MODUL 4

LOOPING (PERULANGAN)

A. TUJUAN

Setelah mempelajari bab operasi kondisi / struktur kontrol siswa diharapkan mampu

1. Memahami dan menjelaskan penggunaan statement perulangan
2. Mampu membuat program sederhana dengan menerapkan konsep perulangan

B. PETUNJUK

1. Berdoa sebelum memulai aktivitas
2. Baca dan pahami tujuan, dasar teori, dan latihan praktikum
3. Kerjakan tugas praktik dengan baik
4. Tanyakan kepada guru jika ada hal yang kurang jelas

C. ALAT DAN BAHAN

1. PC/LAPTOP
2. Software c++
3. Modul

D. DASAR TEORI

Perulangan (looping) merupakan fasilitas untuk melakukan proses yang berulang-ulang sebanyak keinginan pembuat program. Contoh: andi ingin menginputkan data dan mencetaknya sebanyak 100 kali, jika tidak menggunakan perulangan maka andi perlu menuliskan perintah untuk menginput dan mencetak sebanyak 100 kali, tapi jika menggunakan perulangan andi hanya perlu menuliskan beberapa perintah saja.

1. Pernyataan **for**

   Digunakan ketika mengulangi perintah dengan jumlah perulangan yang sudah diketahui.

   Bentuk umum:

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

   Jika pernyataan di dalam for lebih dari satu maka pernyatan-pernyataan tersebut harus diletakkan di dalam tanda kurung.

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )
{
    pernyataan / perintah;
    pernyataan / perintah;
    pernyataan / perintah;
}
```

   Kegunaan dari masing-masing argumen for di atas:

   - Inisialisasi : bagian untuk memberikan nilai awal untuk variabel-variabel tertentu
   - Syarat perulangan : memegang kontrol terhadap pengulangan, karena bagian ini yang akan menentukan suatu perulangan diteruskan atau dihentikan.

- Pengubah nilai pencacah : mangatur kenaikan atau penurunan nilai pencacah.

Contoh:

```
for(a=1;a<=5;a++)
{
cout<<"Hello World! \n"
}
```

Perintah di atas menampilkan kalimat "hello word" sebanyak 5 baris

Tanda a=1 apa artinya?
Ubah tanda a<=5 menjadi a<5 apa yang terjadi?
Tanda a++, apa maksudnya?

Selain berupa angka, pencacah perulangan juga dapat berupa karakter.

Contoh:

```
for(huruf=.Z.;huruf>=.A.;huruf--)
{
Cout<<Abjad ¡±<<huruf<<¡±\n¡±;
}
```

Perintah di atas menampilkan abjad Z-A

Perhatikan perintah operator --

Contoh lain:

```
for (angka = 1; angka <= 6; angka+=2)
{
cout << "Isi dari angka = " << angka << endl;
}
```

Perintah di atas akan menampilkan angka 1, 3, 5.
Mengapa terjadi demikian? Perhatikan perintah angka+=2!

Contoh lain:

```
#include <iostream.h>
#include <conio.h>
int main(){
int bil, n;
cout << "Masukkan n = ";
cin >> n;
for (bil = 0; bil < n; bil++)
{
if (bil % 2 == 0) cout << bil << " ";
}
}
```

a) Latihan 1

```
/* -------------------------- */
/* Program for - bilangan naik */
/* -------------------------- */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
  int a;
  clrscr();
  for(a = 1; a <= 10; ++a)
     cout<<a;

  getch();
}
```

Jelaskan proses perulangannya, beda nya a++ dengan ++a

b) Latihan 2

```
/* -------------------------- */
/* Program for - bilangan turun */
/* -------------------------- */
# include <stdio.h>
# include <conio.h>
# include <iostream.h>

main()
{
  int a;
  clrscr();
  for(a = 10; a >= 1; --a)
     cout<<a;

  getch();
}
```

Jelaskan proses perulangannya, beda nya a-- dengan --a

c) Latihan 3

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

main()
{
    int a;
    for (a = 1; a<=10; a+=2)
        cout<<a;
    getch();
}
```

Jelaskan proses perulangannya dan perbedaan output dari latihan sebelumnya.

2. Pernyataan **WHILE**

Perintah atau instruksi digunakan jika jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (≠0) dan akan berhenti jika kondisinya bernilai salah (=0).

Bentuk umum perulangan **while**

```
while ( syarat )
   Pernyataan / perintah ;
```

Bentuk perulangan **while,** dengan lebih dari perintah / pernyataan,

```
while ( syarat )
{
  Pernyataan / perintah ;
  Pernyataan / perintah ;
}
```

Dua perintah di bawah ini identik

```
for (a = 1; a <= 5; a++)
{
cout << "Hello world \n";
}
```

```
a = 1;
while (a <= 5){
cout << "Hello world \n";
a++;
}
```

Jika Anda menggunakan WHILE, pastikan bahwa suatu saat bagian kondisi sampai bernilai FALSE. Apabila tidak, proses perulangan akan terus berjalan selamanya.

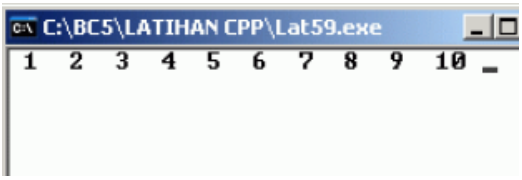
Contoh:

```
#include <iostream.h>
#include <conio.h>
int main()
{
int data, jumlah,cacah;
jumlah = 0;
data = 0;
cacah = 0;
while (data != -1)
{
cout << "Masukkan data angka : ";
cin >> data;
jumlah += data;
cacah++;
}
cout << "Jumlah data adalah : " << jumlah << endl;
cout << "Rata-rata : " << jumlah/cacah;
}
```

program di atas digunakan untuk menjumlahkan sejumlah data angka. Angka yang akan dijumlahkan diinputkan satu-persatu. Proses pemasukan data angka akan berhenti ketika dimasukkan angka –1. Setelah itu tampil hasil penjumlahannya.

a) Latihan 1

```
/* ------------------- */
/* Program while01.cpp */
/* ------------------- */
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
main()
{
   int bil=1;
   clrscr();
   while(bil<=10)
   {
      cout<<bil<<" ";
      ++bil;
   }
   getch();
}
```



Jelaskan proses perulangannya

b) Latihan 2

```
/* ------------------- */
/* Program while02.cpp */
/* ------------------- */
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

main()
{
   int bil=2;
   clrscr();
   while(bil<=10)
   {
      cout<<bil<<" ";
      bil+=2;
   }
  getch();

}
```

Jelaskan proses perulangannya

3. Pernyataan **do-while**

Perintah DO ... WHILE hampir sama dengan WHILE sebelumnya. Gambaran secara umum:

Perbedaan dengan **WHILE** sebelumnya yaitu bahwa pada **DO WHILE** statement perulangannya dilakukan terlebih dahulu baru kemudian di cek kondisinya. Sedangkan **WHILE** kondisi dicek dulu baru kemudia statement perulangannya dijalankan. Akibat dari hal ini adalah dalam **DO WHILE** minimal terdapat 1x perulangan. Sedangkan **WHILE** dimungkinkan perulangan tidak pernah terjadi yaitu ketika kondisinya langsung bernilai **FALSE**.

Contoh:

```
a = 1;
do
{
cout << "Hello world \n";
a++;
}
while(a==0)
```

Perintah di atas akan muncul satu buah Hello Word, bandingkan dengan

```
a = 1;
while(a==0)
{
cout << "Hello world \n";
a++;
}
```

Perintah di atas sama sekali tidak menampilkan Hello World, karena kondisinya langsung **FALSE**.

a) Latihan 1

```
/* ------------------ */
/* Program do - while */
/* ------------------ */
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
main()
{
   int bil=2;
   clrscr();
   do
   {
      cout<<bil<<" ";
      bil+=2;
   }
   while(bil<=10);
   getch();
}
```

Jelaskan proses perulangannya dibandingkan dengan latihan yang sebelumnya.

4. Pernyataan **Nested-For**
   Adalah suatu perulangan for didalam perulangan for lainnya. Bentuk pernyataan nya adalah seperti berikut

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )
{
  for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah)
  {
        pernyataan / perintah;
  }
}
```

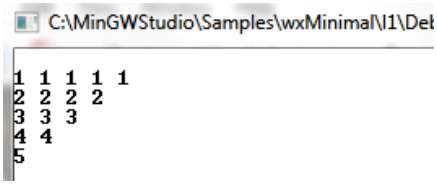
   Di dalam perulangan nested-for, perulangan yang di dalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang di luar diselesaikan.

   Contoh:

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

void main()
{
int a,b;
for (a=1; a<=5; a++)
{
    cout<<"\n";
    for (b=a; b<=5; b++)
    cout<<a<<" ";
}
getch ();
}
```

   Output yang dihasilkan adalah

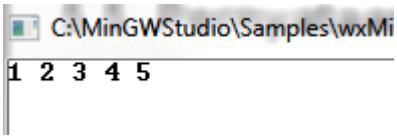


5. Pernyataan **Break**
   Pernyataan break sudah dibahas pada pernyataan pengambilan keputusan *switch.* Pernyataan ini berfungsi untuk keluar dari struktur *switch*. Selain itu pernyataan *break* berfungsi untuk keuar dari perulangan (for, while, dan do-while). Jika pernyataan *break* dikerjakan maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir dari badan perulangan

   Contoh

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

main()
{
int bil = 1;
do
{
    if (bil >=6)
break;
    cout<<bil<<" ";
}
while (bil++);
getch();
}
```



6. Pernyataan **continue**

   Digunakan untuk mengarahkan eksekusi ke iterasi(proses) berikutnya pada loop yang sama, dengan kata lain mengembalikan proses yang sedang dilaksanakan ke-awal loop lagi, tanpa menjalankan sisa perintah dalam loop tersebut.

   Contoh:

```
//Program yang menampilkan bilangan ganjil antara 1 - 20
#include <stdio.h>
main(){
  int i;

 for(i = 1; i <= 20 ; i++){
   if(i%2==0)
    continue;
   printf("%d \t",i);
 }
}
```

   Tampilan program

```
1       3       5       7       9       11      13      15      17      19

Terminated with return code 0
Press any key to continue ...
```

   Sederhananya continue berfungsi untuk melewatkan semua perintah yang berada setelahnya dan langsung melompat ke bagian kondisi perulangan.

E. Tugas PRAKTIKUM

1.

Buatlah program untuk menghitung 10 deret bilangan genap dengan hasilnya :

```
2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 + 18 + 20 = 110
```

2.

Buatlah program untuk menghitung penjumlahan deret bilangan membentuk segitiga siku dengan hasilnya :

```
1                       = 1
1 + 2                   = 3
1 + 2 + 3               = 6
1 + 2 + 3 + 4           = 10
1 + 2 + 3 + 4 + 5       = 15
```

3. Tampilkan bentuk „*“ seperti contoh program sebelumnya, hanya saja tampilannya dibalik , dari kolom besar ke kolom kecil, dengan ketinggian tertentu.
   Contoh tampilan :

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