Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230982**
Nama Lengkap	**Rainie Fanita Chrisabel Hadisantoso**
Minggu ke / Materi	**13 / Rekursif**

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.

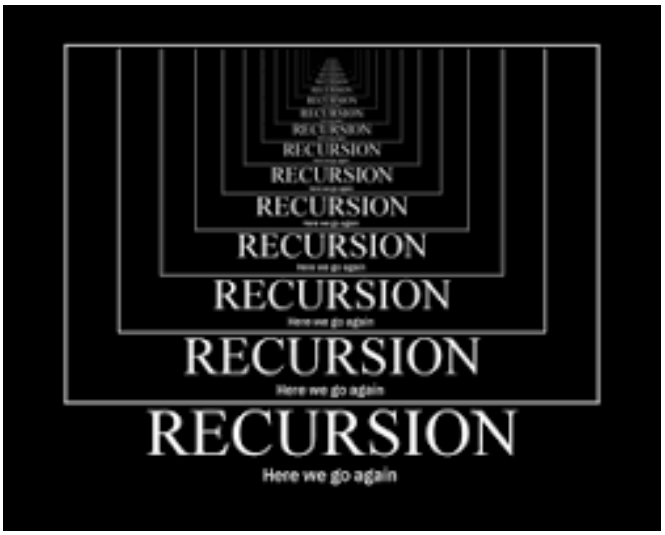
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

MATERI : Percabangan dan Perulangan Kompleks

Dalam pertemuan ke-14 ini akan dibahas mengenai percabangan dan perulagan kompleks atau disebut juga dengan rekursif. Dari pembelajaran ini ada beberapa tujuan yang hendak dibahas, yaitu:

1. Apa itu fungsi rekursif
2. Bagaimana cara membaca dan menguraikan proses kerja fungsi rekursif
3. Bagaimana cara beberapa kasus rekursif bekerja
4. Bagaimana cara membuat fungsi rekursif

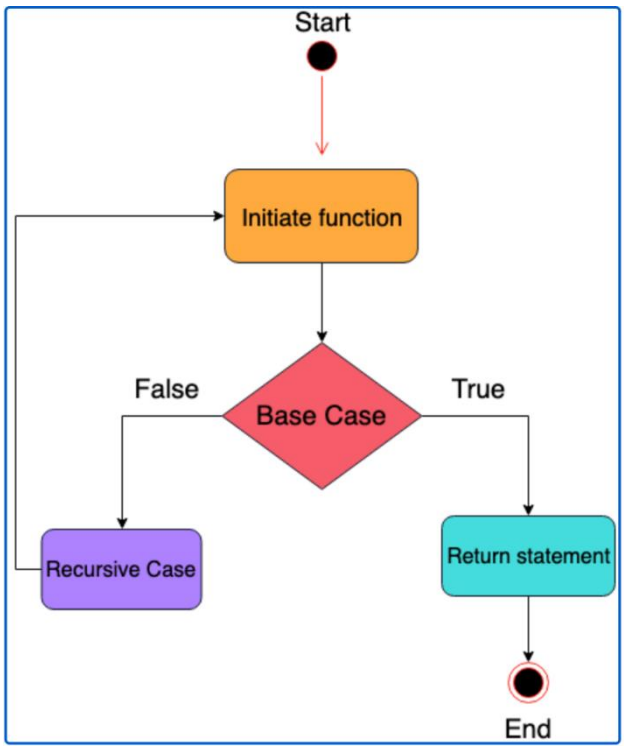
Pengertian Rekursif



Secara sederhana, fungsi rekursif adalah fungsi yang berisi dirinya sendiri. Berisi dirinya sendiri artinya didalam fungsi tersebut ada kode yang mendefinisikan dirinya sendiri. Karena itu fungsi rekursif juga sering disebut juga fungsi yang memanggil dirinya sendiri. Fungsi rekursif merupakan fungsi sistematis yang berulang dengan pola yang terstruktur. Salah satu hal yang perlu diperhatikan dalam menggunakan fungsi rekursif adalah proses fungsi ini dapat berjalan terus menerus menghabiskan memory. Karena potensinya untuk tidak dapat berhenti itu fungsi rekursi dapat bersifat unlimited loop dan menyebabkan program hang up jika tidak dihentikan.

Untuk menghindari hal tersebut, dalam membuat fungsi rekursi perlu diberikan sebuah batasan yang jika terpenuhi maka fungsi rekursif akan berhenti. Karena itu dalam sebuah fungsi rekursif diperlukan 2 bagian penting, yaitu bagian yang memanggil dirinya sendiri dan bagian yang menjadi titik berhenti dari perulangan program rekursif:

1. Base case: batasan yang jika terpenuhi maka program akan berhenti.
2. Rekursive case: bagian dimana ada pemanggilan diri sendiri yang terus menerus diulangi hingga memenuhi batasan yang diminta base case.



Kelebihan dan Kekurangan

Fungsi rekursi memiliki kelebihan dan kekurangan yang perlu dipertimbangkan tergantung pada konteks penggunaannya. Berikut adalah beberapa kelebihan dan kekurangan fungsi rekursi:

Kelebihan:

1. Keterbacaan Kode: Kadang-kadang, masalah dapat diungkapkan dengan cara yang lebih jelas dan konsis melalui rekursi daripada melalui iterasi. Ini dapat membuat kode lebih mudah dipahami dan dikelola.
2. Solusi Pendek: Dalam beberapa kasus, solusi menggunakan rekursi bisa jauh lebih pendek daripada solusi menggunakan iterasi. Hal ini dapat mengurangi jumlah kode yang harus ditulis dan dipelihara.
3. Pengelolaan Struktur Data: Rekursi seringkali cocok untuk masalah yang melibatkan struktur data berhirarki atau berulang, seperti pohon, graf, atau daftar terhubung. Ini karena aliran pemrosesan alami dari rekursi dapat mencerminkan struktur data yang dioperasikan.

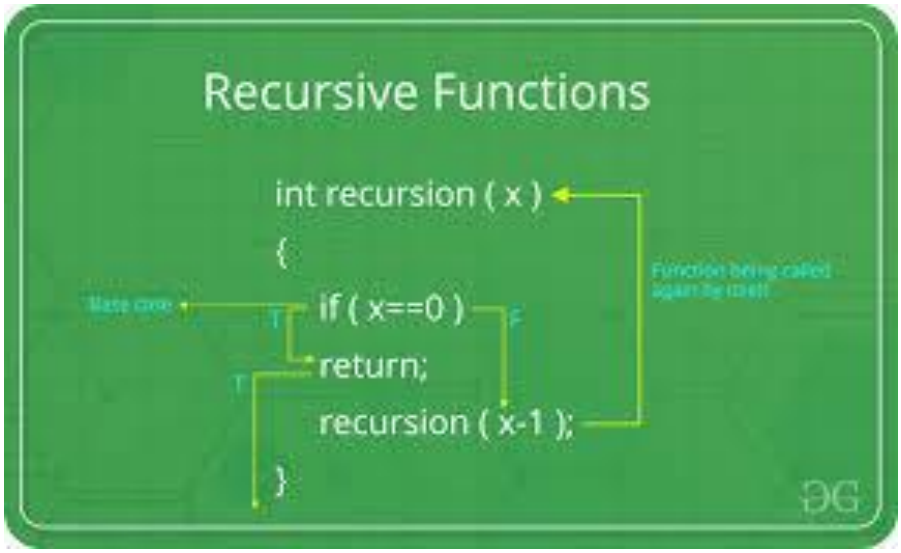
Kekurangan:

1. Overhead Memori dan Waktu: Penggunaan rekursi dapat menyebabkan overhead memori dan waktu yang signifikan, terutama jika rekursi terlalu dalam dan tidak dioptimalkan. Setiap pemanggilan rekursif memerlukan alokasi memori tambahan untuk frame stack, dan pemanggilan rekursif berulang dapat menyebabkan waktu eksekusi yang berlebihan.

2. Ketidakpastian Stack: Jika kedalaman rekursi terlalu besar, dapat terjadi stack overflow, di mana ukuran tumpukan melebihi batas yang ditetapkan oleh sistem. Ini dapat menyebabkan program berhenti secara paksa.
3. Kesulitan Pemeliharaan dan Debugging: Fungsi rekursif seringkali sulit untuk dipelihara dan didebug karena aliran eksekusi yang kompleks dan sulit diikuti. Kesalahan dalam basis kasus atau langkah rekursif dapat sulit dideteksi dan diperbaiki.
4. Kinerja yang Kurang Efisien: Rekursi mungkin tidak selalu merupakan pendekatan yang paling efisien untuk menyelesaikan masalah tertentu. Dalam beberapa kasus, pendekatan iteratif atau menggunakan algoritma yang berbeda dapat memberikan kinerja yang lebih baik.

Pemilihan antara rekursi dan iterasi harus dipertimbangkan dengan cermat tergantung pada kebutuhan spesifik dari masalah yang dihadapi, serta pertimbangan kinerja dan pemeliharaan kode.

Bentuk Umum dan Studi Kasus



Bentuk umum fungsi rekursif berawal dari fungsi def, lalu didalam fungsi terdapat basecase dan recursive case:

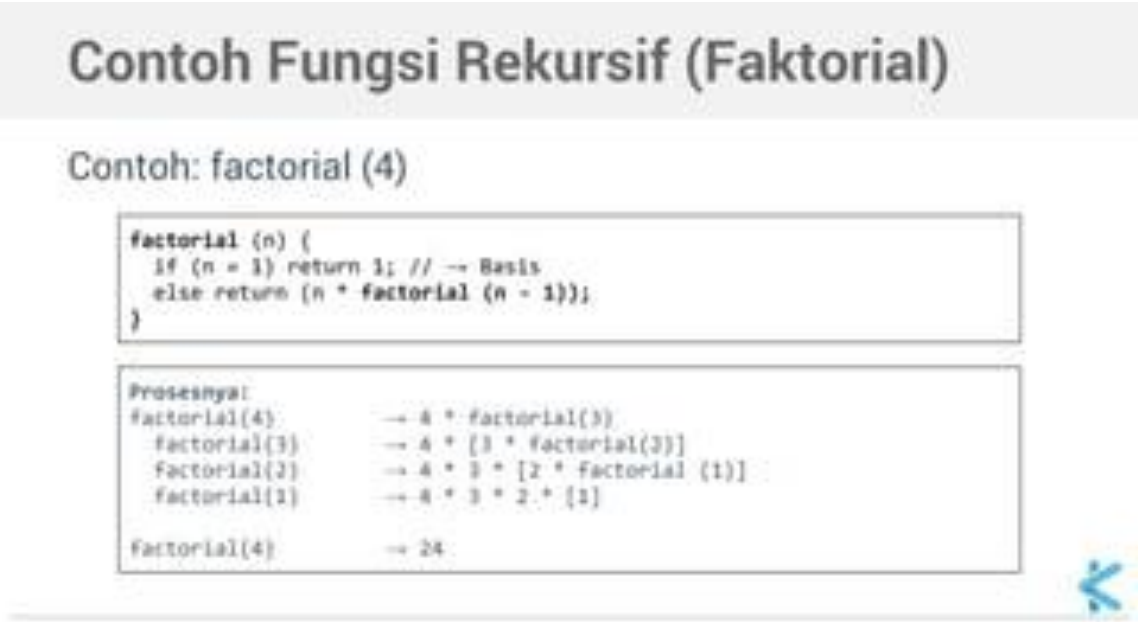
```
def recursive(parameter):
    if ... :
        base case
    else ... :
        recursive()
```

Fungsi faktorial bisa diimplementasikan dengan pendekatan iteratif, yang tidak menggunakan rekursi. Berikut adalah contoh implementasi faktorial menggunakan pendekatan iteratif:

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

1. Inisialisasikan variabel total dengan nilai awal 1.
2. Lakukan iterasi dari 1 hingga n.
3. Pada setiap iterasi, kalikan total dengan nilai iterasi saat ini.
4. Setelah iterasi selesai, total akan berisi hasil faktorial dari n.

Pseudocode dari rumus faktorial menggunakan fungsi rekursif sebagai berikut:



Dari pseudocode diatas kita dapat menuliskan program Python sederhana dengan bentuk fungsi rekursif:

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

Berikut penjelasan kode diatas:

1. Fungsi faktorial didefinisikan dengan satu parameter n.
2. Di dalam fungsi, dilakukan pengecekan apakah nilai n sama dengan 0 atau 1. Jika ya, maka fungsi langsung mengembalikan nilai 1 karena faktorial dari 0 dan 1 adalah 1.

3. Jika nilai n tidak sama dengan 0 atau 1, maka fungsi akan melakukan pemanggilan rekursif kembali ke fungsi faktorial dengan parameter n-1, kemudian hasilnya dikalikan dengan n.
4. Proses ini berulang hingga mencapai basis kasus saat n sama dengan 0 atau 1.
5. Input penggunaan fungsi adalah nilai 4.
6. Fungsi faktorial dipanggil dengan argumen 4.
7. Hasil dari pemanggilan fungsi faktorial dicetak.

Output dari program diatas adalah:

```
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
```

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

Membuat program untuk mengecek bilangan prima dengan fungsi rekursif.

a. Source code

```
def prima(a, b = None):
    if b is None:
        b = a - 1
    while b > 1:
        if a % b == 0:
            return "BUKAN PRIMA"
        else:
            return prima(a, b - 1)
    else:
        return "PRIMA"

n = int(input("Masukkan angka :"))
print(prima(n))
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads\
Masukkan angka :67
PRIMA
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads\
Masukkan angka :76
BUKAN PRIMA
```

c. Penjelasan

- Fungsi prima didefinisikan dengan dua parameter a dan b, dengan b yang defaultnya adalah None.
- Jika b adalah None, maka di dalam fungsi, b diatur sama dengan a - 1.
- Kemudian dilakukan pengulangan while dengan kondisi b > 1.
- Di dalam loop, diperiksa apakah a habis dibagi dengan b. Jika ya, maka bilangan tersebut bukan prima dan fungsi mengembalikan "BUKAN PRIMA".
- Jika a tidak habis dibagi dengan b, maka fungsi prima dipanggil kembali dengan a dan b dikurangi 1.
- Jika b kurang dari atau sama dengan 1, maka bilangan tersebut prima, dan fungsi mengembalikan "PRIMA".
- Input pengguna diminta untuk memasukkan angka.
- Fungsi prima dipanggil dengan argumen yang diambil dari input pengguna.
- Hasil dari pemanggilan fungsi prima dicetak.

SOAL 2

Membuat fungsi rekursif untuk mengecek apakah sebuah kalimat palindrom atau bukan.

a. Source code

```python
def palin(k):
    if len(k) == 0:
        return "PALINDROM"
    else:
        if k[0] == k[-1]:
            return palin(k[1:-1])
        else:
            return "BUKAN PALINDROM"

k = str(input("Masukkan kalimat :"))
print(palin(k))
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads
Masukkan kalimat :hah
PALINDROM
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads
Masukkan kalimat :ha
BUKAN PALINDROM
```

c. Penjelasan

- Fungsi palin didefinisikan dengan satu parameter k.
- Di dalam fungsi, dilakukan pengecekan apakah panjang k sama dengan 0. Jika ya, maka fungsi mengembalikan "PALINDROM".
- Jika panjang k tidak sama dengan 0, maka dilakukan pengecekan apakah karakter pertama (k[0]) sama dengan karakter terakhir (k[-1]) dari string k.
- Jika karakter pertama dan terakhir sama, maka fungsi dipanggil kembali dengan parameter k[1:-1]. Hal ini menghapus karakter pertama dan terakhir dari string k.
- Proses di langkah 3 dan 4 diulangi secara rekursif hingga panjang string k menjadi 0 atau ditemukan perbedaan antara karakter pertama dan terakhir.
- Jika ditemukan perbedaan pada langkah 3, maka fungsi mengembalikan "BUKAN PALINDROM".
- Input pengguna diminta untuk memasukkan sebuah kalimat.
- Fungsi palin dipanggil dengan argumen yang diambil dari input pengguna.
- Hasil dari pemanggilan fungsi palin dicetak.

SOAL 3

Mmebuat fungsi rekursif untuk menghitung jumlah deret ganjil.

a. Source code

```
def jumlah(x,y):
    if x > y:
        return 0
    return x + jumlah(x + 2,y)

x = 1
y = int(input("Masukkan n:"))
print(jumlah(x,y))
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads
Masukkan n:6
9
```

c. Penjelasan

- Fungsi jumlah didefinisikan dengan dua parameter, x dan y.
- Di dalam fungsi, dilakukan pengecekan apakah x lebih besar dari y. Jika ya, maka fungsi mengembalikan 0.
- Jika tidak, fungsi akan mengembalikan nilai x ditambah dengan pemanggilan kembali fungsi jumlah dengan parameter x + 2 dan y.
- Langkah 3 ini menghasilkan pemanggilan rekursif hingga kondisi pada langkah 2 terpenuhi.
- Input pengguna diminta untuk memasukkan nilai y.
- Variabel x diinisialisasi dengan nilai 1.
- Fungsi jumlah dipanggil dengan x dan nilai y yang diambil dari input pengguna.
- Hasil dari pemanggilan fungsi jumlah dicetak.

SOAL 4

Membuat fungsi rekursif untuk mengecek jumlah digit sebuah bilangan. Contohnya “234” jumlah digitnya 2 + 3+ 4 = 9

a. Source code

```
def jmlhdigit(x):
    y = len(x)-1
    if y == 0:
        return x
    else:
        z = int(x[-1])
        x = x[:-1]
```

```
        return z + int(jmlhdigit(x))

x = (input("Masukkan bilangan :"))
print(jmlhdigit(x))
```

b. Output

c. Penjelasan

- Fungsi jmlhdigit didefinisikan dengan satu parameter x.
- Di dalam fungsi, variabel y diinisialisasi dengan panjang string x dikurangi 1.
- Dilakukan pengecekan apakah nilai y sama dengan 0. Jika ya, maka fungsi mengembalikan nilai x.
- Jika nilai y tidak sama dengan 0, dilakukan langkah-langkah berikut:
- Variabel z diinisialisasi dengan nilai integer dari karakter terakhir string x.
- Variabel x diupdate dengan menghilangkan karakter terakhirnya menggunakan slicing.
- Mengembalikan nilai penjumlahan antara nilai z dan hasil pemanggilan rekursif fungsi jmlhdigit dengan parameter x.
- Proses pada langkah 4 berulang hingga panjang string x menjadi 0.
- Input pengguna diminta untuk memasukkan sebuah bilangan.
- Fungsi jmlhdigit dipanggil dengan argumen yang diambil dari input pengguna.
- Hasil dari pemanggilan fungsi jmlhdigit dicetak.

SOAL 5

Membuat fungsi rekursif untuk menghuting kombinasi.

a. Source code

```
def c(a, b):
    if b == 0:
        return 1
    else:
        int((a - b + 1 ) / b)
        return (c(a , b - 1) * (a - b + 1) / (b))
```

```
a = int((input("Masukkan a :")))
b = int((input("Masukkan b :")))

print(c(a, b))
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Download
Masukkan a :5
Masukkan b :4
5.0
```

c. Penjelasan

- Fungsi c didefinisikan dengan dua parameter, a dan b.
- Di dalam fungsi, dilakukan pengecekan apakah b sama dengan 0. Jika ya, maka fungsi mengembalikan nilai 1.
- Jika nilai b tidak sama dengan 0, dilakukan langkah-langkah berikut:
- Dilakukan operasi pembagian integer antara (a - b + 1) dan b, namun nilai hasil operasi ini tidak disimpan atau dikembalikan.
- Fungsi dipanggil kembali secara rekursif dengan parameter a dan b - 1.
- Hasil dari pemanggilan rekursif ini dikalikan dengan (a - b + 1) dan dibagi dengan b, kemudian hasilnya dikembalikan.
- Proses pada langkah 3 berulang hingga nilai b menjadi 0.
- Input pengguna diminta untuk memasukkan nilai a dan b.
- Nilai a dan b diambil dari input pengguna dan diubah menjadi integer.
- Fungsi c dipanggil dengan argumen a dan b.
- Hasil dari pemanggilan fungsi c dicetak

Link Github:

https://github.com/rainiefch/Praktikum-Algoritma-Pemrograman_71230982/tree/35d55311c96d519c987fe50f1e856965e8c99bbc/Latihan%2013