

**Bab 5 : Perulangan**

- Perulangan dalam program
- Counting Loop Dan Pernyataan While
- Menghitung Dalam Perulangan
- Pernyataan FOR
- Perulangan Kondisional
- Desain Perulangan
- Perulangan Bersarang
- Pernyataan do-while dan Perulangan Flag-Controlled
- Ilustrasi Masalah
- Bagaimana debug dan mengecek program
- Kesalahan Umum

**5.1 Perulangan dalam Program**

Sebuah perusahaan yang mempekerjakan 7 orang pegawai, ingin mengulang perhitungan gaji kotor dan gaji bersih sebanyak 7 kali, satu untuk masing-masing pegawai.

**Perbandingan Macam-Macam Perulangan**

Macam	Uraian	Struktur Implementasi C
Counting	Kita dapat menentukan jumlah perulangan yang akan kita butuhkan untuk menyelesaikan soal sebelum eksekusi perulangan	While For
Sentinel-controlled	Masukan sejumlah data berapapun panjangnya yang diakhiri dengan nilai tertentu.	While.for
Endfile-controlled	Masukan berupa data tunggal berapapun panjangnya	While,for
Input validation	Masukan interaktif yang diulang sampai nilai tertentu dalam range dimasukkan	Do-while
General Conditional	Pengulangan proses data sampai keadaan tertentu terpenuhi.	While.for

## 5.2 Counting Loop dan pernyataan WHILE

Perulangan yang ditunjukkan di bawah disebut counting loop karena pengulangannya diatur oleh variabel kontrol yang nilainya menyatakan sebuah perhitungan.

Sebuah counting loop akan mengikuti format umum di bawah ini :

*Menentukan nilai variabel kontrol = 0*

*Variabel kontrol < nilai akhir*

...

*Variabel kontrol + 1*

Kita menggunakan counting loop ketika kita bisa menentukan terlebih dahulu berapa perulangan yang kita butuhkan untuk menyelesaikan masalah. Jumlah ini kemudian yang akan menjadi nilai akhir.

### Pernyataan While

Gambar 5.2 menunjukkan sebuah kerangka program yang menghitung dan menampilkan gaji kotor untuk 7 orang pegawai. Isi perulangannya yaitu gabungan pernyataan yang dimulai di baris ke-tiga. Isi perulangannya meminta masukan daftar gaji pegawai-pegawai, menghitungnya, dan menampilkan gaji pegawai tersebut. Setelah tujuh gaji mingguan ditampilkan, program menampilkan pesan " Semua pegawai sudah diproses"

Contoh

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int juml_peg,jam;
```

```
float perjam,gaji;
```

```
Juml_peg=0;
```

====> membuat nilai awal juml\_peg=0

```
while (juml_peg<7)
```

====> syarat : ketika juml\_peg masih <7

```
{
```

```
printf("Jumlah jam kerja= ");
```

```
scanf("%d",&jam);
```

```
printf("Gaji per jam= ");
```

```
scanf("%f",&perjam);
```

```
gaji=jam*perjam;
```

```
printf("Gaji yang Anda terima Rp%6.2f\n",gaji);
```

```
juml_peg=juml_peg+1;
```

====> membuat nilai juml\_peg bertambah

```
}
```

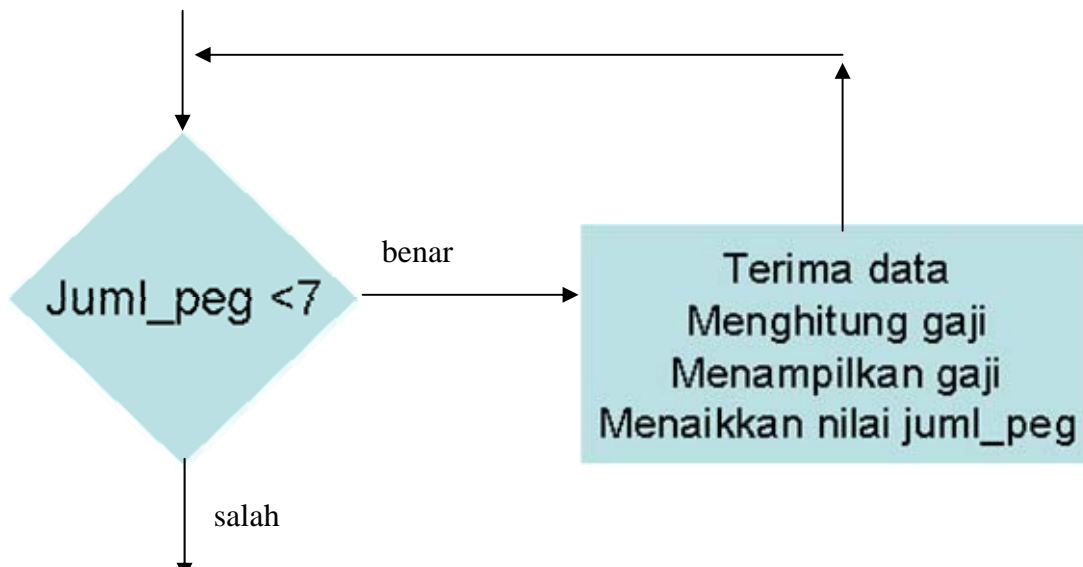
'program kemudian kembali ke line 9 krn

'juml\_peg masih<7

```
printf("\nSemua sudah diproses");
```

```
}
```

flow chart



**Akhir perulangan**

Perbedaan statemen berikut ini :

```
If (juml_peg < 7) { ... }
```

dengan

```
while (juml_peg<7)
```

Di if, fungsi-fungsi setelah syarat yang diberikan akan dikerjakan hanya satu kali. Dalam while, fungsi-fungsi dapat dijalankan lebih dari satu kali.

Variabel `juml_peg` merupakan variabel kontrol dalam program perulangan tersebut

Karena nilainya menentukan apakah fungsi-fungsi di bawahnya masih dijalankan atau tidak.

Sebuah variabel kontrol perulangan harus mengalami 3 tahap berikut ini :

1. Inisialisasi → `juml_peg` ditentukan nilai awalnya yaitu 0
2. Uji → `juml_peg` diuji nilainya sebelum perulangan dimulai
3. Update → `juml_peg` diupdate nilainya (dlm prog ini ditambah satu)

Tanpa tahap update sebuah perulangan akan dijalankan terus menerus oleh komputer. Perulangan ini disebut infinite loop (perulangan tak terhingga).

### 5.3 Menghitung dalam perulangan

Dalam program kali ini, yang dihitung bukan hanya gaji tiap pegawainya namun juga gaji yang telah dibayarkan oleh perusahaan kepada semua karyawan.

```
#include<stdio.h>
```

```
void main()
```

```
{
double totalgaji, jam, perjam, gaji;
int jumlahpeg;
int totalpeg;
printf("Masukkan jumlah pegawai ");
scanf("%d",&totalpeg);
```

```
totalgaji = 0.0;
jumlahpeg=0;
while(jumlahpeg < totalpeg){
printf("Berapa jam kerja Anda = ");
scanf("%lf",&jam);
printf("Berapa gaji perjammu = ");
scanf("%lf",&perjam);
gaji=jam*perjam;
printf("Gajimu Rp%6.2f\n\n",gaji);
totalgaji=totalgaji+gaji;
jumlahpeg=jumlahpeg+1;
}
printf("Semua pegawai sudah selesai diproses datanya\n");
printf("Total gaji yang dibayarkan oleh perusahaan yaitu Rp%8.2f\n",totalgaji);

}
```

Variabel totalgaji merupakan variabel akumulator. Dia mengakumulasikan nilai total gaji dari Proses2. Perulangan sebelumnya. Menginisialisasikan variabel totalgaji=0 adalah Sangat penting sebab bila langkah ini dihilangkan, Berapapun hasilnya nanti, tidak akan dapat dimunculkan dengan benar.

menambahkan nilai gaji pada variabel totalgaji

#### Cara lain menulis Rumus

Cara sederhana	Cara alternatif
Totalpeg=totalpeg+1;	Totalpeg +=1;
Time = time - 1;	Time - = 1;
Totaltime=totaltime+time;	Totaltime += time;
Produk = produk * unit;	Produk *= unit;
N = n* (x+1);	N *= x+1;

**Contoh**

```
do
{
    printf("Masukkan huruf dari A-E");
    scanf("%c",&pilihanhuruf);
}
while(pilihanhuruf < 'A' || pilihanhuruf > 'E');
```

**5.4 Pernyataan For**

C menyediakan pernyataan "For" sebagai bentuk lain untuk pengulangan proses.

Dan mempunyai 3 komponen. Bentuk pernyataan seperti ini:

For (ungkapan 1,ungkapan 2,ungkapan 3) pernyataan

Kegunaan masing masing ungkapan adalah

- Sebagai inisialisasi terhadap variabel pengendali loop
- Sebagai kondisi untuk keluar dari loop
- Sebagai pengatur kenaikan nilai variabel pengendali loop

**5.5 Conditional Loops**

Mempunyai 3 bagian yang mengendalikan perulangan :

1. Inisialisasi
2. Pengecekan terhadap syarat perulangan
3. Perbaruan

**5.6 Desain Perulangan**

Perulangan yang di kontrol oleh nilai tertentu (sentinel)

Sebuah perulangan yang memproses data hingga nilai tertentu (sentinel) di masukkan memiliki bentuk

- Ambil dari baris data
- Selama nilai sentinel belum di jumpai
- Proses baris data
- Ambil baris data yang lain

Untuk program agar mudah di baca, biasanya nama dari sentinel di definisikan dengan konstanta macro

Contoh Sentinel loop

1. inisialisasi nilai sum ke 0
2. ambil skore pertama
3. selama skore bukan sentinel
4. tambahkan skore ke sum
5. ambil skore selanjutnya

Contoh sentinel yang salah

1. inisialisasi nilai sum ke 0
- 2. selama skore bukan sentinel**
3. ambil skore
4. tambahkan skore ke sum

### **Memakai statemen for untuk menerapkan sentinel loop**

Karena pernyataan For menggabungkan unsur inisial, test dan update dalam satu tempat, beberapa programmers lebih menggunakan itu sebagai perlengkapan sentinel controlled loops.

### **Endfile - Kontrol loop dalam akhir file**

Dibawah ini sebuah pseudocode untuk suatu endfile-controlled loop

1. ambil nilai data pertama dan simpan status inputnya
2. selama status input tidak mengindikasi dari endfile
3. proses nilai data
4. ambil nilai data selanjutnya dan simpan status input

### **Loop yang tak terbatas (infinite loop) pada data yang salah**

Loop yang tak terbatas di sini mempunyai arti bahwa apabila menemui kesalahan data fungsi Scanf dan Fscanf dapat dengan cepat membuat loop yang tak terbatas pada saat pernyataan itu juga.

## **5.7 Nested Loops (loop di dalam loop)**

Loop dapat di dalam loop karena struktur pengendali yang lain. Loop di dalam loop terdiri dari loop luar dan satu atau lebih loop di dalam.

Tiap waktu loop luar akan di ulang atau mengalami perulangan. Dan loop dalam akan mengalami perulangan masukkan, kemudian semuanya akan di evaluasi dan semua akan di tampilkan bersama.

## 5.8 Pernyataan do-while dan Perulangan Flag-Controlled

For dan while sama-sama mengecek syarat perulangan sebelum eksekusi fungsi pertama dalam isi perulangan. Hal ini sangat penting untuk menghindari terjadinya perulangan ketika tidak ada data yang dimasukkan atau data yang dimasukkan tidak termasuk dalam range yang diminta.

Ada beberapa situasi dimana melibatkan input dari pengguna.

1. Ambil nilai data
2. Jika nilai data di luar range, kembali ke step pertama.

### Perulangan Flag-Controlled untuk Validasi Input

Syarat perulangan bisa menjadi sangat kompleks. Pada beberapa kasus syaratnya bisa disederhanakan dengan menggunakan flag. Flag adalah sebuah tipe variabel integer yang digunakan untuk menyatakan apakah sebuah 'kejadian' telah terjadi. Sebuah flag mempunyai 2 nilai : 1 (benar) dan 0 (salah).

## 5.9 Ilustrasi Masalah

### Studi kasus

#### **Menghitung Tingkat Radiasi**

### Masalah

Dalam suatu laboratorium rahasia, beberapa yttrium-90 telah menyebar ke ruang analisis komputer. Penyebarannya akan dengan cepat mencapai 150 milirem radiasi perhari. Waktu paruhnya sekitar 3 hari. Oleh karena itu, tingkat radiasi sekarang tiga kali lebih kecil dibanding yang tahun kemarin. Para Laboran Ingin mengetahui kira-kira berapa hari lagi ruangan itu aman untuk dimasuki dimana tingkat radiasi turun hingga 0.466 milirem perhari. Mereka ingin sebuah tabel yang menampilkan tingkat radiasi setiap 3 hari dengan pesan "Tidak Aman".

Tabel harus berhenti ketika tingkat radiasi tepat setingkat di atas tingkat radiasi 1/10 level aman. Tabel itu juga harus menunjukkan kapan hari pertama ruangan itu aman dimasuki.

Analisa

Setiap baris dari tabel akan menunjukkan tanggal dan tingkat radiasi Setiap harinya. Baris terakhir dari tabel harus menunjukkan hari dimana laboratorium sudah aman untuk dimasuki.

Data yang dibutuhkan :

Konstanta

RAD\_AMAN 0.466

FAKTOR\_AMAN 10

Input

double init\_radiasi

Output

int day

double tk\_radiasi

Algoritma

1. Menghitung tingkat berhentinya radiasi =tingkat aman dibagi faktor aman.
2. Meminta pengguna memasukkan tingkat radiasi awal
3. Menghitung dan menampilkan tanggal dan tingkat radoasi 3 hari sekali. Mengindikasikan setiap tingkat radiasi (aman/tidak aman)
4. Menampilkan hari dimana pengguna dapat memasuki laboratorium.

```
#include <stdio.h>
```

```
#define SAFE_RAD = 0.466
```

```
#define SAFETY_FACT 10
```

```
main(void)
```

```
{
    int day;
    double init_radiation, min_radiation;
    min_radiation=SAFE_RAD/SAFETY_FACT;
    printf("Masukkan tingkat radiasi (dalam milirem)>");
    scanf("%lf",&init_radiation);
    day=rad_table(init_radiation,min_radiation);
    printf("\nYou can enter the room on day %d.\n",day);
    return(0);
}
```

```

int rad_table(double init_radiation,double min_radiation)
{
    int day;
    double radiation_lev;
    day=0;
    printf("\n Day Radiation Status\n (milirems)\n);
    for(radiation_lev = init_radiation;
        radiation_lev > min_radiation;
        radiation_lev /= 2.0){
        if(radiation_lev> SAFE_RAD)
            printf("%3d%5c%9.4f Unsafe\n. day, ' ', radiation_lev);
        else
            printf("%3d%5c%9.4f Safe\n", day, ' ', radiation_lev);
        day +=3;
    }
    return (day);
}

```

### Bagaimana Debug dan Mengecek Program

Langkah pertama mendeteksi kesalahan sebuah program yaitu dengan memeriksa keluaran yang dihasilkan untuk menentukan bagian mana dari program yang mengandung kesalahan. Kemudian fokuskan perhatian pada bagian tersebut. Lalu ada 2 hal yang bisa dilakukan setelah langkah tersebut, yaitu :

1. Menggunakan Debugger Program
2. Debug tanpa menggunakan debugger

### Melakukan Debug tanpa Debugger

Jika kita tidak memungkinkan menggunakan debugger, kita harus menerapkan diagnosa khusus pada *printf* dalam titik-titik penting dalam program. Tampilkan saja nilai variabel dalam bagian-bagian tertentu dari program, terutama sebelum dan sesudah sebuah pernyataan algoritma tertentu.

### Kesalahan umum

- Perulangan yang terjadi satu kali lebih banyak atau lebih sedikit dari yang diharapkan
- Lupa menulis tanda titik koma
- Tidak menulis kurung kurawal buka setelah *while* jika ada lebih dari satu pernyataan setelah *while*
- Jumlah kurung buka dan tutup tidak sama