PERTEMUAN 3
PERULANGAN

PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN

ANALISIS KASUS

Mengulang suatu instruksi atau aksi secara berulang-ulang dengan performa yang sama setiap kalinya

Komponen Pengulangan:

- **Kondisi Berhenti**
  Kondisi yang menyebabkan pengulangan suatu saat berhenti
  Dinyatakan oleh ekspresi logic, secara eksplisit maupun implisit
  Pengulangan harus berhenti
- **Badan Pengulangan**
  Aksi yang harus diulang selama kondisi yang ditentukan terpenuhi

BERDASARKAN BANYAK PENGULANGAN

- Aksi akan diulang sebanyak n kali
- Pemrogram tidak perlu mengelola pengulangan tersebut
- Pengulangan pasti akan berhenti suatu saat

Notasi:

```
repeat n times
    Aksi
    { n adalah nama informasi yang terdefinisi nilainya,
      bilangan bulat }
```

BERDASARKAN KONDISI BERHENTI

- Aksi akan dihentikan jika `kondisi-berhenti` dipenuhi
- Aksi minimal akan dilakukan satu kali
- Aksi – Tes kondisi – Aksi – Tes kondisi – [dst] – Tes kondisi bernilai *true*
- Di bahasa C, kondisi perulangan berarti kondisi yang harus dipenuhi untuk aksi dapat diulang

Notasi:

```
repeat
    Aksi
until kondisi-berhenti
```

Bahasa:

```c
do {
    doAksi;
} while (kondisi_perulangan);
```

BERDASARKAN KONDISI ULANG

- Aksi akan dilakukan selama kondisi pengulangan masih dipenuhi
- Aksi mungkin tidak akan dilakukan
- Tes kondisi – [Aksi – Tes kondisi – Aksi – Tes kondisi – [dst]] – Tes kondisi bernilai *false*

Notasi:

```
while (kondisi-pengulangan) do

    Aksi
{ Kondisi berhenti dicapai di titik program ini }
```

Bahasa:

```
while (kondisi_perulangan) {
    doAksi;
}
```

BERDASARKAN DUA AKSI

- Gabungan antara bentuk pengulangan kedua dan ketiga
- Tergantung kondisi berhenti yang dites:
  - Aksi-2 dieksekusi lalu Aksi-1 akan diulang
  - Pengulangan dihentikan karena efek dari Aksi-1 menghasilkan kondisi berhenti

Notasi:

```
iterate
   Aksi-1
stop (kondisi-berhenti)
   Aksi-2
{ Kondisi berhenti dicapai  di titik program ini }
```

```
Program TULISBIL4
{ Dibaca N > 0, menuliskan 1, 2, 3, ..., N berderet ke bawah, dengan bentuk iterate }

KAMUS
   N, i : integer  { nilai yang akan ditulis }

ALGORITMA
   input (N)
   i ← 1
   iterate
      output (i)
   stop (i = N)
      i ← i + 1
   { i = N }
```

BERDASARKAN PENCACAH

- nama-pencacah harus suatu type yang terdefinisi suksesor dan predesesornya
- Aksi dilakukan dengan memperhitungkan harga-harga dari nama-pencacah secara berurutan+

Notasi:

nama-pencacah **traversal** *[range harga]*

Aksi

Bahasa:

```
int i, n;
for (i=0; i<n; i++) {
    doAksi;
}
```

THANK YOU

Tim asprak alpro 💖