Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**<71230989>**
Nama Lengkap	**<Yohanes Nevan Adventus Wibawa>**
Minggu ke / Materi	**13 / 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.

Pengertian Rekrusif

Fungsi rekursif adalah sebuah fungsi yang mengandung dirinya sendiri atau mendefinisikan dirinya sendiri. Fungsi ini sering dikenal sebagai fungsi yang memanggil dirinya sendiri. Fungsi rekursif adalah fungsi matematis yang berulang dengan pola yang teratur, namun penggunaannya harus diawasi agar dapat berhenti dan tidak menyebabkan penggunaan memori yang berlebihan. Penggunaan fungsi rekursif harus dilakukan dengan hati-hati karena dapat menyebabkan loop tak terbatas, yang pada gilirannya bisa menyebabkan program mengalami hang atau tidak merespons.

Fungsi rekursif akan terus berlanjut sampai kondisi penghentian terpenuhi. Oleh karena itu, dalam sebuah fungsi rekursif perlu terdapat dua blok penting: blok yang menentukan kapan proses rekursif berhenti, dan blok yang memanggil fungsi itu sendiri. Dalam rekursif, ada dua bagian utama:

- **Base Case** adalah bagian yang menentukan kapan fungsi rekursif berhenti.
- **Recursive Case** adalah bagian yang mengandung pernyataan yang akan terus diulang hingga mencapai Base Case.

Kelebihan dan kekurangan

Beberapa keunggulan dari fungsi rekursif meliputi:

1. Kode program menjadi lebih ringkas.
2. Masalah yang kompleks dapat dipecah menjadi lebih kecil melalui rekursi.

Namun, fungsi rekursif juga memiliki beberapa kelemahan:

1. Memerlukan lebih banyak memori karena setiap kali fungsi memanggil dirinya sendiri, diperlukan ruang memori tambahan.
2. Kurangnya fisiensi dan kecepatan eksekusi.
3. Sulit untuk di-debug dan kadang-kadang sulit dipahami.

Bentuk Umum dan Studi Kasus

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

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

Setiap fungsi rekursif pasti memiliki solusi iteratif. Sebagai contoh, mari kita lihat kasus perhitungan faktorial. Faktorial adalah hasil perkalian dari deret angka 1 x 2 x 3 x ... x n. Algoritma untuk menghitung faktorial dapat dijelaskan sebagai berikut:

1. Input nilai n.
2. Siapkan variabel total untuk menyimpan hasil perkalian faktorial dan set nilai awalnya menjadi 1.
3. Lakukan loop dari i = 1 hingga n, dengan langkah-langkah:
4. total = total * i.
5. Tampilkan hasil total.

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
```

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

print(faktorial(4))
```

Output =

```
24
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Contoh gambar proses penghitungan:

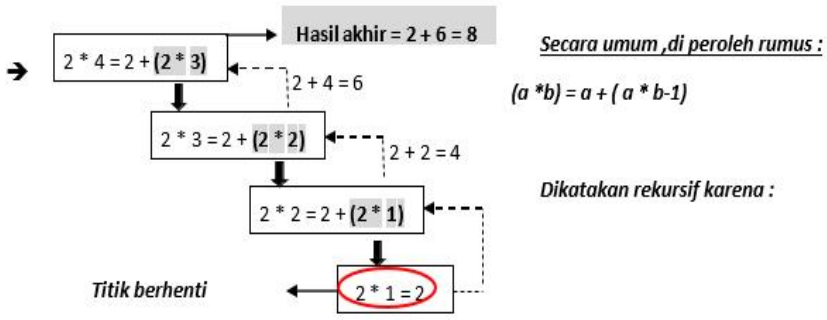
```
calc_factorial(4)                  # 1st call with 4
4 * calc_factorial(3)              # 2nd call with 3
4 * 3 * calc_factorial(2)          # 3rd call with 2
4 * 3 * 2 * calc_factorial(1)      # 4th call with 1
4 * 3 * 2 * 1                      # return from 4th call as number=1
4 * 3 * 2                          # return from 3rd call
4 * 6                              # return from 2nd call
24                                 # return from 1st call
```

Kegiatan Praktikum

Problem dan Solusi 1

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.




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

print(perkalian(2, 4))
```

Output =

```
2 + 2 + 2 + 2 = 8
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Fungsi `perkalian` menggunakan rekursi untuk menghitung perkalian dua bilangan melalui penjumlahan berulang. Jika `bil2` sama dengan 1, fungsi mencetak `bil1 =` dan mengembalikan `bil1`. Jika `bil2` lebih besar dari 1, fungsi mencetak `bil1 +` dan memanggil dirinya sendiri dengan `bil2` dikurangi 1, lalu menambahkan hasilnya dengan `bil1`. Saat memanggil `perkalian(2, 4)`, fungsi ini akan mencetak `2 + 2 + 2 + 2 = 8`, menunjukkan proses penjumlahan berulang untuk menghasilkan hasil akhir.

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 +.

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

print(pangkat(2, 4))
```

Output =

```
2 * 2 * 2 * 2 = 16
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Fungsi `pangkat` menggunakan rekursi untuk menghitung hasil pangkat dari dua bilangan. Jika `bil2` sama dengan 1, fungsi mencetak `bil1 =` dan mengembalikan `bil1`. Jika `bil2` lebih besar dari 1, fungsi mencetak `bil1 *` dan memanggil dirinya sendiri dengan `bil2` dikurangi 1, lalu mengalikan hasilnya dengan `bil1`. Saat `pangkat(2, 4)` dipanggil, fungsi ini mencetak `2 * 2 * 2 * 2 = 16`, menunjukkan proses perkalian berulang untuk menghasilkan hasil akhir.

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

```
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 =

```
1 ,  1 , 2 , 3 , 5 , 8 , 13 ,
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Fungsi fibo menghitung dan mencetak deret Fibonacci hingga elemen ke-n menggunakan pendekatan iteratif. Fungsi dimulai dengan dua bilangan pertama f1 dan f2 yang bernilai 1, kemudian mencetaknya. Dalam loop yang berjalan dari indeks 2 hingga n-1, fungsi menghitung elemen Fibonacci berikutnya sebagai jumlah dari dua elemen sebelumnya, memperbarui f1 dan f2 dengan nilai baru, dan mencetak hasilnya. Saat fibo(7) dipanggil, fungsi ini mencetak tujuh bilangan pertama dari deret Fibonacci: 1, 1, 2, 3, 5, 8, 13.

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

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

print(fibo(7))
```

Output =

```
13
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Fungsi fibo menghitung bilangan Fibonacci ke-n menggunakan rekursi. Base case terjadi saat n sama dengan 1 atau 2, di mana fungsi mengembalikan 1. Untuk nilai n yang lebih besar, fungsi memanggil dirinya sendiri dengan argumen n-1 dan n-2, lalu menjumlahkan hasilnya. Saat fibo(7) dipanggil, fungsi ini akan menghitung bilangan Fibonacci ke-7, yang hasilnya adalah 13.

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.

```python
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))
```

Output =

```
Silahkan masukkan bilangan dan basis
Bilangan : 2
Basis (2/8/16) : 2
10
```

Problem dan Solusi 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.

```python
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-(n-1))! = 1
        return 1
    else:
        # jika n > 0, panggil method permutasi secara rekursif dengan parameter
n-1 sampai n
        return permutasi(m, n-1) * (m-n+1)
        # kembalikan nilai permutasi dengan parameter m dan n-1 dikalikan dengan
(m-n+1)

print(permutasi(10, 4))  # Hasilnya adalah 5040, karena P(10, 4) = 10 * 9 * 8 * 7
= 5040
```

Output =

```
5040
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

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.

SOAL 1

```python
def cek_prima(bilangan, pembagi=None):
    if pembagi is None:
        pembagi = bilangan - 1

    if bilangan <= 1:
        return False
    elif pembagi == 1:
        return True
    elif bilangan % pembagi == 0:
        return False
    else:
        return cek_prima(bilangan, pembagi - 1)

angka = int(input("Masukkan bilangan: "))
if cek_prima(angka):
    print(f"{angka} adalah bilangan prima.")
else:
    print(f"{angka} bukan bilangan prima.")
```

Output =

```
Masukkan bilangan: 2
2 adalah bilangan prima.
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Penjelasan =

Fungsi cek_prima digunakan untuk mengecek apakah suatu bilangan adalah bilangan prima menggunakan rekursi. Jika pembagi tidak diberikan, fungsi menginisialisasinya dengan bilangan - 1. Base case pertama mengembalikan False jika bilangan kurang dari atau sama dengan 1. Base case kedua mengembalikan True jika pembagi sama dengan 1, menandakan bilangan tersebut adalah bilangan prima. Jika bilangan habis dibagi pembagi, fungsi mengembalikan False, menunjukkan bilangan tersebut bukan bilangan prima. Jika tidak, fungsi memanggil dirinya sendiri dengan pembagi - 1. Fungsi ini dipanggil dengan input dari pengguna, dan hasilnya menunjukkan apakah bilangan tersebut adalah bilangan prima atau bukan.

SOAL 2

```python
def cek_palindrom(kalimat):
    kalimat = kalimat.replace(" ", "").lower()
    if len(kalimat) <= 1:
        return True
    elif kalimat[0] == kalimat[-1]:
        return cek_palindrom(kalimat[1:-1])
    else:
        return False

kalimat = input("Masukkan kalimat: ")
if cek_palindrom(kalimat):
    print(f"'{kalimat}' adalah palindrom.")
else:
    print(f"'{kalimat}' bukan palindrom.")
```

Output =

```
Masukkan kalimat: kasur rusak
'kasur rusak' adalah palindrom.
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Penjelasan =

Fungsi cek_palindrom memeriksa apakah suatu kalimat adalah palindrom menggunakan rekursi. Pertama, fungsi menghapus semua spasi dan mengubah semua huruf menjadi huruf kecil untuk konsistensi. Base case terjadi ketika panjang kalimat 0 atau 1, yang dianggap sebagai palindrom. Jika karakter pertama dan terakhir sama, fungsi memanggil dirinya sendiri dengan kalimat yang telah dipotong karakter pertama dan terakhirnya. Jika karakter pertama dan terakhir berbeda, fungsi mengembalikan False, menandakan kalimat tersebut bukan palindrom. Fungsi ini dipanggil dengan input dari pengguna, dan hasilnya menunjukkan apakah kalimat tersebut adalah palindrom atau bukan.

SOAL 3

```python
def deret_ganjil(n):
    if n == 1:
        return 1
    else:
        return n + deret_ganjil(n - 2)

def jumlah_deret_ganjil(n):
    if n <= 0:
        return 0
    else:
        return deret_ganjil(n) + jumlah_deret_ganjil(n - 2)
n = 7
print(f"Jumlah deret ganjil dari 1 hingga {n} adalah {jumlah_deret_ganjil(n)}")
```

Output =

```
Jumlah deret ganjil dari 1 hingga 7 adalah 30
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Penjelasan =

Fungsi rekursif yang menghitung jumlah deret ganjil berdasarkan pola tertentu terdiri dari dua fungsi: `deret_ganjil` dan `jumlah_deret_ganjil`. Fungsi `deret_ganjil` bertugas untuk menghitung nilai ganjil ke-n dalam deret, dengan basis rekursifnya adalah ketika n sama dengan 1, mengembalikan 1. Jika n lebih dari 1, fungsi ini menambahkan n dengan nilai ganjil dari deret sebelumnya, yaitu (n-2). Fungsi `jumlah_deret_ganjil` digunakan untuk menghitung jumlah total dari deret ganjil hingga n.

Rekursifnya adalah ketika n kurang dari atau sama dengan 0, mengembalikan 0. Jika n lebih dari 0, fungsi ini menambahkan n dengan hasil rekursif dari nilai sebelumnya dalam deret (n-2). Kode ini secara efektif menjumlahkan nilai-nilai ganjil dalam deret, seperti 1, 3, 5, 7, dan seterusnya hingga mencapai n, dan mengembalikan jumlah totalnya. Misalnya, jika n adalah 7, fungsi akan menjumlahkan deret ganjil dari 1 hingga 7, menghasilkan hasil akhir dari penjumlahan tersebut.

SOAL 4

```python
def jumlah_digit(bilangan):
    if bilangan < 10:
        return bilangan, str(bilangan)
    else:
        hasil, representasi = jumlah_digit(bilangan // 10)
        sisa = bilangan % 10
        return sisa + hasil, representasi + " + " + str(sisa)

def print_jumlah_digit(bilangan):
    hasil, representasi = jumlah_digit(bilangan)
    print(f"{representasi} = {hasil}")

print_jumlah_digit(234)
```

Output =

```
2 + 3 + 4 = 9
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Penjelasan =

Kode ini terdiri dari dua fungsi: `jumlah_digit` dan `print_jumlah_digit`. Fungsi `jumlah_digit` menghitung jumlah digit bilangan secara rekursif dan membuat string representasi dari proses penjumlahan. Jika bilangan kurang dari 10, fungsi mengembalikan bilangan dan stringnya. Jika lebih dari atau sama dengan 10, fungsi memanggil dirinya sendiri dengan bilangan dibagi 10, menambahkan digit terakhir ke hasil, dan memperbarui string representasinya.

Fungsi `print_jumlah_digit` memanggil `jumlah_digit`, kemudian mencetak string representasi penjumlahan diikuti dengan tanda sama dengan (=) dan hasil akhirnya. Misalnya, saat `print_jumlah_digit(234)` dijalankan, hasilnya akan mencetak "2 + 3 + 4 = 9".

SOAL 5

```python
def kombinasi(n, k):
    if k == 0 or k == n:
        return 1
    else:
        return kombinasi(n-1, k-1) + kombinasi(n-1, k)

n = 5
k = 2
print(f"C({n}, {k}) = {kombinasi(n, k)}")
```

Output =

```
C(5, 2) = 10
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 13>
```

Penjelasan =

Fungsi kombinasi(n, k) digunakan untuk menghitung kombinasi, yaitu jumlah cara memilih k objek dari n objek tanpa memperhatikan urutan. Basis rekursif fungsi ini adalah jika k sama dengan 0 atau k sama dengan n, maka hasilnya adalah 1, karena ada tepat satu cara untuk memilih semua atau tidak memilih sama sekali dari n objek. Untuk kasus rekursif, fungsi menggunakan sifat kombinasi: $C(n, k) = C(n-1, k-1) + C(n-1, k)$ C(n,k)=C(n−1,k−1)+C(n−1,k).

Ini berarti bahwa untuk menghitung jumlah cara memilih k objek dari n, kita bisa menjumlahkan jumlah cara memilih k-1 objek dari n-1 dan jumlah cara memilih k objek dari n-1. Contoh penggunaan fungsi ini diberikan dengan menghitung kombinasi dari 5 objek yang dipilih 2, yang ditampilkan sebagai C(5, 2).

Link Github

Link Github = https://github.com/YohanesNevan/Tugas-Laporan-Alpro-13.git