**1. Stack (Tumpukan)**

**Fungsi:**

Stack adalah struktur data **LIFO (Last In First Out)**, artinya elemen terakhir yang dimasukkan akan keluar pertama.

**Operasi utama:**

* push(item) → menambahkan item ke atas stack.
* pop() → menghapus dan mengembalikan item paling atas.
* peek() → melihat item paling atas tanpa menghapusnya.

**Passing Parameter:**

Biasanya stack dipassing sebagai **objek atau referensi** ke fungsi agar perubahan bisa langsung diterapkan.

**Contoh kehidupan nyata:**

* **Undo dalam berbagai aplikasi**
* **Tumpukan piring di dapur**

**Penjelasan:**

* Saat kamu mencuci piring, piring yang baru dicuci diletakkan di atas tumpukan.
* Saat mengambil piring untuk digunakan, kamu ambil dari atas (yang terakhir ditaruh).

**Kaitan: Last In, First Out (LIFO)**  
Piring terakhir yang ditaruh adalah yang pertama diambil.

**Input**

tumpukan\_piring = []

tambah\_piring(tumpukan\_piring, "Piring A")

tambah\_piring(tumpukan\_piring, "Piring B")

print("Piring yang diambil:", ambil\_piring(tumpukan\_piring))

print("Tumpukan sekarang:", tumpukan\_piring)

**Output**

Piring 'Piring A' ditambahkan ke tumpukan.

Piring 'Piring B' ditambahkan ke tumpukan.

Piring yang diambil: Piring B

Tumpukan sekarang: ['Piring A']

# Stack sebagai list

stack = []

# Operasi Stack

def push(item):

    stack.append(item)

    print(f"{item} telah ditambahkan ke stack.")

def pop():

    if not stack:

        print("Stack kosong. Tidak bisa melakukan pop.")

    else:

        removed = stack.pop()

        print(f"{removed} telah dihapus dari stack.")

def peek():

    if not stack:

        print("Stack kosong. Tidak ada item untuk dilihat.")

    else:

        print(f"Item paling atas di stack adalah: {stack[-1]}")

def tampilkan\_stack():

    if not stack:

        print("Stack kosong.")

    else:

        print("Isi stack saat ini:", stack)

# Menu Interaktif

while True:

    print("\nMenu Stack:")

    print("1. Push (Tambah data)")

    print("2. Pop (Ambil data)")

    print("3. Peek (Lihat paling atas)")

    print("4. Tampilkan Stack")

    print("5. Keluar")

    pilihan = input("Pilih menu (1-5): ")

    if pilihan == "1":

        data = input("Masukkan data yang ingin ditambahkan: ")

        push(data)

    elif pilihan == "2":

        pop()

    elif pilihan == "3":

        peek()

    elif pilihan == "4":

        tampilkan\_stack()

    elif pilihan == "5":

        print("Terima kasih. Program selesai.")

        break

    else:

        print("Pilihan tidak valid. Silakan coba lagi.")

**2. Queue (Antrian)**

**Fungsi:**

Queue adalah struktur data **FIFO (First In First Out)**, artinya elemen pertama yang dimasukkan akan keluar pertama.

**Operasi utama:**

* enqueue(item) → menambahkan item ke belakang queue.
* dequeue() → menghapus dan mengembalikan item paling depan.

**Passing Parameter:**

Seperti Stack, Queue bisa dipassing sebagai referensi (list atau objek queue).

**Contoh kehidupan nyata:**

**Antrian di loket atau ATM**

**Penjelasan:**

* Orang pertama yang datang akan dilayani lebih dulu.
* Orang yang datang belakangan harus menunggu giliran.

**Kaitan: First In, First Out (FIFO)**  
Orang pertama yang masuk antrean, keluar duluan.

Input

antrian\_atm = deque()

tambah\_antrian(antrian\_atm, "Ali")

tambah\_antrian(antrian\_atm, "Budi")

print("Dilayani:", layani\_antrian(antrian\_atm))

print("Antrian sekarang:", list(antrian\_atm))

**Output**

Ali masuk antrian.

Budi masuk antrian.

Dilayani: Ali

Antrian sekarang: ['Budi']

# Antrian sebagai list

queue = []

# Fungsi operasi queue

def enqueue(item):

    queue.append(item)

    print(f"{item} telah ditambahkan ke antrian.")

def dequeue():

    if not queue:

        print("Antrian kosong. Tidak bisa menghapus.")

    else:

        removed = queue.pop(0)

        print(f"{removed} telah dihapus dari antrian.")

def peek():

    if not queue:

        print("Antrian kosong.")

    else:

        print(f"Item paling depan di antrian: {queue[0]}")

def tampilkan\_antrian():

    if not queue:

        print("Antrian kosong.")

    else:

        print("Isi antrian saat ini:", queue)

# Menu Interaktif

while True:

    print("\nMenu Antrian:")

    print("1. Enqueue (Tambah data)")

    print("2. Dequeue (Ambil data)")

    print("3. Peek (Lihat paling depan)")

    print("4. Tampilkan Antrian")

    print("5. Keluar")

    pilihan = input("Pilih menu (1-5): ")

    if pilihan == "1":

        data = input("Masukkan data yang ingin ditambahkan ke antrian: ")

        enqueue(data)

    elif pilihan == "2":

        dequeue()

    elif pilihan == "3":

        peek()

    elif pilihan == "4":

        tampilkan\_antrian()

    elif pilihan == "5":

        print("Terima kasih. Program selesai.")

        break

    else:

        print("Pilihan tidak valid. Silakan coba lagi.")

**3. Graph (Graf)**

**Fungsi:**

Graph adalah struktur data yang terdiri dari **simpul (node)** dan **sisi (edge)**. Digunakan untuk merepresentasikan jaringan seperti rute, hubungan sosial, dll.

**Representasi:**

* **Adjacency List** (paling umum)
* **Adjacency Matrix**

**Passing Parameter:**

Graph biasanya dipassing sebagai **dictionary atau objek**, dan diubah langsung melalui referensi.

**Contoh kehidupan nyata:**

* **Peta jalan atau Google Maps**
* **Rekomendasi Pertemanan di media sosial**

**Penjelasan:**

* Titik (node) = lokasi (seperti kota atau simpang jalan)
* Garis (edge) = jalan yang menghubungkan lokasi tersebut

**Kaitan: Menentukan rute tercepat**  
Contoh: Dari rumah ke sekolah lewat rute paling pendek menggunakan graf (seperti di GPS).

**Input**

peta\_kota = {

"A": [],

"B": [],

"C": [],

}

tambah\_jalan(peta\_kota, "A", "B")

tambah\_jalan(peta\_kota, "B", "C")

print("Peta kota:", peta\_kota)

Output

Peta kota: {'A': ['B'], 'B': ['A', 'C'], 'C': ['B']}

# Struktur graph menggunakan dictionary

graph = {}

def tambah\_teman(nama1, nama2):

    if nama1 not in graph:

        graph[nama1] = []

    if nama2 not in graph:

        graph[nama2] = []

    # Tambahkan teman dua arah

    if nama2 not in graph[nama1]:

        graph[nama1].append(nama2)

    if nama1 not in graph[nama2]:

        graph[nama2].append(nama1)

    print(f"{nama1} dan {nama2} sekarang berteman.")

def lihat\_teman(nama):

    if nama not in graph:

        print(f"{nama} belum memiliki teman.")

    else:

        teman = graph[nama]

        print(f"Teman-teman {nama}: {', '.join(teman)}")

def tampilkan\_graph():

    if not graph:

        print("Belum ada pertemanan yang tercatat.")

    else:

        print("Daftar pertemanan:")

        for orang, teman in graph.items():

            print(f"{orang}: {', '.join(teman)}")

# Menu interaktif

while True:

    print("\nMenu Pertemanan:")

    print("1. Tambah Pertemanan")

    print("2. Lihat Teman Seseorang")

    print("3. Tampilkan Semua Pertemanan")

    print("4. Keluar")

    pilihan = input("Pilih menu (1-4): ")

    if pilihan == "1":

        nama1 = input("Masukkan nama orang pertama: ")

        nama2 = input("Masukkan nama orang kedua: ")

        tambah\_teman(nama1, nama2)

    elif pilihan == "2":

        nama = input("Masukkan nama yang ingin dilihat temannya: ")

        lihat\_teman(nama)

    elif pilihan == "3":

        tampilkan\_graph()

    elif pilihan == "4":

        print("Terima kasih. Program selesai.")

        break

    else:

        print("Pilihan tidak valid. Silakan coba lagi.")

**4. Tree (Pohon)**

**Fungsi:**

Tree adalah struktur data hirarkis. Setiap node bisa punya anak. Salah satu jenis umum adalah **binary tree**.

**Komponen:**

* Node (data)
* Left dan right child (anak kiri dan kanan)

**Passing Parameter:**

Biasanya dipassing sebagai objek atau node root, dan diakses secara rekursif.

**Contoh kehidupan nyata:**

* **Struktur Organisasi Perusahaan**
* **Folder Komputer**

**Penjelasan:**

* Atasan (CEO) berada di puncak (root)
* Di bawahnya ada manajer, supervisor, dan staf (anak-anak dari node sebelumnya)

🧠 **Kaitan: Hirarki**  
Contoh:  
CEO  
├── Manajer 1  
│ ├── Staf A  
│ └── Staf B  
└── Manajer 2  
└── Staf C

**Input**

print("Struktur Organisasi:")

ceo.tampil\_struktur()

**Output**

Struktur Organisasi:

CEO

Manajer 1

Staf A

Staf B

Manajer 2

Staf C

# Struktur data family tree

family\_tree = {}

# Tambah hubungan orang tua-anak

def tambah\_keluarga(anak, orang\_tua):

    if anak not in family\_tree:

        family\_tree[anak] = {"orang\_tua": [], "anak": []}

    if orang\_tua not in family\_tree:

        family\_tree[orang\_tua] = {"orang\_tua": [], "anak": []}

    if orang\_tua not in family\_tree[anak]["orang\_tua"]:

        family\_tree[anak]["orang\_tua"].append(orang\_tua)

        family\_tree[orang\_tua]["anak"].append(anak)

    print(f"Hubungan ditambahkan: {orang\_tua} adalah orang tua dari {anak}.")

def tampilkan\_orang\_tua(nama):

    if nama in family\_tree and family\_tree[nama]["orang\_tua"]:

        print(f"Orang tua dari {nama}: {', '.join(family\_tree[nama]['orang\_tua'])}")

    else:

        print(f"Tidak ada data orang tua untuk {nama}.")

def tampilkan\_anak(nama):

    if nama in family\_tree and family\_tree[nama]["anak"]:

        print(f"Anak-anak dari {nama}: {', '.join(family\_tree[nama]['anak'])}")

    else:

        print(f"Tidak ada data anak untuk {nama}.")

def tampilkan\_semua():

    if not family\_tree:

        print("Belum ada data keluarga.")

    else:

        print("Struktur Family Tree:")

        for nama, data in family\_tree.items():

            ortu = ', '.join(data['orang\_tua']) if data['orang\_tua'] else 'Tidak ada'

            anak = ', '.join(data['anak']) if data['anak'] else 'Tidak ada'

            print(f"{nama} -> Orang tua: {ortu} | Anak: {anak}")

# Menu Interaktif

while True:

    print("\nMenu Family Tree:")

    print("1. Tambah Hubungan (Orang Tua - Anak)")

    print("2. Lihat Orang Tua")

    print("3. Lihat Anak")

    print("4. Tampilkan Semua Data")

    print("5. Keluar")

    pilihan = input("Pilih menu (1-5): ")

    if pilihan == "1":

        anak = input("Masukkan nama anak: ")

        ortu = input("Masukkan nama orang tua: ")

        tambah\_keluarga(anak, ortu)

    elif pilihan == "2":

        nama = input("Masukkan nama yang ingin dilihat orang tuanya: ")

        tampilkan\_orang\_tua(nama)

    elif pilihan == "3":

        nama = input("Masukkan nama yang ingin dilihat anaknya: ")

        tampilkan\_anak(nama)

    elif pilihan == "4":

        tampilkan\_semua()

    elif pilihan == "5":

        print("Program selesai. Terima kasih!")

        break

    else:

        print("Pilihan tidak valid. Coba lagi.")