

TUGAS OTH

SATRIO DWI YANDA ARIFIN

1203230020

IF 03-01

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

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = newNode->prev = NULL;
    return newNode;
}

void insertEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        newNode->next = newNode->prev = newNode;
    } else {
        struct Node* last = (*head)->prev;
        newNode->next = *head;
        (*head)->prev = newNode;
        newNode->prev = last;
        last->next = newNode;
    }
}

void printList(struct Node* head) {
    if (head == NULL) return;
    struct Node* temp = head;
    do {
        printf("Address: %p, Data: %d\n", (void*)temp, temp->data);
        temp = temp->next;
    } while (temp != head);
}
```

```

}

void sortList(struct Node** head) {
    if (*head == NULL || (*head)->next == *head) return;

    struct Node *i, *j;
    for (i = (*head); i->next != *head; i = i->next) {
        for (j = i->next; j != *head; j = j->next) {
            if (i->data > j->data) {

                int tempData = i->data;
                i->data = j->data;
                j->data = tempData;
            }
        }
    }
}

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

    scanf("%d", &N);
    int data;

    for (int i = 0; i < N; i++) {
        scanf("%d", &data);
        insertEnd(&head, data);
    }

    printList(head);

    sortList(&head);

    printList(head);

    return 0;
}

```

```
9
10 struct Node* createNode(int data) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = data;
13     newNode->next = newNode->prev = NULL;
14     return newNode;
15 }
16
17 void insertEnd(struct Node** head, int data) {
18     struct Node* newNode = createNode(data);
19     if (*head == NULL) {
20         *head = newNode;
21         newNode->next = newNode->prev = newNode;
22     } else {
23         struct Node* last = (*head)->prev;
```

8
1
6
Address: 00F21908, Data: 5
Address: 00F21920, Data: 3
Address: 00F21938, Data: 8
Address: 00F21990, Data: 1
Address: 00F219A8, Data: 6
Address: 00F21908, Data: 1
Address: 00F21920, Data: 3
Address: 00F21938, Data: 5
Address: 00F21990, Data: 6
Address: 00F219A8, Data: 8
PS C:\Users\Satrio Dwi Y\Documents\UTP ASD 2024> cd "c:\Users\Satrio Dwi Y\Documents\UTP ASD 2024\" ; if (\$?) { g++ yollan.cpp -o yollan } ; if (\$?) { .\yollan }

```
45 void sortList(struct Node** head) {
58 }
59 }
60
61 int main() {
62     struct Node* head = NULL;
63     int N;
64
65     // Input the number of elements
66     scanf("%d", &N);
67     int data;
68
69     // Input the elements and create the circular doubly linked list
70     for (int i = 0; i < N; i++) {
71         scanf("%d", &data);
```

PS C:\Users\Satrio Dwi Y\Documents\UTP ASD 2024> cd "c:\Users\Satrio Dwi Y\Documents\UTP ASD 2024\" ; if (\$?) { g++ yollan.cpp -o yollan } ; if (\$?) { .\yollan }

3
31
2
123
Address: 009D1908, Data: 31
Address: 009D1920, Data: 2
Address: 009D1938, Data: 123
Address: 009D1908, Data: 2
Address: 009D1920, Data: 31
Address: 009D1938, Data: 123
PS C:\Users\Satrio Dwi Y\Documents\UTP ASD 2024>

Struktur Node dan Fungsi-fungsinya

Definisi Struktur Node

c

Copy code

```
typedef struct Node { int data; struct Node next; struct Node prev; }
```

- **struct Node** mendefinisikan struktur dari sebuah node dalam linked list.
- **data** menyimpan nilai integer dari node.
- **next** adalah pointer yang menunjuk ke node berikutnya.
- **prev** adalah pointer yang menunjuk ke node sebelumnya.

Fungsi createNode

c

Copy code

```
struct Node* createNode(int data) { struct Node* newNode = malloc(sizeof(struct Node)); newNode->data = data; newNode->next = newNode; newNode->prev = newNode; return newNode; }
```

- **createNode** membuat node baru dengan nilai **data**.
- **malloc** mengalokasikan memori untuk node baru.
- **next** dan **prev** dari node baru menunjuk ke dirinya sendiri (circular).

Fungsi insertEnd

c

Copy code

```
void insertEnd(struct Node* head, int data) { struct Node* newNode = createNode(data); if (head == NULL) { head = newNode; return; } struct Node* tail = head; while (tail->next != NULL) { tail = tail->next; } tail->next = newNode; }
```

- **insertEnd** menambahkan node baru di akhir linked list.
- Jika **head** adalah **NULL**, list kosong, node baru menjadi head.
- Jika list tidak kosong, node baru ditambahkan di antara node tail dan head.
- **tail** adalah node terakhir sebelum node baru.
- **newNode** menjadi node terakhir dan menghubungkan dirinya dengan **tail** dan **head**.

Fungsi printList

c

Copy code

```
void printList(struct Node* head) { if (head == NULL) { return; } struct Node* current = head; do { printf("Address: %p, Data: %d\n", current, current->data); current = current->next; } while (current != head); }
```

- **printList** mencetak alamat dan data dari setiap node dalam list.
- Jika **head** adalah **NULL**, list kosong dan tidak ada yang dicetak.
- Loop **do-while** digunakan untuk mengunjungi setiap node mulai dari **head** hingga kembali ke **head**.

Fungsi sortList

c

Copy code

```
sortList if return NULL NULL do
if else
while if
else
while return
```

- `sortList` mengurutkan linked list dengan mengubah posisi node, bukan data.
- `sorted` adalah pointer untuk menjaga list yang sudah diurutkan.
- `curr` adalah node saat ini yang sedang diurutkan.
- Loop `do-while` berjalan sampai semua node diproses.
- `next` menyimpan node berikutnya sebelum `curr` dilepaskan dari list asalnya.
- Node `curr` di-insert pada posisi yang tepat dalam list `sorted`.
- Jika `sorted` masih kosong, `curr` menjadi node pertama `sorted`.
- Node `curr` ditempatkan sebelum node `temp` jika data `curr` lebih kecil dari `temp`.
- Fungsi memastikan bahwa node-node di `sorted` tetap terurut.

Fungsi main

c

Copy code

```
int main int scanf "%d" NULL for int 0 int
scanf "%d" printf "List before sorting:\n"
printf "List after sorting:\n" return 0
```

- Fungsi `main` adalah titik awal program.
- `N` adalah jumlah node yang akan dimasukkan.
- Loop `for` membaca `N` data dan memasukkan ke linked list menggunakan `insertEnd`.
- Mencetak list sebelum pengurutan.
- Mengurutkan list menggunakan `sortList`.
- Mencetak list setelah pengurutan.

Dengan penjelasan ini, setiap bagian kode memiliki fungsinya yang jelas untuk membuat, menampilkan, dan mengurutkan circular double linked list.