PENGULANGAN
SKEMA PEMROSESAN SEKUENSIAL

Tim Pengajar KU1071

Sem. 1 2009-2010

Overview Notasi Pengulangan

1. Berdasarkan jumlah pengulangan

```
repeat n times
  aksi
```

2. Berdasarkan kondisi berhenti

```
repeat
  aksi
until <kondisi-berhenti>
```

Overview Notasi Pengulangan (2)

3. Berdasarkan kondisi pengulangan

```
while <kondisi-ulang> do
  aksi
```

4. Berdasarkan dua aksi

```
iterate
  aksi-1
stop : <kondisi-stop>
  aksi-2
```

5. Berdasarkan pencacah

```
i traversal [<awal>..<akhir>]
  aksi
```

Skema Pemrosesan Sekuensial

- Pemrosesan sekuensial adalah pemrosesan secara satu persatu, dari sekumpulan informasi sejenis yang setiap elemennya dapat diakses dengan keterurutan tertentu (ada suksesor). Jadi seakan-akan **kumpulan elemen merupakan "deret" elemen**.
- Type elemen yang akan diproses: type dasar & type bentukan.
- Kumpulan informasi disimpan sedemikian rupa, sehingga selalu dikenali:
  - Elemen pertama (First_Elmt)
  - Elemen yang siap diproses (Current_Elmt)
  - Elemen yang diakses setelah Current_Elmt (Next_Elmt)
  - Tanda akhir proses (EOP)
- Bagaimana EOP bernilai true?
  - Model **dengan MARK**: elemen terakhir adalah elemen "fiktif", sebetulnya bukan anggota elemen yang diproses
  - Model **tanpa MARK**: elemen terakhir mengandung info yang memberitahukan bahwa elemen tsb adalah elemen terakhir

Skema Pemrosesan Sekuensial **Dengan** MARK (1)

```
SKEMA PEMROSESAN DENGAN MARK
{Tanpa penanganan kasus kosong secara khusus}
```

```
Skema :
   Inisialisasi
   First_Elmt
   while not EOP do
      Proses_Current_Elmt
      Next_Elmt
   {EOP}
   Terminasi
```

Skema Pemrosesan Sekuensial **Dengan** MARK (2)

```
SKEMA PEMROSESAN DENGAN MARK
{Dengan penanganan kasus kosong}
```

```
Skema :
   First_Elmt
   if (EOP) then
      Proses_Kasus_Kosong
   else
      Inisialisasi
      repeat
         Proses_Current_Elmt
         Next_Elmt
      until (EOP)
      Terminasi
```

Skema Pemrosesan Sekuensial **Tanpa** MARK (1)

```
SKEMA PEMROSESAN TANPA MARK
{ Karena tanpa mark, tak ada kasus kosong.
  Akses elemen pertama tidak berbeda dengan akses
  Next_Elmt }
```

```
Skema :
   Inisialisasi
   repeat
      Next_Elmt
      Proses_Current_Elmt
   until (EOP)
   Terminasi
```

Skema Pemrosesan Sekuensial **Tanpa** MARK (2)

```
SKEMA PEMROSESAN TANPA MARK
{ Karena tanpa mark, tak ada kasus kosong.
  Pemrosesan khusus terhadap element pertama.
  Pemrosesan elemen kedua dst mungkin kosong }
```

Skema :

```
   Inisialisasi
   First_Elmt
   Proses_First_Elmt
   while (not EOP) do
      Next_Elmt
      Proses_Current_Elmt
   { EOP }
   Terminasi
```

Studi Kasus Skema Pengulangan

- Jumlah 1 s.d. N
  - Versi 2 (hal 97)
  - Versi 3 (hal 97)
  - Versi 4 (hal 98)
- Jumlahkan dan cacah bilangan
  - Versi 1 (hal 103)
  - Versi 2 (hal 104)
  - Versi 3 (hal 104)
  - Versi 4 (hal 105)

Latihan 1

- Buatlah algoritma yang membaca sebuah bilangan bulat positif N, mengecek apakah N adalah bilangan genap dan >=0, jika ya, menuliskan 0,2,4, … N, menjumlahkan 0+2+4+…+N serta menuliskan hasil penjumlahan. Jika tidak maka berikan pesan kesalahan
- Buatlah algoritma untuk menghitung faktorial dengan menggunakan notasi pengulangan berdasar kondisi berhenti (repeat-until). Gunakan pengecekan masukan

Latihan 2

- Buatlah program untuk membaca nilai UTS dan nilai UAS mahasiswa untuk setiap pelajaran yang diikutinya (0..100) dan diakhiri jika nilai masukan UTS di luar range nilai yang diizinkan, kemudian menghitung dan mencetak rata-rata nilai akhir dari seluruh pelajaran.

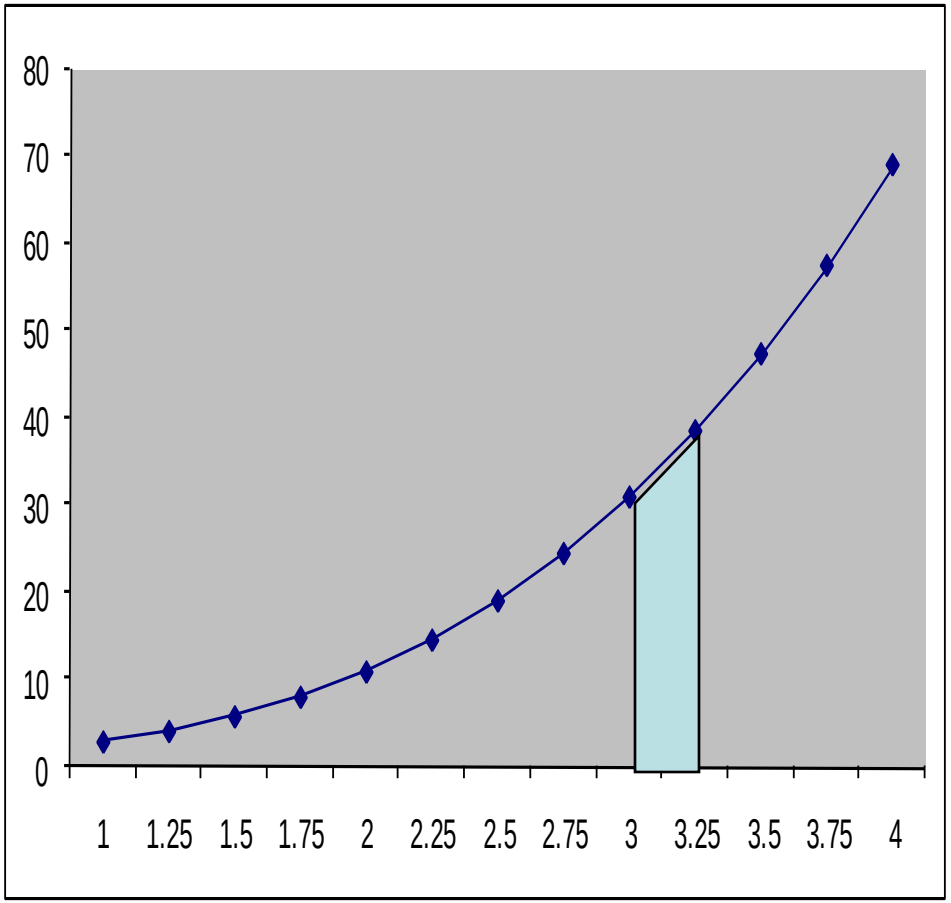
- Gunakan validasi data untuk memastikan nilai UAS pada range 0..100 (jika data tidak memenuhi syarat, input data UAS diulang). Nilai akhir untuk suatu pelajaran dihitung dari rumus (40% * nilai UTS) + (60% * nilai UAS). →

Latihan (lanjutan)

- Contoh:
  - Input:
    - Nilai UTS = 50
    - Nilai UAS = 200
    - Ulangi input nilai (0..100)!
    - Nilai UAS = 100
    - Nilai UTS = 100
    - Nilai UAS = 50
    - Nilai UTS = 9999
  - Output:
    - Nilai rata-rata = 75
  - Input:
    - Nilai UTS = 101
  - Output:
    - **Data kosong, tidak ada nilai rata-rata !**

PR

- Untuk menghitung luas daerah dari suatu kurva yang dibentuk dengan rumus dapat dilakukan dengan menggunakan integral melalui menggunakan pendekatan numerik.
- Pendekatan numerik akan memotong-motong daerah dengan interval tertentu, kemudian dihitung luas masing-masing potongan daerah tersebut dengan menggunakan rumus trapesium secara berulang-ulang.



Ditulis di selembar kertas dengan nama & NIM di bagian kanan atas.

Contoh:
Untuk menghitung luas daerah yang dibangun dari rumus
f(x) = x^3 + x + 1
dari x = 1 sampai x = 4
kita bisa memecah dengan suatu interval (misal 0.25)
makin kecil interval, makin detil hasil yang diperoleh.
Luas daerah didapat dari menghitung luas semua trapesium hasil potongan berdasar interval.

Buatlah algoritma yang menghitung luas daerah yang dibangun dari rumus f(x) = x^3 + x + 1
dari x=a sampai x=b dengan interval delta, dengan a,b,delta merupakan masukan pengguna.
Gunakan skema pengulangan

Penutup

- Dari berbagai macam pengulangan, sebenarnya satu bentuk pengulangan dapat “diterjemahkan” menjadi bentuk yang lain dengan notasi algoritmik yang tersedia. Contohnya pada persoalan penulisan nilai 1..N di atas.
- Instruksi pengulangan tidak dapat berdiri sendiri, tetapi harus disertai dengan instruksi-instruksi lain sebelum dan sesudah pengulangan.
- Persoalannya adalah **memilih bentuk pengulangan yang benar dan tepat** untuk kelas persoalan tertentu. Pada kuliah KU1071 ini, pemilihan bentuk pengulangan yang tepat merupakan salah satu objektif dari pelajaran dan merupakan inti dari “desain” algoritmik.