

# LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

## Praktikum 4 Single Linked List 1



Disusun oleh :

Nama : Shofira Izza Nurrohmah

Kelas : 1 D3 Teknik Informatika A

NRP : 3122500026

Dosen Pembimbing : Entin Martiana Kusumaningtyas S.Kom, M.Kom

DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER

PRODI TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023

## **DAFTAR ISI**

<b>HALAMAN SAMPUL .....</b>	<b>I</b>
<b>DAFTAR ISI .....</b>	<b>II</b>
<b>A. HASIL PERCOBAAN .....</b>	<b>1</b>
<b>B. LATIHAN .....</b>	<b>2</b>
<b>C. LAPORAN RESMI .....</b>	<b>13</b>

## **A. HASIL PERCOBAAN**

1. Implementasikan operasi dasar Single linked list : Menyisipkan sebagai simpul ujung(awal) dari linked list.
2. Implementasikan operasi dasar Single linked list : Membaca atau menampilkan
3. Implementasikan operasi dasar Single linked list : Mencari sebuah simpul tertentu. Tambahkan kondisi jika yang dicari adalah data yang paling depan.
4. Implementasikan operasi dasar Single linked list : Menyisipkan sebagai simpul terakhir
5. Gabungkan semua operasi di atas dalam sebuah Menu Pilihan.

## **B. LATIHAN**

Bangunlah Single linked list di atas adalah Single linked list dengan prinsip FIFO.

## **C. LAPORAN RESMI**

### **Percobaan**

#### **Listing Program**

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

```
typedef struct simpul Node;
struct simpul {
    char nama[256];
    int nrp;
    Node *next;
};
Node *head=NULL;
```

```
void tampil(){
    Node *tampil=head;
    puts("Isi dari SLL");
    while(tampil!=NULL){
        printf("%s\n", tampil->nama);
```

```

        printf("%d\n", tampil->nrp);
        printf("\n");
        tampil=tampil->next;
    }
}

```

```

void awal(){
    Node *simpul=(Node *) malloc(sizeof(Node));
    char nama[256];
    int x;

    simpul->next=NULL;
    printf("\nMasukkan nama : ");
    scanf("%s", &nama);
    printf("NRP yang mau disimpan : ");
    scanf("%d", &x);
    simpul->nrp=x;
    if(head==NULL){
        head=simpul;
    }else{
        simpul->next=head;
        head=simpul;
    }
}

```

```

void cari(){
    if(!head){
        printf("Linked list kosong\n");
    } else {
        int nrp;
        printf("Masukkan NRP yang dicari : ");
        scanf("%d", &nrp);
    }
}

```

```

Node *cari=head;
while(cari!=NULL){
    if (cari->nrp==nrp){
        printf("Data telah ditemukan\n");
        printf("Nama : %s\n", cari->nama);
        return;
    }
    cari=cari->next;
}
printf("Data tidak ditemukan\n");

}
}

```

```

void akhir(){
    Node *simpul=(Node *) malloc(sizeof(Node));
    Node *tail=head;
    char nama[256];
    int x;

    simpul->next=NULL;
    printf("\nMasukkan nama : ");
    scanf("%s", nama);
    printf("NRP yang mau disimpan : ");
    scanf("%d", &x);
    strcpy(simpul->nama, nama);
    simpul->nrp=x;

    if(head==NULL){
        head=simpul;
    }else{

```

```

while(tail->next != NULL){
    tail=tail->next;
}
tail->next=simpul;
}
}

int main()
{
    char jawab='y';
    int pilihan;

    puts("Single Linked List - Insert Awal");

    while((jawab=='y')||(jawab=='Y')) {
        printf("\nMenu pilihan : \n");
        printf("1. Menyisipkan sebagai simpul ujung (awal)\n");
        printf("2. Membaca atau menampilkan\n");
        printf("3. Mencari sebuah simpul tertentu\n");
        printf("4. Menyisipkan sebagai simpul terakhir\n");
        printf("Masukkan pilihan Anda : ");
        scanf("%d", &pilihan);

        switch(pilihan) {
            case 1:
                awal();
                break;
            case 2:
                tampil();
                break;
            case 3:

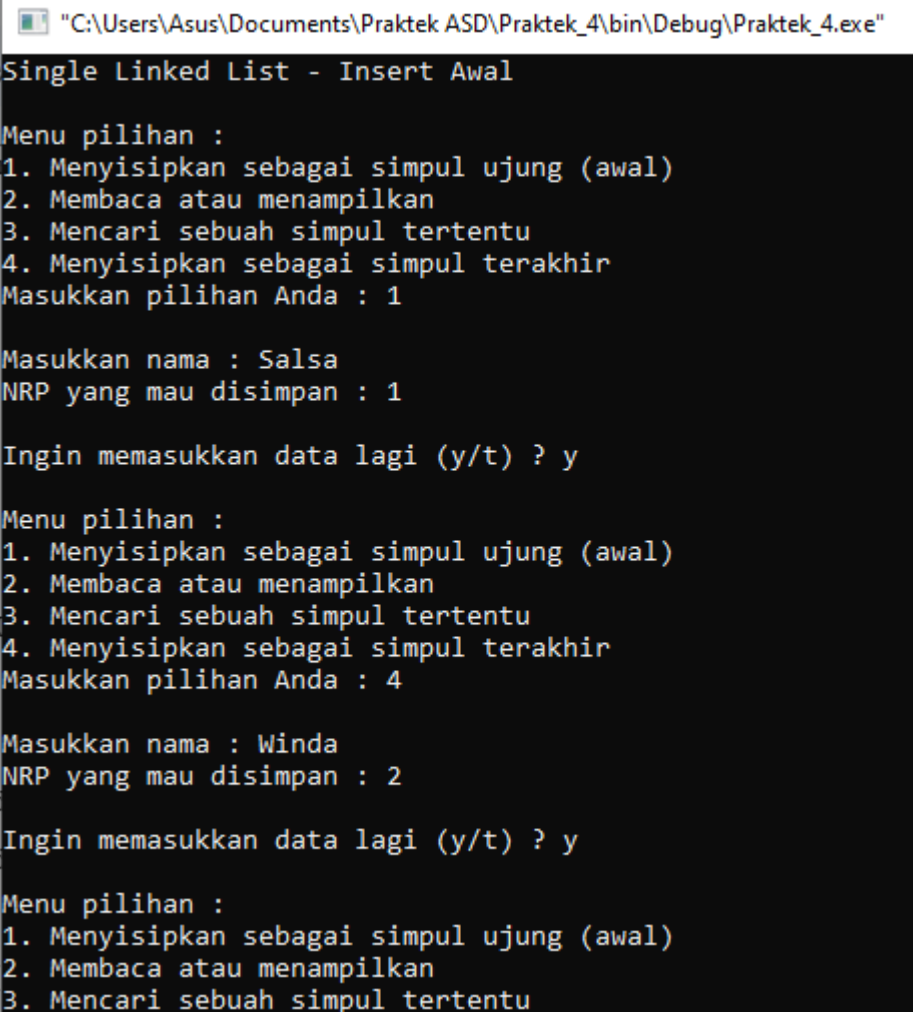
```

```

        cari();
        break;
    case 4:
        akhir();
        break;
    }
    fflush(stdin);
    printf("\nIngin memasukkan data lagi (y/t) ? ");
    scanf("%c", &jawab);
}
return 0;
}

```

## Output



```

"C:\Users\Asus\Documents\Praktek ASD\Praktek_4\bin\Debug\Praktek_4.exe"
Single Linked List - Insert Awal

Menu pilihan :
1. Menyisipkan sebagai simpul ujung (awal)
2. Membaca atau menampilkan
3. Mencari sebuah simpul tertentu
4. Menyisipkan sebagai simpul terakhir
Masukkan pilihan Anda : 1

Masukkan nama : Salsa
NRP yang mau disimpan : 1

Ingin memasukkan data lagi (y/t) ? y

Menu pilihan :
1. Menyisipkan sebagai simpul ujung (awal)
2. Membaca atau menampilkan
3. Mencari sebuah simpul tertentu
4. Menyisipkan sebagai simpul terakhir
Masukkan pilihan Anda : 4

Masukkan nama : Winda
NRP yang mau disimpan : 2

Ingin memasukkan data lagi (y/t) ? y

Menu pilihan :
1. Menyisipkan sebagai simpul ujung (awal)
2. Membaca atau menampilkan
3. Mencari sebuah simpul tertentu

```

```
"C:\Users\Asus\Documents\Praktek ASD\Praktek_4\bin\Debug\Praktek_4.exe"

Menu pilihan :
1. Menyisipkan sebagai simpul ujung (awal)
2. Membaca atau menampilkan
3. Mencari sebuah simpul tertentu
4. Menyisipkan sebagai simpul terakhir
Masukkan pilihan Anda : 2
Isi dari SLL
@hr
1
Winda
2
Ingin memasukkan data lagi (y/t) ? y

Menu pilihan :
1. Menyisipkan sebagai simpul ujung (awal)
2. Membaca atau menampilkan
3. Mencari sebuah simpul tertentu
4. Menyisipkan sebagai simpul terakhir
Masukkan pilihan Anda : 3
Masukkan NRP yang dicari : 2
Data telah ditemukan
Nama : Winda
Ingin memasukkan data lagi (y/t) ?
```

### Analisa :

Program di atas merupakan program C yang mengimplementasikan single linked list (SLL) dengan beberapa fungsi seperti insert di awal, insert di akhir, tampilkan isi SLL, dan cari data dalam SLL. Berikut adalah penjelasan detail dari setiap fungsi dan bagaimana program ini bekerja:

#### 1. Fungsi tampil()

Fungsi ini digunakan untuk menampilkan isi dari seluruh simpul dalam SLL. Pertama-tama, fungsi ini membuat sebuah variabel tampil yang awalnya menunjuk ke simpul pertama dalam SLL (head). Lalu, fungsi ini melakukan looping dari simpul pertama sampai simpul terakhir dalam SLL. Pada setiap simpul, fungsi ini akan menampilkan nama dan nrp yang disimpan dalam simpul tersebut. Terakhir, variabel tampil akan diganti dengan simpul selanjutnya sehingga looping dapat dilakukan terus menerus sampai seluruh isi dari SLL ditampilkan.

#### 2. Fungsi awal()

Fungsi ini digunakan untuk menambahkan simpul baru di awal SLL. Pertama-tama, fungsi ini membuat sebuah simpul baru (simpul) menggunakan fungsi malloc(). Lalu, fungsi ini meminta pengguna untuk memasukkan nama dan nrp untuk simpul baru tersebut. Setelah itu, jika SLL masih kosong (nilai head adalah NULL), simpul baru tersebut akan dijadikan simpul pertama dalam SLL dengan menetapkan head menjadi simpul baru. Namun, jika SLL sudah terisi, simpul baru tersebut



akan disisipkan di awal SLL dengan menetapkan simpul baru tersebut sebagai simpul pertama dan menetapkan simpul pertama sebelumnya sebagai simpul kedua.

### 3. Fungsi cari()

Fungsi ini digunakan untuk mencari sebuah simpul dalam SLL berdasarkan nrp yang diinginkan. Pertama-tama, fungsi ini akan mengecek apakah SLL masih kosong atau tidak. Jika kosong, fungsi akan langsung memberikan pesan bahwa SLL kosong. Jika tidak kosong, fungsi akan meminta pengguna untuk memasukkan nrp yang dicari. Lalu, fungsi ini melakukan looping dari simpul pertama sampai simpul terakhir dalam SLL. Pada setiap simpul, fungsi ini akan membandingkan nrp dalam simpul tersebut dengan nrp yang dicari. Jika nrp yang dicari ditemukan, fungsi akan memberikan pesan bahwa data telah ditemukan beserta nama yang bersesuaian. Namun, jika nrp yang dicari tidak ditemukan, fungsi akan memberikan pesan bahwa data tidak ditemukan.

### 4. Fungsi akhir()

Fungsi ini digunakan untuk menambahkan simpul baru di akhir SLL. Pertama-tama, fungsi ini membuat sebuah simpul baru (simpul) menggunakan fungsi malloc(). Lalu, fungsi ini meminta pengguna untuk memasukkan nama dan nrp untuk simpul baru tersebut. Setelah itu, fungsi ini melakukan looping dari simpul pertama sampai simpul terakhir dalam SLL. Pada setiap simpul, fungsi ini akan memeriksa apakah simpul tersebut adalah simpul terakhir atau tidak (dengan memeriksa apakah simpul tersebut memiliki nilai next yang NULL).

## D. KESIMPULAN

Secara konseptual Array adalah struktur data yang memesan tempat secara berurutan untuk jumlah data statis. Sedangkan Memory Allocation (malloc) adalah sebuah fungsi fasilitas untuk memesan tempat secara berurutan untuk tipe data pointer dengan jumlah data dinamis.

Permasalahan akan timbul jika kita memesan data dengan jumlah yang besar sementara tempat di memori yang berurutan tidak ada. Solusi untuk permasalahan ini adalah dengan menggunakan linked list atau senara berantai. Linked list merupakan deretan elemen yang berdampingan. Akan tetapi, karena elemen-elemen tersebut dialokasikan secara dinamis (menggunakan malloc), sangat penting untuk diingat bahwa kenyataannya, linked list akan terpencar-pencar di memori. Pointer pada suatu elemen berisi alamat untuk data berikutnya sebagai penjamin bahwa semua elemen dapat diakses.