LAPORAN

Algoritma Struktur Data

DISUSUN OLEH:

Muhammad Fauzan Gifari

2209116042

DOSEN PENGAMPU:

Amin Padmo Azam Masa, S.Kom., M.Cs.

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS MULAWARMAN

2023

PENJELASAN

- **Linked List Stack**

  A. Penjelasan LinkedList Stack

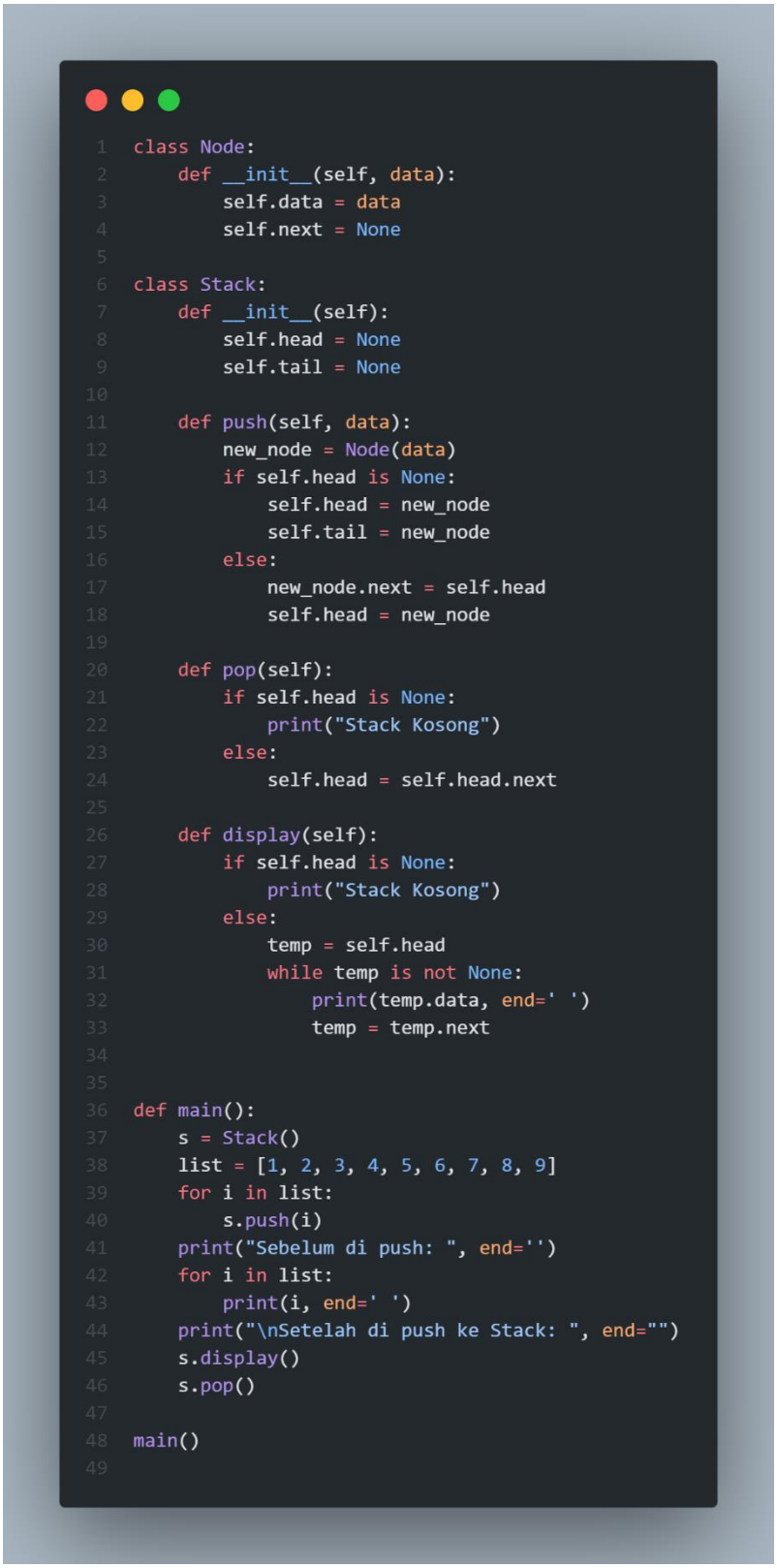
• Linked List Stack adalah struktur data yang digunakan untuk menyimpan dan mengatur data dengan menggunakan konsep Linked List dan Stack. Stack sendiri adalah struktur data yang menerapkan prinsip "Last In First Out" (LIFO), di mana data terakhir yang dimasukkan menjadi data pertama yang dikeluarkan.

Linked List Stack memanfaatkan struktur data Linked List, di mana setiap elemen atau node memiliki pointer yang menunjuk ke elemen berikutnya. Dalam linked list Stack, setiap node atau elemen merepresentasikan data dan memiliki pointer yang menunjuk ke elemen sebelumnya.

Linked list Stack biasanya terdiri dari dua operasi utama, yaitu push (untuk menambahkan data baru ke dalam stack) dan pop (untuk mengeluarkan data teratas atau yang paling baru dari stack). Dalam linked list stack, push dilakukan dengan menambahkan node baru pada elemen teratas (head) dan mengubah pointer head menjadi node baru tersebut. Sedangkan pop dilakukan dengan menghapus node pada elemen teratas (head) mengubah pointer head menjadi node yang berada dibawahnya.

keuntungan dari menggunakan linked list Stack adalah fleksibilitas dalam menambah dan menghapus data, serta kemampuan untuk mengesuaikan ukuran stack secara dinamis sesuai kebutuhan. Namun, penggunaan linked list stack memerlukan alokasi memori yang lebih besar dibandingkan dengan array Stack dan operasi push dan pop memerlukan waktu yang lebih lama karena melibatkan perubahan pointer.

B. Implementasi Linked List Stack



```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Stack:
    def __init__(self):
        self.head = None
        self.tail = None

    def push(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            self.tail = new_node
        else:
            new_node.next = self.head
            self.head = new_node

    def pop(self):
        if self.head is None:
            print("Stack Kosong")
        else:
            self.head = self.head.next

    def display(self):
        if self.head is None:
            print("Stack Kosong")
        else:
            temp = self.head
            while temp is not None:
                print(temp.data, end=' ')
                temp = temp.next


def main():
    s = Stack()
    list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    for i in list:
        s.push(i)
    print("Sebelum di push: ", end='')
    for i in list:
        print(i, end=' ')
    print("\nSetelah di push ke Stack: ", end="")
    s.display()
    s.pop()

main()
```

- **Linked List Queue**

A. Penjelasan Linked List Queue

Date

• Linked List Queue adalah struktur data yang digunakan untuk menyimpan dan mengatur data dengan menggunakan konsep linked list dan Queue. Queue sendiri adalah struktur data yang menerapkan prinsip "First In First Out" (FIFO), dimana data pertama yang dimasukkan menjadi data pertama yang dikeluarkan.

Linked List Queue memanfaatkan struktur data linked list, dimana setiap elemen atau node memiliki pointer yang menunjuk ke elemen berikutnya. Dalam Linked List Queue, setiap node atau elemen merepresentasikan data dan memiliki pointer yang menunjuk ke elemen berikutnya.

Linked List Queue biasanya terdiri dari dua operasi utama, yaitu enqueue (untuk menambahkan data baru ke dalam queue) dan dequeue (untuk mengeluarkan data terdepan atau yang paling lama dari queue). Dalam linked list queue, enqueue dilakukan dengan menambahkan node baru pada elemen terakhir (tail) dan mengubah pointer tail menjadi node baru tersebut. Sedangkan dequeue dilakukan dengan menghapus node pada elemen terdepan (head) dan mengubah pointer head menjadi node yang berada di bawahnya.

B. Implementasi Linked List Queue

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Queue:
    def __init__(self):
        self.head = None
        self.tail = None

    def enqueue(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            self.tail = new_node
        else:
            self.tail.next = new_node
            self.tail = new_node

    def dequeue(self):
        if self.head is None:
            print("Antrian Kosong")
        else:
            self.head = self.head.next

    def display(self):
        if self.head is None:
            print("Antrian Kosong")
        else:
            temp = self.head
            while temp is not None:
                print(temp.data, end=' ')
                temp = temp.next

def main():
    q = Queue()
    list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    for i in list:
        q.enqueue(i)
    print("Sebelum : ", end='')
    q.display()
    print("\nSesudah : ", end='')
    q.display()

main()
```

- **Array Stack**

  A. Penjelasan Array Stack

- Array Stack adalah struktur data yang digunakan untuk menyimpan kumpulan elemen data yang terorganisir dalam urutan tertentu. Array stack memungkinkan akses dan penghapusan elemen data hanya dari satu ujung, yaitu ujung atas atau top.

- Dalam implementasi array stack, array digunakan sebagai penyimpanan data dan variabel top digunakan untuk menunjukkan posisi paling atas atau teratas dari tumpukan. Ketika menambahkan elemen data ke array stack, variabel top akan di increment dan element data akan disimpan pada posisi yang ditunjukkan oleh variabel top. Sebaliknya ketika menghapus elemen data, elemen pada posisi variabel top akan dihapus dan variabel top akan di decrement.

B. Implementasi Array Stack

```python
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def display(self):
        print(self.items)

def main():
    s = Stack()
    data = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    for item in data:
        s.push(item)
    print("Sebelum : ", end='')
    s.display()
    print("Sesudah : ", end='')
    s.display()

main()

```

- **Array Queue**

  A. Penjelasan Array Queue

- Array Queue adalah struktur data yang digunakan untuk menyimpan kumpulan elemen data yang terorganisir dalam urutan tertentu. Array queue menunjukkan akses dan penghapusan elemen data hanya dari dua ujung, yaitu ujung depan atau front dan ujung belakang atau rear.

- Dalam implementasi array queue, array digunakan sebagai penyimpanan data dan variabel front dan rear digunakan ~~pada~~ untuk menunjukkan posisi ujung depan dan belakang dari antrian. Ketika menambahkan elemen data ke array queue, elemen data akan disimpan pada posisi variable rear dan variable rear akan di increment. Sebaliknya, ketika menghapus element data, element pada posisi variabel front akan dihapus dan variabel front akan di increment.

B. Implementasi Array Queue

```python
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def display(self):
        print(self.items)

def main():
    s = Stack()
    data = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    for item in data:
        s.push(item)
    print("Sebelum : ", end='')
    s.display()
    print("Sesudah : ", end='')
    s.display()

main()

```