UNIVERSITAS
ATMA JAYA YOGYAKARTA
serviens in lumine veritatis

Dasar Pemrograman (INFT06204)

Minggu 6
Perulangan

Struktur Perulangan

Struktur perulangan secara umum ada dua bagian:

- ***Kondisi perulangan***
  - Ekspresi boolean yang harus dipenuhi untuk melaksanakan perulangan
- ***Badan perulangan***
  - Bagian algoritma yang diulang

Struktur perulangan disertai pula:

- ***Inisialisasi***
  - Aksi yang dilakukan sebelum perulangan dilakukan pertama kali
- ***Terminasi***
  - Aksi yang dilakukan setelah perulangan selesai dilaksanakan

Struktur Perulangan

- Struktur perulangan secara umum:

  <inisialisasi>

  awal perulangan

  badan perulangan

  akhir perulangan

  <terminasi>

Struktur Perulangan

Beberapa hal yang harus diperhatikan:

- Inisialisasi dan terminasi *tidak selalu harus ada*, namun dalam beberapa kasus inisialisasi biasanya diperlukan.
- **Suatu perulangan harus berhenti**!
- Beberapa struktur perulangan dapat dipakai untuk masalah yang sama, namun ada notasi perulangan yang hanya cocok dipakai untuk masalah tertentu.

Struktur Perulangan

Notasi struktur perulangan:

- Struktur FOR
- Struktur WHILE
- Struktur DO-WHILE

Struktur FOR

- Struktur perulangan FOR digunakan untuk menghasilkan perulangan sejumlah kali yang dispesifikasikan.
- Jumlah perulangan diketahui atau dapat ditentukan sebelum eksekusi.
- Diperlukan *variabel counter/pencacah*.
- Jika cacah perulangan sudah mencapai jumlah yang dispesifikasikan, maka proses perulangan berhenti.

Struktur FOR

- Bentuk umum struktur FOR:

```
for(<ekspresi1>; <ekspresi2>; <ekspresi3>)
{
        aksi1;
        aksi2;
}
```

Struktur FOR

- Bentuk umum struktur FOR menaik (ascending):

```
for(var=nilai_awal; nilai_awal<=nilai_akhir; var++)
{
        aksi1;
        aksi2;
}
```

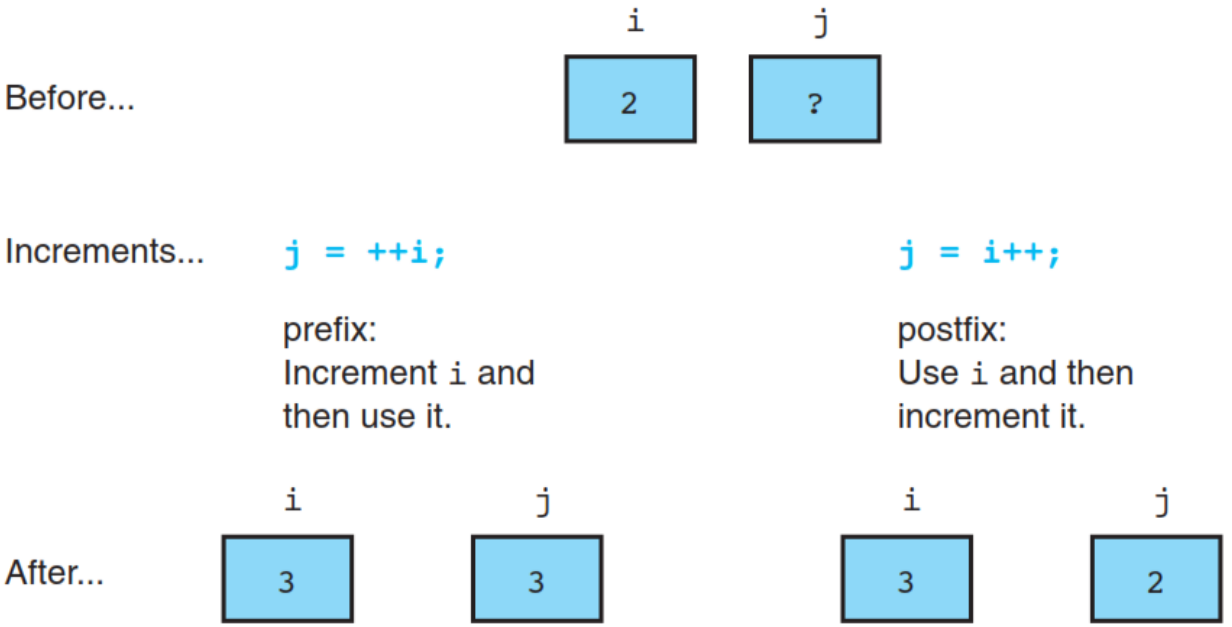
Struktur FOR

Yang harus diperhatikan adalah:

- Pencacah harus dari tipe data yang memiliki *predecessor* dan *successor*, yakni *integer* atau *karakter*.
- Aksi adalah satu atau lebih instruksi yang diulang
- **nilai_awal** harus lebih kecil atau sama dengan **nilai_akhir**
- Pada awalnya, pencacah diinisialisasi dengan **nilai_awal.**
- Nilai pencacah secara otomatis **bertambah** satu setiap kali aksi perulangan dimasuki, sampai akhirnya nilai pencacah sama dengan **nilai_akhir**
- Jumlah perulangan yang terjadi:

  nilai_akhir - nilai_awal+1

Statement with Simple Assignment Operator	Equivalent Statement with Compound Assignment Operator
`count_emp = count_emp + 1;`	`count_emp += 1;`
`time = time - 1;`	`time -= 1;`
`total_time = total_time + times;`	`total time += time;`
`product = product * item;`	`product *= item;`
`n = n * (x + 1);`	`n *= x + 1;`


i
j
Before...
2
?
Increments...
j = ++i;
j = i++;
prefix:
Increment i and
then use it.
postfix:
Use i and then
increment it.
i
j
i
j
After...
3
3
3
2

Struktur FOR

- Contoh:

```
/* mencetak kata ‘Halo’ sebanyak 10 kali */

int i;
for(i=1;i<=10;i++)
    printf(“Halo\n”);
```

Struktur FOR

- Bentuk umum struktur FOR menurun (descending):

```
if(var=nilai_awal; nilai_awal>=nilai_akhir; var--)
{
        aksi1;
        aksi2;
}
```

Struktur FOR

Yang harus diperhatikan adalah:

- Pencacah harus dari tipe data yang memiliki *predecessor* dan *successor*, yakni *integer* atau *karakter*.
- Aksi adalah satu atau lebih instruksi yang diulang
- **nilai_awal** harus lebih besar atau sama dengan **nilai_akhir**
- Pada awalnya, pencacah diinisialisasi dengan **nilai_awal.**
- Nilai pencacah secara otomatis **berkurang** satu setiap kali aksi perulangan dimasuki, sampai akhirnya nilai pencacah sama dengan **nilai_akhir**
- Jumlah perulangan yang terjadi:

  nilai_awal - nilai_akhir+1

Struktur FOR

- Contoh:

```
/* mencetak angka: 9 8 7 6 5 4 3 2 1 0 */


int i;
for(i=9;i>=0;i--)
    printf("%d ",i);
```

Contoh Lain FOR

```
int i;
i=0;                                    //inisialisasi
for(;i<10;)                             //kondisi
{
  printf("%d\n",i);                     //proses
  i++;                                  //inkremen
}
printf("---------------\n");
printf("%d\n",i);                       //terminasi
```

Struktur WHILE

- Bentuk umum struktur WHILE:

```
inisialisasi;
while(<ekspresi>)
{
        aksi1;
        aksi2;
}
terminasi;
```

Struktur WHILE

Beberapa hal yang harus diperhatikan:

- **Aksi** atau runtunan aksi akan dilaksanakan berulang kali selama **<ekspresi>** bernilai **true**. Jika kondisi bernilai **false**, badan perulangan tidak akan dilaksanakan yang artinya perulangan selesai.

Struktur WHILE

- Contoh:

```
/* mencetak kata 'Halo' sebanyak 10 kali */

int i;

i=0;
while(i<10)
{
        printf("Halo\n");
        i++;
}
```

Struktur WHILE

Hal penting yang tidak boleh dilupakan dalam program contoh:

- Melupakan inisialisasi: i=0;
- Tidak menuliskan instruksi yang mengubah kondisi: i++;

Struktur DO-WHILE

- Bentuk umum:

```
inisialisasi;
do
{
        aksi1;
        aksi2;
} while(<ekspresi>);
terminasi;
```

Struktur DO-WHILE

Beberapa hal yang perlu diperhatikan:

- **Aksi** di dalam badan akan diulang sampai kondisi boolean bernilai **false**. Jika kondisi boolean masih **true**, perulangan masih terus dilakukan.
- Karena proses perulangan suatu saat harus berhenti, maka di dalam badan perulangan harus ada **aksi** yang mengubah nilai peubah **<ekspresi>**.

Struktur DO-WHILE

- Contoh:

```
/* mencetak kata 'Halo' sebanyak 10 kali */

int i;

i=1;
do
{
    printf("Halo\n");
    i++;
} while(i<=10);
```

WHILE atau DO-WHILE ?

- Meskipun kadang bisa digunakan untuk fungsi yang sama, ketiga struktur perulangan sebaiknya digunakan sesuai dengan kasus yang dihadapi:
  - Jika banyaknya perulangan dapat dipastikan
    →gunakan struktur FOR
  - Jika perulangan dihentikan jika kondisi tertentu dipenuhi
    →gunakan WHILE atau DO-WHILE

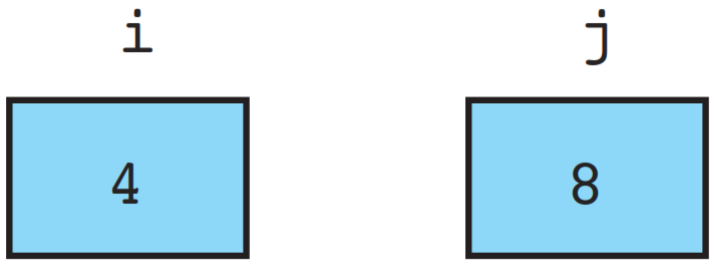
WHILE atau DO-WHILE ?

WHILE	DO-WHILE
Pemeriksaan kondisi dilakukan di **awal** perulangan	Pemeriksaan kondisi dilakukan di **akhir** perulangan
Badan perulangan **mungkin tidak pernah** dieksekusi	Badan perulangan **paling sedikit 1 kali** dieksekusi

- Gunakan struktur WHILE pada kasus yang mengharuskan terlebih dahulu pemeriksaan kondisi objek tersebut sebelum dimanipulasi.
- Gunakan struktur DO-WHILE pada kasus yang terlebih dahulu memanipulasi objek, baru kemudian memeriksa kondisi objek tersebut.

Latihan (1)

Berapakah nilai n, m dan p dibawah ini?




```
n = ++i * --j;
m = i + j--;
p = i + j;
```

Latihan (2)

Jika nilai n=8, berapakah output dari n dan ganjil pada program dibawah?

```
int jumlah=0, ganjil, n;
for (ganjil=1;
    ganjil<n;
    ganjil += 2)
jumlah = jumlah+ganjil;
printf ("Jumlah bilangan ganjil positif kurang dari %d is %d. \n",n,ganjil);
```

Latihan (3)

Apakah output dari potongan program dibawah?

```
i = 0;
while (i <= 5) {
    printf("%3d %3d\n", i, 10 - i);
    i = i + 1;
}
```

Latihan (4)

Tulis program dengan output

0	1
1	2
2	4
3	8
4	16
5	32
6	64

Latihan (5)

Buat program untuk menampilkan bilangan genap dari 2 sampai 20;

Latihan (6)

Buat program untuk menghitung nilai rata-rata dari nilai-nilai yang dimasukkan

- Banyak data tidak ditentukan di awal
- Progam akan terus menerima inputan sampai pengguna tidak ingin lagi memasukkan inputan baru

Question and Answer

Terima kasih