PRAKTIKUM 5
PENGULANGAN PROSES 1

A. Tujuan :

1. Menjelaskan proses pengulangan menggunakan pernyataan for
2. Menjelaskan proses pengulangan menggunakan pernyataan while
3. Menjelaskan proses pengulangan menggunakan pernyataan do-while

B. DASAR TEORI

Pernyataan *for*

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 diimpelementasikan dengan kode program yang pendek.

Pernyataan pertama yang digunakan untuk keperluan pengulangan proses adalah pernyataan *for*. Bentuk pernyataan ini :

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

Kegunaan dari masing-masing ungkapan pada pernyataan *for.*

- 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 pernyatan 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;
   .
   .
   .
}
```

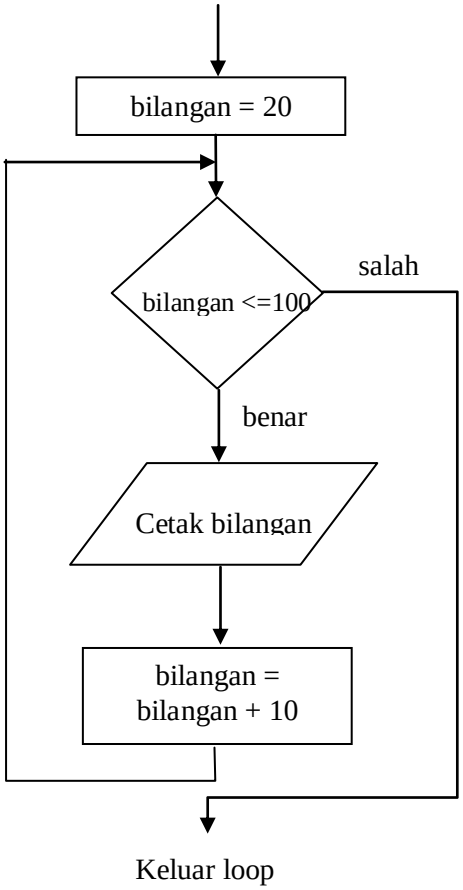
Contoh penggunaan *for*, misalnya untuk menampilkan deretan angka sebagai berikut :

```
20
30
40
50
.
.
.
100
```

Untuk keperluan ini, pernyataan *for* yang digunakan berupa :

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

Kalau digambarkan dalam bentuk diagram alir, akan terlihat sbb :



Gambar 5.1. Diagram alir for

```
/* File program : for1.c
Contoh pemakaian for untuk membentuk deret naik */

#include <stdio.h>

main()
{
     int bilangan;

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

Contoh eksekusi :

```
20
30
40
50
60
70
80
90
100
```

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

```
bilangan += 10
```

yang sama artinya dengan

```
bilangan = 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. Cara ini dapat digunakan untuk memperoleh deret sebagai berikut :

```
60
50
40
30
20
10
```

Untuk itu selengkapnya program yang dibutuhkan adalah sebagai berikut :

```
/* File program : for2.c
   Contoh pemakaian for untuk membentuk deret turun */

#include <stdio.h>

main()
{
     int bilangan;

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

Contoh eksekusi :

```
60
50
40
30
20
10
```

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;  )
                                   ungkapan
                                   kosong
  {
     printf("%d\n", bilangan);
     bilangan += 10;
  }
```

Tampak bahwa ungkapan yang biasa dipakai untuk inisialisasi variabel pengendali *loop* tak ada. Sebagai gantinya pengendalian *loop* diatur sebelum pernyataan for, berupa

```
bilangan = 20;
```

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
```

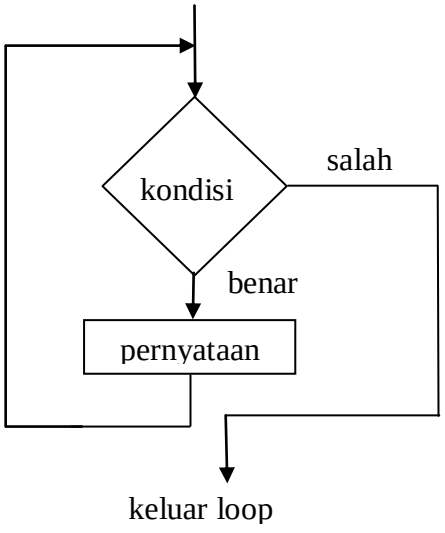
Suatu pertanyaan mungkin timbul “Lalu bagaimana caranya kalau ingin keluar dari *loop* pada bentuk di atas?”. Caranya adalah dengan menggunakan pernyataan yang dirancang khusus untuk keluar dari *loop*. Mengenai hal ini akan dibahas pada sub bab yang lain.

Pernyataan *while*

Pada pernyataan *while*, pengecekan terhadap loop dilakukan di bagian awal (sebelum tubuh loop). Lebih jelasnya, bentuk pernyataan *while* adalah sebagai berikut :

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

dengan pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Proses pengulangan terhadap pernyataan dijelaskan pada gambar berikut :



Gambar 5.2. Diagram alir *while*

Dengan melihat gambar 5.2, 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.

Contoh pemakaian *while* misalnya untuk mengatur agar tombol yang ditekan oleh pemakai program berupa salah satu diantara 'Y','y', 'T' atau 't'. Impelementasinya :

```
/*File program : pilihan.c
Untuk membaca tombol Y atau T */

#include <stdio.h>

main()
{
     char pilihan;
     /* diberi nilai salah lebih dahulu */
     int sudah_benar = 0;

     printf("Pilihlah Y atau T.\n");

/* program dilanjutkan jika tombol Y,y,T atau t ditekan */

     while(!sudah_benar)
     {
        pilihan = getchar();        /* baca tombol */
        sudah_benar = (pilihan == 'Y') || (pilihan == 'y')||
           (pilihan == 'T') || (pilihan == 't');
     }

     /* memberi keterangan tentang pilihan */
     switch(pilihan)
     {
     case 'Y':
     case 'y':
          puts("\nPilihan anda adalah Y");
          break;
     case 'T':
     case 't':
          puts("\nPilihan anda adalah T");
     }
}
```

Contoh eksekusi :

```
Pilihlah Y atau T
Pilihan anda adalah Y
```

Inisialisasi terhadap variabel sudah_benar yang akan dijalankan pada kondisi *while* dengan memberi nilai awal bernilai *false* (`sudah_benar = 0`) dimaksudkan agar tubuh loop dijalankan minimal sekali.

Contoh lain pemakaian *while* dapat dilihat 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. Karena itu, tombol ENTER inilah yang dijadikan kondisi penghitungan jumlah spasi maupun karakter seluruhnya. Lengkapnya, kondisi yang dipakai dalam *while* berupa :

```
while((kar = getchar()) != '\n')
```

Ungkapan di atas mempunyai arti :

- Bacalah sebuah karakter dan berikan ke variabel **kar**
- Kemudian bandingkan apakah karakter tersebut = '\n' (ENTER)

Ungkapan menghasilkan nilai benar jika tombol yang ditekan bukan ENTER. Pada program kalau tombol yang ditekan bukan ENTER , maka :

- Jumlah karakter dinaikkan sebesar satu melalui pernyataan : `jumkar++;`
- Kalau karakter berupa SPASI, maka jumlah spasi dinaikkan sebesar satu, melalui pernyataan : `if (kar == ' ') jumspasi++;`

```
/* File program : jumkar.c
Menghitung jumlah kata dan karakter dalam suatu kalimat */

#include <stdio.h>

main()
{
     char kar;
     int jumkar = 0, jumspasi = 0;

     puts("Masukkan sebuah kalimat dan akhiri dgn ENTER.\n");
     puts("Saya akan menghitung jumlah karakter ");
     puts("pada kalimat tersebut.\n");

     while((kar = getchar()) != '\n')
     {
          jumkar++;
          if (kar == ' ') jumspasi++;
     }

     printf("\nJumlah karakter          = %d", jumkar);
     printf("\nJumlah SPASI             = %d\n\n", jumspasi);
}
```

Contoh eksekusi :

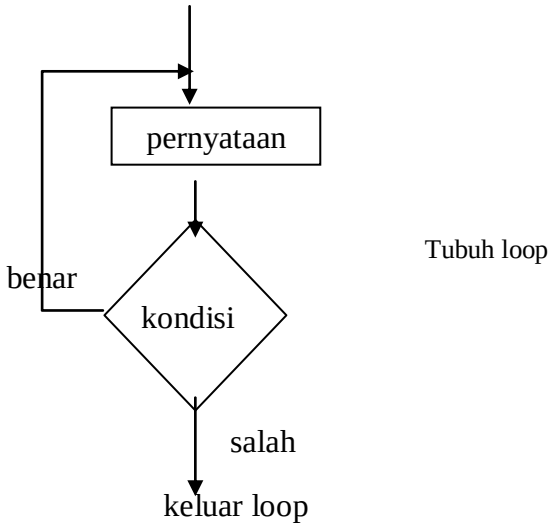
```
Masukkan sebuah kalimat, akhiri dgn ENTER.
Saya akan menghitung jumlah karakter pada kalimat tersebut.

Belajar bahasa C sangat menyenangkan

Jumlah karakter  = 36
Jumlah SPASI     = 4
```

Pernyataan *do-while*

Bentuk pernyataan ***do-while***

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

Pada pernyataan *do-while*, tubuh *loop* berupa pernyataan,dengan pernyataan bisa berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Pada pernyataan *do*, mula-mula pernyataan dijalankan. Selanjutnya, kondisi diuji. Sendainya kondisi bernilai benar, maka pernyataan dijalankan lagi, kemudian kondisi diperiksa kembali, dan seterusnya. Kalau kondisi bernilai salah pada saat dites, maka pernyataan tidak dijalankan lagi. Untuk lebih jelasnya dapat dilihat pada Gambar 4.3. Berdasarkan Gambar 5.3 terlihat bahwa tubuh *loop* minimal akan dijalankan sekali.



Gambar 5.3. Diagram alir *do-while*

Program berikut memberikan contoh pemakaian *do-while* untuk mengatur penampilan tulisan "BAHASA C" sebanyak sepuluh kali.

Contoh:

```
i = 0;
do
{
   puts("BAHASA C");
   i++;
} while(i<10);
```

Pada program di atas, variabel pencacah dipakai untuk menghitung jumlah tulisan yang sudah ditampilkan pada layar. Selama nilai pencacah kurang dari 10, maka perintah

```
puts("BAHASA C");
```

akan dilaksanakan kembali

Penanganan pembacaan tombol pada contoh program **pilihan.c** yang memakai *while* di atas, kalau diimplementasikan dengan memakai *do-while* adalah sebagai berikut

---

```
/* File program : pilihan2.c
Untuk membaca tombol Y atau T */

#include <stdio.h>
main()
{
     char pilihan;
     int sudah_benar;

     printf("Pilihlah Y atau T.\n");

/* program dilanjutkan kalau tombol Y,y,T atau t ditekan */
     do
     {
         pilihan = getchar( ); /* baca tombol */
         sudah_benar = (pilihan == 'Y') || (pilihan== 'y')||
           (pilihan == 'T') || (pilihan == 't');
     } while(! sudah_benar);

     /* memberi keterangan tentang pilihan */
     switch(pilihan)
     {
     case 'Y':
     case 'y':
          puts("\nPilihan anda adalah Y");
```

```
            break;
      case 'T':
      case 't':
            puts("\nPilihan anda adalah T");
      }
}
```

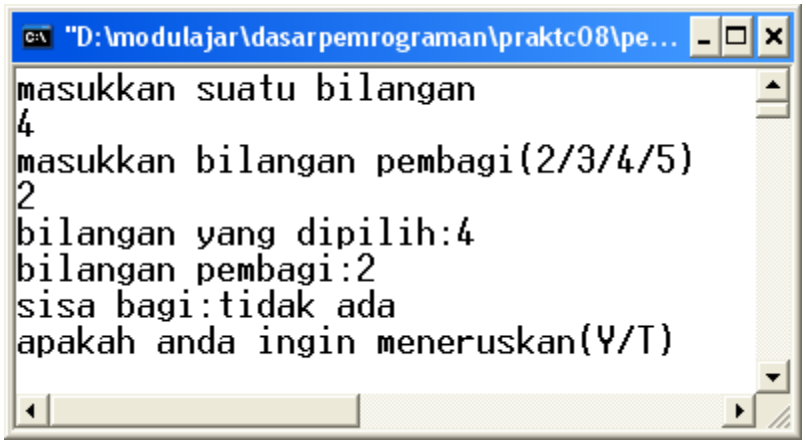
Contoh eksekusi :

```
Pilihlah Y atau T
Pilihan anda adalah T
```

Mula-mula tombol dibaca dengan menggunakan *getchar()* dan kemudian diberikan ke variabel pilihan. Sesudah itu, variabel **sudah_benar** akan diisi dengan nilai benar (1) atau salah (0) tergantung dari nilai pilihan. Kalau pilihan berisi salah satu diantara 'Y','y', 'T' atau 't', maka sudah berisi salah satu diantara 'Y', 'y', 'T' atau 't', maka **sudah_benar** akan berisi benar. Nilai pada vaiabel **sudah_benar** ini selanjutnya dijadikan sebagai kondisi *do-while*. Pengulangan terhadap pembacaan tombol akan dilakukan kembali selama **sudah_benar** benilai salah.

C. TUGAS PENDAHULUAN

1. Buatlah program untuk menentukan sisa hasil pembagian antara bilangan yang dimasukkan dengan bilangan pembagi . Apabila sisa baginya=0 maka dicetak tidak ada dan kalau ada sisa baginya, maka sisa bagi tersebut ditampilkan. Bila program tersebut dijalankan, hasilnya adalah sebagai berikut:



D. PERCOBAAN

1. Dengan menggunakan pernyataan for, buatlah program untuk menjumlahkan integer mulai dari 1 sampai dengan harga batas yang dibaca dari keyboard. Penjumlahan dapat dilakukan dengan menggunakan sebuah tempat penampungan hasil penjumlahan, dan penjumlahan dilakukan satu per satu terhadap angka mulai 1 sampai dengan angka yang dibaca dengan hasil sementara yang telah disimpan. Jika angka terakhir telah dijumlahkan, maka penampungan hasil sementara menjadi hasil akhir.

   ```
   Tampilan:
   Masukkan integer positif :10
   Jumlah 1 sampai 10=55
   ```

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. Buatlah program yang menentukan sebuah integer dengan acak, dan membaca sebuah integer berulang kali sampai integer yang dibaca sama dengan integer yang ditentukan secara acak.

   ```
   Tampilan:
   (misalkan angka hasil pengacakan adalah)
   Angka tebakan:34
   Tebakan terlalu kecil
   Angka Tebakan:55
   Tebakan Terlalu besar
   Angka Tebakan:50
   Tebakan benar
   ```

4. Buatlah program yang menerima sebuah respons atau jawaban ’s’ untuk ’sudah’ dan ’b’ untuk ’belum’ terhadap sebuah pertanyaan, ’Anda sudah sholat??’. Program akan terus menanyakan jawaban sampai jawaban yang diberikan ’s’ atau ’b’. Jika jawaban yang diberikan adalah ’s’, tampilkan pesan di layar ’bagus!’, dan jika

jawaban yang diberikan adalah ’b’, tampilkan pesan di layar, ’sholat adalah ibadah’.

```
Tampilan:
Anda sudah sholat??(s/b) 8
Anda sudah sholat??(s/b) <
Anda sudah sholat??(s/b) b
Bagus!

Anda sudah sholat??(s/b) 9
Anda sudah sholat??(s/b) n
Anda sudah sholat??(s/b) b
Sholat adalah ibadah
```

E. LAPORAN RESMI

1. Buatlah flowchart dari seluruh percobaan yang telah dilakukan!