

TUGAS PRAKTIKUM ALGORITMA & STRUKTUR DATA

Jilid 4



Oleh :

Nama : Rosi Arif Mulyadi

NRP : 3121522021

Prodi : D3 Teknik Informatika PENS PSDKU Sumenep

Kelas : 1 ITA D3 Sumenep

Dosen :

LUSIANA AGUSTIEN M.Kom

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

MODUL 2.1

LINKED LIST (SINGLE LINKED LIST – Input elemen dari beberapa posisi)

B. Kegiatan Praktikum

1. Implementasikan dan tentukan output percobaan yang ada dalam modul praktikum ini dan lakukan analisa pada tiap fungsi yang dibuat.

Jawab :

Listing Program :

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

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

```
struct node
```

```
{
```

```
    int val;
```

```
    struct node *next;
```

```
};
```

```
void print_list(struct node *head)
```

```
{
```

```
    printf("H->");
```

```
    while(head)
```

```
    {
```

```
        printf("%d->", head->val);
```

```
        head = head->next;
```

```
    }
```

```
    printf("|||\n\n");
```

```
}
```

```
void insert_front(struct node **head, int value)
```

```
{
```

```
    struct node * new_node = NULL;
```

```
    new_node = (struct node *)malloc(sizeof(struct node));
```

```
    if(new_node == NULL)
```

```

{
    printf("tidak bisa menginputkan node baru, memori penuh");
}

new_node->val = value;
new_node->next = *head;

*head = new_node;
}

```

```

void insert_end(struct node **head, int value)
{
    struct node * new_node = NULL;
    struct node * last = NULL;

    new_node = (struct node *)malloc(sizeof(struct node));

    if(new_node == NULL)
    {
        printf("tidak bisa menginputkan node baru, memori penuh");
    }

    new_node->val = value;
    new_node->next = NULL;

    if( *head == NULL)
    {
        *head = new_node;
        return;
    }

    last = *head;
    while(last->next) last = last->next;

    last->next = new_node;
}

```

```

void insert_after(struct node *head, int value, int after)

```

```

{
    struct node * new_node = NULL;
    struct node *tmp = head;

    while(tmp)
    {
        if(tmp->val == after)
        {
            new_node = (struct node *)malloc(sizeof(struct node));

            if(new_node == NULL)
            {
                printf("tidak bisa menginputkan node baru, memori penuh");
            }

            new_node->val = value;
            new_node->next = tmp->next;
            tmp->next = new_node;
            return;
        }

        tmp = tmp->next;
    }
}

```

```

void insert_before(struct node **head, int value, int before)
{
    struct node * new_node = NULL;
    struct node * tmp = *head;

    new_node = (struct node *)malloc(sizeof(struct node));

    if(new_node == NULL)
    {
        printf("tidak bisa menginputkan node baru, memori penuh");
        return;
    }
}

```

```

new_node->val = value;

if((*head)->val == before)
{
    new_node->next = *head;
    *head = new_node;
    return;
}

while(tmp && tmp->next)
{
    if(tmp->next->val == before)
    {
        new_node->next = tmp->next;
        tmp->next = new_node;
        return;
    }

    tmp = tmp->next;
}

free(new_node);
}

void main()
{
    int count = 0, i, val, after, before;
    struct node * head = NULL;

    printf("Inputkan jumlah elemen/node yang akan di inputkan : ");
    scanf("%d", &count);

    for(i = 0; i < count; i++)
    {
        printf("Enter %d th elemen/node : ", i);
        scanf("%d", &val);
        insert_front(&head, val);
    }
}

```

```
printf("Initial List : ");  
print_list(head);
```

```
printf("Inputkan nilai untuk di sisipkan pada awal list : ");  
scanf("%d", &val);  
insert_front(&head, val);
```

```
printf("List setelah input data baru : ");  
print_list(head);
```

```
printf("Inputkan nilai untuk di sisipkan pada akhir list : ");  
scanf("%d", &val);  
insert_end(&head, val);
```

```
printf("List setelah input data baru : ");  
print_list(head);
```

```
printf("Inputkan nilai yang akan di sisipkan pada list : ");  
scanf("%d", &val);
```

```
printf("Di sisipkan setelah ? : ");  
scanf("%d", &after);
```

```
insert_after(head, val, after);
```

```
printf("List setelah input data baru : ");  
print_list(head);
```

```
printf("Inputkan nilai yang akan di sisipkan pada list : ");  
scanf("%d", &val);
```

```
printf("Di sisipkan sebelum ? : ");  
scanf("%d", &before);
```

```
insert_before(&head, val, before);
```

```
printf("List setelah input data baru : ");
```

```

    print_list(head);
}

```

Output :

```

"D:\New Linked List 4\bin\Debug\New Linked List 4.exe"
Inputkan jumlah elemen/node yang akan di inputkan : 4
Enter 0 th elemen/node : 10
Enter 1 th elemen/node : 20
Enter 2 th elemen/node : 30
Enter 3 th elemen/node : 40
Initial List : H->40->30->20->10->|||

Inputkan nilai untuk di sisipkan pada awal list : 0
List setelah input data baru : H->0->40->30->20->10->|||

Inputkan nilai untuk di sisipkan pada akhir list : 0
List setelah input data baru : H->0->40->30->20->10->0->|||

Inputkan nilai yang akan di sisipkan pada list : 25
Di sisipkan setelah ? : 30
List setelah input data baru : H->0->40->30->25->20->10->0->|||

Inputkan nilai yang akan di sisipkan pada list : 50
Di sisipkan sebelum ? : 40
List setelah input data baru : H->0->50->40->30->25->20->10->0->|||

Process returned 0 (0x0)   execution time : 100.677 s
Press any key to continue.

```

Penjelasan :

- Fungsi print_list digunakan untuk mengisi inputan linked list pada program.
- Fungsi insert_front digunakan untuk mengisi inputan pada linked list di awal inputan.
- Fungsi insert_end digunakan untuk mengisi inputan pada linked list di akhir inputan.
- Fungsi insert_after digunakan untuk mengisi inputan yang disisipkan pada salah satu inputan tapi diinput setelah inputan tersebut.
- Fungsi insert_before digunakan untuk mengisi inputan yang disisipkan pada salah satu inputan tapi diinput sebelum inputan tersebut.

C. Tugas Praktikum

1. Implementasikan 3 algoritma penambahan node baru (end, after, dan Before) pada program percobaan yang ada pada modul sebelumnya (modul 2).

Jawab :

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

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

```
struct node
```

```
{
```

```
    char nama[50];
```

```

    int val;
    struct node *next;
};

void print_list(struct node *head)
{
    printf("H->");

    while(head)
    {
        printf("%s %i ->", head->nama, head->val);
        head = head->next;
    }

    printf("|||\n\n");
}

void insert_front(struct node **head, char nama[], int value)
{
    struct node * new_node = NULL;

    new_node = (struct node *)malloc(sizeof(struct node));

    if(new_node == NULL)
    {
        printf("tidak bisa menginputkan node baru, memori penuh");
    }

    strcpy(new_node->nama, nama);
    new_node->val = value;
    new_node->next = *head;

    *head = new_node;
}

void insert_end(struct node **head, char nama[], int value)
{
    struct node * new_node = NULL;

```



```

struct node * last = NULL;

new_node = (struct node *)malloc(sizeof(struct node));

if(new_node == NULL)
{
    printf("tidak bisa menginputkan node baru, memori penuh");
}

strcpy(new_node->nama,nama);
new_node->val = value;
new_node->next = NULL;

if( *head == NULL)
{
    *head = new_node;
    return;
}

last = *head;
while(last->next) last = last->next;

last->next = new_node;
}

void insert_after(struct node *head, char nama[],int value, int after)
{
    struct node * new_node = NULL;
    struct node *tmp = head;

    while(tmp)
    {
        if(tmp->val == after)
        {
            new_node = (struct node *)malloc(sizeof(struct node));

            if(new_node == NULL)
            {

```

```

        printf("tidak bisa menginputkan node baru, memori penuh");
    }

    strcpy(new_node->nama,nama);
    new_node->val = value;
    new_node->next = tmp->next;
    tmp->next = new_node;
    return;
}

tmp = tmp->next;
}
}

void insert_before(struct node **head, char nama[], int value, int before)
{
    struct node * new_node = NULL;
    struct node * tmp = *head;

    new_node = (struct node *)malloc(sizeof(struct node));

    if(new_node == NULL)
    {
        printf("tidak bisa menginputkan node baru, memori penuh");
        return;
    }

    strcpy(new_node->nama,nama);
    new_node->val = value;

    if((*head)->val == before)
    {
        new_node->next = *head;
        *head = new_node;
        return;
    }

    while(tmp && tmp->next)

```

```

{
    if(tmp->next->val == before)
    {
        new_node->next = tmp->next;
        tmp->next = new_node;
        return;
    }

    tmp = tmp->next;
}

free(new_node);
}

void main()
{
    int count = 0, i, val, after, before;
    char nama[10];
    struct node * head = NULL;

    printf("Inputkan jumlah elemen/node yang akan di inputkan : ");
    scanf("%d", &count);
    fflush(stdin);

    for(i = 0; i < count; i++)
    {
        printf("Enter %d th elemen/node : ", i);
        scanf("%s", &nama);
        fflush(stdin);
        printf("Enter %d th elemen/node : ", i);
        scanf("%d", &val);
        fflush(stdin);
        insert_front(&head, nama, val);
    }

    printf("Initial List : ");
    print_list(head);

```

```
printf("Inputkan nama untuk di sisipkan pada awal list : ");
scanf("%s", &nama);
fflush(stdin);
printf("Inputkan nilai untuk di sisipkan pada awal list : ");
scanf("%d", &val);
fflush(stdin);
insert_front(&head, nama, val);
```

```
printf("List setelah input data baru : ");
print_list(head);
```

```
printf("Inputkan nama untuk di sisipkan pada awal list : ");
scanf("%s", &nama);
fflush(stdin);
printf("Inputkan nilai untuk di sisipkan pada akhir list : ");
scanf("%d", &val);
fflush(stdin);
insert_end(&head, nama, val);
```

```
printf("List setelah input data baru : ");
print_list(head);
```

```
printf("Inputkan nama untuk di sisipkan pada awal list : ");
scanf("%s", &nama);
fflush(stdin);
printf("Inputkan nilai yang akan di sisipkan pada list : ");
scanf("%d", &val);
fflush(stdin);
```

```
printf("Di sisipkan setelah ? : ");
scanf("%d", &after);
fflush(stdin);
```

```
insert_after(head, nama, val, after);
```

```
printf("List setelah input data baru : ");
print_list(head);
```

```

printf("Inputkan nama untuk di sisipkan pada awal list : ");
scanf("%s", &nama);
fflush(stdin);
printf("Inputkan nilai yang akan di sisipkan pada list : ");
scanf("%d", &val);
fflush(stdin);

printf("Di sisipkan sebelum ? : ");
scanf("%d", &before);
fflush(stdin);

insert_before(&head, nama, val, before);

printf("List setelah input data baru : ");
print_list(head);
}

```

Output :

```

D:\New Linked List 6\bin\Debug\New Linked List 6.exe
Inputkan jumlah elemen/node yang akan di inputkan : 4
Enter 0 th elemen/node : Agus
Enter 0 th elemen/node : Agus
Enter 1 th elemen/node : Mia
Enter 1 th elemen/node : Mia
Enter 2 th elemen/node : Rian
Enter 2 th elemen/node : Rian
Enter 3 th elemen/node : Kinan
Enter 3 th elemen/node : Kinan
Initial List : H->Kinan 0 ->Rian 0 ->Mia 0 ->Agus 0 ->|||

Inputkan nama untuk di sisipkan pada awal list : Ardi
Inputkan nilai untuk di sisipkan pada awal list : 1
List setelah input data baru : H->Ardi 1 ->Kinan 0 ->Rian 0 ->Mia 0 ->Agus 0 ->|||

Inputkan nama untuk di sisipkan pada awal list : Bayu
Inputkan nilai untuk di sisipkan pada akhir list : 5
List setelah input data baru : H->Ardi 1 ->Kinan 0 ->Rian 0 ->Mia 0 ->Agus 0 ->Bayu 5 ->|||

Inputkan nama untuk di sisipkan pada awal list : Dea
Inputkan nilai yang akan di sisipkan pada list : 3
Di sisipkan setelah ? : 1
List setelah input data baru : H->Ardi 1 ->Dea 3 ->Kinan 0 ->Rian 0 ->Mia 0 ->Agus 0 ->Bayu 5 ->|||

Inputkan nama untuk di sisipkan pada awal list : Maya
Inputkan nilai yang akan di sisipkan pada list : 4
Di sisipkan sebelum ? : 0
List setelah input data baru : H->Ardi 1 ->Dea 3 ->Maya 4 ->Kinan 0 ->Rian 0 ->Mia 0 ->Agus 0 ->Bayu 5 ->|||

Process returned 0 (0x0)   execution time : 109.368 s
Press any key to continue.

```