PERTEMUAN 3
CONTROL STATEMENT DENGAN DO .. WHILE ...
DAN
NESTED LOOP

A. Tujuan Pembelajaran

1. Mampu menerapkan perulangan menggunakan bentuk do ... while …
2. Mampu menerapkan perulangan dalam perungan atau nested loop

B. Uraian Materi

1.1 Do ... While

Intruksi do ... while ... merupakan intruksi perulangan yang sama dengan intruksi perulangan while. Perbedaan pada while dan do .. while adalah pada intruksi while kondisi akan di evaluasi atau di uji terlebih dahulu sebelum intruksi perulangan di kerjakan, sedangkan pada intruksi do ... while kondisi akan di evaluasi atau di uji setelah intruksi perulangan dikerjakan.

Syntax Perulangan Menggunakan **do ... while ...**

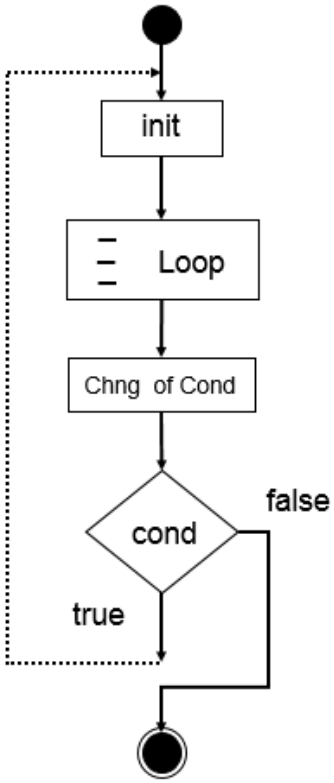
```
Init;
do
{
    - Statement 1
    - Statement 2
    -
    -
    - Statement n
      Change Of Cond;
}while (cond);
```

Cara kerja:

1. Melakukan inisiaslisai awal atau memberikan nilai awal
2. Mengerjakan looping dan mengerjakan
3. Memeriksa kondisi pada intruksi while
   a. Jika kondisi benar, maka mengerjakan looping kembali
   b. Jika kondisi salah, maka keluar dari looping

Init → digunakan untuk inisialisasi kondisi, atau melakukan pemberian nilai awal

cond → suatu pernyataan atau ungkapan yang mengandung nilai **benar (true)** atau **salah (false)**



Gambar 1. Flowchar do ... while ...

Contoh penggunaan do ... while ...

Listing 11.1 : dowhile.cpp

```
1   #include<iostream>
2   using namespace std;
3   int main()
4   {
5         int i = 1;
6         do
7         {
8               cout<< i;
9               i++;
10        }while(i <= 4);
11  }
```

Output : 1 2 3 4 5

Nilai (i)	tercetak	i++	Kondisi
1	1	2	True
2	2	3	True
3	3	4	True
4	4	5	False

Dimulai dari inisilasasi, di mana nilai i di isi 1, kemudian langsung melaksanakan intruksi yang ada pada looping sehingga di cetak 1, kemudian melaksanakan intruksi (i++) yaitu nilai i tambah 1 dilanjutkan dengan memeriksa kondisi (i <=5) karena nilai i lebih kecil dari 5 maka menghasilkan nilai benar. Intruksi akan terus berulang sampai dengan nilai kondisi bernilai salah baru akan keluar dari looping.

Contoh penggunaan do ... while ...

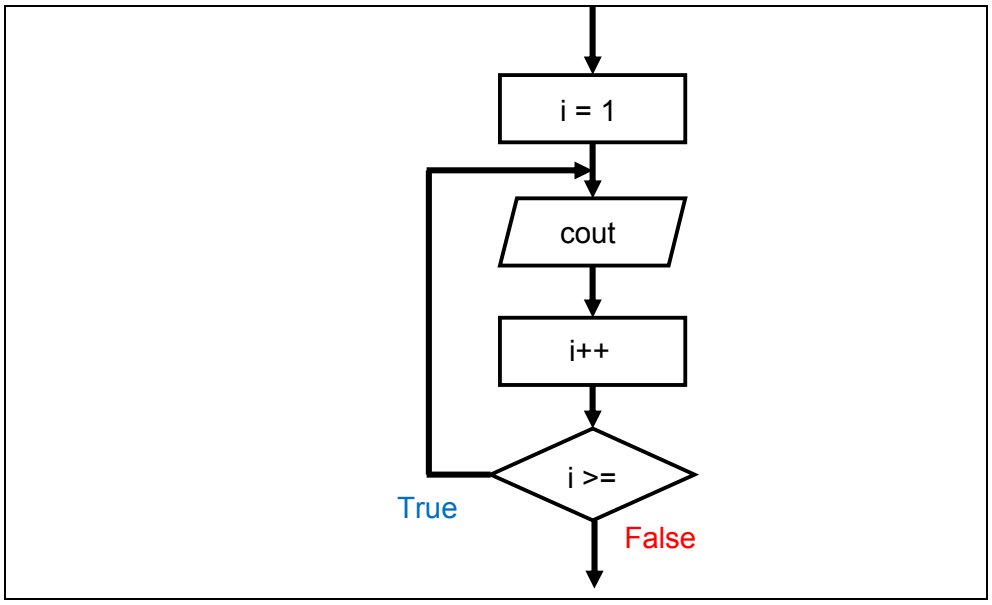
Listing 11.2 : dowhile.cpp

```
1   #include<iostream>
2   using namespace std;
3   int main()
4   {
5         int i = 1;
6         do
7         {
8               cout<< i;
9               i++;
10        }while(i >= 3);
11  }
```

Output : 1

Nilai (i)	tercetak	i++	Kondisi
1	1	2	False

Dari program diatas minimal akan mencetak 1, karena akan melakukan looping atau intruksi yang ada pada looping akan dikerjakan terlebih dahulu kemudian baru akan memeriksa kondisi yang ada pada intruksi while



1.2 Loop dalam Loop (Nested Loop)

Intruksi loop yang berada dalam intruksi loop yang lain di kenal dengan nested loop atau loop tersarang, untuk ilustrasi perhatikan gambar berikut: Terdapat dua buah program, yaitu **program A** dan sebuah penggalan **program B**:

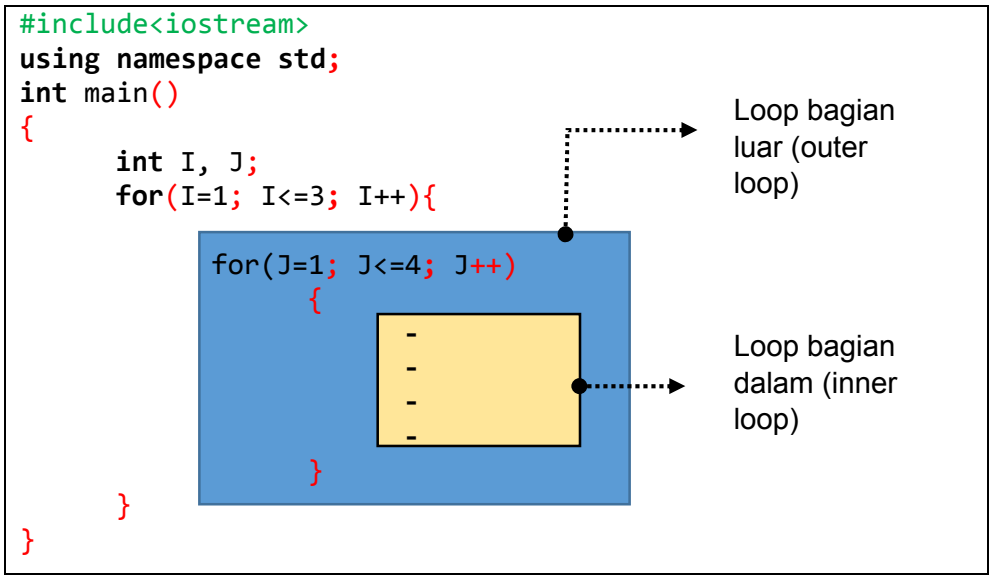
Program A	Penggalan Program B

`#include<iostream>`
`using namespace std;`
`int main()`
`{`
`int I, J;`
`for(I=1; I<=3; I++){`
`-`
`-`
`Loop`
`-`
`-`
`1`
`}`
`}`
	`for(J=1; J<=4; J++)`
	`{`
	`-`
	`-`
	`Loop`
	`-`
	`-`
	`}`

Loop program A akan dikerjakan sebanyak 3 kali	Loop Program B akan dikerjakan sebanyak 4 kali

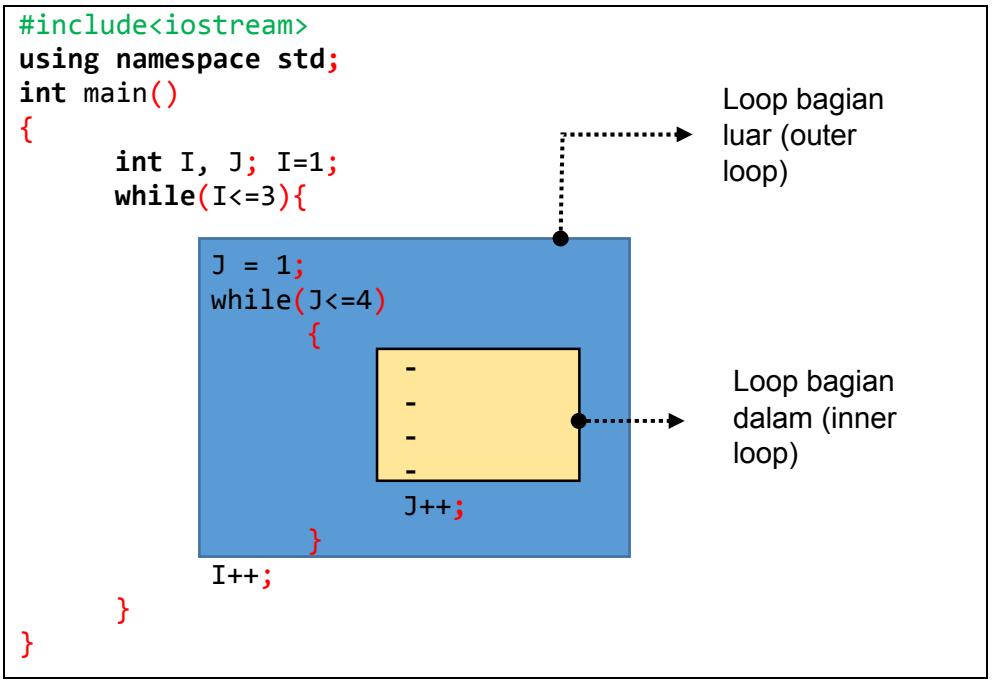
Pada program di atas **program A** terdapat loop menggunakan for yang dikerjakan sebanyak 3 kali, sedangkan pada penggalan **program B** terdapat loop menggunakan for yang dikerjakan sebanyak 4 kali. Apabila penggalan program B dipindahkan atau dimasukkan ke dalam program A menggantikan **kotak 1**, maka akan terbentuk loop di dalam loop, biasanya

disebut dengan **NESTED Loop** atau **loop tersarang**, menjadi program yang baru sebagai berikut:



Pada program di atas terdapat loop di dalam loop di mana pada outer loop (loop luar) intruksi loop akan dilaksanakan sebanyak 3 kali sedangkan pada inner loop(loop dalam) intruksi loop akan dilaksanakan sebanyak 4 kali, sehingga program di atas jika dijalankan akan melaksanakan loop sebanyak 3x4 = 12 kali loop.

Jika bentuk nested loop dituliskan menggunakan intruksi while maka akan terbentuk sebagai berikut:



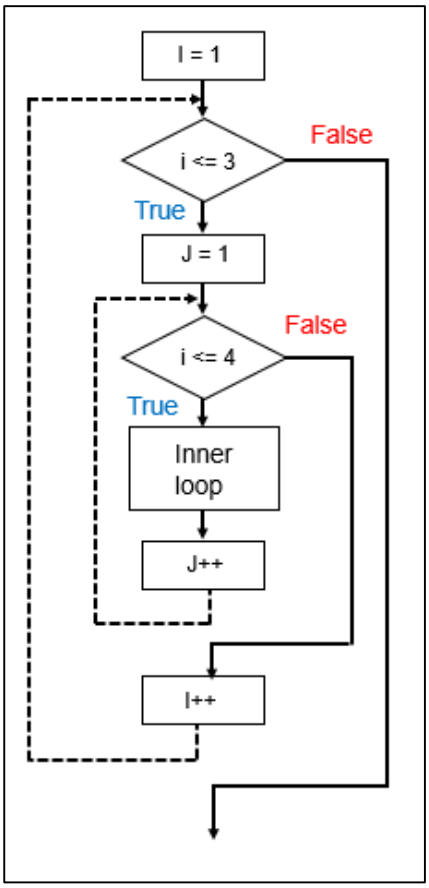
Jika bentuk nested loop dituliskan menggunakan gabungan while dan for maka akan terbentuk nested loop sebagai berikut:

While untuk outer loop dan for untuk inner loop

```
#include<iostream>
using namespace std;
int main()
{
      int I, J;
      while(I<=3){
            for(J=1; J<=4; J++)
                  {
                        -
                        -
                        -
                        -
                  }
            I++;
      }
}
```

For untuk outer loop dan while untuk inner loop

```
#include<iostream>
using namespace std;
int main()
{
      int I, J;
      for(I=1; I<=3; I++){
            J = 1;
            while(J<=4)
                  {
                        -
                        -
                        -
                        -
                        J++;
                  }
      }
}
```



Gambar 11. 1 Flowchart Nested Loop

1.3 Contoh – Contoh Penggunaan Nested Loop

Contoh 1: Loop 2 x 3

```
int I, J;
for(I=1; I<=2; I++){
   for(J=1; J<=3; J++)
        {
            cout<<"\nUnpam";
        }
}
```

Akan tercetak Unpam sebanyak 6 baris:

Unpam
Unpam
Unpam
Unpam
Unpam
Unpam

Perhatikan tabel berikut, tabel berikut memerlihatkan perkembangan/perubahan nilai I dan J sebagai pembentuk loop.

I	J	Cetak
1	1	Unpam
	2	Unpam
	3	Unpam

	4	Keluar dari loop dalam dan kembali ke loop luar
2	1 2 3 4	Unpam Unpam Unpam Keluar dari loop dalam dan kembali ke loop luar
I = 3 → keluar dari loop luar (*outer loop*)		

Pelaksanaan intruksi dimulai dari inisilisasi **nilai I = 1**, **I** di beri nilai **1** kemudian dilanjutkan dengan masuk ke loop luar (*outer loop*) dan memeriksa **kondisi I <= 2**, jika kondisi **benar** maka akan inisialisasi **nilai J=1** dan masuk kedalam loop dalam (inner loop) kemudian memeriksa **kondisi J <= 3**, jika kondisi bernilai **benar** maka akan melaksanakan intruksi looping yang ada pada loop dalam (*inner loop*) yang di ulang sebanyak 3 kali untuk mencetak perkataan **Unpam**, dan ketika **nilai J = 4** maka akan keluar dari loop dalam (*inner loop*) dan kembali ke loop luar (*outer loop*). Proses ini terus berulang sampai dengan kondisi **nilai I = 3** maka akan keluar dari outer loop (*loop luar*) dan looping berhenti secara keseluruhan.

Contoh 2.a: loop 2 x 3

```
int I, J;
for(I=1; I<=2; I++){
   for(J=1; J<=3; J++)
        {
           cout<<<J<"\n";
        }
}
```

Tercetak:
1
2
3
1
2
3

Hasil cetakan akan menurun ke bawah karena ada "**\n**", "**\n**" digunakan untuk intruksi pindah baris. Setiap kali setelelah mencetak nilai J, maka akan pindah baris.

Contoh 2.b: loop 2 x 3

```
int I, J;
   for(I=1; I<=2; I++){
      for(J=1; J<=3; J++)
      {
         cout<<J;
      }
}
```

Tercetak: 1 2 3 1 2 3

Hasil cetakan akan kesamping, karena tidak ada "**\n**" yang identik dengan new line atau intruksi pindah baris.

Contoh 2.c : loop 2 x 3

```
int I, J;
for(I=1; I<=2; I++){
   for(J=1; J<=3; J++)
        {
          cout<<<J;
        }
   cout<<endl;
}
```

I	J
1	1 2 3
2	1 2 3

Tercetak
1 2 3
1 2 3

Setiap 3 kali mencetak nilai J, akan turun satu baris dengan intruksi

`cout<<endl;`

Contoh 2.d : loop 3 x 2

```
int I, J;
for(I=1; I<=3; I++){
   for(J=1; J<=2; J++)
        {
          cout<<<J;
        }
   cout<<endl;
}
```

I	J
1	1 2
2	1 2
3	1 2

Tercetak
1 2
1 2
1 2

Setiap 2 kali mencetak nilai J, akan turun satu baris dengan intruksi

`cout<<endl;`

Contoh 2.e : loop 2 x 3

```
int I, J;
for(I=1; I<=2; I++){
   for(J=1; J<=3; J++)
        {
          cout<<<I;
        }
   cout<<endl;
}
```

I	J
1	1 1 1
2	2 2 2

Tercetak
1 1 1
2 2 2

Yang di cetak adalah nilai I.

Contoh 3 : loop 4 x 4

```
int I, J;
for(I=1; I<=4; I++){
   for(J=I; J<=4; J++)
        {
          cout<<J<<" ";
        }
   cout<<endl;
}
```

I	J
1	1 2 3 4
2	2 3 4
3	3 4
4	4

Tercetak
1 2 3 4
2 3 4
3 4
4

Nilai awal dari J di buat menjadi nilai I (J = I),
Ketika nilai I = 1, nilai J yang di cetak mulai dari 1 – 4, ketika nilai I = 2, nilai j yang di cetak 2 – 4 , ketika nilai I = 3, nilai J yang di cek 3 – 4, ketika nilai I = 4, nilai J yang dicetak 4.

Contoh 4

```
int N, X, T, I, J;
X = 1; T = 0; N = 4;
for(I=1; I<6; I+=2){
   for(J=I; J<10; J+=3)
        {
          T = T + N;
          N = N + X;
          X = X + 2;
          cout<<T;
        }
   cout<<endl;
}
```

Tercetak : **4 9 17**
30 50 79
119 172

I	J	T = 0 T =T+N	N = 4 N =N+X	X = 1 X =X+2
1	1 4 7	**4** **9** **17**	5 8 13	3 5 7
3	3 6 9	**30** **50** **79**	20 29 40	9 11 13
5	5 8	**119** **172**	53 68	15 17

Perhatikan perubahan masing – masing nilai di atas,

Yang di cetak adalah nilai **T**

Contoh 5 : loop 5 x 3

```
int var, I, J;
var = 65;
for(I=1; I<6; I++){
  for(J=1; J<4; J++)
  {
    cout<<(char)var<<" ";
    var++;
  }
  cout<<endl;
}
```

Tercetak :

A B C
D E F
G H I
J K L
M N O

I	J
1	A B C
2	D E F
3	G H I
4	J K L
5	M N O

Nilai **65** adalah karakter **A** dalam **ASCII.**
Dengan intruksi:
`cout<<(char)var<<" ";`
yang dicetak adalah karakternya bukan nilainya angkanya.
65 = Karakter **A**
67 = Karakter **B**
68 = Karakter **C**
69 = Karakter **D**
Dan seterunya sampai dengan
79 = Karakter **O**

Contoh 6 :

```
int var, I, J;
var = 'A';
for(I=1; I<6; I++){
  for(J=1; J<I; J++)
  {
    cout<<var<<" ";
    var++;
  }
  cout<<endl;
}
```

Tercetak:

A
B C
D E F
G H I J
K L M N O

I	J
1	A
2	B C
3	D E F
4	G H I J
5	K L M N O

Pada loop dalam (inner loop), kondisi di buat **J < I**, sehingga jumlah yang di cetak sejumlah dengan nilai I yang sekarang.
Misal:
Ketika nilai I = 1 → yang di cetak hanya 1, yaitu karakter **A** yang bernilai sama dengan 65 nilai numeriknya.
Ketika nilai I = 2 → yang di cetak berjumlah 2, yaitu karakter **B** dan Karakter **C**.
Sampai dengan
Nilai I = 5 → yang di cetak berjumlah 5, yaitu Karakter **K, L, M, N, O**

Contoh 7 :

```
int var, I, J;
var = 'A';
for(I=1; I<6; I++){
  for(J=1; J<7-I; J++)
  {
    cout<<(char)var<<"
    ";
    var++;
  }
  cout<<endl;
}
```

Tercetak:

A B C D E
F G H I
J K L
M N
O

I	J
1	A B C D E
2	F G H I
3	J K L
4	M N
5	O

Pada loop dalam (inner loop), kondisi di buat **J < 7 - I**, sehingga jumlah yang di cetak sejumlah dengan 7 dikurangi dengan nilai I yang sekarang.
Misal:
Ketika nilai I = 1 → (**7 - 1 = 6**) karena batas nilainya kurang dari 6, sehingga yang di cetak sejumlah 5, yaitu karakter **A, B, C, D, E.**

C. Soal Latihan / Tugas

1. Apa yang tercetak bila penggalan program berikut di jalankan, serta berikan penjelasan dengan menggunakan tabel:

A.
```
int I, J;
for(I=0; I<=4; I+=2){
  for(J=1; J<=8; J+=3)
  {
    cout<<J<<" ";
  }
  cout<<endl;
}
```

B.
```
int I, J;
for(I=0; I<=4; I+=2){
  for(J=1; J<=8; J+=3)
  {
    cout<<I<<" ";
  }
  cout<<endl;
}
```

C.
```
int T = 0, I, J;
for(I=0; I<=4; I+=2){
  for(J=1; J<=8; J+=3)
  {
    cout<<T<<" ";
  }
  cout<<endl;
}
```

D.
```
int I, J;
for(I=0; I<=4; I+=2){
  for(J=1; J<=8; J+=3)
  {
    T = T + J;
  }
  cout<<T<<" "<<endl;
}
```

2. Susun program untuk mencetak nilai-nilai, sehingga tercetak sebagai berikut:

A. **1 2 3 4 5** **1 2 3 4 5** **1 2 3 4 5**	**B.** **1 1 1 1 1** **2 2 2 2 2** **3 3 3 3 3**
C. **1 1 1 1 1** **2 2 2 2** **3 3 3** **4 4** **5**	**A.** **1 2 3 4 5** **6 7 8 9** **10 11 12** **13 14** **15**

3. Seorang nasabah meminjam uang sebesar 10 juta rupiah, dengan bunga sebesar 1.5 persen dihitung dari sisa hutang. Setiap akhir bulan nasabah tersebut harus mencicil 10 persen dari saldo hutangnya.
   Susun program untuk mencetak daftar cicilan yang harus dibayar tiap akhir bulan sampai sisa hutangnya kurang dari 1 juta rupiah.

D. Bibliography

Deitel, P., & Deitel, H. (2014). *C++ How To Program* (9th ed.). United State of America: Pearson.

Kirch-Prinz, U., & Kirch-Prinz, U. (2002). *A Complete Guide to Programming in C++.* Sudbury: Jones and Bartlett Publishers.

S, R. A. (2018). *Logika Algoritma dan Pemrograman Dasar.* Bandung: Modula.

Sjukani, M. (2014). *Algoritma dan Struktur Data 1 dengan C, C++ dan Java* (Edisi 9 ed.). Jakarta: MItra Wacana Media.