LAPORAN PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

MATERI: CIRCULAR DOUBLE LINKED LIST

Oleh:

Shinta Putri Nirmala

NIM: 1203230052

Kelas: IF 03-01

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM SURABAYA

TAHUN 2024

SOAL

Deskripsi Tugas

Andi baru saja belajar tentang struktur data sirkular double linked list. Dia ingin melatih kemampuannya dengan membuat sebuah program untuk mengelola data dalam struktur tersebut.

Andi ingin membuat sebuah program C yang akan meminta pengguna untuk memasukkan sejumlah data ke dalam sirkular double linked list. Setelah itu, program akan mengurutkan list tersebut secara ascending dengan syarat "***Never change the data. Change the position of the nodes***." Artinya, Anda dilarang **mengubah/memanipulasi data** pada setiap node, namun Anda dapat **mengubah posisi** dari node-node tersebut.

Format Masukan

- Baris pertama berisi sebuah bilangan bulat N $(1 \leq N \leq 10)$, menyatakan jumlah data yang akan dimasukkan ke dalam list.
- N baris berikutnya berisi sebuah bilangan bulat A_i $(1 \leq A_i \leq 10)$, menyatakan data yang dimasukkan ke dalam list.

Format Keluaran

- Tampilkan list (memory addres & data) sebelum pengurutan.
- Tampilkan list (memory addres & data) setelah pengurutan.

PENJELASAN SOURCE CODE

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

// Mendefinisikan struktur 'alamat' yang akan digunakan sebagai node dalam linked list
struct alamat {
    int data;             // Menyimpan data dalam node
    struct alamat *prev;  // Pointer ke node sebelumnya dalam linked list
    struct alamat *next;  // Pointer ke node berikutnya dalam linked list
};
//bagian ini hanya mendefinisikan struktur alamat yang memiliki tiga anggota: data, prev, dan next. Data digunakan untuk menyimpan nilai integer, prev dan next digunakan untuk menunjuk ke node sebelumnya dan berikutnya dalam linked list. Fungsi untuk membuat node baru dengan data yang diberikan
struct alamat* createNode(int data) {
    // Mengalokasikan memori untuk node baru menggunakan malloc
    struct alamat *newNode = (struct alamat*)malloc(sizeof(struct alamat));
    // Mengatur data node sesuai dengan parameter yang diberikan
```

```
    newNode->data = data;
    // Mengatur pointer prev dan next ke NULL karena node baru belum ditautkan
ke linked list manapun
    newNode->prev = NULL;
    newNode->next = NULL;
    // Mengembalikan pointer ke node baru yang telah dibuat
    return newNode;
}
//Fungsi ini membuat dan menginisialisasi node baru dengan data yang
diberikan. Memori dialokasikan menggunakan malloc, dan kemudian nilai data
diatur sesuai dengan parameter. Pointer prev dan next diatur ke NULL karena
node baru belum ditautkan ke linked list manapun.\
// Fungsi untuk memasukkan node baru ke dalam linked list
void insertNode(struct alamat **head, int data) {
    // Membuat node baru dengan menggunakan fungsi createNode
    struct alamat *newNode = createNode(data);
    // Jika linked list masih kosong (head == NULL)
    if (!*head) {
        // Mengatur pointer next dan prev pada node baru agar menunjuk ke
dirinya sendiri
        newNode->next = newNode;
        newNode->prev = newNode;
        // Mengatur head untuk menunjuk ke node baru yang telah dibuat
        *head = newNode;
    } else {
        // Jika linked list tidak kosong
        // Mencari node terakhir dalam linked list
        struct alamat *tail = (*head)->prev;
        // Menghubungkan node terakhir dengan node baru
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}
//Fungsi ini memasukkan node baru ke dalam linked list. Jika linked list masih
kosong, node baru akan menjadi satu-satunya node dan akan menunjuk ke dirinya
sendiri. Jika tidak, node baru akan ditambahkan di belakang (atau sebelum head
jika linked list merupakan linked list berputar).
// Fungsi untuk menampilkan isi linked list
void displayList(struct alamat *head) {
    // Jika linked list kosong, tidak perlu menampilkan apapun
    if (!head) return;
    // Pointer sementara untuk melintasi linked list
    struct alamat *temp = head;
    // Melakukan perulangan untuk menampilkan setiap node dalam linked list
    do {
        printf("Address: %p, Data: %d\n", (void*)temp, temp->data);
```

```
        // Menggeser pointer temp ke node berikutnya dalam linked list
        temp = temp->next;
    } while (temp != head);  // Melakukan perulangan sampai kembali ke head
(linked list berbentuk circular)
}
//Fungsi ini digunakan untuk menampilkan seluruh isi dari linked list.
Menggunakan loop do-while untuk mencetak setiap node, dan looping akan
berhenti saat kita kembali ke head, yang menandakan akhir dari linked list.
// Fungsi untuk menukar posisi dua node dalam linked list
void swapNodes(struct alamat **head, struct alamat *node1, struct alamat
*node2) {
    // Jika node1 dan node2 sama, tidak perlu melakukan pertukaran
    if (node1 == node2) return;

    // Mendeklarasikan variabel lokal untuk menyimpan pointer sebelum dan
sesudah node1 dan node2
    struct alamat *prev1 = node1->prev;
    struct alamat *next1 = node1->next;
    struct alamat *prev2 = node2->prev;
    struct alamat *next2 = node2->next;

    // Melakukan pengecekan untuk menentukan posisi relatif antara node1 dan
node2
    if (node1->next == node2) {
        // Jika node2 berada setelah node1
        // Mengatur pointer next dan prev pada node1 dan node2 untuk
menyesuaikan pertukaran
        node1->next = next2;
        node1->prev = node2;
        node2->next = node1;
        node2->prev = prev1;
        // Mengatur pointer next dan prev pada node-node sebelum dan sesudah
node1 dan node2 agar menunjuk ke node yang baru ditukar
        if (next2) next2->prev = node1;
        if (prev1) prev1->next = node2;

    } else if (node2->next == node1) {
        // Jika node1 berada setelah node2
        // Mengatur pointer next dan prev pada node1 dan node2 untuk
menyesuaikan pertukaran
        node2->next = next1;
        node2->prev = node1;
        node1->next = node2;
        node1->prev = prev2;
        // Mengatur pointer next dan prev pada node-node sebelum dan sesudah
node1 dan node2 agar menunjuk ke node yang baru ditukar
        if (next1) next1->prev = node2;
        if (prev2) prev2->next = node1;
```

```
    } else {
        // Jika node1 dan node2 tidak berdampingan
        // Menghubungkan node-node sebelum dan sesudah node1 dan node2 dengan
node-node yang baru ditukar
        if (prev1) prev1->next = node2;
        if (next1) next1->prev = node2;
        if (prev2) prev2->next = node1;
        if (next2) next2->prev = node1;
        // Mengatur pointer next dan prev pada node1 dan node2 untuk
menyesuaikan pertukaran
        node1->next = next2;
        node1->prev = prev2;
        node2->next = next1;
        node2->prev = prev1;
    }

    // Memperbarui head jika node1 atau node2 adalah head
    if (*head == node1) *head = node2;
    else if (*head == node2) *head = node1;
}
//Fungsi ini bertujuan untuk menukar posisi dua node dalam linked list.
Pertama, diperiksa apakah node1 dan node2 sama. Jika iya, tidak perlu
melakukan apa-apa. Jika tidak, dilakukan penyesuaian pointer next dan prev di
antara node-node yang bersangkutan.
// Fungsi untuk mengurutkan linked list
void sortList(struct alamat **head) {
    // Jika linked list kosong atau hanya memiliki satu node, tidak perlu
dilakukan pengurutan
    if (!*head || (*head)->next == *head) return;

    // Mendeklarasikan variabel lokal untuk melakukan pengurutan
    struct alamat *i = *head;
    struct alamat *j = NULL;
    int swapped;

    // Melakukan pengurutan dengan algoritma Bubble Sort
    do {
        swapped = 0;
        j = i->next;
        // Melakukan perulangan untuk membandingkan setiap pasangan node dalam
linked list
        while (j != *head) {
            // Jika data pada node ke-i lebih besar dari data pada node ke-j,
maka dilakukan pertukaran
            if (i->data > j->data) {
                swapNodes(head, i, j);  // Memanggil fungsi swapNodes untuk
pertukaran
```

```
                swapped = 1;  // Menandakan bahwa ada pertukaran yang terjadi
            }
            j = j->next;  // Menuju ke node berikutnya untuk dibandingkan
        }
        i = i->next;  // Menuju ke node berikutnya untuk iterasi berikutnya
    } while (swapped);  // Melakukan iterasi hingga tidak ada pertukaran yang
terjadi, menunjukkan bahwa linked list sudah terurut
}
//Fungsi ini bertanggung jawab untuk mengurutkan linked list. Algoritma
pengurutan yang digunakan adalah Bubble Sort. Iterasi dilakukan hingga tidak
ada pertukaran yang terjadi, menunjukkan bahwa linked list sudah terurut.
// Fungsi utama
int main() {
    int N, data;
    struct alamat *head = NULL;

    // Membaca jumlah data yang akan dimasukkan
    scanf("%d", &N);

    // Memasukkan data ke dalam linked list
    for (int i = 0; i < N; i++) {
        scanf("%d", &data);
        insertNode(&head, data);  // Memanggil fungsi insertNode untuk
memasukkan data
    }

    // Menampilkan isi linked list sebelum pengurutan
    displayList(head);

    printf("\n");

    // Mengurutkan linked list
    sortList(&head);

    // Menampilkan isi linked list setelah pengurutan
    displayList(head);

    // Mengosongkan memori dengan menghapus setiap node dalam linked list
    struct alamat *current = head;
    struct alamat *nextNode;

    do {
        nextNode = current->next;
        free(current);
        current = nextNode;
    } while (current != head);

    return 0;
```

```
}
//Fungsi utama program. Pertama, membaca jumlah data yang akan dimasukkan, kemudian memasukkan data tersebut ke dalam linked list. Setelah itu, menampilkan isi linked list sebelum pengurutan. Kemudian, mengurutkan linked list menggunakan fungsi sortList, dan menampilkan isi linked list setelah pengurutan. Terakhir, mengosongkan memori dengan menghapus setiap node dalam linked list menggunakan free.
```

OUTPUT

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode> cd "c:\Users\User\D
SCode\PRAKTIKUM 9\" ; if ($?) { gcc latihan.c -o latihan } ; if ($?) { .\latihan }
3
31
2
123
Address: 0000023AF64CB260, Data: 31
Address: 0000023AF64CB280, Data: 2
Address: 0000023AF64CB320, Data: 123

Address: 0000023AF64CB280, Data: 2
Address: 0000023AF64CB260, Data: 31
Address: 0000023AF64CB320, Data: 123
PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode\PRAKTIKUM 9>
```