Gabriel Chrisoberryl Guevara Kaawoan
1203230041
IF 03-03

Source Code

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

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

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

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

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

```
void swapNodes(Node** head, Node* a, Node* b) {
    if (a == b || *head == NULL) return;

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

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

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

void sortList(Node** head) {
    if (*head == NULL) return;
    int swapped;
```

```
    Node* ptr1;
    Node* lptr = NULL;

    do {
        swapped = 0;
        ptr1 = *head;

        while (ptr1->next != lptr && ptr1->next != *head) {
            if (ptr1->data > ptr1->next->data) {
                swapNodes(head, ptr1, ptr1->next);
                swapped = 1;
                Node* temp = ptr1;
                ptr1 = ptr1->prev;
                ptr1->next = temp;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}


int main() {
    int N;
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int data;
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        insertEnd(&head, data);
    }

    printf("\nList before sorting:\n");
    printList(head);

    sortList(&head);

    printf("\nList after sorting:\n");
    printList(head);
```

```
    return 0;
}
```

Penjelasan Source Code

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

- Menyertakan pustaka standar input/output (stdio.h) dan pustaka standar untuk fungsi manajemen memori (stdlib.h).

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

- Mendefinisikan struktur Node untuk merepresentasikan setiap node dalam circular doubly linked list. Setiap node memiliki integer data, serta pointer ke node berikutnya (next) dan sebelumnya (prev).

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

- Fungsi createNode mengalokasikan memori untuk node baru, menginisialisasi data dengan nilai yang diberikan, dan mengatur pointer next dan prev untuk menunjuk pada dirinya sendiri, menandakan satu node dalam list sirkular.

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

- Fungsi insertEnd menambahkan node baru di akhir circular doubly linked list. Fungsi ini menggunakan createNode untuk membuat node baru. Jika list kosong (*head == NULL), node baru menjadi head. Jika tidak, pointer diperbarui untuk memasukkan node baru di akhir list dan mempertahankan struktur sirkular.

```
void printList(Node* head) {
```

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

- Fungsi printList mencetak data dari setiap node dalam list mulai dari head. Fungsi ini terus mencetak sampai kembali ke head, menandakan satu siklus penuh melalui list sirkular.

```
void swapNodes(Node** head, Node* a, Node* b) {
    if (a == b || *head == NULL) return;

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

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

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

```
        a->prev = bPrev;
        bNext->prev = a;
        a->next = bNext;
    }
}
```

- Fungsi swapNodes menukar posisi dua node dalam circular doubly linked list dengan memperbarui pointer next dan prev mereka, tanpa menukar data mereka. Fungsi ini menangani kasus di mana node bersebelahan dan tidak bersebelahan, serta memperbarui pointer head jika perlu.

```
void sortList(Node** head) {
    if (*head == NULL) return;
    int swapped;
    Node* ptr1;
    Node* lptr = NULL;

    do {
        swapped = 0;
        ptr1 = *head;

        while (ptr1->next != lptr && ptr1->next != *head) {
            if (ptr1->data > ptr1->next->data) {
                swapNodes(head, ptr1, ptr1->next);
                swapped = 1;
                Node* temp = ptr1;
                ptr1 = ptr1->prev;
                ptr1->next = temp;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
```

- Fungsi sortList mengurutkan circular doubly linked list menggunakan algoritma bubble sort. Fungsi ini secara berulang menukar node yang berdekatan yang tidak dalam urutan yang benar dan terus melakukannya sampai tidak ada pertukaran yang diperlukan (list sudah diurutkan).

```
int main() {
    int N;
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    Node* head = NULL;
    for (int i = 0; i < N; i++) {
```

```
        int data;
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        insertEnd(&head, data);
    }

    printf("\nList before sorting:\n");
    printList(head);

    sortList(&head);

    printf("\nList after sorting:\n");
    printList(head);

    return 0;
}
```

- Fungsi main :
  - ❖ Meminta pengguna memasukkan jumlah elemen N.
  - ❖ Menginisialisasi list kosong (head = NULL).
  - ❖ Mengulang sebanyak N kali untuk membaca elemen dari pengguna dan memasukkannya di akhir list menggunakan insertEnd.
  - ❖ Mencetak list sebelum diurutkan menggunakan printList.
  - ❖ Mengurutkan list menggunakan sortList.
  - ❖ Mencetak list setelah diurutkan menggunakan printList.
  - ❖ Mengembalikan 0, menandakan eksekusi berhasil.

Hasil Run Program

```
PS C:\Users\varry> cd "c:\Users\varry\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter the number of elements: 6
Enter element 1: 4
Enter element 2: 32
Enter element 3: 7
Enter element 4: 5
Enter element 5: 24
Enter element 6: 2

List before sorting:
address:00D32E90 4
address:00D315F0 32
address:00D31608 7
address:00D31620 5
address:00D31638 24
address:00D31650 2

List after sorting:
address:00D31650 2
address:00D32E90 4
address:00D31620 5
address:00D31608 7
address:00D31638 24
address:00D315F0 32
PS C:\Users\varry>
```