6. Percabangan dan Perulangan Kompleks

6.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menjelaskan dan menggunakan struktur kontrol percabangan dan perulangan kompleks.
2. Dapat membuat program dengan struktur kontrol percabangan dan perulangan kompleks

6.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

6.3 Materi

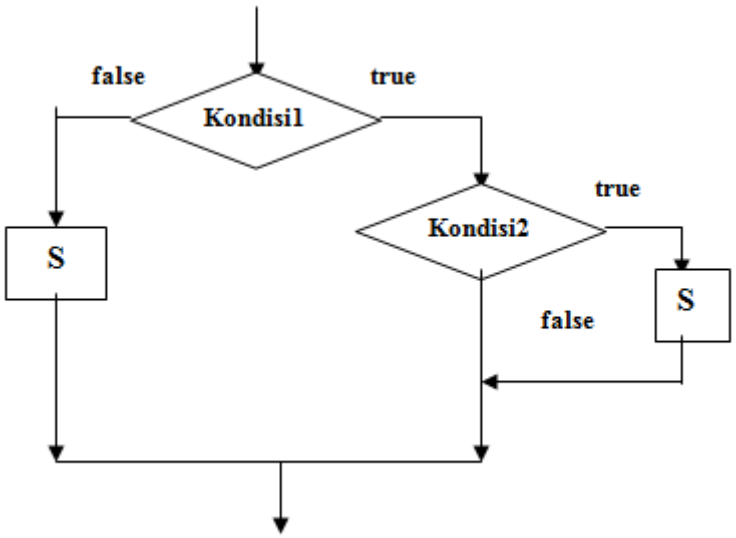
6.3.1 Struktur Percabangan Kompleks

Percabangan di mana kondisi pemilihan tidak hanya satu tetapi bisa terdiri atas banyak alternatif. Perintah-perintah yang dikerjakan juga bisa lebih dari satu. Percabangan kompleks bentuk 1:

```
1 if kondisi1:
2     if kondisi2:
```

```
        S
        S
else:
    S
    S
```

Gambar flowchart bentuk 1 ada pada 6.1.



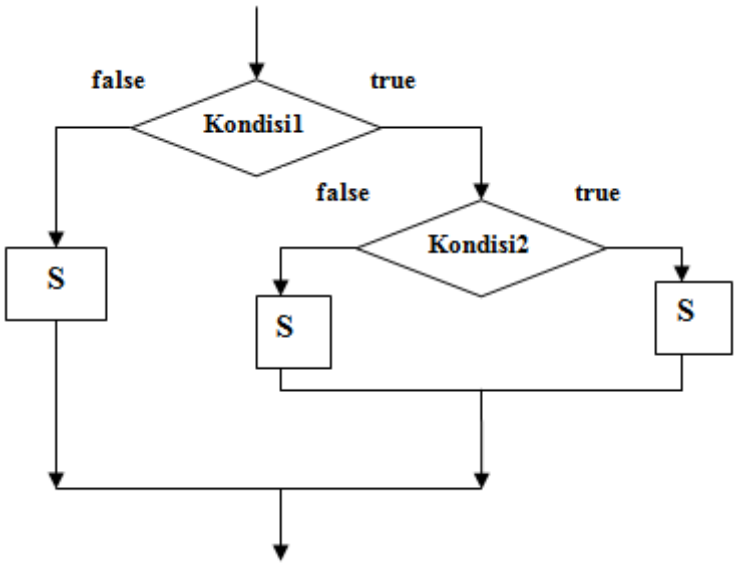
Gambar 6.1: Flowchart Percabangan Kompleks Bentuk 1

Percabangan kompleks bentuk 2:

```
if kondisi1:
    if kondisi2:
        S
        S
    else:
        S
        S
else:
    S
    S
```

Gambar flowchart bentuk 2 ada pada 6.2
Percabangan kompleks bentuk 3:

```
if kondisi1:
    S
    if kondisi2:
        S
        S
else:
    if kondisi3:
```



Gambar 6.2: Flowchart Percabangan Kompleks Bentuk 2

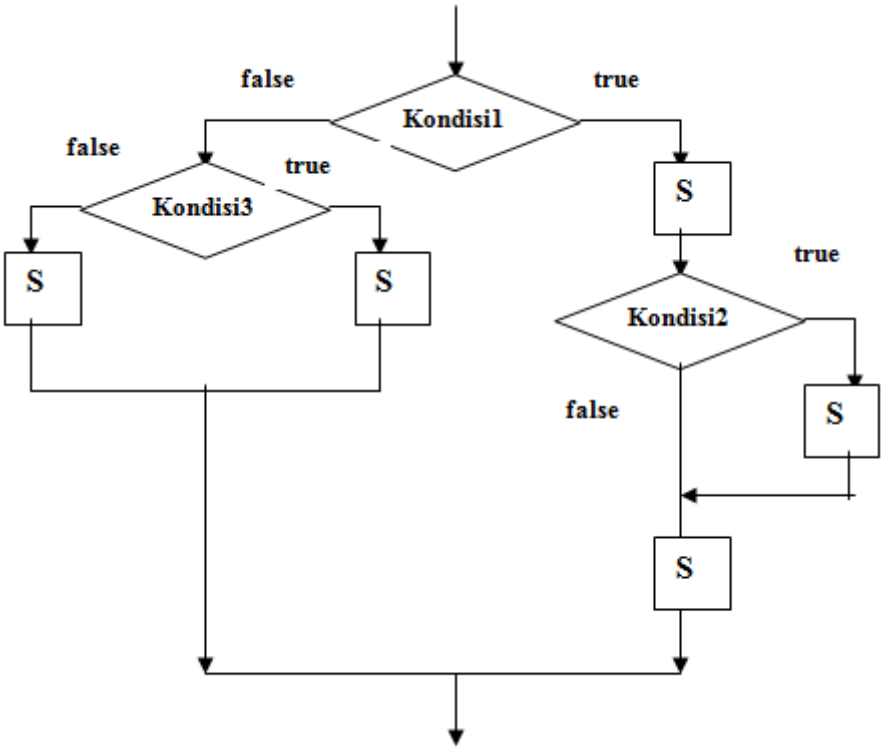
```
        S
        S
    else:
        S
        S
```

Gambar flowchart bentuk 3 ada pada 6.3.
Percabangan kompleks bentuk 4:

```
if kondisi1:
    S
    if kondisi2:
        S
        S
    else:
        S
        S
else:
    if kondisi3:
        S
        S
    else:
        S
        S
S
```

Gambar flowchart bentuk 4 ada pada 6.4.
Percabangan kompleks bentuk 5:

```
if kondisi1:
    if kondisi2:
```



Gambar 6.3: Flowchart Percabangan Kompleks Bentuk 3

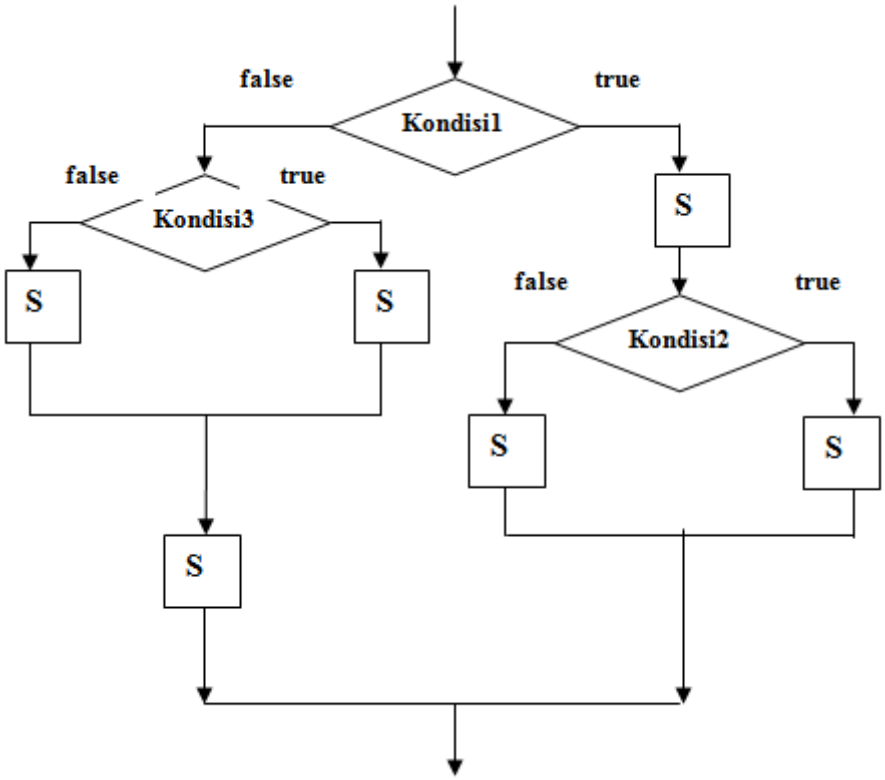
```
        if kondisi3:
            if kondisi4:
                S
```

Gambar flowchart bentuk 5 ada pada 6.5.
Percabangan kompleks bentuk 6:

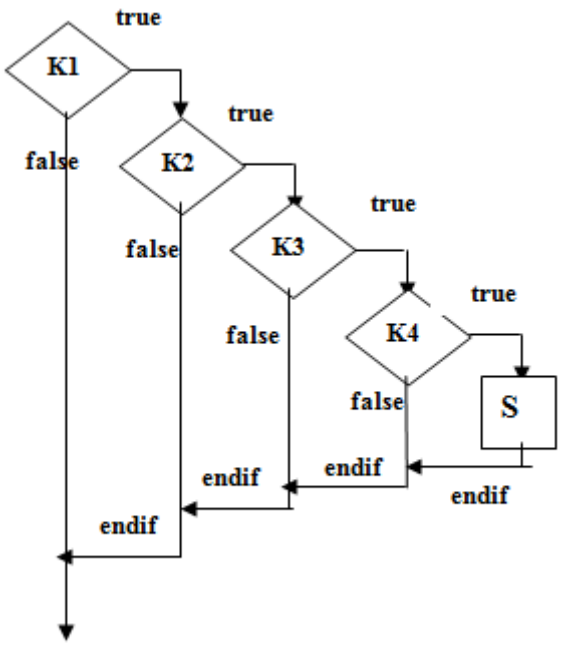
```
if kondisi1:
    S
else:
    if kondisi2:
        S
    else:
        if kondisi3:
            S
        else:
            if kondisi4:
                S
            else:
                S
```

Gambar flowchart bentuk 6 ada pada 6.6.

Pada kasus tertentu IF bertingkat memiliki keuntungan tersendiri. Dengan menggunakan IF bertingkat maka eksekusi perintah menjadi lebih baik dan efisien. Dengan menggunakan IF bertingkat maka tidak semua kondisi IF dikerjakan sehingga waktu eksekusi lebih cepat. Sedangkan

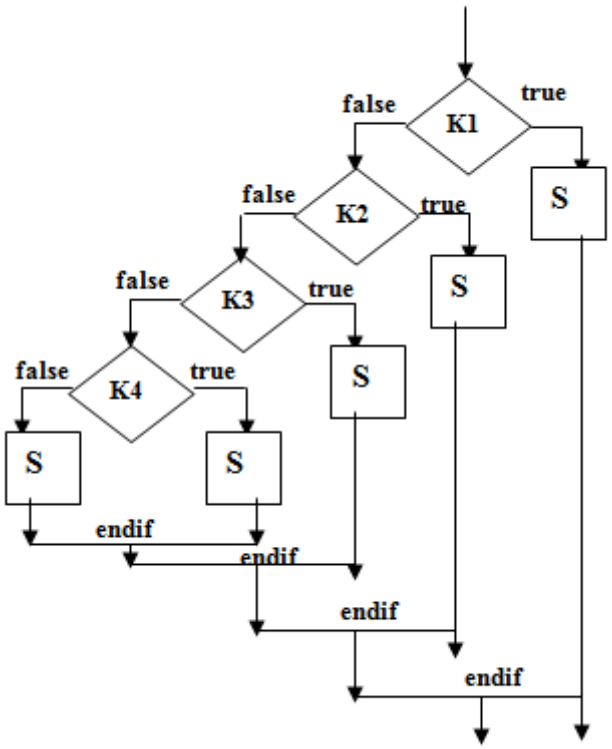


Gambar 6.4: Flowchart Percabangan Kompleks Bentuk 4



Gambar 6.5: Flowchart Percabangan Kompleks Bentuk 5

pada IF biasa semua kondisi IF pasti akan dicoba satu per satu walaupun mungkin pada akhirnya hanya satu saja perintah IF yang terpenuhi. Hal ini tentu memperlambat proses. Contoh kasus yang cocok untuk IF bertingkat adalah kasus konversi nilai angka menjadi nilai huruf dengan



Gambar 6.6: Flowchart Percabangan Kompleks Bentuk 6

batasan-batasan nilai yang sudah ditentukan sebelumnya.

```
if (kondisi1):
    instruksi1
elif(kondisi2):
    instruksi2
elif(kondisi3):
    instruksi3
elif(kondisi4):
    instruksi4
```

Bedakan dengan:

```
if (kondisi1):
    instruksi1
if(kondisi2):
    instruksi2
if(kondisi3):
    instruksi3
if(kondisi4):
    instruksi4
```

6.3.2 Struktur Perulangan Kompleks

Break

Perintah ini digunakan untuk menghentikan proses perulangan yang sedang terjadi. Biasanya disebabkan oleh suatu kondisi tertentu yang diimple-mentasikan menggunakan perintah IF.

```
for i in range(1000):
    print(i)
    if i==10:
        break
```

Output:

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

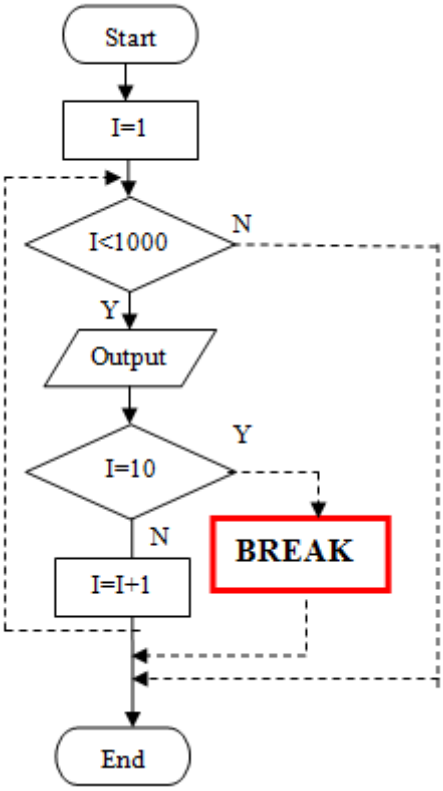
Program di atas akan menampilkan angka 1 sampai dengan 10 walaupun di perulangan sudah diset dari 1 sampai dengan 1000. Hal ini karena perintah break yang diberikan pada saat kondisi i=10. Angka 10 masih ditampilkan karena perintah untuk mencetak diletakkan sebelum perintah break. Gambar break dari kode program demo break ada pada 6.7:
Contoh program break lain:

```
    for i in range(1000):
        if i==10:
            break
        print(i)
```

Output:

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

Program di atas akan menampilkan angka 1 sampai dengan 9 walaupun di perulangan sudah diset dari 1 sampai dengan 1000. Hal ini karena perintah break yang diberikan pada saat kondisi i=10. Angka 10 tidak ditampilkan karena perintah untuk mencetak diletakkan sesudah perintah break. Gambar break dari kode program demo break ada pada 6.8:



Gambar 6.7: Flowchart Program Menggunakan Break

Continue

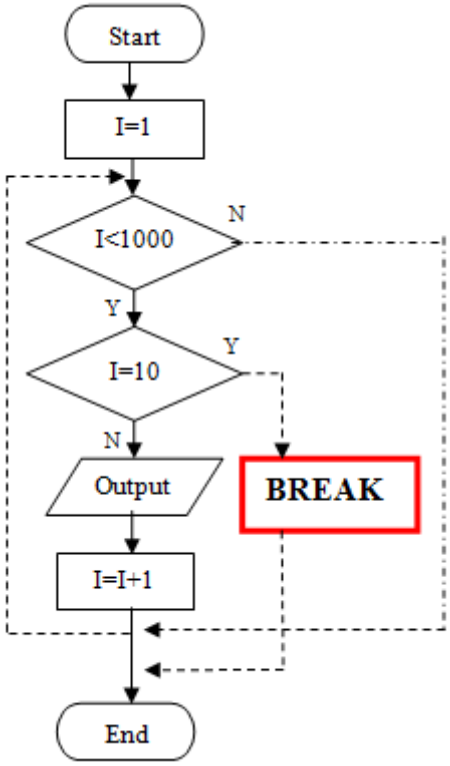
Perintah continue menyebabkan proses perulangan kembali ke awal mula, dengan mengabaikan statement-statement berikutnya setelah continue. Biasanya perintah continue juga diimplementasikan menggunakan perintah IF.
Contoh program continue:

```
    for i in range(10):
        if i==5:
            continue
        print(i)
```

Output:

```
0
1
2
3
4
6
7
8
9
```

Program di atas tidak menampilkan angka 5 karena pada saat angka 5 akan ditampilkan, perintah



Gambar 6.8: Flowchart Program Menggunakan Break 2

continue dijalankan, sehingga perintah mencetak di bagian bawahnya tidak akan dikerjakan dan langsung dilanjutkan ke perulangan berikutnya!
Gambar break dari kode program demo continue ada pada 6.9:

Perulangan Bertingkat

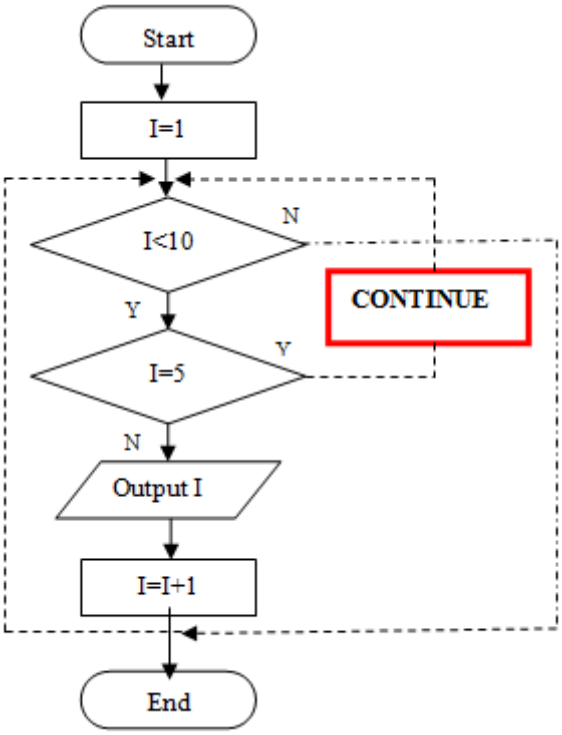
Yang dimaksud dengan struktur perulangan kompleks adalah bentuk per-ulangan di mana di dalam suatu perulangan terdapat perulangan lain, sehingga terjadi perulangan bertingkat yang mengakibatkan waktu proses semakin lama. Ada banyak algoritma tertentu yang tepat untuk meng-gunakan struktur perulangan kompleks. Masalah yang dapat diselesaikan dengan perulangan kompleks adalah masalah matriks yang menggunakan array 2 dimensi, masalah game board seperti catur dan minesweeper, ataupun masalah pengolahan citra digital seperti algoritma untuk mendeteksi tepi citra, algoritma untuk mengubah citra berwarna menjadi grayscale, atau berbagai hal lain. Intinya, masalah yang dapat diselesaikan menggunakan perulangan kompleks biasanya memiliki pola grid (kotak-kotak) yang memiliki lebar dan panjang. Berikut ini adalah sebuah contoh perulangan kompleks:

Program 1:

```
for i in range(m):
    <lakukan perintah ini>
    <lakukan perintah itu>
```

Program 2:

```
for j in range(n):
```



Gambar 6.9: Flowchart Program Menggunakan Continue

```
    <lakukan perintah ini>
    <lakukan perintah itu>
```

Program 1 mengerjakan perulangan sebanyak 10 kali, sedangkan program 2 mengerjakan perulangan sebanyak 5 kali. Kedua program tersebut berjalan sendiri-sendiri. Jika perulangan pada program 2 "dimasukkan" ke dalam perulangan program 1, maka menjadi:

```
for i in range(m):
    for j in range(n):
        <lakukan perintah ini di inner>
        <lakukan perintah itu di inner>
    <lakukan perintah lain di outer>
    <lakukan perintah lain lagi di outer>
```

Bagian for i menjadi outer loop, sedangkan bagian for j menjadi inner loop. Alur pengerjaannya adalah sebagai berikut: untuk setiap 1x s/d m outer loop dijalankan akan dikerjakan n kali inner loop.

Untuk sintaks dalam bentuk sama saja. Perhatikan contoh berikut:

```
    while(i<=m):
        while(j<=n):
            <lakukan perintah ini di inner>
            <lakukan perintah itu di inner>
        <lakukan perintah lain di outer>
        <lakukan perintah lain lagi di outer>
```

Bagian while i menjadi outer loop, sedangkan bagian while j menjadi inner loop. Alur pengerjaannya adalah sebagai berikut: untuk setiap 1x s/d m outer loop dijalankan akan dikerjakan n kali inner loop.

6.4 Kegiatan Praktikum

Praktikum akan membuat beberapa program menggunakan perulangan dan percabangan kompleks dengan beberapa studi kasus tampilan deret berikut:

■ **Contoh 6.1** Anda diminta untuk membuat suatu deret dengan tampilan sebagai berikut:

```
n=5
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Pembahasan: Untuk dapat membuat program seperti di atas logika adalah sebagai berikut:

- Karena n = misalnya 5 maka n digunakan sebagai jumlah baris, yang dalam perulangan nested adalah outer loop, i=1 s/d n.
- Untuk inner loop kita perhatikan bahwa untuk baris i=1 maka dilakukan perulangan ke kanan 1x (atau sama dengan i x)
- Untuk setiap loop pada inner, dilakukan j=1 s/d i, dan tampilkan i nya. Untuk inner loop jangan dilakukan enter, sehingga harus diperhatikan end=" pada printnya.
- Lakukan hal tersebut sampai selesai. Untuk setiap akhir j, lakukan enter (ganti baris)

Kode program dapat dibuat sebagai berikut:

```
1     n=int(input("Masukkan n = "))
2     for i in range(1,n+1):
3         for j in range(1,i+1):
4             print(i," ",end='')
5         print()
```

■

■ **Contoh 6.2** Anda diminta untuk membuat suatu deret dengan tampilan sebagai berikut:

```
n=5
1 2 3 4 5
5 4 3 2 1
1 2 3 4 5
5 4 3 2 1
1 2 3 4 5
```

Pembahasan: Untuk dapat membuat program seperti di atas logika adalah sebagai berikut:

- Karena n = misalnya 5, maka n akan menjadi outer loop (baris). i=1 s/d n.
- Untuk inner loop kita juga akan melakukan perulangan smp dengan n atau turun dari n s/d 1, tergantung barisnya genap atau ganjil. Jika ganjil inner loop bergerak naik, jika genap maka inner loop bergerak turun, j=n s/d 0, step -1.
- Untuk setiap loop pada inner, setelah habis maka lakukan enter / ganti baris dari outer loop.

Kode program dapat dibuat sebagai berikut:

```
        n=int(input("Masukkan n = "))
        for i in range(1,n+1):
            if i%2==1:
                for j in range(1,n+1):
                    print(j," ",end='')
            else:
                for j in range(n,0,-1):
                    print(j," ",end='')
            print()
```

■

■ **Contoh 6.3** Anda diminta untuk membuat suatu deret dengan tampilan sebagai berikut:

```
n=6
X O X O X O
X O X O X
X O X O
X O X
X O
X
```

Pembahasan: Untuk dapat membuat program seperti di atas logika adalah sebagai berikut:

- Karena n = misalnya 6, maka n akan menjadi outer loop (baris). i=0 s/d n.
- Untuk inner loop kita juga akan melakukan perulangan dari 1 smp dengan n-i. Untuk setiap j ganjil maka tampilkan 'X', sedangkan jika genap maka tampilkan 'O'.
- Untuk setiap loop pada inner, setelah habis maka lakukan enter / ganti baris dari outer loop.

Kode program dapat dibuat sebagai berikut:

```
        n=int(input("Masukkan n = "))
        for i in range(0,n+1):
            for j in range(1,n-i+1):
                print("X",end='') if j\%2==1 else print("O",end='')
            print()
```

■

■ **Contoh 6.4** Anda diminta untuk membuat suatu deret dengan tampilan pada gambar 6.10.

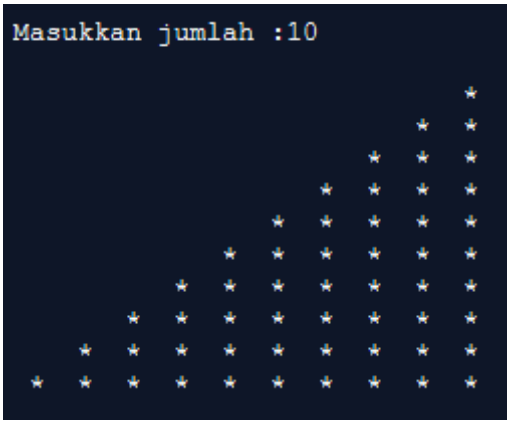
Pembahasan: Untuk dapat membuat kode program seperti di atas, maka logikanya adalah sebagai berikut:

- Karena n = misalnya 10, maka n akan menjadi outer loop (baris). i=n turun smp dengan 1.
- Untuk inner loop kita harus menuliskan terlebih dahulu spasi kosong sebanyak n-i-1. Setelah itu digabungkan dengan menampilkan bintang (*) sebanyak sisanya (1 s/d n-i-1)
- Untuk setiap loop pada inner, setelah habis maka lakukan enter / ganti baris dari outer loop.

Kode program dapat dibuat sebagai berikut:

```
        hasil = ""

        x = int(input("Masukkan jumlah :"))
        bar = x
```



Gambar 6.10: Soal Segitiga

```
5           # Looping Baris
6           while bar >= 0:
7
8               # Looping Kolom Spasi Kosong
9               kol = bar
10              while kol > 0:
11                  hasil += "   "
12                  kol -= 1
13
14              # Looping Kolom Bintang
15              kanan = 1
16              while kanan < (x - (bar-1)):
17                  hasil += " * "
18                  kanan += 1
19
20              hasil = hasil + "\n"
21              bar -= 1
22
23          print (hasil)
```

■

6.5 Latihan Mandiri

Latihan 6.1 Buatlah program untuk mencari bilangan prima terdekat dari suatu bilangan yang diinputkan oleh pengguna (n) dan nilai bilangan prima tersebut < n. Contoh: input n=12, maka prima terdekat < 12 adalah 11 Contoh: input n=21, maka prima terdekat < 21 adalah 19 ■

Latihan 6.2 Buatlah program untuk menampilkan deret seperti di bawah ini. n diinputkan secara dinamis

```
contoh: n = 6
720 6 5 4 3 2 1
120 5 4 3 2 1
24 4 3 2 1
6 3 2 1
2 2 1
1 1
```

■

Latihan 6.3 Buatlah program untuk menampilkan deret seperti di bawah ini. n diinputkan secara dinamis

```
contoh: tinggi = 5, lebar = 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
```

■