MAKALAH SINGLE LINKED LIST DALAM BAHASA

PEMROGRAMAN C



Oleh :

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

Contents

[BAB I 2](#_Toc119265010)

[PENDAHULUAN 2](#_Toc119265011)

[1.1 LATAR BELAKANG 2](#_Toc119265012)

[1.2 RUMUSAN MASALAH 3](#_Toc119265013)

[1.3 MANFAAT 3](#_Toc119265014)

[1.4 TUJUAN 4](#_Toc119265015)

[BAB II 4](#_Toc119265016)

[PEMBAHASAN 4](#_Toc119265017)

[2.1 PENGENALAN 4](#_Toc119265018)

[2.2 PENGERTIAN SINGLE LINKED LIST 5](#_Toc119265019)

[2.3 SINGLE LINGKED LIST NON CIRCULAR 6](#_Toc119265020)

[2.4 SINGLE LINKED LIST CIRCULAR 17](#_Toc119265021)

[2.5 CONTOH PROGRAM 29](#_Toc119265022)

[BAB III 36](#_Toc119265023)

[PENUTUP 36](#_Toc119265024)

[DAFTAR PUSTAKA 37](#_Toc119265025)

# BAB I

# PENDAHULUAN

## LATAR BELAKANG

Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambungmenyambung, dinamis dan terbatas.

- Linked List sering disebut juga Senarai Berantai

- Linked List saling terhubung dengan bantuan variabel pointer

- Masing-masing data dalam Linked List disebut dengan node (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa struct yang terdiri dari beberapa field.

Single Linked List adalah sebuah LINKED LIST yang menggunakan sebuah variabel pointer saja untuk menyimpan banyak data dengan metode LINKED LIST, suatu daftar isi yang saling berhubungan. Single Linked List adalah sekumpulan dari node yang saling terhubung dengan node lain melalui sebuah pointer. Single Linked List hanya memiliki satu arah dan tidak memiliki dua arah atau bulak balik, dua arah tersebut disebut dengan double linked list.

Pada Implementasinya, Single Linked List terdapat dua variasi yaitu circular dan noncircular.

## 1.2 RUMUSAN MASALAH

Berdasarkan judul dari makalah ini, Saya merumuskan masalah yaitu single linked list dalam bahasa pemrograman bahasa C!

## 1.3 MANFAAT

Manfaat yang diperoleh dari makalah tersebut adalah sebagai berikut:

a. Supaya kita mengerti atau mengetahui apa yang di maksud dengan senarai(single linked list) dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

b. Supaya kita mengerti atau mengetahui variasi senarai(single lingked list) dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

c. Supaya kita mengerti atau mengetahui apa yang di maksud dengan single Linked List dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

## 1.4 TUJUAN

1. Mengetahui tentang array

2. Dapat menerapkan Bahasa c dan array

# BAB II

# PEMBAHASAN

## 2.1 PENGENALAN

Apabila setiap kali anda ingin menambahkan data anda selalu menggunakan pointer yang baru, anda akan membutuhkan banyak sekali variable pointer (penunjuk). Oleh karena itu, ada baiknya jika anda hanya menggunakan satu variabel pointer saja untuk menyimpan banyak data dengan metode yang kita sebut linked list. Jika kita terjemahkan, ini berarti suatu daftar isi yang saling berhubungan. Struktur data adalah cara menyimpan atau merepresentasikan data di dalam komputer agar bisa dipakai secara efisien Sedangkan data adalah representasi dari fakta dunia nyata. Fakta atau keterangan tentang kenyataan yang disimpan, direkam atau direpresentasikan dalam bentuk tulisan, suara, gambar, sinyal atau symbol. Secara garis besar type data dapat dikategorikan menjadi : 1. Type data sederhana

a. Type data sederhana tunggal, misalnya Integer, real, boolean dan karakter

b. Type data sederhana majemuk, misalnya String

2. Struktur Data, meliputi

a. Struktur data sederhana, misalnya array dan record

b. Struktur data majemuk, yang terdiri dari Linier : Stack, Queue, serta List dan Multilist

Non Linier : Pohon Biner dan Graph Pemakaian struktur data yang tepat di dalam proses pemrograman akan menghasilkan algoritma yang lebih jelas dan tepat, sehingga menjadikan program secara keseluruhan lebih efisien dan sederhana. Struktur data yang ″standar″ yang biasanya digunakan dibidang informatika adalah :

• List linier (Linked List) dan variasinya

• Multilist

• Stack (Tumpukan)

• Queue (Antrian)

• Tree ( Pohon )

• Graph ( Graf )

Salah satu bentuk struktur data yang berisi kumpulan data yang tersusun secara sekuensial, saling bersambungan, dinamis dan terbatas adalah linked list (senarai berkait). Suatu linked list adalah suatu simpul (node) yang dikaitkan dengan simpul yang lain dalam suatu urutan tertentu. Suatu simpul dapat berbentuk suatu struktur atau class. Simpul harus mempunyai satu atau lebih elemen struktur atau class yang berisi data. Secara teori, linked list adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer.

## 2.2 PENGERTIAN SINGLE LINKED LIST

* **Single** : artinya field pointer-nya hanya satu buah saja dan satu arah serta pada akhir node, pointernya menunjuk NULL.
* **Linked List** : artinya node-node tersebut saling terhubung satu sama lain. Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data. Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

Linked list adalah [strukur data](https://www.trivusi.web.id/2022/06/mengenal-struktur-data.html" \t "_blank) linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu **nilai data** dan **pointer ke simpul elemen berikutnya**. Berbeda dengan [array](https://www.trivusi.web.id/2022/07/struktur-data-array.html), elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.

|  |
| --- |
| [ilustrasi linked list](https://media.geeksforgeeks.org/wp-content/uploads/20220518123801/Linkedlist1.png) |
| *Sumber: geeksforgeeks.org* |

Simpul pertama dari linked list disebut sebagai **head** atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Singly linked list adalah linked list unidirectional. Jadi, kita hanya dapat melintasinya dalam satu arah, yaitu dari simpul kepala ke simpul ekor.

|  |
| --- |
| [Ilustrasi Singly linked list](https://www.simplilearn.com/ice9/free_resources_article_thumb/Types-of-Linked-List/types_of_linked_list-singly-linked-list-img1.PNG) |

## 2.3 SINGLE LINGKED LIST NON CIRCULAR

Pembuatan struct bernama tnode berisi 2 field, yaitu field data bertipe integer dan MODUL PRAKTIKUM ALGORITMA & STRUKTUR DATA Jurusan Teknik Elektro – Universitas Negeri Malang - 2016 field next yang bertipe pointer dari tnode. Deklarasi node dengan struct: struct tnode { int data; struct tnode \*next; Gambar 1. Sebuah node pada Single Linked List Asumsikan kita memiliki sejumlah node yang selalu menoleh ke sebelah dalam arah yang sama. Atau, sebagai alat bantu, kita bisa mengasumsikan beberapa orang yang bermain kereta api. Yang belakang akan memegang yang depan, dan semuanya menghadap arah yang sama. Setiap pemain adalah node. Dan tangan pemain yang digunakan untuk memegang bahu pemain depan adalah variabel next. Sampai di sini, kita baru saja mendeklarasikan tipe data dasar sebuah node. Selanjutnya, kita akan mendeklarasikan beberapa variabel pointer bertipe struct tnode. Beberapa variabel tersebut akan kita gunakan sebagai awal dari linked list, node aktif dalam linked list, dan node sementara yang akan digunakan dalam pembuatan node di linked list. Berikan nilai awal NULL kepada mereka. Deklarasi node untuk beberapa keperluan, seperti berikut ini: struct tnode \*head=NULL, \*curr=NULL, \*node=NULL; Dengan demikian, sampai saat ini, telah dimiliki tiga node. Satu sebagai kepala (head), satu sebagai node aktif dalam linked list (curr) dan satu lagi node sementara (node). Untuk mempermudah pengingatan, ingatlah gambar anak panah yang mengarah ke kanan. Head akan berada pada pangkal anak panah, dan node-node berikutnya akan berbaris ke arah bagian anak panah yang tajam. Gambar 2. Single Linked List Apabila diperhatikan, setiap node memiliki petunjuk untuk node sebelahnya. Node terakhir akan diberikan nilai NULL. Dengan demikian, setiap node kebagian jatah. int i; for (i=0; i data = i;

1. **Bentuk Node Single Linked List non Circular**

Pengertian:Single : artinya field pointer-nya hanya satu buah saja dan satu arah serta pada akhir node, pointernya menunjuk NULLLinked List : artinya node-node tersebut saling terhubung satu sama lain.Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data.Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

1. **Pembuatan Single Linked List non Circular**

**Pembuatan Single Linked List non Circular (1):**  
Deklarasi Nodetypedef struct TNode{int data;TNode \*next;};Penjelasan:Pembuatan struct bernama TNode yang berisi 2 field, yaitu field data bertipe integer dan field next yang bertipe pointer dari TNodeSetelah pembuatan struct, buat variabel head yang bertipe pointer dari TNode yang berguna sebagai kepala linked list.

**Pembuatan Single Linked List non Circular (2)**  
Digunakan keyword new yang berarti mempersiapkan sebuah node baru berserta alokasi memorinya, kemudian node tersebut diisi data dan pointer nextnya ditunjuk ke NULL.TNode \*baru;baru = new TNode;baru->data = databaru;baru->next = NULL;

1. **Cara lain alokasi pointer**

Menggunakan alokasi memori secara manualMenggunakan header stdlib.h atau malloc.hMenggunakan fungsi:<pointer type> \*malloc(int size);

1. **Sllnc menggunakan head**

Dibutuhkan satu buah variabel pointer: headHead akan selalu menunjuk pada node pertamaDeklarasi Pointer Penunjuk Kepala Single Linked ListManipulasi linked list tidak bisa dilakukan langsung ke node yang dituju, melainkan harus menggunakan suatu pointer penunjuk ke node pertama dalam linked list (dalam hal ini adalah head). Deklarasinya sebagai berikut:TNode \*head;

Fungsi Inisialisasi Single LinkedList

void init(){

head = NULL;

}

Function untuk mengetahui kosong tidaknya SingleLinkedListJika pointer head tidak menunjuk pada suatu node maka kosong

int isEmpty(){

if

(head == NULL) return 1;

else return 0;

1. **SLLNC dengan HEAD Penambahan data di depan**

Penambahan node baru akan dikaitan di node paling depan, namun pada saat pertama kali (data masih kosong), maka penambahan data dilakukan dengan cara: node head ditunjukkan ke node baru tersebut.Pada prinsipnya adalah mengkaitkan node baru dengan head, kemudian head akan menunjuk pada data baru tersebut sehingga head akan tetap selalu menjadi data terdepan.

Penambahan node baru akan dikaitan di node paling depan, namun pada daat pertama kali (data masih kosong), maka penambahan data dilakukan dengan cara : node head ditunjukan ke node baru tersebut

Prinsipnya adalah mengkaitkan node baru dengan head, kemudian head akan menunjuk pada data baru tersebut sehingga head akan tetap sekaku menjadi data terdepan.

**SLLNC dengan HEAD void insertDepan(int databaru){TNode \*baru;**

baru = new TNode;

baru->data = databaru;

baru->next = NULL;

if(isEmpty()==1){head=baru;

head->next = NULL;

}

else {baru->next = head;

head = baru;

printf(”Data masuk\n”);

1. **SLLNC dengan Head Penambahan data di belakang**

Penambahan data dilakukan di belakang, namun pada saat pertama kali, node langsung ditunjuk oleh head.Penambahan di belakang lebih sulit karena kita membutuhkan pointer bantu untuk mengetahui node terbelakang, kemudian setelah itu, dikaitkan dengan node baru. Untuk mengetahui data terbelakang perlu digunakan perulangan. Penambahan dilakukan di belakang, namin pada saat pertama kali, node langsung ditunjuk oleh head, membutuhkan pointer bantu untuk mengetahui node terbelakang kemudian dikaitkan dengan node baru, perlu digunakan perulangan.

**SLLNC dengan HEAD void insertBelakang (int databaru){**

TNode \*baru,\*bantu;

baru = new TNode;

baru->data = databaru;

baru->next = NULL;

if(isEmpty()==1){

head=baru;

head->next = NULL;

}

else {

bantu=head;

while(bantu->next!=NULL){bantu=bantu->next;

bantu->next = baru;printf("Data masuk\n“);

**SLLNC dengan HEAD void tampil(){ TNode \*bantu; bantu = head;**

if(isEmpty()==0){while(bantu!=NULL){cout<<bantu->data<<" ";

bantu=bantu->next;

}

printf(“\n”);

} else printf(“Masih kosong\n“);

1. **SLLNC dengan HEAD menghapus data di depan**

Function di atas digunakan untuk menampilkan semua isi list, di mana linked list ditelusuri satu-persatu dari awal node sampai akhir node. Penelusuran ini dilakukan dengan menggunakan suatu pointer bantu, karena pada prinsipnya pointer head yang menjadi tanda awal list tidak boleh berubah/berganti posisi.Penelusuran dilakukan terus sampai node terakhir ditemukan menunjuk ke nilai NULL. Jika tidak NULL, maka node bantu akan berpindah ke node selanjutnya dan membaca isi datanya dengan menggunakan field next sehingga dapat saling berkait.Jika head masih NULL berarti data masih kosong!

Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penggunaan suatu pointer lain (hapus) yang digunakan untuk menunjuk node yang akan dihapus, barulah kemusian menghapus pointer menggunakan perintah delete. Sebelum data terdepan terhapus, terlebih dahulu head harus menunhuk ke alamat berikutnya agar list tidak putus, jika head masih NULL berarti data masih kosong.

1. **SLLNC dengan HEAD Function untuk menghapus data terdepan**

void hapusDepan (){TNode \*hapus;

int d;

if (isEmpty()==0){if(head->next != NULL){hapus = head;

d = hapus->data;head = head->next;

delete hapus;

} else {

d = head->data;

head = NULL;

}

printf(“%d terhapus\n“,d);

} else cout<<"Masih kosong\n";

1. **SLLNC dengan HEAD menghapus data di belakang**

Function di atas akan menghapus data teratas (pertama) yang ditunjuk oleh head pada linked listPenghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penggunakan suatu pointer lain yang digunakan untuk menunjuk node yang akan dihapus, misalnya pointer hapus dan barulah kemudian menghapus pointer hapus dengan menggunakan perintah delete.Sebelum data terdepan dihapus, head harus ditunjukkan ke node sesudahnya terlebih dahulu agar list tidak putus, sehingga node setelah head lama akan menjadi head baru (data terdepan yang baru).Jika head masih NULL maka berarti data masih kosong!

Membutuhkan pointer bantu dan hapus. Pointer hapus digunakan untuk menunjuk node yang akan dihapus, Pointer bantu untuk menunjuk node sebelum node yang akan dihapus yang akan menjadi node yang terakhir. Pointer bantu digunakan untuk menunjuk ke nilai NULL selalu bergerak sampai sebelum node yang akan dihapus kemudian pointer hapus diletakan setelah pointer bantu. Selanjutnya pointer hapus akan menunjuk ke NULL.

**SLLNC dengan HEAD Hapus Belakang void hapusBelakang(){**

TNode \*hapus,\*bantu;

int d;

if (isEmpty()==0){if(head->next != NULL){bantu = head;

while(bantu->next->next!=NULL){bantu = bantu->next;

}

hapus = bantu->next;

d = hapus->data;bantu->next = NULL;

delete hapus;

} else {

d = head->data;

head = NULL;printf(“%d terhapus\n“,d);

} else printf(“Masih kosong\n“);

1. **SLLNC dengan HEAD Membutuhkan pointer bantu dan hapus.**

Pointer hapus digunakan untuk menunjuk node yang akan dihapus, dan pointer bantu digunakan untuk menunjuk node sebelum node yang dihapus yang kemudian selanjutnya akan menjadi node terakhir.Pointer bantu akan digunakan untuk menunjuk ke nilai NULL.Pointer bantu akan selalu bergerak sampai sebelum node yang akan dihapus, baru kemudian pointer hapus diletakkan setelah pointer bantu. Setelah itu pointer hapus akan dihapus, pointe bantu akan menunjuk ke NULL.

1. **SLLNC dengan HEAD Function untuk menghapus semua elemen Linked List**

void clear(){

TNode \*bantu,\*hapus;

bantu = head;

while(bantu!=NULL){hapus = bantu;

bantu = bantu->next;delete hapus;

}

head = NULL;

1. **SLLNC dengan HEAD & TAIL**

Dibutuhkan dua buah variabel pointer: head dan tailHead akan selalu menunjuk pada node pertama, sedangkan tail akan selalu menunjuk pada node terakhir. Dibutuhkan dua variabel pointer head dan tail. Head selalu menunjuk ke node pertama, sedangkan tail selalu menunjuk ke node terakhir

Kelebihan dari single liked list dengan heag dan tail adalah pada penambahan data di belakang, hanya dibutuhkan tail yang mengikat node baru saja tanpa harus menggunakan perulangan pointer bantu.

Inisialisasi LinkedList

TNode \*head, \*tail;

Fungsi Inisialisasi Linked

Listvoid init(){head = NULL;

tail = NULL;

}

Function untuk mengetahui kosong tidaknya Linked List

int isEmpty(){

if(tail == NULL) return 1;

else return 0;

1. **SLLNC dengan HEAD & TAIL menghapus di depan**

Penghapusan node tidak boleh dilakukan jika keadaan node sadang ditunjuk oleh pointer, maka harus dilakukan penunjukan terlebih dahulu dengan pointer hapus pada head, kemudian dilakukan pergeseran head ke node berikutnya sehingga data setelah head menjadi head baru, kemudian menghapus pointer hapus dengan menggunakan perintak delete. Jika tail masih NULL maka berarti list masih kosong.

Pengkaitan node baru ke linked list di depan

Penambahan data baru di depan akan selalu menjadi head.

void insertDepan(int databaru){

TNode \*baru;

baru = new TNode;

baru->data = databaru;

baru->next = NULL;

if(isEmpty()==1){head=tail=baru;tail->next=NULL;

}

else {

baru->next = head;

head = baru;

printf(”Data masuk\n”);

Function untuk menghapus data di depanvoid hapusDepan(){

TNode \*hapus;

int d;

if (isEmpty()==0){if(head!=tail){hapus = head;

d = hapus->data;

head = head->next;

delete hapus;

}

else {d = tail->data;

head=tail=NULL;

}printf(“%d terhapus\n“,d);

} else printf("Masih kosong\n“);

Function di atas akan menghapus data terdepan (pertama) yang ditunjuk oleh head pada linked listPenghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan pointer hapus pada head, kemudian dilakukan pergeseran head ke node berikutnya sehingga data setelah head menjadi head baru, kemudian menghapus pointer hapus dengan menggunakan perintah delete.Jika tail masih NULL maka berarti list masih kosong!

1. **SLLNC dengan HEAD & TAIL menghapus di belakang**

Penghapusan tidak boleh dilakukan jika keasaan node dedang ditunjuk oleh pointer, maka harus dilakukan penunjukan terlebih dahulu dengan mengginakan variabel hapus pada tail. JIka Tail masih NULL maka berarti list masih kosong. Dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya dampai debelim tail, sehingga tail dapat ditunjukan ke bantu, dan bantu tersebut akan nenjadi tail yang baru. Setelah itu hapus pointer hapus dengan menggunakan menggunakn perintah delete.

Penambahan Data di belakangvoid tambahBelakang(int databaru){

TNode \*baru,\*bantu;

baru = new TNode;

baru->data = databaru;

baru->next = NULL;

if(isEmpty()==1){head=baru;

tail=baru;

tail->next = NULL;

}

else {

tail->next = baru;

printf("Data masuk\n“);

Kelebihan dari Single Linked List dengan Head & Tail adalah pada penambahan data di belakang, hanya dibutuhkan tail yang mengikat node baru saja tanpa harus menggunakan perulangan pointer bantu.Function untuk menampilkan isi linked list:

void tampil(){

TNode \*bantu;

bantu = head;

if(isEmpty()==0){while(bantu!=NULL){printf(“%d\n”,bantu->data);

bantu=bantu->next;

}

printf(“\n”);

} else printf(“Masih kosong\n“);

Function untuk menghapus data di belakang:void hapusBelakang(){

TNode \*bantu,\*hapus;

int d;

if (isEmpty()==0){bantu = head;

if(head!=tail){while(bantu->next!=tail){bantu = bantu->next;

}hapus = tail;

tail=bantu;

d = hapus->data;

delete hapus;

tail->next = NULL;

}

else {

d = tail->data;

head=tail=NULL;

cout<<d<<" terhapus\n";

} else cout<<"Masih kosong\n";

Function di atas akan menghapus data terbelakang (terakhir) yang ditunjuk oleh tail pada linked listPenghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan variabel hapus pada tail, kemudian dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya sampai sebelum tail, sehingga tail dapat ditunjukkan ke bantu tersebut, dan bantu tersebut akan menjadi tail yang baru. Setelah itu hapus pointer hapus dengan menggunakan perintah delete.Jika tail masih NULL maka berarti list masih kosong!

Function untuk menghapus semua elemen LinkedListvoid clear(){

TNode \*bantu,\*hapus;

bantu = head;

while(bantu!=NULL){hapus = bantu;

bantu = bantu->next;

delete hapus;

}

head = NULL;

tail = NULL;

## 2.4 SINGLE LINKED LIST CIRCULAR

Single Linked List Circular adalah Single Linked List yang pointer nextnya menunjuk pada dirinya sendiri. Jika Single Linked List tersebut terdiri dari beberapa node,

maka pointer next pada node terakhir akan menunjuk ke node terdepannya.

Pengertian:

Single : artinya field pointer-nya hanya satu buah saja dan satu arah.

Circular : artinya pointer next-nya akan menunjuk pada dirinya sendiri sehingga berputar

Ilustrasi Single Linked List Circular

- Setiap node pada linked list mempunyai field yang berisi pointer ke node

  berikutnya, dan juga memiliki field yang berisi data.

- Pada akhir linked list, node terakhir akan menunjuk ke node terdepan

  sehingga linked list tersebut berputar. Node terakhir akan menunjuk lagi

  ke head.

PEMBUATAN SINGLE LINKED LIST CIRCULAR

Deklarasi node

Dibuat dari struct berikut ini:

**typedef struct TNode{**

**int data;**

**TNode \*next;**

**};**

**Penjelasan:**

- Pembuatan struct bernama TNode yang berisi 2 field, yaitu field data

  bertipe integer dan field next yang bertipe pointer dari TNode

- Setelah pembuatan struct, buat variabel haed yang bertipe pointer dari

  TNode yang berguna sebagai kepala linked list.

**Pembentukan node baru**

Digunakan keyword new yang berarti mempersiapkan sebuah node baru

berserta alokasi memorinya.

TNode \*baru;

baru = new TNode;

baru->data = databaru;

baru->next = baru;

1. **SINGLE LINKED LIST CIRCULAR MENGGUNAKAN HEAD**

- Dibutuhkan satu buah variabel pointer: head

- Head akan selalu menunjuk pada node pertama

Deklarasi Pointer Penunjuk Kepala Single Linked List

Manipulasi linked list tidak bisa dilakukan langsung ke node yang dituju,

melainkan harus melalui node pertama dalam linked list. Deklarasinya sebagai berikut:

TNode \*head;

Fungsi Inisialisasi Single LinkedList

void init(){

head = NULL;

}

Function untuk mengetahui kosong tidaknya Single LinkedList

int isEmpty(){

if(head == NULL) return 1;

else return 0;

}

**PENAMBAHAN DATA**

Penambahan data di depan

Penambahan node baru akan dikaitan di node paling depan, namun pada saat

pertama kali (data masih kosong), maka penambahan data dilakukan pada

head nya.

Pada prinsipnya adalah mengkaitkan data baru dengan head, kemudian head

akan menunjuk pada data baru tersebut sehingga head akan tetap selalu

menjadi data terdepan. Untuk menghubungkan node terakhir dengan node

terdepan dibutuhkan pointer bantu.

void insertDepan(int databaru){

TNode \*baru,\*bantu;

baru = new TNode;

baru->data = databaru;

baru->next = baru;

if(isEmpty()==1){

head=baru;

head->next=head;

}

else {

bantu = head;

while(bantu->next!=head){

bantu=bantu->next;

}

baru->next = head;

head = baru;

bantu->next = head;

}

cout<<"Data masuk\n";

}

Penambahan data di belakang Penambahan data dilakukan di belakang, namun pada saat pertama kali data langsung ditunjuk pada head-nya. Penambahan di belakang lebih sulit karena kita membutuhkan pointer bantu untuk mengetahui data terbelakang, kemudian dikaitkan dengan data baru. Untuk mengetahui data terbelakang perlu digunakan perulangan.

void insertBelakang (int databaru)

{

TNode \*baru,\*bantu;

baru = new TNode;

baru->data = databaru;

baru->next = baru;

if(isEmpty()==1){

head=baru;

head->next=head;

}

else {

bantu = head;

while(bantu->next != head){

bantu=bantu->next;

}

bantu->next = baru;

baru->next = head;

}

cout<<"Data masuk\n";

}

“Bagaimana dengan penambahan di tengah?”

MENAMPILKAN DATA

Function untuk menampilkan isi single linked list

void tampil(){ TNode \*b;

b = head;

if(isEmpty()==0)

{

do

{

cout<data<<" ";

b=b->next;

}

while(b!=head);

cout<<<"Masih kosong\n";

}

- Function di atas digunakan untuk menampilkan semua isi list, di mana linked list ditelusuri satu-persatu dari awal node sampai akhir node. Penelusuran ini dilakukan dengan menggunakan suatu variabel node bantu, karena pada prinsipnya variabel node head yang menjadi tanda awal list tidak boleh berubah/berganti posisi.

- Penelusuran dilakukan terus sampai node terakhir ditemukan menunjuk ke head lagi. Jika belum sama dengan head, maka node bantu akan berpindah ke node selanjutnya dan membaca isi datanya dengan menggunakan field next sehingga dapat saling berkait.

- Jika head masih NULL berarti data masih kosong!

**PENGHAPUSAN DATA**

Function untuk menghapus data terdepan

void hapusDepan ()

{ TNode \*hapus,\*bantu;

if (isEmpty()==0)

{

int d;

hapus = head; d = head->data;

if(head->next != head){

bantu = head;

while(bantu->next!=head){

bantu=bantu->next;

}

head = head->next;

delete hapus;

bantu->next = head;

}else{

head=NULL;

}

cout<<<" terhapus\n";

}

else cout<<"Masih kosong\n";

}

- Function di atas akan menghapus data teratas (pertama) yang ditunjuk oleh head  pada linked list

- Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus ditampung dahulu pada variabel hapus dan barulah kemudian menghapus variabel hapus dengan menggunakan perintah delete.

- Sebelum data terdepan dihapus, head harus ditunjukkan ke data sesudahnya  terlebih dahulu sehingga data setelah head lama akan menjadi head baru (data terdepan yang baru).

- Jika head masih NULL maka berarti data masih kosong!

Penghapusan data di belakang:

void hapusBelakang()

{ TNode \*hapus,\*bantu;

if (isEmpty()==0)

{

int d;

hapus = head;

if(head->next == head){

head = NULL;

}

else

{

bantu = head;

while(bantu->next->next != head){

bantu = bantu->next;

}

hapus = bantu->next;

d = bantu->data;

bantu->next = head;

delete hapus;

}

cout<<<" terhapus\n";

}

else

cout<<"Masih kosong\n";

}

- Membutuhkan pointer bantu dan hapus.

- Pointer hapus digunakan untuk menunjuk node yang akan dihapus, dan pointer bantu digunakan untuk menunjuk node sebelum node yang dihapus.

- Pointer bantu akan digunakan untuk menunjuk ke nilai NULL.

- Pointer bantu akan selalu bergerak bersama dengan pointer hapus tapi letak pointer bantu harus selalu dibelakang pointer hapus.

Function untuk menghapus semua elemen Linked List

void clear(){ TNode \*bantu,\*hapus;

bantu = head;

while(bantu->next!=head){

hapus = bantu;

bantu = bantu->next;

delete hapus;

}

head = NULL;

}

1. **SINGLE LINKED LIST MENGGUNAKAN HEAD DAN TAIL**

- Dibutuhkan dua buah variabel pointer: head dan tail

- Head akan selalu menunjuk pada node pertama, sedangkan tail akan

selalu menunjuk pada node terakhir.

Inisialisasi LinkedList

TNode \*head, \*tail;

Fungsi Inisialisasi LinkedList

void init(){

head = NULL;

tail = NULL;

}

Function untuk mengetahui kosong tidaknya LinkedList

int isEmpty(){

if(tail == NULL) return 1;

else return 0;

}

PENAMBAHAN DATA

Pengkaitan node baru ke linked list di depan

Penambahan data baru di depan akan selalu menjadi head.

void insertDepan(int databaru){

TNode \*baru;

baru = new TNode;

baru->data = databaru;

baru->next = baru;

if(isEmpty()==1){

head=baru;

tail=baru;

head->next=head;

tail->next=tail;

}

else {

baru->next = head;

head = baru;

tail->next = head;

}

cout<<"Data masuk\n";

}

Penambahan Data di belakang Pada penambahan data di belakang, data akan selalu dikaitkan dengan tail, karena tail terletak di node paling belakang. Setelah dikaitkan dengan node baru, maka node baru tersebut akan menjadi tail.

void tambahBelakang(int databaru){ TNode \*baru;

baru = new TNode;

baru->data = databaru;

baru->next = baru;

if(isEmpty()==1){

head=baru;

tail=baru;

head->next=head;

tail->next=tail;

}

else

{

tail->next = baru;

tail = baru;

tail->next = head;

}

cout<<"Data masuk\n";

}

Kelebihan dari Single Linked List dengan Head & Tail adalah pada penambahan data di belakang, hanya dibutuhkan tail yang mengikat node baru saja tanpa harus menggunakan perulangan pointer bantu.

Function untuk menampilkan isi linked list:

void tampil(){ TNode \*b;

b = head; if(isEmpty()==0)

{

do

{ cout<data<<" ";

b=b->next;

}

while(b!=tail->next);

cout<<<"Masih kosong\n";

}

Pada prinsipnya sama dengan function tampil sebelumnya.

Function untuk menghapus data di depan

void hapusDepan(){ TNode \*hapus;

if (isEmpty()==0){ int d;

hapus = head;

d = head->data;

if(head != tail){

hapus = head;

head = head->next;

tail->next = head;

delete hapus;

}else{

head=NULL;

tail=NULL;

}

cout<<<" terhapus\n";

}

else cout<<"Masih kosong\n";

}

- Function di atas akan menghapus data terdepan (pertama) yang ditunjuk oleh head pada linked list

- Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan variabel hapus pada head, kemudian dilakukan pergeseran ke node berikutnya sehingga data setelah head menjadi head baru, kemudian menghapus variabel hapus dengan menggunakan perintah delete.

- Jika tail masih NULL maka berarti data masih kosong!

Function untuk menghapus data di belakang:

Dengan menggunakan Single Linked List ber-Head dan Tail, pengahapusan data di belakang akan mudah dilakukan, tidak seperti pada Single Linked List hanya ber-Head saja.

void hapusBelakang(){ TNode \*hapus,\*bantu;

if (isEmpty()==0){ int d;

if(head == tail){ d = tail->data;

head = NULL;

tail = NULL;

}

else

{

bantu = head;

while(bantu->next != tail){

bantu = bantu->next;

}

hapus = tail;

tail = bantu;

d = hapus->data;

tail->next = head;

delete hapus;

}

cout<<<" terhapus\n";

}

else cout<<"Masih kosong\n";

}

- Function di atas akan menghapus data terbelakang (terakhir) yang ditunjuk oleh tail pada linked list

- Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan variabel hapus pada tail, kemudian dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya sampai sebelum tail, sehingga tail dapat ditunjukkan ke bantu tersebut, dan bantu tersebut akan menjadi tail yang baru. Setelah itu hapus variabel hapus dengan menggunakan perintah delete.

- Jika tail masih NULL maka berarti data masih kosong!

Function untuk menghapus semua elemen LinkedList

void clear(){ TNode \*bantu,\*hapus;

if(isEmpty() == 0){ bantu = head;

while(bantu->next!=head){

hapus = bantu;

bantu = bantu->next;

delete hapus;

}

head = NULL;

tail = NULL;

}

}

- Menggunakan pointer bantu yang digunakan untuk bergerak sepanjang

list, dan menggunakan pointer hapus yang digunakan untuk menunjuk

node-node yang akan dihapus.

- Pada saat pointer hapus menunjuk pada node yang akan dihapus, pointer

bantu akan bergerak ke node selanjutnya, dan kemudian pointer hapus

akan didelete.

## 2.5 CONTOH PROGRAM

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

void insertBelakang();

void menu();  
void createDepanHead();

void hapusBelakang();

void hapusDepan();  
  
struct node{    //deklarasi linked list

    int info; //data tipe integer

    struct node \*link;//pointer link

};  
  
  
struct node \*first=NULL,\*last=NULL;

void createDepanHead();

void display();

int isEmpty();  
  
void main()

{

    menu();

}  
  
void menu()

{

    system("cls");

    int choice;

    printf("\n\nEnter the choice:\n");

    printf("\n\t1. Tambah Depan\n\t2. Display ...\n\t3. Tambah Belakang\n\t4. Hapus Depan\n\t5.

Hapus Belakang\n");

    printf("\nchoose :");

    scanf("%d",&choice);

        switch(choice){

            case 1: createDepanHead(); break;  
            case 2: display(); break;  
            case 3: insertBelakang();break;  
            case 4: hapusDepan();break;  
            case 5: hapusBelakang();break;

            default: system("cls"); printf("\n\n\n\n\n\t\t\t !!! invalid choice!!!\n");

            getch();

            menu();

        }

}

int isEmpty(){//fungsi untuk mengetahui

//apakah LL kosong atau ada isinya

if(first == NULL) return 1;//jika firts==

//NULL maka return=1

else return 0;//selain itu maka return 0

}

void createDepanHead(){

    struct node \*new1=malloc(sizeof(struct node));

    printf("Enter the item tobe created: ");

    scanf("%d", &new1->info);

    new1->link=NULL;    if(isEmpty()==1){

  first=new1;

  first->link = NULL;

  }    else {

new1->link = first;

first = new1;

  }    printf("Data masuk\n");getch();

{menu();

}  
  
}

void display(){

struct node \*bantu;

bantu = first;

if(isEmpty()==0){

while(bantu!=NULL){

printf(" %d-> ",bantu->info);

bantu=bantu->link;

}printf("NULL");

printf("\n");

} else printf("Masih kosong\n");

getch();

{

    menu();

}

}  
  
void insertBelakang()

{

 struct node \*bantu,\*new1=malloc(sizeof(struct node));

    printf("Enter the item tobe created: ");

    scanf("%d", &new1->info);

    new1->link=NULL;

  if(isEmpty()==1){

  first=new1;

  first->link = NULL;

  }    else {        bantu=first;

      while(bantu->link!=NULL){        bantu=bantu->link=new1;

}

        bantu->link=new1;

    }    printf("Data masuk\n");

getch();

{menu();

}

}

void hapusDepan (){

struct node \*hapus;

int d;

if (isEmpty()==0){

 if(first->link != NULL){

hapus = first;

d = hapus->info;

first = first->link;

 } else {

d = first->info;

first = NULL;

 }

 printf("Data %d terhapus\n",d);

} else printf("Masih kosong\n");getch();

{

    menu();

}

}

void hapusBelakang(){

struct node \*hapus,\*bantu;

int d;

if (isEmpty()==0){

 if(first->link != NULL){

bantu = first;

while(bantu->link->link!=NULL){

  bantu = bantu->link;

}

hapus = bantu->link;

d = hapus->info;

       bantu->link = NULL;

 } else {

d = first->info;

first = NULL;

 }

 printf("%d terhapus\n",d);

} else printf("Masih kosong\n");getch();

{

    menu();

}

}

# BAB III

# PENUTUP

Demikianlah Makalah ini saya buat dengan sebaik-baiknya. Terima kasih buat Orang tua yang selalu mendukung saya baik materil maupun moril,dan terima kasih untuk abang dan kakak assisten laboratorium paket applikasi, juga terima kasih buat kawan-kawan yang telah membantu dalam mengerjakan makalah ini.semoga makalah ini dapat berguna buat kita semua dan buat angkatan ke depannya kelak.

lebih dan kurangnya saya mohon maaf, kepada abang dan kakak saya ucapkan terima kasih

Wassalammu’alaikum Wr.Wb.

# DAFTAR PUSTAKA

1. <http://christ-13.blogspot.com/2012/07/single-linked-list-circular-dan-non.html>
2. <https://slideplayer.info/slide/3272830/>
3. <https://prezi.com/cafacxrurcxu/single-linked-list-non-circular-sllnc/>
4. <http://taufikkipo.blogspot.com/2012/07/single-linked-list-non-circular.html>
5. <file:///C:/Users/ASUS/Downloads/Documents/ASD-Modul-6-Linked-List.pdf>
6. <https://www.belajarstatistik.com/blog/2022/01/25/deklarasi-single-linked-list-dalam-bahasa-c/>