



# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

<b>NIM</b>	<b>71231042</b>
<b>Nama Lengkap</b>	<b>Revaldo Fransisco Hohary</b>
<b>Minggu ke / Materi</b>	<b>14 / Rekursif</b>

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS KRISTEN DUTA WACANA  
YOGYAKARTA  
2024

## BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

### MATERI 1

#### A. Pengertian Rekursif

Fungsi rekursif, juga disebut sebagai fungsi yang memanggil dirinya sendiri, adalah fungsi yang berisi dirinya sendiri atau fungsi yang mendefinisikan dirinya sendiri. Fungsi rekursif memiliki pola matematis yang berulang dan biasanya terstruktur. Sangat penting untuk memperhatikan fungsi ini agar dapat berhenti dan tidak menghabiskan memori. Anda harus berhati-hati saat menggunakan fungsi rekursif karena fungsi ini dapat memiliki lingkaran tak terbatas, yang akan menyebabkan program hang up. Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi. Jadi, dua blok penting harus ada dalam fungsi rekursif: blok yang menjadi titik berhenti dari proses rekursif dan blok yang memanggil dirinya sendiri. Rekursif terdiri dari dua bagian:

- Base Case adalah bagian dimana penentu bahwa fungsi rekursif itu berhenti
- Rekursif Case adalah bagian Dimana terdapat statement yang akan terus diulang-terus menerus hingga mencapai Base Case

#### B. Kelebihan dan Kekurangan

Beberapa keunggulan fungsi rekursif adalah sebagai berikut:

- Kode program lebih singkat dan elegan.
- Masalah kompleks dapat di breakdown menjadi sub masalah yang lebih kecil di dalam rekursif.

sedangkan kelemahan dari rekrusif itu sendiri adalah:

- Memakan memori yang lebih besar karena setiap kali bagian darinya dipanggil maka dibutuhkan sejumlah runag memori tambahan.
- Mengorbankan efisiensi dan kecepatan.
- Fungsi rekursif sulit dilakukan debugging dan kadang sulit dimengerti.

#### C. Bentuk Umum dan Studi Kasus

Bentuk umum pada fungsi rekrusif pada Python:

```
def function_name(parameter_list):  
    ...  
    function_name(...)
    ...
```

Sebenarnya, setiap fungsi rekursif pasti memiliki solusi iteratifnya. Contoh kasus faktorial berikut adalah sebagai berikut: Faktorial adalah menghitung perkalian deret angka  $1 \times 2 \times 3 \times \dots \times n$ . Algoritma yang digunakan untuk menghitung faktorial adalah sebagai berikut:

- Tanyakan n
- Siapkan variabel total untuk menampung hasil perkalian faktorial dan set nilai awal dengan
- Loop dari  $i = 1$  hingga n untuk mengerjakan:
- $total = total * i$

e. Tampilkan total

Dengan menggunakan fungsi rekursif maka faktorial dapat dihitung dengan rumus pada gambar

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n - 1) & \text{if } n > 0 \end{cases}$$

Gambar 13.1 : Rumus

Pseudocode dapat dibuat secara rekursif dari rumus 13.1, seperti yang ditunjukkan pada gambar 13.2. Kode Python yang dapat dibuat dari pseudocode ini adalah:

**Pseudocode (recursive):**

```
function factorial is:
input: integer n such that n >= 0
output: [n × (n-1) × (n-2) × ... × 1]

    1. if n is 0, return 1
    2. otherwise, return [ n × factorial(n-1) ]

end factorial
```

Gambar 13.2 : Pseudocode rekursif

```
def faktorial(n):
    if n==0 or n==1:
        return 1
    else:
        return faktorial(n-1) * n
print(faktorial(4))
```

Hasil outputnya : 24

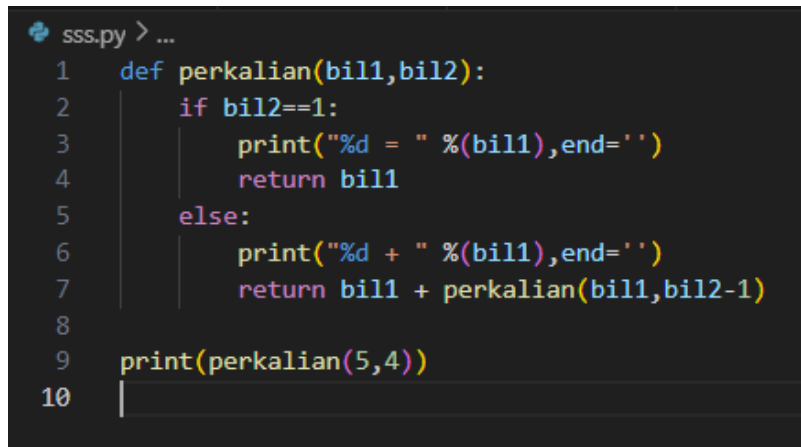
```
1.
2. calc_factorial(4)           # 1st call with 4
3. 4 * calc_factorial(3)      # 2nd call with 3
4. 4 * 3 * calc_factorial(2)  # 3rd call with 2
5. 4 * 3 * 2 * calc_factorial(1) # 4th call with 1
6. 4 * 3 * 2 * 1              # return from 4th call as number=1
7. 4 * 3 * 2                  # return from 3rd call
8. 4 * 6                      # return from 2nd call
9. 24                         # return from 1st call
```

Gambar 13.3: Proses Perhitungan

#### D. Kegiatan Praktikum

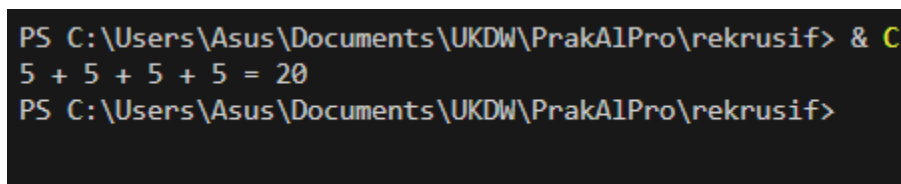
##### a. Problem dan Solusi 1

Pada kegiatan praktikum akan dilakukan beberapa percobaan kasus yang dapat diselesaikan dengan menggunakan fungsi rekursif. Kasus 13.1 Buatlah sebuah program yang dapat melakukan perkalian antara 2 buah bilangan dengan menggunakan fungsi rekursif. Misalkan kita ingin mengalikan angka 5 dengan 4. Dengan metode penjumlahan diperoleh  $5 \times 4 = 5 + 5 + 5 + 5 = 20$ .



```
sss.py > ...
1  def perkalian(bil1,bil2):
2      if bil2==1:
3          print("%d = " %(bil1),end='')
4          return bil1
5      else:
6          print("%d + " %(bil1),end='')
7          return bil1 + perkalian(bil1,bil2-1)
8
9  print(perkalian(5,4))
10 |
```

Gambar 13.4 : kode program



```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekursif> & C
5 + 5 + 5 + 5 = 20
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekursif>
```

Gambar 13.5 : Output yang dihasilkan

##### b. Problem dan Solusi

Kasus 13.2: Buat program yang menggunakan fungsi rekursif untuk memangkatkan dua angka. Misalkan kita ingin memangkatkan angka 2 dengan 4. Dengan menggunakan teknik penjumlahan, kita akan mendapatkan  $5 \times 4 = 5 * 5 * 5 * 5 = 16$ . Untuk menjawab pertanyaan tersebut, logikanya hampir sama dengan 13.4 sebelumnya, kecuali operatornya diganti dengan "\*" daripada "+". Kode program adalah:

```

1  def pangkat(bil1,bil2):
2      if bil2==1:
3          print("%d = " %(bil1),end='')
4          return bil1
5      else:
6          print("%d * " %(bil1),end='')
7          return bil1 * pangkat(bil1,bil2-1)
8
9  print(pangkat(5,4))
10

```

Gambar 13.6 : kode program

```

PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekursif> & C:
5 * 5 * 5 * 5 = 625
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekursif>

```

Gambar 13.7 : Outputnya

### c. Problem dan Solusi 3

Tini adalah anak yang pelupa, ia mendapatkan tugas untuk mencari bilangan pada deret Fibonacci dengan urutan tertentu. Dari pada harus selalu menghitung dari awal, bantulah Tono dengan membuatkan program yang menampilkan bilangan tertentu pada deret Fibonacci sesuai dengan urutan yang diinputkan user. Yang perlu diingat, berikut ini adalah bentuk deret Fibonacci. 1 1 2 3 5 8 13 21 34 ... n Bilangan fibonacci adalah bilangan yang berasal dari penjumlahan 2 bilangan sebelumnya. Secara iteratif dapat dibuat program sebagai berikut:

```

1  def fibo(n):
2      f1,f2=1,1
3      print(f1," ",f2," ",end='')
4      for i in range(2,n):
5          fib = f1+f2
6          f1 = f2
7          f2 = fib
8          print(fib," ",end='')
9
10 fibo(7)
11

```

Gambar 13.8 : Kode program

```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> & C
1 , 1 , 2 , 3 , 5 , 8 , 13 ,
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> █
```

Gambar 13.9 : Outputnya

**E. Link GitHub**

<https://github.com/Frealy0901/PrakAlpro14.git>

## BAGIAN 2: LATIHAN MANDIRI (60%)

### SOAL 1

Pada soal ini kita diminta untuk membuat program pengecekan bilangan prima dengan menggunakan fungsi rekursif

```
latihan1.py > ...
1  def is_prime(n, divisor=2):
2      if n <= 1:
3          return False
4      elif divisor * divisor > n:
5          return True
6      elif n % divisor == 0:
7          return False
8      else:
9          return is_prime(n, divisor + 1)
10
11  try:
12      bilangan = int(input("Masukkan bilangan: "))
13      if is_prime(bilangan):
14          print(f"{bilangan} adalah bilangan prima.")
15      else:
16          print(f"{bilangan} bukan bilangan prima.")
17  except ValueError:
18      print("Masukkan harus berupa bilangan bulat.")
```

Hasil ouputnya :

```
PS C:\Users\Asus\Documents\UKDW\PrakAIPro\rekursif> & C:/U
Masukkan bilangan: 10
10 bukan bilangan prima.
PS C:\Users\Asus\Documents\UKDW\PrakAIPro\rekursif> |
```

## SOAL 2

Pada soal nomor 2 kita akan membuat fungsi rekursif mengetahui suatu kalimat itu polindrum atau bukan:

```
latihan2.py > ...
1 def is_palindrome(s):
2     # Basis: Jika panjang kalimat kurang dari atau sama dengan 1, maka palindrom
3     if len(s) <= 1:
4         return True
5     # Jika karakter pertama dan terakhir tidak sama, bukan palindrom
6     elif s[0] != s[-1]:
7         return False
8     # Rekursi: Cek sisa kalimat tanpa karakter pertama dan terakhir
9     else:
10        return is_palindrome(s[1:-1])
11
12 # Contoh penggunaan
13 kalimat = input("Masukkan kalimat: ").lower() # Ubah ke huruf kecil untuk memperhitungkan kapitalisasi
14 if is_palindrome(kalimat):
15     print(f'{kalimat}' adalah palindrom.")
16 else:
17     print(f'{kalimat}' bukan palindrom.")
18
```

Hasil Ouput :

```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> & C:/Users/
Masukkan kalimat: kodok
'kodok' adalah palindrom.
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> 
```

## SOAL 3

Pada soal ketiga disini kita diminta untuk menghitung jumlah deret ganjil(n):

Pada bagian pertama didalam def kita membuat if n sama dengan 1 akan mengembalikan 1. Pada bagian 4 jika tidak akan mengembalikan jumlah\_deret\_ganjil(n-1) ditambahkan (n dikali 2 dikurangi 1) dan pada bagian terakhir akan mengprint.



```

latihan3.py > jumlah_deret_ganjil
1  def jumlah_deret_ganjil(n):
2      if n == 1:
3          return 1
4      else:
5          return jumlah_deret_ganjil(n-1) + (n*2 - 1)
6  print(jumlah_deret_ganjil(5)) # Output: 25

```

Hasil Output :

```

PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> & C
25
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> 

```

#### SOAL 4

Pada soal nomor 4 disini kita akan menghitung jumlah angka yang akan dimasukan oleh pengguna: Pada bagian pertama kita membuat def jumlah\_digit(n), didalam def itu terdapat if n lebih kecil dari 10 maka akan mengembalikan n. Dan jika tidak maka n dimodulokan dan ditambah jumlah digit( n pembagian bulat 10). Untuk bagian ketujuh kita akan membuat jika pengguna salah memasukan input: Pada bagian delapan kita buat bilangan baru dan meminta inputan integer dari pengguna. Pada bagian Sembilan jika bilangan lebih besar atau sama dengan 0 maka jumlah\_digit(bilangan akan disimpan di result dan akan mengeluarkan bilangan dan result. Pada bagian 12 jika tidak maka akan mengeluarkan angka harus lebih besar atau sama dengan 0.

```

latihan4.py > ...
1  def jumlah_digit(n):
2      if n < 10:
3          return n
4      else:
5          return n % 10 + jumlah_digit(n // 10)
6
7  try:
8      bilangan = int(input("Masukkan bilangan: "))
9      if bilangan >= 0:
10         result = jumlah_digit(bilangan)
11         print(f"Jumlah digit dari {bilangan} adalah {result}.")
12     else:
13         print("Masukkan harus lebih besar atau sama dengan 0.")
14 except ValueError:
15     print("Masukkan harus berupa bilangan bulat.")

```

Hasil ouputnya:

```

PS C:\Users\Asus\Documents\UKDW\PrakAIPro\rekrusif> & C:/U
Masukkan bilangan: 10
Jumlah digit dari 10 adalah 1.
PS C:\Users\Asus\Documents\UKDW\PrakAIPro\rekrusif>

```

## SOAL 5

Pada soal nomor lima kita diminta untuk membuat fungsi rekursif untuk menghitung kombinasi :

Pada bagian pertama kita membuat def kombinasi, dalam def kombinasi terdapat if y sama dengan 0 atau y sama dengan x dan akan mengembalikan 1 dan jika y lebih besar dari x maka akan mengembalikan 0 jika tidak ada dalam dua hal itu maka akan mengembalikan kombinasi(x dikurangi 1 dan y -1) ditambahkan kombinasi (x-1 dan y) dan terakhir pada bagian 8 akan mengprint.

```
latihan5.py > kombinasi
1  def kombinasi(x, y):
2      if y == 0 or y == x:
3          return 1
4      elif y > x:
5          return 0
6      else:
7          return kombinasi(x-1, y-1) + kombinasi(x-1, y)
8  print(kombinasi(5, 2))
```

Hasil ouputnya:

```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> & C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif\latihan5.py
10
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\rekrusif> 
```