MODUL 4 Pernyataan Berulang

4.1 Tujuan Praktikum

Setelah praktikum pada modul 1 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

1) Mengetahui penggunaan pernyataan berulang FOR
2) Mengetahui penggunaan pernyataan berulang WHILE
3) Mengetahui penggunaan pernyataan berulang DO-WHILE
4) Dapat membedakan penggunaan FOR, WHILE dan DO-WHILE.

4.2 Materi

Pernyataan perulangan adalah pernyataan yang akan menjalankan pernyataan yang mengikutinya secara berulang sampai syarat tertertentu terpenuhi. Ada 3 buah pernyataan perulangan, yaitu FOR, WHILE dan DO-WHILE.

4.2.1 Pernyataan FOR

Statement FOR adalah suatu perintah yang berfungsi untuk melakukan proses pengulangan, dimana jumlah pengulangannya sudah diketahui. Bentuk dari statement FOR adalah:

```
for ( init; condition; increment )
{
  statement(s);
}
```

Tahapan proses pada pernyataan FOR adalah sebagai berikut:

- Tahapan **init** dieksekusi pertama kali. Pada **init** terdapat variabel kontrol dan nilai awalnya.
- Nilai variabel control akan bertambah dan berkurang sesuai dengan nilai **increment**.
- Setiap nilai variabel control berubah, proses berikutnya adalah mengevaluasi **condition**. Jika **condition** bernilai benar, maka statement(s) akan dijalankan. Sebaliknya jika **condition** bernilai salah, maka statement(s) tidak akan dijalankan.

Contoh program:

```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5   // for loop execution
6   for( int a = 10; a < 20; a = a + 1 ) {
7     cout << "value of a: " << a << endl;
8   }
```

```
 9
10   return 0;
11 }
```

Contoh program:

```
 1 #include <iostream>
 2 using namespace std;
 3 int main ()
 4 {
 5    for (int i = 1; i <= 10; i++)
 6     { cout << i << '\n'; }
 7 }
```

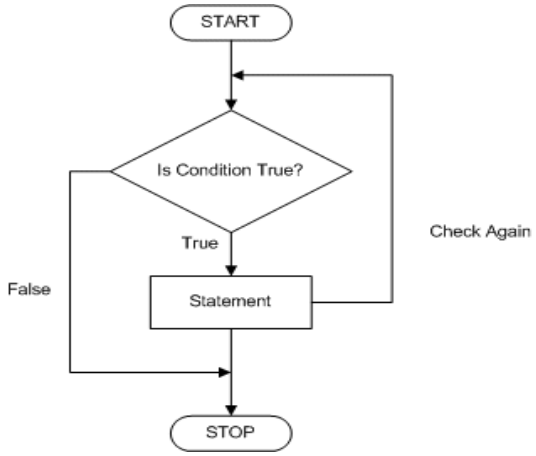
CONTOH :

4.2.2 Pernyataan While

Statement WHILE adalah suatu perintah yang berfungsi untuk melakukan proses pengulangan, dimana pengulangan akan terus dilakukan jika kondisi tertentu dipenuhi. Jika banyaknya pengulangan tidak diketahui maka statement FOR tidak dapat digunakan, sehingga harus digunakan statement WHILE atau DO-WHILE. Bentuk dari statement WHILE adalah:

```
while (syarat_logika)
{
        pernyataan;
}
```

Statement WHILE berfungsi untuk melakukan proses pengulangan eksekusi pernyataan, dimana pengulangan akan berhenti jika syarat_logika bernilai False atau Salah.



Dalam proses pengulangan selalu ada kondisi yang mengkontrol pengulangan tersebut untuk berhenti. Kontrol pengulangan tidak selalu melibatkan variabel penghitung.

Contoh:

Program untuk menampilkan karakter yang dimasukkan lewat keyboard secara berulang-ulang hingga karakter '*' dimasukkan.

```
1  #include <iostream>
2  using namespace std;
3
4  int main ()
5  {
6    // Local variable declaration:
7    char x ='a';
8
9    // while loop execution
10   while( x != '*')
11  {
12      cout << "Masukkan huruf: " << endl;
13      cin >> x;
14   }
15
16   return 0;
17 }
```

Statement WHILE dapat digunakan untuk pengulangan dengan jumlah pengulangan yang diketahui.

CONTOH:

```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5    // Local variable declaration:
6    int a = 10;
7
8    // while loop execution
9    while( a < 20 ) {
10     cout << "value of a: " << a << endl;
11     a++;
12   }
13
14   return 0;
15 }
```

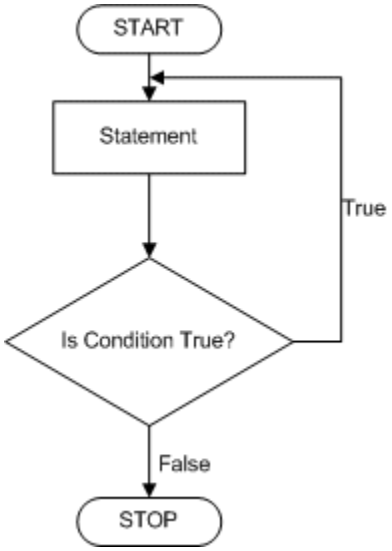
Hasil:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

4.2.3 Pernyataan Do-While

Statement Do-While adalah suatu perintah yang berfungsi untuk melakukan proses pengulangan eksekusi pernyataan, dimana pengulangan akan berhenti jika syarat_logika yang ada di akhir pengulangan bernilai salah atau False.

```
do {
  pernyataan;
}
while (syarat_logika);
```




```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5
```

```
   int a = 10;

   do {
      cout << "value of a: " << a << endl;
      a = a + 1;
   } while( a < 20 );

   return 0;
}
```

4.2.4 Pernyataan Nested FOR

Untuk menangani masalah tertentu, ada kemungkinan statement FOR yang digunakan lebih dari satu dan statement FOR yang satu dapat berada didalam statement FOR yang lain. Bentuk yang seperti ini disebut FOR Bersarang (nested FOR)

Contoh :

Ingin dibuat program untuk menampilkan bentuk seperti berikut ini:

```
*
**
***
****
```

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

int main () {

  for( int i = 1; i <= 4; i = i + 1 )
  {
    for( int j = 1; j <= i; j = j + 1 )
          {cout <<"*";}
        cout <<endl;
  }

  return 0;
}
```

4.2.5 Penggunaan Statement FOR Dalam Persamaan yang Mengandung Simbol ∑ atau ∏

Untuk menangani persamaan matematika/statistika yang mengandung simbol ∑ atau ∏, perlu bantuan statement pengulangan. Diberikan bentuk persamaan sebagai berikut:

$$hasil = \sum_{i=1}^{n} (i)$$

Dari bentuk tersebut diatas, terlihat bahwa ada 3 komponen penting yaitu:

- Hasil penjumlahan disimpan di variable “hasil”
- Yang masuk kedalam sigma adalah (i)
- Variabel penghitung (indeks) adalah i dan berjalan mulai 1 sampai dengan n

Bentuk pengulangan dengan menggunakan FOR, adalah:

int n=10;
int hasil=0;
 	
For(int i = 1; i <= n; i = i + 1)
{hasil = hasil + (i) ;}
Cout << hasil;
	N diberi harga tertentu
	hasil diberi harga awal 0 karena bentuk Sigma
	Variable penghitung I berjalan mulai 1 hingga N
	Mulai pengulangan
	$hasil_{baru}$= $hasil_{lama}$+{persamaan yang masuk dalam sigma}
	Akhir pengulangan
	Cetak nilai Y

Untuk persamaan:

$$hasil = \prod_{i=1}^{n} (i)$$

Dari bentuk tersebut diatas, terlihat bahwa ada 3 komponen penting yaitu:

- Hasil perkalian disimpan di variable hasil
- Yang masuk kedalam phi adalah (i)
- Variabel penghitung (indeks) adalah I dan berjalan mulai 1 sampai dengan N

Bentuk pengulangan dengan menggunakan FOR, adalah:

Int n=10; Int hasil=1; For(int i = 1; i <= n; i = i + 1) {hasil = hasil * i;} Cout << hasil;	N diberi harga tertentu hasil diberi harga awal 1 karena bentuk Phi Variable penghitung I berjalan mulai 1 hingga N Mulai pengulangan $hasil_{baru}$= $hasil_{lama}$ * {persamaan yang masuk dalam sigma} Akhir pengulangan Cetak nilai Y

4.3 Latihan

1. Buat program untuk menampilkan deret (sampai suku ke-n) : 1 3 9 27 .....

2. Buat program yang pada awalnya meminta data lebar dan tinggi, lalu buat gambar persegipanjang berdasarkan data lebar dan tinggi. Misalkan lebar=10 dan tinggi=4, maka gambar yang dihasilkan adalah:

```
**********
*        *
*        *
**********
```

3. Buat tampilan berikut ini:

a.	b.	c.	d.
111	1	1	1
222	22	12	21
333	333	123	321
444	4444	1234	4321

4. Buat program untuk menghitung:
   a. a^b
   b. n!
   c. menghitung rata-rata

Catatan : Untuk semua soal masing-masing dikerjakan dengan menggunakan statement FOR, WHILE dan REPEAT.