1 Konsep

Tulis ringkasan atau penjelasan hal-hal berikut, dengan kata-kata kalian sendiri mengenai, Konsep struktur data Stack, operasi-operasi yang terdapat pada stack, dan implementasi operasi-operasi stack tersebut dengan bahasa Pemrograman Python.

Jawaban

1. Konsep Struktur Data Stack

Stack merupakan sebuah struktur data yang penambahan dan penghapusan datanya hanya dapat di lakukan pada satu ujung yang sama (top). konsep ini biasa dikenal dengan nama LIFO (Last In First Out) jadi data yang terakhir kali ditambahkan akan pertama dihapus

Stack banyak kita temui dalam kehidupan sehari hari sebagai tumpukan, misalkan tumpukan baju di lemari jika kita ingin mengambil baju pada posisi bawah maka kita harus mengambil baju yang ada di atasnya terlebih dahulu. Dan ketika ada baju baru ditambahkan maka akan berada di posisi paling atas

2. operasi-operasi yang terdapat pada stack

```
- stack   Digunakan untuk inisialisasi awal untuk tempat data stack
- push    Digunakan untuk penambahan data baru pada posisi top
- pop     Digunakan untuk penghapusan data yang terdapat di posisi top dari stack. dan fungsi mengambalikan/return data yang di pop
- peek    Digunakan untuk mengetahui data apa yang terletak pada posisi top
- isEmpty Digunakan untuk pengecekan apakah stack dalam keadaan kosong atau tidak
- size    Digunakan untuk mengetahui jumlah data yang terdapat pada stack
```

3. implementasi operasi-operasi stack

Berikut adalah implementasi operasi - operasi stack dalam python:

```
def stack():
    st=[]
    return st
def push(st,data):
    st.append(data)
def pop(st):
    data=st.pop()
    return data
def peek(st):
    return st[-1]
def isEmpty(st):
    return st==[]
def size(st):
    return len(st)
```

2 Implementasi - 1

2.1 Konversi Desimal ke Biner - NPM Genap

Buatlah Fungsi dan code yang diperlukan untuk mengkonversi bilangan desimal ke biner dengan menggunakan struktur data Stack. Tampilkan juga visualisasi stack, seperti yang ditunjukkan pada Gambar 1

```
binaryNum=des2Bin(25)
print('Biner =', binaryNum)
```

```
25  div 2= 12  sisa  1 : push(stack, 1 )

| 1 |
-----

12  div 2= 6  sisa  0 : push(stack, 0 )

| 0 |
| 1 |
-----

6  div 2= 3  sisa  0 : push(stack, 0 )

| 0 |
| 0 |
| 1 |
-----

3  div 2= 1  sisa  1 : push(stack, 1 )

| 1 |
| 0 |
| 0 |
| 1 |
-----

1  div 2= 0  sisa  1 : push(stack, 1 )

| 1 |
| 1 |
| 0 |
| 0 |
| 1 |
-----

Biner = 11001
```

```
binaryNum=des2Bin(19)
print('Biner =', binaryNum)
```

```
19  div 2= 9  sisa  1 : push(stack, 1 )

| 1 |
-----

9  div 2= 4  sisa  1 : push(stack, 1 )

| 1 |
| 1 |
-----

4  div 2= 2  sisa  0 : push(stack, 0 )

| 0 |
| 1 |
| 1 |
-----

2  div 2= 1  sisa  0 : push(stack, 0 )

| 0 |
| 0 |
| 1 |
| 1 |
-----

1  div 2= 0  sisa  1 : push(stack, 1 )

| 1 |
| 0 |
| 0 |
| 1 |
| 1 |
-----

Biner = 10011
```

Gambar 1: Konversi Desimal ke Biner

```
# Import modul

from google.colab import drive
drive.mount('/content/drive')

import sys
sys.path.append('/content/drive/MyDrive/Colab_Notebooks/')

from stack import *
```

```
# Jawaban
def des2Bin(num):
    binStc = stack()
    result=''

    while num != 0 :
        temp = num % 2
        push(binStc,temp)
        div = num // 2
        print(f"{num} div 2 = {div} sisa {temp} : push(stack, {temp})\n")
        num = div

        for i in binStc:
            print(f"| {i} |")
        print( '-----\n')

    while not isEmpty(binStc):
        result += str(pop(binStc))
    return result
```

```
binaryNum = des2Bin(25)
print('Biner = ', binaryNum)
```

```
25 div 2 = 12 sisa 1 : push(stack, 1)

| 1 |
-----

12 div 2 = 6 sisa 0 : push(stack, 0)

| 0 |
| 1 |
-----

6 div 2 = 3 sisa 0 : push(stack, 0)

| 0 |
| 0 |
| 1 |
-----

3 div 2 = 1 sisa 1 : push(stack, 1)

| 1 |
| 0 |
| 0 |
| 1 |
-----

1 div 2 = 0 sisa 1 : push(stack, 1)

| 1 |
| 1 |
| 0 |
| 0 |
| 1 |
-----

Biner =  11001
```

Gambar 3: Pengecekan Jumlah dan Jenis Kurung-2 3 Implementasi - 2 3.1 Pengecekan Tag HTML Buatlah fungsi pengecekan tag-tag syntax HTML, dengan ketentuan sebagai berikut :

1. Gunakan Modul Stack yang sudah ada
2. Input berupa syntax HTML yang terdiri dari beberapa baris. [Petunjuk] Gunakan syntax triple single quotation marks atau triple double quotation marks pada multiline text, dan gunakan method split pada string. Lihat contoh input pada Gambar 4, Gambar 5, Gambar 6.
3. return value adalah True(jika sudah memenuhi kaidah tag html), dan False (jika tidak memenuhikaidah tag html)

4. Jika return value adalah False, tampilkan juga kesalahan penulisan html yang telah dibuat, sepertipada Gambar 4, Gambar 5, Gambar 6.

In [3]:
```
inp="""
<html>
<body>
<title>
Praktikum Struktur Data
</title>
<h1>
Stack
</h1>
</body>
</html>
"""
print(inp)
isValidateTagHTML(inp)
```

```
<html>
<body>
<title>
Praktikum Struktur Data
</title>
<h1>
Stack
</h1>
</body>
</html>
```

Out[3]: True

```python
# Jawaban
def isValidateTagHTML(strHTML):
    strHTML = strHTML.split()
    tagHTML = stack()
    matched = True
    pesan = ['Error : ']
    no = 1

    for i in strHTML:

        if "</" in i  and '>' in i:
            if isEmpty(tagHTML):
                return matched, pesan + [f'{no}. Kelebihan Close Tag : {i}']
            else:
                temp = pop(tagHTML)
                if temp[1:-1].lower() == i[2:-1].lower():
                    matched = matched and True
                else:
                    matched = matched and False
                    pesan.append(f'{no}. Open Tag {i} --> {temp}')
                    no+=1
        elif "<" in i and ">" in i :
            push(tagHTML,i)

    if not(isEmpty(tagHTML)):
        return matched , pesan + [f"{no}. Kelebihan Open Tag : {tagHTML[0]}"]
    if matched == False:
        return matched , pesan + [f'{no}. Tag Tidak Sesuai ']
    else:
        return matched
```

'2. Open Tag </html> --> <body>',

```python
inp = """
<html>
<body>
<title>
Praktikum Struktur Data
<h1>
Stack
</h1>
</body>
</html>
"""
print(inp)
```

✓ 0s completed at 9:17 AM