13. Fungsi Rekursif

13.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menjelaskan tentang fungsi rekursif.
2. Dapat membaca dan menguraikan cara kerja fungsi rekursif.
3. Dapat menjelaskan beberapa kasus rekursif.
4. Dapat membuat fungsi rekursif.

13.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

13.3 Materi

13.3.1 Pengertian Rekursif

Fungsi rekursif adalah fungsi yang berisi dirinya sendiri atau fungsi yang mendefinisikan dirinya sendiri. Fungsi ini sering disebut sebagai fungsi 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 memori. Fungsi

rekursif merupakan fungsi yang harus digunakan secara hati-hati karena fungsi ini dapat bersifat unlimited loop sehingga menyebabkan program hang up.

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:

- **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

13.3.2 Kelebihan dan Kekurangan

Beberapa keunggulan fungsi rekursif adalah sebagai berikut:

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

Sedangkan kelemahan fungsi rekursif adalah:

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

13.3.3 Bentuk Umum dan Studi Kasus

Bentuk umum fungsi rekursif pada Python:

```
1    def function_name(parameter_list):
2        ...
3        function_name(...)
4        ...
```

Sebenarnya semua fungsi rekursif pasti memiliki solusi iteratifnya. Misalnya pada contoh kasus faktorial berikut: Faktorial adalah menghitung perkalian deret angka 1x2x3x ... x n. Algoritma untuk menghitung faktorial adalah:

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:
4. total = total * i
5. Tampilkan total

Dengan menggunakan fungsi rekursif maka faktorial dapat dihitung dengan rumus pada gambar 13.1

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

Gambar 13.1: Rumus faktorial

Dari rumus 13.1 dapat dibuat pseudocode secara rekursif seperti pada gambar 13.2
Dari pseudocode, maka kode Python yang dapat dibuat 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 faktorial

```
1  def faktorial(n):
2      if n==0 or n==1:
3          return 1
4      else:
5          return faktorial(n-1) * n
6
7  print(faktorial(4))
```

Hasil akhir adalah 24.

Bagaimana proses perhtiungan yang terjadi? Berikut adalah gambarannya:

```
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
```

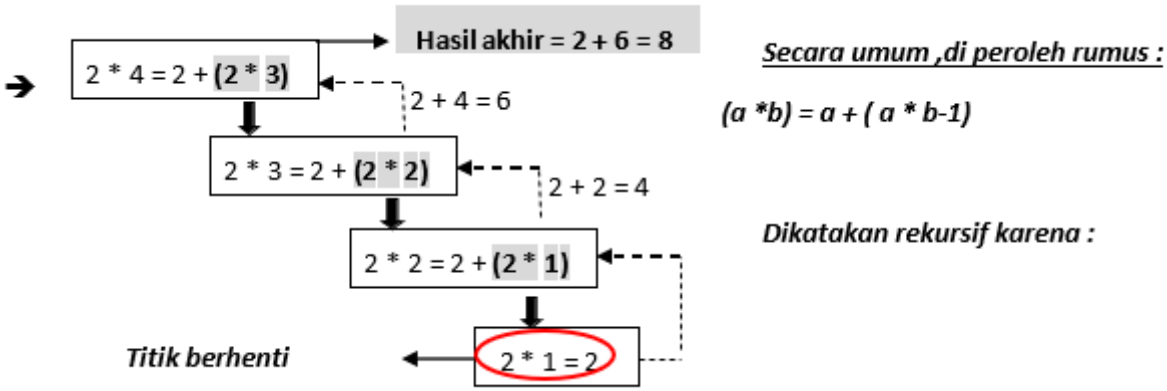
Gambar 13.3: Proses perhitungan faktorial rekursif

13.4 Kegiatan Praktikum

13.4.1 Problem dan Solusi 1

Pada kegiatan praktikum akan dilakukan beberapa percobaaan 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 2 dengan 4. Dengan metode penjumlahan diperoleh 2*4 = 2 + 2 + 2 + 2 = 8.



Gambar 13.4: Proses perhitungan perkalian rekursif

Untuk menjawab soal tersebut dapat dilihat logikanya pada gambar:
Kode program:

```
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(2,4))
```

Hasil: 2 + 2 + 2 + 2 = 8

13.4.2 Problem dan Solusi 2

Kasus 13.2 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.

Untuk menjawab soal tersebut dapat dilihat logikanya hampir sama dengan 13.4 sebelumnya. Hanya saja operatornya diganti dengan * bukan +.

Kode program:

```
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(2,4))
```

Hasil: 2 * 2 * 2 * 2 = 16

13.4.3 Problem dan Solusi 3

Kasus 13.3 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:

```
    def fibo(n):
        f1,f2=1,1
        print(f1,", ",f2,", ",end='')
        for i in range(2,n):
            fib = f1+f2
            f1 = f2
            f2 = fib
            print(fib,", ",end='')

    fibo(7)
```

Output adalah: 1, 1, 2, 3, 5, 8, 13

Secara rekursif rumus fibonacci adalah:

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

Gambar 13.5: Rumus Fibonacci

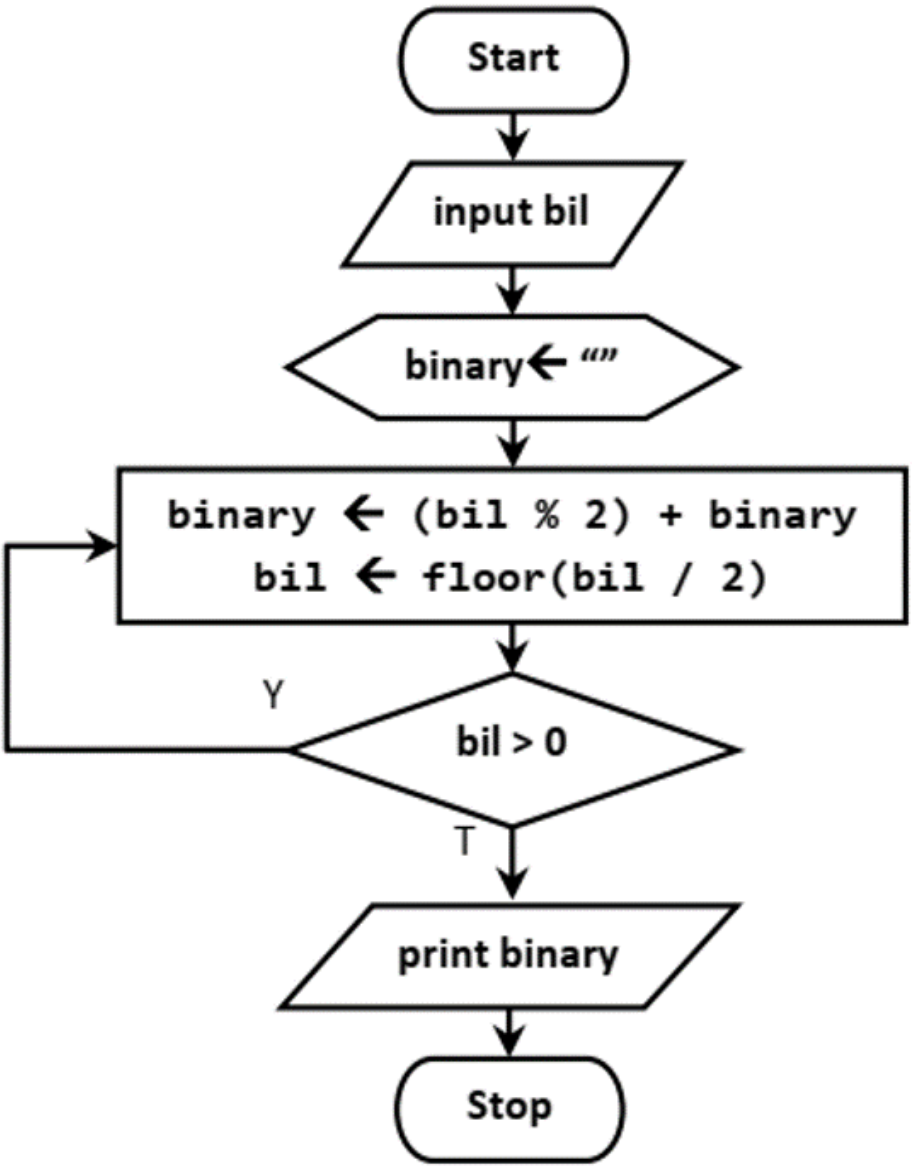
```
    def fibo(n):
        if n==1 or n==2:
            return 1
        else:
            return fibo(n-1) + fibo(n-2)

    print(fibo(7))
```

Hasil adalah 13

13.4.4 Problem dan Solusi 4

Kasus 13.4 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.



Gambar 13.6: Desimal ke Biner

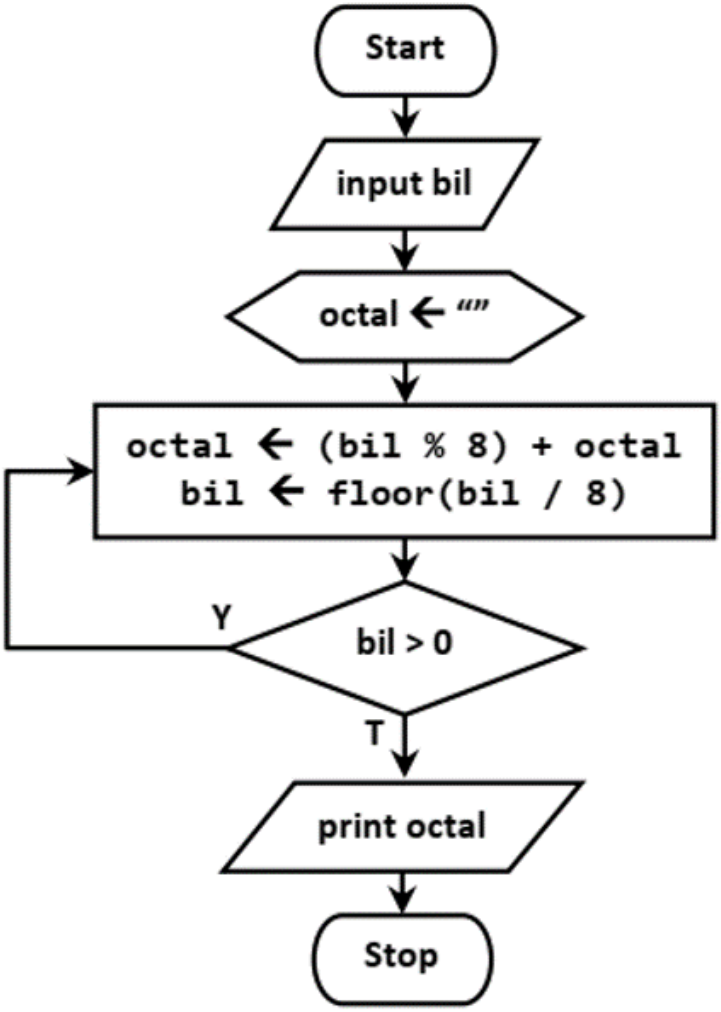
Untuk bisa menyelesaikan masalah ini, anda harus tahu cara mengkonversi basis bilangan dari basis 10 ke basis tertentu. Misalnya diketahui angka adalah 7 dalam basis 10, untuk konversi ke basis 2 dilakukan cara seperti pada gambar 13.6

Sedangkan cara konversi ke basis 8 dilakukan sesuai gambar 13.7

Sedangkan cara konversi ke basis 16 dilakukan sesuai gambar 13.8

Program yang dibuat dapat ditulis secara rekursif sebagai berikut:

```
1 def toBasis(n,base):
2     convertString = "0123456789ABCDEF"
3     if n < base:
4         return convertString[n]
5     else:
```



Gambar 13.7: Desimal ke Oktal

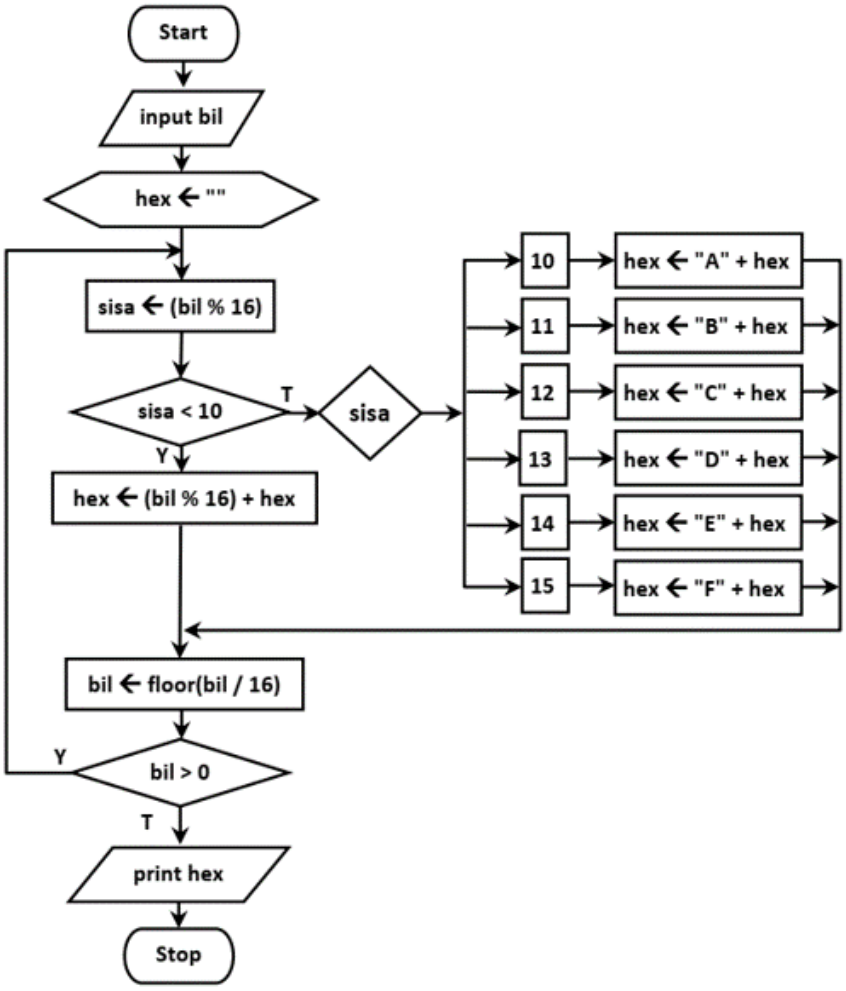
```
6           return toBasis(n//base,base) + convertString[n\%base]
7
8   print("Silahkan masukkan bilangan dan basis")
9   angka = int(input("Bilangan : "))
10  basis = int(input("Basis (2/8/16) : "))
11  print(toStr(angka,basis))
```

Contoh Output:

```
Bilangan : 9
Basis : 8
11
```

Penjelasan:

- Program di atas dimulai dengan memasukkan parameter bilangan dan basisnya dalam integer
- Program akan menyiapkan string yang berisi angka seluruh basis bilangan, dari 0-9, A-F.
- Program akan memeriksa apakah n < basis, jika iya maka kembalikan posisi string[n]
- Program akan menjalankan rekursif case dengan cara membagi bilangan / basis dan ditambahkan (dijejerkan) dengan string[n%basis]



Gambar 13.8: Desimal ke Hexa

13.4.5 Problem dan Solusi 5

Kasus 13.5 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 n
        return (permutasi(m,n-1) * (m-n+1))
        #kembalikan nilai permuteRec dengan parameter m dan n-1 dikalikan dengan m-n+1

```

```
11 print(permutasi(10,4))
```

13.5 Latihan Mandiri

Latihan 13.1 Vidi adalah adik Tono yang sedang belajar bilangan prima. Vidi mengalami kesulitan untuk menentukan suatu bilangan bilangan prima atau bukan. Untuk membantu adiknya Tono kemudian membuat program untuk pengecekan bilangan prima dengan menggunakan fungsi rekursif. Bantulah Tono untuk menyelesaikan tugas tersebut. ■

Latihan 13.2 Buatlah fungsi rekursif mengetahui suatu kalimat adalah palindrom atau bukan! ■

Latihan 13.3 Buatlah fungsi rekursif untuk menghitung jumlah deret ganjil dari 1 + 3 + 7 + . . . + n! ■

Latihan 13.4 Buatlah fungsi rekursif untuk mengetahui jumlah digit dari suatu bilangan. Seperti misalnya tulisan: "234" maka jumlah digitnya adalah 2+3+4 = 9! ■

Latihan 13.5 Buatlah fungsi rekursif untuk menghitung kombinasi! ■