Arka dwi indrastata

1203230017

IF-03-02

OTH CIRCULAR DOUBLE LINKED LIST

***CODE**

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

typedef struct Node *NodePointer;

struct Node {
    int value;
    NodePointer next;
    NodePointer prev;
} *head = NULL, *tail = NULL;

NodePointer createNode(int val) {
    NodePointer temp = (NodePointer)malloc(sizeof(struct Node));
    temp->value = val;
    temp->next = NULL;
    temp->prev = NULL;
    return temp;
}

void insert_last(int val) {
    NodePointer temp = createNode(val);
    if (head == NULL) {
        head = tail = temp;
        head->next = head;
        head->prev = head;
    } else {
        temp->next = head;
        temp->prev = tail;
        tail->next = temp;
        head->prev = temp;
        tail = temp;
    }
}

void swap_nodes(NodePointer a, NodePointer b) {
```

```
    if (a == b) return;

    NodePointer aPrev = a->prev;
    NodePointer aNext = a->next;
    NodePointer bPrev = b->prev;
    NodePointer bNext = b->next;

    if (a->next == b) {
        a->next = bNext;
        a->prev = b;
        b->next = a;
        b->prev = aPrev;

        if (aPrev != NULL) aPrev->next = b;
        if (bNext != NULL) bNext->prev = a;
    } else if (b->next == a) {
        b->next = aNext;
        b->prev = a;
        a->next = b;
        a->prev = bPrev;

        if (bPrev != NULL) bPrev->next = a;
        if (aNext != NULL) aNext->prev = b;
    } else {
        a->next = bNext;
        a->prev = bPrev;
        b->next = aNext;
        b->prev = aPrev;

        if (aNext != NULL) aNext->prev = b;
        if (aPrev != NULL) aPrev->next = b;
        if (bNext != NULL) bNext->prev = a;
        if (bPrev != NULL) bPrev->next = a;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}
```

```c
void sort_ascending() {
    if (head == NULL) return;

    int swapped;
    NodePointer ptr1;
    NodePointer lptr = NULL;

    do {
        swapped = 0;
        ptr1 = head;

        do {
            if (ptr1->next != head && ptr1->value > ptr1->next->value) {
                swap_nodes(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        } while (ptr1->next != head);

        lptr = ptr1;
    } while (swapped);
}

void print_list() {
    if (head == NULL) return;

    NodePointer temp = head;
    do {
        printf("Address: %p, Data: %d\n", (void*)temp, temp->value);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}

int main() {
    int num_elements, i, value;

    scanf("%d", &num_elements);

    if (num_elements < 1 || num_elements > 10) {
        printf("Jumlah data harus antara 1 dan 10.\n");
        return 1;
    }

    for (i = 0; i < num_elements; i++) {
        printf("Masukkan data ke-%d: ", i + 1);
        scanf("%d", &value);
        insert_last(value);
```

```
    }

    printf("\n");
    print_list();

    sort_ascending();

    print_list();

    return 0;
}
```

***OUTPUT**

```
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

Address: 00BD13B0, Data: 5
Address: 00BD13C8, Data: 3
Address: 00BD13E0, Data: 8
Address: 00BD0DE8, Data: 1
Address: 00BD0E00, Data: 6

Address: 00BD0DE8, Data: 1
Address: 00BD13C8, Data: 3
Address: 00BD13B0, Data: 5
Address: 00BD0E00, Data: 6
Address: 00BD13E0, Data: 8

PS E:\Coding\praktikum asd>
```

```
3
Masukkan data ke-1: 31
Masukkan data ke-2: 2
Masukkan data ke-3: 123

Address: 00AF13B0, Data: 31
Address: 00AF13C8, Data: 2
Address: 00AF13E0, Data: 123

Address: 00AF13C8, Data: 2
Address: 00AF13B0, Data: 31
Address: 00AF13E0, Data: 123
```

***PENJELASAN**

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

typedef struct Node *NodePointer;

struct Node {
    int value;
    NodePointer next;
    NodePointer prev;
} *head = NULL, *tail = NULL;
```

1terdapat 2 header file yaitu <stdio.h> dan <stdlib.lib>.fungsi stdio.h untuk standar input output dan stdio.lib untuk standar library

2. NodePointer: Tipe pointer ke struct Node.

3.struct Node: Struktur data yang mendefinisikan node untuk linked list. Setiap node memiliki nilai integer (value), pointer ke node berikutnya (next), dan pointer ke node sebelumnya (prev).

4.head dan tail: Pointer global untuk menunjuk ke node pertama dan terakhir dalam linked list.

```c
NodePointer createNode(int val) {
    NodePointer temp = (NodePointer)malloc(sizeof(struct Node));
    temp->value = val;
    temp->next = NULL;
```

```
    temp->prev = NULL;
    return temp;
}
```

#createNode: Membuat node baru dengan nilai yang diberikan (val).

1.malloc: Mengalokasikan memori untuk node baru.

2.temp: Pointer ke node baru yang diinisialisasi dengan nilai yang diberikan dan next serta prev diatur ke NULL.

3.return: Mengembalikan pointer ke node baru.

```
void insert_last(int val) {
    NodePointer temp = createNode(val);
    if (head == NULL) {
        head = tail = temp;
        head->next = head;
        head->prev = head;
    } else {
        temp->next = head;
        temp->prev = tail;
        tail->next = temp;
        head->prev = temp;
        tail = temp;
    }
}
```

#insert_last: Menambahkan node baru dengan nilai yang diberikan ke akhir linked list.

1.createNode: Membuat node baru (temp) dengan nilai yang diberikan.

2.if (head == NULL): Jika linked list kosong, node baru menjadi head dan tail. Node ini menunjuk ke dirinya sendiri sebagai next dan prev.

3.else: Jika linked list tidak kosong, node baru ditempatkan di antara tail dan head. tail diupdate untuk menunjuk ke node baru.

```
void swap_nodes(NodePointer a, NodePointer b) {
    if (a == b) return;

    NodePointer aPrev = a->prev;
    NodePointer aNext = a->next;
    NodePointer bPrev = b->prev;
```

```
    NodePointer bNext = b->next;

    if (a->next == b) {
        a->next = bNext;
        a->prev = b;
        b->next = a;
        b->prev = aPrev;

        if (aPrev != NULL) aPrev->next = b;
        if (bNext != NULL) bNext->prev = a;
    } else if (b->next == a) {
        b->prev = a;
        a->next = b;
        a->prev = bPrev;

        if (bPrev != NULL) bPrev->next = a;
        if (aNext != NULL) aNext->prev = b;
    } else {
        a->next = bNext;
        a->prev = bPrev;
        b->next = aNext;
        b->prev = aPrev;

        if (aNext != NULL) aNext->prev = b;
        if (aPrev != NULL) aPrev->next = b;
        if (bNext != NULL) bNext->prev = a;
        if (bPrev != NULL) bPrev->next = a;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}
```

#swap_nodes: Menukar dua node (a dan b) dalam linked list.

1.if (a == b): Jika node yang sama diberikan, tidak ada operasi swap yang dilakukan.

2.Pointer adjustments: Menyesuaikan pointer next dan prev dari node dan tetangganya untuk menukar posisi node a dan b.

3.Update head and tail: Memperbarui pointer head dan tail jika salah satu node yang ditukar adalah head atau tail.

```
void sort_ascending() {
    if (head == NULL) return;

    int swapped;
    NodePointer ptr1;
    NodePointer lptr = NULL;

    do {
        swapped = 0;
        ptr1 = head;

        do {
            if (ptr1->next != head && ptr1->value > ptr1->next->value) {
                swap_nodes(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        } while (ptr1->next != head);

        lptr = ptr1;
    } while (swapped);
}
```

#sort_ascending: Mengurutkan linked list dalam urutan menaik menggunakan algoritma bubble sort.

2.swapped: Menunjukkan apakah ada node yang ditukar dalam satu iterasi.

3.ptr1: Pointer yang digunakan untuk iterasi melalui node.

4.do-while loop: Melakukan iterasi sampai tidak ada swap yang terjadi.

5.if (ptr1->value > ptr1->next->value): Menukar dua node jika node saat ini memiliki nilai lebih besar dari node berikutnya.

```
void print_list() {
    if (head == NULL) return;

    NodePointer temp = head;
    do {
        printf("Address: %p, Data: %d\n", (void*)temp, temp->value);
        temp = temp->next;
```

```
    } while (temp != head);
    printf("\n");
}
```

#print_list: Mencetak nilai dari setiap node dalam linked list beserta alamat memori mereka.

1.if (head == NULL): Jika linked list kosong, fungsi keluar.

2.do-while loop: Mengiterasi melalui linked list dan mencetak data dari setiap node sampai kembali ke head.

```
int main() {
    int num_elements, i, value;

    scanf("%d", &num_elements);

    if (num_elements < 1 || num_elements > 10) {
        printf("Jumlah data harus antara 1 dan 10.\n");
        return 1;
    }

    for (i = 0; i < num_elements; i++) {
        printf("Masukkan data ke-%d: ", i + 1);
        scanf("%d", &value);
        insert_last(value);
    }

    printf("\n");
    print_list();

    sort_ascending();

    print_list();

    return 0;
}
```

#main: Fungsi utama yang mengeksekusi program.

1.num_elements: Menyimpan jumlah elemen yang akan dimasukkan ke dalam linked list.

2.scanf: Membaca jumlah elemen dari input pengguna.

3.if (num_elements < 1 || num_elements > 10): Memeriksa apakah jumlah elemen berada dalam rentang yang diperbolehkan (1-10). Jika tidak, program keluar dengan pesan kesalahan.

4.for loop: Mengiterasi untuk memasukkan nilai ke dalam linked list.

5.insert_last: Menambahkan nilai baru ke akhir linked list.

6.print_list: Mencetak linked list sebelum dan sesudah pengurutan.

7.sort_ascending: Mengurutkan linked list dalam urutan menaik.