3
Memory address: 00F61700, Data: 5
Memory address: 00F62500, Data: 5
Memory address: 00F62560, Data: 6
Memory address: 00F62530, Data: 8
```

```
PS C:\AlPro\ASD> cd "c:\AlPro\ASD\" ; if ($?)
Masukkan jumlah data: 4
Masukkan data ke-1: 3
Masukkan data ke-2: 31
Masukkan data ke-3: 2
Masukkan data ke-4: 123

List sebelum pengurutan:
Memory address: 01141700, Data: 3
Memory address: 01142500, Data: 31
Memory address: 01142518, Data: 2
Memory address: 01142530, Data: 123

List setelah pengurutan:
Memory address: 01142518, Data: 2
Memory address: 01141700, Data: 3
Memory address: 01142500, Data: 31
Memory address: 01142530, Data: 123
PS C:\AlPro\ASD>
```