

MENGENDALIKAN PROSES PROGRAM

Proses yang ada pada suatu program tidak hanya proses yang berurutan. Selain proses berurutan, terdapat juga proses percabangan, proses pengulangan, dan proses lompatan.

Bahasa C menyediakan beberapa statement yang dapat digunakan untuk mengendalikan proses dari suatu program

PROSES PERCABANGAN

Proses percabangan akan menyebabkan satu atau beberapa perintah diproses atau sebaliknya satu atau beberapa perintah tidak diproses, tergantung dari hasil kondisi yang diseleksi.

1. Statement *if*

Statement *if* adalah salah satu statement yang digunakan untuk penyeleksian kondisi. Statement *if* ini mempunyai beberapa variasi bentuk, yaitu :

a. Bentuk *if Tunggal*

Bentuk umum :

**if (kondisi)
statement ;**

Bentuk ini menunjukkan jika kondisi yang diseleksi benar (bernilai logika 1) maka statement yang mengikutinya akan diproses. Sebaliknya jika kondisi yang diseleksi salah (bernilai logika 0) maka statement berikutnya tidak akan diproses.

Bentuk **if** harus diikuti dengan kondisi yang diapit oleh tanda kurung biasa, sedangkan statement dapat berupa statement tunggal atau statement jamak (compound statement). Statement jamak harus diapit oleh tanda kurung kurawal.

Contoh :

```
if (x<0)
```

```
x = -x;
```

Kondisi yang digunakan sebagai pengujian dapat berupa ekspresi yang lebih kompleks, dengan syarat ekspresi tersebut menghasilkan sebuah nilai tunggal.

Contoh :

```
if (scanf("%d", &n) !=1)
```

```
printf("jenis input yang diberikan salah \n");
```

```
/* Contoh proram menggunakan statement if */
```

```

main()
{
int usia;
clrscr();
printf("Berapakah usia anda? : ");
scanf("%d", &usia);
if(usia>=40)
{
printf("\nWah..... anda sudah cukup tua");
printf("\nJangan makan makanan yang mengandung kolestrol tinggi");
printf("\nTidak baik untuk kesehatan");
}
}

```

Jika program dijalankan, outputnya adalah :

Berapakah usia anda?

20

Perhatikan contoh program di atas. Statement yang mengikuti **if** adalah statement jamak, maka diapit oleh tanda kurung kurawal dan setiap satu baris statement diakhiri dengan tanda titik koma (;).

b. Bentuk if - else

Bentuk umum :

```

if (kondisi)
statement;
else
statement;

```

Statement setelah **if** atau setelah **else** dapat berupa statement kosong, statement tunggal, atau statement jamak. Apabila statement yang mengikuti **if** atau **else** berupa statement jamak maka statement tersebut harus diapit oleh tanda kurung kurawal dan setiap baris statement diakhiri tanda titik koma. Bentuk diatas akan memproses statement setelah **if** jika kondisi yang diseleksi benar dan jika salah maka yang diproses adalah statement setelah **else**.

/* Contoh proram menggunakan statement if-else dengan kondisi jamak*/

```

main()
{
char jawab;
printf("\nAnda mau ikut? ");
jawab=getche();

```

```

if(jawab=='Y' || jawab=='y')
printf("\nCepat ganti pakaian");
else
printf("\nTolong jaga rumah");
}

```

Jika program dijalankan, outputnya adalah :
Anda mau ikut?

y

Cepat ganti pakaian

- Statement **if-else** dapat ditulis dengan dua buah statement **if** tunggal.

c. Bentuk if - else - if - ... else

Bentuk umum :

```

if (kondisi1)
statement;
else if (kondisi2)
statement;
else if (kondisi3)
statement;
:
:
else
statement;

```

Penyeleksian akan dilakukan mulai dari kondisi yang pertama (kondisi1), jika benar maka statement yang mengikutinya akan diproses dan penyeleksian dihentikan (kondisi yang lain tidak akan diseleksi). Jika kondisi yang pertama (kondisi1) salah maka penyeleksian dilakukan pada kondisi berikutnya (kondisi2). Jika kondisi2 benar maka statement yang mengikutinya akan diproses dan penyeleksian dihentikan. Apabila salah maka akan diseleksi kondisi berikutnya.

/ Contoh proram menggunakan statement if-else-if */*

```

main()
{
float angka;
char jawab;
clrscr();
printf("\nMasukkan nilai angka (0-100) ");

```

```

scanf("%f",&angka);
if(angka<50)
huruf='E';
else if(angka>=50 && angka<60)
huruf='D';
else if(angka>=60 && angka<70)
huruf='C';
else if(angka>=70 && angka<85)
huruf='B';
else
huruf='A';
printf("\nNilai Huruf : %c", huruf);
}

```

Jika program dijalankan, outputnya adalah :

Masukkan nilai angka (0-100)

60

Nilai Huruf : C

d. Bentuk if Bersarang (nested if)

Bentuk umum :

```

if (kondisi1)
if (kondisi2)
:
if (kondisi-n)
statement;
else
statement;
:
else
statement;
else
statement;

```

Kondisi yang diseleksi pertama kali adalah kondisi yang paling luar (kondisi1). Jika kondisi1 salah maka yang diproses adalah statement setelah **else** yang terluar (pasangan dari **if** yang bersangkutan). Jika **else** tidak ditulis maka penyeleksian dihentikan. Jika kondisi1 benar maka penyeleksian dilanjutkan pada kondisi2, jika kondisi2 salah maka yang diproses adalah statement setelah **else** pasangan dari kondisi2. Jika else tidak ditulis maka penyeleksian dihentikan.

Demikian seterusnya proses penyeleksian tetap dilanjutkan apabila kondisi sebelumnya bernilai benar.

Bentuk dari statement nested-if ini merupakan bentuk yang cukup membingungkan, terutama apabila jumlah **else** tidak sama dengan jumlah **if**.

Perhatikan penggalan program berikut ini.

```
if(n>0)
    if(n%2 == 0)
        printf("positif dan genap\n");
    else
        printf("positif dan ganjil");
```

else pada contoh di atas merupakan pasangan dari **if** yang terdekat, yaitu **if(n%2==0)** Jika **else** dimaksudkan untuk pasangan **if(n>0)**, maka dapat digunakan tanda kurung kurawal, seperti pada contoh berikut.

```
if(n>0)
{
    if(n%2 == 0)
        printf("positif dan genap\n");
    }
    else
        printf("tidak positif");
```

Kondisi dari statement **if** dapat berupa variabel atau kondisi jamak (dengan menggunakan operator logika).

OPERATOR ?

Operator ? dapat digunakan sebagai pengganti bentuk statement if - else. Operator ini juga disebut dengan nama ternary operator , karena operator ini menggunakan tiga buah ungkapan (ternary expression).

Bentuk umum :

(ungkapan1) ? ungkapan2 : ungkapan3

Jika ungkapan1 benar, maka ungkapan2 yang akan digunakan sebagai hasil dari operator ini. Sebaliknya jika ungkapan1 bernilai salah maka ungkapan3 yang akan digunakan sebagai hasilnya. Ungkapan dapat berupa suatu fungsi.

Contoh :

```
#include <stdio.h>
main()
{
```

```

int x;
printf("Masukkan nilai suatu integer ? ");
scanf("%d",&x);
(x % 2 == 0) ? printf("Nilai genap\n") : printf("Nilai ganjil\n");
}

```

Jika program dijalankan :
 Masukkan nilai suatu integer ? 10
 Nilai genap

2. Statement *switch*

Statement switch merupakan salah satu statement yang digunakan untuk penyeleksian kondisi. Statement switch dapat berbentuk statement switch tunggal atau statement switch bersarang (nested switch).

a. Statement switch Tunggal

Bentuk umum :

```

switch(kondisi)
{
case konstanta1:
statement-statement;
break;
case konstanta2:
statement-statement;
break;
:
default:
statement-statement;
}

```

Statement **switch** akan menyeleksi kondisi yang diberikan kemudian membandingkan hasilnya dengan konstanta-konstanta yang ada di **case**. Apabila hasil dari kondisi sama dengan nilai dari konstanta tertentu, misalnya konstanta2, maka yang akan diproses adalah statement yang ada pada case konstanta2 sampai ditemui statement break. Jika semua konstanta yang dibandingkan tidak ada yang sama, maka yang diproses adalah statement yang berada di **default**. Bentuk default ini sifatnya optional.

```

/* Contoh proram menggunakan statement switch*/
main()
{
int bil;

```

```

printf("\nKetikkan bilangan bulat antara 1 sampai dengan 3 ");
scanf("%d",&bil);
switch(bil)
{
    case 1:
printf("\nAnda mengetikkan satu");
break;
    case 2:
printf("\nAnda mengetikkan dua");
break;
    case 3:
printf("\nAnda mengetikkan tiga");
break;
    default :
printf("\nAnda mengetikkan bilangan yang salah");
}
}
}

```

b. Statement switch bersarang (nested switch)

Statement switch bersarang adalah statement switch yang berada di dalam statement switch yang lainnya.

PROSES PENGULANGAN

Proses pengulangan menyebabkan satu atau lebih statement diproses secara berulang-ulang. Misalnya akan dibuat tampilan "Universitas Gunadarma" sebanyak 10 kali. Tampilan ini dapat dibuat dengan menggunakan statement printf() sebanyak 10 kali, tetapi hal tersebut tidak efisien.

Proses pengulangan ini dapat dilakukan dengan menggunakan statement pengulangan yaitu for, while, do-while. Dengan statement pengulangan, statement yang prosesnya akan diulang-ulang cukup dituliskan sekali saja.

1. Statement *for*

Bentuk umum :

for(awal; akhir; peningkatan) statement;

awal adalah suatu statement yang memberikan nilai awal bagi suatu variabel. Pemberian nilai awal hanya dilakukan sekali saja yaitu pada waktu proses pengulangan mulai dilaksanakan. Pemberian nilai awal tidak hanya terbatas untuk satu variabel saja, melainkan dapat pula digunakan untuk beberapa variabel sekaligus. Antara satu variabel dengan variabel lainnya dipisahkan dengan tanda koma (,).

akhir adalah suatu statement yang menunjukkan suatu kondisi yang harus dipenuhi agar pengulangan masih dapat terus dilaksanakan. Akhir dapat berupa ekspresi relasional ataupun ekspresi logika.

peningkatan adalah suatu statement yang merubah nilai-nilai variabel pengontrol pengulangan setiap saat pengulangan dilakukan.

statement setelah for dapat berupa statement tunggal atau statement jamak. Untuk statement jamak harus diapit dengan tanda kurung kurawal ({ }).

a. Pengulangan Positif

Pengulangan positif merupakan pengulangan yang peningkatannya positif untuk variabel pengontrol pengulangan.

Contoh :

```
#include<stdio.h>
main()
{
int I;
for(I=2;I<=5;I++) printf("%d",I);
}
```

Tentukan output dari program di atas!

b. Pengulangan Negatif

Pengulangan negatif merupakan pengulangan dengan penurunan nilai untuk variabel pengontrol pengulangannya.

Contoh :

```
#include <stdio.h>
main()
{
int I;
for(I=5; I>1: I--) printf("%d",I);
}
```

Tentukan output dari program di atas!

c. Pengulangan Dengan Argumen yang Tidak Lengkap

Argumen yang digunakan dalam statement for tidak harus selalu lengkap.

Apabila bagian awal dihilangkan, maka bentuknya seperti :

```
for(; akhir; peningkatan)
```

Bagian akhir dihilangkan, seperti :

```
for(awal; ; peningkatan)
```

Bagian peningkatannya dihilangkan seperti :

for(awal; akhir;)

Meskipun argumen yang digunakan boleh tidak lengkap, namun **titik koma** pemisah argumennya harus ditulis dengan lengkap, yaitu dua tanda titik koma.

d. Pengulangan Tak Berhingga

Bila ketiga argumen dari statement for tidak digunakan, maka akan terjadi pengulangan yang tidak berhingga.

for(; ;)

Untuk menghentikan pengulangan tak berhingga tersebut digunakan statement **break**.

e. Nested for

Di dalam statement for boleh terdapat statement for yang lain. Bentuk seperti ini disebut nested for.

2. Statement while

Bentuk umum :

while(kondisi)statement;

Statement dapat berupa statement kosong, statement tunggal, ataupun statement jamak yang diproses berulang-ulang. Proses pengulangan masih akan dilakukan jika kondisi yang diseleksi di statemet **while** masih bernilai benar dan pengulangan dihentikan jika kondisinya sudah bernilai salah.

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

```
/* Mencetak bilangan bulat dari 1 - 10 beserta totalnya dengan statement
```

```
while*/
```

```
main()
```

```
{
```

```
int bil=1, total=0;
```

```
clrscr();
```

```
while(bil<=10)
```

```
{
```

```
total+=bil;
```

```
printf("\n%10d %10d", bil, total);
```

```
bil++;
```

```
}
```

```
}
```

Output program di atas adalah :

```
1    1
2    3
```

```
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
```

3. Statement *do-while*

Bentuk umum :

do statement **while(kondisi);**

Proses pengulangan masih dilakukan jika kondisi yang diseleksi di while masih bernilai benar dan pengulangan dihentikan jika kondisinya bernilai salah.

Perbedaan utama antara statement do-while dan statement while adalah letak dari kondisi yang akan diseleksi. Untuk statement while, kondisi yang diseleksi terletak di awal pengulangan, sehingga sebelum masuk ke dalam lingkup pengulangan, kondisinya harus benar. Sedangkan pada statement do-while, kondisi yang diseleksi terletak di akhir. Ini berarti paling sedikit sebuah pengulangan dilakukan oleh statement do-while.

```
#include<stdio.h>
/* Mencetak bilangan bulat dari 1 - 10 beserta totalnya dengan statement
do-while */
main()
{
int bil=1, total=0;
clrscr();
do
{
total+=bil;
printf("\n%10d %10d", bil, total);
bil++;
}
while(bil<=10);
}
```

Output program di atas adalah :

```
1 1
```

```
2 3
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
```

4. Statement *break* dan *continue*

Statement **break** akan menghentikan loop (pengulangan) dan melanjutkan ke perintah-perintah berikutnya.

```
/* Contoh program menggunakan break */
```

```
main()
{
int i;
clrscr();
for(i=1;i<=10;i++)
{
if(i==6)
break;
printf("%5d",i);
}
printf("\nAkhir pengulangan");
}
```

Output dari program ini :

```
1 2 3 4 5
```

Akhir pengulangan

Statement **continue** menyebabkan proses pengulangan kembali ke awal pengulangan dengan mengabaikan statement-statement setelah statement continue.

```
/* Contoh program menggunakan continue*/
```

```
main()
{
int i;
clrscr();
for(i=1;i<=10;i++)
{
```

```

if(i==6)
continue;
printf("%5d",i);
}
printf("\nAkhir pengulangan");
}

```

Output dari program ini :

1 2 3 4 5 6 7 8 9 10

Akhir pengulangan

PROSES LOMPATAN

1. Statement *goto*

Statement goto dapat digunakan untuk melompat dari suatu proses ke bagian proses yang lainnya di dalam program.

Bentuk umum :

goto label;

Nama label harus ditulis dengan diakhiri tanda titik koma (;).

```

#include<stdio.h>
main()
{
float A,B;
printf("Masukkan nilai A : "); scanf("%f",&A);
printf("Masukkan nilai B : "); scanf("%f",&B);
if(B==0) goto Tak_Berhingga;
printf("%f dibagi dengan %f sama dengan %f\n",A,B,A/B);
goto Selesai;
Tak_Berhingga:
printf("%f dibagi dengan 0 nilainya tak berhingga",A);
Selesai:
}

```