



MODUL #10

PEMROGRAMAN TERSTRUKTUR

LOOP (PENGULANGAN) PADA C++

NUR ALAMSYAH

PERNYATAAN PENGULANGAN (LOOP)

Pernyataan pengulangan (LOOP) merupakan salah satu cara untuk [memanipulasi aliran program](#), sehingga kita bisa membuat program yang fleksibel berdasarkan keinginan pengguna. Pernyataan pengulangan berfungsi untuk mengulangi beberapa *substatement* hingga kondisi terpenuhi. Di dalam bahasa pemrograman C++ kita bisa menggunakan pernyataan **for**, **WHILE** dan **DO-WHILE**.

Pengulangan (atau dalam bahasa inggris disebut dengan **loop**) adalah instruksi program yang bertujuan untuk mengulang beberapa baris perintah.

Perhatikan contoh program berikut:

pengulangan.cpp

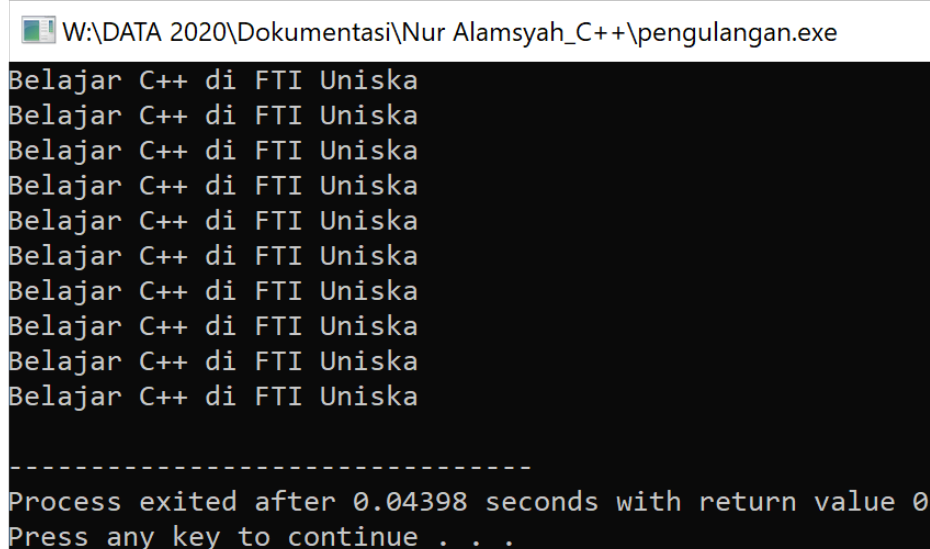
```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout<<"Belajar C++ di FTI Uniska"<<endl;
6      cout<<"Belajar C++ di FTI Uniska"<<endl;
7      cout<<"Belajar C++ di FTI Uniska"<<endl;
8      cout<<"Belajar C++ di FTI Uniska"<<endl;
9      cout<<"Belajar C++ di FTI Uniska"<<endl;
10     cout<<"Belajar C++ di FTI Uniska"<<endl;
11     cout<<"Belajar C++ di FTI Uniska"<<endl;
12     cout<<"Belajar C++ di FTI Uniska"<<endl;
13     cout<<"Belajar C++ di FTI Uniska"<<endl;
14     cout<<"Belajar C++ di FTI Uniska"<<endl;
15
16     return 0;
17 }
```

Contoh program diatas adalah untuk menampilkan kelayar tulisan "**Belajar C++ di FTI Uniska**" sebanyak 10 kali. Bagaimana kalo ingin menampilkan tulisan tersebut sebanyak 100 kali, 1000 kali 10.000 kali atau sejuta kali?? Jika kita tidak menggunakan Loop (Pengulangan) maka betapa panjangnya code yang harus kita buat.

Maka **loop** (pengulangan) ini diperlukan untuk mengatasi permasalahan diatas, perhatikan code berikut:

```
pengulangan.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      char C;
6      for(C=0; C<10;C++){
7          cout<<"Belajar C++ di FTI Uniska"<<endl;
8      }
9      return 0;
10 }
```

Maka hasilnya akan sama seperti dibawah ini:



```
W:\DATA 2020\Dokumentasi\Nur Alamsyah_C++\pengulangan.exe
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
Belajar C++ di FTI Uniska
-----
Process exited after 0.04398 seconds with return value 0
Press any key to continue . . .
```

Dalam bahasa C++ terdapat tiga buah jenis struktur pengulangan, yaitu :

1.Struktur **while**

2.Struktur **do-while**

3.Struktur **for**

Perulangan Bersarang

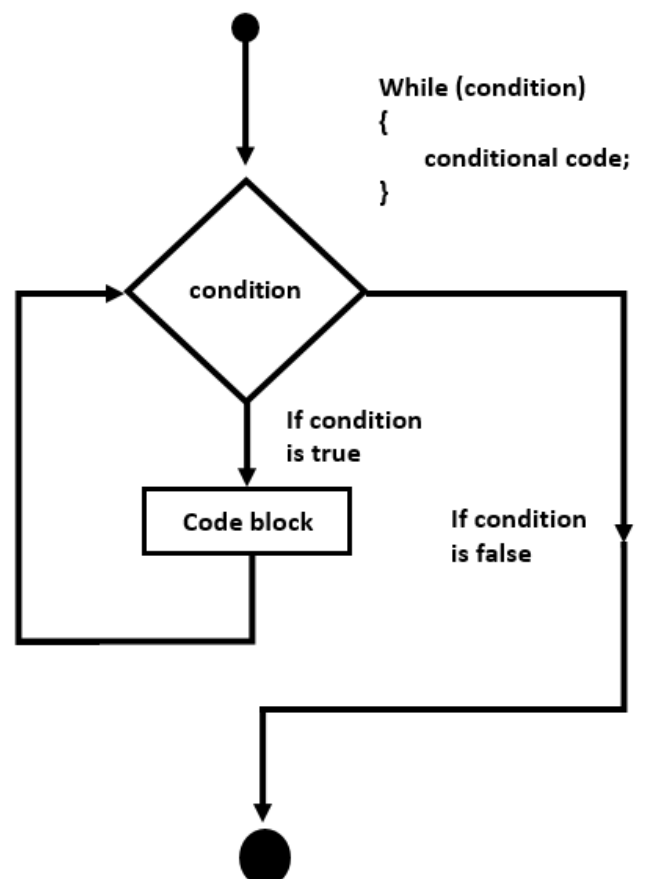
Perulangan bersarang adalah sebutan untuk **perulangan di dalam perulangan**. Konsep seperti ini sering dipakai untuk memecahkan masalah programming yang cukup kompleks. Semua jenis perulangan bisa dibuat dalam bentuk perulangan bersarang, termasuk perulangan **while**, **do while** dan **for**. Dalam bahasa inggris, perulangan bersarang ini dikenal dengan sebutan **nested loop**.

1. Struktur **while**

Struktur pengulangan jenis ini adalah pengulangan yang melakukan pemeriksaan kondisi diawal blok struktur.

Bentuk umum pengulangan **while**

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



Contoh program pengulangan **while**

Praktikkan: Buat file baru bernama **"while.cpp"** dan simpan difolder

```
while.cpp
1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      int C;
6      C = 0;
7
8      while (C<10) {
9          cout <<"Saya sangat menyukai C++"<<endl;
10         C++;
11     }
12     return 0;
13 }
```

Struktur **while** bersarang

Contoh program pengulangan **while bersarang**

Praktikkan: Buat file baru bernama **"whilebersarang.cpp"** dan simpan difolder

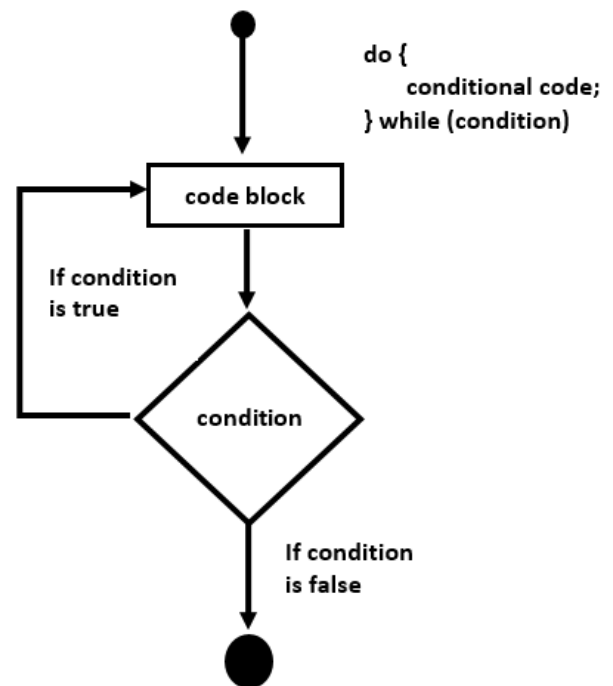
```
whilebersarang.cpp
1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      int j =10;
6      int k;
7
8      while (j>=1) {
9          k=1;
10         while (k<=j){
11             cout <<k*j<<' ';
12             k++;
13         }
14         cout <<endl;
15         j--;
16     }
17     return 0;
18 }
```

2. Struktur **do-while**

Berbeda dengan struktur **while** yang melakukan pemeriksaan kondisi diawal blok perulangan, pada struktur **do-while** kondisi justru ditempatkan dibagian akhir. Hal ini menyebabkan struktur pengulangan ini minimal akan melakukan **satu kali proses pengulangan** walaupun kondisi yang didefinisikan tidak terpenuhi (bernilai salah).

Bentuk umum **pengulangan do-while**

```
do {  
    Statemen_yang_akan_diulang;  
} while (kondisi);
```



Contoh program pengulangan **do-while**

Praktikkan: Buat file baru bernama "dowhile.cpp" dan simpan difolder

```
dowhile.cpp  
1  #include<iostream>  
2  using namespace std;  
3  
4  int main(){  
5      int C = 0;  
6  
7      do {  
8          cout <<"Saya sangat menyukai C++" <<endl;  
9          C++;  
10     } while (C<10);  
11  
12     return 0;  
13 }
```


Struktur **do-while** bersarang

Contoh program pengulangan **dowhile** bersarang

Praktikkan:

Buat file baru bernama “*dowhilebersarang.cpp*” dan simpan difolder

```
dowhilebersarang.cpp
1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      int j =10;
6      int k;
7
8      do {
9          k=1;
10         while (k<=j){
11             cout <<k*j<<' ';
12             k++;
13         }
14         cout <<endl;
15         j--;
16     } while (j>=1);
17     return 0;
18 }
```

3. Struktur **for**

Struktur pengulangan jenis ini biasanya digunakan untuk melakukan pengulangan yang telah diketahui banyaknya. Kita harus memiliki variabel sebagai indeksnya.

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',...)**

Bentuk umum **for** ada dua Jenis yaitu **increment** (naik satu nilai) dan **decrement** (turun satu nilai)

```
// Untuk pengulangan yang sifatnya menaik (increment)
for (variabel =nilai_awal;kondisi;variabel++) {
    Statemen_yang_akan_diulang;
}

// Untuk pengulangan yang sifatnya menurun (decrement)
for (variabel =nilai_awal;kondisi;variabel - - ) {
    Statemen_yang_akan_diulang;
}
```

Contoh program **for increment (naik satu nilai)**

Praktikkan: Buat file baru bernama “for_increment.cpp” dan simpan difolder

```
for_increment.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      char A;
6      for(A=0; A<20;A++){
7          cout<<"Belajar C++ di FTI Uniska"<<endl;
8      }
9      return 0;
10 }
```


Contoh program **for decrement (turun satu nilai)**

Praktikkan: Buat file baru bernama "for_decrement.cpp" dan simpan difolder

```
for_decrement.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      char A;
6      for(A=20; A>0;A--){
7          cout<<"Belajar C++ di FTI Uniska"<<endl;
8      }
9      return 0;
10 }
```

Apabila masih merasa bingung tentang perbedaan antara pengulangan yang sifatnya menaik dan menurun, coba perhatikan program dibawah ini:

Praktikkan: Buat file baru bernama "for_increment_decrement.cpp" dan simpan difolder

```
for_increment_decrement.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout<<"PENGULANGAN INCREMENT (NAIK SATU NILAI)"<<endl;
6      for(int A=0; A<10;A++){
7          cout<< A <<endl;
8      }
9      cout<<endl;
10     cout<<"PENGULANGAN DECREMENT (TURUN SATU NILAI)"<<endl;
11     for(int B=10; B>00;B--){
12         cout<< B <<endl;
13     }
14     return 0;
15 }
```

Struktur **for** bersarang

Contoh program **for bersarang**

Praktikkan: Buat file baru bernama "**for_bersarang.cpp**" dan simpan difolder

```
for_bersarang.cpp
1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      for (int j=1; j<=10; j++){
6          for (int k=1; k<=j; k++) {
7              cout << k*j << ' ';
8          }
9          cout<<endl;
10     }
11     return 0;
12 }
```