Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230976**
Nama Lengkap	**Galih Pramana Chandra Prasetya**
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%)

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 Rekursif

Fungsi rekursif adalah fungsi yang "memanggil diri sendiri". Bayangkan sebuah kotak yang di dalamnya terdapat cermin. Jika kita melihat ke cermin, kita akan melihat diri kita sendiri di dalam cermin, begitu seterusnya. Hal ini mirip dengan fungsi rekursif, di mana fungsi tersebut memanggil dirinya sendiri untuk menyelesaikan suatu masalah.

Fungsi rekursif memiliki pola yang terstruktur dan berulang. Fungsi ini harus memiliki kondisi berhenti agar tidak terjadi **loop tak terbatas** yang dapat menyebabkan program berhenti bekerja (hang up).

Oleh karena itu, penggunaan fungsi rekursif perlu dilakukan dengan hati-hati. Pastikan fungsi tersebut memiliki kondisi berhenti yang jelas agar tidak terjadi masalah pada program.

Berikut beberapa poin penting tentang fungsi rekursif:

- **Memanggil diri sendiri:** Fungsi rekursif memanggil dirinya sendiri untuk menyelesaikan suatu masalah.
- **Pola terstruktur:** Fungsi rekursif memiliki pola yang terstruktur dan berulang.
- **Kondisi berhenti:** Fungsi rekursif harus memiliki kondisi berhenti agar tidak terjadi loop tak terbatas.
- **Penggunaan hati-hati:** Fungsi rekursif perlu digunakan dengan hati-hati agar tidak terjadi masalah pada program.

Contoh penggunaan fungsi rekursif:

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

print(faktorial(5))  # Output: 120
```

Bayangkan sebuah fungsi yang terus berjalan sampai mencapai kondisi tertentu. Nah, fungsi rekursif ini punya dua bagian penting untuk memastikan prosesnya berhenti:

1. **Blok Titik Berhenti:** Bagian ini menentukan kapan fungsi rekursif harus berhenti.
2. **Blok Panggil Diri Sendiri:** Bagian ini berisi instruksi yang akan diulang terus menerus hingga mencapai "titik berhenti".

Secara singkat, rekursif terdiri dari dua elemen:

- **Base Case:** Bagian ini menandakan kapan fungsi rekursif harus berhenti dan tidak perlu diulang lagi.
- **Recursive Case:** Bagian ini berisi instruksi yang akan diulang terus menerus sampai mencapai "base case".

```python
def faktorial(n):
  if n == 0:  # Base case: Jika n = 0, maka faktorialnya 1
    return 1
  else:
    return n * faktorial(n - 1)  # Recursive case: Panggil diri sendiri dengan n - 1
```

Dengan kata lain, fungsi rekursif "memanggil dirinya sendiri" untuk menyelesaikan suatu masalah, tapi dengan kondisi yang semakin kecil atau sederhana, hingga akhirnya mencapai "base case" dan prosesnya berhenti.

Kelebihan dan Kekurangan

Kelebihan:

1. **Kode lebih singkat dan elegan:** Fungsi rekursif memungkinkan kita menulis kode yang lebih ringkas dan mudah dipahami. Hal ini karena struktur berulang di dalam fungsi dapat diwakili dengan cara yang lebih sederhana.
2. **Memecah masalah kompleks:** Fungsi rekursif membantu kita memecah masalah yang kompleks menjadi sub-masalah yang lebih kecil dan lebih mudah diselesaikan. Hal ini dapat membuat kode lebih terstruktur dan mudah dipelihara.

Kekurangan:

1. **Konsumsi memori:** Fungsi rekursif membutuhkan memori yang lebih besar karena setiap kali fungsi dipanggil, ruang memori baru diperlukan untuk menyimpan data dan instruksi. Hal ini dapat menjadi masalah, terutama untuk program dengan memori terbatas.
2. **Efisiensi dan kecepatan:** Fungsi rekursif bisa lebih lambat dibandingkan dengan solusi iteratif untuk masalah yang sama. Hal ini karena fungsi rekursif melibatkan pemanggilan fungsi berulang, yang dapat memakan waktu dan sumber daya.
3. **Debugging dan pemahaman:** Fungsi rekursif bisa lebih sulit untuk di-debug dan dipahami dibandingkan dengan solusi iteratif. Hal ini karena struktur rekursif dapat membuat alur program lebih kompleks dan sulit untuk dilacak.

Berikut ini contoh perbandingan code yang menggunakan fungsi rekursif dan code yang tidak menggunakan fungsi rekursif,

Kode Rekursif:

```
def faktorial_iteratif(n):
    hasil = 1
    for i in range(1, n + 1):
        hasil *= i
    return hasil

print(faktorial_iteratif(5))  # Output: 120
```

Kelebihan:

- Kode lebih singkat dan mudah dipahami.
- Struktur berulang diwakili dengan cara yang lebih sederhana.

Kekurangan:

- Membutuhkan memori yang lebih besar.
- Kurang efisien dan lebih lambat.
- Sulit untuk di-debug dan dipahami.

Kode Iteratif:

```
def faktorial_iteratif(n):
    hasil = 1
    for i in range(1, n + 1):
        hasil *= i
    return hasil

print(faktorial_iteratif(5))  # Output: 120
```

Kelebihan:

- Lebih hemat memori.

- Lebih efisien dan lebih cepat.
- Lebih mudah untuk debugging

Kekurangan:

- Kode lebih panjang dan tidak elegan seperti kode rekursif.
- Struktur berulang diwakili dengan cara yang lebih detail.

Pilihan antara kode rekursif dan iteratif tergantung pada situasi dan jenis masalah yang ingin dipecahkan.

- Gunakan kode rekursif jika kode menjadi lebih singkat dan mudah dipahami, dan prioritas utama adalah kesederhanaan kode.
- Gunakan kode iteratif jika masalah kompleks dapat dipecahkan dengan lebih efisien, dan prioritas utama adalah performa dan kemudahan debugging.

Bentuk Umum dan Studi Kasus

Fungsi rekursif di Python umumnya memiliki struktur berikut:

```
def nama_fungsi(parameter):
  # Kondisi berhenti (base case)
  if kondisi_berhenti:
    return nilai_kembalian

  # Panggilan rekursif
  else:
    return nama_fungsi(parameter_baru)
```

Penjelasan:

- **Nama fungsi:** adalah nama yang diberikan untuk fungsi rekursif.
- **Parameter:** Merupakan variabel yang nilainya dapat diubah di dalam fungsi.
- **Kondisi berhenti:** Merupakan kondisi yang menentukan kapan fungsi rekursif harus berhenti dan tidak perlu diulang lagi.
- **Nilai kembalian:** Merupakan nilai yang dikembalikan oleh fungsi rekursif.
- **Panggilan rekursif:** Merupakan bagian di mana fungsi rekursif memanggil dirinya sendiri dengan parameter baru. Parameter baru ini biasanya merupakan hasil dari proses rekursif sebelumnya.

Contoh kasus fungsi rekursif:

1. Menghitung faktorial suatu bilangan.

Faktorial dihitung dengan mengalikan bilangan itu dengan semua bilangan bulat positif di bawahnya.
Contohnya, faktorial 5 adalah 5 x 4 x 3 x 2 x 1 = 120.

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

print(faktorial(5))  # Output: 120
```

2. Membalikkan Kalimat

Fungsi rekursif untuk membalikkan urutan kata dalam sebuah kalimat.

```python
def balik_kalimat(kalimat):
  if not kalimat or len(kalimat.split()) == 1:
    return kalimat
  else:
    kata_pertama = kalimat.split()[0]
    kalimat_sisa = " ".join(kalimat.split()[1:])

    kalimat_dibalik = balik_kalimat(kalimat_sisa)


    return kalimat_dibalik + " " + kata_pertama


kalimat = "Ini adalah kalimat contoh"
kalimat_dibalik = balik_kalimat(kalimat)


print(f"Kalimat Asli: {kalimat}")
print(f"Kalimat Dibalik: {kalimat_dibalik}")
```

Output:

```
Kalimat Asli: Ini adalah kalimat contoh
Kalimat Dibalik: contoh kalimat adalah Ini
```

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

Sc:

```
def cekprima(a, b = None):
    if a <= 1:
        print("Bukan bilangan prima")
        return False
    if b is None:
        b = a - 1
    if b < 2:
        print("Ini bilangan prima")
        return True
    if a % b == 0:
        print("Bukan bilangan prima")
        return False
    return cekprima(a, b - 1)

a = int(input("Masukkan angka: "))
cekprima(a)
```

output:

```
PS D:\Pra Alpro sem 2\pra al
Masukkan angka: 11
Ini bilangan prima
PS D:\Pra Alpro sem 2\pra al
Masukkan angka: 12
Bukan bilangan prima
```

Penjelasan:

- Program meminta pengguna untuk memasukkan sebuah bilangan (a)
- Jika ‘a’ kurang dari atau sama dengan 1, outputnya adalah "Bukan bilangan prima" dan ‘False’.
- Jika b tidak diberikan (bernilai None), maka b diinisialisasi dengan a - 1.

- Jika b kurang dari 2, outputnya adalah "Ini bilangan prima" dan True.
- Jika a habis dibagi oleh b (a % b == 0), outputnya adalah "Bukan bilangan prima" dan False.
- fungsi memanggil dirinya sendiri dengan b dikurangi 1 (cekprima(a, b - 1)), dan proses ini diulangi sampai semua kemungkinan pembagi diperiksa.
- Berdasarkan hasil pemeriksaan, fungsi mencetak apakah bilangan tersebut adalah bilangan prima atau bukan.

SOAL 2

Sc:

```
def hanya_huruf(s):
    return ''.join(char.lower() for char in s if char.isalpha())

def palindrom(s):
    if len(s) < 1:
        return True
    else:
        if s[0] == s[-1]:
            return palindrom(s[1:-1])
        else:
            return False

s = input("Masukkan kata: ")
s = hanya_huruf(s)
if palindrom(s):
    print("Palindrom")
else:
    print("Bukan Palindrom")
```

Output:

```
Masukkan kata: kasur rusak
Palindrom
```

Penjelasan:

- **Fungsi 'hanya_huruf**:' Membersihkan string dari karakter non-alfabet dan mengubahnya menjadi huruf kecil.
- **Fungsi palindrom**: Memeriksa secara rekursif apakah string adalah palindrom.
- **Input dan Output**: Mengambil input dari pengguna, memprosesnya untuk hanya menyisakan huruf, dan mengecek apakah hasilnya adalah palindrom, lalu mencetak hasilnya

SOAL 3

Sc:

```python
n1 = 1
n2 = int(input("Masukkan limit bilangan (n): "))

def jumlah_ganjil(n1, n2):
    if n1 > n2:
        return 0
    return n1 + jumlah_ganjil(n1 + 2, n2)

print("Jumlah bilangan ganjil dalam rentang:", jumlah_ganjil(n1, n2))
```

Output:

```
Masukkan limit bilangan (n): 15
Jumlah bilangan ganjil dalam rentang: 64
```

Penjelasan:

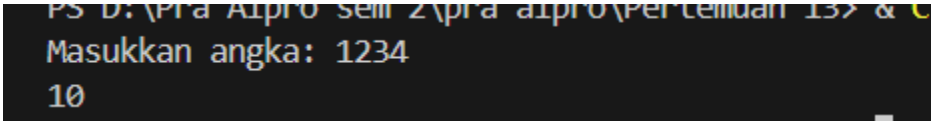
- **Base case:** output 0 jika n1 lebih besar dari n2.
- **Rekursi:** Menambahkan n1 ke hasil penjumlahan rekursif dari n1 + 2 hingga mencapai n2.

SOAL 4

```python
def jumlahangka(n):
    if len(n) == 0:
        return 0
    else:
        o = int(n[-1])
        n = n[:-1]
        return o + jumlahangka(n)

n = input("Masukkan angka: ")
print(jumlahangka(n))
```

Output:



Penjelasan:

- **Base case**: output 0 jika string n kosong.
- **Rekursi**: Menambahkan digit terakhir dari ‘n’ ke hasil penjumlahan rekursif dari sisa string ‘n’.

SOAL 5

```python
def kombinasi(r, n):
    if n == 0:
        return 1
    else:
        int((r-n+1)/ n)
        return (kombinasi(r,n-1) * (r-n+1) / (n))

r = int((input("Masukkan r :")))
n = int((input("Masukkan n :")))

print(kombinasi(r, n))
```

Output:

```
PS D:\Pra Alpro sem 2\p
Masukkan r :10
Masukkan n :5
252.0
```

Penjelasan:

- **Base case**
  Jika n adalah 0, hasilnya adalaj 1 karena kombinasi apapun dengan 0 elemen adalah 1.
- **Rekursi:**

  Menghitung kombinasi dengan mengurangi n satu per satu hingga mencapai 0, sementara menghitung hasil kombinasi berdasarkan formula rekursif:

$$\binom{r}{n} = \frac{(r-n+1) \times \binom{r}{n-1}}{n}$$

Link github: https://github.com/GalihPramana/Praktikum-Alpro-71230976.git