

Praktikum 4 (1/4)

PERULANGAN *for*

A. TUJUAN

1. Menjelaskan proses pengulangan menggunakan pernyataan *for*
2. Menjelaskan tentang variasi pernyataan *for*
3. Menjelaskan tentang pernyataan *for* dengan menentukan jumlah langkah

B. DASAR TEORI

Mengulang suatu proses merupakan tindakan yang banyak dijumpai dalam pemrograman. Pada semua bahasa pemrograman, pengulangan proses ditangani dengan suatu mekanisme yang disebut *loop*. Dengan menggunakan *loop*, suatu proses yang berulang misalnya menampilkan tulisan yang sama seratus kali pada layar dapat diimplementasikan dengan kode program yang pendek.

Pada pemrograman proses perulangan dapat dibagi menjadi 2 bagian utama yaitu:

1. Perulangan yang sudah di ketahui jumlah perulangannya sebelum perulangan tersebut di lakukan.
2. Perulangan yang belum di ketahui jumlah perulangannya sebelum perulangan tersebut di lakukan. Dalam hal ini dapat dibagi menjadi dua bagian yaitu:
 - a) kondisi perulangan diperiksa diawal perulangan.
 - b) kondisi perulangan diperiksa diakhir perulangan.

Untuk kasus 1 seharusnya menggunakan perulangan dengan pernyataan *for* dan akan dipelajari pada praktikum ini. Sedangkan pada kasus 2a dan 2b akan dibahas pada praktikum 4-2.

Bentuk pernyataan *for* :

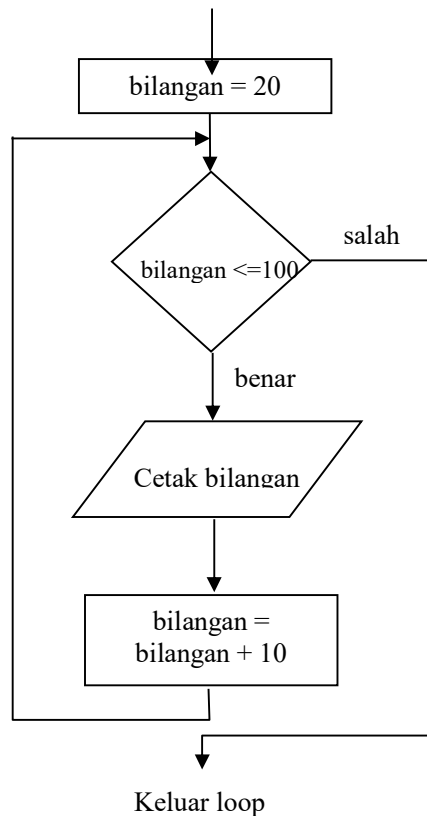
```
for (ungkapan1; ungkapan2; ungkapan3)
    pernyataan;
```

Keterangan:

- Ungkapan1 : digunakan untuk memberikan inisialisasi terhadap variabel pengendali *loop*.
- Ungkapan2 : dipakai sebagai kondisi untuk keluar dari *loop*.
- Ungkapan3 : dipakai sebagai pengatur kenaikan nilai variabel pengendali *loop*.

Ketiga ungkapan dalam *for* tersebut harus dipisahkan dengan tanda titik koma (;). Dalam hal ini pernyataan bisa berupa pernyataan tunggal maupun jamak. Jika pernyataannya berbentuk jamak, maka pernyataan-pernyataan tersebut harus diletakkan di antara kurung kurawal buka ({} dan kurung kurawal tutup (}), sehingga formatnya menjadi :

```
for (ungkapan1; ungkapan2; ungkapan3)
{
    pernyataan;
    pernyataan;
    .
    .
    .
}
```



Gambar 4.1. Diagram alir for

```
for(bilangan = 20; bilangan <= 100; bilangan += 10)
    printf("%d\n", bilangan);
```

Pada program di atas, kenaikan terhadap variabel pengendali *loop* sebesar 10 (positif), yang dinyatakan dengan ungkapan

```
bilangan += 10
```

Pada contoh yang melibatkan pernyataan *for* di atas, kenaikan variabel pengendali *loop* berupa nilai positif. Sebenarnya kenaikan terhadap variabel pengendali *loop* bisa diatur bernilai negatif.

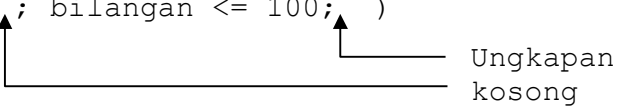
```
for (bilangan = 60; bilangan >= 10; bilangan -= 10)
    printf("%d\n", bilangan);
```

Kadang-kadang dijumpai adanya pernyataan *for* yang tidak mengandung bagian ungkapan yang lengkap (beberapa ungkapan dikosongkan). Dengan cara ini, pernyataan

```
for (bilangan = 20; bilangan <= 100; bilangan += 10)
    printf("%d\n", bilangan);
```

dapat ditulis menjadi :

```
bilangan = 20;          /* inisialisasi di luar for */
for ( ; bilangan <= 100; )
{
    printf("%d\n", bilangan);
    bilangan += 10;
}
```



Ungkapan
kosong

Pengosongan ini juga dilakukan pada ungkapan yang biasa dipakai untuk menaikkan nilai variabel pengendali *loop*. Sebagai gantinya, di dalam tubuh *loop* diberikan pernyataan untuk menaikkan nilai variabel pengendali *loop*, yaitu berupa

```
bilangan += 10;
```

Ungkapan yang tidak dihilangkan berupa `bilangan <= 100`. Ungkapan ini tetap disertakan karena dipakai sebagai kondisi untuk keluar dari *loop*.

Sesungguhnya ungkapan yang dipakai sebagai kondisi keluar dari *loop* juga bisa dihilangkan, sehingga bentuknya menjadi

```
for ( ; ; )
    pernyataan
```

Suatu pertanyaan mungkin timbul “Lalu bagaimana caranya kalau ingin keluar dari *loop* pada bentuk di atas?”. Caranya adalah dengan menggunakan pernyataan yang dirancang khusus untuk keluar dari *loop*. Mengenai hal ini akan dibahas pada praktikum selanjutnya.

C. TUGAS PENDAHULUAN

Buatlah desain flowchart untuk setiap soal dalam percobaan

D. PERCOBAAN

1. Gunakan loop *for* untuk menampilkan nilai 1 sampai dengan 20 dalam baris-baris yang terpisah.
2. Hitunglah bilangan triangular dari masukan pengguna, yang dibaca dari keyboard dengan menggunakan *scanf()*. Bilangan triangular adalah penjumlahan dari bilangan masukan dengan seluruh bilangan sebelumnya, sehingga bilangan triangular dari 7 adalah : $7 + 6 + 5 + 4 + 3 + 2 + 1$
3. Gunakan loop *for* untuk menampilkan seluruh karakter dari A sampai dengan Z dalam baris-baris yang terpisah.
4. Gunakan loop *for* dengan kenaikan variabel negatif untuk menampilkan seluruh karakter dari Z sampai dengan A dalam baris-baris yang terpisah.
5. Gunakan loop *for* untuk membuat program sebagai berikut:
input : n
output : 1 3 5 7 ... m (m = bilangan ganjil ke n)
6. Gunakan loop *for* untuk membuat program sebagai berikut:
input : n
output : 1 -2 3 -4 5 -6 7 -8 ... n
7. Gunakan loop *for* untuk membuat program sebagai berikut:
input : n
output : $1*2*3*4*5*... *n$ (faktorial)

E. LAPORAN RESMI

1. Tulis listing program dari semua percobaan yang dilakukan.
2. Kemudian tuliskan outputnya. Terangkan kenapa demikian.

Praktikum 4 (2/4)

PERULANGAN *while*

A. TUJUAN

1. Menjelaskan perbedaan *for* dan *while* serta *do-while*
2. Menjelaskan proses pengulangan menggunakan pernyataan *while*
3. Menjelaskan proses pengulangan menggunakan pernyataan *do-while*

B. DASAR TEORI

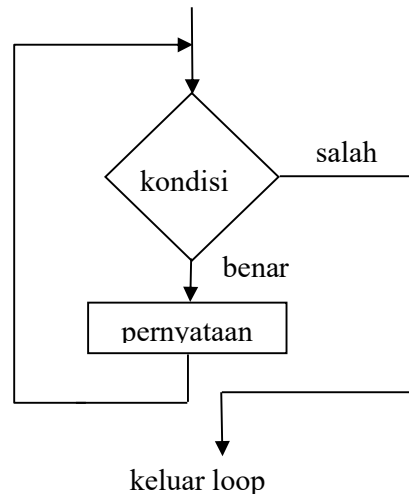
Seperti yang sudah dijelaskan pada bab sebelumnya, bahwa perulangan yang sudah diketahui berapa kali akan dijalankan, maka sebaiknya menggunakan perulangan *for*. Sedangkan yang belum diketahui berapakah kali akan diulangi maka dapat menggunakan *while* atau *do-while*. Menggunakan pernyataan *while* ketika kondisi perulangan diperiksa sebelum menjalankan pernyataan dan menggunakan *do-while* ketika perulangan kondisi perulangan tersebut diperiksa setelah menjalankan pernyataan.

Pada pernyataan *while*, pemeriksaan terhadap loop dilakukan di bagian awal (sebelum tubuh loop). Pernyataan *while* akan diulangi terus selama kondisi bernilai benar, atau diulangi selama kondisinya benar, jika kondisinya salah maka perulangan (loop) dianggap selesai. Lebih jelasnya, bentuk pernyataan *while* adalah sebagai berikut :

```
while (kondisi)
    pernyataan;
```

dengan pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Dengan melihat gambar 4-2.1, tampak bahwa ada kemungkinan pernyataan yang merupakan tubuh loop tidak dijalankan sama sekali, yaitu kalau hasil pengujian kondisi *while* yang pertama kali ternyata bernilai salah.

Proses pengulangan terhadap pernyataan dijelaskan pada gambar berikut :



Gambar 4-2.1 Diagram alir *while*

C. TUGAS PENDAHULUAN

Desain flowchart untuk setiap soal dalam percobaan

D. PERCOBAAN

1. Gunakan loop *while* untuk membuat program yang dapat menampilkan semua karakter yang diketikkan di keyboard sampai yang diketikkan pada keyboard huruf 'X' (x besar).
2. Gunakan loop *while* untuk membuat program yang dapat mencari total angka yang dimasukkan dengan tampilan sebagai berikut :

```
Masukkan bilangan ke-1 : 5
Mau memasukkan data lagi [y/t] ? y
```

```
Masukkan bilangan ke-2 : 3
Mau memasukkan data lagi [y/t] ? t
```

```
Total bilangan = 8
```

3. Pada program no 2 tambahkan rata-rata, maksimum dan minimum dari angka yang dimasukkan.
4. Gunakan *while* pada program yang digunakan untuk menghitung banyaknya karakter dari kalimat yang dimasukkan melalui keyboard (termasuk karakter spasi). Untuk mengakhiri pemasukan kalimat, tombol ENTER ('\\n') harus ditekan
Input : Ketikkan sembarang kalimat
Output : jumlah karakter = m
 jumlah spasi =n

E. LAPORAN RESMI

1. Tulis listing program dari semua percobaan yang dilakukan.
2. Kemudian tuliskan outputnya. Terangkan kenapa demikian.
3. Tambahkan program yang ada pada soal no 4 dengan menghitung huruf kecil, huruf besar dan karakter angka.