BAB IV
KONTROL PROGRAM

Suatu algoritme disusun terurut dari atas ke bawah, demikian pula halnya dengan baris-baris program yang diproses oleh komputer berurutan dari atas ke bawah; kecuali untuk kondisi tertentu harus melompat ke baris yang ditentukan, kembali ke baris sebelumnya, dan sebagainya. Oleh karena itu, sering dilakukan pengontrolan pemrosesan instruksi agar sesuai dengan masalah yang sedang dipecahkan. Perhatikan contoh algoritme berikut:

```
jika rataan < 50 maka
    cetak 'TIDAK LULUS'.
```

yang menunjukkan bahwa instruksi 'cetak TIDAK LULUS' akan dilakukan hanya jika nilai rataan < 50. Contoh lain adalah:

```
selama counter < n lakukan:
• hasil ← hasil * 9
• counter ← counter + 1
```

yang berarti bahwa dua baris paling bawah akan diproses terus-menerus selama terpenuhi kondisi 'counter < n'.

STRUKTUR SELEKSI IF

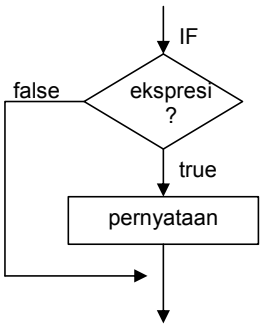
Ada dua bentuk struktur seleksi IF, yaitu bentuk IF dan IF/ELSE seperti berikut:

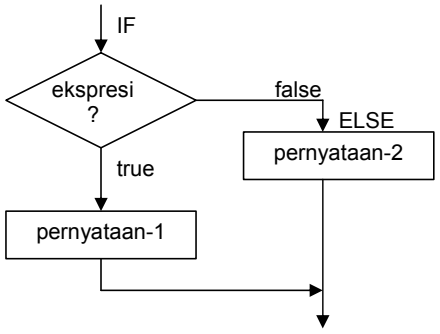
```
if (ekspresi) pernyataan;
```

dan

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

Struktur seleksi IF ini dapat digambarkan dalam bentuk diagram alir:





Bagian ELSE selalu dipadankan dengan IF terdekat yang belum terpadankan. Dan pernyataan majemuk (lebih dari satu instruksi) dituliskan di dalam tanda kurung kurawal '{ ….. }'. Perhatikan beberapa contoh penggunaan IF di bawah ini:

```
if (rataan < 50) printf("\nTIDAK LULUS");

hitung = 0;
if (a <= b)
    if (a == b)
        ++hitung;
        else {
        a = b;
        hitung = 1;
        }
```

Bandingkan struktur IF di atas dengan yang dituliskan di bawah ini:

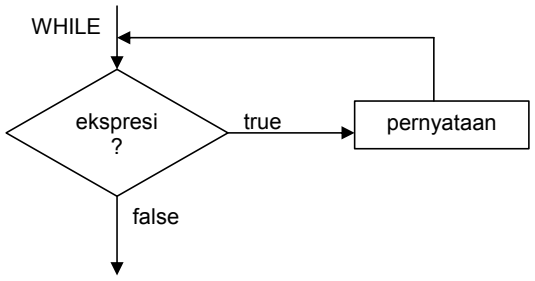
```
hitung = 0;
if (a <= b) {
    if (a == b)
        ++hitung;
    }
    else {
        a = b;
        hitung = 1;
    }
```

STRUKTUR WHILE

Jika struktur IF termasuk ke dalam golongan pernyataan *conditional*, maka struktur WHILE ini termasuk pernyataan *looping* karena digunakan untuk melakukan sekelompok instruksi secara berulang-ulang. Bentuk pernyataan ini adalah:

```
while (ekspresi)
    pernyataan;
```

yang berarti bahwa pernyataan yang dituliskan di dalam struktur while ini akan diproses terus-menerus selama ekspresi logika bernilai benar. Jika saat pertama kali struktur while diproses, nilai ekspresi adalah salah, maka pernyataan tersebut sama sekali tidak dilakukan. Dengan diagram alir, struktur ini dapat digambarkan sebagai berikut:



Perhatikan cuplikan program berikut dan perkirakan bagaimana keluarannya:

```
int x = 1;
while (x<=20) {
      printf("%d", x);
      if (x%5 == 0)
            printf("\n");
            else
            printf("\t");
      x += 2;
}
```

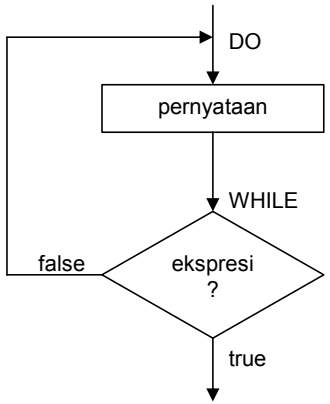
STRUKTUR DO-WHILE

Struktur ini hampir sama dengan struktur WHILE, tetapi pernyataan yang dituliskan di dalamnya akan diproses terus-menerus sampai ekspresi bernilai benar. Artinya, selama ekspresi bernilai salah maka pernyataan di dalamnya akan diproses terus; dan pada saat pertama kali struktur ini diproses, pernyataan di dalamnya otomatis akan dilaksanakan tanpa melihat nilai ekspresi terlebih dahulu.

Bentuk umum struktur DO-WHILE adalah:

```
do
      pernyataan
while (ekspresi);
```

yang secara diagram alir dapat digambarkan sebagai berikut:



Perhatikan dan pelajari cuplikan program berikut serta bandingkan hasilnya dengan struktur WHILE pada contoh sebelumnya:


```
int counter = 2;
do {
      if (counter%2 == 0)
            printf("%d\n", counter);
      counter += 2;
} while (counter<20);
```

STRUKTUR FOR

Proses *looping* yang menggunakan penghitung (*counter*) yang sudah pasti (misalnya variabel k = 1,2, ….., n; j = m, m-1, ….., 5; dan sebagainya) dapat dibuat menggunakan pernyataan FOR dengan bentuk sebagai berikut:

```
FOR (inisialisasi ; ekspresi-1 ; ekspresi-2)
      pernyataan;
```

Inisialisasi digunakan sebagai nilai awal dari penghitung, ekspresi-1 sebagai batas akhir penghitung, dan ekspresi-2 menentukan nilai penambahan atau pengurangan dari penghitung yang digunakan. Perhatikan contoh berikut:

```
int counter, total = 0;
for (counter = 0 ; counter < 5 ; counter++)
      total += counter;
printf("\n%d", total);
for (counter = 2 ; counter < 13 ; counter += 2)
      printf("\n%d", counter);
```

STRUKTUR SWITCH

Struktur SWITCH digunakan untuk mengimplementasikan pernyataan IF ganda yang bersarang (*nested* IF) dengan bentuk sebagai berikut:

```
SWITCH (ekspresi)
      pernyataan;
```

Program berikut merupakan contoh penggunaan pernyataan SWITCH untuk menghitung banyaknya huruf mutu tertentu yang dibaca dari layar monitor, yang hasilnya disimpan pada variabel nA, nB, nC, nD, dan nE.

```
#include <stdio.h>
main() {
      int grade, nA=0, nB=0, nC=0, nD=0, nE=0;
      printf("\nMasukkan huruf mutu, dan akhiri dengan EOF\n");
      while ((grade=getchar) != EOF) {
            switch (grade) {
                  case 'A': case 'a':
                        ++nA; break;
                  case 'B': case 'b':
                        ++nB; break;
```

```
                case 'C': case 'c':
                    ++nC; break;
                case 'D': case 'd':
                    ++nD; break;
                case 'E': case 'e':
                    ++nE; break;
                case '\n': case ' ';
                    break;
                default:
                    printf("Salah data\n");
            }
        }
        printf("\nA: %d", nA);
        printf("\nB: %d", nB);
        printf("\nC: %d", nC);
        printf("\nD: %d", nD);
        printf("\nE: %d", nE);
        return 0;
}
```

STRUKTUR JUMP

Proses looping yang sedang berlangsung dapat diatur kembali dengan menggunakan pernyataan **break** (yang mengakhiri struktur while, do-while, for, atau switch terdekat yang mencakupnya), dan pernyataan **continue** (yang mengakhiri proses iterasi yang sedang berlangsung. Perhatikan dan telaah contoh program berikut, serta apa yang terjadi jika pernyataan 'break' diganti dengan 'continue':

```
#include <stdio.h>
main() {
    int counter;
    for (counter = 1 ; counter <= 10 ; counter++) {
        if (counter == 5)
            break;
        printf("\n%d", counter);
    }
    printf("\nNilai counter saat ini : %d", counter);
    return 0;
}
```

LATIHAN 4.
Terjemahkan setiap jawaban algoritme pada LATIHAN 2 ke dalam bahasa pemrograman C dan proses program tersebut menggunakan komputer.