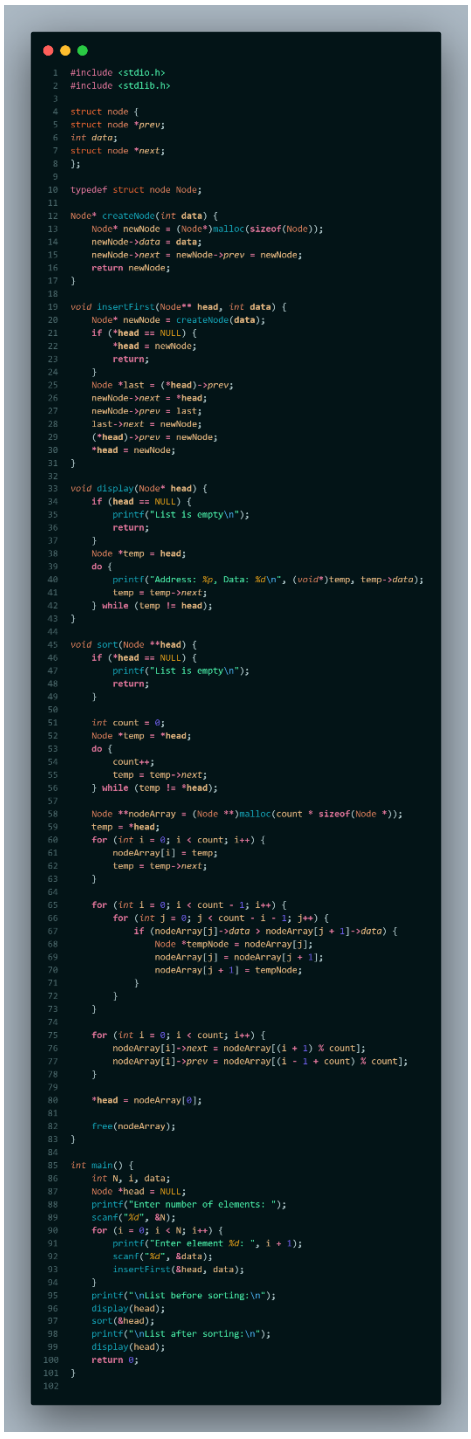
Nama : Gardha Dananjaya

NIM : 1203230105

SS Source Code



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

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

typedef struct node Node;

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

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

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

void sort(Node **head) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }

    int count = 0;
    Node *temp = *head;
    do {
        count++;
        temp = temp->next;
    } while (temp != *head);

    Node **nodeArray = (Node **)malloc(count * sizeof(Node *));
    temp = *head;
    for (int i = 0; i < count; i++) {
        nodeArray[i] = temp;
        temp = temp->next;
    }

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (nodeArray[j]->data > nodeArray[j + 1]->data) {
                Node *tempNode = nodeArray[j];
                nodeArray[j] = nodeArray[j + 1];
                nodeArray[j + 1] = tempNode;
            }
        }
    }

    for (int i = 0; i < count; i++) {
        nodeArray[i]->next = nodeArray[(i + 1) % count];
        nodeArray[i]->prev = nodeArray[(i - 1 + count) % count];
    }

    *head = nodeArray[0];

    free(nodeArray);
}

int main() {
    int N, i, data;
    Node *head = NULL;
    printf("Enter number of elements: ");
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        insertFirst(&head, data);
    }
    printf("\nList before sorting:\n");
    display(head);
    sort(&head);
    printf("\nList after sorting:\n");
    display(head);
    return 0;
}
```

PENJELASAN

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

Header untuk input output dan library.

```c
struct node {
struct node *prev;
int data;
struct node *next;
};

typedef struct node Node;
```

Sebuah struct dengan nama node yang memilki 3 pareameter yaitu *prev, data dan *next, dan memberi alias berupa “Node” agar memudahkan pemanggilan.

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

Fungsi untuk membuat dan menginisialisasi node baru dan memberi alokasi memory untuk node baru, serta mengatur “data”, “next”, “prev” dan mengembalikan nilai newNode.

```c
void insertFirst(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node *last = (*head)->prev;
    newNode->next = *head;
    newNode->prev = last;
    last->next = newNode;
    (*head)->prev = newNode;
    *head = newNode;
}
```

Fungsi ini untuk memasukkan node baru yang telah diinputkan kedalam ikatan paling depan karena insertFirst. Jika linked list masih kosong, node baru ditetapkan sebagai kepala (head) dan jika linkes list sudah ada isinya maka node baru dihubungkan dengan node terakhir dan node pertama, kemudian head diperbarui untuk menunjuk ke node baru.

```c
void display(Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
```

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

Fungsi ini untuk menampilkan isi dalam Linked List, jika Linked List tidak ada isinya maka akan muncul pesan “ List is empty”, jika linked list ada isinya maka akan mencetak memory address dan value yang ada dalam memory tersebut.

```
void sort(Node **head) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }

    int count = 0;
    Node *temp = *head;
    do {
        count++;
        temp = temp->next;
    } while (temp != *head);

    Node **nodeArray = (Node **)malloc(count * sizeof(Node *));
    temp = *head;
    for (int i = 0; i < count; i++) {
        nodeArray[i] = temp;
        temp = temp->next;
    }

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (nodeArray[j]->data > nodeArray[j + 1]->data) {
                Node *tempNode = nodeArray[j];
                nodeArray[j] = nodeArray[j + 1];
                nodeArray[j + 1] = tempNode;
            }
        }
    }

    for (int i = 0; i < count; i++) {
        nodeArray[i]->next = nodeArray[(i + 1) % count];
        nodeArray[i]->prev = nodeArray[(i - 1 + count) % count];
    }
```

```
    *head = nodeArray[0];

    free(nodeArray);
}
```

Fungsi ini menggunakan algoritma bubble sort dan mengurutkan secara ascending. Jika linked list kosong maka muncul pesan “List is empty”. Pertama-tama menghitung jumlah data yang ada di linked list dengan loop do-while. Membuat array dan mengurutkan berdasarkan data, lalu mengatur pointer untuk prev dan next sesuai urutan. Mengatur head = node pertama.

```
int main() {
    int N, i, data;
    Node *head = NULL;
    printf("Enter number of elements: ");
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        insertFirst(&head, data);
    }
    printf("\nList before sorting:\n");
    display(head);
    sort(&head);
    printf("\nList after sorting:\n");
    display(head);
    return 0;
}
```

Kemudian dalan fungsi main utama terdapat pendeklarasian variabel x dan data bertipe integer serta inisialisasi node kosong dengan *head diinisiasikan dengan NULL. Kemudian meminta inputan banyaknya data yang ingin dimasukkan oleh pengguna. Terdapat perulangan for dengan batas banyaknya jumlah inputan yang diinginkan pengguna dengan setiap perulangan akan meminta inputan nilai data yang nilainya akan dimasukkan dalam linked list dengan memanggil fungsi insertFirst. Kemudian ada pesan List before sorting lalu menampilkan data yang ada di linked list kemudia memanggil fungsi sort dan melakukan sorting kemudian menampilkan data yang terlah di sorting.

SS OUTPUT

```
PS C:\VsCode\C 2\output> & .\'OTH_3.exe'
Enter number of elements: 5
Enter element 1: 5
Enter element 2: 3
Enter element 3: 8
Enter element 4: 1
Enter element 5: 6

List before sorting:
Address: 00D30CA8, Data: 6
Address: 00D30C90, Data: 1
Address: 00D30C78, Data: 8
Address: 00D30C60, Data: 3
Address: 00D32F80, Data: 5

List after sorting:
Address: 00D30C90, Data: 1
Address: 00D30C60, Data: 3
Address: 00D32F80, Data: 5
Address: 00D30CA8, Data: 6
Address: 00D30C78, Data: 8
PS C:\VsCode\C 2\output>
```

```
PS C:\VsCode\C 2\output> & .\'OTH_3.exe'
Enter number of elements: 3
Enter element 1: 31
Enter element 2: 2
Enter element 3: 123

List before sorting:
Address: 00900C78, Data: 123
Address: 00900C60, Data: 2
Address: 00902F80, Data: 31

List after sorting:
Address: 00900C60, Data: 2
Address: 00902F80, Data: 31
Address: 00900C78, Data: 123
PS C:\VsCode\C 2\output>
```