

LAPORAN PRATIKUM
PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

“PERULANGAN FOR DAN NASTED FOR”

disusun Oleh:

AFIF RAHMAN SALEH

NIM 251153105

Dosen Pengampu: Dr. WAHYUDI, S.T, M.T

Asisten Pratikum: AUFAN TAUFIQURRAHAMAN



DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN

2025

DAFTAR ISI

DAFTAR ISI.....	i
KATA PENGANTAR.....	ii
BAB I.....	1
PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Tujuan	1
1.3 Manfaat	2
BAB II	3
PEMBAHASAN	3
2.1 Penjelasan Awal.....	3
2.2 Kode 1 : PerulanganFor1	3
2.2 Kode 2 : PerulanganFor2	4
2.3 Kode 3 : PerulanganFor3	5
2.4 Kode 4 : PerulanganFor3	7
2.5 Kode 5 : PerulanganFor3	8
2.6 Kode 6 : PerulanganFor3	10
2.7 Kode 7 : PerulanganFor3	11
BAB III.....	13
KESIMPULAN	13
3.1 Kesimpulan	13
3.2 Saran	13
DAFTAR PUSTAKA	14

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai bentuk dokumentasi dan pertanggungjawaban atas kegiatan praktikum mata kuliah Algoritma dan Pemrograman, khususnya pada topik “Perulangan For dan Nested For”.

Dalam penyusunan laporan ini, penulis berusaha untuk menjelaskan secara sistematis mulai dari latar belakang, tujuan, dan manfaat praktikum, hingga pembahasan kode program, analisis hasil, serta kesimpulan dan saran untuk pengembangan lebih lanjut. Materi yang dibahas diharapkan dapat menjadi referensi bagi mahasiswa atau praktikan dalam memahami dan mengimplementasikan algoritma perulangan secara efektif.

Padang, 2025

Penulis

BAB I

PENDAHULUAN

1.1 Latar Belakang

Algoritma dan pemrograman merupakan dasar penting dalam dunia teknologi informasi. Dalam era digital saat ini, kemampuan untuk merancang algoritma yang efisien dan mengimplementasikannya melalui bahasa pemrograman seperti Java sangat dibutuhkan untuk menyelesaikan berbagai masalah komputasi.

Perulangan (loop) adalah salah satu konsep fundamental dalam pemrograman yang memungkinkan eksekusi kode berulang kali, sehingga mengurangi redundansi dan meningkatkan efisiensi program. Bab ini fokus pada perulangan for dan nested for di Java, yang sering digunakan untuk mengelola data dalam bentuk array, matriks, atau pola berulang lainnya. Pemahaman konsep ini penting untuk mahasiswa atau praktikan agar dapat membangun aplikasi yang lebih kompleks dan optimal.

1.2 Tujuan

Tujuan dari praktikum ini adalah untuk:

1. Memahami konsep dasar perulangan for dalam bahasa pemrograman Java.
2. Mempelajari implementasi nested for (perulangan bersarang) untuk menangani struktur data multidimensi.
3. Melatih kemampuan dalam menulis kode Java yang menggunakan perulangan untuk menyelesaikan masalah praktis.
4. Mengembangkan keterampilan debugging dan analisis kode untuk memastikan program berjalan dengan benar.

1.3 Manfaat

Manfaat yang diperoleh dari praktikum ini meliputi:

1. Meningkatkan pemahaman mahasiswa tentang struktur kontrol dalam pemrograman, yang penting untuk pengembangan perangkat lunak.
2. Membantu mahasiswa dalam pembelajaran algoritma yang efisien, sehingga dapat mengoptimalkan waktu eksekusi program.
3. Mendorong kreativitas mahasiswa dalam menyelesaikan masalah melalui pola perulangan, seperti dalam pengolahan data atau pembuatan pola grafis.
4. Mempersiapkan mahasiswa untuk topik pemrograman lanjutan, seperti pengolahan array dan algoritma pencarian/penyortiran.

BAB II PEMBAHASAN

2.1 Penjelasan Awal

Bab ini membahas tujuh contoh kode program yang menggunakan perulangan for dan nested for dalam bahasa pemrograman Java. Setiap kode akan dijelaskan langkah kerjanya secara rinci, diikuti dengan analisis hasil yang diperoleh berdasarkan landasan teori algoritma dan pemrograman. Landasan teori utama meliputi konsep perulangan (loop) sebagai struktur kontrol yang memungkinkan eksekusi berulang berdasarkan kondisi, penggunaan variabel untuk penyimpanan nilai sementara, serta input/output untuk interaksi dengan pengguna.

Perulangan for terdiri dari tiga bagian: inisialisasi, kondisi, dan increment/decrement, yang memastikan loop berjalan secara sistematis tanpa risiko infinite loop jika kondisi tidak terpenuhi. Nested for (perulangan bersarang) digunakan untuk struktur multidimensi, seperti pola atau matriks, di mana loop luar mengontrol dimensi utama dan loop dalam mengatur detail.

2.2 Kode 1 : PerulanganFor1

Kode Program 2.1

```
public static void main(String[] args) {  
    for (int i=1; i≤10; i++) {  
        System.out.println(i);  
    }  
}
```

Penjelasan Langkah Kerja:

1. Mulai program dengan mendeklarasikan kelas.
2. Inisialisasikan variable i dengan nilai 1.
3. Periksa kondisi $i \leq 10$; jika benar, blok kode dijalankan.
4. Di dalam blok, nilai i dicetak ke layar menggunakan `System.out.println(i)`, yang menambahkan baris baru setelah setiap output.
5. Setelah eksekusi blok, i diincrement sebesar 1 (`i++`).
6. Langkah 3-5 diulang hingga $i > 10$, sehingga loop berhenti.

Output:

Output 2.1

```
1
2
3
4
5
6
7
8
9
10
```

Analisis Hasil:

Hasil eksekusi adalah pencetakan angka 1 hingga 10, masing-masing pada baris terpisah. Berdasarkan teori perulangan for, ini menunjukkan pengulangan dengan 10 siklus, yang efisien untuk tugas sederhana seperti penomoran. Output ini sesuai dengan prinsip kontrol alur, dimana setiap pengulangan mengubah nilai i secara bertahap, memastikan tidak ada pengulangan kode.

2.2 Kode 2 : PerulanganFor2

Kode Program 2.2

```
public static void main(String[] args) {  
    for (int i=1; i≤10; i++) {  
        System.out.print(i + " " );  
    }  
  
}
```

Penjelasan Langkah Kerja:

1. Kelas PerulanganFor2 dan metode main dideklarasikan.
2. Loop for dimulai dengan i = 1.
3. Kondisi i <= 10 dievaluasi; jika benar, blok dijalankan.
4. System.out.print(i + " ") mencetak nilai i diikuti spasi, tanpa baris baru.
5. i diincrement, dan loop kembali ke langkah 3 hingga i > 10.

Output :

```
1 2 3 4 5 6 7 8 9 10
```

Analisis Hasil: Output adalah "1 2 3 4 5 6 7 8 9 10" pada satu baris. Perbedaan dengan kode 1 terletak pada penggunaan print alih-alih println, yang mempertahankan output dalam satu baris. Landasan teori menjelaskan bahwa print tidak menambahkan newline, sehingga cocok untuk output horizontal. Ini menggambarkan fleksibilitas I/O dalam Java, di mana pemilihan metode output memengaruhi format hasil, mendukung algoritma yang memerlukan presentasi data dalam bentuk baris tunggal, seperti daftar atau array.

2.3 Kode 3 : PerulanganFor3

Kode Program 2.3


```

public static void main(String[] args) {
    int jumlah = 0;
    for (int i=1; i<=10; i++) {
        System.out.print(i);
        jumlah=jumlah+i ;
        if (i<10) {
            System.out.print( " + ");
        }
    }
    System.out.println();
    System.out.println("jumlah = " +jumlah);
}

```

Penjelasan Langkah Kerja:

1. Inisialisasikan variable jumlah dengan 0.
2. Loop for dimulai dengan i = 1.
3. Periksa Kondisi i <= 10; jika benar, cetak i.
4. Perbarui jumlah dengan menambahkan i.
5. Jika i < 10, cetak " + "; jika tidak, lanjut.
6. Increment i, dan loop berulang hingga i > 10.
7. Setelah loop, cetak baris baru, diikuti dengan "jumlah = " dan nilai jumlah.

Output :

Output2.3

```

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
jumlah = 55

```

Analisis Hasil:

Output menampilkan "1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10" dan "jumlah = 55". Ini adalah implementasi rumus penjumlahan aritmatika (1 sampai n), di mana $n=10$. Landasan teori akumulasi dalam loop menunjukkan efisiensi untuk menghitung total tanpa kode berulang. Struktur kondisional if memastikan format ekspresi matematis yang benar, menggambarkan integrasi kontrol alur dengan operasi aritmatika, yang berguna dalam algoritma komputasi numerik.

2.4 Kode 4 : PerulanganFor3

Kode Program :

Kode Program 2.4

```
public static void main(String[] args) {  
  
    int jumlah=0;  
    int batas;  
    Scanner input = new Scanner (System.in);  
    System.out.print( "masukkan nilaia batas = ");  
    batas = input.nextInt();  
    input.close();  
    for (int i= 1; i≤batas ; i++) {  
        System.out.print( i);  
        jumlah= jumlah +i;  
        if (i<batas ) {  
            System.out.print(" + ");  
        }else {  
            System.out.print(" = ");  
        }  
    }  
}
```

Penjelasan Langkah Kerja:

1. Import Scanner untuk input.
2. Inisialisasikan jumlah = 0, dan deklarasi batas.
3. Buat scanner untuk meminta pengguna memasukkan batas.
4. batas dibaca dari input, dan tutup scanner.
5. Loop for dimulai dengan $i = 1$ hingga $i \leq \text{batas}$.
6. Di dalam loop cetak i dan akumulasi jumlah.
7. Kondisi if menentukan apakah cetak " + " atau " = " berdasarkan i relatif terhadap batas.
8. Setelah loop, cetak jumlah.

Output:

Output 2.4

```
masukkan nilai batas = 5
1 + 2 + 3 + 4 + 5 = 15
```

Analisis Hasil:

Jika input batas = 5, output adalah "1 + 2 + 3 + 4 + 5 = 15". Kode ini memperluas kode 3 dengan input yang beragam. Landasan teori input/output menekankan pentingnya Scanner untuk fleksibilitas program, sementara loop tetap efisien.

2.5 Kode 5 : PerulanganFor3

Kode Program:

Kode Program 2.5

```
public static void main(String[] args) {  
    for (int line = 1; line ≤ 5; line++){  
        for ( int j = 1 ; j ≤ (-1 * line +5) ; j ++){  
            System.out.print(".");  
        }  
        System.out.print(line);  
        System.out.println( );  
    }  
}
```

Penjelasan Langkah Kerja:

1. Loop line dari 1 hingga 5 mengontrol baris.
2. Loop dalam j menghitung titik: $(-1 * \text{line} + 5)$ menentukan jumlah titik (5, 4, 3, 2, 1 untuk line 1-5).
3. Cetak titik dalam loop j.
4. Setelah titik, cetak line dengan baris baru.

Output :

Output 2.5

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

Analisis Hasil:

Ini menunjukkan nested loop untuk pola geometris. Landasan teori nested for menjelaskan kontrol dimensi: loop line untuk baris, dan loop j untuk kolom. Dengan $(-1 * \text{line} + 5)$ adalah rumus linier untuk jumlah titik menurun.

2.6 Kode 6 : PerulanganFor3

Kode Program :

Kode Program 2.6

```
public static void main(String[] args) {  
    for (int i = 1; i ≤ 5 ; i++){  
        for (int j = 1; j ≤ 5 ; j++) {  
            System.out.print("*");  
        }  
        System.out.println();  
    }  
}
```

Penjelasan Langkah Kerja:

1. Loop luar (i) dari 1 hingga 5 untuk baris.
2. Loop dalam (j) dari 1 hingga 5 mencetak "*" untuk setiap kolom.
3. Setelah loop dalam, baru cetak baris

Output :

Output 2.6

```
*****
*****
*****
*****
*****
```

Analisis Hasil:

Output adalah grid 5x5 bintang:

Landasan teori nested for menunjukkan struktur untuk matriks persegi. Dengan loop luar mengatur tinggi dan dalam mengatur lebar, mendukung algoritma pengolahan array multidimensi.

2.7 Kode 7 : PerulanganFor3

Kode Program :

Kode Program 2.7

```
public static void main(String[] args) {
    for ( int i = 0 ; i ≤ 5; i++) {
        for ( int j = 0 ; j ≤ 5; j++) {
            System.out.print(i+j + " ");
        }
        System.out.println();
    }
}
```

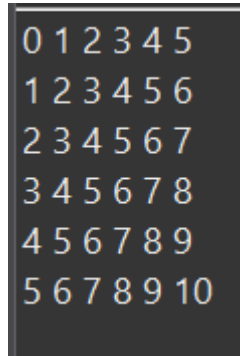
Penjelasan Langkah Kerja:

1. Loop luar (i) dari 0 hingga 5.
2. Loop dalam (j) dari 0 hingga 5, untuk mencetak (i + j) + " ".

3. Baris baru setelah setiap loop dalam.

Output :

Output 2.7



0	1	2	3	4	5
1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10

Analisis Hasil:

Output adalah tabel penjumlahan.

Ini menghasilkan tabel adisi. Landasan teori nested for dengan operasi aritmatika menunjukkan kemampuan untuk menghasilkan data terstruktur, seperti tabel kebenaran atau matriks, yang esensial dalam algoritma komputasi numerik dan analisis data.

BAB III

KESIMPULAN

3.1 Kesimpulan

Laporan praktikum ini membahas konsep dasar algoritma dan pemrograman dengan fokus pada perulangan for dan nested for dalam bahasa Java. Bab Pendahuluan menjelaskan latar belakang pentingnya pemahaman perulangan untuk efisiensi program, tujuan praktikum yang meliputi pemahaman konsep, implementasi, dan analisis kode, serta manfaat seperti peningkatan keterampilan debugging dan persiapan untuk topik lanjutan.

Bab Pembahasan menganalisis tujuh contoh kode, mulai dari perulangan sederhana untuk pencetakan angka dan akumulasi nilai, hingga nested for untuk pola geometris dan tabel numerik. Setiap kode dijelaskan langkah kerjanya, dianalisis hasilnya berdasarkan landasan teori seperti struktur kontrol loop, akumulasi data, dan manipulasi I/O, menunjukkan aplikasi praktis dalam pengolahan data linier dan multidimensi.

3.2 Saran

Untuk pengembangan lebih lanjut, mungkin kita dapat mengeksplorasi variasi perulangan lainnya, seperti while dan do-while, untuk membandingkan beberapa kejadian yang berbeda. Selain itu kita dapat memperdalam untuk melalui proyek kecil, misalnya membuat aplikasi kalkulator atau generator pola. Praktikum ini juga dapat diperluas dengan studi kasus nyata, seperti analisis algoritma sorting menggunakan nested loop, untuk meningkatkan keterampilan pemecahan masalah. Terakhir, penggunaan alat debugging seperti IDE (Integrated Development Environment) disarankan untuk meminimalkan kesalahan kode dan mempercepat pembelajaran.

DAFTAR PUSTAKA

- [1] H. Schildt, Java: A Beginner's Guide, 8th ed., McGraw-Hill Education, New York, 2018.
- [2] R. Sedgewick and K. Wayne, Algorithms, 4th ed., Addison-Wesley, Boston, 2011.
- [3] Oracle Corporation, "The Java Tutorials: The for Statement," Oracle, 2023. [Online]. Available: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html>. [Accessed: Oct. 15, 2023].
- [4] D. Flanagan, Java in a Nutshell, 7th ed., O'Reilly Media, Sebastopol, 2019.
- [5] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Data Structures and Algorithms in Java, 6th ed., Wiley, Hoboken, 2014.