LAPORAN RESMI
MODUL IV
POLA DAN STRUKTUR PERULANGAN
ALGORITMA DAN DASAR PEMROGRAMAN

NAMA	**: MOH. NAFIS MUZADI**
N.R.P	**: 250441100142**
DOSEN	**: SRI HERAWATI, S.Kom., M.Kom.**
ASISTEN	**: OKTAVIA PUTRI ROICHATUL JANNAH**
TGL PRAKTIKUM	**: 23 OKTOBER 2025**

Disetujui : 27 OKTOBER 2025
Asisten

OKTAVIA PUTRI ROICHATUL J.
24.04.411.00057

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, salah satu konsep dasar yang sangat penting adalah *looping* atau perulangan. Perulangan memungkinkan suatu blok kode dijalankan berulang kali selama kondisi tertentu terpenuhi. Dengan cara ini, programmer tidak perlu menulis kode yang sama berulang-ulang, sehingga program menjadi lebih efisien, ringkas, dan mudah dipahami.

Dalam bahasa pemrograman Python, terdapat dua jenis perulangan yang umum digunakan, yaitu *for* dan *while*. Kedua jenis perulangan ini berfungsi untuk menyelesaikan berbagai permasalahan seperti pengulangan proses, perhitungan matematis, hingga pembentukan pola tertentu. Misalnya, perulangan sering digunakan untuk menampilkan deret angka, membuat pola bintang, menghitung bilangan prima, atau menghasilkan urutan angka *Fibonacci*.

Selain itu, perulangan juga dapat dikombinasikan menjadi perulangan bersarang (*nested loop*) untuk membentuk struktur data atau pola yang lebih kompleks. Contohnya adalah dalam pembuatan piramida angka, pola biner, atau simulasi sederhana yang membutuhkan logika berulang, seperti sistem kasir dan pengecekan data pengguna.

Melalui pemahaman konsep perulangan, mahasiswa dapat belajar bagaimana logika pemrograman bekerja secara sistematis. Praktikum mengenai pola dan struktur perulangan ini bertujuan agar mahasiswa mampu memahami, menerapkan, serta mengembangkan berbagai bentuk algoritma berbasis perulangan. Dengan demikian, mahasiswa dapat menyusun solusi yang efisien dan terstruktur dalam menghadapi permasalahan pemrograman di dunia nyata.

1.2 TUJUAN

- Mahasiswa mampu memahami bagaimana pola dan struktur dibangun menggunakan perulangan dalam Python.
- Mahasiswa dapat menyajikan contoh penggunaan perulangan *for* dan *while* untuk Menyusun kombinasi, perhitungan matematis, dan pola grafis.

BAB II
DASAR TEORI

2.1 Perulangan Bersarang (*Nested Loops*)

Nested loop adalah perulangan di dalam perulangan lainnya. Ini sering digunakan saat kita ingin mengulangi beberapa operasi untuk setiap elemen dari loop luar. Misalnya, dalam pembentukan pola dua dimensi seperti matriks atau tabel, perulangan bersarang menjadi sangat penting untuk mengakses elemen dalam dua dimensi tersebut.

Contoh :

```
# Loop luar (outer loop)
for i in range(1, 4): # Loop ini akan berjalan 3 kali(i = 1, 2, 3)
    print(f"Loop luar i = {i}")

    #Loop dalam (inner loop)
    for j in range(1, 4): # Loop ini juga akan berjalan 3 kali
untuk setiap iterasi dari loop luar
        print(f" Loop dalam j = {j}")
```

Penjelasan :

1 Perulangan i pertama (i = 1) :

Mencetak “Loop luar i = 1 ”

a. Perulangan j pertama (j = 1)

Mencetak “Loop dalam j = 1 ”

b . Perulangan j kedua (j = 2)

Mencetak “Loop dalam j = 2 ”

c. Perulangan j ketiga (j = 3)

Mencetak “Loop dalam j = 3 ”

2 Perulangan i pertama (i = 2) :

Mencetak “Loop luar i = 2 ”

a. Perulangan j pertama (j = 1)

Mencetak “Loop dalam j = 1 ”

b. Perulangan j kedua (j = 2)

Mencetak “Loop dalam j = 2 ”

c. Perulangan j ketiga (j = 3)

Mencetak “Loop dalam j = 3 ”

3 Perulangan i pertama (i = 3) :

Mencetak “Loop luar i = 3 ”

a. Perulangan j pertama (j = 1)

Mencetak “Loop dalam j = 1 ”

b. Perulangan j kedua (j = 2)

Mencetak “Loop dalam j = 2 ”

c. Perulangan j ketiga (j = 3)

Mencetak “Loop dalam j = 3 ”

Output :

```
Loop luar i = 1
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
Loop luar i = 2
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
Loop luar i = 3
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
```

2.2 Struktur Perulangan dan Kombinasi

Struktur berulang seperti deret biner, angka Fibonacci, atau perhitungan FPB dapat diprogram seara efisien menggunakan perulangan. Setiap struktur ini memiliki pola pengulangan yang khas, yang dapat dioptimalkan dengan menggunakan teknik pengulangan yang tepat.

Contoh :

```python
while b != 0:
    a, b = b, a % b

print(f"FPB-nya adalah: {a}")
```

Penjelasan :

a = 24 , b = 3

1. Perulangan pertama:
   a. Kondisi b != 0 adalah benar (b = 36).
   b. Nilai a diperbarui menjadi 36 (nilai b).
   c. Nilai b diperbarui menjadi 24 % 36, yang menghasilkan 24.
   d. Setelah perulangan: a = 36, b = 24. 2.
2. Perulangan kedua:
   a. Kondisi b != 0 masih benar (b = 24).
   b. Nilai a diperbarui menjadi 24 (nilai b).
   c. Nilai b diperbarui menjadi 36 % 24, yang menghasilkan 12.
   d. Setelah perulangan: a = 24, b = 12. 3.
3. Perulangan ketiga:
   a. Kondisi b != 0 masih benar (b = 12).
   b. Nilai a diperbarui menjadi 12 (nilai b).
   c. Nilai b diperbarui menjadi 24 % 12, yang menghasilkan 0.
   d. Setelah perulangan: a = 12, b = 0. 4.
4. Perulangan selesai:
   a. Kondisi b != 0 tidak lagi benar (b = 0), sehingga perulangan berhenti.
5. Hasil:
   a. Nilai a yang tersisa adalah 12, yang merupakan FPB dari 24 dan 36.

Setiap iterasi perulangan, nilai b akan di-update dengan sisa bagi (a % b), dan iterasi terus berlanjut hingga b menjadi nol. Nilai a terakhir saat b = 0 adalah FPB-nya.

Output :

```
FPB-nya adalah: 12
```

2.3 Pola Matematika dalam Perulangan

Banyak pola matematika, seperti deret Fibonacci, perhitungan FPB, dan lainnya dapat diselesaikan menggunakan pendekatan berbasis loop. Misalnya, urutan bilangan Fibonacci dapat dihitung dengan memperbarui nilai dua bilangan sebelumnya pada setiap iterasi, sedangkan FPB (Faktor Persekutuan Terbesar) dapat ditemukan menggunakan algoritma Euclidean berbasis modulus, yang terus

menerus mengulang hingga mencapai hasil yang diinginkan.

Contoh Deret Fibonacci :

```
n = 100 #batas angka
a, b = 0, 1

print("Bilangan Fibonacci hingga", n)
while a <= n:
    print(a, end=" ")
    a, b = b, a + b
```

Penjelasan :

1. Inisialisasi:
   a. n diatur ke 100, yang merupakan batas atas untuk bilangan Fibonacci yang akan dicetak.
   b. a diinisialisasi dengan 0 (bilangan Fibonacci pertama) dan b diinisialisasi dengan 1 (bilangan Fibonacci kedua).'

2. Perulangan pertama:
   a. Kondisi a <= n adalah benar (a = 0 ≤ 100).
   b. Bilangan a (0) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (1), dan b diupdate menjadi hasil penjumlahan a + b (0 + 1), sehingga setelah perulangan: a = 1, b = 1.

3. Perulangan kedua:
   a. Kondisi a <= n masih benar (a = 1 ≤ 100).
   b. Bilangan a (1) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (1), dan b diupdate menjadi hasil penjumlahan a + b (1 + 1), sehingga setelah perulangan: a = 1, b = 2.

4. Perulangan ketiga:
   a. Kondisi a <= n masih benar (a = 1 ≤ 100).
   b. Bilangan a (1) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (2), dan b diupdate menjadi hasil penjumlahan a + b (1 + 2), sehingga setelah perulangan: a = 2, b = 3.

5. Perulangan keempat:
   a. Kondisi a <= n masih benar (a = 2 ≤ 100).
   b. Bilangan a (2) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (3), dan b diupdate menjadi hasil penjumlahan a + b (2 + 3), sehingga setelah perulangan: a = 3, b = 5.

6. Proses berlanjut:
   a. Proses ini berulang, di mana setiap iterasi mencetak nilai a dan memperbarui nilai a dan b hingga a menjadi lebih besar dari 100. Beberapa nilai yang dicetak antara lain 3, 5, 8, 13, 21, 34, 55, 89, dan terakhir 144, di mana pada iterasi ini kondisi a <= n menjadi salah.

7. Akhir perulangan:
   a. Ketika a mencapai 144, kondisi a <= n tidak lagi benar (144 > 100), sehingga perulangan berhenti.

Output :

```
Bilangan Fibonacci hingga 100
0 1 1 2 3 5 8 13 21 34 55 89
```

2.4 Pola Grafis dalam Perulangan

Dengan memanfaatkan nested loops, kita bisa mencetak pola visual seperti piramida bintang atau angka. Struktur seperti ini membantu kita dalam memahami logika kontrol aliran program, serta memberikan pemahaman tentang bagaimana sebuah program dapat digunakan untuk menyusun representasi visual atau grafis.

Contoh :

```
n = 5

for i in range(1, n + 1):
    for j in range(n - i):
        print(' ', end=' ')

    for k in range(1, i + 1):
        print(k, end=' ')
    print()
```

Penjelasan :

1. Perulangan i Pertama (i = 1):

a. Perulangan j (j = 0 hingga 3) :

Karena n - i = 5 - 1 = 4, maka perulangan j berjalan 4 kali dan mencetak 4 spasi. Ini menggeser angka yang akan dicetak pada baris ini.

b. Perulangan k (k = 1 hingga 1) :

Perulangan k berjalan 1 kali dan mencetak angka 1.

c. Setelah kedua perulangan selesai, program pindah ke baris baru.

2. Perulangan i Kedua (i = 2):

a. Perulangan j (j = 0 hingga 2) :

Karena n - i = 5 - 2 = 3, maka perulangan j berjalan 3 kali dan mencetak 3 spasi untuk menggeser angka.

b. Perulangan k (k = 1 hingga 2) :

Perulangan k berjalan 2 kali, mencetak angka 1 2.

c. Setelah selesai, pindah ke baris baru.

3. Perulangan i Ketiga (i = 3):

a. Perulangan j (j = 0 hingga 1) :

Karena n - i = 5 - 3 = 2, maka perulangan j berjalan 2 kali, mencetak 2 spasi.

b. Perulangan k (k = 1 hingga 3) :

Perulangan k berjalan 3 kali, mencetak angka 1 2 3.

c. Pindah ke baris berikutnya.

4. Perulangan i Keempat (i = 4):

a. Perulangan j (j = 0 hingga 0) :

Karena n - i = 5 - 4 = 1, maka perulangan j berjalan 1 kali, mencetak 1 spasi.

b. Perulangan k (k = 1 hingga 4) :

Perulangan k berjalan 4 kali, mencetak angka 1 2 3 4.

c. Pindah ke baris berikutnya.

5. Perulangan i Kelima (i = 5):

a. Perulangan j (tidak berjalan karena n - i = 0) :

Tidak ada spasi yang dicetak, karena n - i = 0.

b. Perulangan k (k = 1 hingga 5) :

Perulangan k berjalan 5 kali, mencetak angka 1 2 3 4 5.

c. Setelah selesai, program tidak lagi pindah baris karena ini adalah perulangan terakhir.

Output :

```
    1
  1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

BAB III
TUGAS PENDAHULUAN

27/10 2025

3.1 Soal

1. Mengapa penggunaan perulangan sangat penting dalam pemrograman? Berikan beberapa contoh keuntungannya!
2. Jelaskan tentang bagaimana perulangan bersarang (nested loop) bekerja, dan berikan contoh implementasinya!
3. Apa perbedaan utama antara perulangan for dan while dalam Python? Jelaskan kapan lebih baik menggunakan for dibandingkan while, dan sebaliknya!
4. Apa potensi masalah yang bisa muncul saat menggunakan nested loops dalam pemrograman? Jelaskan disertai contohnya!
5. Apa yang dimaksud dengan deret Fibonacci? Jelaskan cara menghitung angka Fibonacci menggunakan struktur perulangan dan berikan contohnya!
6. Buatlah program dengan studi kasus sebagai berikut:
   Seorang siswa ingin membuat program sederhana untuk menampilkan pola piramida angka secara otomatis menggunakan Python. Program harus meminta pengguna memasukkan jumlah baris pola yang diinginkan. Setiap baris menampilkan deretan angka naik secara berurutan sehingga membentuk pola seperti piramida.
   Dengan ketentuan program sebagai berikut:
   - Gunakan struktur nested loop (perulangan bersarang).
   - Pola harus menampilkan angka dari 1 hingga baris ke-n.
   - Program harus tetap rapi, dengan jarak spasi yang sesuai agar berbentuk segitiga.

3.2 Jawaban

1. Perulangan sangat penting dalam pemrograman karena memungkinkan kita untuk menjalankan suatu blok kode secara berulang-ulang tanpa harus menulis kode yang sama berkali-kali.

```
contoh:
for i in range(1,3):
    print ("angka ke-", i)
output
angka ke-1
angka ke-2
angka ke-3
contoh lain:
i=1
while i<=3:
    print ("halo kak okka")
    i+=1
output:
halo kak okka
halo kak okka
halo kak okka
```

2. Nested loop adalah perulangan didalam perulangan lain. cara kerjanya, setiap kali perulangan luar berjalan satu kali, perulangan dalam akan berjalan penuh dari awal sampai akhir.

```
contohnya:
for i in range (2):
    for j in range (2):
        print (i,j)
output
00
01
10
11
```

3. Perulangan for digunakan untuk mengulang sesuatu dengan jumlah yang sudah diketahui, sedangkan while digunakan untuk mengulang selama suatu kondisi masih benar. Gunakan for jika jumlah perulangan sudah diketahui. Gunakan while jika jumlah perulangan tidak pasti dan tergantung pada kondisi tertentu yang bisa berubah saat program berjalan.

4. Masalah utama dari nested loops adalah program bisa melambat karena jumlah perulangan jadi banyak, dan kadang bisa bingung dibaca atau mudah salah jika loop terlalu banyak atau pengaturan indeksnya kurang tepat.

Contoh:

```
angka = [1,2,3]
huruf = ["a","b","c"]
for a in angka:
    for h in huruf:
        print(a,h)
```

OUTPUT:

```
1 a
1 a
1 a
2 b
2 b
2 b
3 c
3 c
3 c
```

27/10 2025

5. Deret Fibonacci adalah barisan bilangan yang dimulai dengan 0 dan 1, dimana setiap angka berikutnya dalam barisan tersebut diperoleh dari penjumlahan dua angka sebelumnya secara berurut-turut. Cara menghitungnya adalah dengan cara menempatkan dua angka awal yaitu 0 dan 1, lalu dalam setiap perulangan hitung angka baru dengan menjumlahkan keduanya, setelah itu, geser nilainya dan ulangi prosesnya sampai jumlah angka yang diinginkan tercapai.

CONTOH:

```
n = int(input("Masukkan jumlah angka Fibonacci : "))#misal 9
a, b = 0, 1

for i in range(n):
    print(a, end=" ")
    a, b = b, a+b
```

OUTPUT:

```
0 1 1 2 3 5 8 13 21
```

```
n = int(input("Masukkan jumlah barisan piramida : "))#misal 5

for i in range(1, n+1):
    print(" "*(n-i), end="")
    for j in range(1, i+1):
        print(j, end=" ")

    print()
```

OUTPUT:

```
Masukkan jumlah barisan piramida : 5
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

BAB IV
IMPLEMENTASI

4.1 Tugas Praktikum

4.1.1 Tugas Praktikum Soal No. 1

Mas rusdi kini mengawasi beberapa baris lampu di taman kota, dan setiap baris memiliki jumlah lampu yang berbeda. Untuk membantu temannya, mas narji ingin membuat program Python yang dapat menampilkan kondisi setiap lampu. Program harus meminta input jumlah baris lampu, lalu jumlah lampu di setiap baris. Setiap lampu diberi nomor urut, dan program menampilkan pesan "Lampu ke-[x] pada baris [y] menyala." Jika nomor lampu merupakan kelipatan 3, tampilkan "Lampu ke-[x] pada baris [y] rusak." Selain itu, jika baris lampu adalah baris terakhir, tambahkan pesan "Periksa sambungan daya utama." Program ini membantu mas rusdi dan temannya mengetahui kondisi setiap lampu dengan cepat dan efisien tanpa harus memeriksa secara manual satu per satu.

4.1.2 Tugas Praktikum Soal No. 2

Pak wowo kini bekerja di perusahaan besar dengan dua shift, pagi dan malam. Ia ingin menghitung total gaji mingguan dengan memperhitungkan lembur dan bonus shift malam. Buatlah program Python yang menggunakan perulangan for selama 7 hari. Program harus meminta input jumlah jam lembur dan apakah hari itu termasuk shift malam (y/n). Setiap hari, gaji pokok Pak wowo adalah Rp100.000, dengan tambahan lembur: 1–3 jam = Rp25.000 per jam, 4 jam = Rp100.000, 6 jam = Rp200.000, 8 jam = Rp300.000. Jika lembur lebih dari 8 jam, tampilkan "Lembur melebihi batas, tidak dihitung." Jika hari tersebut shift malam, tambahkan bonus Rp50.000. Setelah 7 hari, program menampilkan total jam lembur, total bonus shift malam, dan total gaji seminggu. Program ini harus menangani input tidak valid dan tetap berjalan hingga semua data lengkap dimasukkan dengan benar.

4.1.3 Tugas Praktikum Soal No. 3

bjirka senang membuat pola angka untuk menenangkan pikirannya. Kali ini, ia ingin membuat dua piramida angka yang saling berhadapan seperti cermin. Jika pengguna memasukkan angka n = 5. Program harus menggunakan tiga perulangan bersarang (*nested loop*) — satu untuk sisi kiri piramida, satu untuk ruang kosong di

tengah, dan satu untuk sisi kanan piramida. Gunakan logika agar jumlah angka dan spasi berubah sesuai barisnya sehingga pola tetap simetris untuk nilai n berapa pun.

4.2 Source Code

4.2.1 Source Code Soal No. 1

```
n = int(input("Masukkan Jumlah baris Lampu : "))
for i in range(1, n + 1):
    k = int(input(f"Masukkan jumlah lampu pada baris ke - {i} :
"))
    for j in range(1, k + 1):
        if j % 3 != 0:
            print("Lampu ke -", j, "pada baris ke -", i,
"menyala")
        else:
            print("Lampu ke -", j, "pada baris ke -", i,
"rusak")
    if i == n:
        print("Periksa sambungan daya utama")
```

4.2.2 Source Code Soal No. 2

```
gaji = 100000
jam = 0
malam = 0
total_gaji = 0


for i in range(1, 8):
    bonus = 0
    print("Hari ke-", i)
    while True:
        try :
            lembur = int(input("Masukkan jumlah jam lembur (0-
8) : "))
            if lembur < 0:
                print("Input tidak boleh negatif.")
                continue
            elif lembur > 8:
                print("Lembur melebihi batas, tidak
dihitung.")
                lembur = 0
            break
        except ValueError:
            print("Input harus berupa angka. Masukkan kembali
jam lembur anda (0-8)")
    if lembur >= 0 and lembur <= 3:
```

```
        bonus = 25000 * lembur
    elif lembur == 4 or lembur == 5:
        bonus = 100000
    elif lembur == 6 or lembur == 7:
        bonus = 200000
    elif lembur == 8:
        bonus = 300000
    else:
        bonus = 0
    jam += lembur
    shift = input("Apakah shift malam? (y/n) : ").lower()
    while shift not in ["y", "n"]:
        shift = input("Input y untuk ya atau input n untuk
tidak. silahkan masukkan kembali (y/n) : ").lower()
    if shift == "y":
        bonusshift = 50000
        malam += 50000
    else :
        bonusshift = 0
    print("Gaji hari ini: Rp ", gaji + bonus + bonusshift)
    print()

    total_gaji += gaji + bonus + bonusshift

print()
print("Total jam lembur :", jam)
print("Total bonus shift malam: Rp", malam)
print("Total gaji : Rp", total_gaji)
```

4.2.3 Source Code Soal No. 3

```
n = int(input("Masukkan jumlah baris piramida yang diingingkan
: "))

for i in range(1, n + 1):
    for kn in range(1, i + 1):
        print(kn, end=' ')

    for spasi in range(2 * (n - i)):
        print('  ', end='')

    for kr in range(i, 0, -1):
        print(kr, end=' ')

    print()
```

4.3 Hasil

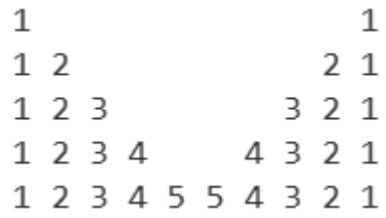
4.3.1 Hasil Soal No. 1

```
Masukkan Jumlah baris Lampu : 2
Masukkan jumlah lampu pada baris ke - 1 : 5
Lampu ke - 1 pada baris ke - 1 menyala
Lampu ke - 2 pada baris ke - 1 menyala
Lampu ke - 3 pada baris ke - 1 rusak
Lampu ke - 4 pada baris ke - 1 menyala
Lampu ke - 5 pada baris ke - 1 menyala
Masukkan jumlah lampu pada baris ke - 2 : 3
Lampu ke - 1 pada baris ke - 2 menyala
Lampu ke - 2 pada baris ke - 2 menyala
Lampu ke - 3 pada baris ke - 2 rusak
Periksa sambungan daya utama
```

4.3.2 Hasil Soal No. 2

```
Hari ke- 1
Masukkan jumlah jam lembur (0-8) : 3
Apakah shift malam? (y/n) : y
Gaji hari ini: Rp  225000

Hari ke- 2
Masukkan jumlah jam lembur (0-8) : 5
Apakah shift malam? (y/n) : n
Gaji hari ini: Rp  200000

Hari ke- 3
Masukkan jumlah jam lembur (0-8) : 8
Apakah shift malam? (y/n) : y
Gaji hari ini: Rp  450000

Hari ke- 4
Masukkan jumlah jam lembur (0-8) : 9
Lembur melebihi batas, tidak dihitung.
Apakah shift malam? (y/n) : n
Gaji hari ini: Rp  100000

Hari ke- 5
Masukkan jumlah jam lembur (0-8) : 1
Apakah shift malam? (y/n) : n
Gaji hari ini: Rp  125000

Hari ke- 6
Masukkan jumlah jam lembur (0-8) : 4
Apakah shift malam? (y/n) : n
Gaji hari ini: Rp  200000

Hari ke- 7
Masukkan jumlah jam lembur (0-8) : 2
Apakah shift malam? (y/n) : y
Gaji hari ini: Rp  200000


Total jam lembur : 23
Total bonus shift malam: Rp 150000
Total gaji : Rp 1500000
```

4.3.3 Hasil Soal No. 3



4.4 Penjelasan

4.4.1 Penjelasan Soal No. 1

Program ini memakai dua perulangan *for*. Perulangan pertama digunakan untuk menentukan jumlah baris lampu, sedangkan perulangan kedua menampilkan kondisi tiap lampu di setiap baris. Jika nomor lampu merupakan kelipatan 3, maka lampu dianggap rusak. Pada baris terakhir, program juga menampilkan pesan "Periksa sambungan daya utama." Program ini membantu menampilkan kondisi lampu secara otomatis tanpa harus memeriksa satu per satu.

4.4.2 Penjelasan Soal No. 2

Program ini menghitung gaji Pak Wowo selama seminggu dengan memperhitungkan lembur dan bonus shift malam. Setiap hari, pengguna memasukkan jam lembur dan status shift malam. Bonus lembur dihitung sesuai jumlah jam, sedangkan shift malam mendapat tambahan Rp50.000. Semua hasil kemudian dijumlahkan untuk menampilkan total lembur, total bonus, dan total gaji seminggu. Program ini mempermudah perhitungan gaji secara otomatis dan cepat.

4.4.3 Penjelasan Soal No. 3

Program ini menggunakan tiga perulangan *for* untuk membuat dua piramida bintang yang saling berhadapan. Perulangan pertama mengatur jumlah baris, perulangan kedua mencetak bintang di sisi kiri, perulangan ketiga mencetak spasi di tengah, lalu perulangan terakhir mencetak bintang di sisi kanan. Hasilnya, terbentuk dua piramida simetris seperti cermin dengan bentuk yang tetap rapi meskipun nilai n diubah.

BAB V
PENUTUP

5.1 Analisa

Pada praktikum ini, setiap soal menggunakan perulangan untuk menyelesaikan masalah berbeda. Pada soal pertama, digunakan dua perulangan *for* untuk menampilkan kondisi lampu di setiap baris. Program juga menambahkan logika kondisi untuk menentukan lampu yang rusak dan memberi peringatan pada baris terakhir. Pada soal kedua, perulangan digunakan untuk menghitung total gaji dalam satu minggu berdasarkan jam lembur dan bonus shift malam. Sedangkan pada soal ketiga, digunakan tiga perulangan *for* bersarang untuk membuat pola piramida ganda yang simetris.

Dari hasil percobaan, terlihat bahwa penggunaan perulangan membuat program menjadi lebih mudah, terstruktur, dan efisien. Mahasiswa juga dapat memahami bagaimana logika kondisi (*if*) dan perulangan bekerja bersama untuk menghasilkan hasil yang sesuai dengan kebutuhan program.

Secara keseluruhan, praktikum ini membantu mahasiswa mengenal cara berpikir logis dan sistematis dalam membuat program. Dengan memahami konsep perulangan, mahasiswa dapat menyusun solusi pemrograman yang lebih praktis, efisien, serta dapat diterapkan pada berbagai permasalahan di kehidupan nyata.

5.2 Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa :

- Perulangan sangat membantu dalam menyelesaikan masalah yang bersifat berulang secara otomatis.
- Struktur *for* dan *while* memiliki fungsi berbeda namun sama-sama penting dalam membangun logika program.
- Penerapan perulangan bersarang (*nested loop*) dapat digunakan untuk membuat pola dan struktur data yang lebih kompleks, seperti pola bintang atau pengecekan kondisi lampu.
- Dengan memanfaatkan perulangan dan logika kondisi, program menjadi lebih efisien, mudah dibaca, serta menghemat waktu penulisan kode.