IK141 Struktur Data

Pendahuluan Struktur Data

Dynamic Single Linked List

Di Susun Oleh :

Wendi Kardian, 2100016

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

27 Februari 2022

1. Implementasi dan Hasil

Implementasi dan Hasil

A. Secara umum setiap elemen linked list dapat dibagi menjadi 2 bagian, yaitu data dan alamat element selanjutnya. Untuk lebih memahami mengenai linked list, silahkan buat program dengan menggunakan struktur data linked list dengan detail informasi:Sebuah linked list terdiri dari beberapa elemen, yang setiap elemennya akan digunakan untuk menampung data berikut :

a. Id (int)

b. Nama (String dengan panjang maksimal 15 karakter)

c. Usia (bilangan bulat)

a. Proses mengimport library serta membuat structure yang nanti akan dijadikan sebagai linked list dengan dijadikan sebagai tipe data buatan Bernama element yang didalam nya berisikan

1. Data yang terdiri dari id(int), nama(char), serta usia(int)
2. Next untuk menunjukan alamat element selanjutnya

praktikum1.c

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

typedef struct Element{
    int id;
    char nama[15];
    int usia;
    struct Element * next;
}element;
```

b. Membuat **fungsi utama**, dimana didalam fungsi ini akan melakukan deklarasi variable pointer linked list yang diberi nama New. Lalu deklarasi beberapa variable untuk melakukan input data yang nanti akan di isi oleh user. Setelah melakukan pengisian data dari user dan dimasukan ke dalam variable tertentu, setiap variable tertentu akan digunakan sebagai parameter untuk berbagai fungsi dan prosedur untuk melakukan operasi pada linked list tersebut. Fungsi dan prosedur tersebut diantaranya adalah sebagai berikut

1. createElement(id, nama, usia) -> untuk membuat element baru linked list.
2. addAwal(id, nama, usia, &New) -> untuk menyisipkan element baru kedalam linked list tersebut dipaling depan
3. addAkhir(id, nama, usia, New) -> untuk menyisipkan element diakhir linked list
4. addAfter(id, nama, usia, id2, New) -> untuk menyisipkan element baru setelah element dengan id tertentu dalam linked list tertentu

5. addBefore(id, nama, usia, id2, New) -> untuk menyisipkan element baru sebelum element dengan id tertentu dalam linked list tertentu
6. hapus(id, &New) -> untuk menghapus element dengan id tertentu dalam sebuah linked list.
7. cetakList(New) -> untuk mencetak list yang sudah dimasukan dalam list tertentu

```
110  int main(){
111      element *New;
112      int id, id2;
113      char nama[15];
114      int usia;
115      scanf("%d", &id);
116      scanf("%s", nama);
117      scanf("%d", &usia);
118      New = createElement(id, nama, usia);
119      scanf("%d", &id);
120      scanf("%s", nama);
121      scanf("%d", &usia);
122      addAwal(id, nama, usia, &New);
123      scanf("%d", &id);
124      scanf("%s", nama);
125      scanf("%d", &usia);
126      addAkhir(id, nama, usia, New);
127      scanf("%d", &id);
128      scanf("%s", nama);
129      scanf("%d", &usia);
130      scanf("%d", &id2);
131      addAfter(id, nama, usia, id2, New);
132      scanf("%d", &id);
133      scanf("%s", nama);
134      scanf("%d", &usia);
135      scanf("%d", &id2);
136      addBefore(id, nama, usia, id2, New);
137      scanf("%d", &id);
138      hapus(id, &New);
```

```
    scanf("%d", &id);
    scanf("%s", nama);
    scanf("%d", &usia);
    addAkhir(id, nama, usia, New);
    scanf("%d", &id);
    hapus(id, &New);
    scanf("%d", &id);
    hapus(id, &New);
    cetakList(New);
    return 1;
}
```

c. **Fungsi createElement** yang bertipe data pointer element dengan menerima parameter id, nama, dan usia yang nanti akan dimasukan kedalam linked list. Pertama-tama kita akan meminta C untuk mengalokasikan memory untuk linked list yang nanti akan dibuat menggunakan function malloc dengan parameter besarnya ukuran dari tipedata element tersebut dan nilainya akan dimasukan kedalam variable pointer New. Lalu variable pointer New akan diisi dengan id, nama, usia yang telah dikirimkan kedalam parameter tersebut dan

diisi kedalam data structure tersebut. Setelah data tersebut dimasukan, kemudian varibel New akan dikembalikan.

```
element * createElement(int id, char nama[15], int usia){
    element * New ;
    New = (element *) malloc (sizeof(element));
    New->id = id;
    strcpy (New->nama, nama);
    New->usia = usia;
    New->next = NULL;
    return New;
}
```

d. **Prosedur addAwal** yang memiliki berisikan parameter id, nama, usia, lalu pointer list yang nanti akan diisikan, dimana di prosedur ini akan memanggil fungsi createElement yang nanti nilai returnnya akan diisikan ke dalam variable pointer New. Dimana alamat selanjutnya (next) dari variable pointer tersebut akan diisi dengan alamat dari list yang sebelumnya dan variable New ini akan dijadikan head dari linked list tersebut.

```
void addAwal(int id, char nama[15], int usia, element
**myList) {
    element * New;
    New = createElement(id, nama, usia);
    New->next = *myList;
    *myList = New;
    New = NULL;
}
```

e. **Prosedur addAkhir** yang memiliki berisikan parameter id, nama, usia, lalu pointer list yang nanti akan diisikan, dimana di prosedur ini akan memanggil fungsi createElement yang nanti nilai returnnya akan diisikan ke dalam variable pointer New. Kemudian prosedur ini akan melakukan pengecekan berkala menggunakan while untuk menentukan element mana yang paling ujung, ciri element paling ujung memiliki nilai next = NULL. Apabila sudah ditemukan maka element terakhir nilai next tersebut akan diisi dengan alamat pointer variable New.

```
void addAkhir(int id, char nama[15], int usia, element
*MyList){
    element *New, *temp;
    New = createElement(id, nama, usia);
    temp=MyList;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=New;
    New=NULL;
}
```

f. **Prosedur addAfter** yang berisikan 4 parameter yaitu data -> (id, nama, usia), data id yang nanti akan dijadikan rujukan untuk melakukan insertnya (setelah id tersebut), kemudian dari list yang mana nanti akan dimodifikasi. Prosedur akan melakukan pengecekan dimana id itu berada, jika id itu sudah ditemukan maka akan disisipkan data variable New (setelah diisi oleh CreateElement) dan akan dihubungkan value next dari element-element tersebut.

```
void addAfter (int id, char nama[15], int usia,int id2,
element *myList) {
    element *temp, *New;
    New = createElement(id, nama, usia);
    temp=myList;
    while(temp->id!=id2 && temp->next!=NULL){
        temp=temp->next;
    }
    if(temp->id==id2){
        New->next=temp->next;
        temp->next=New;
        New=NULL;
    }else{
        printf("Data %d tidak ditemukan\n",id2 );
    }

}
```

g. **Prosedur addBefore** yang berisikan 4 parameter yaitu data -> (id, nama, usia), data id yang nanti akan dijadikan rujukan untuk melakukan insertnya (sebelum id tersebut), kemudian dari list yang mana nanti akan dimodifikasi. Prosedur akan melakukan pengecekan dimana id itu berada dicarinya berdasarkan element sebelumnya (temp->next->id), jika id itu sudah ditemukan maka akan disisipkan data variable New (setelah diisi oleh CreateElement) dan akan dihubungkan value next dari element-element tersebut.

```
void addBefore(int id, char nama[15], int usia,int id2,
element *myList){
    element *New, *temp;
    New = createElement(id, nama, usia);
    temp = myList;
    while(temp->next->id!=id2 && temp->next->next != NULL){
        temp = temp->next;
    }
    if(temp->next->id == id2){
        New->next = temp->next;
        temp->next = New;
        New = NULL;
    }else{
        printf("Data %d tidak ditemukan \n", id2);
    }
}
```

h. **Prosedur hapus** berisikan parameter id yang akan dihapus, serta list mana yang ingin di modifikasi. Di dalam prosedur tersebut dilakukan pengecekan terlebih dahulu di ataranya :

1. Melakukan pengecekan apakah elemen pertama tersebut adalah id yang dicari, apabila iya id tersebut akan dihapus lalu dikosongkan menggunakan free, kemudian alamat selanjutnya akan dijadikan sebagai head dari element tersebut. Kemudian, apabila tidak akan dicek apakah di element selanjutnya merupakan id yang dicari sampai ujung element tersebut. Apabila ditemukan ternyata id element selanjutnya ada id tersebut maka alat next dari id tersebut akan di pindahkan 1 langkah lebih ke depannya dan alamat toDelete (temp->next) tersebut akan dikosongkan.

```c
void hapus(int id, element **myList){
    element *temp, *toDelete;
    temp = *myList;
    if(temp->id == id){
        *myList = temp->next;
        free(temp);
    }else{
        while(temp->next->id != id && temp->next->next !=
        NULL){
        temp = temp->next;
        }
        if(temp->next->id == id){
            toDelete = temp->next;
            temp->next = toDelete->next;
            free(toDelete);
        }else{
            printf("Data tidak %d tidak di temukan", id);
        }
    }
}
```

i. **Prosedur cetakList** menerima 1 parameter yaitu myList, dimana myList tersebut akan dilakukan proses pencetakan dari element head sampai tail dari linked list tersebut.

```c
void cetakList(element *MyList){
    element * temp;
    temp=MyList;
    printf("%d %s %d \n",temp->id, temp->nama, temp->usia);
    do{
        temp=temp->next;
        printf("%d %s %d \n",temp->id, temp->nama,
        temp->usia);
    }while(temp->next!=NULL);
}
```

OUTPUT DARI PROGRAM :

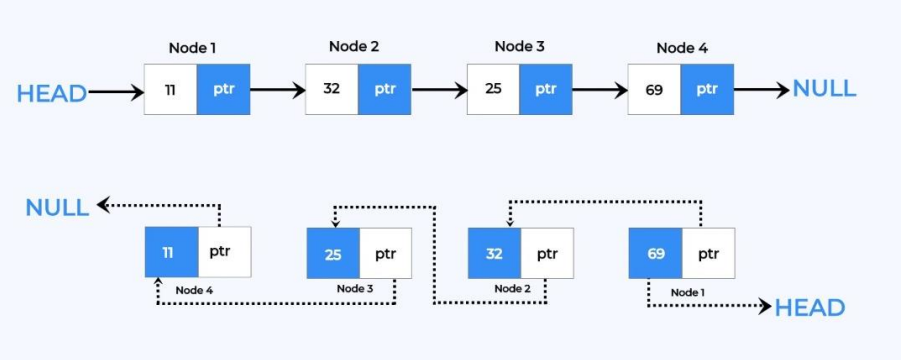
```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Praktikum 3> cd "d:\UPI
\SEMESTER 2\STRUKTUR DATA\Praktikum 3\" ; if ($?) { gcc pr
aktikum1.c -o praktikum1 } ; if ($?) { .\praktikum1 }
0
Dewi
20
1
Tina
19
2
Fio
18
3
Jeni
15
1
4
David
21
3
2
5
Budi
19
3
1
4 David 21
0 Dewi 20
5 Budi 19
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Praktikum 3>
```

B. Dari soal yang dibuat buatkan sebuah fungsi atau prosedur yang dapat mereverse data element linked list. Nama fungsinya adalah reverseList(element * list);Fungsi utama yang akan digunakan adalah:

a. **Fungsi utama,** fungsi utama berisika deklarasi variable pointer linked list yang diberi nama New, lalu deklarasi variable yang lainnya yang nantinya akan digunakan untuk melakukan input data yang akan dilakukan oleh user, setiap data akan dimasukan kedalam linked List createElement dan addAkhir (sudah dijelaskan pada soal sebelumya), setelah data dimasuka maka nanti akan melakukan reverseList dengan membawa parameter List mana yang akan direverse, lalu nanti hasilnya akan dicetak kedalam console. Reverse merupakan proses mengubah urutan linked list dimana element yang paling depan akan diubah menjadi element paling belakang, dan sebaliknya.

```c
int main(){
    element * New;
    int id, n, i;
    char nama[15];
    int usia;
    scanf("%d", &n);
    for(i=0; i<n; i++){
        scanf("%d", &id);
        scanf("%s", nama);
        scanf("%d", &usia);
        if(i==0){
            New = createElement(id, nama, usia);
        }else{
            addAkhir(id, nama, usia, New);
        }
    }
    reverseList(&New);
    cetakList(New);
    return 1;
}
```

b. **Prosedur reverseList** yang menerima parameter alamat dari list mana yang nanti urutannya akan direverse. Untuk proses alur reverse dijelaskan di gambar dibawah ini.



Sumber gambar : Previnsta

Dalam melakukan reverse linkedList memerlukan 3 variabel pointer bantuan untuk menyimpan dan mengingat alamat pointer tersebut saat nanti proses pengubahan pointer, agar tidak kehilangan jejak setelah pointer tersebut diubah. Misalnya 3 variabel tersebut diberi nama prev, current, next. Current, akan berisikan linked list yang akan dimodifikasi, Next akan berisikan alamat selanjutnya dari linked list current tersebut. Lalu setelah linked list tersebut pindah ke alamat selanjutnya, variable pointer prev akan menampung alamat sebelumnya agar nanti dapat dihubungkan Kembali dengan pointer selanjutnya di value nextnya. Proses pertukaran alamat tersebut akan terus dilakukan, hingga nanti nilai current tersebut bernilai

NULL atau sudah berada diujung linked list tersebut, dan variable pointer prev teersebut akan dijadikan sebagai head dari linked list tersebut.

```
void reverseList(element **myList){
    element *prev = NULL;
    element *current = *myList;
    element *next;
    while(current != NULL){
        next = current->next;
        current->next= prev;
        prev=current;
        current = next;
    }
    *myList = prev;
}
```

OUTPUT DARI PROGRAM :

```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Praktikum 3> cd "d:\UPI
\SEMESTER 2\STRUKTUR DATA\Praktikum 3\" ; if ($?) { gcc pr
aktikum2.c -o praktikum2 } ; if ($?) { .\praktikum2 }
3
0
Dewi
20
1
Tina
19
2
Fio
18
2 Fio 18
1 Tina 19
0 Dewi 20
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Praktikum 3>
```

2. Kesimpulan

Kesimpulan
Single linked list merupakan salah satu bentuk ADT yang terdiri dari beberapa struct didalamnya yang bisa terus ditambahkan elementnya dengan menghubungkan structure next elementnya dengan alamat pointer element selanjutnya yang terus bisa dihubungkan (dinamis) seperti gerbong kereta. Jika array itu bersifat statis, dimana element yang ditampungnya tidak flexible, maka linked list bersifat dinamis dinama jumlah element didalamnya flexible tanpa harus dideklarasikan terlebih dahulu. Secara umum, setiap linked list terdiri dari 2 bagian yaitu data dari list tersebut (contoh id,

nama, umur) lalu alamat dari list selanjutnya. Dalam linked list terdapat beberapa operasi yang bisa dilakukan seperti createElement, addAwal, addAkhir, addAfter, addBefore, hapus, cetakList, reverseList.