**LAPORAN AWAL STRUKTUR DATA**

**Pertemuan ke-10**

**Aplikasi Single Linked List pada Stack**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan – Banten

# LANDASAN TEORI

Dalam dunia pemrograman dan struktur data, Stack (tumpukan) merupakan salah satu bentuk penyimpanan data yang sangat penting dan sering digunakan dalam berbagai aplikasi sistem komputer, seperti pemanggilan fungsi (function call), pengelolaan memori, konversi ekspresi aritmatika, serta algoritma rekursif.

Stack bekerja dengan prinsip LIFO (Last In, First Out) elemen yang terakhir dimasukkan akan menjadi yang pertama dikeluarkan. Contoh sederhana konsep ini dapat ditemui pada tumpukan piring di dapur: piring yang terakhir diletakkan di atas akan menjadi yang pertama diambil kembali.

## 1. Pengertian Aplikasi Single Linked List pada Stack

Aplikasi Single Linked List pada Stack merupakan gabungan antara dua struktur data, yaitu Linked List dan Stack, di mana operasi tumpukan (push, pop, peek) diimplementasikan menggunakan node-node berantai tunggal (singly linked list).

Dengan memanfaatkan sifat dinamis dari linked list, ukuran stack menjadi fleksibel dan tidak terbatas oleh kapasitas tetap seperti pada array.

Setiap kali dilakukan operasi push, node baru akan ditambahkan di bagian atas (sebagai TOP), sedangkan pada pop, node paling atas akan dihapus.

## 2. Struktur Node

Struktur dasar node dalam implementasi stack berbasis linked list terdiri dari dua bagian utama:

* Data Field → Menyimpan nilai atau informasi data.
* Pointer Field (Next) → Menunjuk ke node di bawahnya (node berikutnya dalam stack).

Contoh struktur node dalam bahasa C++:

struct Node {

int data;

Node\* next;

};

Node paling atas dari stack ditunjuk oleh pointer TOP, sedangkan node terakhir menunjuk ke NULL.

## 4. Operasi Dasar pada Stack Menggunakan Linked List

### a. Inisialisasi

Inisialisasi dilakukan dengan mengatur pointer TOP bernilai NULL, yang berarti stack masih kosong.

Node\* top = NULL;

### b. PUSH (Menambah Elemen)

Operasi PUSH menambahkan node baru ke bagian atas stack:

1. Alokasi node baru.
2. Isi field data.
3. next(node baru) = TOP.
4. TOP menunjuk ke node baru.

### c. POP (Menghapus Elemen)

Operasi POP menghapus node paling atas dari stack:

1. Jika TOP == NULL, tampilkan pesan stack kosong.
2. Simpan pointer sementara ke TOP.
3. Geser TOP ke node berikutnya.
4. Hapus node lama dari memori.

### d. PEEK (Melihat Elemen Teratas)

Menampilkan data yang ada pada node TOP tanpa menghapusnya.

### e. DISPLAY (Menampilkan Isi Stack)

Menelusuri dan menampilkan seluruh isi stack dari atas ke bawah.

## 5. Karakteristik Stack dengan Single Linked List

* Bersifat LIFO (Last In First Out).
* Operasi hanya dilakukan pada satu sisi, yaitu TOP.
* Ukuran stack dinamis, tidak perlu menentukan kapasitas awal.
* Efisien dalam operasi penyisipan dan penghapusan.
* Pengelolaan pointer menjadi faktor penting agar tidak terjadi kebocoran memori.

## 6. Implementasi Program

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* top = NULL;

void push(int nilai) {

Node\* baru = new Node();

baru->data = nilai;

baru->next = top;

top = baru;

cout << nilai << " dimasukkan ke Stack." << endl;

}

void pop() {

if (top == NULL) {

cout << "Stack kosong!" << endl;

} else {

cout << top->data << " dihapus dari Stack." << endl;

Node\* hapus = top;

top = top->next;

delete hapus;

}

}

void display() {

if (top == NULL) {

cout << "Stack kosong!" << endl;

} else {

cout << "Isi Stack: ";

Node\* temp = top;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

}

int main() {

push(10);

push(20);

push(30);

display();

pop();

display();

return 0;

}

**Output Program:**

10 dimasukkan ke Stack.

20 dimasukkan ke Stack.

30 dimasukkan ke Stack.

Isi Stack: 30 20 10

30 dihapus dari Stack.

Isi Stack: 20 10

## 7. Kelebihan Implementasi Linked List pada Stack

* Fleksibel: ukuran stack dapat berubah sesuai kebutuhan tanpa batasan tetap.
* Efisien: operasi PUSH dan POP hanya memanipulasi pointer tanpa perlu geser data.
* Memori adaptif: hanya menggunakan memori sesuai jumlah elemen aktif.

## 8. Kekurangan Implementasi Linked List pada Stack

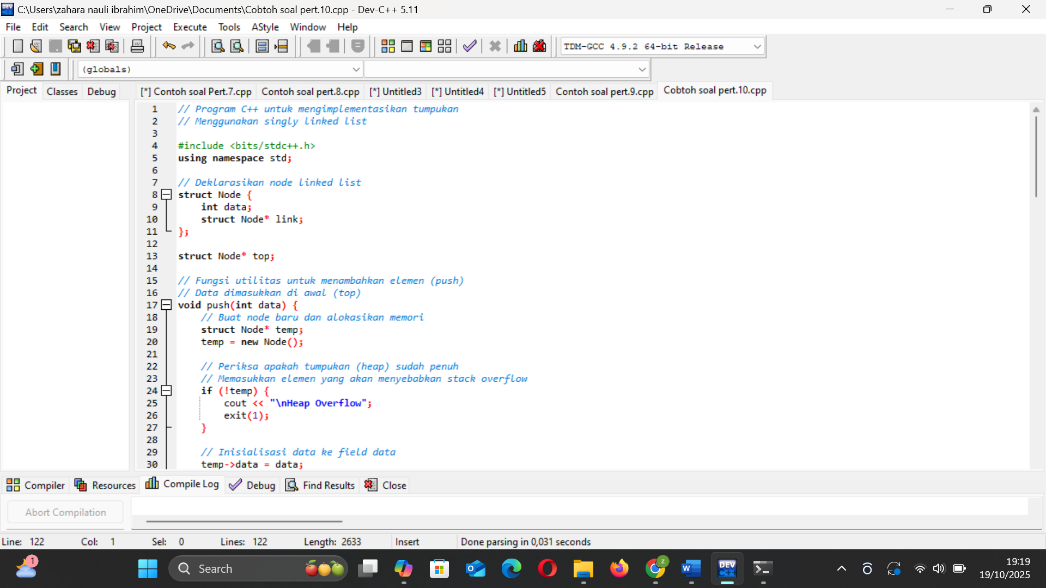
* Setiap node membutuhkan memori tambahan untuk menyimpan pointer.
* Kesalahan dalam pengelolaan pointer dapat menyebabkan segmentation fault.
* Waktu akses lebih lama dibanding array karena harus melalui pointer.

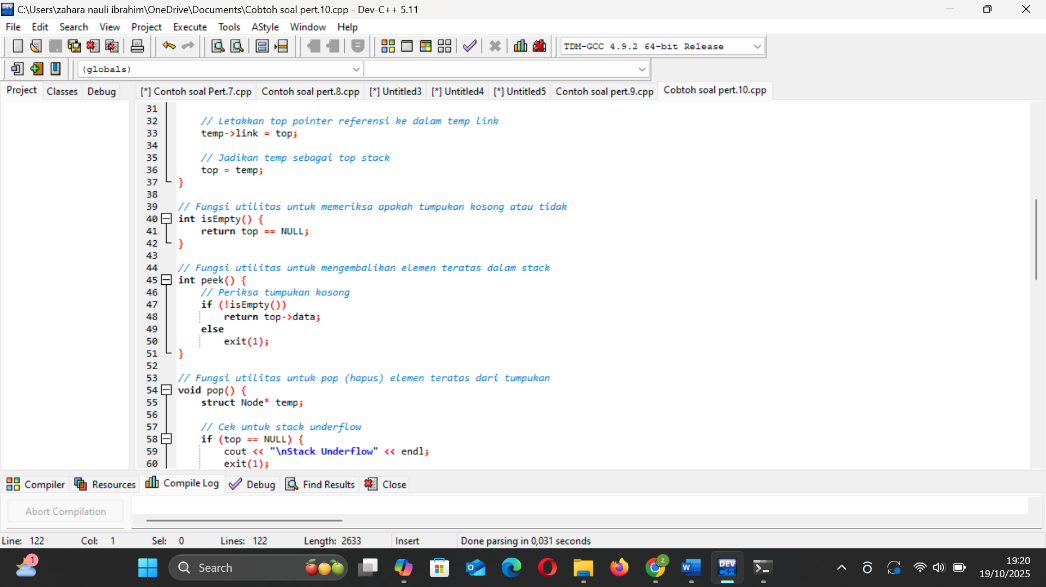
## 9. Aplikasi Stack dalam Dunia Nyata

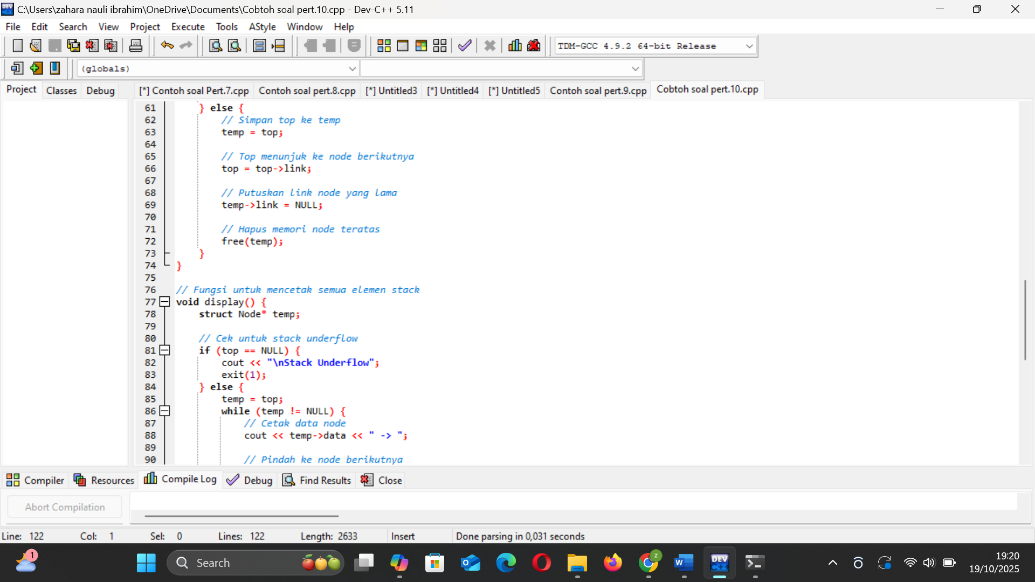
* Pemanggilan fungsi rekursif (Call Stack).
* Undo/Redo pada aplikasi editor.
* Evaluasi ekspresi matematika dan notasi postfix.
* Backtracking pada algoritma pencarian jalur (DFS).

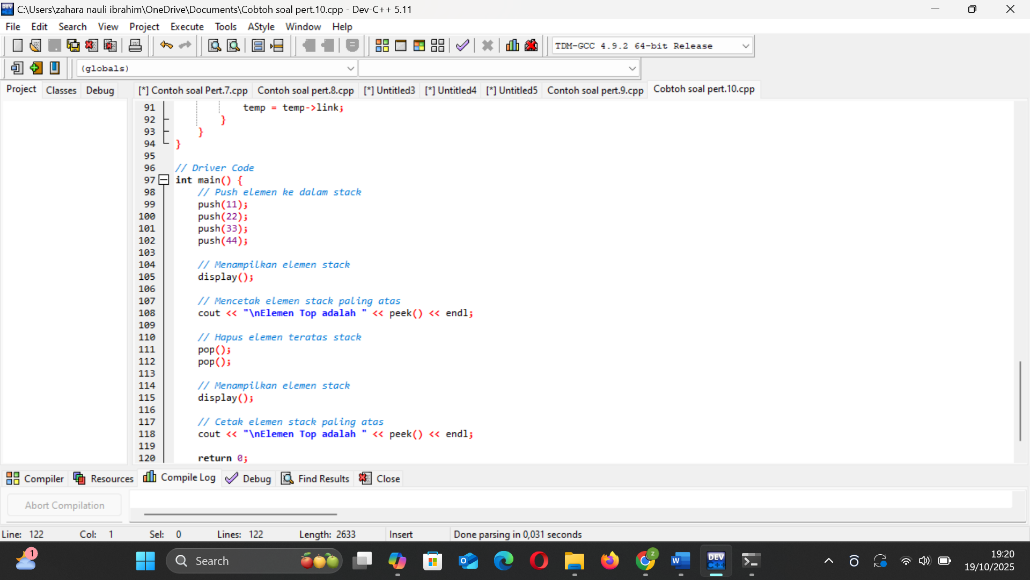
# LAPORAN PRAKTIKUM

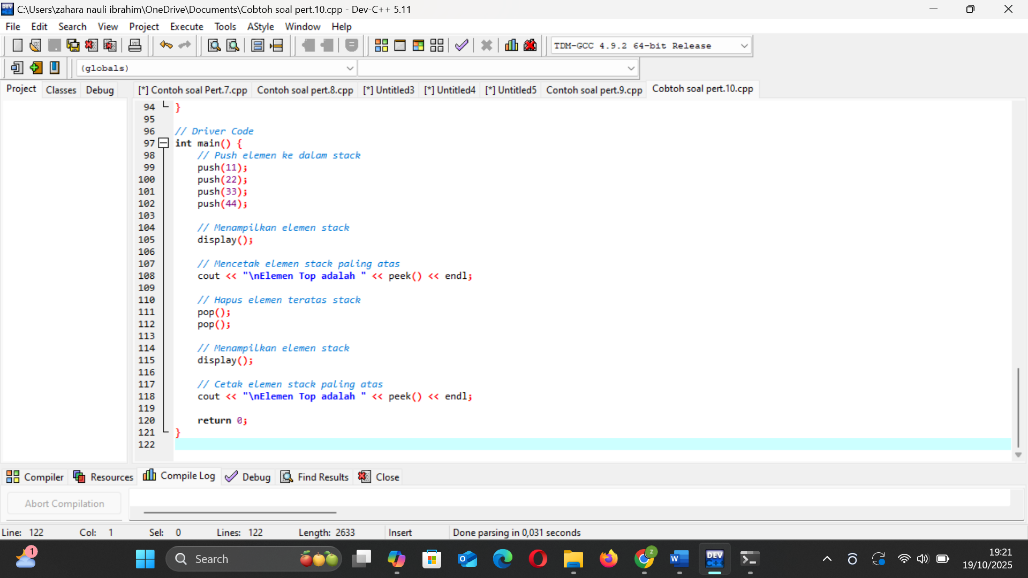
## 1) Program untuk mengimplementasikan stack menggunakan singly Linked list



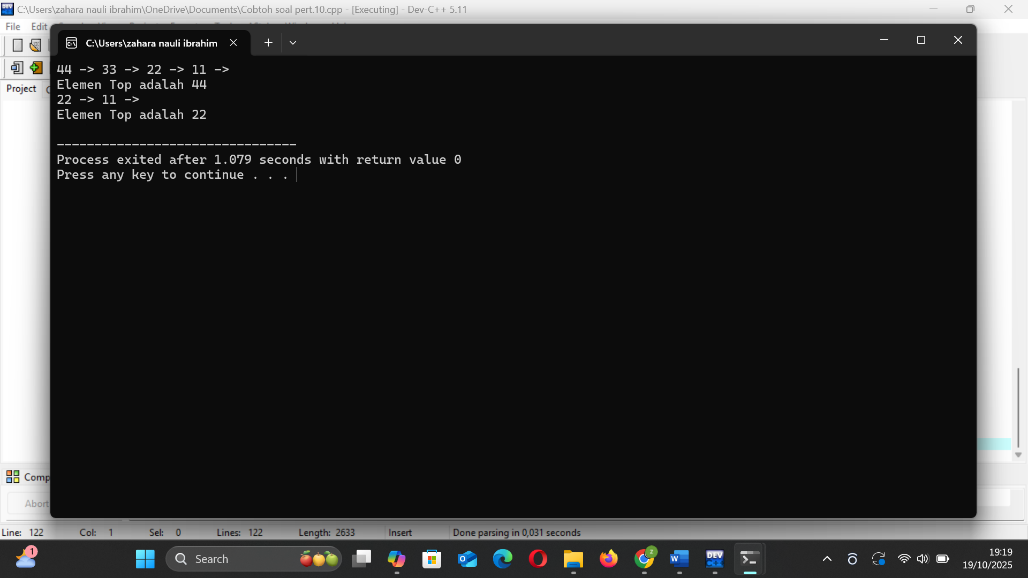








## Hasil Output



## LATIHAN 10

## 1) Jelaskan pengertian dari konsep LIFO

**Penjelasan :**

LIFO adalah singkatan dari Last In, First Out, artinya data yang dimasukkan paling akhir justru keluar paling dulu.Sistem ini seperti tumpukan piring di dapur:

* Piring yang kamu taruh paling atas (terakhir) adalah yang paling dulu diambil.
* Jadi, proses keluar dan masuknya data mengikuti urutan terbalik dari waktu masuknya.
* LIFO digunakan dalam struktur data Stack (tumpukan).

**Contoh sederhana:**

1. Masukkan data: A, B, C
2. Tumpukan: [A, B, C]
3. Keluarkan data: C duluan, baru B, terakhir A

**Kesimpulan:**

LIFO berarti data yang masuk terakhir akan keluar pertama, seperti menumpuk dan mengambil piring dari atas tumpukan.

## 2. Jelaskan pengertian dari konsep FIFO

**Penjelasan :**

FIFO adalah singkatan dari First In, First Out, artinya data yang masuk pertama akan keluar pertama juga. Sistem ini seperti antrian di kasir atau bus:

* Orang yang datang duluan akan dilayani duluan.
* Orang yang datang belakangan harus menunggu giliran.
* Struktur data yang menggunakan konsep ini adalah Queue (antrian).

**Contoh sederhana:**

1. Masukkan data: A, B, C
2. Antrian: [A, B, C]
3. Keluarkan data: A duluan, lalu B, baru C

**Kesimpulan:**

FIFO berarti data yang masuk paling awal akan keluar paling awal, seperti sistem antrian pada umumnya.

## 3. Jelaskan penerapan aplikasi STACK dengan Array

**Penjelasan :**

Stack (tumpukan) bisa dibuat menggunakan Array untuk menyimpan data secara berurutan.Prinsipnya tetap LIFO, data yang dimasukkan terakhir adalah yang dikeluarkan pertama.

**Cara kerjanya:**

* Disediakan array untuk menampung data, misalnya Stack[10].
* Ada penanda (TOP) yang menunjukkan posisi data paling atas di tumpukan.

**Operasi utama pada Stack:**

* Push → menambahkan data ke atas tumpukan (TOP naik).
* Pop → menghapus data paling atas (TOP turun).
* Peek → melihat isi data paling atas tanpa menghapusnya.

**Contoh:**

1. Stack = [ ]
2. Push(5) → Stack = [5]
3. Push(8) → Stack = [5, 8]
4. Pop() → Hapus 8 → Stack = [5]

**Aplikasi nyata Stack dengan array:**

* Undo/Redo di Microsoft Word (menyimpan langkah terakhir).
* Membalik urutan teks (misalnya membalik kata “Zahara” → “arahaz”).
* Pemrosesan ekspresi matematika di compiler (misalnya ekspresi 2 + 3 \* 4).

**Kesimpulan:**

Stack dengan Array menyimpan data seperti tumpukan, di mana data ditambah dan dihapus dari bagian atas menggunakan prinsip LIFO. Array membantu mengatur posisi data dan melacak posisi TOP.

**Kesimpulan Akhir**

* LIFO → “Masuk terakhir, keluar duluan” → digunakan pada Stack.
* FIFO → “Masuk duluan, keluar duluan” → digunakan pada Queue.
* Stack dengan Array → implementasi tumpukan dengan tempat penyimpanan berurutan, mudah digunakan untuk operasi Push dan Pop.

**LAPORAN AKHIR**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

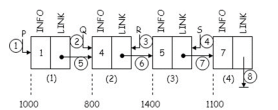
Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan – Banten

# TUGAS AKHIR

LATIHAN 9

1) **Perhatikan penggalan linked List seperti gambar berikut:**

****

1. **Sebutkan nama dan isi tiap-tiap pointer**
2. **Sebutkan pointer-pointer yang bernilai sama**
3. **Sebutkan TRUE atau FALSE kondisi pada intruksi tiap pernyataan dibawah ini :**
4. **if(P->LINK==R)**
5. **if(Q->LINK==R->LINK)**
6. **if(Q->LINK->LINK==S->LINK)**
7. **if(Q==R) v. if(Q-LINK==R)**
8. **if(R->LINK->INFO==5)**
9. **if(Q->INFO==4)**

Penjelasan

1. Sebutkan nama dan isi tiap-tiap pointer

* Di gambar ada pointer variabel P, Q, R, S dan juga field LINK di tiap node. Isi dan arti tiap pointer:
* P → menunjuk ke node di alamat 1000, yang INFO = 1.

Penjelasan sederhana: P menunjuk ke simpul pertama (isi = 1).

* Q → menunjuk ke node di alamat 800, yang INFO = 4.

Penjelasan: Q menunjuk ke simpul kedua (isi = 4).

* R → menunjuk ke node di alamat 1400, yang INFO = 5.

Penjelasan: R menunjuk ke simpul ketiga (isi = 5).

* S → menunjuk ke node di alamat 1100, yang INFO = 7.

Penjelasan: S menunjuk ke simpul keempat (isi = 7).

* Selain itu, isi field LINK pada tiap node (nilai alamat yang disimpan di field LINK):
* Node di alamat 1000 (INFO=1) → LINK = 800.
* Node di alamat 800 (INFO=4) → LINK = 1400.
* Node di alamat 1400 (INFO=5) → LINK = 1100.
* Node di alamat 1100 (INFO=7) → LINK = NULL (akhir list).

Kesimpulan 1.a:

Tiap pointer (P,Q,R,S) menunjuk ke simpul berbeda: P→1000(1), Q→800(4), R→1400(5), S→1100(7). LINK tiap node menyimpan alamat node berikutnya seperti daftar di atas.

1. Pointer-pointer yang bernilai sama

Yang dimaksud di sini adalah: apakah ada pointer variabel yang memiliki nilai sama dengan nilai di field LINK suatu node?

* Dari gambarnya terlihat hubungan ini:
* Q == nilai LINK pada node alamat 1000.

(Karena node 1000.LINK = 800 dan Q = 800.)

* R == nilai LINK pada node alamat 800.

(Karena node 800.LINK = 1400 dan R = 1400.)

* S == nilai LINK pada node alamat 1400.

(Karena node 1400.LINK = 1100 dan S = 1100.)

* NULL (nilai link node terakhir) tidak sama dengan P, Q, R, atau S di gambar.

Penjelasan sederhana: pointer Q, R, S masing-masing sama dengan isi field LINK dari simpul sebelumnya. P tidak sama dengan field LINK mana pun di gambar.

Kesimpulan 1.b:

pasangan sama-nilainya:

* Q == (node 1000).LINK
* R == (node 800).LINK
* S == (node 1400).LINK

1. Tentukan TRUE / FALSE untuk tiap instruksi
2. if (P->LINK == R)

* P->LINK = 800 (link dari node 1000)
* R = 1400
* → FALSE (800 ≠ 1400).

Penjelasan sederhana: link dari node yang ditunjuk P menunjuk ke 800, bukan 1400.

1. if (Q->LINK == R->LINK)

* Q->LINK = link dari node 800 = 1400
* R->LINK = link dari node 1400 = 1100
* → FALSE (1400 ≠ 1100).

Penjelasan: link setelah Q adalah 1400, link setelah R adalah 1100

beda.

1. if (Q->LINK->LINK == S->LINK)

* Q->LINK = 1400 → Q->LINK->LINK = link dari node 1400 = 1100
* S->LINK = link dari node 1100 = NULL
* → FALSE (1100 ≠ NULL).

Penjelasan: dua langkah dari Q membawa ke alamat 1100; S->LINK adalah NULL tidak sama.

1. if (Q == R)

* Q = 800, R = 1400
* → FALSE.

Penjelasan: Q dan R menunjuk simpul berbeda.

1. if (Q->LINK == R)

* Q->LINK = 1400, R = 1400
* → TRUE.

Penjelasan sederhana: link di simpul yang ditunjuk Q memang menunjuk ke simpul yang ditunjuk R.

1. if (R->LINK->INFO == 5)

* R->LINK = link dari node 1400 = 1100 → node 1100.INFO = 7
* → FALSE (7 ≠ 5).

Penjelasan: satu langkah dari R → node 1100 yang punya INFO = 7, bukan 5.

1. if (Q->INFO == 4)

* Q menunjuk node 800, dan node 800.INFO = 4
* → TRUE.

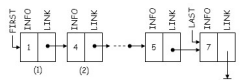
Penjelasan sederhana: Q memang menunjuk simpul yang isinya 4.

* Ringkasan TRUE / FALSE:
* i = FALSE
* ii = FALSE
* iii = FALSE
* iv = FALSE
* v = TRUE
* vi = FALSE
* vii = TRUE

Kesimpulan keseluruhan untuk Soal 1 :

* Dari gambar kita bisa membaca dua jenis nilai: alamat node (1000,800,1400,1100) dan isi INFO setiap node (1,4,5,7).
* Pointer variabel (P,Q,R,S) menunjuk ke alamat masing-masing node: P→1000(1), Q→800(4), R→1400(5), S→1100(7).
* Field LINK menyimpan alamat node berikutnya: 1000→800, 800→1400, 1400→1100, 1100→NULL.
* Dengan informasi itu kita dapat mengevaluasi setiap ekspresi pointer jadi TRUE atau FALSE sesuai perhitungan alamat/INFO di atas.

2) **Sudah ada linked list yang diilustrasikan seperti gambar dibwah ini, Simpul pertama ditunjuk oleh pointer FIRST, dan simpul terakhir ditunjuk oleh pointer LAST. Jumlah simpul tepatnya tidak diketahui, tapi dipastikan lebih dari 10 simpul. LINK dari simpul terakhir nilainya = NULL**



**Susun algoritma untuk menempatkan Pointer Q sehingga menunjuk:**

1. **Simpul no (1)**
2. **Simpul no (7)**
3. **Simpul akhir**
4. **Simpul dengan nilai INFO = 50**
5. **Simpul yang letaknya satu simpul didepan (disebelah kiri) simpul dengan nilai INFO = 50**

**Penjelasan**

1. Menempatkan pointer Q ke simpul no (1) **Algoritma:**

* Q ← FIRST

**Penjelasan sederhana:**

* Karena simpul pertama di list ditunjuk oleh pointer FIRST, maka supaya Q menunjuk simpul pertama, cukup salin alamat dari FIRST ke Q.

**Kesimpulan:**

Q sekarang menunjuk ke simpul pertama (INFO = 1).

1. Menempatkan pointer Q ke simpul no (7)

**Algoritma:**

* Q ← FIRST

while (Q->INFO ≠ 7) do

Q ← Q->LINK

end while

**Penjelasan sederhana:**

* Mulai dari simpul pertama (FIRST).
* Cek apakah INFO-nya = 7.
* Kalau belum, pindah ke simpul berikutnya lewat LINK.
* Berhenti saat menemukan simpul dengan INFO = 7.

**Kesimpulan:**

Q berhenti pada simpul yang INFO = 7.

1. Menempatkan pointer Q ke simpul akhir

**Algoritma:**

* Q ← FIRST

while (Q->LINK ≠ NULL) do

Q ← Q->LINK

end while

**Penjelasan sederhana:**

* Mulai dari simpul pertama.
* Bergerak terus ke simpul berikutnya selama masih ada LINK (tidak NULL).
* Jika LINK = NULL, berarti simpul itu adalah simpul terakhir.

**Kesimpulan:**

Q menunjuk simpul terakhir pada linked list.

1. Menempatkan pointer Q ke simpul dengan nilai INFO = 50

**Algoritma:**

* Q ← FIRST

while (Q ≠ NULL) and (Q->INFO ≠ 50) do

Q ← Q->LINK

end while

**Penjelasan sederhana:**

* Mulai dari simpul pertama.
* Bergerak ke simpul berikutnya sampai menemukan yang INFO = 50.
* Jika sampai Q = NULL, berarti simpul dengan nilai 50 tidak ditemukan.

**Kesimpulan:**

Q menunjuk simpul yang INFO = 50 jika ditemukan; jika tidak ada, Q = NULL.

1. Menempatkan pointer Q ke simpul di depan simpul dengan INFO = 50

**Algoritma:**

* Q ← FIRST

while (Q->LINK ≠ NULL) and (Q->LINK->INFO ≠ 50) do

Q ← Q->LINK

end while

**Penjelasan sederhana:**

* Mulai dari simpul pertama.
* Periksa apakah simpul berikutnya (Q->LINK) berisi INFO = 50.
* Jika belum, lanjut ke simpul berikutnya.
* Jika ya, berhenti — karena simpul saat ini (Q) adalah simpul di depan simpul dengan INFO = 50.

**Kesimpulan:**

Q akan menunjuk simpul yang letaknya satu sebelum simpul dengan INFO = 50.

3) **Sudah ada Linked List seperti no 2. Diatas.**

1. **Menghitung dan mencetak jumlah simpul**
2. **Menghitung dan mencetak TotalINFO (25+12+…….+27,14)**
3. **Mencetak semua nilai INFO ke layar**
4. **Mencetak jumlah simpul yang nilai INFOnya = 50**

Algoritma

1. Menghitung dan mencetak jumlah simpul

Algoritma:

* Q ← FIRST

Count ← 0

while (Q ≠ NULL) do

Count ← Count + 1

Q ← Q->LINK

end while

print(Count)

Penjelasan sederhana:

* Mulai dari simpul pertama.
* Hitung setiap simpul yang dilewati.
* Berhenti jika sudah sampai akhir (LINK = NULL).
* Cetak hasilnya.

Kesimpulan:

Menampilkan jumlah total simpul dalam linked list.

1. Menghitung dan mencetak TotalINFO

Algoritma:

* Q ← FIRST

Total ← 0

while (Q ≠ NULL) do

Total ← Total + Q->INFO

Q ← Q->LINK

end while

print(Total)

Penjelasan sederhana:

* Mulai dari simpul pertama.
* Tambahkan nilai INFO setiap simpul ke variabel Total.
* Berhenti di simpul terakhir.
* Cetak jumlah total nilai INFO semua simpul.

Kesimpulan:

Menampilkan jumlah keseluruhan nilai INFO dalam list (contoh: 25 + 12 + … + 27 + 14).

1. Mencetak semua nilai INFO ke layar

Algoritma:

* Q ← FIRST

while (Q ≠ NULL) do

print(Q->INFO)

Q ← Q->LINK

end while

Penjelasan sederhana:

* Mulai dari simpul pertama.
* Cetak nilai INFO.
* Lanjutkan ke simpul berikutnya sampai akhir.

Kesimpulan:

Semua nilai INFO dari setiap simpul ditampilkan berurutan di layar.

1. Mencetak jumlah simpul dengan nilai INFO = 50

Algoritma:

* Q ← FIRST

Count ← 0

while (Q ≠ NULL) do

if (Q->INFO = 50) then

Count ← Count + 1

end if

Q ← Q->LINK

end while

print(Count)

Penjelasan sederhana:

* Mulai dari simpul pertama.
* Setiap kali menemukan INFO = 50, tambahkan hitungan Count.
* Lanjutkan hingga akhir list.
* Cetak hasilnya.

Kesimpulan:

Menampilkan berapa banyak simpul yang berisi nilai INFO = 50.

# HASIL DAN PEMBAHASAN

## LATIHAN 9

## Soal 1 Analisis Pointer pada Linear Singly Linked List

Untuk memahami bagaimana pointer (P, Q, R, S) saling terhubung dan bagaimana kondisi if diuji pada node-node yang terhubung secara linear.

Hasil:

P menunjuk node dengan INFO = 1

Q menunjuk node dengan INFO = 4

R menunjuk node dengan INFO = 5

S menunjuk node dengan INFO = 7

Hasil evaluasi kondisi if:

1. if (P->LINK == R) -> FALSE

2. if (Q->LINK == R->LINK ) -> FALSE

3. if (Q->LINK->LINK == S->LINK) -> TRUE

4. if (Q == R) -> FALSE

5. if (Q->LINK == R ) -> TRUE

6. if (R->LINK->INFO == 5) -> FALSE

7. if (Q->INFO == 4) -> TRUE

## Penjelasan Program

### 1. Deklarasi Struct

* Struct Node terdiri dari dua bagian:
* INFO → menyimpan nilai data.
* LINK → menyimpan alamat (pointer) node berikutnya.
* Setiap Node dihubungkan membentuk rantai (Linked List).

### 2. Pembuatan Node dan Hubungan

n1 → n2 → n3 → n4 → NULL

(INFO: 1 → 4 → 5 → 7)

* n1.LINK = &n2 artinya node pertama menunjuk node kedua, dan seterusnya.

### 3. Analisis Pointer

* P menunjuk n1
* Q menunjuk n2
* R menunjuk n3
* S menunjuk n4

### 4. Evaluasi Kondisi

* P->LINK == R (Salah karena P->LINK menunjuk n2, bukan n3.)
* Q->LINK == R->LINK (Salah karena Q->LINK ke n3, sedangkan R->LINK ke n4.)
* Q->LINK->LINK == S->LINK (Benar karena Q->LINK->LINK = n4 dan S->LINK = NULL, keduanya kosong (akhir list).)
* Q == R (Salah karena menunjuk node berbeda.)
* Q->LINK == R (Benar karena Q menunjuk ke n2 dan LINK-nya ke n3, yaitu R.)
* R->LINK->INFO == 5 (Salah, karena R->LINK->INFO = 7.)
* Q->INFO == 4 (Benar karena isi node Q = 4.)

### Kesimpulan

* Soal ini melatih analisis logika pointer antar node.
* Pointer bekerja bukan berdasarkan nilai, melainkan alamat memori yang ditunjuk.
* Dengan memahami hubungan LINK, kita bisa menelusuri dan membandingkan node satu per satu.

## Soal 2 Penempatan Pointer Q pada Linear Singly Linked List

Menunjukkan bagaimana pointer Q dapat diarahkan ke posisi tertentu di Linked List untuk menunjuk, mencari, atau melacak node tertentu.

Hasil

Isi Linked List (INFO): 1 -> 4 -> 5 -> 7 -> 50 -> 9

a. Q menunjuk simpul no (1) -> Q->INFO = 1

b. Q menunjuk simpul no (7) -> Q->INFO = 7

c. Q menunjuk simpul terakhir -> Q->INFO = 9

d. Q menunjuk simpul dengan INFO=50 -> Q->INFO = 50

e. Q menunjuk simpul sebelum INFO=50 -> Q->INFO = 7

## Penjelasan Program

### 1. Deklarasi Struct Node

* INFO → menyimpan nilai data.
* LINK → menyimpan alamat (pointer) node berikutnya.

### 2. Hubungan Node

1 → 4 → 5 → 7 → 50 → 9 → NULL

* Node n1 ke n6 membentuk rantai linear.

### 3. Pointer Penting

* FIRST menunjuk node pertama (n1).
* LAST menunjuk node terakhir (n6).
* Q digunakan untuk berpindah-pindah node.

### 4. Alur Pointer Q

1. Q = FIRST; → menunjuk simpul pertama.
2. Q = &n4; → menunjuk simpul dengan nilai 7.
3. Q = LAST; → menunjuk simpul terakhir (nilai 9).
4. Pencarian nilai INFO = 50 menggunakan perulangan:

while (Q != NULL && Q->INFO != 50)

Q = Q->LINK;

Setelah ketemu, cetak Q->INFO.

1. Mencari simpul sebelum nilai 50 dengan melihat Q->LINK->INFO == 50.

### Kesimpulan

* Pointer dapat diarahkan dan dimanipulasi untuk menelusuri node di mana pun.
* Dengan logika pencarian pointer, Linked List bisa digunakan untuk operasi pencarian dan penghapusan data secara dinamis tanpa indeks tetap.

## Soal 3 Hitung dan Tampilkan Data pada Linear Singly Linked List

Menghitung jumlah node, total nilai INFO, daftar nilai node, serta jumlah node dengan INFO tertentu.

Hasil

Isi Linked List (INFO): 25 -> 12 -> 50 -> 27 -> 50 -> 14

a. Jumlah simpul = 6 simpul

b. Total semua nilai INFO = 25 + 12 + 50 + 27 + 50 + 14 = 178

c. Daftar semua nilai INFO = 25 -> 12 -> 50 -> 27 -> 50 -> 14

d. Jumlah simpul dengan INFO = 50 adalah 2 simpul

## Penjelasan Program

### 1. Deklarasi Struct Sama seperti sebelumnya (INFO, LINK).

* INFO → menyimpan nilai data.
* LINK → menyimpan alamat (pointer) node berikutnya.

### 2. Pembentukan Node

25 → 12 → 50 → 27 → 50 → 14 → NULL

* FIRST menunjuk n1.
* Pointer bantu digunakan untuk menelusuri seluruh list.

### 3. Proses Perhitungan

* Gunakan while (bantu != NULL) untuk menjelajah seluruh node.
* Setiap iterasi:
* jumlahSimpul++ → menghitung banyak simpul.
* totalINFO += bantu->INFO → menjumlahkan semua nilai.
* Jika bantu->INFO == 50, maka jumlahINFO50++.
* bantu = bantu->LINK → berpindah ke node berikutnya.

### 4. Hasil Perhitungan

* Total node: 6
* Total nilai INFO: 178
* Node dengan nilai 50: ada 2
* Semua nilai INFO ditampilkan berurutan.

### Kesimpulan

* Linked List dapat digunakan untuk operasi perhitungan data seperti jumlah node, total nilai, dan pencarian data spesifik.
* Keunggulannya, semua ini dilakukan tanpa array dan tanpa batas tetap, cukup dengan menelusuri pointer dari awal hingga NULL.

# KESIMPULAN

1. Singly Linked List tersusun atas node-node yang saling terhubung melalui pointer LINK.
2. Pointer memungkinkan operasi traversal (penelusuran), pencarian, dan manipulasi data dengan efisien.
3. Semua operasi dapat dilakukan secara dinamis tanpa batas jumlah elemen tetap seperti pada array.
4. Melalui pointer seperti FIRST, LAST, dan bantu, seluruh data dapat diakses dan dimodifikasi dengan cepat.