BAB III
PENGAMBILAN KEPUTUSAN

Tujuan :

1. Menjelaskan tentang operator kondisi (operator relasi dan logika)
2. Menjelaskan penggunaan pernyataan if
3. Menjelaskan penggunaan pernyataan if-else
4. Menjelaskan penggunaan pernyataan if dalam if
5. Menjelaskan penggunaan pernyataan else-if
6. Menjelaskan penggunaan pernyataan switch

3.1 Operator Kondisi

Banyak persoalan yang diperlukan untuk membuat keputusan. Contoh yang sederhana berupa cara mengatur agar komputer bisa menyimpulkan bahwa suatu bilangan merupakan bilangan genap atau bilangan ganjil. Untuk keperluan pengambilan keputusan semacam itu, C menyediakan beberapa jenis pernyataan, berupa

- Pernyataan *if*
- Pernyataan *if-else*, dan
- Pernyataan *switch*

Pernyataan-pernyataan tersebut memerlukan suatu kondisi, sebagai basis dalam pengambilan keputusan. Kondisi umum yang dipakai berupa keadaan benar dan salah. Oleh karena itu pembahasan pada bab ini akan diawali dengan pengenalan operator yang membentuk kondisi benar dan salah.

Operator yang digunakan untuk menghasilkan kondisi benar dan salah, bisa berupa operator relasi dan bisa juga berupa operator logika. Berikut ini dibahas masing-masing jenis operator serta tabel prioritas masing-masing operator.

3.1.1. Operator Relasi

Operator relasi biasa dipakai untuk membandingkan dua buah nilai. Hasil pembandingan berupa keadaan benar atau salah. Keseluruhan operator relasi pada C ditunjukkan pada Tabel 3-1.

Tabel 3-1. Operasi relasi

Operator	Makna
>	Lebih dari
>=	Lebih dari atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

Khususnya untuk operator relasi sama dengan (==) harap dibedakan dengan operator (=) yang merupakan operator penugasan *(assignment)*. Contoh:

Pembandingan		**Hasil**
`1 > 2`	⟶	Salah
`1 < 2`	⟶	Benar
`A == 1`	⟶	Benar, jika A bernilai `1`
	↘	Salah, jika A tidak bernilai 1
`'A' < 'B'`	⟶	Benar, karena kode ASCII untuk karakter ‘A’ kurang dari kode ASCII untuk karakter ‘B’ [*)]
`kar == 'Y'`	⟶	Benar, jika kar berisi 'Y'
	↘	Salah, jika kar tidak berisi 'Y'

[*)] Dalam daftar ASCII standar, kode untuk karakter ‘A’ = 65 sedangkan karakter ‘B’ = 66, ‘C’ = 67, ‘D’ = 68 dan seterusnya sampai dengan karakter ‘Z’ = 90.

3.1.2. Operator Logika.

Operator logika biasa dipakai untuk menghubungkan ekspresi relasi. Keseluruhan operator logika ditunjukkan pada tabel 3-2.

Tabel 3-2. Operator logika

Operator	Makna
&&	dan (AND)
||	atau (OR)
!	tidak (NOT)

Bentuk pemakaian operator && dan || adalah

operand1 operator operand2

Baik **operand1** maupun **operand2** dapat berupa ekspresi relasi ataupun ekspresi logika. Hasil ekspresi bias bernilai benar atau salah. Pada C nilai hasil dari sebuah ekspresi relasi atau ekspresi logika jika dinyatakan dengan angka adalah :

- Salah → nilai = 0
- Benar → nilai != 0 (misalnya nilai = 1)

Tabel 3-3 memberikan penjelasan hasil operasi ekspresi logika yang menggunakan operator && maupun || untuk berbagai kemungkinan keadaan operand-nya.

Tabel 3-3. Kemungkinan pada operasi logika && dan ||

Operand1	Operand2	Hasil	
		||	&&
Salah	Salah	0	0
Salah	Benar	1	0
Benar	Salah	1	0
Benar	Benar	1	1

Tampak bahwa operator **atau** (||) menghasilkan nilai 1 jika ada operand yang benar. Hasil berupa 0 jika semua operand adalah salah. Adapun operator logika **dan** (&&) memberikan hasil 1 hanya jika kedua operand adalah benar.

Beberapa contoh ekspresi logika di antaranya :

- `(kar > 'A') && (kar < 'Z')`

  Hasil operasi logika && adalah benar hanya jika `kar > 'A'` dan `kar < 'Z'` (dalam hal ini yang diperbandingkan adalah kode ASCII dari karakter tsb).
- `(pilihan == 'Y') || (pilihan == 'y')`

  Hasil operasi logika|| adalah benar jika pilihan berupa `'Y'` atau `'y'`

Sedangkan bentuk pemakaian operator logika ! adalah :

!operand

dengan **operand** dapat berupa ekspresi logika ataupun ekspresi relasi.

Hasil operasi ! bernilai :

- 1 jika**operand** bernilai salah
- 0 jika**operand** bernilai benar

Perhatikan contoh potongan program di bawah ini :

```
if (!sudah_benar)
 printf("Masukan Anda salah!\n");
```

Pada contoh potongan program di atas, dilakukan pengecekan kondisi terhadap nilai dari variabel **sudah_benar**. Jika variabel **sudah_benar** bernilai 0, maka kondisi **!sudah_benar** akan bernilai benar *(true)* sehingga instruksi :

```
printf("Masukan Anda salah!\n");
```

akan diproses. Penjelasan lebih rinci tentang pengecekan kondisi dengan pernyataan *if* dibahas pada sub bab 3.2.

3.1.3 Prioritas Operator Logika dan Relasi

Tabel berikut ini memberikan penjelasan singkat mengenai prioritas di antara berbagai operator logika dan operator relasi.

Tabel 3-4 Prioritas operator logika dan relasi

Tertinggi :	!
	> >= < <=
	= = !=
	&&
Terendah:	||

Berdasarkan prioritas yang ditunjukkan pada tabel 3-4, maka ekspresi seperti

```
(kar > 'A') && (kar < 'Z')
```

sama saja kalau ditulis menjadi

```
kar > 'A' && kar < 'Z'
```

Hanya saja penulisan dengan menggunakan tanda kurung akan lebih memberikan kejelasan.

3.2 Pernyataan *if*

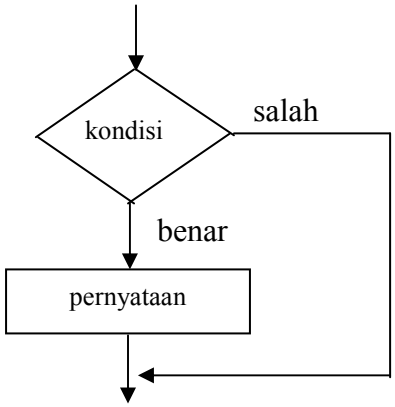
Pernyataan *if* mempunyai bentuk umum :

```
if (kondisi )
   pernyataan;
```

Bentuk ini menyatakan :

- jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan yang mengikutinya tidak akan diproses.

Mengenai kodisi harus ditulis diantara tanda kurung, sedangkan pernyataan dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Diagram alir dapat dilihat seperti gambar 3.1



Gambar 3.1. Diagram alir *if*

Contoh penggunaan pernyataan *if* adalah untuk menentukan besarnya potongan harga yang diterima oleh seorang pembeli, berdasarkan kriteria :

- tidak ada potongan harga jika total pembelian kurang dari Rp. 100.000 (dalam hal ini potongan harga diinisialisasi dengan nol).
- bila total pembelian lebih dari atau sama dengan Rp. 100.000, potongan harga yang diterima dirubah menjadi sebesar 5% dari total pembelian.

```
/* File program : discount.c
Contoh penggunaan if untuk menghitung nilai discount */

#include <stdio.h>
```

```
main()
{
   double total_pembelian, discount = 0;
   /* discount diinisialisasi dengan nilai 0 */

   printf("Total pembelian   = Rp ");
   scanf("%lf", &total_pembelian);

   if(total_pembelian >= 100.000)
      discount = 0.05 * total_pembelian;
   printf("Besarnya discount = Rp %.2lf\n", discount);
}
```

Contoh eksekusi :

```
Total pembelian   = Rp 200000
Besarnya discount = Rp 10000.00
```

Untuk pernyataan ***if*** yang diikuti dengan pernyataan majemuk, bentuknya adalah sebagai berikut :

```
if (kondisi)
{                   /* tanda awal pernyataan majemuk*/
    pernyataan-1;
    pernyataan-2;
    .
    .
    .
    pernyataan-n;
}                   /* tanda akhir pernyataan majemuk */
```

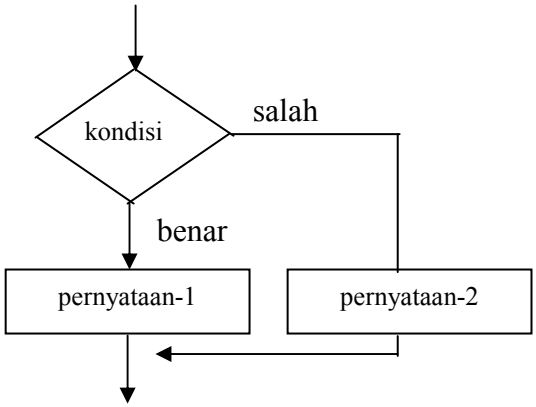
Pernyataan-pernyataan yang berada dalam tanda kurung { dan } akan dijalankan hanya bila kondisi *if* bernilai benar.

3.3. Pernyataan *if-else*

Pernyataan ***if-else*** memiliki bentuk :

```
if (kondisi)
   pernyataan-1;
else
   pernyataan-2;
```

Diagram alir dapat dilihat seperti gambar 3.2.



Gambar 3.2. Diagram alir *if-else*

Arti dari pernyataan *if-else* :

- Jika kondisi benar, maka **pernyataan-1** dijalankan.
- Sedangkan bila kondisi bernilai salah, maka **pernyataaan-2** yang dijalankan.

Masing-masing **pernyataan-1** dan **pernyataan-2** dapat berupa sebuah pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong.

Contoh penggunaan pernyataan *if-else* adalah untuk menyeleksi nilai suatu bilangan pembagi. Jika nilai bilangan pembagi adalah nol, maka hasil pembagian dengan nilai nol akan mendapatkan hasil tak berhingga. Jika ditemui nilai pembaginya nol, maka proses pembagian tidak akan dilakukan.

```
/* File program : bagi.c
Pemakaian if-else untuk menyeleksi bilangan pembagi */

#include <stdio.h>

main()
{
   float a, b;

   printf("Masukkan nilai a : ");
   scanf("%f", &a);
   printf("Masukkan nilai b : ");
   scanf("%f", &b);

   if (b == 0)
     printf("\n%g dibagi dengan nol = TAK BERHINGGA\n", a);
   else
```

```
      printf("\n%g dibagi dengan %g  = %g\n", a, b, a/b);
}
```

Contoh eksekusi :

```
Masukkan nilai a : 5
Masukkan nilai b : 0
5 dibagi dengan nol = TAK BERHINGGA
```

3.4. Pernyataan *if* di dalam *if*

Di dalam suatu pernyataan *if* (atau *if-else*) bisa saja terdapat pernyataan *if* (atau *if-else)* yang lain. Bentuk seperti ini dinamakan sebagai *nested if*. Secara umum, bentuk dari pernyataan ini adalah sebagai berikut :

```
if (kondisi-1)
    if (kondisi-2)
    .
    .
            if(kondisi-n)
                pernyataan;
            else
                pernyataan;
                .
                .
    else
        pernyataan;
else
    pernyataan;
```

- Kondisi yang akan diseleksi pertama kali adalah kondisi yang terluar (**kondisi-1**). Jika **kondisi-1** bernilai salah, maka statemen *else* yang terluar (pasangan *if* yang bersangkutan) yang akan diproses. Jika *else* (pasangannya tsb) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Jika **kondisi-1** bernilai benar, maka kondisi berikutnya yang lebih dalam (**kondisi-2**) akan diseleksi. Jika **kondisi-2** bernilai salah, maka statemen *else* pasangan dari *if* yang bersangkutan yang akan diproses. Jika *else* (untuk **kondisi-2**) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Dengan cara yang sama, penyeleksian kondisi akan dilakukan sampai dengan **kondisi-n,** jika kondisi-kondisi sebelumnya bernilai benar.

```
/* File program : diskriminan1.c
Program untuk menghitung diskriminan dan akar-akar persamaan
kuadrat menggunakan if bersarang */

#include <stdio.h>
#include <math.h>

main()
{
   float a, b, c, d = 0;
   double x1, x2, imaginair;

   printf("MENCARI AKAR-AKAR PERSAMAAN KUADRAT a+bx+c=0\n");
   printf("\nMasukkan nilai a : ");
   scanf("%f", &a);
   printf("Masukkan nilai b : ");
   scanf("%f", &b);
   printf("Masukkan nilai c : ");
   scanf("%f", &c);

   d = b*b-4*a*c;  /* menghitung diskriminan */

   if (d >= 0)
      if (d == 0)
      {
         x1 = -b / (2 * a);
         printf("\nDua akar real kembar yaitu : \n");
         printf("x1 = x2 = %g\n", x1);
      }
      else
      {
         x1 = (-b + sqrt(d))/(2*a);
         x2 = (-b - sqrt(d))/(2*a);
         printf("\nDua akar real berlainan yaitu :\n");
         printf("x1 = %g\n", x1);
         printf("x2 = %g\n", x2);
      }
   else
   {
      imaginair = (sqrt(-d)/(2*a));
      x1 = -b/(2*a);
      printf("\nDua akar imaginair berlainan yaitu : \n");
      printf("x1 = %g + %gi\n", x1, imaginair);
      printf("x2 = %g - %gi\n", x1, imaginair);
   }
}
```

Contoh eksekusi :

```
MENCARI AKAR-AKAR PERSAMAAN KUADRAT a+bx+c=0
Masukkan nilai a : 3
Masukkan nilai b : 6
Masukkan nilai c : 2

Dua akar real berlainan yaitu :
X1 = -0.42265
X2 = -1.57735
```

3.5 Pernyataan else-*if*

Contoh implementasi *nested if* ini misalnya pembuatan sebuah program kalkulator sederhana. User memberikan masukan dengan format :

operand1 operator operand2

Jenis operasi yang dikenakan bergantung pada jenis **operator** yang dimasukkan oleh user. Oleh karena itu program akan mengecek apakah **operator** berupa tanda '*', '/', '+', ataukah tanda '-' .

- Jika operator berupa tanda '*' maka **operand1** akan dikalikan dengan **operand2**.
- Jika operator berupa tanda '/' maka **operand1** akan dibagi dengan **operand2**.
- Jika operator berupa tanda '+' maka **operand1** akan dijumlahkan dengan **operand2**.
- Jika operator berupa tanda '-' maka **operand1** akan dikurangi dengan **operand2**.
- Kalau operator yang dimasukkan bukan merupakan salah satu dari jenis operator di atas, maka ekspresi tersebut tidak akan diproses, dan user akan mendapatkan pesan berupa : "Invalid operator !"

```
/* File program : kalkulator1.c
Contoh penggunaan else if untuk mengimplementasikan program
kalkulator sederhana */

#include <stdio.h>

main()
{
   int  valid_operator = 1;
   /* valid_operator diinisialisasi dengan logika 1 */
   char  operator;
   float  number1, number2, result;
```

```
    printf("Masukkan 2 buah bilangan dan sebuah operator\n");
    printf("dengan format : number1 operator number2\n\n");
    scanf("%f %c %f", &number1, &operator, &number2);

    if(operator == '*')
        result = number1 * number2;
    else if(operator == '/')
        result = number1 / number2;
    else if(operator == '+')
        result = number1 + number2;
    else if(operator == '-')
        result = number1 - number2;
    else
        valid_operator = 0;

    if(valid_operator)
        printf("\n%g %c %g is %g\n", number1, operator,
               number2, result );
    else
        printf("Invalid operator!\n");
}
```

Contoh eksekusi :

```
Masukkan 2 buah bilangan dan sebuah operator
dengan format : number1 operator number2

23.2 + 12
23.2 + 12 is 35.2
```

3.6 Pernyataan *switch*

Pernyataan ***switch*** merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah alternatif, misalnya untuk menggantikan pernyataan ***if*** bertingkat.

Bentuk umum pernyataan ***switch*** adalah :

```
switch (ekspresi)
        {
            case konstanta-1:
                pernyataan-1;
                ......
                break;
            case konstanta-2:
                .
                .
                .
            case konstanta-n:
                pernyataan-n;
                .......
                break;
            default:
                .......
                .......
                break;
    }
```

dengan **ekspresi** dapat berupa ekspresi bertipe integer atau bertipe karakter. Demikian juga **konstanta-1**, **konstanta-2**, …, **konstanta-n** dapat berupa konstanta integer atau karakter. Setiap pernyataan-i (**pernyataan-1**, … , **pernyataan-n**) dapat berupa pernyataan tunggal ataupun pernyataan jamak. Dalam hal ini urutan penulisan pernyataan *case* tidak berpengaruh. Proses penyeleksian berlangsung sebagai berikut :

- pengujian pada *switch* akan dimulai dari **konstanta-1**. Kalau nilai **konstanta-1** cocok dengan ekspresi maka **pernyataan-1** dijalankan. Kata kunci *break* harus disertakan di bagian akhir setiap pernyataan *case*, yang akan mengarahkan eksekusi ke akhir *switch*.
- Kalau ternyata **pernyataan-1** tidak sama dengan nilai **ekspresi**, pengujian dilanjutkan pada **konstanta-2**, dan berikutnya serupa dengan pengujian pada **konstanta-1**.
- Jika sampai pada pengujian *case* yang terakhir ternyata tidak ada kecocokan, maka pernyataan yang mengikuti kata kunci *default* yang akan dieksekusi. Kata kunci *default* ini bersifat opsional.
- Tanda kurung kurawal tutup (}) menandakan akhir dari proses penyeleksian kondisi *case*.

Di bawah ini contoh program pemakaian pernyataan *switch* untuk menggantikan *if-else* bertingkat pada program **kalkulator1.c** di atas.

```
/* File program : kalkulator2.c
Contoh penggunaan pernyataan switch untuk mengimplementasikan
kalkulator sederhana */

#include <stdio.h>

main()
{
   int  valid_operator = 1;
   char  operator;
   float  number1, number2, result;

   printf("Masukkan 2 buah bilangan dan sebuah operator\n");
   printf("dengan format : number1 operator number2\n\n");
   scanf("%f %c %f", &number1, &operator, &number2);
   switch(operator) {
   case '*' : result = number1 * number2; break;
   case '/' : result = number1 / number2; break;
   case '+' : result = number1 + number2; break;
   case '-' : result = number1 - number2; break;
   default :  valid_operator = 0;
   }
   if(valid_operator)
      printf("%g %c %g is %g\n", number1, operator,
number2,result);
   else
      printf("Invalid operator!\n");
}
```

Contoh eksekusi :

```
Masukkan 2 buah bilangan dan sebuah operator
Dengan format : number1 operator number2

23.2 = 12
invalid operator !
```

Kesimpulan :

- Operator kondisi adalah operator yang digunakan untuk menghasilkan kondisi benar *(true)* dan salah *(false)*, yang terdiri atas operator relasi dan operator logika.
- Operator relasi biasa dipakai untuk membandingkan dua buah nilai.
- Operator logika biasa dipakai untuk menghubungkan ekspresi relasi.
- Untuk penyeleksian kondisi dalam rangka pengambilan keputusan bisa digunakan salah satu dari pernyataan berikut ini :

  a. Pernyataan *if,* bentuk umumnya :

```
if (kondisi )
   pernyataan;
```

  b. Pernyataan *if-else,* bentuk umumnya :

```
if (kondisi)
   pernyataan-1;
else
   pernyataan-2;
```

  c. Pernyataan *if* di dalam *if,* bentuk umumnya :

```
if (kondisi-1)
      if (kondisi-2)
      .
      .
            if(kondisi-n)
                  pernyataan;
            else
                  pernyataan;
                  .
                  .
      else
            pernyataan;
else
      pernyataan;
```

  d. Pernyataan *else-if,* bentuk umumnya :

```
if (kondisi-1)
      pernyataan-1;
else if (kondisi-2)
      pernyataan-2;
                  .
                  .
else  if(kondisi-n)
      pernyataan-n;
else
      pernyataan-(n+1);
```

e. Pernyataan *switch,* bentuk umumnya :

```
switch (ekspresi)
        {
            case konstanta-1:
                pernyataan-1;
                ......
                break;
            case konstanta-2:
                .
                .
                .
            case konstanta-n:
                pernyataan-n;
                .......
                break;
            default:
                .......
                .......
                break;
        }
```

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1. Gunakan statemen *if* untuk membandingkan nilai dari sebuah variabel integer (**sum**) dengan nilai 65. Jika lebih kecil, maka tampilkan pesan : "Maaf, Anda harus mencoba lagi!".

2. Jika variabel **total** sama dengan variabel **tebak**, cetaklah nilai dari **total**, jika tidak sama, maka cetaklah nilai dari **tebak.**

3. Jika variabel **sum** sama dengan 10 dan variabel **total** kurang dari 20, maka tampilkan pesan : "Tidak sesuai!"

4. Jika variabel **flag** sama dengan 1 atau variabel **letter** bukan ‘X’, maka *assign* nilai 0 kepada variabel **exit_flag**, jika tidak, maka set **exit_flag** sama dengan 1.

5. Tulislah kembali pernyataan-pernyataan di bawah ini dengan menggunakan pernyataan *switch*

```
if( letter == 'X' )
   sum = 0;
else if ( letter == 'Z' )
   valid_flag = 1;
else if( letter == 'A' )
   sum = 1;
else
   printf("Unknown letter -->%c\n", letter );
```