Nama : muamar haikal fauzan
Nim : 1203230118
Kelas : IF 03 02

SOURCE CODE

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

// Definisi struktur node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
```

definisi struktur `Node` yang memiliki tiga anggota: `data` untuk menyimpan nilai, `next` untuk menunjukkan ke node berikutnya, dan `prev` untuk menunjukkan ke node sebelumnya.

```c
// Fungsi untuk membuat node baru
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

Fungsi ini digunakan untuk membuat node baru dengan nilai `data` yang diberikan dan mengembalikan pointer ke node tersebut.

```c
// Fungsi untuk menambahkan node baru ke dalam list
void insertNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
```

```
    if (*head == NULL) {
        *head = newNode;
        (*head)->next = *head;
        (*head)->prev = *head;
    } else {
        struct Node* last = (*head)->prev;
        last->next = newNode;
        newNode->prev = last;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}
```

Fungsi ini digunakan untuk menambahkan node baru ke dalam list. Jika list kosong, maka node baru akan menjadi head. Jika tidak, node baru akan ditambahkan di belakang dan dihubungkan dengan node-node lainnya.

```
// Fungsi untuk mengurutkan list secara ascending
void sortList(struct Node** head) {
    if (*head == NULL) return;
```

Fungsi ini mengurutkan list secara ascending dengan menggunakan algoritma pengurutan sirkular double linked list. Data pada node tidak diubah, hanya posisi node-node yang diubah.

```
    struct Node *current = *head, *index = NULL;
    int temp;

    do {
        index = current->next;

        while (index != *head) {
            if (current->data > index->data) {
                temp = current->data;
                current->data = index->data;
```

```
            index->data = temp;
        }
        index = index->next;
    }

    current = current->next;
  } while (current != *head);
}

// Fungsi untuk menampilkan list beserta alamat memori dan data
void displayListWithAddress(struct Node* head) {
  if (head == NULL) return;
```

Fungsi ini menampilkan alamat memori dan data dari setiap node dalam list sebelum dan setelah pengurutan.

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

// Fungsi untuk menghapus seluruh list
void deleteList(struct Node** head) {
  if (*head == NULL) return;
```

Fungsi ini menghapus seluruh list dari memory setelah penggunaan.

```
  struct Node *current = *head, *temp = NULL;

  do {
    temp = current;
    current = current->next;
```

```
        free(temp);
    } while (current != *head);

    *head = NULL;
}

int main() {
    struct Node* head = NULL;
    int n, data;

    printf("Masukkan jumlah data : ");
    scanf("%d", &n);

    if (n < 1 || n > 10) {
        printf("Jumlah data tidak valid.\n");
        return 1;
    }

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

    printf("\nList sebelum pengurutan:\n");
    displayListWithAddress(head);

    sortList(&head);

    printf("List setelah pengurutan:\n");
    displayListWithAddress(head);

    deleteList(&head);

    return 0;
}
```

* Program dimulai dari main yang merupakan entry point dari program.
* Pengguna diminta untuk memasukkan jumlah data dan data itu sendiri.
* Data dimasukkan ke dalam sirkular double linked list menggunakan fungsi insertNode.
* List sebelum pengurutan ditampilkan dengan menggunakan fungsi displayListWithAddress.
* List diurutkan menggunakan fungsi sortList.
* List setelah pengurutan ditampilkan dengan menggunakan fungsi displayListWithAddress.
* Seluruh list dihapus dari memory menggunakan fungsi deleteList.

Program selesai.

HASIL OUTPUT