**IK141 Struktur Data**

# Pendahuluan Struktur Data

# Dynamic Single Linked List 2

**Di Susun Oleh :**

**Wendi Kardian, 2100016**

**PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER**

**FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PENDIDIKAN INDONESIA**

**5 Maret 2022**

## 1. Implementasi dan Hasil

**Implementasi dan Hasil**

### A) Study Kasus

Buatlah struktur data ingle linked list untuk menyimpan data Jemaah haji tersebut dengan ketentuan penyimpanan data dalam linked list adalah: Data yang disimpan adalah data Nama, Gender dan Usia

1. Data yang disimpan adalah data Nama, Gender, dan Usia.
2. Jemaah haji yang dipanggil untuk melaksanakan ibadah haji berdasarkan urutan list dari first ke last
3. Setiap data Jemaah dipanggil maka harus dihapus dari linked list
4. Jika ada Jemaah haji baru yang daftar, maka data akan disimpan di dalam list dengan ketentuan :
    a. Jemaah yang usianya lebih tua akan didahulukan dalam antrian
    b. Jika ada Jemaah yang usianya sama maka urutan akan disesuikan berdasarkan urutan pendaftaran, didahulukan yang paling awal daftar
    c. Perempuan dan laki-laki dengan usia sama, maka perempuan akan diprioritaskan urutannya.
5. Kode untuk mendaftar adalah 1, Kode untuk memanggil dan menghapus dari daftar adalah 2, kode untuk mencetak adalah 3 dan kode untuk mengakhiri program adalah 0

### B) Penjelasan Algoritma

1. Pertama perlu di deklarasikan terlebih dahulu structure linked list yang nantinya berisikan data nama, gender, usia serta diperlukan penghubung untuk node berikutnya yaitu node.

```
struct node{
    char nama[15];
    char gender[10];
    int usia;
    struct node *next;
};

struct node *start = NULL;
```

2. Di program utama (int main) akan dideklarasikan beberapa tipe data dan akan dibuatkan sebuat menu, dimana didalamya berisikan menu dimana user dapat memilih menu yang nanti akan dipilihnya

```c
int main(){
    int option = 4, count;
    int usia,i, j;
    int toDelete;
    struct node *ptr;
    char nama[15], gender[10];
    do{
        scanf("%d", &option);
        switch(option){
            case 1:
            scanf("%d", &count);
            for(i = 0; i< count; i++){
```

```c
        case 2:
            scanf("%d", &toDelete);
            for(j = 0; j<toDelete ; j++){
                start = delete_beg(start);
            }
            break;
        case 3:
            if(start == NULL){
                printf("Daftar List Kosong \n");
            }else{
                start = display(start);
            }
            break;
        }

    }while(option != 0);
```

3. Apabila user menginputkan no.1 User akan diminta untuk menambahkan data kedalam linked list, dimana untuk urutan penyimpanan menyesuaikan rule yang sudah dijelaskan disoal, jadi terdapat kondisi yang akan mengatur bagaimana nanti data akan disimpan, bisa saja data disimpan di awal, di akhir, di tengah. Untuk menentukan urutan data disimpan berdasarkan usia, gender, serta urutan input data.

```c
do{
    scanf("%d", &option);
    switch(option){
        case 1:
        scanf("%d", &count);
        for(i = 0; i< count; i++){
            scanf("%s", nama);
            scanf("%s", gender);
            scanf("%d", &usia);
            if(start == NULL){
                start = createList(nama,gender,usia, start);
            }else if(start->next == NULL){
                if(start->usia > usia){
                    start = insert_end(nama, gender,usia,start);
                }else{
                    if(start->usia == usia){
                        if(strcmp(start->gender, "Perempuan") == 0 && strcmp(gender, "Laki-laki") == 0){
                            start = insert_end(nama, gender,usia,start);
                        }else if(strcmp(start->gender, "Perempuan") == 0  && strcmp(gender, "Perempuan") == 0){
                            start = insert_end(nama, gender,usia,start);
                        }else if(strcmp(gender, "Perempuan") == 0 && strcmp(ptr->gender, "Laki-laki")==0){
                            start = insert_bef(nama, gender,usia,ptr->nama,start);
                        }
                        else{
                            start = insert_beg(nama, gender,usia,start);
                        }
                    }else{
                        start = insert_beg(nama, gender,usia,start);
                    }
                }
```

```
            }
        }else{
            ptr = start;
            int i;
            while(ptr->usia > usia && ptr->next != NULL){
                ptr = ptr->next;
            }

            if(ptr->usia == usia){
                if(strcmp(ptr->gender, "Perempuan") == 0 && strcmp(gender, "Laki-laki") == 0){
                    while(strcmp(ptr->next->gender, "Laki-laki") == 0){
                        ptr = ptr->next;
                    }
                    if(ptr->next == NULL){
                        start = insert_end(nama, gender,usia,start);
                    }else{
                        start = insert_after(nama, gender, usia, ptr, start);
                    }


                }else if(strcmp(start->gender, "Perempuan") == 0  && strcmp(gender, "Perempuan") == 0){
                    start = insert_after(nama, gender, usia, ptr, start);
                }else if(strcmp(gender, "Perempuan") == 0 && strcmp(ptr->gender, "Laki-laki")==0){
                    if(start == ptr){
                        start = insert_beg(nama, gender,usia,start);
                    }
                    else{
                        start = insert_bef(nama, gender,usia,ptr->nama,start);

                    }
                }
                else{
                    while(strcmp(ptr->next->gender, "Laki-laki") == 0){
                        ptr = ptr->next;
```

```
                else{
                    while(strcmp(ptr->next->gender, "Laki-laki") == 0){
                        ptr = ptr->next;
                    }
                    if(ptr->next == NULL){
                        start = insert_end(nama, gender,usia,start);
                    }else{
                        start = insert_after(nama, gender, usia, ptr->next, start);
                    }

                }
            }else if(start->usia < usia){
                start = insert_beg(nama, gender,usia,start);
            }
            else if(ptr->usia < usia){
                start = insert_bef(nama, gender,usia,ptr->nama,start);

            }
            else{
                start = insert_end(nama, gender,usia,start);
            }
        }
    }
    break;
```

4. Untuk menghapus antrian, user dapat menginputkan no.2 dimana akan merujuk kepada sebuah function delete_beg.

```
        case 2:
            scanf("%d", &toDelete);
            for(j = 0; j<toDelete ; j++){
                start = delete_beg(start);
            }
            break;
```

```
struct node * delete_beg(struct node *start){
    struct node *ptr;
    ptr = start;
    start = start->next;
    free(ptr);
    return start;
}
```

5. Untuk menampilkan semua data yang telah di inputkan kedalam linked list, user dapat menginputkan no.3 dimana akan memanggil sebuah function Bernama display untuk menampilkan data data yang terdapat di dalam linked list tersebut.

```
193             case 3:
194                 if(start == NULL){
195                     printf("Daftar List Kosong \n");
196                 }else{
197                     start = display(start);
198                 }
199                 break;
200             }
```

6. Program akan terus berjalan sampai user menekan no.0 untuk menghentikan program.

```
}while(option != 0);
```

7. Terdapat beberapa function didalamnya untuk menambahkan data ke dalam linked list, tergantung situasi dan kondisi dari linked list tersebut, Adapun fungsi-fungsinya diantaranya sebagai berikut:

   a. CreateList untuk membuat list baru

```
struct node *createList(char nama[], char gender[], int usia, struct node *start){
    struct node *new_node;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = NULL;
    start = new_node;
    return start;
}
```

   b. Insert_end untuk menambahkan node baru di akhir linked list tersebut

```
struct node *insert_end(char nama[], char gender[], int usia, struct node *start){
    struct node *new_node, *ptr;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = NULL;
    start->next = new_node;
    return start;
}
```

   c. Insert_beg untuk menambahkan node baru di awal linked list tersebut

```
49  struct node *insert_beg(char nama[], char gender[], int usia,struct node *start){
50      struct node *new_node;
51      new_node = (struct node *)malloc(sizeof(struct node));
52      strcpy(new_node->nama,nama);
53      strcpy(new_node->gender,gender);
54      new_node->usia = usia;
55      new_node->next = start;
56      start = new_node;
57      return start;
58
59  }
```

d. Insert_after untuk menambahkan element baru setelah node tertentu

```
struct node *insert_after(char nama[], char gender[], int usia, struct node *loc,struct node *start){
    struct node *new_node;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = loc->next;
    loc->next = new_node;
    return start;
}
```

e. Insert_bef untuk menambahkan element baru sebelum node tertentu

```
struct node *insert_bef(char nama[], char gender[], int usia, char cari[],struct node *start){
    struct node *new_node, *ptr;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama,nama);
    strcpy(new_node->gender,gender);
    new_node->usia = usia;
    ptr = start;
    while(strcmp(ptr->next->nama, cari) != 0){
        ptr = ptr->next;
    }
    new_node->next =ptr->next;
    ptr->next = new_node;
    return start;
}
```

**C) Souce Code**

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


struct node{
    char nama[15];
    char gender[10];
    int usia;
    struct node *next;
};

struct node *start = NULL;


struct node *createList(char nama[], char gender[], int usia,
struct node *start){
    struct node *new_node;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = NULL;
    start = new_node;
```

```
    return start;
}

struct node *insert_end(char nama[], char gender[], int usia,
struct node *start){
    struct node *new_node, *ptr;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = NULL;
    start->next = new_node;
    return start;
}


struct node *display(struct node *start){
    struct node *ptr;
    ptr = start;
    while(ptr != NULL){
        printf("\n%s\n%s\n%d\n", ptr->nama, ptr->gender, ptr-
>usia);
        ptr = ptr->next;
    }
    return start;
}

struct node *insert_beg(char nama[], char gender[], int
usia,struct node *start){
    struct node *new_node;
    new_node = (struct node *)malloc(sizeof(struct node));
    strcpy(new_node->nama,nama);
    strcpy(new_node->gender,gender);
    new_node->usia = usia;
    new_node->next = start;
    start = new_node;
    return start;

}

struct node *insert_after(char nama[], char gender[], int
usia, struct node *loc,struct node *start){
    struct node *new_node;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama, nama);
    strcpy(new_node->gender, gender);
    new_node->usia = usia;
    new_node->next = loc->next;
```

```
    loc->next = new_node;
    return start;
}

struct node *insert_bef(char nama[], char gender[], int usia,
char cari[],struct node *start){
    struct node *new_node, *ptr;
    new_node = (struct node *) malloc(sizeof(struct node));
    strcpy(new_node->nama,nama);
    strcpy(new_node->gender,gender);
    new_node->usia = usia;
    ptr = start;
    while(strcmp(ptr->next->nama, cari) != 0){
        ptr = ptr->next;
    }
    new_node->next =ptr->next;
    ptr->next = new_node;
    return start;
}

struct node * delete_beg(struct node *start){
    struct node *ptr;
    ptr = start;
    start = start->next;
    free(ptr);
    return start;
}



int main(){
    int option = 4, count;
    int usia,i, j;
    int toDelete;
    struct node *ptr;
    char nama[15], gender[10];
    do{
        scanf("%d", &option);
        switch(option){
            case 1:
            scanf("%d", &count);
            for(i = 0; i< count; i++){
                scanf("%s", nama);
                scanf("%s", gender);
                scanf("%d", &usia);
                if(start == NULL){
                    start = createList(nama,gender,usia,
start);
```

```
                }else if(start->next == NULL){
                    if(start->usia > usia){
                        start = insert_end(nama,
gender,usia,start);
                    }else{
                        if(start->usia == usia){
                            if(strcmp(start->gender,
"Perempuan") == 0 && strcmp(gender, "Laki-laki") == 0){
                                start = insert_end(nama,
gender,usia,start);
                            }else if(strcmp(start->gender,
"Perempuan") == 0  && strcmp(gender, "Perempuan") == 0){
                                start = insert_end(nama,
gender,usia,start);
                            }else if(strcmp(gender,
"Perempuan") == 0 && strcmp(ptr->gender, "Laki-laki")==0){
                                start = insert_bef(nama,
gender,usia,ptr->nama,start);
                            }
                            else{
                                start = insert_beg(nama,
gender,usia,start);
                            }
                        }else{
                            start = insert_beg(nama,
gender,usia,start);
                        }
                    }
                }else{
                    ptr = start;
                    int i;
                    while(ptr->usia > usia && ptr->next !=
NULL){
                        ptr = ptr->next;
                    }

                    if(ptr->usia == usia){
                        if(strcmp(ptr->gender, "Perempuan") ==
0 && strcmp(gender, "Laki-laki") == 0){
                            while(strcmp(ptr->next->gender,
"Laki-laki") == 0){
                                ptr = ptr->next;
                            }
                            if(ptr->next == NULL){
                                start = insert_end(nama,
gender,usia,start);
                            }else{
                                start = insert_after(nama,
```

```
gender, usia, ptr, start);
                            }


                        }else if(strcmp(start->gender,
"Perempuan") == 0  && strcmp(gender, "Perempuan") == 0){
                            start = insert_after(nama, gender,
usia, ptr, start);
                        }else if(strcmp(gender, "Perempuan")
== 0 && strcmp(ptr->gender, "Laki-laki")==0){
                            if(start == ptr){
                                start = insert_beg(nama,
gender,usia,start);
                            }
                            else{
                                start = insert_bef(nama,
gender,usia,ptr->nama,start);

                            }
                        }
                        else{
                            while(strcmp(ptr->next->gender,
"Laki-laki") == 0){
                                ptr = ptr->next;
                            }
                            if(ptr->next == NULL){
                                start = insert_end(nama,
gender,usia,start);
                            }else{
                                start = insert_after(nama,
gender, usia, ptr->next, start);
                            }

                        }
                    }else if(start->usia < usia){
                        start = insert_beg(nama,
gender,usia,start);
                    }
                    else if(ptr->usia < usia){
                        start = insert_bef(nama,
gender,usia,ptr->nama,start);

                    }
                    else{
                        start = insert_end(nama,
gender,usia,start);
                    }
                }
```

```
        }
        break;
        case 2:
            scanf("%d", &toDelete);
            for(j = 0; j<toDelete ; j++){
                start = delete_beg(start);
            }
            break;
        case 3:
            if(start == NULL){
                printf("Daftar List Kosong \n");
            }else{
                start = display(start);
            }
            break;
        }

    }while(option != 0);

    return 0;

}
```

**D) PENGUJIAN**

1. Input pertama

```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 4> cd "d:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 4\" ; if ($?) { gcc programHaji2.c -o programHaji2 } ; if ($?) { .\programHaji2 }
1
4
Shopia
Perempuan
40
Ahmad
Laki-laki
45
Beta
Perempuan
50
Budi
Laki-laki
53
2
1
3

Beta
Perempuan
50

Ahmad
Laki-laki
45

Shopia
Perempuan
40
```

2. Input keduas

```
PROBLEMS 31  OUTPUT  DEBUG CONSOLE  TERMINAL

1
5
Perempuan
26
Toni
Laki-laki
50
Shopi
Perempuan
50
Firman
Laki-laki
50
Mustika
Perempuan
53
3

Mustika
Perempuan
53

Shopi
Perempuan
50

Toni
Laki-laki
50

Firman
Laki-laki
50

Ratna
Perempuan
26
```

** Lanjutan input sebelumnya **

```
Ratna
Perempuan
26
1
1
Rini
Perempuan
43
3

Mustika
Perempuan
53

Shopi
Perempuan
50

Toni
Laki-laki
50

Firman
Laki-laki
50

Rini
Perempuan
43

Ratna
Perempuan
26
```

## 2. Kesimpulan

| **Kesimpulan** |
| --- |
| Single Linked List merupakan sebuah terminology dari struktur data dimana membahas bagaimana data dapat disimpan secara dinamik dan flexible jumlahnya berbeda dengan array yang hanya bisa di deklarasikan secara static. Dalam linked list terdapat data yang berisikan informasi serta next yang berisikan alamat pointer node selanjutnya. Dalam linked list dapat di implementasikan menjadi sebuah kasus dimana data yang disimpan kedalam suatu linked list tersebut urutannya disesuaikan dengan kondisi aturan dari data tersebut, dimana dalam kasus ini dapat dilihat bahwa posisi penyimpanan node dalam suatu linked list dilihat dari usia, gender, serta urutan input data jamaah yang diinputkan kedalam program ini. |