Bab 3

Perulangan

3.1 Perulangan

Perulangan atau looping memungkinkan kita untuk mengeksekusi potongan kode berulang-ulang hingga mencapai suatu kondisi. Ada 2 jenis perulangan dalam bahasa C, yaitu while dan for.

3.1.1 Pernyataan while

Perulangan menggunakan while mirip dengan perulangan for, tetapi memiliki fungsi yang lebih sedikit. Perulangan while akan terus mengeksekusi sejumlah kode selama kondisi dari while terpenuhi. Sintaksnya adalah sebagai berikut:

```
//initial value misal, i = 0
while (<Ekspresi/Kondisi>) {
    // Potongan kode yang ingin dieksekusi
    .
    .
    .
    // increment/decrement misalnya, i++
}
```

Cara kerja perulangan while mirip dengan if. Jika pada **if** potongan kode akan dieksekusi **sekali saja** apabila ekspresi/kondisi bernilai **TRUE**, pada **while** potongan kode akan **terus dieksekusi** hingga ekspresi/kondisi menghasilkan **FALSE**.

Contohnya jika diinginkan menjalankan sejumlah kode program 10 kali:

```
Int n = 0;
while(n<10) {
    n++;
}
```

Sehingga pada contoh diatas :

1. Pada awalnya, variabel n bernilai 0.
2. Sequence selanjutnya adalah while, dan n bernilai kurang dari 10 (TRUE), maka kode didalam while akan dijalankan, yakni n++.
3. Setelah melakukan increment, kembali ke statement while untuk memeriksa apakah n masih kurang dari 10 setelah diincrement
4. Karena setelah n di-increment nilainya masih 1 dan kurang dari 10, maka while akan dijalankan lagi hingga n bernilai 10 yang berarti tidak memenuhi kondisi while.

Perulangan while akan mengeksekusi kode selama-lamanya, jika kondisi dari while selalu benar (true / tidak nol):

```
while ( 1 )
{
    //Do something
}
```

3.1.2 Pernyataan for

Perulangan menggunakan for pada bahasa C memungkinkan untuk menjalankan sejumlah kode selama beberapa kali. Perulangan for membutuhkan sebuah variabel iterator, biasanya digunakan notasi i, tetapi bisa juga menggunakan nama variabel lain. Dalam perulangan for digunakan operator ++ disebut dengan operator **increment** atau operator -- merupakan operator **decrement**. Kedua operator ini digunakan untuk menambah (increment)/mengurangi (decrement) nilai dari suatu variabel sebanyak satu. Terdapat dua cara untuk menggunakan operator ini. Sintaks untuk pernyataan for adalah sebagai berikut:

```
for (init_statement; kondisi/ekspresi; end_statement) {
    //  Potongan kode yang dieksekusi
    .
    .
}
```

Cara kerjanya adalah sebagai berikut :

- Bagian init_statement digunakan untuk inisialisasi variabel yang akan digunakan dalam perulangan. Bagian ini hanya dijalankan sekali saka pada saat awal perulangan.
- Selanjutnya kondisi/ekspresi akan dievaluasi. Jika menghasilkan TRUE, maka akan mengeksekusi potongan kode. Jika menghasilkan FALSE, perulangan berhenti.
- Setelah potongan kode selesai dieksekusi, akan mengeksekusi bagian end_statement. Biasanya bagian ini digunakan sebagai increment/decrement.
- Lalu akan mengevaluasi ekspresi/kondisi lagi, dan begitu seterusnya.

Contohnya jika diinginkan menjalankan sejumlah kode program 10 kali:

```
int i;
for(i=0; i<10;i++) {
        printf("%d\n",i);
}
```

Cara kerja kode diatas yaitu:

1. Awalnya i bernilai 0.

2. For statement akan memeriksa nilai i apakah kurang dari 10.
3. Apabila TRUE maka jalankan kode dalam for block, yakni print nilai i.
4. Setelah command dalam block for selesai dijalankan, maka variabel i akan diincrement (menggunakan operator increment jadi nilai i bertambah 1), dan di periksa lagi.
5. Apabila i kurang dari 10, maka command dalam block dieksekusi, apabila tidak maka for loop akan berhenti.

Perulangan for dapat digunakan untuk melakukan iterasi dari semua nilai array. Contohnya jika diinginkan untuk menambah semua nilai dalam array:

```
int array[10]={1,2,3,4,5,6,7,8,9,10};
int sum = 0;
int i;
for(i=0; i<10;i++) {
        sum+ = array[i];
}
* sum sekarang akan berisi a[0] + a[1] + . . .  + a[9]
*/
printf("Jumlah  dari  array  adalah    %d\n",  sum);
```

3.2 Loop Directives

Ada dua cara untuk mengatur alur dari perulangan dalam bahasa C, menggunakan break dan continue.

3.2.1 Break

Perintah break digunakan untuk **menghentikan** perulangan (secara paksa). Apabila perintah break pada suatu perulangan dijalankan, maka perulangan tersebut akan **dihentikan (secara paksa) dari titik dimana perintah break muncul**.

Berikut adalah contoh pernyataan break:

```
int n =0;
while(1) {
        n++;
        if(n==10){
        break
        }
}
```

Pada kode tersebut, break akan menghentikan perulangan setelah 10 iterasi, walaupun kondisi dari perulangan while masih terpenuhi.

3.2.2 Continue

Kebalikan dari perintah break, perintah continue digunakan untuk **melanjutkan perulangan**. Pada perulangan, apabila menemui perintah continue, maka perintah-perintah di bawah continue akan **diabaikan** dan **kembali akan**

mengevaluasi ekspresi/kondisi. Sedangkan pada perulangan for akan langsung mengekekusi bagian end_statement kemudian mengevaluasi ekspresi/kondisi.

Berikut adalah contoh pernyataan continue:

```
int n =0;
while(n<10) {
    n++ ;
    /*cek apakah n bilangan ganjil*/
    if(n%2==1){
    /*kembali ke permulaan while jika ganjil*/
    continue;
    }
    /*kode berikut hanya dijalankan jika n genap*/
    printf("Bilangan %d adalah genap.In", n);
}
```

Pada kode tersebut, continue menyebabkan perintah printf dilewati ketika nilai n ganjil dan kode akan langsung kembali mengevaluasi ekspresi/kondisi pada while.