Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230970**
Nama Lengkap	**Gregorius Daniel Jodan Perminas**
Minggu ke / Materi	**13 / Fungsi 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

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.

Pengertian Rekursif

Fungsi rekursif merupakan salah satu teknik pemrograman di Python yang memanggil dirinya sendiri. Fungsi rekursif merupakan fungsi matematis yang berulang dan memiliki pola yang terstruktur, namun biasanya fungsi ini perlu diperhatikan agar fungsi ini dapat berhenti dan tidak menghabiskan banyak memori. Fungsi ini akan terus berjalan hingga suatu kondisi terpenuhi. Fungsi rekursif terdiri dari dua bagian utama

- 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

Kelebihan dan Kekurangan Fungsi Rekursif

Kelebihan:

- Fungsi rekursif dapat menghasilkan kode program yang lebih singkat dibandingkan dengan penggunaan loop.
- Masalah kompleks dapat di breakdown menjadi sub masalah yang lebih kecil di dalam rekursif

Kelemahan:

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

Bentuk Umum Fungsi Rekursif

Bentuk umum fungsi rekursif Python dapat dilihat di bawah ini:

```
# FUNGSI REKURSIF
def fungsi_rekursif(parameter):
    # Base case
    if kondisi_base:
        return nilai_base

    # Rekursif case
    else:
        return ekspresi_rekursif
```

Studi Kasus Fungsi Rekursif

BILANGAN FAKTORIAL

```python
# Rekursif Faktorial
def faktorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * faktorial(n - 1)
print(faktorial(5))
```

Gambar 1 - Program Rekursif Bilangan Faktorial

```
120
```

Gambar 2 - Output Program Rekursif Bilangan Faktorial

Cara kerja program:

- Fungsi faktorial menerima argumen n.
- Jika n adalah 0 atau 1, fungsi tersebut akan langsung mengembalikan nilai 1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti.
- Jika n lebih besar dari 1, fungsi tersebut akan mengembalikan hasil perkalian n dengan faktorial (n-1). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri.

Pemanggilan di dalam fungsi:

- faktorial(5) return 5 * faktorial(4)
- faktorial(4) return 4 * faktorial(3)
- faktorial(3) return 3 * faktorial(2)
- faktorial(2) return 2 * faktorial(1)
- faktorial(1) adalah base case, yang return 1

Jadi faktorial(5) mengembalikan nilai 5 * 4 * 3 * 2 * 1 = 120

PERKALIAN DALAM BENTUK PENJUMLAHAN

```python
# Rekursif Perkalian
def perkalian(bil1, bil2):
    if bil2 == 1:
        print(bil1, end = " = ")
        return bil1
    else:
        print(bil1, end = " + ")
        return bil1 + perkalian(bil1, bil2 - 1)

print(perkalian(5, 5))
```

Gambar 3 - Program Rekursif Perkalian

```
5 + 5 + 5 + 5 + 5 = 25
```

Gambar 4 - Output Program Rekursif Perkalian

Cara kerja program:

- Fungsi perkalian menerima dua argumen yaitu bil1 dan bil2.
- Jika bil2 adalah 1, fungsi tersebut akan mencetak nilai bil1 dan string " = ". Hal ini dilakukan dengan perintah print(bil1, end = " = "). Parameter end dalam perintah print digunakan untuk menentukan karakter yang dicetak di akhir baris. Setelah itu fungsi akan mengembalikan nilai bil1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi akan berhenti setelah mencapainya.
- Jika bil2 lebih besar dari 1, fungsi tersebut akan mencetak nilai bil1 dan string " + ". Hal ini dilakukan dengan perintah print(bil1, end = " + "). Setelah itu fungsi akan mengembalikan hasil penjumlahan bil1 dengan perkalian(bil1, bil2 - 1). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi:

- Perkalian(5, 5) return 5 + perkalian(5, 4).
- perkalian(5, 4) return 5 + perkalian(5, 3).
- perkalian(5, 3) return 5 + perkalian(5, 2).
- perkalian(5, 2) return 5 + perkalian(5, 1).
- perkalian(5, 1) adalah base case jadi return bil1 yaitu 5.
- Jadi, perkalian(5, 5) mengembalikan 5 + 5 + 5 + 5 + 5 = 25

BILANGAN PERPANGKATAN

```python
# Rekursif Perpangkatan
def perpangkatan(bil1,bil2):
    if bil2 == 0:
        return 1
    elif bil2 == 1:
        print(bil1, end = " = ")
        return bil1
    else:
        print(bil1, end = " * ")
        return bil1 * perpangkatan(bil1, bil2-1)

print(perpangkatan(2,5))
```

Gambar 5 - Program Rekursif Bilangan Perpangkatan

```
2 * 2 * 2 * 2 * 2 = 32
```

Gambar 6 - Output Program Rekursif Bilangan Perpangkatan

Cara kerja program:

- Fungsi perpangkatan menerima dua argumen yaitu bil1 dan bil2.
- Jika bil2 adalah 0, fungsi tersebut akan langsung mengembalikan nilai 1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti.
- Jika bil2 adalah 1, fungsi tersebut akan mencetak nilai bil1 dan string " = ", lalu mengembalikan nilai bil1. Ini juga termasuk base case dari rekursi.
- Jika bil2 lebih besar dari 1, fungsi tersebut akan mencetak nilai bil1 dan string " * ", lalu mengembalikan hasil perkalian bil1 dengan perpangkatan(bil1, bil2 - 1). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi:

- perpangkatan(2, 5) return 2 * perpangkatan(2, 4).
- perpangkatan(2, 4) return 2 * perpangkatan(2, 3).
- perpangkatan(2, 3) return 2 * perpangkatan(2, 2).
- perpangkatan(2, 2) return 2 * perpangkatan(2, 1).
- perpangkatan(2, 1) adalah base case, jadi mencetak 2 = dan return 2.
- Jadi, perpangkatan(2, 5) mengembalikan 2 * 2 * 2 * 2 * 2 = 32

BILANGAN FIBONACCI

```
# Rekursif Fibonacci
def fibonacci (n):
    if n == 1 or n == 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

print(fibonacci(4))
```

Gambar 7 - Program Rekursif Bilangan Fibonacci

```
3
```

Gambar 8 - Output Program Rekursif Bilangan Fibonacci

Cara kerja program:

- Fungsi fibonacci menerima satu argumen yaitu n.
- Jika n adalah 1 atau 2, fungsi tersebut akan langsung mengembalikan nilai 1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti.
- Jika n lebih besar dari 2, fungsi tersebut akan mengembalikan hasil penjumlahan fibonacci(n-1) dan fibonacci(n-2). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi:

fibonacci(4)

= fibonacci(3) + fibonacci(2) *# Memanggil fungsi untuk n-1 dan n-2, karena n bukan 1 atau 2*

= (fibonacci(2) + fibonacci(1)) + fibonacci(2) *# Memecah fibonacci(3)*

= ((1) + (1)) + (1) *# fibonacci(2) dan fibonacci(1) keduanya melakukan return nilai 1*

= 3

PERMUTASI

```
# REKURSIF PERMUTASI
def permutasi(n, r):
    if r == 0:
        return 1
    else:
        return n * permutasi(n - 1, r - 1)

print(permutasi(5, 3))
```

Gambar 9 - Program Rekursif Permutasi

```
60
```

Gambar 10 - Output Program Rekursif Permutasi

Cara kerja program:

- Fungsi permutasi menerima dua argumen yaitu n dan r.
- Jika r adalah 0, fungsi tersebut akan langsung mengembalikan nilai 1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti. Ini karena permutasi n = *integer* dengan r = 0 hasilnya akan selalu 1.
- Jika r lebih besar dari 0, fungsi tersebut akan mengembalikan hasil perkalian n dengan permutasi(n - 1, r - 1). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi:

permutasi(5, 3)

= 5 * permutasi(4, 2) # Memanggil fungsi untuk n-1 dan r-1

= 5 * (4 * permutasi(3, 1)) # Memecah permutasi(4, 2)

= 5 * (4 * (3 * permutasi(2, 0))) # Memecah permutasi(3, 1)

= 5 * (4 * (3 * (1))) # Permutasi(2, 0) adalah base case maka mengembalikan 1

= 60

Referensi:

Fungsi Rekursif Python: Cara Menggunakan dan Contoh - Pemburu Kode

Fungsi Rekursi · Rumah Coding

Pengertian dan Contoh Bilangan Prima serta Cara Menentukannya (detik.com)

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

LINK GITHUB: https://github.com/danieljodan/PrakAlPro13_A_71230970.git

SOAL 1

PROGRAM LATIHAN MANDIRI 13.1

```
# Rekursif Check Prima
def prima(n, i = 2):
    if n == 2:
        return True
    if n < 2 or n % i == 0:
        return False
    if i * i > n:
        return True

    return prima(n, i + 1)

print(prima(7))
```

OUTPUT PROGRAM

```
True
```

Penjelasan Program:

Teori Untuk Mengecek Apakah Suatu Bilangan Prima:

- Cek apakah bilangan tersebut lebih besar dari 1, karena bilangan prima harus lebih besar dari angka 1.
- Cek apakah bilangan tersebut bisa habis dibagi oleh bilangan selain 1 dan dirinya sendiri. Cara mengeceknya, **mulai dari angka 2 hingga akar dari bilangan tersebut**, lalu periksa apakah ada bilangan yang membagi bilangan tersebut. Jika tidak ada bilangan yang membagi bilangan tersebut, maka bilangan tersebut adalah prima.

Berikut adalah cara kerja fungsi prima(n, i) pada program di atas:

- Fungsi prima menerima dua argumen yaitu n dan i.
- Jika n adalah 2, fungsi tersebut akan langsung mengembalikan nilai True. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti. Ini karena 2 adalah bilangan prima.
- Jika n kurang dari 2 atau n habis dibagi i (yaitu, n % i == 0), maka n bukan bilangan prima dan fungsi mengembalikan False.
- Jika i * i > n, maka kita sudah mencoba semua kemungkinan pembagi sesuai dengan teori sebelumnya maka n pasti bilangan prima, jadi fungsi mengembalikan True.

- Jika tidak ada dari kasus di atas yang berlaku, fungsi memanggil dirinya sendiri dengan i + 1 sebagai pembagi berikutnya untuk dicoba.

SOAL 2

PROGRAM LATIHAN MANDIRI 13.2

```
def palindrom(kalimat: str, counter: int = None):
    kalimat_lower = kalimat.lower()
    kalimat_alpha = [x for x in kalimat_lower if x.isalpha()]
    kalimat_bersih = "".join(kalimat_alpha)
    if counter is None:
        counter = len(kalimat_bersih)
    if counter <= 1:
        return True
    if kalimat_bersih[0] != kalimat_bersih[-1]:
        return False
    return palindrom(kalimat_bersih[1:-1], counter - 2)

print(palindrom("Katak."))
```

OUTPUT PROGRAM

```
True
```

Cara kerja program:

- Fungsi palindrom pertama kali dipanggil dengan argumen "Katak.". Fungsi ini kemudian mengubah semua huruf dalam kalimat menjadi huruf kecil dengan kalimat.lower() dan menghapus semua karakter non-alfabet dengan perintah [x for x in kalimat_lower if x.isalpha()], yang menghasilkan kalimat_bersih bernilai "katak".
- Karena counter adalah None, fungsi menetapkan counter ke panjang kalimat_bersih, yaitu 5.
- Karena counter lebih besar dari 1 dan karakter pertama dan terakhir dari kalimat_bersih sama (keduanya adalah ‘k’), fungsi memanggil dirinya sendiri dengan argumen "ata" dan counter dikurangi 2, yaitu 3 (counter disamakan dengan jumlah huruf)
- Dalam pemanggilan fungsi berikutnya, kalimat_bersih adalah "ata" dan counter adalah 3. Karena counter lebih besar dari 1 dan karakter pertama dan terakhir dari kalimat_bersih sama (keduanya adalah ‘a’), fungsi memanggil dirinya sendiri dengan argumen "t" dan counter dikurangi 2, yaitu 1.
- Dalam pemanggilan fungsi berikutnya, kalimat_bersih adalah "t" dan counter adalah 1. Karena counter kurang dari atau sama dengan 1, fungsi mengembalikan True yang berarti “Katak.” adalah palindrom.

SOAL 3

PROGRAM LATIHAN MANDIRI 13.3

```python
# Fungsi rekursif untuk menghitung jumlah deret ganjil
def deretGanjil(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    elif n % 2 != 0:
        return n + deretGanjil(n-2)
    else:
        return deretGanjil(n-1)

print(deretGanjil(10))
```

OUTPUT PROGRAM

```
25
```

Cara kerja program:

- Fungsi deretGanjil pertama kali dipanggil dengan argumen 10. Karena 10 adalah bilangan genap, fungsi memanggil dirinya sendiri dengan argumen 9.
- Dalam pemanggilan fungsi berikutnya, n adalah 9, yang merupakan bilangan ganjil. Oleh karena itu, fungsi mengembalikan 9 + deretGanjil(7).
- Dalam pemanggilan fungsi berikutnya, n adalah 7, yang juga merupakan bilangan ganjil. Oleh karena itu, fungsi mengembalikan 7 + deretGanjil(5).
- Proses ini berlanjut dengan fungsi yang memanggil dirinya sendiri dengan argumen 5, 3, dan 1, yang semuanya adalah bilangan ganjil.
- Akhirnya, ketika n adalah 1, fungsi mengembalikan 1, yang merupakan base case dari rekursi.
- Oleh karena itu, deretGanjil(10) mengembalikan 9 + 7 + 5 + 3 + 1 = 25.
- Jadi, deretGanjil(10) mengembalikan jumlah deret ganjil hingga 10, yaitu 25.

SOAL 4

PROGRAM LATIHAN MANDIRI 13.4

```python
# Fungsi rekursif untuk mengetahui jumlah digit dari suatu bilangan
def digit(n = 0):
    n = int(n)
    if n == 0:
        return 0
    else:
        return n % 10 + digit(n // 10)

print(digit("234"))
```

OUTPUT PROGRAM

```
9
```

Cara kerja program:

- Fungsi digit menerima satu argumen yaitu n. Argumen ini diubah menjadi integer menggunakan fungsi int().
- Jika n adalah 0, fungsi akan langsung mengembalikan nilai 0. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti. Ini karena jumlah digit dari 0 adalah 0.
- Jika n bukan 0, fungsi akan mengembalikan n % 10 + digit(n // 10). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi digit(23):

= 4 + digit(23) # Karena 234 bukan 0, kita memanggil fungsi digit(n // 10)

= 4 + (3 + digit(2)) # Memecah digit(23)

= 4 + (3 + (2 + digit(0))) # Memecah digit(2)

= 4 + (3 + (2 + 0)) # digit(0) adalah base case dan mengembalikan 0

= 9

SOAL 5

PROGRAM LATIHAN MANDIRI 13.5

```
# Fungsi rekursif untuk menghitung kombinasi
def kombinasi (n, r):
    if r == 0 or r == n:
        return 1
    else:
        return kombinasi(n-1, r-1) + kombinasi(n-1, r)

print(kombinasi(5, 2))
```

OUTPUT PROGRAM

```
10
```

Cara kerja program:

- Fungsi kombinasi menerima dua argumen yaitu n dan r.

- Jika r adalah 0 atau r sama dengan n, fungsi akan langsung mengembalikan nilai 1. Ini adalah base case dari rekursi, yang berarti kondisi di mana rekursi berhenti. Ini karena kombinasi dari n objek yang diambil 0 pada satu waktu atau n pada satu waktu selalu 1.
- Jika r bukan 0 dan r bukan n, fungsi akan mengembalikan kombinasi(n-1, r-1) + kombinasi(n-1, r). Ini adalah kasus rekursif, di mana fungsi memanggil dirinya sendiri dengan argumen yang berbeda.

Pemanggilan di dalam fungsi kombinasi(5, 2):

= kombinasi(4, 1) + kombinasi(4, 2) # Karena 2 bukan 0 dan 2 bukan 5, kita memanggil fungsi (n-1, r-1) dan (n-1, r)

= (kombinasi(3, 0) + kombinasi(3, 1)) + (kombinasi(3, 1) + kombinasi(3, 2)) # Memecah kombinasi(4, 1) dan kombinasi(4, 2)

= ((1) + (kombinasi(2, 0) + kombinasi(2, 1))) + ((kombinasi(2, 0) + kombinasi(2, 1)) + (kombinasi(2, 1) + kombinasi(2, 2))) # Memecah kombinasi(3, 1) dan kombinasi(3, 2)

= ((1) + ((1) + (kombinasi(1, 0) + kombinasi(1, 1)))) + (((1) + (kombinasi(1, 0) + kombinasi(1, 1))) + ((kombinasi(1, 0) + kombinasi(1, 1)) + (1))) # Memecah kombinasi(2, 1)

= ((1) + ((1) + ((1) + (1)))) + (((1) + ((1) + (1))) + (((1) + (1)) + (1))) # r == 0 atau r == n adalah base case dan mengembalikan 1

= 10