MODUL IV
STATEMENT PENGULANGAN

I. TUJUAN

1. Mengimplementasikan struktur pengulangan dalam bahasa C++
2. Menerapkan sintaks-sintaks pengulangan dalam menyelesaikan persoalan
3. Mahasiswa mampu menyelesaikan persoalan tentang pengulangan

II. DASAR TEORI

Dalam pembuatan program, terkadang kita harus melakukan pengulangan suatu aksi, misalnya untuk melakukan perhitungan berulang dengan menggunakan formula yang sama. Sebagai contoh, misalnya kita ingin membuat program yang dapat menampilkan sebuah teks "*Saya sedang belajar bahasa C++*" sebanyak 5 kali, maka kita tidak perlu untuk menuliskan 5 buah statemen melainkan kita hanya tinggal menempatkan satu buah statemen ke dalam suatu struktur pengulangan. Dengan demikian program kita akan lebih efesien.

Sebagai gambaran bagi Anda untuk dapat lebih memahami konsep pengulangan, coba Anda perhatikan terlebih dahulu contoh kode program di bawah ini.

Kode Program :

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

int main() {
  cout<<"Saya sangat menyukai C++"<<endl;
  cout<<"Saya sangat menyukai C++"<<endl;
  cout<<"Saya sangat menyukai C++"<<endl;
  cout<<"Saya sangat menyukai C++"<<endl;
  cout<<"Saya sangat menyukai C++"<<endl;

  return 0;
}
```

Output Program :

```
 Saya sangat menyukai C++
 Saya sangat menyukai C++
 Saya sangat menyukai C++
 Saya sangat menyukai C++
 Saya sangat menyukai C++
```

1. Struktur For

Struktur *for* ini digunakan untuk menuliskan jenis pengulangan yang banyaknya sudah pasti atau telah diketahui sebelumnya. Oleh karena itu, di sini kita harus melakukan inisialisai nilai untuk kondisi awal pengulangan dan juga harus menuliskan kondisi untuk menghentikan proses pengulangan. Adapun bentuk umum dari pendefinisian struktur for(untuk sebuah statemen) dalam bahasa C++ adalah sebagai berikut.

```
for (ekpresi1; ekspresi2; ekspresi3)
    Statemen_yang_akan_diulang;
```

Sedangkan untuk dua statemen atau lebih.

```
for (ekpresi1; ekspresi2; ekspresi3) {
    Statemen_yang_akan_diulang1;
    Statemen_yang_akan_diulang2;
    …
}
```

Keterangan :

Ekspresi1 ➔ digunakan sebagai proses inisialisasi variabel yang akan dijadikan sebagai pencacah (counter) dari proses pengulangan, dengan kata lain ekspresi ini akan dijadikan sebagai kondisi awal.

Ekspresi2 ➔ digunakan sebagai kondisi akhir, yaitu kondisi dimana proses pengulangan harus dihentikan. Perlu untuk diketahui bahwa pengulangan masih akan dilakukan selama kondisi akhir bernilai benar.

Ekspresi3 ➔ digunakan untuk menaikkan (*increment*) atau menurunkan (*decrement*) nilai variabel yang digunakan sebagai pencacah. Apabila pengulangan yang kita lakukan bersifat menaik, maka kita akan menggunakan statemen *increment*, sedangkan apabila pengulangan yang akan kita lakukan bersifat menurun maka kita harus menggunakan statemen *decrement*.

Berikut ini contoh untuk mengilustrasikan struktur pengulangan *for* yang telah diterangkan di atas.

```
for (int j=0; j<5; j++) {
    /* Statemen yang akan diulang */
    ...
}
```

Pada sintak di atas, mula-mula kita menginisialisasi variabel j dengan nilai 0, kemudian karena ekspresi (0 < 5) bernilai benar maka program akan melakukan statemen untuk pertama kalinya. Setelah itu variabel j akan dinaikkan nilainya sebesar 1 melalui statemen *increment* *j*++ sehingga nilai j sekarang menjadi 1. Sampai di sini program akan mengecek ekspresi (j < 5). Oleh karena ekspresi (2 < 10) bernilai benar, maka program akan melakukan statemen yang kedua kalinya. Begitu seterusnya sampai nilai j bernilai 4. Namun pada saat variabel j telah bernilai 5 maka program akan keluar dari proses pengulangan. Hal ini disebabkan oleh karena ekspresi (5 < 5) bernilai salah.

Untuk membuktikan hal tersebut, perhatikan contoh program di bawah ini dimana kita akan menampilkan teks "Saya sedang belajar bahasa C++" sebanyak 10 kali.

Kode Program :

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

int main() {
  int C;
  for (C=5; C>0; C++) {
     cout<<"Saya sangat menyukai C++"<<endl;
  }
  return 0;
}
```

Output Program :

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

2. Struktur While

Pada struktur pengulangan jenis ini kondisi akan diperiksa di bagian awal. Hal ini tentu menyebabkan kemungkinan bahwa apabila ternyata kondisi yang kita definisikan tidak terpenuhi (bernilai salah), maka proses pengulangan pun tidak akan pernah dilakukan. Adapun bentuk umum dari struktur *while* adalah seperti yang tampak di bawah ini.

```
while (ekspresi) {
    Statemen_yang_akan_diulang1;
    Statemen_yang_akan_diulang2;
    …
}
```

Sebagai pembanding dengan struktur pengulangan *for* di atas, maka di sini dituliskan kembali program yang akan menampilkan teks "Saya sedang belajar bahasa C++" dengan menggunakan struktur *while*. Adapun sintaknya adalah sebagai berikut.

Kode Program :

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

int main() {
   int C;  // Mendeklarasikan varaiebl C sebagai
                  // indeks pengulangan
   C = 0;  // Melakukan inisialisasi nilai terhadap
                  // variabel C

   while (C<5) {
     cout<<"Saya sangat menyukai C++"<<endl;
     C++;  /* Statemen ini berguna untuk menaikkan nilai,
              dan setelah bernilai 5,
              maka pengulangan akan dihentikan */
   }
   return 0;
}
```

Output Program :

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

3. Struktur do-While

Berbeda dengan struktur *while* dimana kondisinya terletak di awal blok pengulangan, pada struktur *do-while* kondisi diletakkan di akhir blok pengulangan. Hal ini menyebabkan bahwa statemen yang terdapat di dalam blok pengulangan ini pasti akan dieksekusi minimal satu kali, walaupun kondisinya bernilai salah sekalipun. Maka dari itu struktur *do-while* ini banyak digunakan untuk kasus-kasus pengulangan yang tidak mempedulikan benar atau salahnya kondisi pada saat memulai proses pengulangan. Adapun bentuk umum dari struktur pengulangan *do-while* adalah seperti yang tertulis di bawah ini.

```
do {
    Statemen_yang_akan_diulang;
    …
} while(ekspresi);
```

Mungkin bagi Anda yang baru mengenal bahasa pemrograman C++ akan merasa bingung untuk membedakan struktur pengulangan *while* dan *do-while*. Maka dari itu, perhatikan dulu dua buah contoh program berikut ini.

Kode Program :

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

int main() {
   int C = 0;

     do {
       cout<<"Saya sangat menyukai C++"<<endl;
       C++;
     } while (C < 5);
   return 0;
}
```

Output Program :

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

III. LATIHAN

1. Kasus :

Membuat program dengan menggunakan statement *for* untuk membedakan pengulangan yang sifatnya menaik dan menurun. Program akan memilki output sebagai berikut :

```
PENGULANGAN MENAIK
1
2
3
4
5

PENGULANGAN MENURUN
5
4
3
2
1
```

Solusi :

```
#include <iostream>
using namespace std;
int main()
{
  cout<<"PENGULANGAN MENAIK"<<endl;
  for (int C=0; C<5; C++) {
    cout<<C+1<<endl;
  }

  // Membuat spasi vertikal
  cout<<'\n';     // dapat ditulis cout<<endl;
  cout<<"PENGULANGAN MENURUN"<<endl;
  for (int J=5; J>0; J--) {
    cout<<J<<endl;
  }
  return 0;
}
```

2. Kasus :

Membuat program dengan menggunakan statement *for* yang dapat menampilkan deret bilangan bulat, dengan output sebagai berikut :

```
10    20    30    40    50    60    70    80    90    100
9     18    27    36    45    54    63    72    81
8     16    24    32    40    48    56    64
7     14    21    28    35    42    49
6     12    18    24    30    36
5     10    15    20    25
4     8     12    16
3     6     9
2     4
1
```

Solusi :

```
#include <iostream>

using namespace std;

int main() {
  int J = 10;
  int K;

  while (J >= 1) {
    K = 1;
    while (K <= J) {
      cout<<K*J<<' ';
      K++;
    }
    cout<<'\n';
    J--;
  }

  return 0;
}
```

3. Kasus :

Membuat program dengan menggunakan statement *While* untuk menghitung nilai faktorial dari sebuah bilangan bulat, dengan output sebagai berikut :

Masukan bilangan yang akan dihitung : 5
5! = 5 x 4 x 3 x 2 x 1 =120

Solusi :

```
#include <iostream>
using namespace std;
int main()
{
  int BIL;
  long faktorial = 1;

  cout<<"Masukkan bilangan yang akan dihitung: ";
  cin>>BIL;
  int C = BIL;
  cout<<C<<"! = ";
  while (C >= 1) {
    faktorial *= C;      // faktorial = faktorial * C;
    if (C != 1) {
      cout<<C<<" x ";
    } else {
      cout<<C<<" = ";
    }
    C--;    // Menurunkan nilai variabel C
  }
  cout<<faktorial;
  return 0;
}
```

4. **Kasus :**

Membuat program dengan menggunakan statement *do - While* untuk menentukan nilai faktor persekutuan terbesar dari dua buah bilangan bulat. Sebagai contoh kita memasukan dua buah bilangan bulat yaitu 8 dan 12, maka factor persekutuan terbesar dari kedua bilangan tersebut adalah 4. Untuk lebih jelasnya perhatikan tabel kelipatan di bawah ini.

Bilangan	**Faktor**
8	1,2,**4**,8
12	1,2,3,**4**,612

Program tersebut mempunyai output sebagai berikut :

```
Masukan bilangan pertama : 8
Masukan bilangan kedua : 12

Faktor persekutuan terbesar = 4
```

Solusi :

```
#include <iostream>
using namespace std;
int main()
{
  int Bil1, Bil2;
  int sisa;

  cout<<"Masukkan bilangan pertama  : "; cin>>Bil1;
  cout<<"Masukkan bilangan kedua    : "; cin>>Bil2;

  // Melakukan pertukaran nilai
  if (Bil1 < Bil2) {
    int temp = Bil1;
    Bil1 = Bil2;
    Bil2 = temp;
  }

  do {
    sisa = Bil1 % Bil2;
    Bil1 = Bil2;
    Bil2 = sisa;
  } while (sisa != 0);

  cout<<"\nFaktor persekutuan terbesar = "<<Bil1;

  return 0;
}
```

IV. TUGAS

1. Buatlah sebuah program dengan statement **pengulangan**, dimana dapat menghitung total nilai dari suatu bilangan yang diinputkan. Dengan tampilan output sebagai berikut : **[bobot : 30]**

```
Masukan bilangan : 5
Total Nilai = 5 + 4 + 3 + 2 + 1  = 15
```

2. Buatlah sebuah program dengan statement **pengulangan**, dimana dapat menghitung hasil pangkat suatu bilangan. Dengan tampilan output sebagai berikut : **[bobot : 30]**

```
Masukan bilangan  : 2
Masukan pencacah  : 3
Hasil pangkat     : 8
```

3. Buatlah sebuah program dengan statement **pengulangan** untuk menentukan kelipatan persekutuan terkecil (KPK) dari dua buah bilangan bulat. Sebagai contoh KPK dari bilangan 8 dan 12 adalah **24**. Untuk lebih jelasnya perhatikan tabel kelipatan di bawah ini.

8	16	**24**	32	40	48	…
12	**24**	36	48	60	72	…

Program tersebut mempunyai output sebagai berikut : **[bobot : 40]**

```
Masukkan bilangan ke-1 : 12
Masukkan bilangan ke-2 : 8
KPK : 24
```

~ Selamat Berlatih ~