Nama: Yonathan Hari Dharmawan

Kelas: IF 03-03

NIM: 1203230050

1. Source Code

#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;

    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("%p %d\n", (void\*)temp, temp->data);

        temp = temp->next;

    } while (temp != head);

}

void swapNodes(Node\*\* head, Node\* a, Node\* b) {

    if (a == b) return;

    Node\* aPrev = a->prev;

    Node\* aNext = a->next;

    Node\* bPrev = b->prev;

    Node\* bNext = b->next;

    if (aNext == b) {

        a->next = bNext;

        b->prev = aPrev;

        a->prev = b;

        b->next = a;

        aPrev->next = b;

        bNext->prev = a;

    } else if (bNext == a) {

        b->next = aNext;

        a->prev = bPrev;

        b->prev = a;

        a->next = b;

        bPrev->next = a;

        aNext->prev = b;

    } else {

        a->next = bNext;

        a->prev = bPrev;

        b->next = aNext;

        b->prev = aPrev;

        aPrev->next = b;

        aNext->prev = b;

        bPrev->next = a;

        bNext->prev = a;

    }

    if (\*head == a) {

        \*head = b;

    } else if (\*head == b) {

        \*head = a;

    }

}

void sortList(Node\*\* head) {

    if (\*head == NULL) return;

    Node\* current = \*head;

    Node\* index = NULL;

    int swapped;

    do {

        swapped = 0;

        current = \*head;

        while (current->next != \*head) {

            index = current->next;

            if (current->data > index->data) {

                swapNodes(head, current, index);

                swapped = 1;

            } else {

                current = current->next;

            }

        }

    } while (swapped);

}

int main() {

    Node\* head = NULL;

    int N, A;

    printf("Berapa data yang ingin diinput: ");

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {

        scanf("%d", &A);

        insertEnd(&head, A);

    }

    printf("Data sebelum diurutkan:\n");

    printList(head);

    sortList(&head);

    printf("Data setelah diurutkan:\n");

    printList(head);

    return 0;

}

#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;

    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("%p %d\n", (void\*)temp, temp->data);

        temp = temp->next;

    } while (temp != head);

}

void swapNodes(Node\*\* head, Node\* a, Node\* b) {

    if (a == b) return;

    Node\* aPrev = a->prev;

    Node\* aNext = a->next;

    Node\* bPrev = b->prev;

    Node\* bNext = b->next;

    if (aNext == b) {

        a->next = bNext;

        b->prev = aPrev;

        a->prev = b;

        b->next = a;

        aPrev->next = b;

        bNext->prev = a;

    } else if (bNext == a) {

        b->next = aNext;

        a->prev = bPrev;

        b->prev = a;

        a->next = b;

        bPrev->next = a;

        aNext->prev = b;

    } else {

        a->next = bNext;

        a->prev = bPrev;

        b->next = aNext;

        b->prev = aPrev;

        aPrev->next = b;

        aNext->prev = b;

        bPrev->next = a;

        bNext->prev = a;

    }

    if (\*head == a) {

        \*head = b;

    } else if (\*head == b) {

        \*head = a;

    }

}

void sortList(Node\*\* head) {

    if (\*head == NULL) return;

    Node\* current = \*head;

    Node\* index = NULL;

    int swapped;

    do {

        swapped = 0;

        current = \*head;

        while (current->next != \*head) {

            index = current->next;

            if (current->data > index->data) {

                swapNodes(head, current, index);

                swapped = 1;

            } else {

                current = current->next;

            }

        }

    } while (swapped);

}

int main() {

    Node\* head = NULL;

    int N, A;

    printf("Berapa data yang ingin diinput: ");

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {

        scanf("%d", &A);

        insertEnd(&head, A);

    }

    printf("Data sebelum diurutkan:\n");

    printList(head);

    sortList(&head);

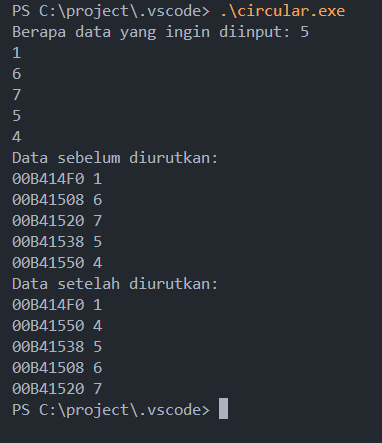
    printf("Data setelah diurutkan:\n");

    printList(head);

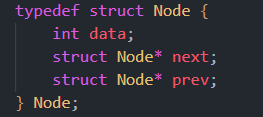
    return 0;

}

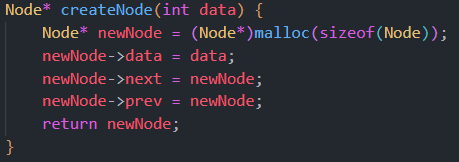
1. OUTPUT



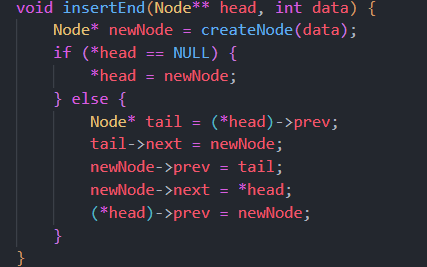
1. Penjelasan



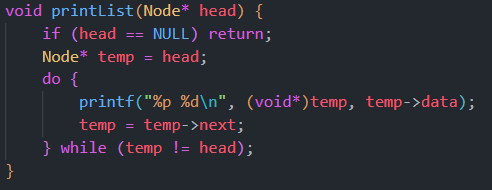
* Struktur Node merepresentasikan sebuah node pada linked list doubly circular.
* data menyimpan nilai data.
* next adalah pointer ke node berikutnya.
* prev adalah pointer ke node sebelumnya.



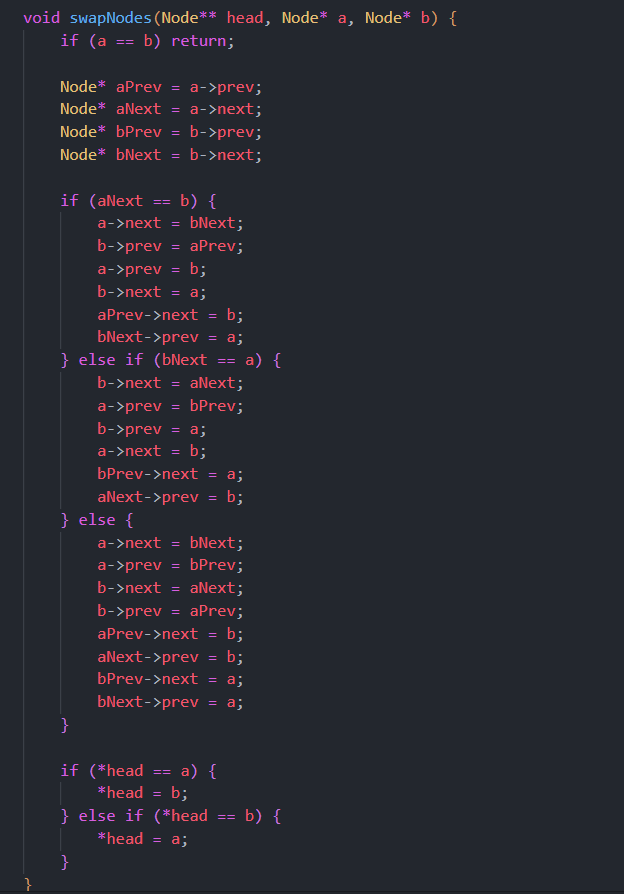
* Fungsi ini membuat node baru dengan nilai **data**.
* Mengalokasikan memori untuk node baru dan menginisialisasi pointer **next** dan **prev** untuk menunjuk ke dirinya sendiri (karena circular).



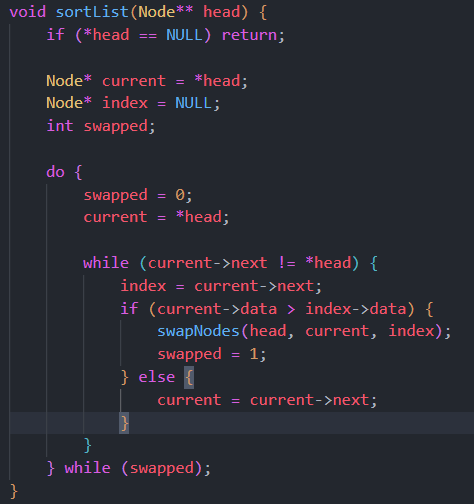
* Fungsi ini menambahkan node baru di akhir linked list.
* Jika list kosong, node baru menjadi head.
* Jika tidak, node baru dihubungkan di akhir list dan tautan diperbarui untuk mempertahankan circular doubly linked list.



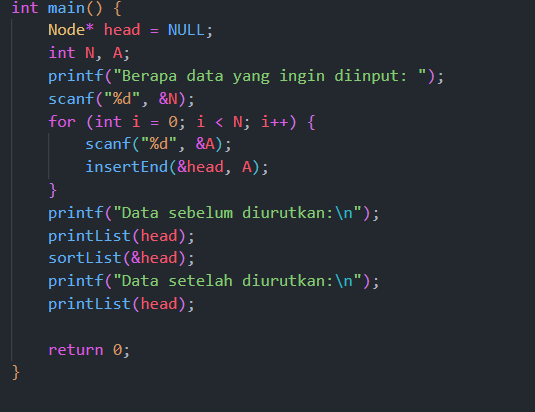
* Fungsi ini mencetak semua elemen dalam list.
* Mengiterasi dari **head** hingga kembali ke **head** lagi karena list bersifat circular.



* **Fungsi ini menukar posisi dua node a dan b dalam list.**
* **Menangani tiga kasus: node berdekatan dan a sebelum b, node berdekatan dan b sebelum a, atau node yang tidak berdekatan.**

****

* **Fungsi ini mengurutkan list menggunakan metode Bubble Sort.**
* **Mengiterasi list dan menukar node yang tidak dalam urutan yang benar hingga tidak ada lagi pertukaran yang terjadi.**

****

* **Fungsi main mengelola input pengguna untuk memasukkan jumlah dan nilai data.**
* **Menggunakan fungsi insertEnd untuk menambahkan data ke list.**
* **Mencetak data sebelum diurutkan, mengurutkan data, lalu mencetak data setelah diurutkan.**