Universitas Padjadjaran

LOOPING STATEMENT (PENGULANGAN)

Mira Suryani, S.Pd., M.Kom

S-1 Teknik Informatika

Jatinangor, 4 Oktober 2018

Tujuan

- Mahasiswa mampu menggunakan struktur kontrol pengulangan (while, do-while, for) untuk mengeksekusi blok tertentu pada program beberapa kali.
- Mahasiswa mampu menggunakan pernyataan-pernyataan percabangan (break, continue) yang digunakan untuk mengatur arah dari aliran program.

Pokok Bahasan

Pada bagian ini akan dibahas topik-topik tentang :

- Perulangan (for, while, do while)
- Lompatan : *break , continue*
- Dalam pembahasan diberikan contoh-contoh program yang relevan dengan perintah-perintah dalam struktur control untuk pengulangan

Pendahuluan

- Ada 3 bentuk dari skema pengulangan yaitu
  - for, while dan do .. while.
- Perulangan **for** dengan **while** memiliki kemiripan yaitu **melakukan aksi minimal nol kali**, sedangkan do while melakukan aksi minimal satu kali.
- Kadang-kadang dikatakan juga bahwa for identik dengan while.
- Setiap bentuk pengulangan memiliki karakteristik tertentu
- Para programer harus bisa membedakan kapan setiap bentuk tersebut digunakan sehingga pengerjaan program (algoritma) menjadi efektif dan efisien.

Pengulangan Dengan Statement “ for “

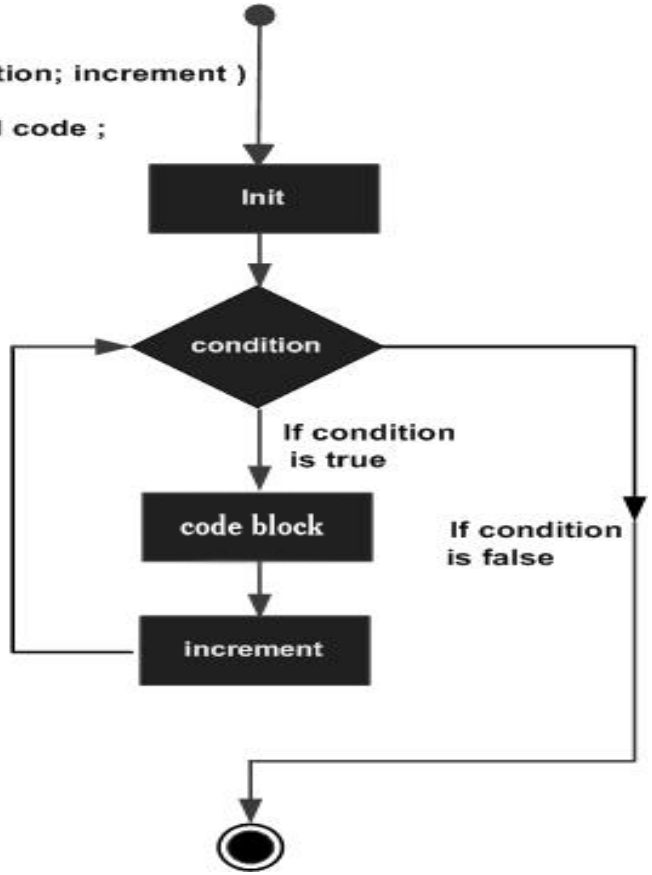
- Keyword **for** digunakan untuk melakukan perulangan dengan nilai seleksi kondisi_loop adalah sebuah angka numeric yang bisa berupa bilangan bulat maupun pecahan.
- Biasanya banyak pengulangan sudah diketahui terlebih dahulu.

Bentuk perintah **for** :

```
for(nilai_awal ; kondisi_loop ; perubahan) {
    AKSI // statement yang akan dijalankan berulang
 }
```

Pengulangan Dengan Statement “ for “

```
for( init; condition; increment )
{
    conditional code ;
}
```



Contoh :

- Mencetak angka dari 0 sampai 3 (increment)

```
for (int i=0; i<4; i++) {
    cout  <<  i  << endl;
}
```

- Mencetak angka dari 3 sampai 1 (decrement)

```
for (int i=3; i>0; i--){
    cout << i <<endl;
}
```

Contoh : menghitung Sigma(i) = 1 + 2 + 3 + ... + n (i = 1.. n)

```
main() {
    int  n, sigma = 0;
    cout << "Masukkan bilangan integer positif"; cin >> n;
    for (int i = 1; i <= n; i++) {
            sigma += i;
    }
    cout << "Jumlah dari << n <<" bil. pertama adalah : " << sigma;
}
```

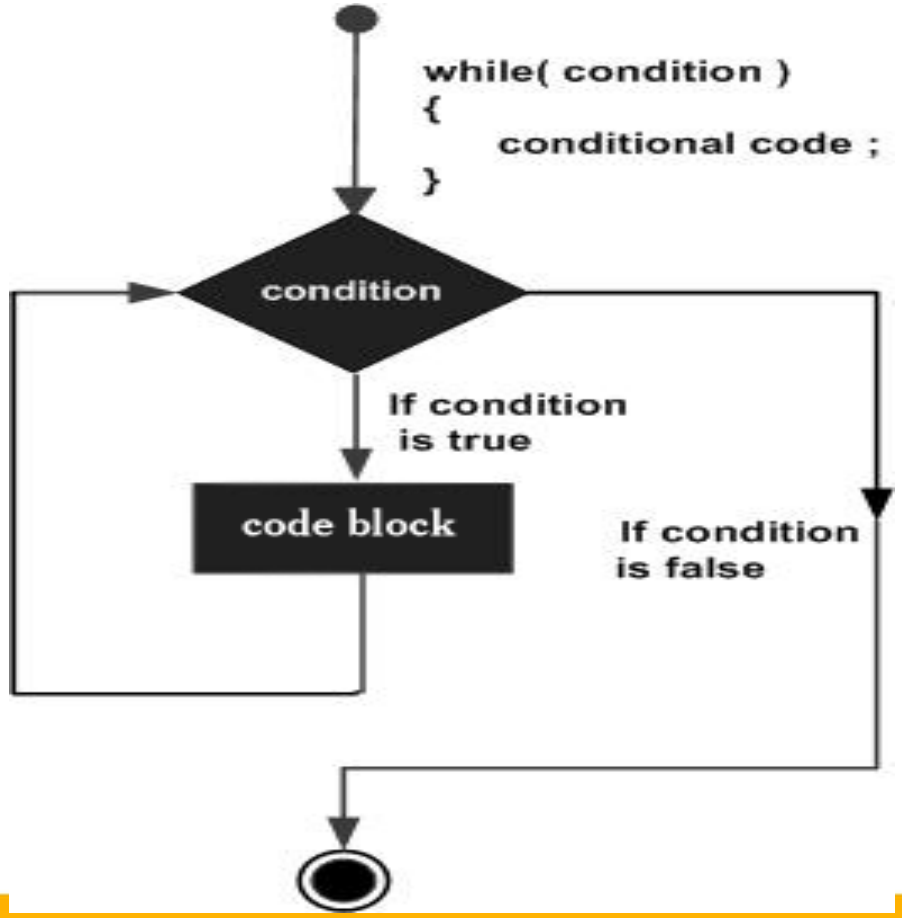
Pengulangan Dengan Statement ” while ”

- While merupakan proses looping dengan seleksi tertentu.
- Selama seleksi masih mempunyai nilai true maka loop akan terus berjalan.

- Bentuk perintah while :

```
while ( kondisi_seleksi ) {
      AKSI  // statement yang akan dijalankan berulang
      PERUBAHAN  // perubahan untuk stop looping
 }
```

Pengulangan Dengan Statement ” while ”



❑ Contoh: mencetak angka 0 .. 2

```
int i = 0;
while(i < 3){
      cout << i << endl;
      i++;
}
```

- Menghitung Sigma(i) = 1 + 2 + 3 + ... + n (i = 1.. n) dengan while

```
main() {
  int i=1, n, sigma = 0;
  cout << "Masukkan bilangan integer positif"; cin >> n;
  while (i<=n) {
      sigma += i;
      i++;
  }
  cout << "Jumlah dari << n <<" bil. pertama adalah : "<< sigma;
}
```

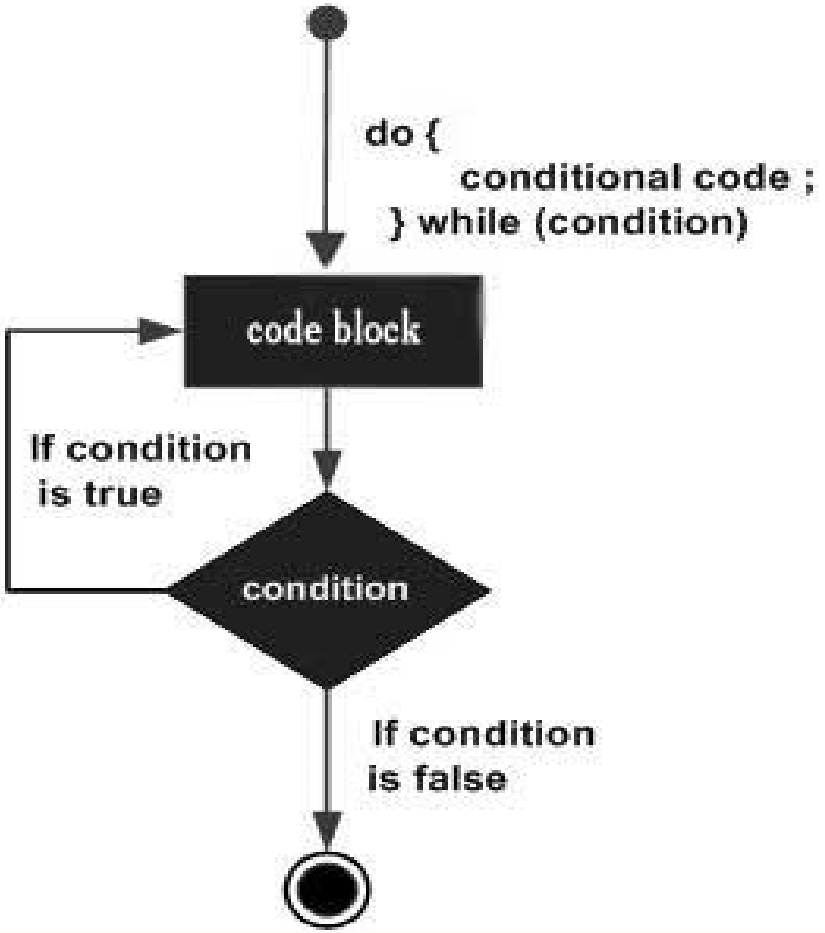
Pengulangan Dengan Statement ” do while ”

- Do while digunakan untuk melakukan looping minimal satu kali pengerjaan.
- Bentuk perintah do while adalah :

```
do {
    AKSI // statement yang akan dijalankan berulang
    PERUBAHAN   // perubahan untuk stop looping
} while(kondisi_seleksi );
```

- Pertama adalah statement di dalam blok do-while dikerjakan satu kali dahulu.
- Baru setelah itu seleksi terjadi di keyword **while**.
- Jika menghasilkan **true** maka statement akan dikerjakan ulang, sebaliknya jika false maka akan keluar dari loop.

Pengulangan Dengan Statement ” do while ”



Contoh: mencetak angka 0 .. 2

```
int i = 0;
do {
   cout<<i<<endl;
   i++;
} while(i < 3);
```

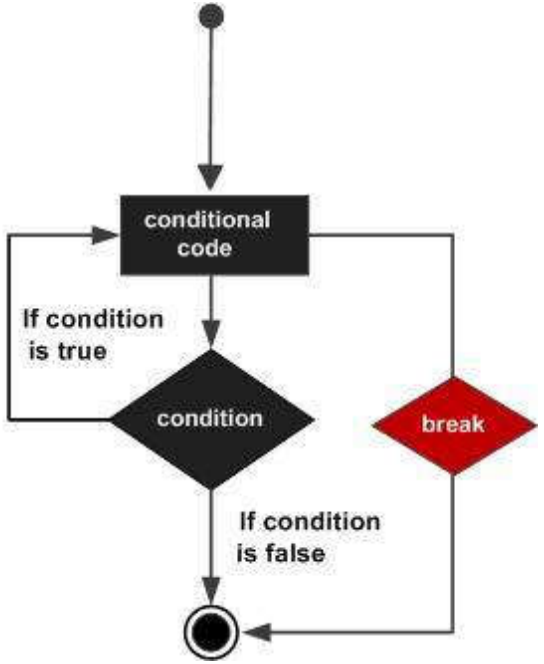
- menghitung faktorial n! = (n) (n-1) ... (3) (2) (1)

```
main() {
   int n, f = 1;
   cout << "Masukkan bil integer positif : "; cin >> n;
   do {
      f *= n;
      n--;
   } while (n >= 1);
   cout << n << " faktorial adalah : " << f << endl;
}
```

Pernyataan break

- Pernyataan break digunakan untuk keluar dari suatu pengulangan (loop) sehingga perintah-perintah lain sesudah perintah break dalam suatu loop tidak akan dikerjakan.
- Contoh : mencetak angka 1 .. 3

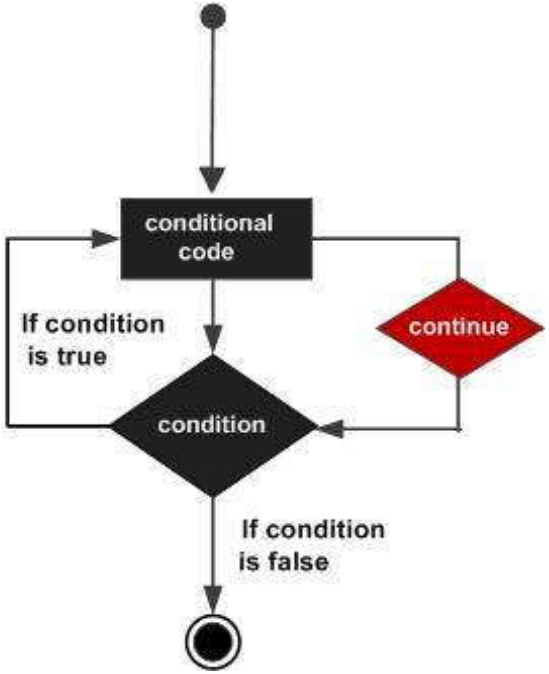
```
int bil;

for (bil = 1; bil <= 10 ; bil++) {
    if (bil == 4){
        break;
    }
    cout << bil << endl;
}
```



Pernyataan continue

- Pernyataan continue dimaksudkan untuk mengarahkan eksekusi lompat ke kondisi pernyataan for, do…while atau while sehingga kondisi akan dievaluasi lagi.
- Contoh : Mencetak 0 1 2 4

```
int i = 0;
while (i < 5) {
    if (i == 3){
        i++;
        continue;
    }
    cout << i << endl;
    i++;
 }
```



Perulangan Bersarang (Nested Loop)

- Nested for

```
for ( init; kondisi_luar; increment ){
    for ( init; kondisi_dalam; increment )   {
            statement(x);
    }
    statement(y);
}
```

❑ Nested while

```
while(kondisi_luar) {
    while(kondisi_dalam) {
        statement(x);
    }
    statement(y);
}
```

- Nested do while

```
do {
    statement(x);
    do {
        statement(y);
    } while( kondisi_dalam );
} while( kondisi_luar );
```

contoh : Mencetak bintang sebanyak 3 baris dan 5 kolom.

```
main() {
        for (int i=1; i<=3; i++) {
                for (int j=1; j<=5; j++) {
                        cout << " * ";
                }
                cout << endl;
        }
}
```

```
main() {
        int i=1;
        while(i<=3) {
                int j=1;
                while (j<=5 ) {
                        cout << " * ";
                        j++;
                }
                cout << endl;
                i++;
        }
}
```

Analisis

- Perhatikan bahwa pada pencetakan bintang tersebut menempati koordinat yang bersesuaian yaitu :

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5

- Untuk menampilkan posisi koordinat i,j bisa diganti dengan perintah:

```
cout << i << "," << j << " ";
```

Studi Kasus 1 : cetak bintang

- Dengan memanfaatkan posisi koordinat dengan ukuran n x n maka bisa dibuat berbagai bentuk pola. Misalkan dari ukuran 5 x 5 didapatkan pola sebagai berikut :

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

- Akan dibentuk susunan bintang pada posisi diagonal sbb

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

- Maka akan dicetak bintang sesuai koordinat pada kondisi baris=kolom dan baris+kolom=6 (i == j || i + j == 6)

```
main() {
  for (int i=1; i<=5; i++) {
    for (int j=1; j<=5; j++) {
      if (i==j || (i+j==6)){
        cout << " * ";
      }
      else {
        cout << "   ";
      }
    }
    cout << endl;
  }
}
```

```
main() {
  for (int i=1; i<=7; i++) {
    for (int j=1; j<=7; j++)  {
      if ((i+j<=8 && i<=j)||(i+j>=8 && i>=j)){
        cout << " * ";

      }
      else {
        cout << "   ";
      }
    }
    cout << endl;
  }
}
```

Studi Kasus 2 : cetak angka

- Akan dibuat sebuah program untuk membalikkan nilai integer yang diinputkan.
- Misalkan dimasukkan suatu angka integer 12345 maka akan dihasilkan angka integer 54321.
- Untuk menyelesaikan permasalahan ini digunakan operator modulo 10 untuk memotong angka yang paling kanan (satuan) dan pembagian dengan nilai 10 untuk membuang angka yang dipotong.
- Angka satuan yang dipotong secara berulang dijumlahkan dengan bilangan penampung (hasil) yang dikalikan dengan 10.

```
main() {
   long m, d, n=0;
   cout << "Masukkan satu bilangan integer positif :";
   cin >> m;
   while (m > 0) {
       d = m % 10;
       m /= 10;
       n = 10*n + d;
   }
   cout << "Hasil pembalikan bilangan adalah : " << n;
}
```

Studi Kasus 3 : Bilangan prima

- Untuk menentukan apakah suatu bilangan merupakan bilangan prima, maka harus dapat ditentukan bahwa bilangan tersebut habis dibagi dengan 1 dan bilangan itu sendiri.
- Bilangan prima pertama adalah 2.
- Untuk menentukan bilangan prima berikutnya dengan melakukan pembagian bilangan yang bersangkutan dengan bilangan yang dibawahnya mulai dari bilangan 2.
- Jika bilangan tersebut habis dibagi dengan bilangan sebelumnya (bersisa nol) maka bilangan tersebut pasti bukan bilangan prima.
- Sebaliknya jika tidak habis dibagi maka bilangan tersebut pasti bilangan prima

Bilangan prima

```
main() {
   int prima=1;                          // true sbg kondisi awal
   for (int i=2; i < 100; i++) {
      for (int j=2; j < i; j++) {
         if (i % j == 0) {
            prima = 0;                   //false
            break;                       //keluar loop
         }
      }
      if (prima == 1)
         cout << "Bilangan "<< i <<" adalah bilangan prima"<<endl;

      prima = 1;                        // kembalikan
   }
}
```


ANY
QUESTIONS?

Sesi Berakhir
TERIMA KASIH