



PERULANGAN

ALGORITMA & PEMROGRAMAN I



Institut Teknologi Sumatera

TUJUAN

- Mahasiswa memahami jenis-jenis pengulangan dan penggunaannya serta memahami elemen-elemen dalam pengulangan.
- Mahasiswa dapat menggunakan notasi pengulangan yang sesuai dengan benar
- Mahasiswa dapat memanfaatkan jenis-jenis pengulangan dengan tepat dalam menyelesaikan persoalan sederhana yang diberikan.

MENULIS 1 DAN 2

Tuliskan program yang menuliskan angka 1 dan 2 dan selanjutnya 1+2 ke layar

Contoh keluaran:

```
1
2
3
```

```
...
int main () {
// KAMUS

// ALGORITMA
    cout << 1 << endl;
    cout << 2 << endl;
    cout << 1+2 << endl;
    return 0;
}
```

MENULIS 1 S.D. 3

Tuliskan program yang menuliskan angka 1 s.d. 3 dan selanjutnya 1+2+3 ke layar

Contoh keluaran:

```
1
2
3
6
```

```
...
int main () {
// KAMUS

// ALGORITMA
    cout << 1 << endl;
    cout << 2 << endl;
    cout << 3 << endl;
    cout << 1+2+3 << endl;
    return 0;
}
```

MENULIS 1 S.D. 10

Tuliskan program yang menuliskan angka 1 s.d. 10 dan selanjutnya
 $1+2+3+\dots+10$ ke layar

Contoh keluaran:

```
1
2
3
4
5
6
7
8
9
10
55
```

```
...
int main () {
// KAMUS


// ALGORITMA
    cout << 1 << endl;
    cout << 2 << endl;
    cout << 3 << endl;
    cout << 4 << endl;
    cout << 5 << endl;
    cout << 6 << endl;
    ... //lanjutkan sendiri!!
    cout << 10 << endl;
    cout << 1+2+3+4+5+6+7+8+9+10 <<
endl;
    return 0;
}
```

MENULIS 1 S.D. 100

Tuliskan program yang menuliskan angka 1 s.d. 100 dan selanjutnya $1+2+3+\dots+100$ ke layar

Contoh keluaran:

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



```
...
int main () {
// KAMUS

// ALGORITMA
    cout << 1 << endl;
    cout << 2 << endl;
    cout << 3 << endl;
    cout << 4 << endl;
    cout << 5 << endl;
    cout << 6 << endl;
    ... //lanjutkan sendiri!!
    cout << 100 << endl;
    cout << 1+2+3+4+5+6+7+8+9+10+11+12+
    ... // lanjutkan sendiri!!!
    return 0;
}
```

BAGAIMANA KALAU...

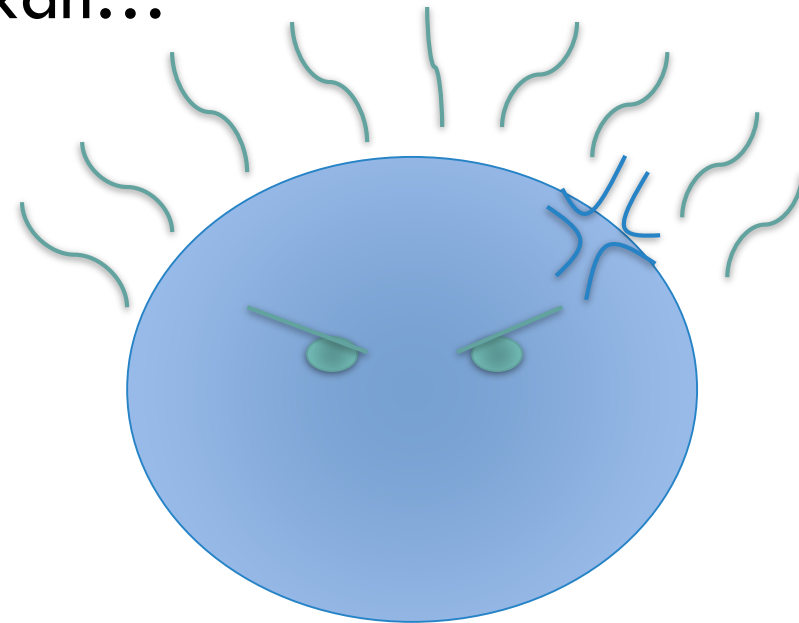
Anda diminta menulis dan menjumlahkan...

1 s.d. 1000 ???

1 s.d. 10000 ???

1 s.d. 1000000 ???

....



PENGULANGAN: LATAR BELAKANG

Melakukan suatu **instruksi**, bahkan **aksi**, secara **berulang-ulang**

- Komputer: memiliki performansi yang sama
- Manusia: punya kecenderungan untuk melakukan kesalahan (karena letih atau bosan)



PENGULANGAN / LOOPING

Elemen:

- **Kondisi pengulangan:** ekspresi logik
- **Badan pengulangan:** aksi yang diulang

Jenis-jenis notasi pengulangan:

1. Berdasarkan kondisi pengulangan di akhir : **while**
2. Berdasarkan kondisi pengulangan di awal : **do-while**
3. Berdasarkan pencacah : **for**

STUDI KASUS UNTUK CONTOH

Tuliskan program yang menerima masukan sebuah integer misalnya N dan menuliskan angka $1, 2, 3, \dots, N$ dan menuliskan $1+2+3+\dots+N$ ke layar.

Asumsikan $N > 0$.

Contoh:

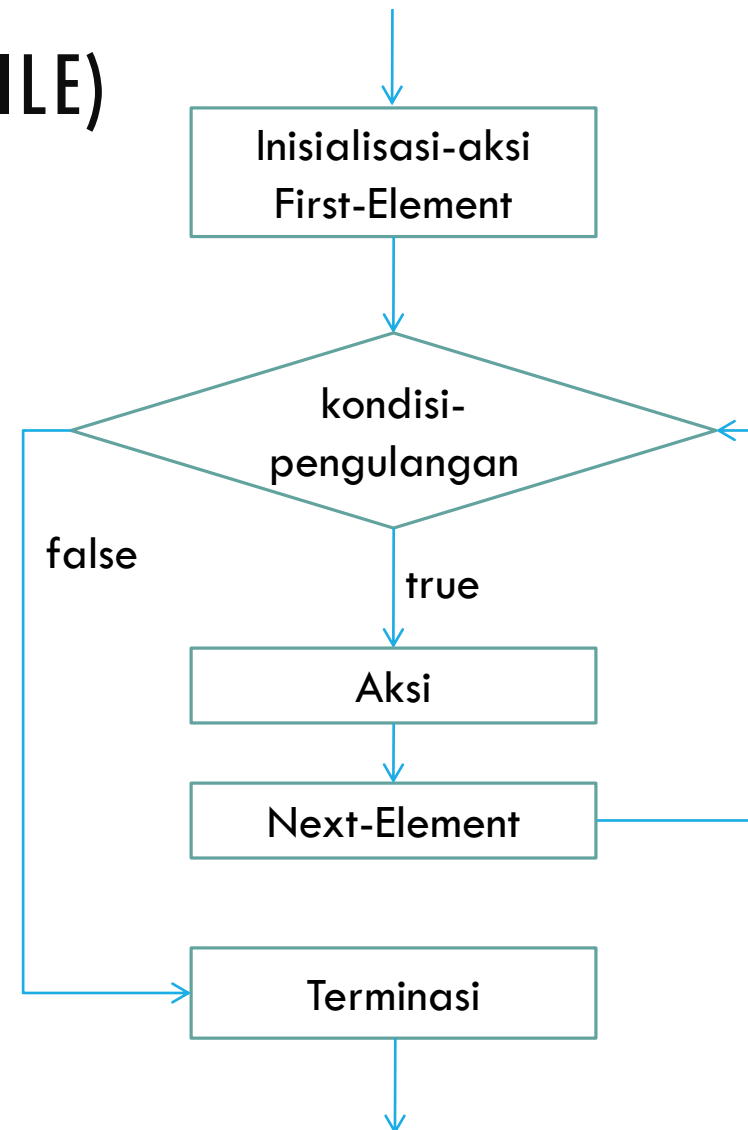
$N = 1$
Tampilan di layar:
1
1

$N = 5$
Tampilan di layar:
1
2
3
4
5
15

$N = 10$
Tampilan di layar:
1
2
3
4
5
6
7
8
9
10
55

1. PENGULANGAN BERDASARKAN KONDISI PENGULANGAN DI AWAL (WHILE)

```
Inisialisasi-aksi  
First-Element  
  
while (kondisi-pengulangan)  
{  
    Aksi  
    Next-Element  
}  
//Kondisi-pengulangan=false  
  
Terminasi
```



WHILE

Pengulangan dikendalikan oleh elemen pengulangan yang diinisialisasi sebagai **First-Element** dan diubah nilainya dalam badan pengulangan menjadi **Next-Elem**

Aksi akan dilakukan selama **kondisi-pengulangan** masih dipenuhi (berharga true)

Tes terhadap **kondisi-pengulangan** dilakukan setiap kali sebelum aksi dilaksanakan

Pengulangan ini berpotensi untuk menimbulkan **Aksi** “kosong” (tidak pernah melakukan apa-apa) karena pada test yang pertama, **kondisi-pengulangan** tidak dipenuhi (berharga false) sehingga langsung ke luar loop

CONTOH – WHILE

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

int main () {
    int i = 1;

    while(i <= 5) {
        cout << "Perulangan ke-" << i << endl;
        i++;
    }
}
```

LATIHAN

- Modifikasi contoh sebelumnya, sehingga jumlah perulangan dapat ditentukan sendiri oleh pengguna.
- **Input:** N. **Output:** perulangan sebanyak N kali.

LATIHAN

- Buatlah program untuk menampilkan suatu tabel perkalian. Misalnya perkalian antara bilangan 1 s.d 10 dengan bilangan N, maka akan tampil:

$$1 * N = X$$

$$2 * N = X$$

$$3 * N = X$$

$$4 * N = X$$

- Begitu seterusnya sampai dengan $10 * N = X$. Mintalah pengguna untuk menentukan nilai N.

LATIHAN

- Buatlah program yang meminta pengguna untuk memasukkan bilangan sebanyak N buah. Kemudian tentukan nilai minimal dan maksimal dari bilangan-bilangan tersebut.
- **Input:** N . **Output:** bilangan minimal dan maksimal.

CONTOH — WHILE

```
// Program JumlahAngka
// Menghitung 1+2+3+...+N; N > 0
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    cin >> N;
    sum = 0; //Inisialisasi
    i = 1;   //First-Element
    while (i <= N) { //Kondisi-pengulangan
        cout << i << endl; //Aksi
        sum = sum + i;      //Aksi
        i = i + 1;          //Next-Element
    } // i > N
    cout << sum << endl;    //Terminasi
    return 0;
}
```

```
// Alternatif ekspresi
while (i <= N) {
    cout << i << endl;
    sum+=i;
    i++;
} // i > N
```

CONTOH – WHILE

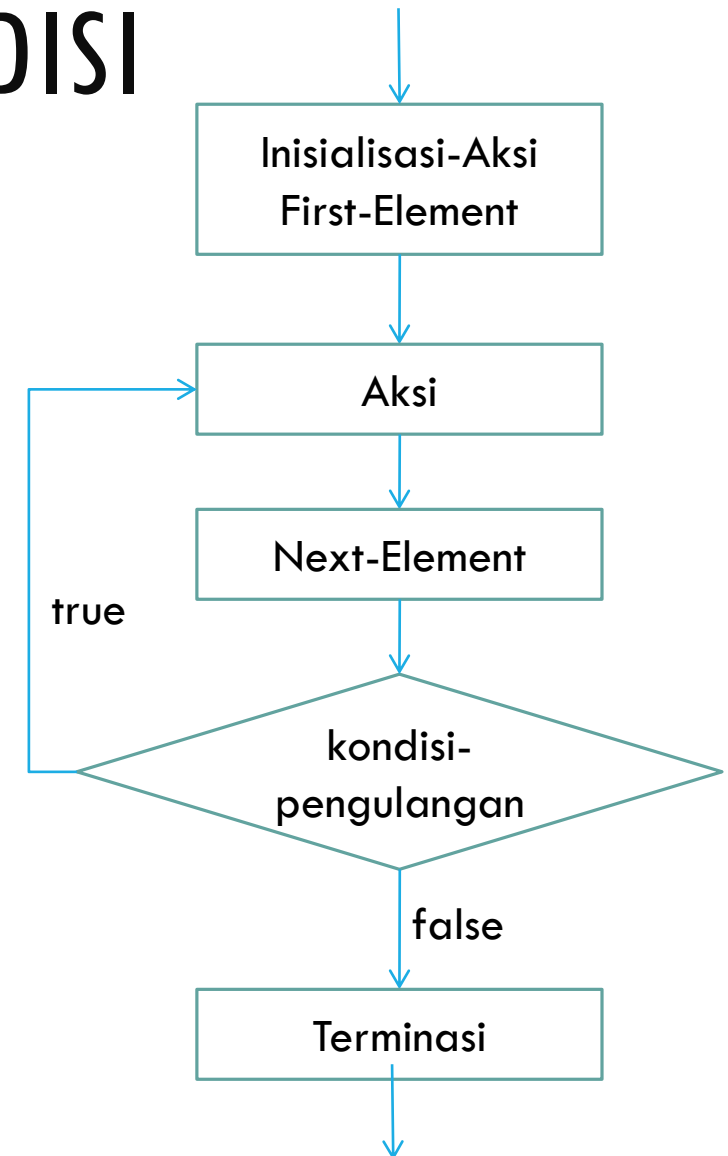
```
// Program JumlahAngka
// Menghitung 1+2+3+...+N; N > 0
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    cin >> N;
    sum = 0; //Inisialisasi
    i = 1;   //First-Element
    while (i <= N) { //Kondisi-pengulangan
        cout << i << endl; //Aksi
        sum = sum + i;      //Aksi
        i = i + 1;          //Next-Element
    } // i > N
    cout << sum << endl; //Terminasi
    return 0;
}
```

```
// Alternatif ekspresi
while (i <= N) {
    cout << i << endl;
    sum+=i;
    i++;
} // i > N
```

2. PENGULANGAN BERDASARKAN KONDISI PENGULANGAN DI AKHIR (DO-WHILE)

```
Inisialisasi-Aksi  
First-Element  
  
do  
{  
    Aksi  
    Next-Element  
} while (kondisi-pengulangan);  
  
Terminasi
```



DO-WHILE

- Pengulangan dikendalikan oleh elemen pengulangan yang diinisialisasi sebagai *First-Element* dan diubah nilainya dalam badan pengulangan menjadi *Next-Element*
- *Aksi minimal* akan dilakukan **satu kali** karena pada waktu eksekusi pengulangan yang pertama tidak dilakukan test terhadap *kondisi-pengulangan*
- *Aksi* akan dihentikan jika *kondisi-pengulangan* tidak dipenuhi (berharga false), akan diulang jika *kondisi-pengulangan* tercapai
- Test terhadap kondisi pengulangan dilakukan setelah *Aksi* dilaksanakan
- Pengulangan berpotensi mengalami “kebocoran”, jika ada kemungkinan bahwa seharusnya *Aksi* tidak pernah boleh dilakukan untuk kasus tertentu

CONTOH – DO-WHILE

```
// Program JumlahAngka
// Menghitung 1+2+3+...+N; N > 0
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    cin >> N;
    sum = 0; //Inisialisasi-aksi
    i = 1;   //First-element
    do {
        cout << i << endl; //Aksi
        sum = sum + i;      //Aksi
        i = i + 1;          //Next-Element
    } while (i <= N); //Kondisi Pengulangan
    cout << sum << endl;   //Terminasi
    return 0;
}
```

```
// Alternatif ekspresi
do {
    cout << i << endl;
    sum+=i;
    i++;
} while ( i <= N );
```

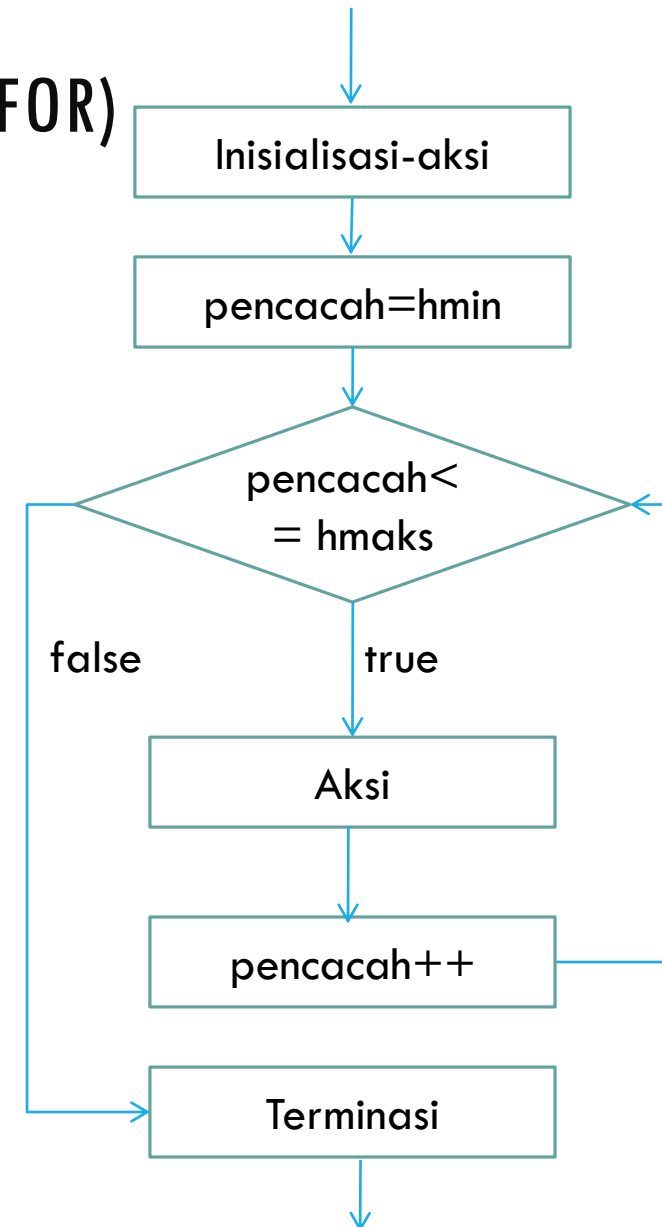
3. PENGULANGAN BERDASARKAN PENCACAH (FOR)

Inisialisasi-aksi

```
for (pencacah = hmin;  
      pencacah <= hmaks;  
      pencacah++)  
{  
    Aksi  
}
```

Terminasi

pencacah++ →
pencacah = *pencacah* + 1



FOR

- *Pencacah* harus suatu variable dengan type yang terdefinisi suksesor dan predesesornya, misalnya integer
- Nilai pencacah adalah dari *hmin* s.d. *hmaks*
- *Aksi* akan dilakukan dengan memperhitungkan harga-harga dari pencacah yang di-"jelajahi"
- Harga *pencacah* di-increment melalui operasi *pencacah++* (alias *pencacah=pencacah+1*), setiap kali *Aksi* selesai dilaksanakan
- Jika *pencacah=hmaks*, maka pengulangan berhenti

BENTUK UMUM LOOP FOR

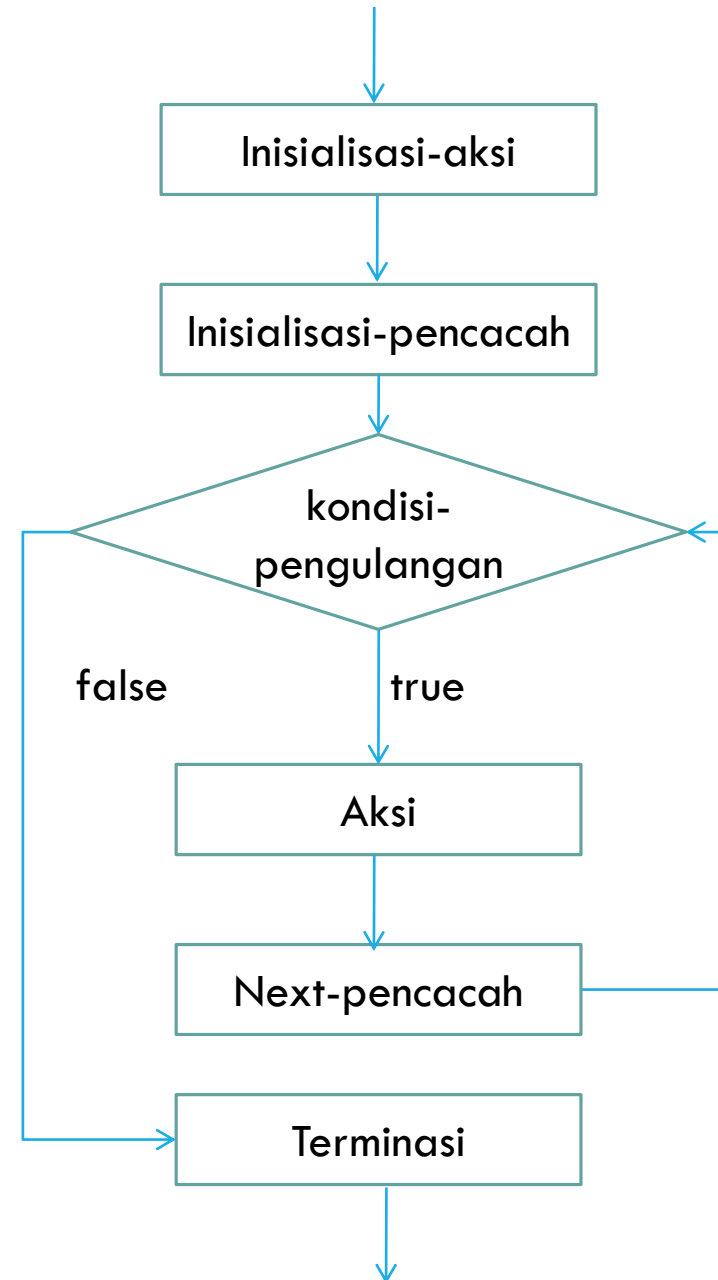
Inisialisasi-aksi

```
for (Inisialisasi-pencacah;  
      kondisi-pengulangan;  
      Next-pencacah) {
```

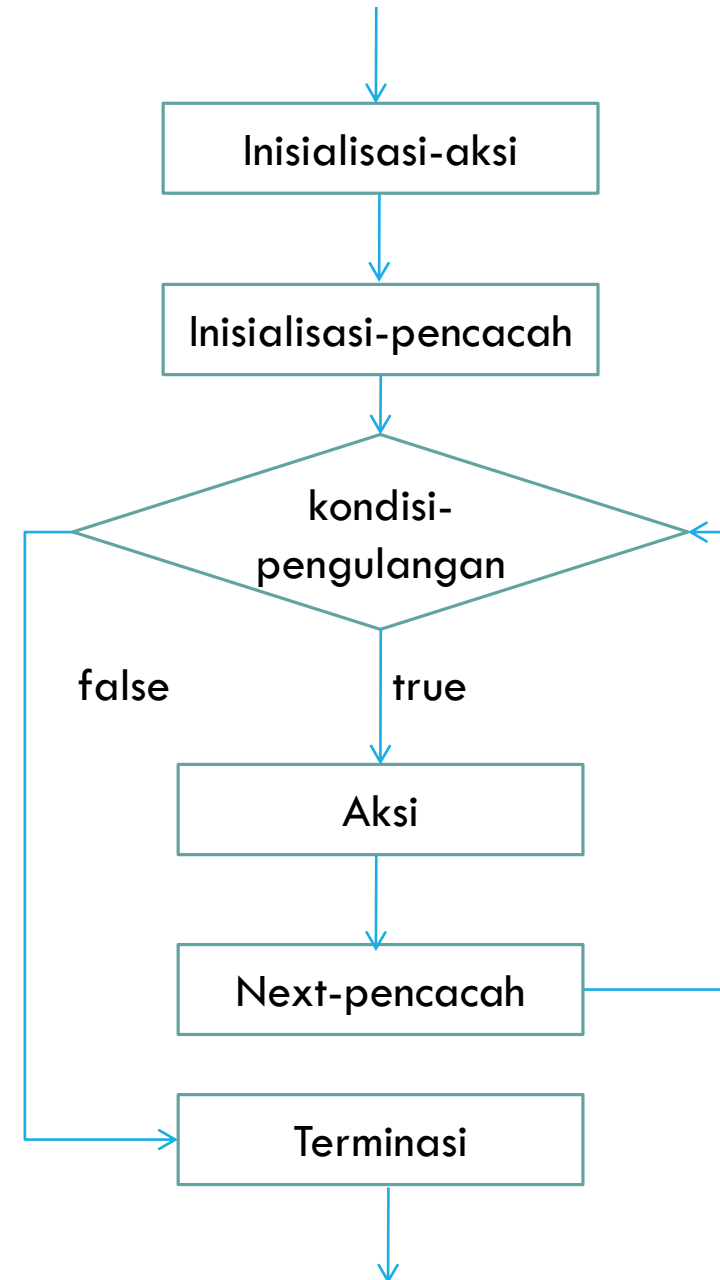
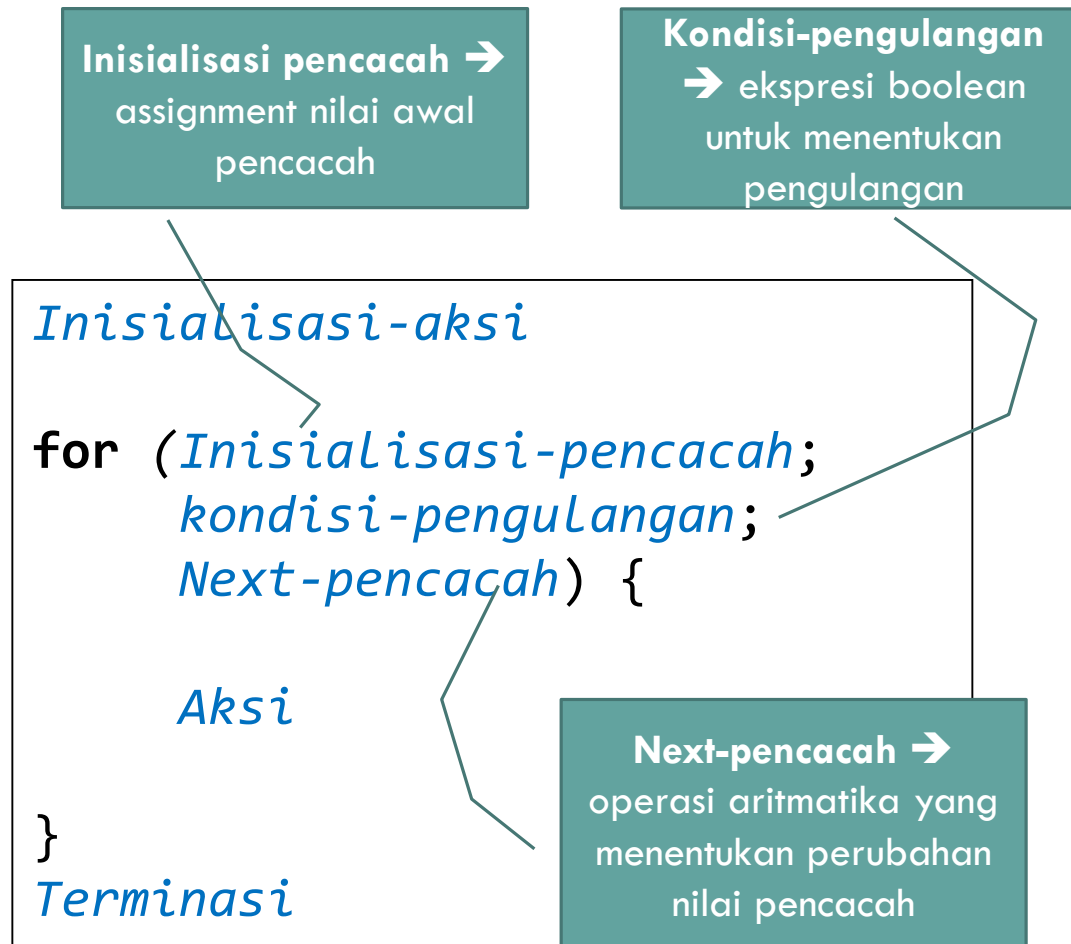
Aksi

```
}
```

Terminasi



BENTUK UMUM LOOP FOR



CONTOH — FOR

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

int main () {
    int i;

    for(i=1; i <= 5; i++){
        cout << "Perulangan ke-" << i << endl;
        i++;
    }
}
```

CONTOH — FOR

```
// Program JumlahAngka
// Menghitung 1+2+3+...+N; N > 0
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    cin >> N;
    sum = 0; //Inisialisasi-aksi
    for (i = 1; i <= N; i++) {
        cout << i << endl; //Aksi
        sum = sum + i;      //Aksi, alternatif: sum+=i;
    }
    cout << sum << endl;    //Terminasi
    return 0;
}
```

pencacah: i
Inisialisasi -pencacah: i = 1
kondisi pengulangan: i <= N
Next-pencacah : i++



LATIHAN SOAL

SOAL 1 : BERAPA HELLO DI LAYAR?

```
stop = 1; // artinya stop=true  
do {  
    cout << "Hello" << endl;  
} while (stop);
```

Tak
terhingga

```
do {  
    cout << "Hello" << endl;  
} while (0); // 0 = false
```

1 kali

```
do {  
    cout << "Hello" << endl;  
} while (1); // 1 = true
```

Tak
terhingga

SOAL 1 : BERAPA HELLO DI LAYAR?

```
i = 1;
while (i < 5) {
    cout << "Hello" << endl;
    i = i + 1;
} // i>=5
```

4 kali

```
while (0) { // 0 = false
    cout << "Hello" << endl;
} // false
```

0 – tidak ada
Hello yang
tertulis

```
while (1) { // 1 = true
    cout << "Hello" << endl;
} // false
```

Tidak
terhingga

SOAL-2

Buatlah program yang menerima masukan 10 buah bilangan integer (dari keyboard) dan menuliskan ke layar jumlah total ke-10 integer tersebut.

Contoh:

Masukan:	Tampilan di layar :
2	18
1	
0	
-9	
7	
13	
2	
2	
1	
-1	

SOLUSI SOAL-2 — ALTERNATIF 1

```
// Program Jumlah10Angka
// Menerima masukan 10 buah integer dan menjumlahkan totalnya
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    sum = 0;    //Inisialisasi

    for (i = 1; i <= 10; i++) {
        cin >> N;
        sum = sum + N; //Alternatif instruksi: sum+=N;
    }

    cout << sum << endl;    //Terminasi

    return 0;
}
```


SOLUSI SOAL-2 — ALTERNATIF 2

```
// Program Jumlah10Angka
// Menerima masukan 10 buah integer dan menjumlahkan totalnya
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, i, sum;

    // ALGORITMA
    sum = 0;           //Inisialisasi-aksi
    i = 1;             //First-Element
    do {
        cin >> N;
        sum = sum + N; //Alternatif: sum+=N;
        i = i + 1;     //Next-element, Alternatif: i++;
    } while (i <= 10);
    cout << sum << endl; //Terminasi
    return 0;
}
```

SOAL-3

- Buatlah program yang membaca sejumlah bilangan integer dari keyboard sampai pengguna memasukkan angka -999 (angka -999 tidak termasuk bilangan yang diolah).
- Tuliskan berapa banyak bilangan yang dimasukkan, nilai total, dan rata-rata semua bilangan
- Jika dari masukan pertama sudah menuliskan -999, maka tuliskan pesan “Tidak ada data yang diolah”

SOAL-3

Contoh-1:

Input :

-1
12
-6.1
10
2.5
-999

Output:

Banyak bilangan = **5**

Jumlah total = **17**

Rata-rata = **3.4**

Contoh-2

Input :

-999

Output:

Tidak ada data yang diolah

```

// Program ProsesReal
// Menerima masukan sejumlah bilangan real sampai pengguna memasukkan
// -999 dan dan menampilkan banyak bilangan, total, dan rata-ratanya
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, count, sum;
    float rata;
    // ALGORITMA
    sum = 0; count = 0;    //Inisialisasi
    cin >> N;              //First-Element
    while (N != -999) {    //Kondisi-pengulangan
        count++;           //Alternatif: count=count+1;
        sum+=N;            //Alternatif: sum=sum+N;
        cin >> N;          //Next-Element
    } //N=-999
    //Terminasi
    if (count > 0) {
        cout << "Banyak bilangan = " << count << endl;
        cout << "Jumlah total = " << sum << endl;
        rata = (float)sum/(float)count;
        cout << "Rata-rata = " << rata << endl;
    } else { // count == 0
        cout << "Tidak ada data yang diolah" << endl;
    }
    return 0;
}

```

Alternatif Solusi Soal-3

SOAL-4

- Buatlah program untuk membaca sekumpulan bilangan bulat (integer) yang diakhiri -999 (-999 tidak termasuk), dan mencetak banyaknya bilangan genap, ganjil, positif, dan negatif. Bilangan 0 adalah bilangan genap, tetapi tidak positif atau pun negatif.

SOAL-4

Contoh-1

Input : **-9**

12

0

-6

27

62

-999

Output :

Bilangan genap ada = **4**

Bilangan ganjil ada = **2**

Bilangan positif ada = **3**

Bilangan negatif ada = **2**

Contoh-2

Input : **-999**

Output :

Bilangan genap ada = **0**

Bilangan ganjil ada = **0**

Bilangan positif ada = **0**

Bilangan negatif ada = **0**

Alternatif Solusi Soal-4

```
#include <iostream>
using namespace std;
int main () {
    // KAMUS
    int N, countgenap, countganjil, countpos, countneg;
    // ALGORITMA
    countgenap=0; countganjil=0; countpos=0; countneg=0;
    cin >> N;    //First-Element
    while (N != -999) {
        if (N % 2 == 0) {
            countgenap++;
        } else { // N % 2 != 0
            countganjil++;
        }
        if (N > 0) {
            countpos++;
        } else if (N < 0) {
            countneg++;
        }
        cin >> N;    //Next-Element
    } //N=-999
    cout << "Bilangan genap ada = " << countgenap << endl;
    cout << "Bilangan ganjil ada = " << countganjil << endl;
    cout << "Bilangan positif ada = " << countpos << endl;
    cout << "Bilangan negatif ada = " << countneg << endl;
    return 0;
}
```

SOAL-5: LAGU ANAK AYAM

Masih ingatkah dengan lagu Anak Ayam??

Anak ayam turunlah 5
Mati satu tinggallah 4
Mati satu tinggallah 3
Mati satu tinggallah 2
Mati satu tinggal induknya

generalisasi

Anak ayam turunlah **N**
Mati satu tinggallah **N-1**
Mati satu tinggallah **N-2**
....
Mati satu tinggallah **2**
Mati satu tinggal induknya

Buatlah 3 versi program yang menerima masukan sebuah integer positif > 0 , misalnya N , dan menuliskan lirik lagu Anak Ayam dengan memanfaatkan pengulangan:

- do-while
- while
- for