Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71231023**
Nama Lengkap	**Yehezkiel Darren Putra Wardoyo**
Minggu ke / Materi	**14/Recursive**

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

Rekursif atau fungsi rekursif adalah sebuah fungsi yang berisi dirinya sendiri atau fungsi yang mendefinisikan dirinya sendiri. Hal ini berbeda dengan fungsi iteratif yang menggunakan perulangan **for** atau **while** untuk menyelesaikan tugas. Fungsi rekursif sendiri bersifat matematis yang berulang dan memiliki pola yang terstuktur. Fungsi rekursif sendiri biasanya digunakan untuk memecahkan persoalan matematika yang berulang seperti perkalian, penjumlahan, faktorial, dll. Ketika kita ingin menggunakan fungsi rekursif ada hal yang harus diperhatikan yaitu titik henti atau kondisi berhenti. Jika sebuah fungsi rekursif tidak memiliki titik henti maka fungsi akan tidak berhenti dan menghasilkan recurssion error seperti pada contoh berikut:

```
def helloworld(n):
    print(n)
    helloworld(n)

helloworld("print")
```

Program akan tetap dijalankan dan akan menghasilkan kata ”print”, namun program tetap akan berjalan sampai menghasilkan *RecursionError.* Kondisi ini dinamakan *hang up*.

```
RecursionError: maximum recursion depth exceeded while calling a Python object
```

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:

1. Base Case adalah bagian dimana penentu bahwa fungsi rekursif itu berhenti.
2. Rekursif Case adalah bagian dimana terdapat statement yang akan terus diulang-terus menerus hingga mencapai Base Case

Contoh dari program rekursif yang baik adalah sebagai berikut:

```
def perkalian(bil1,bil2):
    if bil2==1:
        return bil1
    else:
        return bil1+perkalian(bil1,bil2-1)
```

Pada contoh program diatas, fungsi perkalian memiliki struktur rekursif yang baik. Fungsi tersebut memilki dua kriteria sebuah program rekursif yang baik, kondisi berhenti dan kondisi rekursi/memanggil dirinya sendiri. Kondisi berhenti pada fungsi diatas yaitu ketika bil2 memiliki nilai sama dengan 1 maka menghasilkan nilai bil1. Kondisi rekursi di atas yaitu menjumlahkan bil1 dengan fungsi perkalian dengan parameter bil1 dan bil2-1.

Kelebihan dan Kekurangan

Fungsi rekursif memilki keunggulan dan juga kelemahannya. Keunggulan dari fungsi rekursif adalah :

1. Kode program lebih terlihat singkat dan elegan. Dibandingkan dengan menggunakan perulangan **for** dan **while**, fungsi rekursif lebih mempersingkat isi code. Sebagai contoh berikut ini :

```
def faktorialfor(bil1):
    hasil=1
    while bil1>0:
        hasil*=bil1
        bil1-=1
    return hasil
```

Jika diubah dalam bentuk rekursif akan menjadi seperti ini:

```
def faktorialrekur(bil1):
    if bil1<=0:
        return 1
    return bil1*faktorialrekur(bil1-1)
```

2. Masalah kompleks dapat dibreakdown menjadi sub-masalah yang lebih kecil di dalam rekursif. Contoh sebagai berikut:

```
def fibonacci_recursive(n):
      if n <= 1:
            return n
      else:
            return
      fibonacci_recursive(n-1)      +
      fibonacci_recursive(n-2)
```

jika mengambil contoh n = 4 maka akan didapatkan diagram treenya sebagai berikut:

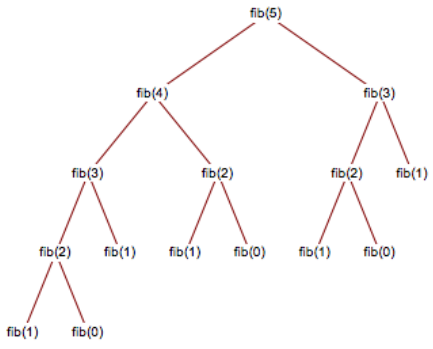

Diagram Tree Fibbonaci

Dengan menggunakan fungsi rekursi, kita dapat memecah problem yang memiliki nilai besar menajdi beberapa masalah yang lebih kecil.

Sedangkan kekurangan dari fungsi rekursif adalah sebagai berikut:

1. Memakan memori yang lebih besar karena setiap kali bagian dirinya dipanggil maka dibutuhkan sejumlah ruang memori tambahan. Pengunaan perulangan lebih menghemat penggunaan memori dalam menjalankan program dibandingkan penggunaan fungsi rekursif.
2. Mengorbankan efisiensi dan kecepatan. Menggunakan metode rekursif berarti membagi problem yang besar menjadi problem yang lebih kecil dan kecil lagi sampai tidak bisa dibagi lagi. Setelah semua problem tidak bisa dibagi lagi, maka komputer baru mulai memproses dari problem terkecil dahulu sampai ke bagian yang kompleks.
3. Fungsi rekursif sulit dilakukan debugging dan kadang sulit dimengerti.

Bentuk Umum dan Studi Kasus

Bentuk umum dari fungsi rekursif pada Python sebagai berikut:

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

Fungsi rekursif sebenarnya adalah sebuah fungsi yang iteratif, yang berarti selalu bertambah sampai kepada titik henti. Sebagai contoh, saya akan menampilkan sebuah program rekursif untuk menghitung bilangan eksponen. Eksponen adalah menghitung total perkalian keseluruhannya dari 1 sampai n (a x a x a x ....x a, sebanyak n buah). Kita bisa menerapkannya pada bentuk perulangan dan juga rekursif. Algoritma dari menghitung bilangan eksponen yaitu:

1. Tanyakan bilangan 1 sebagai angka yang ingin di kalikan dan bilangan 2 sebagai banyaknya jumlah perkalian bilangan 1.
2. Siapkan variabel kosong yang akan digunakan untuk menampung semua hasil perkalian tersebut.
3. Loop dari i=1 hingga n untuk mengerjakan:
4. total=total*bilangan 1
5. tampilkan total

Contoh implementasi kode adalah sebagai berikut:

```
def eksponenfor(bil,n):
    total=1
    for i in range(n):
        total*=bil
    return total
```

Jika diubah dalam bentuk rekursif maka akan menjadi sebagai berikut:

```
def eksponenrekur(bil,n):
    if n==1:
        return bil
    return bil*eksponenrekur(bil,n-1)
```

Bentuk eksponen rekursif memiliki struktur yaitu titik henti ketika n sama dengan 1 akan menghasilkan nilai bil dan struktur rekursifnya yaitu bil*fungsi eksponenrekur dengan parameter bil dan n-1.

Contoh berikutnya adalah menentukan apakah sebuah bilangan termasuk bilangan prima atau tidak. Saya akan menjelaskan dalam 2 metode, perulangan dan rekursi. Algoritmanya adalah sebagai berikut:

1. Tanyakan n
2. Periksa apakah bilangan tersebut kurang dari atau sama dengan 1. Jika ya, kembalikan False karena bilangan prima lebih besar dari 1.
3. Periksa apakah bilangan tersebut sama dengan 2. Jika ya, kembalikan True karena 2 adalah bilangan prima.
4. Periksa apakah angkanya genap. Jika ya, kembalikan False karena semua bilangan genap yang lebih besar dari 2 bukanlah bilangan prima.
5. Mulai perulangan dari 3 hingga akar kuadrat dari angka tersebut.
6. Periksa apakah bilangan tersebut habis dibagi dengan bilangan saat ini dalam perulangan. Jika ya, kembalikan False karena bilangan tersebut bukan bilangan prima.
7. Jika perulangan selesai tanpa menemukan faktor apa pun, kembalikan True karena bilangan tersebut adalah bilangan prima.

Perulangan:

```
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True
```

Rekursi:

```
def is_prime(number, divisor):
    if number <= 1:
        return False
    if divisor == 1:
```

```
        return True
    if number % divisor == 0:
        return False
    return is_prime(number, divisor - 1)
```

Pada bagian fungsi rekursi terdapat dua jenis kondisi, kondisi titik henti dan kondisi rekursi. Kondisi berhenti pada fungsi tersebut adalah number kurang dari sama dengan 1, divisor sama dengan 1, number%divisor sama dengan 0. Jika number kurang dari sama dengan 1 atau number%divisor sama dengan 0 yang terpenuhi maka menghasilkan False. Jika divisor sama dengan 1 terpenuhi maka menghasilkan True. Kondisi rekursi adalah memanggilkan fungsi itu sendiri dengan nilai divisor berkurang satu.

BAGIAN 2: LATIHAN MANDIRI (60%)

Link repo : https://github.com/YehezkielDarren/Teori-PrakAlPro/tree/b692c8101d0cf618d4a2db62a195e5371f4b1634/Pertemuan-14

SOAL 1

```python
def primeNumber(number,i=2):
    if number>i:
        if number%i==0:
            return False
        return primeNumber(number,i+1)
    return False if number<i else True
print(primeNumber(11))
```

Output:

```
True
```

Penjelasan:

1. Pada fungsi ini terdapat dua kondisi, base case dan recursive case.
2. Base casenya yaitu menghasilkan nilai False jika number kurang dari 2 dan selain itu mengembalikan True.
3. Recursive case terjadi ketika number lebih dari 2. Kondisi terpenuhi maka akan mengecek kembali apakah number habis dibagi dengan divisornya, jika ya maka akan False, jika tidak maka akan memanggil dirinya sendiri lagi dengan nilai divisor bertambah 1.

SOAL 2

```python
def palindrom(kata):
    if len(kata)<=1:
        return True
    if kata[0]!=kata[-1]:
        return False
    else:
        return palindrom(kata[1:-1])

print(palindrom("kasurrusak"))
```

Output :

```
True
```

Penjelasan:

1. Pada fungsi ini terdapat dua kondisi, base case dan recursive case. Base case terjadi ketika panjang kata kurang dari sama dengan 1 atau huruf awal tidak sama dengan huruf terakhir.
2. Jika panjang kata kurang dari sama dengan 1 maka akan menghasilkan True.
3. Jika huruf awal tidak sama dengan huruf terakhir maka menghasilkan False.
4. Recursive case nya adalah memanggil dirinya sendiri dengan kata dimulai dari huruf kedua sampai kedua terakhir. Pemanggilan fungsi dilakukan sampai terpenuhinya kondisi nomor 2 atau 3.

SOAL 3

```python
def deretGanjjil(n,i=1):
    if i<=n:
        return i+deretGanjjil(n,i+2)
    return 0

print(deretGanjjil(100))
```

Output:

```
2500
```

Penjelasan:

1. Terdapat dua kondisi, base case dan recursive case.
2. Recursive case terjadi ketika nilai i kurang dari sama dengan n maka akan mengembalikan nilai i ditambah memanggil dirinya sendiri.
3. Base casenya adalah mengembalikan nilai 0
4. Program berjalan secara iterative (bertambah seiring waktu)

SOAL 4

```python
def sumstringDigit(digitString:str):
    return int(digitString[0])+sumstringDigit(digitString[1:]) if len(digitString)>=1 else 0

digit="09108"
hasil=sumstringDigit(digit)
penjumlahanDigit="+".join(digit)
print(f"{digit} maka jumlah digitnya adalah {penjumlahanDigit} = {hasil}")
```

Output:

```
09108 maka jumlah digitnya adalah 0+9+1+0+8 = 18
```

Penjelasan:

1. Base case terpenuhi ketika nilai panjang string kurang dari 1 maka nilainya 0.

2. Recursive case terpenuhi ketika panjang string lebih dari sama dengan 1 maka akan menjumlahkan nilai string indeks awal yang diubah menjadi int dan memanggil dirinya kembali dengan string mulai dari 1 sampai akhir.

SOAL 5

```
def combinations(n,r):
    if r==0 or n-r==0:
        return 1
    elif r==1 or n-r==1:
        return n
    elif r==2 or n-r==2:
        return n*(n-1)/2
    else:
        return combinations(n-1,r-1)+combinations(n-1,r)

print(combinations(6,3))
```

Output:

```
20.0
```

Penjelasan:

1. Base case:
    a. Menghasilkan 1 jika nilai r sama dengan 0 atau n-r sama dengan 0
    b. Menghasilkan nilai n jika r sama dengan 1 atau n-r sama dengan 1
    c. Menghasilkan nilai n*(n-2)/2 jika r sama dengan 2 atau nilai n-r sama dengan 2
2. Jika recursive case terpenuhi maka maka akan terbentuk dua percabangan yaitu fungsi kombinasi dengan parameter n-1 dan r-1, lalu fungsi kombinasi dengan parameter n-1 dan r.
3. Hasilnya di jumlahkan