MODUL -5

Pengantar Perulangan

Pengertian Perulangan Bersarang (Nested Loop)

Perulangan bersarang adalah sebutan untuk **perulangan di dalam perulangan**. Konsep seperti ini sering dipakai untuk memecahkan masalah programming yang cukup kompleks. Semua jenis perulangan bisa dibuat dalam bentuk perulangan bersarang, termasuk perulangan FOR, WHILE dan DO WHILE. Dalam bahasa inggris, perulangan bersarang ini dikenal dengan sebutan **nested loop**. Berikut contoh format dasar perulangan bersarang dalam bahasa C:

```
1
    for (start1; condition1; increment1)
2   {
3     // kode program
4     for (start2; condition2; increment2)
5     {
6        // kode program
      }
7   }
8
```

Di dalam perulangan bersarang terdapat istilah **outer loop** dan **inner loop**. Sesuai dengan namanya, *outer loop* adalah sebutan untuk perulangan luar, sedangkan *inner loop* sebutan untuk perulangan dalam.

Pada contoh di atas, *outer loop* adalah perulangan di baris 1, sedangkan *inner loop* adalah perulangan di baris 4. Kode program di dalam *outer loop* akan dijalankan sejumlah kondisi perulangan di outer saja. Sedangkan kode program yang ada di dalam perulangan *inner loop* akan dijalankan sebanyak perulangan *outer * inner*.

Tidak ada batasan seberapa banyak "kedalaman" dari sebuah perulangan bersarang. Kita bisa saja membuat perulangan di dalam perulangan di dalam perulangan, dst. Tentu saja semakin banyak perulangan yang "bersarang", kode programnya juga akan makin kompleks.

Dalam membuat perulangan berasarang kita juga harus sangat teliti dalam penggunaan tanda kurung kurawal "{ }" untuk menandakan awal dan akhir sebuah blok kode program.

Tidak jarang hasilnya jadi berantakan karena salah menulis posisi penutup kurung kurawal.

Contoh Kode Program Perulangan Bersarang (Nested Loop)

Mari kita masuk ke contoh kode program dari perulangan bersarang dalam bahasa C. Namun sebelum itu, kita berangkat dari perulangan "normal" terlebih dahulu.

Latihan pertama, bisakah anda merancang kode program untuk membuat daftar perkalian 3 dari 1 sampai 10? Hasil akhir yang saya inginkan adalah seperti ini:

```
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

Tentu saja baris di atas bisa dihasilkan dengan membuat perintah **printf** sebanyak 10 kali, namun hal itu tidak efisien. Hasil di atas harus dibuat menggunakan perulangan **FOR**.
Baik, berikut kode program yang saya gunakan:

```
1
2   #include <stdio.h>
3
4   int main(void)
5   {
6     for (int i = 1; i <= 10; i++) {
7       printf("3 * %i = %i\n",i,3*i);
8     }
9     return 0;
   }
```

Disini saya menggunakan sebuah perulangan FOR. Perulangan mulai dari variabel counter **i = 1** sampai **i <= 10**. Sepanjang perulangan, tampilkan hasil dari **printf("3 * %i = %i\n",i,3*i)**.
Jika anda sudah memahami konsep dasar perulangan FOR, maka kode program ini seharusnya bisa dipahami dengan mudah.

Berikutnya, bagaimana dengan membuat kode program untuk menghasilkan daftar perkalian yang sama seperti di atas, tapi kali ini bukan hanya perulangan 3 saja, tapi dari 1

– 10. Maksudnya, saya ingin membuat kode program yang menghasilkan 100 baris daftar perkalian, mulai dari:

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
....
....
```

Sampai dengan:

```
....
....
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

Ada beberapa cara untuk membuat daftar perkalian seperti ini, misalnya bisa dengan membuat 10 kali perulangan FOR (1 perulangan untuk setiap angka). Namun cara paling efisien adalah menggunakan *perulangan bersarang*.

Berikut contoh kode program yang bisa dipakai:

```
1
2     #include <stdio.h>
3
4     int main(void)
5     {
6       for (int i = 1; i <= 10; i++) {
7         for (int j = 1; j <= 10; j++) {
8           printf("%i * %i = %i\n",i,j,j*i);
9         }
10        printf("\n");
11      }
12      return 0;
    }
```

- Jenis – jenis perulangan dalam bahasa C++ terdiri dari :
- Perulangan for
- Perulangan nested – for
- Perulangan tak terhingga
- Pernyataan go to
- Perulangan while
- Perulangan do – while
- Pernyataan break
- Perulangan continue

Struktur perulangan

- Struktur perulangan secara umum terdiri dari dua bagian, yaitu:

1. Kondisi perulangan, yaitu ekspresi Boolean yang harus dipenuhi untuk melaksanakan pengulangan
2. Body pengulangan, yaitu bagian algoritma yang diulang

- Terdapat struktur perulangan yang biasanya disertai dengan bagian, yaitu:

1. Inisialisasi, yaitu aksi yang dilakukan sebelum pengulangan dilakukan pertama kali
2. Terminasi, yaitu aksi yang dilakukan setelah pengulangan selesai dilaksanakan

Struktur Perulangan

- **Penulisan struktur perulangan secara umum :**

<inisialisasi>

awal pengulangan

badan pengulangan

akhir pengulangan

<terminasi>

Perulangan for

- **Perulangan for digunakan untuk menghasilkan pengulangan sejumlah kali yang telah dispesifikasikan.**
- **Jumlah pengulangan dapat ditentukan sebelum dieksekusi.**
- **Bentuk umum pernyataan for :**
- **Bila pernyataan di dalam for lebih dari satu maka pernyataan tersebut harus diletakan seperti dibawah ini :**

```
For(inisialisasi : syarat pengulangan;
pengubah nilai pencacah)
{
    pernyataan / perintah;
    pernyataan / perintah;
```

Kegunaan dari masing – masing argument for adalah :

- **Inisialisasi : merupakan bagian untuk memberikan nilai awal untuk variabel – variabel tertentu**
- **Syarat pengulangan : memegang kontrol terhadap pengulangan, karena bagian ini yang akan menentukan suatu perulangan diteruskan atau dihentikan**
- **Pengubah nilai pencacah : mengatur kenaikan atau penurunan nilai Pencacah**

Struktur For

Dalam C++ terdapat 3 jenis struktur perulangan, yaitu : Struktur For, Struktur While dan Struktur Do While. Pada kesempatan kali ini kita akan membahas **Struktur Perulangan For dalam C++, Lengkap Contoh dan Penjelasan.** Struktur pengulangan / perulangan jenis for biasanya digunakan untuk melakukan perulangan yang telah diketahui banyaknya. Biasanya jenis perulangan for dianggap sebagai jenis perulangan yang paling mudah dipahami.

Untuk melakukan perulangan dengan menggunakan struktur perulangan for, kita harus memiliki sebuah variabel sebagai indeksnya. Namun perlu sekali untuk diperhatikan bahwa tipe data dari variabel yang akan digunakan sebagai indeks haruslah tipe data yang mempunyai urutan yang teratur, misalnya tipe data int (0,1,2, ...) atau char ('a' , 'b' , 'c' , ...).

Adapun **bentuk umum dari struktur perulangan for** adalah seperti yang tampak dibawah ini:

```
// Untuk perulangan yang sifatnya menaik (increment)
// Pastikan nialai awal < kondisi saat berjalan
for(variabel = niali_awal ; kondisi_saat_berjalan ; variable++)
{
    Statemen_yang_akan_diulang;
}

// Untuk perulangan yang sifatnya menurun (decrement)
// Pastikan nialai awal > kondisi saat berjalan
```

```
for(variabel = nilai_awal ; kondisi_saat_berjalan ; variable--)
{
    Statemen_yang_akan_diulang;
}
```

Sebagai catatan bahwa jika kita melakukan perulangan yang sifatnya menaik (increment) maka nilai awal dari variabel yang kita definisikan haruslah lebih kecil dari nilai akhir yang dituliskan dalam kondisi (kondisi saat berjalan). Sebaliknya jika kita akan melakukan perulangan yang sifatnya menurun (decrement) maka nilai awal harus lebih besar dari nilai akhir.

Contoh Program dengan Struktur For

Berikut ini contoh program yang menunjukkan perulangan dengan menggunakan struktur for. Program akan ditulis dalam dua tipe, yaitu perulangan for yang sifatnya menaik (increment) dan perulangan for yang sifatnya menurun (decrement).

Contoh perulangan For yang sifatnya menaik (increment)

```
#include <iostream>
using namespace std;

int main(){
    int MD;     for(MD=0;MD<8;MD++){
        cout<<"Belajar C++ Bareng MateriDosen.Com"<<endl;
    }

    return 0;
}
```

Contoh perulangan For yang sifatnya menurun (decrement)

```
#include <iostream>

using namespace std;

int main(){
    int MD;

    for(MD=8;MD>0;MD--){
        cout<<"Belajar C++ Bareng MateriDosen.Com"<<endl;
```

```
    }

    return 0;
}
```

Kedua program diatas jika dijalankan akan menghasilkan hasil yang sama, yaitu sebagai berikut:

Contoh Perulangan For, While & Do While menggunakan C++

Kali ini saya akan membagikan project yang saya buat dengan C++ yaitu membuat perulangan dengan c++ menggunakan perulangan for, perulangan while dan menggunakan do while.

Kenapa C++? Karena biasa kita akan diajarkan untuk mengasah logika pemrograman kita menggunakan C++ hehe makanya sampai saat ini peminat C++ masih banyak ya untuk kalangan yang baru belajar untuk mengasah logikanya hehe

Perulangan Nested – for

- **Perulangan nested for adalah suatu perulangan for di dalam perulangan for yang lainnya**
- **Didalam penggunaan nested-for, perulangan yang didalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan.**
- **Bentuk umum pernyataan Nested for, sebagai berikut :**

```
    for ( inisialisasi; syarat
pengulangan; pengubah nilai
pencacah )
{
    for ( inisialisasi; syarat
```

Perulangan Tak Terhingga

- **Perulangan tak terhingga merupakan perulangan (loop) yang tak pernah berhenti atau terus mengulang perintah**
- **Hal ini sering terjadi, ketika terjadi kesalahan dalam penanganan kondisi yang dipakai untuk keluaran.**

GO TO :

untuk membuat pernayataan goto bekerja, dibutuhkan dua pernyataan yaitu GOTO dan LABEL. Pernyataan GOTO menggunakan keyword “goto” diikuti nama label berfungsi untuk memberi tahu CPU agar melompat ke baris yang miliki label dengan nama tersebut. Pernyataan LABEL di isi dengan nama label diakhiri dengan tanda titik dua, berfungsi untuk menandai suatu baris program, tempat dimana loncatan CPU karena GOTO mendarat.
Contoh Penulisan Goto

Ketika CPU bertemu dengan pernyataan GOTO maka CPU pada saat itu juga akan melompat ke label dengan nama yang tertera pada pernyataan GOTO. Dan untuk penempatan GOTO beserta label GOTO kita bebas meletakanya dimana saja tetapi harus berada pada satu function scope.

```
#include <iostream>
using namespace std;

int main ()
{
    int angka=0;
    cobaLagi:
    cout<<"Masukan Angka : ";cin>>angka;
    if (angka!=5)goto cobaLagi;

    return 0;
}
```

Contoh program di atas adalah contoh penggunaan pernyataan GOTO. Program di atas akan meminta pengguna untuk memasukan angka apapun, jika pengguna memasukan angka 5 maka pernaytaan GOTO akan dieksekusi, CPU akan meloncat ke label “cobaLagi” yang diletakan di atas.

Peletakan Label dari GOTO bisa diletakan di baris sebelum pernyataan GOTO, kita juga bisa meletakan Label GOTO di baris setelah dari pernyataan GOTO.

Contoh Program :

```
#include <iostream>
using namespace std;

int main ()
{
    char loncat=0;

    cout<<"Masukan Angka [y/n] : ";cin>>loncat;
    cout <<"1"<<endl;
    if (loncat=='y')goto loncatan;
    cout <<"2"<<endl;
    loncatan:
    cout <<"3"<<endl;

    return 0;
}
```

Dalam bahasa C++ tersedia suatu fasilitas yang digunakan untuk melakukan proses yang berulangulang sebanyak keinginan kita. Misalnya saja, bila kita ingin menginput dan mencetak bilangan dari 1 sampai 100 bahkan 1000, tentunya kita akan merasa kesulitan. Namun dengan struktur perulangan proses, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja. Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam-macam.

Sebuah / kelompok instruksi diulang untuk jumlah pengulangan tertentu. Baik yang terdifinisikan sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian : (1) Kondisi pengulangan yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan;

(2) Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan

diulang.

Perintah atau notasi dalam struktur pengulangan adalah :

1) Pernyataan **while**
2) Pernyataan **do..while**
3) Pernyataan **for**
4) Pernyataan **continue dan break**
5) Pernyataan **go to**

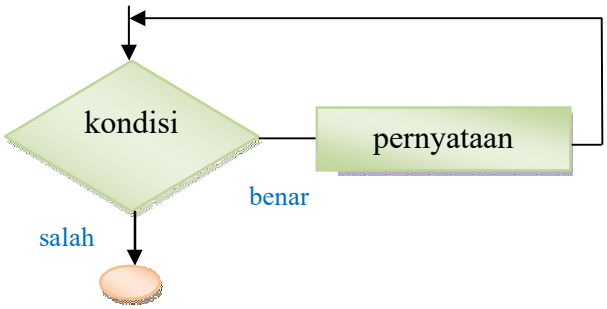
(1) Struktur Perulangan " WHILE"

Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (≠0) dan akan berhenti bila kondisinya bernilai salah (=0).

Bentuk Umumnya :

```
while (kondisi)
{
    Pernyataan ;
}
```

Pengujian ungkapan pada while dilakukan sebelum bagian pernyataan, Oleh karena itu ada kemungkinan bagian pernyataan pada while tidak dijalankan sama sekali, yaitu kalau kondisi yang pertama kali bernilai salah. Perhtikan gambar flowchart berikut :



Gambar. 21. Diagram flowchart Perulanagn dengan While

Catatan :
Jika menggunakan while pastikan bahawa pada suatu saat ungkapan pada while bernilai salah. Jika tidak demikian pernyataan yang mengikutinya akan dijalankan selamanya.

Berikut contoh program perulangan dengan while untuk menampilkan " C++" sebanyak 10 kali.

```
//Program menampilkan " C++ " sebanyak 10 kali dengan while
# include <iostream.h>
# include <conio.h>
void main( )
{
   int  i ;          // sebagai variabel pencacah untuk menyatakan
   jumlah tulisan sebanyak 10 kali.

   clrscr ( );
   i = 0             // mula-mulai I diisi dengan 0

   while ( i < 10)
    {
       cout << " c++ " << endl;
          i ++;             // menaikkan pencacah sebesar 1
    }
}
```

(2) Struktur do-while

Pada dasarnya struktur perulangan do....while sama saja dengan struktur while, hanya saja pada proses perulangan dengan while, seleksi berada di while yang letaknya di atas sementara pada perulangan do....while, seleksi while berada di bawah batas perulangan. Jadi dengan menggunakan struktur do…while sekurang-kurangnya akan terjadi satu kali perulangan.

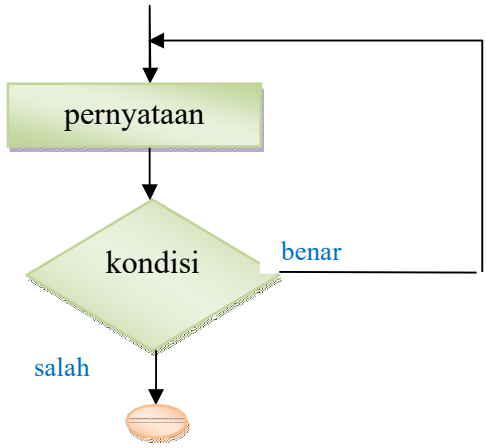
Pada struktur *do-while* kondisi pengecekan ditempatkan di bagian akhir. Hal ini menyebabkan struktur pengulangan ini minimal akan melakukan satu kali proses walaupun kondisi yang didefinisikan tidak terpenuhi (bernilai salah). Bentuk umum dari struktur *do-while*

```
do
   {
    pernyataan 1;
    pernyataan 2;
   ….
    pernyataan n ;
  }while (ungkapan)
```

Catatan :

- Bagian pernyataan1 hingga N dijalankan secara berulang sampai dengan ungkapan bernilai salah (=0).
- Berbeda dengan while, pengujian ungkapan dilakukan di bagian belakang (setelah bagian pernyataan).
- Dengan demikian bagian pernyataan pada pengujian do – while minimal akan dijalankan satu kali.

Perhatikan bentuk flowchart do .. while berikut ini:



Gambar 22. Diagram flowchart perulanagn dengan Do - While

Berikut program modifikasi prgram sebelumnya untuk menampilkan tulisan C++ 10 kali dengan do – while.

```
//Program menampilkan " C++ " sebanyak 10 kali dengan do – while *
# include <iostream.h> #
include <conio.h> void
main( )
{
    int  i ;            // sebagai variabel pencacah untuk menyatakan
    jumlah tulisan sebanyak 10 kali.
    clrscr ( );

   i = 0                // mula-mulai I diisi dengan 0
```

```
    do
       {
          cout << " c++ " << endl;
             i ++;              // menaikkan pencacah sebesar 1
       } while ( i < 10);
}
```

(3) Struktur Perulangan "FOR"

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana.

Pernyataan for digunakan untuk melakukan looping. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama *kondisi* terpenuhi, maka pernyataan akan terus dieksekusi.

Bentuk umum perulangan for adalah sebagai berikut :

```
for ( ungkapan1; ungkapan2; ungkapan3)
      Pernyataan;
```

Keterangan :

- Ungkapan1 merupakan pernyataan inisialisasi
- Ungkapan 2 sebagai kondisi yang menentukan pengulangan thd pernyataan
- Ungkapan 3 sebagai pengatur variabel yang digunakan di ungkapan1

Contoh program FOR untuk menampilkan bilangan genap

```
//Program FOR untuk menentukan bilangan genap, yang
//ditampilakn dari besar ke kecil menuji nol
# include <iostream.h>
# include <conio.h>
void main( )
{
      int n;
      clrscr ( );
```

```
    cout <<" menampilkan deret bilangan genap "<<endl;
    cout <<" kurang atau sama dengan n"<<endl;
    cin >> n;
              // jika bilangan ganjil, maka kurangi sebesar 1
    if (n % 2)
    n- - ;
              // tampilkan deret bilangan genap dari besar
               //ke kecil menuju nol

    for ( ; n>= 0; n-=2)
            cout << n << ' ' ;
}
```

Pernyataan *for* dapat berada di dalam pernyataan *for* lainnya yang biasa disebut ***nested for***.

Contoh program For di dalam for sbb :

```
//Program untuk menampilkan bentu segituga karakter '*'
//dengan menggunakan for di dalam for
# include <iostream.h>
# include <conio.h>
void main( )
{
   int tinggi,          //menyatakan tinggi puncak
      baris,            //pencacah untuk baris
      kolom;            //pencacah untuk kolom

      clrscr ( );
      cout << "Tinggi segi tiga : ";
      cin >> tinggi;

      cout <<endl;
      for (baris = 1; baris <= tinggi; baris++)
          {
            for (kolom = 1; kolom <= baris ; kolom++)
            cout <<'*';
            cout << endl;
           }
   }
```