Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230978**
Nama Lengkap	**Jonathan Satriani Gracio Andrianto**
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%)

Pengertian Rekursif

Fungsi rekursif adalah fungsi yang berisi dirinya sendiri atau mendefinisikan dirinya sendiri. Fungsi ini sering kali disebut sebagai fungsi yang melakukan panggilan terhadap dirinya sendiri. Fungsi rekursif adalah jenis fungsi matematis yang berulang dengan pola yang terstruktur. Namun, perlu diperhatikan bahwa fungsi ini harus diatur agar bisa berhenti pada suatu titik dan tidak menyebabkan penggunaan memori berlebihan. Penggunaan fungsi rekursif harus dilakukan dengan hati-hati karena dapat menyebabkan infinite loop.

Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi, oleh karena itu dalam sebuah fungsi rekursif perlu terdapat 2 blok penting, yaitu blok yang menjadi titik berhenti dari sebuah proses rekursif dan blok yang memanggil dirinya sendiri. Di dalam rekursif terdapat 2 bagian:

Di dalam fungsi rekursif terdapat dua blok atau bagian, yaitu blok yang menjadi titik henti dari proses rekursif dan blok yang memanggil dirinya sendiri.

- **Base Case** : Kondisi yang menentukan kapan rekursi harus berhenti.
- **Rekursif Case** : Bagian rekursif di mana fungsi memanggil dirinya sendiri hingga mencapai Base Case

Kelebihan dan Kekurangan

Kelebihan dari fungsi rekursif :

1. Kode program lebih singkat dan elegan.
2. Masalah kompleks dapat dipecah menjadi submasalah yang lebih kecil di dalam rekursif.

Kekurangan dari fungsi rekursif :

1. Memakan memori yang lebih besar karena setiap kali fungsi dipanggil, dibutuhkan ruang memori tambahan.
2. Mengorbankan efisiensi dan kecepatan.
3. Fungsi rekursif sulit dilakukan debugging dan kadang sulit dimengerti.

Bentuk Umum dan Studi Kasus

Bentuk umum fungsi rekursif pada Python:

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

Sebenarnya, setiap fungsi rekursif pasti memiliki solusi iteratif. Sebagai contoh, pada kasus faktorial: Faktorial adalah hasil perkalian dari deret angka 1x2x3x ... x n. Algoritma untuk menghitung faktorial adalah sebagai berikut :

1. Tanyakan n
2. Siapkan variabel total untuk menampung hasil perkalian faktorial dan set nilai awal dengan 0
3. Loop dari i = 1 hingga n untuk mengerjakan:total = total * i
4. Tampilkan total

Dengan menggunakan fungsi rekursif maka faktorial dapat dihitung dengan rumus pada gambar 1.1

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

1.1 Rumus faktorial

Dari rumus 1.1 dapat dibuat pseudocode secara rekursif seperti pada gambar 1.2.

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
```

1.2 Pseudocode rekursif factorial

Dari pseudocode, maka kode Python yang dapat dibuat adalah :

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

print(faktorial(4))
```

Hasilnya adalah 24.

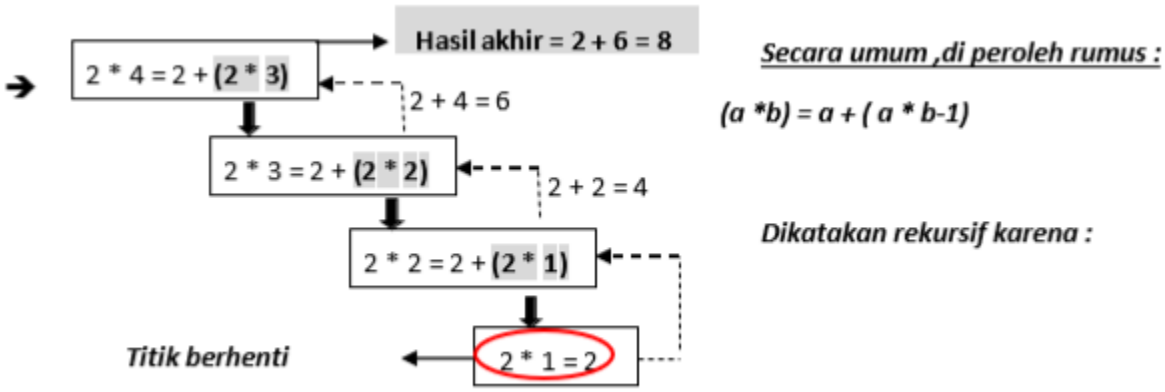
Gambaran prosesnya bisa dilihat di bawah :

```
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
```

1.3 Proses perhitungan faktorial rekursif

Kegiatan Praktikum

Kasus 1 Buatlah sebuah program yang dapat melakukan perkalian antara 2 buah bilangan dengan menggunakan fungsi rekursif. Misalkan kita ingin mengalikan angka 2 dengan 4. Dengan metode penjumlahan diperoleh 2*4 = 2 + 2 + 2 + 2 = 8.



1.4 Proses perhitungan perkalian rekursif

Dari proses pada gambar 1.4, kita dapat buat kode program :

```
def perkalian(bil1,bil2):
    if bil2==1:
        print(f"{bil1} = ",end='')
        return bil1
    else:
        print(f"{bil1} + ",end='')
        return bil1 + perkalian(bil1,bil2-1)

print(perkalian(2,4)) #output = 2 + 2 + 2 + 2 = 8
```

Kasus 2 Buatlah sebuah program yang dapat melakukan pemangkatan antara 2 buah bilangan dengan menggunakan fungsi rekursif. Misalkan kita ingin memangkatkan angka 2 dengan 4. Dengan metode penjumlahan diperoleh 2**4 = 2 * 2 * 2 * 2 = 16. Dapat dilihat logikanya hampir sama dengan 1.4 sebelumnya. Hanya saja operatornya diganti dengan * bukan +. Kode Program :

```
def pangkat(bil1,bil2):
    if bil2==1:
        print(f"{bil1} = ",end='')
        return bil1
    else:
        print(f"{bil1} * ",end='')
        return bil1 * pangkat(bil1,bil2-1)

print(pangkat(2,4)) #output = 2 * 2 * 2 * 2 = 16
```

Kasus 1.3 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.

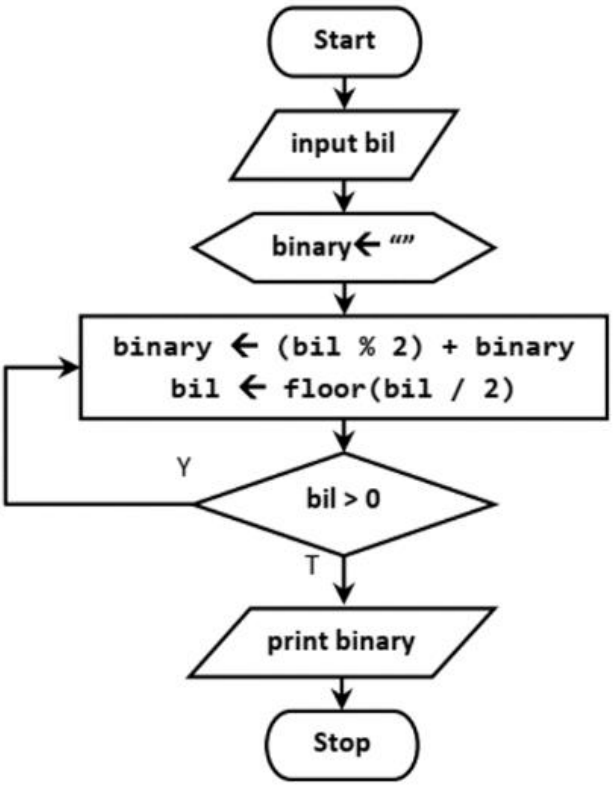
Bilangan fibonacci adalah bilangan yang berasal dari penjumlahan 2 bilangan sebelumnya. Secara rekursif rumus fibonacci adalah :

$$F(n) = \begin{cases} 1, \ n = 1 \ dan \ 2 \\ F(n-1) + F(n-2), \ n > 2 \end{cases}$$

1.5 Rumus Fibonacci

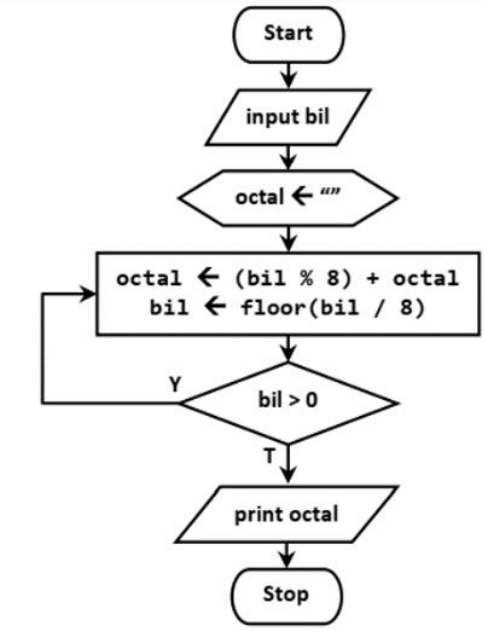
```
def fibo(n):
    if n==1 or n==2:
        return 1
    else:
        return fibo(n-1) + fibo(n-2)

print(fibo(9)) #output = 34
```

Kasus 4 Buatlah program yang dapat mengkonversi suatu bilangan dari basis 10 ke basis lainnya. Input berupa bilangan dalam bais 10 dan basis bilangan (selain basis 10). Program harus dibuat secara rekursif.

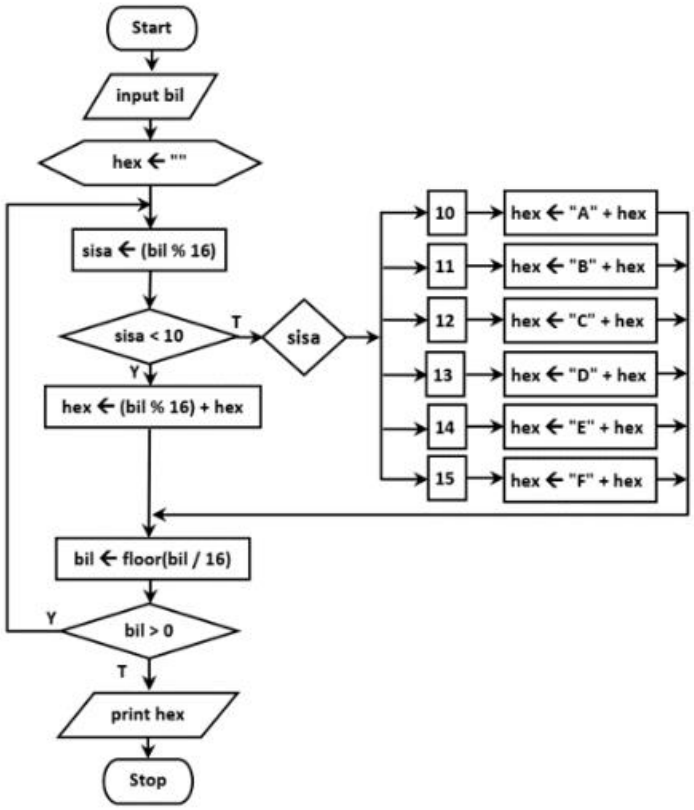


1.6 Desimal ke Biner

Sedangkan cara konversi ke basis 8 dilakukan sesuai gambar 1.7. Sedangkan cara konversi ke basis 16 dilakukan sesuai gambar 1.8.



1.7 Desimal ke Oktal



1.8 Desimal ke Hexa

Program yang dibuat dapat ditulis secara rekursif sebagai berikut :

```
def toBasis(n,base):
    convertString = "0123456789ABCDEF"
    if n < base:
        return convertString[n]
    else:
        return toBasis(n//base,base) + convertString[n % base]
print("Silahkan masukkan bilangan dan basis")
angka = int(input("Bilangan : "))
basis = int(input("Basis (2/8/16) : "))
print(toBasis(angka,basis))
'''
Bilangan : 9
Basis (2/8/16) : 8
11
'''
```

Kasus 5 Buatlah program untuk menghitung permutasi secara rekursif!. Permutasi memiliki rumus: P(n,r) = n! / (n-r)!

Untuk bisa menjawab soal tersebut kita harus tahu pola menghitung permutasi. Berikut contoh pola menghitung permutasi P(m,n): P(3,1) = P(3,0) * 3, jika n=0 maka m P(5,2) = P(5,1) * 4, jika n=0 maka m P(5,4) = P(5,3) * 2, jika n=0 maka m P(10,2) = P(10,1) * 9, jika n=0, maka m

Oleh karena itu berikut adalah kode programnya:

```
def permutasi (m, n):
    #m : batas atas dan n : batas bawah, dimana m > n
    if(n == 0):
    #jika n = 0, maka hasil adalah m!/(m-1)! = 1
        return 1
    else:
    #jika n > 1 panggil method permutasi secara rekursif dengan parameter n-1
sampai n8
        return (permutasi(m,n-1) * (m-n+1))
    #kembalikan nilai permuteRec dengan parameter m dan n-1 dikalikan dengan m-
n+1

print(permutasi(10,4)) #output = 5040
```

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

Source Code

```
def prima(bil, x = 2):
    if bil == 2 :
        return True
    elif bil < 2 or bil % x == 0 :
        return False
    elif x * x > bil :
        return True
    else:
        return prima(bil, x + 1)

n = int(input("masukkan bilangan = "))
if prima(n):
    print(f"{n} adalah bilangan prima")
else:
    print(f"{n} bukan bilangan prima")
```

Output

```
masukkan bilangan = 3
3 adalah bilangan prima
PS D:\kuliah\SEM 2\PR. Al
lap 14/lat1.py"
masukkan bilangan = 4
4 bukan bilangan prima
```

Penjelasan

Jadi kita buat fungsi prima dengan parameter ‘bil’ yaitu inputan user dan ‘x’ yaitu untuk mengecek bilangan prima atau tidak. Parameter ‘x’ sudah memiliki nilai default yaitu 2. Kemudian kita cek jika bil = 2 maka akan true. Jika bil kurang dari 2 atau bil habis dbagi x maka false. Terakhir jika x kuadrat lebih besar dari bil maka true. Dan elsenya berupa recursife case yaitu return prima bil,x+1 dimana x akan bertambah 1 jika kondisi sebelumnya tidak terpenuhi dan memeriksa pembagi berikutnya.

Soal 2

Source Code

```python
def palindrom(kalimat):
    if len(kalimat) < 2:
        return True
    elif kalimat[0] == kalimat[-1]:
        return palindrom(kalimat[1:-1])
    return False

kal = input("masukkan kalimat = ")
kal = kal.replace(' ', '').lower()
if palindrom(kal) :
    print("kalimat tersebut palindrom")
else :
    print("kalimat tersebut bukan palindrom")
```

Output

```
masukkan kalimat = kasurrusak
kalimat tersebut palindrom
```

Penjelasan

Pertama kita buat fungsi Bernama palindrom dengan parameter kalimat. Kemudian dalam fungsi tersebut kita buat base case Panjang kalimat kurang dari 2, karena sudah pasti palindrom. Kemudian pada elif kita beri kondisi array index 0 kalimat sama dengan aray index terakhir kalimat maka fungsi ini akan rekursi dan memanggil dirinya sendiri jika index kalimat awal dan akhir tidak sama akan

mengreturn nilai false. Terakhir kita minta input dari user dan inputan user tersebut kita hapus spasinya dan kita buat jadi huruf kecil semua agar mudah dibandingkan.

Soal 3

Source Code

```
def deret_ganjil(x):
    if x <= 0:
        return 0
    else:
        return (2 * x - 1) + deret_ganjil(x - 1)

bil = int(input("masukkan bilangan = "))
print(deret_ganjil(bil))
```

Output

```
masukkan bilangan = 4
16
```

Penjelasam

Pertama kita buat fungsi deret_ganjil dengan parameter ‘x’. Kita beri base casenya dengan kondisi x <= 0 karena sudah pasti hasilnya 0. Kemudian pada else kita returnkan nilai jumlah deret ganjil dengan rekursi yang bisa dilihat di source code.

Soal 4

Source Code

```
def jumlah_digit(x) :
    if x == 0 :
        return 0
    else :
        return x % 10 + jumlah_digit(x // 10)

bil = int(input("masukkan bilangan = "))
print(jumlah_digit(bil))
```

Output

```
masukkan bilangan = 234
9
```

Penjelasan

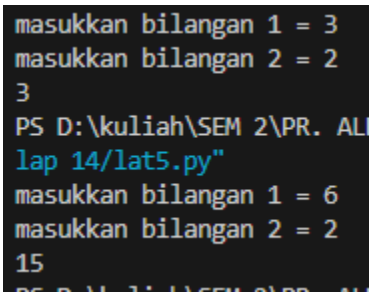
Pertama kita buat fungsi dengan nama jumlah_digit dengan parameter ‘x’. Base casenya adalah jika nilai x nya itu 0 karena sudah pasti hasilnya 0. Recursive casenya mengembalikkan nilai x % 10 + jumlah_digit(x // 10) untuk mengambil digit terakhir dan menjumlahkannya.

Soal 5

Source Code

```
def kombinasi(x, y):
    if y == 0 or x == y :
        return 1
    else :
        return kombinasi(x - 1, y - 1) + kombinasi(x - 1, y)

bil1 = int(input("masukkan bilangan 1 = "))
bil2 = int(input("masukkan bilangan 2 = "))
print(kombinasi(bil1, bil2))
```

Output



Penjelasan

Pertama kita buat fungsi yang Bernama kombinasi dengan parameter ‘x’ dan ‘y’. Base casenya mempunyai kondisi y = 0 atay x = y yang sudah pasti bernilai 1. Kemudian di else, kita lakukan rekursif dengan menambah kombinasi(x - 1, y - 1) dengan kombinasi(x - 1, y).

Link Github

https://github.com/JonathanAndrianto123/laporan_alpro13_71230978.git