

Perulangan While & Do While

Pengantar Komputer dan Software II

Konsep Dasar

Perulangan While dan Do While, masing - masing merupakan bentuk perulangan yang tujuannya untuk menuliskan dengan efisien sekumpulan aksi yang memiliki pola tertentu.

Perulangan While dan Perulangan Do While merupakan dua perulangan yang berbeda. Namun memiliki kemiripan, yaitu bagian kondisinya tidak harus ditentukan setelah berapa kali perulangan aksi akan berhenti, yang terpenting apakah masih memenuhi kondisinya atau tidak.

Bentuk umum perulangan While :

```
while ( ...kondisi ... ){  
...badan perulangan...  
};
```

Bentuk umum perulangan Do While :

```
do{  
...badan perulangan...  
} while ( ...kondisi... );
```

Penjelasan

- ❖ Pada perulangan while, akan dicek kondisi terlebih dahulu sebelum melakukan aksi pada badan perulangan. Setelah aksi dilakukan, kembali mengecek apakah kondisi masih terpenuhi. Jika masih, maka lakukan aksi pada badan perulangan, dan seterusnya hingga kondisi tidak terpenuhi lagi.
- ❖ Namun pada perulangan do while, akan dikerjakan dahulu satu kali aksi pada badan perulangan. Kemudian baru mengecek apakah kondisi terpenuhi, jika iya, maka lakukan aksi pada badan perulangan, lalu cek lagi kondisi dan seterusnya hingga kondisi tidak terpenuhi lagi.
- ✓ Jadi perbedaannya, pada do while, minimal program sudah menjalankan 1 kali aksi sebelum melakukan pengecekan kondisi.
- ✓ Tetapi pada while, bisa saja aksi tidak dijalankan apabila saat pertama kali pengecekan kondisi sudah tidak terpenuhi.

Bandingkan dengan perulangan for.

Pada kondisi perulangan for ada 3 bagian yaitu :

(indeks_mulai; batas_indeks; iterasi)

Pada perulangan for harus jelas indeks bergerak naik atau turun, lalu dengan adanya batas, maka akan terukur berapa kali perulangan aksi akan dilakukan.

Permasalahan pada perulangan For dapat diterapkan pada perulangan While atau mungkin Do While.

Tetapi tidak sebaliknya.

Contoh program perulangan While (cetak bintang):

```
#include<iostream>
using namespace std;
int main(){
    int i=0;
    while(i<=5){
        cout<<"*";
        i++;
    };
}
```

Tampilan :



Pertanyaan : mengapa bintangnya ada 6?
bukan 5?

Karena i dimulai dari 0 bertambah jadi 1, 2,
3, 4 dan terakhir di 5.

Sehingga banyaknya ada 6.

Jika tidak ada i++ maka nilai i akan tetap 0,
sehingga kondisi akan selalu terpenuhi
akibatnya bintang akan tercetak tanpa henti.

Contoh program perulangan Do While (cetak bintang):

```
#include<iostream>
using namespace std;
int main(){
    int i=0;
    do{
        cout<<"*";
        i++;
    }while(i<=5);
}
```

Tampilan :



Mengapa pada contoh ini hasilnya sama dengan While?

untuk kasus ini prosesnya :

i mulai dari 0, lakukan aksi, i bertambah menjadi 1, lalu cek apakah $i \leq 5$, karena sekarang $i=2$ maka lakukan aksi lagi, dan seterusnya.

Terhitung ada 6 kali melakukan aksi.

Sehingga bintangnya ada 6.

Modifikasi cetak bintang While (nilai awal i=8)

```
#include<iostream>
using namespace std;
int main(){
    int i=8;
    while(i<=5){
        cout<<"*";
        i++;
    };
}
```

Tampilan :



Tampilannya kosong, atau tidak menampilkan apa - apa.

Karena pada perulangan while dicek terlebih dahulu apakah kondisinya benar. Kita lihat bahwa nilai i mula - mula adalah 8. Sementara pada kondisi $i \leq 5$, jelas ini tidak terpenuhi, maka aksi tidak akan dijalankan.

Modifikasi cetak bintang Do While (nilai awal i=8)

```
#include<iostream>
using namespace std;
int main(){
    int i=8;
    do{
        cout<<"*";
        i++;
    }while(i<=5);
}
```

Tampilan :



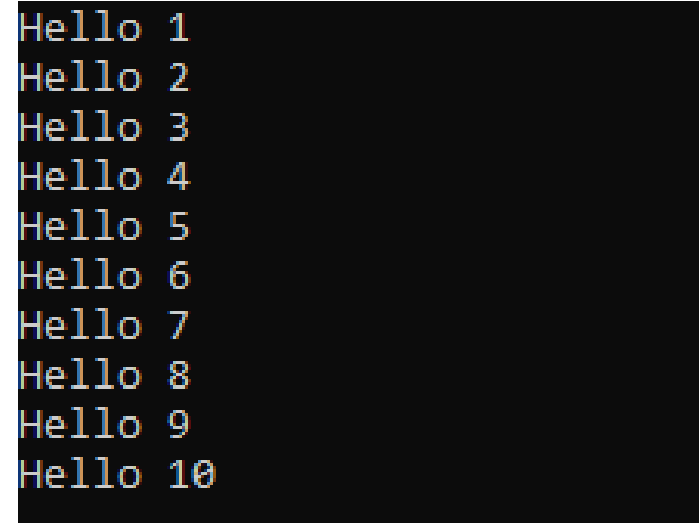
Tampilannya ada 1 bintang

Karena pada perulangan do while, yang penting dijalankan aksinya 1 kali dahulu sebelum melakukan pengecekan kondisi.

Program menampilkan tulisan berindeks sampai sebanyak bilangan tertentu

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    while(i<=10){
        cout<<"Hello "<<i<<endl;
        i++;
    };
}
```

Tampilan :

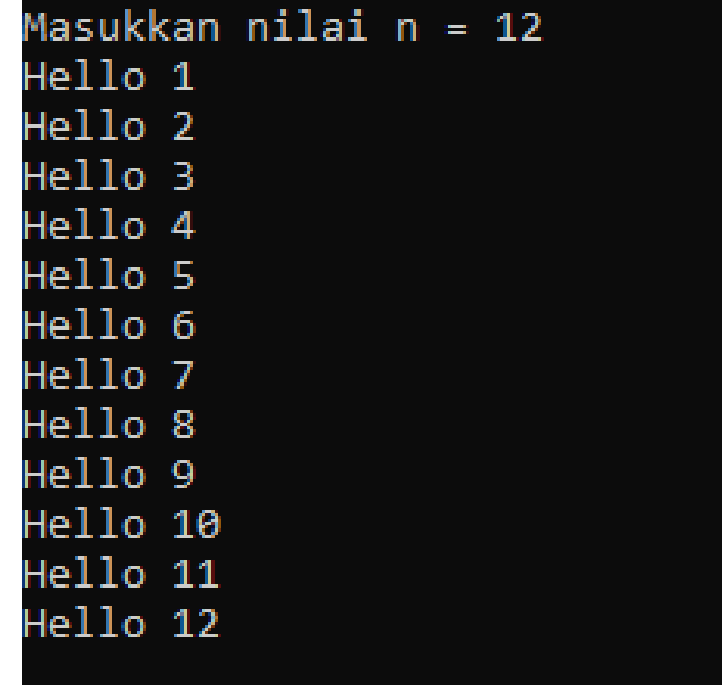


```
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
```

Modifikasi (banyaknya perulangan ditentukan dari input user)

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    int n;
    cout<<"Masukkan nilai n = ";
    cin>>n;
    while(i<=n){
        cout<<"Hello "<<i<<endl;
        i++;
    };
}
```

Tampilan :

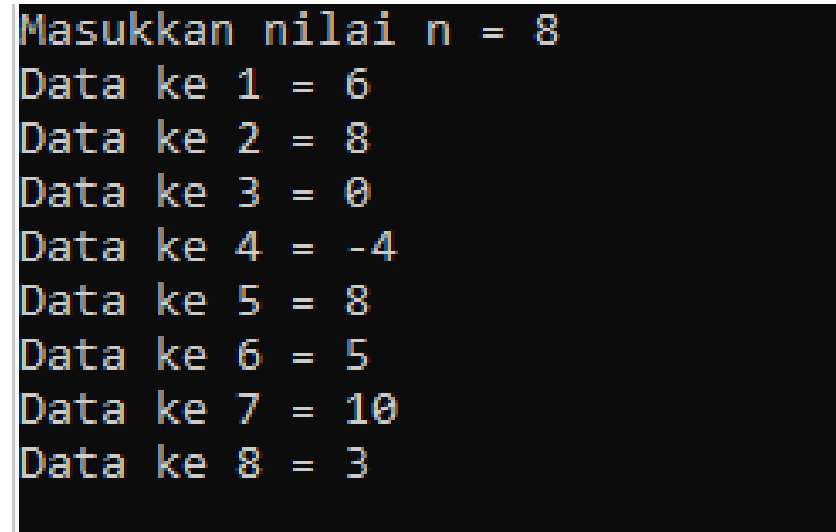


```
Masukkan nilai n = 12
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
Hello 11
Hello 12
```

Program menerima input banyaknya data, lalu meminta user menginputkan nilai setiap data satu per satu hingga selesai

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    int n,x;
    cout<<"Masukkan nilai n = ";
    cin>>n;
    while(i<=n){
        cout<<"Data ke "<<i<<" = ";
        cin>>x;
        i++;
    };
}
```

Tampilan :

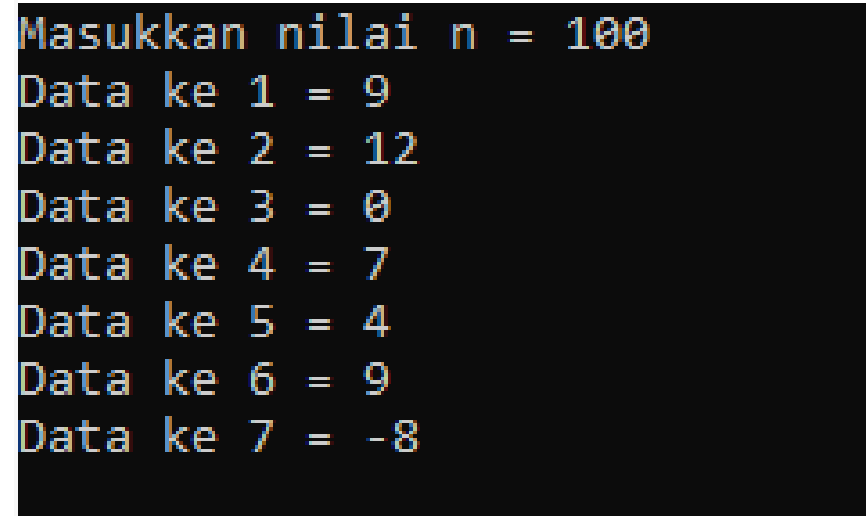


```
Masukkan nilai n = 8
Data ke 1 = 6
Data ke 2 = 8
Data ke 3 = 0
Data ke 4 = -4
Data ke 5 = 8
Data ke 6 = 5
Data ke 7 = 10
Data ke 8 = 3
```

Program menerima input banyaknya data, lalu meminta user menginputkan nilai setiap data satu per satu, namun akan berhenti setelah menerima input bilangan negatif

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    int n,x;
    cout<<"Masukkan nilai n = ";
    cin>>n;
    while(i<=n && x>=0){
        cout<<"Data ke "<<i<<" = ";
        cin>>x;
        i++;
    };
}
```

Tampilan :



```
Masukkan nilai n = 100
Data ke 1 = 9
Data ke 2 = 12
Data ke 3 = 0
Data ke 4 = 7
Data ke 5 = 4
Data ke 6 = 9
Data ke 7 = -8
```

Inputan n bernilai 100, artinya akan menerima 100 data.

Namun karena data ke-7 bernilai negatif, sesuai kriteria program, maka program berhenti mengulangi aksi.

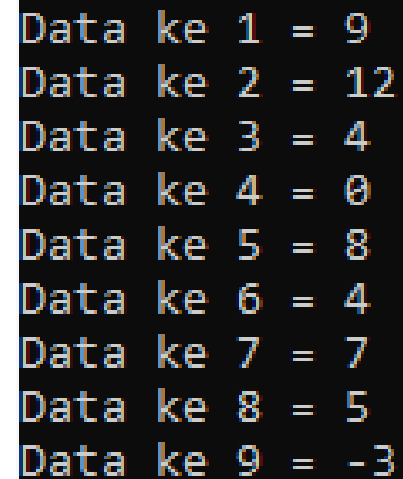
Perhatikan pada kondisi (i<=n && x>=0).

Artinya kondisi untuk tetap melakukan perulangan adalah saat nilai i masih kurang dari atau sama dengan n dan nilai x sebagai data yang diinputkan harus lebih dari sama dengan nol.

Modifikasi (tidak ada batasan untuk banyaknya data)

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    int x;
    while(x>=0){
        cout<<"Data ke "<<i<<" = ";
        cin>>x;
        i++;
    };
}
```

Tampilan :



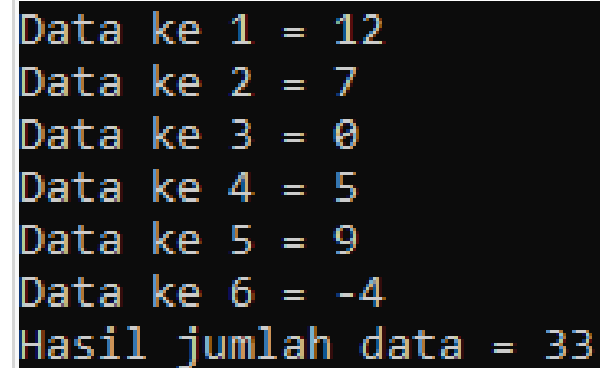
```
Data ke 1 = 9
Data ke 2 = 12
Data ke 3 = 4
Data ke 4 = 0
Data ke 5 = 8
Data ke 6 = 4
Data ke 7 = 7
Data ke 8 = 5
Data ke 9 = -3
```

Pada contoh ini tidak ada nilai n yang diinputkan. Jadi user dapat menginputkan berapa pun banyaknya data selama nilainya tidak negatif. Inilah yang menjadi kelebihan perulangan While atau Do While, dibandingkan perulangan For.

Modifikasi (setelah menerima semua data, program menampilkan hasil jumlah semua data tanpa data negatif yang terakhir)

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    int x,sum=0;
    while(x>=0){
        cout<<"Data ke "<<i<<" = ";
        cin>>x;
        sum = sum + x;
        i++;
    };
    cout<<endl;
    cout<<"Hasil jumlah data = "<<sum-x;
}
```

Tampilan :



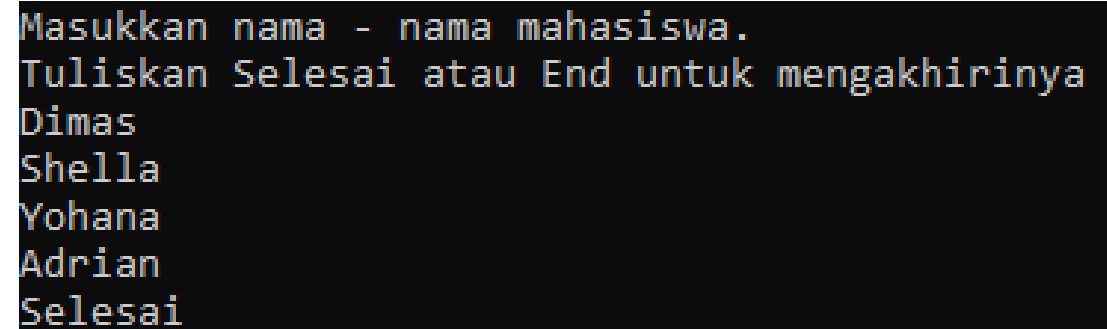
```
Data ke 1 = 12
Data ke 2 = 7
Data ke 3 = 0
Data ke 4 = 5
Data ke 5 = 9
Data ke 6 = -4
Hasil jumlah data = 33
```

Pada bagian output ditampilkan $\text{sum}-x$, Karena nilai sum masih mengandung nilai x yang terakhir (yang negatif). Sehingga kita perlu membuang nilai tersebut dengan cara mengurangi sum dengan x . Cara lain : gunakan kombinasi dengan konsep percabangan. Secara detail akan dipelajari di minggu 11.

Program menerima input data teks satu per satu, hingga menemukan bahwa ada inputan kata - kata tertentu sehingga perulangan berhenti.

Tampilan :

```
#include<iostream>
using namespace std;
int main(){
    string x;
    cout<<"Masukkan nama - nama mahasiswa."<<endl;
    cout<<"Tuliskan Selesai atau End untuk mengakhirinya"<<endl;
    while(x!="Selesai" && x!="End"){
        cin>>x;
    };
}
```



```
Masukkan nama - nama mahasiswa.
Dimas
Shella
Yohana
Adrian
Selesai
```

Artinya selama inputan bukan kata Selesai dan juga bukan kata End, maka program masih bisa terus menerima inputan string.

Catatan : perhatikan case sensitive.
Pada contoh ini kata selesai berbeda dengan Selesai. Sehingga mungkin perlu ditambahkan lagi kondisinya dengan && x!="selesai" dan seterusnya

Program pengenceran larutan. Diinputkan jumlah mol zat dan volume pelarut mula - mula. Hitung konsentrasi (Molaritas). Lalu akan ditambahkan pelarut terus menerus sampai konsentrasinya (M) kurang dari 0.5. Tampilkan output konsentrasi dan volume akhir.

```
#include<iostream>
using namespace std;
int main(){
    float n,V,M,V_add;
    cout<<"Masukkan jumlah mol zat (n) = ";
    cin>>n;
    cout<<"Masukkan volume pelarut mula - mula (V) dalam liter = ";
    cin>>V;
    M=n/V;
    while(M>=0.5){
        cout<<"Tambahkan pelarut sebanyak (liter) = ";
        cin>>V_add;
        V=V+V_add;
        M=n/V;
    }
    cout<<"Konsentrasi akhir larutan (M) = "<<M<<endl;
    cout<<"Volume akhir larutan (liter) = "<<V;
}
```

Tampilan :

```
Masukkan jumlah mol zat (n) = 5
Masukkan volume pelarut mula - mula (V) dalam liter = 2.5
Tambahkan pelarut sebanyak (liter) = 0.5
Tambahkan pelarut sebanyak (liter) = 0.5
Tambahkan pelarut sebanyak (liter) = 1
Tambahkan pelarut sebanyak (liter) = 1
Tambahkan pelarut sebanyak (liter) = 1.5
Tambahkan pelarut sebanyak (liter) = 1.5
Tambahkan pelarut sebanyak (liter) = 2
Konsentrasi akhir larutan (M) = 0.47619
Volume akhir larutan (liter) = 10.5
```

Gunakan tipe variable float, sebab inputan dan output bernilai real

Latihan 4

Buatlah program yang meminta user untuk menginputkan tiga bilangan bulat a, b dan c (satu per satu). Apabila di antara a, b, dan c ada yang bernilai sama maka program akan kembali meminta user menginputkan a, b dan c seterusnya hingga ketiganya berbeda nilai.

Contoh tampilan :

```
Masukkan nilai a = 6
Masukkan nilai b = 4
Masukkan nilai c = 4
Masukkan nilai a = 1
Masukkan nilai b = 1
Masukkan nilai c = 2
Masukkan nilai a = 8
Masukkan nilai b = 8
Masukkan nilai c = 8
Masukkan nilai a = 3
Masukkan nilai b = 4
Masukkan nilai c = 5
```