

OTH PRAKTIKUM

Nama : Nabila Amilatul Jannah

Kelas : IF 03 02

NIM : 1203230103

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

typedef struct ListNode {
    int value;
    struct ListNode *nextNode;
    struct ListNode *prevNode;
} ListNode;

ListNode *listHead = NULL;
ListNode *listTail = NULL;

void insertListNode(int value) {
    ListNode *newNode = (ListNode *)malloc(sizeof(ListNode));
    newNode->value = value;

    if (listHead == NULL) {
        listHead = listTail = newNode;
        newNode->nextNode = newNode->prevNode = newNode;
    } else {
        listTail->nextNode = newNode;
        newNode->prevNode = listTail;
        newNode->nextNode = listHead;
        listHead->prevNode = newNode;
        listTail = newNode;
    }
}

void printList() {
    if (listHead == NULL) {
        printf("List is empty\n");
        return;
    }
}
```

```

    ListNode *currentNode = listHead;
    do {
        printf("Address: %016lx, Data: %d\n", (unsigned long)currentNode,
currentNode->value);
        currentNode = currentNode->nextNode;
    } while (currentNode != listHead);
}

void swapListNodes(ListNode *nodeA, ListNode *nodeB) {
    ListNode *aPrev = nodeA->prevNode, *aNext = nodeA->nextNode;
    ListNode *bPrev = nodeB->prevNode, *bNext = nodeB->nextNode;

    if (nodeA->nextNode == nodeB) { // nodeA and nodeB are adjacent
        aPrev->nextNode = nodeB;
        nodeB->prevNode = aPrev;
        nodeB->nextNode = nodeA;
        nodeA->prevNode = nodeB;
        nodeA->nextNode = bNext;
        bNext->prevNode = nodeA;
    } else {
        aPrev->nextNode = nodeB;
        aNext->prevNode = nodeB;
        bPrev->nextNode = nodeA;
        bNext->prevNode = nodeA;

        nodeA->nextNode = bNext;
        nodeA->prevNode = bPrev;
        nodeB->nextNode = aNext;
        nodeB->prevNode = aPrev;
    }

    if (listHead == nodeA) listHead = nodeB;
    else if (listHead == nodeB) listHead = nodeA;

    if (listTail == nodeA) listTail = nodeB;
    else if (listTail == nodeB) listTail = nodeA;
}

void sortList() {
    if (listHead == NULL) return;

```

```

int swapped;
ListNode *currentNode;

do {
    swapped = 0;
    currentNode = listHead;

    do {
        ListNode *nextNode = currentNode->nextNode;
        if (currentNode->value > nextNode->value) {
            swapListNodes(currentNode, nextNode);
            swapped = 1;
        } else {
            currentNode = nextNode;
        }
    } while (currentNode != listTail);
} while (swapped);
}

int main() {
    int numData;
    printf("Masukkan jumlah data: ");
    scanf("%d", &numData);

    for (int i = 0; i < numData; i++) {
        int data;
        printf("Masukkan data ke- %d: ", i + 1);
        scanf("%d", &data);
        insertListNode(data);
    }

    printf("\nList sebelum sorting:\n");
    printList();

    sortList();

    printf("\nList setelah sorting:\n");
    printList();

    return 0;
}

```

```
}
```

Penjelasan Kode

```
typedef struct ListNode {  
    int value;  
    struct ListNode *nextNode;  
    struct ListNode *prevNode;  
} ListNode;
```

Int value; Menyimpan nilai int untuk node tersebut. Struct ListNode *nextNode; Pointer menunjuk ke node berikutnya dalam list. Struct ListNode *prevNode; Pointer menunjuk ke node sebelumnya .

```
ListNode *listHead = NULL;  
ListNode *listTail = NULL;
```

Pointer yang menunjuk ke node pertama (head) dan node terakhir (tail) dari linked list. Pada saat inisialisasi di set ke NULL bahwa menandakan list kosong.

```
void insertListNode(int value) {  
    ListNode *newNode = (ListNode *)malloc(sizeof(ListNode));  
    newNode->value = value;
```

Insert List bertujuan untuk menyisipkan node baru ke dalam linked list. ListNode adalah struktur yang mempresentasikan simpul dalam linked list. Setiap simpul memiliki nilai dan pointer ke simpul sebelumnya dan sesudahnya. Malloc bertujuan untuk mengalokasikan ukuran memori sebesar 'sizeof(ListNode)'.

```
if (listHead == NULL) {  
    listHead = listTail = newNode;  
    newNode->nextNode = newNode->prevNode = newNode
```

Mengecek list apakah list kosong (ditandai dengan 'listhead' yang bernilai NULL.

Jika kosong, 'listHead' dan 'listTail' diset ke 'newNode' kemudian menyambungkan 'nextNode' dan 'prevNode' dari 'newNode' ke dirinya sendiri, membentuk circular list dengan satu node.

```
} else {  
    listTail->nextNode = newNode;  
    newNode->prevNode = listTail;  
    newNode->nextNode = listHead;  
    listHead->prevNode = newNode;  
    listTail = newNode;
```

Apabila list tidak kosong, listTail->nextNode = newNode; : Menghubungkan node terakhir (listTail) ke node baru (newNode).

newNode->prevNode = listTail; : Menetapkan node sebelumnya dari newNode ke listTail.

`newNode->nextNode = listHead;` : Menghubungkan node baru (`newNode`) ke `listHead`,
`listHead->prevNode = newNode;` : Menetapkan node sebelumnya dari `listHead` ke node baru (`newNode`).

`listTail = newNode;` : Memperbarui `listTail` sehingga menunjuk ke node baru, yang sekarang menjadi node terakhir dalam list.

```
void printList() {  
    if (listHead == NULL) {  
        printf("List is empty\n");  
        return;  
    }  
}
```

`if (listHead == NULL) {` : Mengecek apakah `listHead` adalah `NULL`, yang menandakan bahwa linked list kosong.

`printf("List is empty\n");` : Jika linked list kosong, maka menampilkan "List is empty".

```
ListNode *currentNode = listHead;  
do {  
    printf("Address: %016lx, Data: %d\n", (unsigned long)currentNode,  
currentNode->value);  
    currentNode = currentNode->nextNode;  
} while (currentNode != listHead);  
}
```

`ListNode *currentNode = listHead;` : Membuat pointer `currentNode` yang diinisialisasi dengan `listHead`, menunjuk ke node pertama dalam linked list

Struktur `do {.....} while` digunakan bahwa setidaknya satu kali iterasi terjadi, bahkan jika list hanya memiliki satu node.

`printf("Address: %016lx, Data: %d\n", (unsigned long)currentNode, currentNode->value);` : Cetak alamat memori (address) dari `currentNode` dan nilai data (value) yang disimpan di node tersebut. Format `%016lx` digunakan untuk mencetak alamat memori dalam format heksadesimal 16 digit.

`currentNode = currentNode->nextNode;` : Pindahkan `currentNode` ke node berikutnya dalam list dengan mengakses `nextNode`

```
void swapListNodes(ListNode *nodeA, ListNode *nodeB) {  
    ListNode *aPrev = nodeA->prevNode, *aNext = nodeA->nextNode;  
    ListNode *bPrev = nodeB->prevNode, *bNext = nodeB->nextNode;
```

`ListNode *aPrev = nodeA->prevNode, *aNext = nodeA->nextNode;` : Menyimpan pointer ke node sebelumnya (`prevNode`) dan node berikutnya (`nextNode`) dari `nodeA`.

`ListNode *bPrev = nodeB->prevNode, *bNext = nodeB->nextNode;` : Menyimpan pointer ke node sebelumnya (`prevNode`) dan node berikutnya (`nextNode`) dari `nodeB`.

Pertukaran Node yang berdekatan

```
if (nodeA->nextNode == nodeB) { // nodeA and nodeB are adjacent
    aPrev->nextNode = nodeB;
    nodeB->prevNode = aPrev;
    nodeB->nextNode = nodeA;
    nodeA->prevNode = nodeB;
    nodeA->nextNode = bNext;
    bNext->prevNode = nodeA;
```

Pertukaran Node yang tidak berdekatan

```
} else {
    aPrev->nextNode = nodeB;
    aNext->prevNode = nodeB;
    bPrev->nextNode = nodeA;
    bNext->prevNode = nodeA;

    nodeA->nextNode = bNext;
    nodeA->prevNode = bPrev;
    nodeB->nextNode = aNext;
    nodeB->prevNode = aPrev;
}
```

```
if (listHead == nodeA) listHead = nodeB;
else if (listHead == nodeB) listHead = nodeA;
```

Memperbarui list head, if (listHead == nodeA) listHead = nodeB; : Jika nodeA adalah listHead, perbarui listHead ke nodeB.

else if (listHead == nodeB) listHead = nodeA; : Jika nodeB adalah listHead, perbarui listHead ke nodeA.

Memperbarui list tail, if (listTail == nodeA) listTail = nodeB; : Jika nodeA adalah listTail, perbarui listTail ke nodeB.

else if (listTail == nodeB) listTail = nodeA; : Jika nodeB adalah listTail, perbarui listTail ke nodeA.

```
void sortList() {
    if (listHead == NULL) return;

    int swapped;
    ListNode *currentNode;
```

int swapped; : Variabel flag (swapped) untuk melacak apakah ada elemen yang ditukar selama iterasi, menandakan apakah perlu iterasi lebih lanjut.

ListNode *currentNode; : Pointer (currentNode) untuk mengiterasi melalui list.

```
do {  
    swapped = 0;  
    currentNode = listHead;
```

do { ... } while (swapped); : Loop luar memastikan bahwa iterasi dilakukan berulang kali hingga tidak ada lagi elemen yang ditukar (swapped tetap 0 setelah satu iterasi lengkap).

swapped = 0; : Mengatur swapped menjadi 0 sebelum memulai iterasi baru.

currentNode = listHead; : Menginisialisasi currentNode ke listHead pada awal setiap iterasi luar.

```
do {  
    ListNode *nextNode = currentNode->nextNode;  
    if (currentNode->value > nextNode->value) {
```

do { ... } while (currentNode != listTail); : Loop dalam mengiterasi setiap node dalam list hingga mencapai listTail.

ListNode *nextNode = currentNode->nextNode; : Mengambil node berikutnya (nextNode) untuk dibandingkan dengan currentNode.

if (currentNode->value > nextNode->value) { ... } : Membandingkan nilai dari currentNode dan nextNode.

```
        swapListNodes(currentNode, nextNode);  
        swapped = 1;  
    } else {  
        currentNode = nextNode;  
    }  
} while (currentNode != listTail);  
} while (swapped);  
}
```

swapListNodes(currentNode, nextNode); : Menukar posisi currentNode dan nextNode jika nilai currentNode lebih besar daripada nextNode.

swapped = 1; : Mengatur swapped menjadi 1 untuk menandakan bahwa terjadi pertukaran dalam iterasi ini.

```
int main() {  
    int numData;  
    printf("Masukkan jumlah data: ");  
    scanf("%d", &numData);
```

Menampilkan pesan untuk pengguna memasukan jumlah data dan membaca input pengguna 'scanf' dan menyimpannya ke dalam variable 'numData'

```
for (int i = 0; i < numData; i++) {  
    int data;  
    printf("Masukkan data ke- %d: ", i + 1);  
    scanf("%d", &data);  
    insertListNode(data);  
}
```

for (int i = 0; i < numData; i++) { ... } : Loop for untuk mengulangi proses input sebanyak numData kali.

printf("Masukkan data ke- %d: ", i + 1); : Menampilkan pesan kepada pengguna untuk memasukkan data ke-i+1.

scanf("%d", &data); : Membaca input pengguna dan menyimpannya ke dalam variabel data.

insertListNode(data); : Memanggil fungsi insertListNode untuk menyisipkan data ke dalam linked list.

```
printf("\nList sebelum sorting:\n");  
printList();  
  
sortList();  
  
printf("\nList setelah sorting:\n");  
printList();
```

printf("\nList sebelum sorting:\n"); : Menampilkan pesan bahwa list akan dicetak sebelum pengurutan.

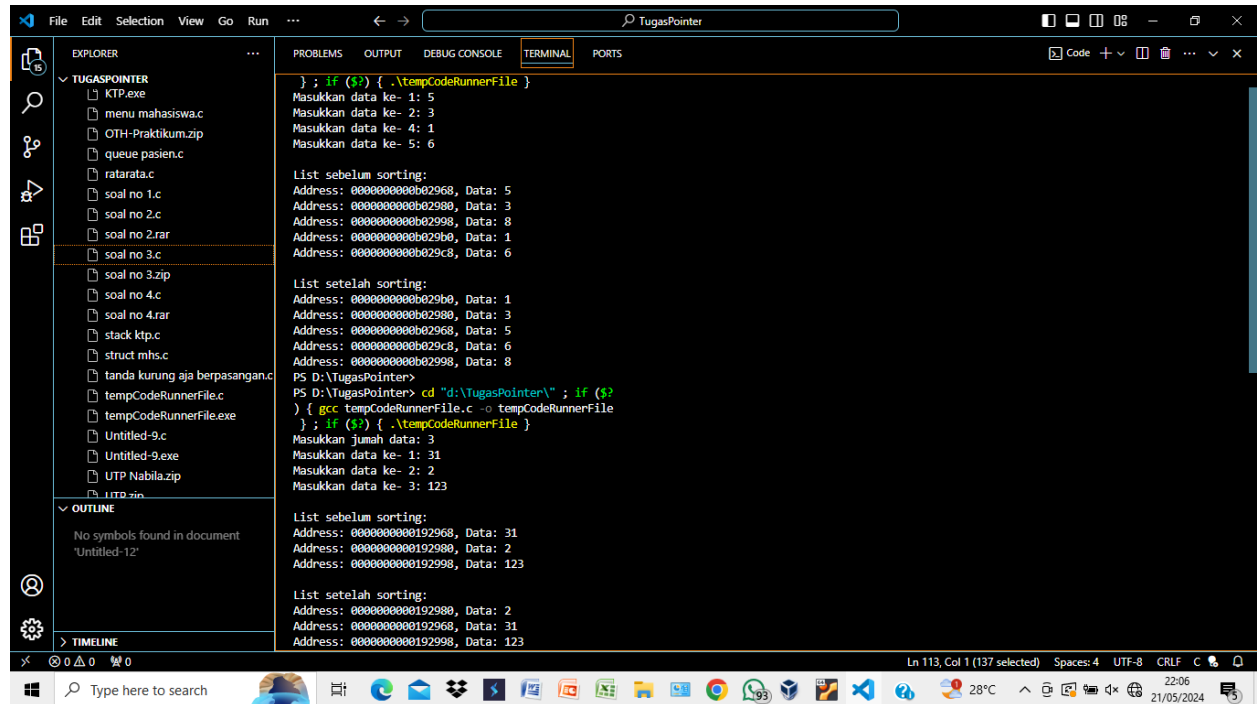
printList(); : Memanggil fungsi printList untuk mencetak isi linked list sebelum pengurutan.

sortList(); : Memanggil fungsi sortList untuk mengurutkan isi linked list

printf("\nList setelah sorting:\n"); : Menampilkan pesan bahwa list akan dicetak setelah pengurutan.

printList(); : Memanggil fungsi printList untuk mencetak isi linked list setelah pengurutan.

OUTPUT



The screenshot shows the Visual Studio Code interface with the 'TugasPointer' project open. The Explorer pane on the left lists files under 'TUGASPOINTER', including 'KTP.exe', 'menu mahasiswa.c', 'OTH-Praktikum.zip', 'queue pasien.c', 'ratarata.c', 'soal no 1.c', 'soal no 2.c', 'soal no 2.rar', 'soal no 3.c', 'soal no 3.zip', 'soal no 4.c', 'soal no 4.rar', 'stack ktp.c', 'struct mhs.c', 'tanda kurung aja berpasangan.c', 'tempCodeRunnerFile.c', 'tempCodeRunnerFile.exe', 'Untitled-9.c', 'Untitled-9.exe', 'UTP Nabila.zip', and 'UTP.zip'. The Outline pane shows 'No symbols found in document' for 'Untitled-12'. The Terminal pane displays the output of a C program execution, showing memory addresses and data values before and after sorting.

```
}; if ($?) { .\tempCodeRunnerFile }  
Masukkan data ke- 1: 5  
Masukkan data ke- 2: 3  
Masukkan data ke- 4: 1  
Masukkan data ke- 5: 6  
  
List sebelum sorting:  
Address: 000000000b02968, Data: 5  
Address: 000000000b02980, Data: 3  
Address: 000000000b02998, Data: 8  
Address: 000000000b029b0, Data: 1  
Address: 000000000b029c8, Data: 6  
  
List setelah sorting:  
Address: 000000000b029b0, Data: 1  
Address: 000000000b02980, Data: 3  
Address: 000000000b02968, Data: 5  
Address: 000000000b029c8, Data: 6  
Address: 000000000b02998, Data: 8  
PS D:\TugasPointer>  
PS D:\TugasPointer> cd "d:\TugasPointer\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }  
Masukkan jumlah data: 3  
Masukkan data ke- 1: 31  
Masukkan data ke- 2: 2  
Masukkan data ke- 3: 123  
  
List sebelum sorting:  
Address: 000000000192968, Data: 31  
Address: 000000000192980, Data: 2  
Address: 000000000192998, Data: 123  
  
List setelah sorting:  
Address: 000000000192980, Data: 2  
Address: 000000000192968, Data: 31  
Address: 000000000192998, Data: 123
```

Ln 113, Col 1 (137 selected) Spaces: 4 UTF-8 CRLF 22:06 21/05/2024