Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**<71230972>**
Nama Lengkap	**<Oktavian Christ Putranto>**
Minggu ke / Materi	**13 / Fungsi Rekrusif**

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.

Rekursif

Rekrusif sendiri sederhananya ialah bentuk perulangan yang lebih keren/fancy Dimana terdapat pemanggilan fungsi yang memanggil dirinya sendiri dalam pembentukan fungsi rekrusif kita harus perlu ekstra hati hati dalam penempatan base case(Base case adalah bagian yang mendefinisikan fungsi rekursif untuk berhenti.) dan rekrusif case (Recursive case adalah bagian yang instruksinya akan diulang terus menerus hingga mencapai base case)

Dalam penggunaan fungsi rekresif sendiri kita perlu ekstra hati-hati dikarenakan terdapat perulangan tak terbatas yang dapat menyebabkan program mogok dan dalam strukturnya sendiri kita justru akan menjumpai if else sebagai bentuk perulangannya.

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 fungsi rekursif adalah:

- Sulit dipahami
- Tidak mudah melakukan debugging pada fungsi rekrusif
- Memakan memori yang lebih besar karena setiap kali bagian dirinya dipanggil maka dibutuhkan sejumlah ruang memori tambahan.
- Mengorbankan efisiensi dan kecepatan.

Contoh bentuk fungsi rekrusif:

```
def faktorialganjil(jumlah,awal):
    if awal==0:
        awal=1
    if jumlah==0:
        return 1
    else:
        return awal*faktorialganjil(jumlah-1,awal+2) #rekrusif case
```

```
print(faktorialganjil(5,0))
print(faktorialganjil(2,0))
print(faktorialganjil(10,0)
```

Output :

945

3

654729075

Penjelasan :

Dari code tersebut dapat kita perhatikan jelas bahwa terdapat pemanggilan fungsi itu sendiri untuk melakukan perulangan, bentuk pemanggilan fungsi inilah yang disebut sebagai rekrusif case. Dan perlu diperhatikan bahwa rekrusif harus selalu mengarah mendekati base case dalam kasus ini mendekati kondisi Dimana

```
if jumlah==0:
        return 1
```

oleh karena itu dibentuk rekrusif case seperti ini

```
else:
        return awal*faktorialganjil(jumlah-1,awal+2) #rekrusif case
```

hal ini perlu diperhatikan dengan teliti karena apabila terjadi kesalahan dalam rekrusif case maka yang terjadi ialah perulangan nonstop hingga akhirnya dipaksa untuk berhenti

contoh rekrusifcas eyang salah :

```
def faktorialganjil(jumlah,awal):
    if awal==0:
        awal=1
    if jumlah==0:
        return 1
    else:
        return awal*faktorialganjil(jumlah,awal+2) #rekrusif case


print(faktorialganjil(5,0))
print(faktorialganjil(2,0))
print(faktorialganjil(10,0))
```

Output :

```
Traceback (most recent call last):
  File "c:\Users\ASUS VivoBook\Downloads\rekrusif\ganjil.py", line 10, in <module>
    print(faktorialganjil(5,0))
  File "c:\Users\ASUS VivoBook\Downloads\rekrusif\ganjil.py", line 7, in faktorialganjil
    return awal*faktorialganjil(jumlah,awal+2) #rekrusif case
  File "c:\Users\ASUS VivoBook\Downloads\rekrusif\ganjil.py", line 7, in faktorialganjil
    return awal*faktorialganjil(jumlah,awal+2) #rekrusif case
  File "c:\Users\ASUS VivoBook\Downloads\rekrusif\ganjil.py", line 7, in faktorialganjil
    return awal*faktorialganjil(jumlah,awal+2) #rekrusif case
  [Previous line repeated 995 more times]
  File "c:\Users\ASUS VivoBook\Downloads\rekrusif\ganjil.py", line 2, in faktorialganjil
    if awal==0:
RecursionError: maximum recursion depth exceeded in comparison
```

Penjelasan :

Seperti contoh tersebut ketika rekrusif case tidak mengarah ke titik stop/base case maka eror seperti itulah yang akan muncul sebagai outputnya.

Kekurangan fungsi rekrusif dalam segi waktu :

Code fibonachi fungsi rekrusif:

```python
import time
start = time.time()

def fibo(n):
    if n==1 or n==2:
        return 1
    else:
        return fibo(n-1) + fibo(n-2)
print(fibo(35))
```

```
end = time.time()
length = end - start
print("waktu yang dibutuhkan adalah", length, "detik")
```

Output :

9227465

waktu yang dibutuhkan adalah 1.413733959197998 detik

ketika input diubah menjadi

```
print(fibo(40))
```

Outputnya :

102334155

waktu yang dibutuhkan adalah 15.130281686782837 detik

terlihat terjadi peningkatan yang signifikan dari segi waktu yang dibutuhkan

sedangkan untuk code fibonachi biasa tanpa fungsi rekrusif tidak perlu waktu selama itu contoh:

```
import time
start = time.time()
def fibo(n):
    f1,f2=1,1
    for i in range(2,n):
        fib = f1+f2
        f1 = f2
        f2 = fib
    print(fib,", ",end='')
fibo(35)
end = time.time()
length = end - start
print("waktu yang dibutuhkan adalah", length, "detik")
```

Output : 9227465 , waktu yang dibutuhkan adalah 0.0 detik

Saat input diubah menjadi :

```
fibo(120)
```

maka waktunya tetap sama yaitu 5358359254990966640871840 , waktu yang dibutuhkan adalah 0.0 detik

dapat disimpulkan fungsi fibonachi tanpa rekrusihf jauh lebih memakan waktu bahkan hingga berkali kali lipat jumlahnya.

Cara kerja fungsi rekrusif untuk contoh kasus factorial:

Code :

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

Output : 24

Cara kerja :

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
```

Seperti terlihat dari ilustrasinya pada tiap tahapannya maka diambil dari factorial (4) yang didalamnya disusun oleh angka 4*factorial(3) dst…. Hingga terbuka basecase diamana factorial(1) bernilai 1 sehingga dari 4*3*2*1 akan dijumlahkan seperti ilustrasi diatas.

SOAL 1

Kita diminta untuk membuat program untuk pengecekan bilangan prima dengan menggunakan fungsi rekursif.

Code :

```
def cek_prima(a,b=2):
    if a <= 1:
        return ("bukan prima")
    elif b**2 > a:
        return ("prima")
    elif a % b == 0:
        return ("bukan prima")
    else:
        return cek_prima(a, b + 1)

print(cek_prima(97))
rint(cek_prima(100))
print(cek_prima(101))
print(cek_prima(2))
print(cek_prima(3))
```

Output :

prima

bukan prima

prima

prima

prima

penjelasan :

```
return cek_prima(a, b + 1)
```

code ini merupakan bagian rekrusif case untuk mengarah ke base casenya yaitu :

```
      elif b**2 > a:
            return ("prima")
      elif a % b == 0:
            return ("bukan prima")
```

dari 2 kemungkinan itu maka akan menghasilkan nilai prima ketika a dapat habis dibagi oleh b dengan nilai awal default=2 dan akan terus bertambah tiap rekrusifnya berjalan maka akan mengahasilkan nilai bukan prima ketika b kuadrat sudah melebihi a dan belum ditemukan kondisi dimana a%b==0 maka akan keluar print (“prima”)

SOAL 2

Buatlah fungsi rekursif mengetahui suatu kalimat adalah palindrom atau bukan!

Code :

```
# rekrusif palindrom
def palindrom(huruf,awal=0,akhir=0):
    if akhir==0:
        akhir=len(huruf)-1
    if huruf[awal] != huruf[akhir]:
        return (f"{huruf} bukan palindrom")
    if awal >= akhir:
        return (f"{huruf} palindrom")
    else:
        return palindrom(huruf,awal+1,akhir-1)

print(palindrom("kak"))
print(palindrom("khukhu"))
print(palindrom("popoopop"))
```

output :

kak palindrom

khukhu bukan palindrom

popoopop palindrom

penjelasan :

cara kerja code ini pada dasarnya mengecek huruf awal dan dibandingkan dengan huruf akhirnya dengan basecase berupa 2 kemungkinan yaitu saat awal sudah melebihi akhir maka akan mengindikasikan bahwa kata itu merupakan palindrom namun saat indeks huruf [awal] dan [akhir] tidak sama maka akan keluar hasil bukan palindrom.

```
    if huruf[awal] != huruf[akhir]:
        return (f"{huruf} bukan palindrom")
    if awal >= akhir:
        return (f"{huruf} palindrom")
```

rekrusif case :

```
else:
        return palindrom(huruf,awal+1,akhir-1)
```

setiap melalui else maka rekrusif case akan ter triger dan menjalankan fungsi itu kembali dgn parameter awal +1 dan akhir -1 sehingga akan mendekati base caenya.

```
if akhir==0:
        akhir=len(huruf)-1
```

code ini untuk memberi indeks terakhir dari huruf pada variable akhir.

SOAL 3

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

Sejujurnya saya masih ga paham maksud soalnya karena setelah angka 7 bisa jadi angka 15 karena 2^3 ataupun bisa jadi angka 13 dari 2*3 dan karena soalnya deret ganjil saya jadi semakin bingung sehingga akhirnya saya buat code sebagai berikut :

```
def faktorialganjil(n,awal=1):
    if n%2==0:
        return ("n bukan bilangan ganjil")
    if n==1:
        return 1
    else:
        return n+faktorialganjil(n-2,awal)

print(faktorialganjil(5))
print(faktorialganjil(7))
print(faktorialganjil(12))
```

code ini menjumlahkan deret ganjil dari angka terbesarnya apa dan menjumlahkan deret bilangan ganjilnya.

Output :

9

16

n bukan bilangan ganjil

SOAL 4

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

Code :

```
def jum_digit(bil,bagi=10):
    if bil//bagi < 1:
        return bil
    else:
        belakang=bil%bagi
        bil=bil//bagi
        return belakang+jum_digit(bil,bagi)

print(jum_digit(231234))
print(jum_digit(123))
print(jum_digit(234))
```

output :

15

6

9

Penjelasan :

Ide dasar code ini ialah dengan mengambil digit terakhir dari bilangan nya memakai % (modulo) bilangan kemudian bilangan akan diubah untuk mengambil sisa digit selain digit terakhirnya dan dijumlahkan lalu berulang hingga mencapai basecase dan kemudian dijumlahkandengan belkang/digit akhir tiap angka untuk mendapatkan hasil akhirnya.

SOAL 5

Buatlah fungsi rekursif untuk menghitung kombinasi!

Code :

```
def kombinasi(a,b):
    if a < b:
        return "a harus lebih besar dari b"
    if b == 0:
        return 1
    if b == 1:
        return a
    if a == b:
        return 1
    else:
        return kombinasi(a - 1, b) + kombinasi(a - 1, b - 1)
```

```
print(kombinasi(7,3))
print(kombinasi(7,1))
print(kombinasi(10,3))
```

output :

35

7

120

Penjelasan:

```
if a < b:
        return "a harus lebih besar dari b"
    if b == 0:
        return 1
    if b == 1:
        return a
    if a == b:
        return 1
```

seluruh basecase tersebut menghasilkan nilai yang ditentukan.

Dan rekrusif casenya ialah :

```
else:
        return kombinasi(a - 1, b) + kombinasi(a - 1, b - 1)
```