TUGAS STRUKTUR DATA

Tugas Ini Dibuat Guna Memenuhi Tugas Struktur Data

Dosen pengampu:

Adam bachtiar, s.kom, M.MT

Disusun Oleh :

Naila Meilisyah

NIM : 24241064

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI

FAKULTAS SAINS, TEKNIK DAN TERAPAN

UNIVERSITAS PENDIDIKAN MANDALIKA MATARAM

2025

```
=====PROGRAM SEDERHANA UNTUK IMPLEMENTASI STACK DENGAN LINKED-LIST=====
Tentukan berapa kapasitas stack : 4

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 1
Masukkan isi stack : 90
Stack : ['90']

Menambah isi Stack Pilih [Ya/Tidak] : ya
Masukkan isi stack : 89
Stack : ['90', '89']

Menambah isi Stack Pilih [Ya/Tidak] : ya
Masukkan isi stack : 86
Stack : ['90', '89', '86']

Menambah isi Stack Pilih [Ya/Tidak] : tidak

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 3
Ukuran stack saat ini adalah: 3

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 4
Puncak stack adalah: 86
```

```
Masukkan pilihan anda : 4
Puncak stack adalah: 86

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 5
Apakah stack penuh? False

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 2
Data 86 telah dihapus.
Stack : ['90', '89']

Pilih menu berikut ini :
1. Menambah isi stack
2. Menghapus isi stack
3. Cek Ukuran Stack saat ini
4. Cek Puncak Stack
5. Cek Stack Full
6. Keluar

Masukkan pilihan anda : 6
Program selesai.
PS C:\Users\elsan\OneDrive\Dokumen\Modul 3>
```

PENJELASAN BARIS-PERBARIS

Bagian Fungsi Stack

```
def create_node(data, next_node=None):
    return {"data": data, "next": next_node}
```

Fungsi membuat node stack.

- data: nilai yang disimpan.
- next_node: referensi ke node sebelumnya dalam stack.
- Node direpresentasikan sebagai dictionary: {"data": ..., "next": ...}.

```
def push(stack, data):
    if is_full(stack):
        print("Stack sudah penuh, tidak bisa menambahkan elemen.")
        return False
    new_node = create_node(data, stack["top"])
    stack["top"] = new_node
    stack["count"] += 1
    return True
```

Menambahkan elemen ke puncak stack.

- Cek dulu apakah stack penuh (is_full).
- Buat node baru dan tautkan ke node sebelumnya (stack["top"]).
- Update top ke node baru dan tingkatkan count.

```
def pop(stack):
    if is_empty(stack):
        print("Stack kosong, tidak bisa menghapus.")
        return None
    data = stack["top"]["data"]
    stack["top"] = stack["top"]["next"]
    stack["count"] -= 1
    return data
```

Menghapus elemen dari puncak stack.

- Jika kosong, tidak bisa pop.
- Ambil data dari top.
- Geser top ke node sebelumnya (top["next"]).
- Kurangi count.

```
def size(stack):
    return stack["count"]
```

Mengembalikan jumlah elemen saat ini dalam stack.

```
def peek(stack):
    if is_empty(stack):
        return None
    return stack["top"]["data"]
```

Melihat elemen puncak stack tanpa menghapusnya.

```
def is_empty(stack):
    return stack["top"] is None
```

Cek apakah stack kosong.

```
def is_full(stack):
    return stack["count"] >= stack["capacity"]
```

Cek apakah stack sudah mencapai kapasitas maksimal.

```
def get_stack_list(stack):
    current = stack["top"]
    elements = []
    while current:
        elements.insert(0, current["data"])
        current = current["next"]
    return elements
```

Mengonversi isi stack ke bentuk list dari bawah ke atas.

- insert(0, ...) agar urutan dari dasar ke puncak.

Program Utama

```
print("=====PROGRAM SEDERHANA UNTUK IMPLEMENTASI STACK DENGAN LINKED-LIST=====")
```

kapasitas = int(input("Tentukan berapa kapasitas stack : "))

Cetak judul dan minta input kapasitas maksimal stack.

```
stack = {
    "top": None,
    "count": 0,
    "capacity": kapasitas
}
```

Inisialisasi stack.

- top: menunjuk ke node teratas (None artinya kosong).
- count: jumlah elemen saat ini.
- capacity: batas maksimal elemen.

```
while True:
    print("\nPilih menu berikut ini : ")
    print("1. Menambah isi stack")
    print("2. Menghapus isi stack")
    print("3. Cek Ukuran Stack saat ini")
    print("4. Cek Puncak Stack")
    print("5. Cek Stack Full")
    print("6. Keluar")
```

Menampilkan menu interaktif berulang kali.

pilihan = input("\nMasukkan pilihan anda : ")

Input pilihan menu dari pengguna.

Menu 1 - Push

```
if pilihan == "1":
    while True:
        data = input("Masukkan isi stack : ")
        if push(stack, data):
            print(f"Stack : {get_stack_list(stack)}")
        else:
            break
```

```
        lanjut = input("\nMenambah isi Stack Pilih [Ya/Tidak] : ").lower()
        if lanjut != "ya":
            break
```

Tambahkan data ke stack.

- Ulangi selama user mengetik "ya".
- Cek jika penuh, proses berhenti.

Menu 2 - Pop

```
elif pilihan == "2":
    hasil = pop(stack)
    if hasil is not None:
        print(f"Data {hasil} telah dihapus.")
        print(f"Stack : {get_stack_list(stack)}")
```

Hapus satu elemen dari stack, tampilkan elemen yang dihapus.

Menu 3 - Ukuran

```
elif pilihan == "3":
    print(f"Ukuran stack saat ini adalah: {size(stack)}")
```

Tampilkan jumlah elemen dalam stack.

Menu 4 - Puncak Stack

```
elif pilihan == "4":
    puncak = peek(stack)
    if puncak is not None:
        print(f"Puncak stack adalah: {puncak}")
    else:
        print("Stack masih kosong.")
```

Lihat elemen teratas stack.

Menu 5 - Penuh?

```
elif pilihan == "5":
    print(f"Apakah stack penuh? {is_full(stack)}")
```

Tampilkan apakah stack sudah penuh atau belum.

Menu 6 - Keluar

```
elif pilihan == "6":
```

```
    print("Program selesai.")
    break
```

Akhiri program.

Jika input tidak valid

```
else:
    print("Pilihan tidak valid. Silakan coba lagi.")
```

Tampilkan pesan error jika pilihan menu tidak sesuai.

Jika kamu ingin, aku juga bisa bantu visualisasi alur stack-nya.

Langkah - Langkah Saat Menjalankan Program

1. Pertama, saya mulai menjalankan program Stack yang sudah saya buat menggunakan bahasa Python. Begitu program dijalankan, hal pertama yang ditampilkan adalah permintaan untuk menentukan kapasitas stack. Di sini, saya diminta untuk memasukkan berapa banyak elemen maksimal yang dapat ditampung oleh stack. Saya pun memasukkan angka 5, yang artinya stack ini hanya bisa menampung lima data saja.

2. Setelah saya menentukan kapasitas, program langsung menampilkan menu utama yang berisi beberapa opsi yang bisa saya pilih. Di antaranya adalah:

- Menambah isi stack
- Menghapus isi stack
- Mengecek ukuran stack saat ini
- Melihat data pada puncak stack
- Mengecek apakah stack sudah penuh
- Dan opsi untuk keluar dari program

Menu tersebut muncul secara berulang sampai saya memilih untuk keluar, jadi saya bisa melakukan berbagai operasi stack satu per satu sesuai keinginan saya.

3. Pada langkah selanjutnya, saya memilih menu 1, yaitu opsi untuk menambah isi stack. Saat diminta memasukkan data, saya mencoba memasukkan angka '10', kemudian saya lanjutkan dengan memasukkan angka '20', '30', '40', dan terakhir '45'. Setiap kali saya menambahkan data, program menampilkan isi stack terbaru. Namun setelah saya mencoba menambahkan elemen keenam, program secara otomatis menolak data tersebut dan menampilkan pesan bahwa stack sudah penuh. Ini menunjukkan bahwa program berhasil mendeteksi batas kapasitas yang saya tetapkan sebelumnya, yaitu lima elemen saja.

4. Kemudian saya kembali ke menu utama dan memilih menu 2, yaitu opsi untuk menghapus isi stack. Karena prinsip kerja stack adalah LIFO (Last In First Out), maka data yang paling terakhir dimasukkan akan menjadi yang pertama keluar. Dalam kasus ini, data '45' yang saya masukkan terakhir akan langsung dihapus. Program pun menampilkan pesan bahwa data tersebut telah berhasil dihapus, dan kemudian memperlihatkan isi stack yang telah diperbarui.

5. Selanjutnya, saya memilih menu 3 untuk mengecek ukuran stack. Program lalu menampilkan jumlah data yang saat ini masih ada di dalam stack. Setelah satu data dihapus sebelumnya, ukuran stack yang tadinya lima menjadi empat. Informasi ini sangat berguna untuk memantau kapasitas stack secara real time.

6. Saya lanjut ke menu 4 untuk melihat elemen yang ada di puncak stack. Program menampilkan bahwa data yang sekarang ada di puncak stack adalah angka '40', yaitu data terakhir yang saya masukkan sebelum data '45' yang sudah dihapus. Ini menunjukkan bahwa fungsi peek dalam program berjalan dengan baik, memberikan saya informasi tanpa menghapus data tersebut.

7. Berikutnya, saya memilih menu 5 untuk mengecek apakah stack sudah penuh. Karena saya sebelumnya telah menghapus satu data dari stack, maka jumlah elemen saat ini adalah empat dari total kapasitas lima. Program pun menampilkan hasil False, yang artinya stack belum penuh. Artinya, masih ada ruang untuk menambahkan satu data lagi.

8. Terakhir, saya memilih menu 6 untuk mengakhiri program. Program pun menampilkan pesan “Program selesai.” yang menandakan bahwa semua proses telah dihentikan. Dengan ini, saya menyelesaikan percobaan saya terhadap program stack dengan linked list secara sederhana namun fungsional.