Algoritma dan Struktur Data

Algoritma Pemrograman

Bekti Wulandari, M.Pd
Kelas B TE
2014

Program

- Program: sederetan perintah-perintah yang harus dikerjakan oleh komputer untuk menyelesaikan masalah.
- 3 level bahasa pemrograman:
  1. Bahasa tingkat rendah
  2. Bahasa tingkat menengah
  3. Bahasa tingkat tinggi

Bahasa Tingkat Rendah

- Bahasa mesin
- Berisi: kode-kode mesin yg hanya dapat diinterpretasikan langsung oleh mesin komputer.
- Berupa kode numerik 0 dan 1
- Microcode: sekumpulan instruksi dalam bahasa mesin
- (+) : Eksekusi cepat
- (-) : Sulit dipelajari manusia

Bahasa Tingkat Menengah

- Bahasa Assembly
- Bahasa simbol dari bahasa mesin
- Contoh: ADD, SUB, dll
- Macro instruksi: sekumpulan kode dalam bahasa assembly
- (+) : Eksekusi cepat, masih dapat dipelajari daripada bahasa mesin, file kecil
- (-) : Tetap sulit dipelajari, program sangat panjang

Bahasa Tingkat Tinggi

- The 3rd Generation Programming Language
- Lebih dekat dengan bahasa manusia
- Memberi banyak fasilitas kemudahan dalam pembuatan program, mis.: variabel, tipe data, konstanta, struktur kontrol, loop, fungsi, prosedur, dll
- Contoh: Pascal, Basic, C++, Java
- (+) : Mudah dipelajari, mendekati permasalahan yang akan dipecahkan, kode program pendek
  (-) : Eksekusi lambat, tidak dapat dilakukan langsung oleh komputer (membutuhkan translator)

Translator

- Translator: penerjemah dari bahasa tingkat tinggi ke bahasa tingkat rendah.
- Assembler merupakan penerjemah bahasa Assembly ke bahasa mesin.

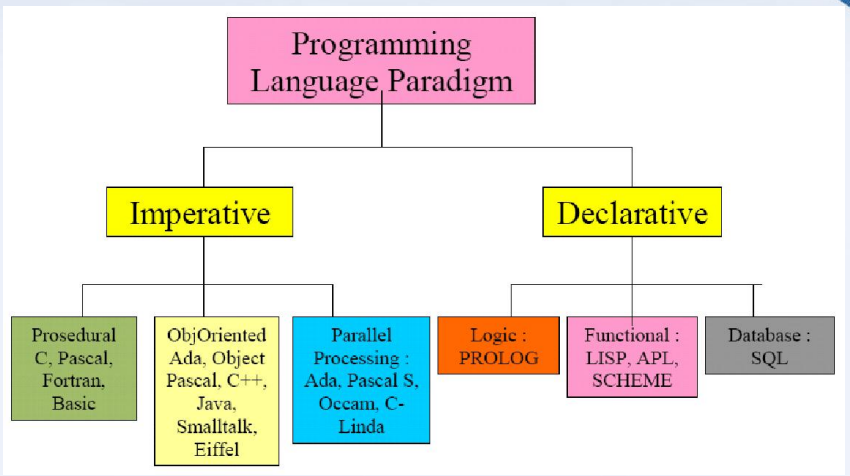
Interprenter

- Perintah diterjemahkan baris demi baris jadi program tidak dianalisis seluruhnya dulu tapi bersamaan dengan jalannya program.
- Interpreter adalah suatu program komputer yang mampu menerjemahkan program dari bahasa tingkat tinggi yang dimengerti oleh manusia dan langsung menjalankan program tersebut.
- (+) : mudah bagi user, debugging cepat
- (-) : eksekusi program lambat,
  tidak langsung menjadi program executable
- Contoh: Basic, List

Compiler

- Seluruh program diterjemahkan.
- Semua perintah (pascal, C++) dirubah dalam bentuk exe atau bahasa asembly.

Paradigma Bahasa Pemrograman



Algoritma

- Algoritma: sederetan langkah-langkah logis yang disusun secara sistematis untuk memecahkan suatu masalah.
- Disebut Logis karena setiap langkah bisa diketahui dengan pasti.
- Algoritma lebih merupakan alur pemikiran untuk menyelesaikan suatu pekerjaan atau suatu masalah.

Syarat Algoritma

- Algoritma harus tidak ambigu
- Algoritma harus tepat
- Algoritma harus pasti
- Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas.
- Algoritma memiliki nol atau lebih masukkan,
- Algoritma memiliki satu atau lebih keluaran.
- Algoritma harus efektif

Belajar Memprogram dan Belajar Bahasa Pemrograman

- Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami.
- Belajar bahasa pemrograman adalah belajar memakai suatu bahasa, aturan tata bahasanya, instruksi-instruksinya, tata cara pengoperasian compiler-nya untuk membuat program yang ditulis dalam bahasa itu saja.

Notasi Algoritma

- Penulisan algoritma tidak tergantung dari spesifikasi bahasa pemrograman dan komputer yang mengeksekusinya.
- Notasi algoritma bukan notasi bahasa pemrograman tetapi dapat diterjemahkan ke dalam berbagai bahasa pemrograman.

Notasi Algoritma

1. Uraian kalimat deskriptif (narasi)

Contoh:

Algoritma Kelulusan_mhs

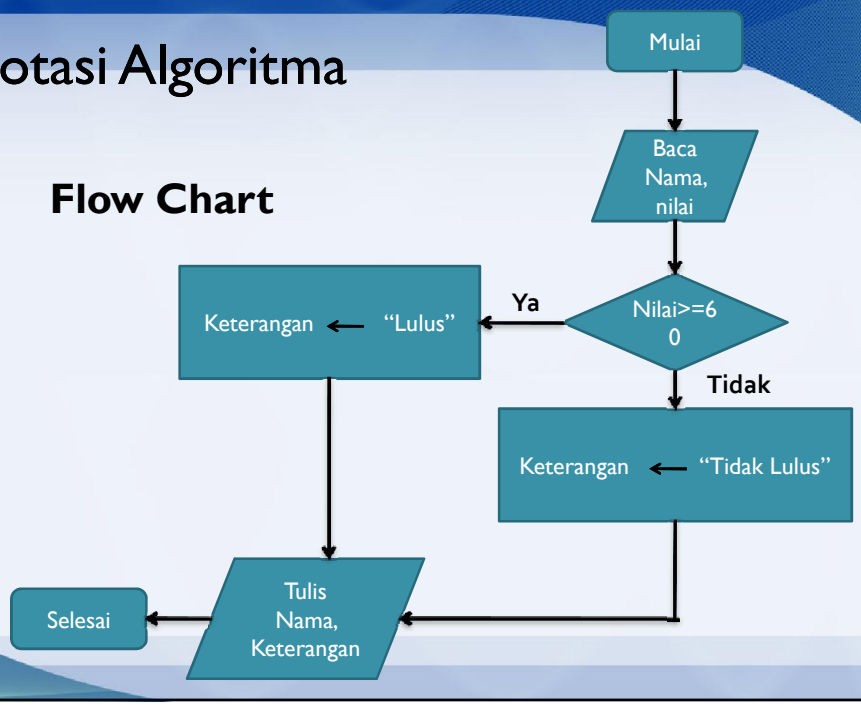
Diberikan nama dan nilai mahasiswa, jika nilai tersebut lebih besar atau sama dengan 60 maka mahasiswa tersebut dinyatakan lulus jika nilai lebih kecil dari 60 maka dinyatakan tidak lulus.

DESKRIPSI :

1. baca nama dan nilai mahasiswa.
2. jika nilai >= 60 maka
3. keterangan ← lulus
4. tetapi jika
5. keterangan ← tidak lulus.
6. tulis nama dan keterangan

Notasi Algoritma

2. **Flow Chart**



Notasi Algoritma

3. **Pseudo Code**
Ada 3 bagian: Judul, Deklarasi, Deskripsi.

```
Algoritma kelulusan
Deklarasi
    nama, keterangan : string
    nilai : integer
Deskripsi
    read (nama, nilai)
    if nilai >= 60 then
            keterangan ← 'lulus'
    else
            keterangan ← 'tidak lulus'
    write(nama, keterangan)
```

Aturan Pseudo Code

- **Judul algoritma**
  Bagian yang terdiri atas nama algoritma dan penjelasan (spesifikasi) tentang algoritma tersebut. Nama sebaiknya singkat dan menggambarkan apa yang dilakukan oleh algoritma tersebut.
- **Deklarasi**
  Bagian untuk mendefinisikan atau mendeklarasikan semua apa yang digunakan atau dibutuhkan dalam pemrograman.
- **Deskripsi**
  Bagian ini berisi uraian langkah-langkah penyelesaian masalah.

Operator Aritmetik

- /, *, div, mod → level tinggi
- +, - → level rendah
- Mod dan div hanya untuk bilangan bulat!

- Contoh :
- 5 – 2 + 1 = ?
- 5 – 2 * 3 = ?
- (6 + 3 * 2) / 6 – 2 * 3 = ?

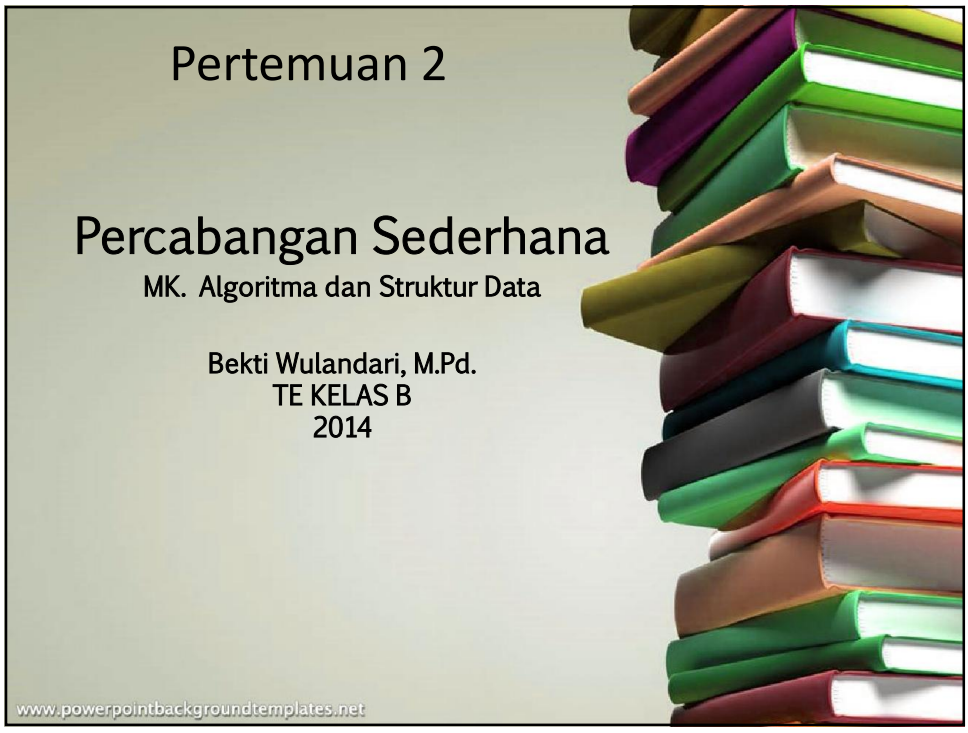
Tipe Data

- Bilangan bulat (Shortint , Integer, Longint, Byte, Word)
- Boolean (Boolean, ByteBool , WordBool, LongBool)
- Bilangan real (Real, Single, Double, Extended, Comp)
- Karakter
- String

Latihan

Buatlah notasi algoritma untuk :

a. Menghitung luas dan keliling lingkaran dengan memasukkan nilai jari-jari
b. Mengidentifikasi suatu bilangan apakah bilangan tersebut ganjil atau genap



Definisi Percabangan

- Percabangan adalah suatu suatu perintah (pernyataan) yang memungkinkan suatu perintah (pernyataan) dieksekusi jika suatu kondisi terpenuhi atau tidak terpenuhi.
- Jika suatu kondisi terpenuhi, maka perintah akan dilaksanakan.
- Jika kondisi tidak terpenuhi, maka perintah yang lainnya yang dilaksanakan.

Definisi Percabangan

(lanjutan)

- Percabangan (brancing) di dalam pemrograman digunakan oleh komputer untuk menentukan langkah kerja instruksi.
- Percabangan menggunakan operator kondisional yang akan menghasilkan nilai boolean (benar/true atau salah/false).
- Jika nilai yang dihasilkan benar, maka perintah (instruksi) akan dilaksanakan, sedangkan jika salah, maka instruksi tidak akan dilaksanakan atau melaksanakan instruksi lainnya.

Macam Percabangan

1. Satu Kasus

- if kondisi then aksi1
- Notasi algoritma :

```
if kondisi-terpenuhi (true) then
    laksanakan_aksi
endif
```

- kondisi berupa ekspresi yang menghasilkan true /false
- aksi berupa instruksi yang akan dilaksanakan jika kondisi yang dipasangkan dengan aksi yang bersangkutan bernilai benar. Bila kondisi bernilai salah, tidak ada pernyataan apapun yang dikerjakan

- Contoh

  Mencetak pesan “bilangan genap” jika bilangan tersebut genap.

Macam Percabangan

(lanjutan 1)

2. Dua Kasus

- if kondisi then aksi1 else aksi2
- Notasi Algoritma

  if kondisi-terpenuhi (true) then
  laksanakan_aksi
  else kondisi_tidak_terpenuhi (false)
  endif
- Digunakan untuk menguji sebuah kondisi dimana jika kondisi terpenuhi maka perintah yang telah ditentukan akan dijalankan, tetapi jika kondisi tidak terpenuhi maka perintah yang lain yang akan dijalankan.

- Contoh

  Mencetak pesan “bilangan genap” jika bilangan tersebut bilangan genap, atau “bilangan ganjil” jika bilangan tersebut ganjil.

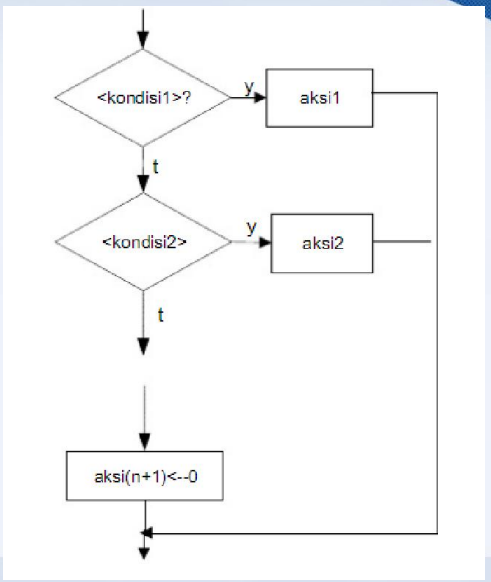
Macam Percabangan

(lanjutan 1)

3. Tiga Kasus atau Lebih

- if kondisi1 then
  aksi1
  else
  if kondisi2 then
  aksi2
  else aksi3
  endif
  endif
- Hampir sama dengan bentuk percabangan kedua tetapi kondisi yang diuji lebih dari satu.

- Flowchart



- Contoh
  Membaca temperatur air (T) pada tekanan normal (dalam satuan derajat celsius). Lalu menentukan apakah wujud air tersebut dalam keadaan padat (T≤ 0), cair (0<T<100), gas (T≥100)

```
Algoritma :
Program.....
Deklarasi
    T :.....
Algoritma
    Read (...)
    if T≤ 0 then
    Write ('.....')
        else if (.....) and (.....) then
            Write ('.....')
              if T≥100 then
                write('.....')
            end if
       end if
    endif
```

Soal

1. Karyawan honorer di PT ABC digaji berdasarkan jumlah jam kerja selama satu minggu. Upah per jam Rp 2000,-. Bila jumlah jam kerja lebih besar dari 48 jam, maka sisanya dianggap sebagai jam lembur. Upah lembur Rp 3000,-/jam. Tulislah algoritma yang membaca nama pegawai jumlah jam kerja seorang karyawan selama satu minggu, lalu menentukan upah mingguannya.
2. Menampilkan bilangan terbesar dari tiga bilangan yang dimasukkan!
3. Buatlah algoritma untuk menentukan apakah bilangan bulat yang dimasukkan tersebut bilangan positif, bilangan negatif, atau bilangan nol!

Soal

(lanjutan)

4. Buatlah algoritma yang membaca sebuah titik P(x,y) di bidang kartesian, lalu menentukan di kuadran mana letak titik tersebut!
5. Mengurutkan tiga bilangan yang dimasukkan dari kecil ke besar.
6. Mengurutkan tiga bilangan yang dimasukkan dari kecil ke besar dimana bilangan yang dimasukkan tidak boleh ada yang sama.

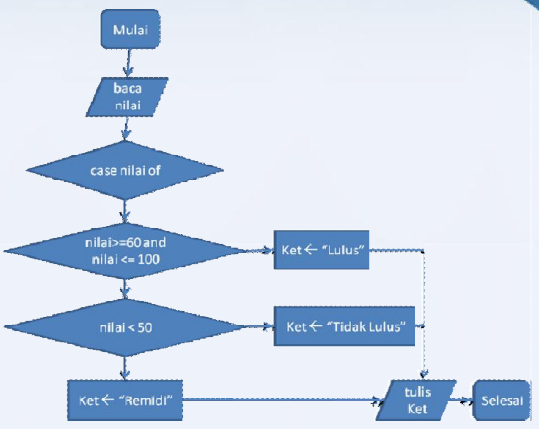
Struktur CASE

- Digunakan untuk memilih jika terdapat lebih dari dua kondisi
- Case ekspresi of
  nilai1: aksi1
  nilai2: aksi2
  nilai3: aksi3
  .....
  nilaiN:aksiN
  otherwise:aksiX
  endcase

Contoh 1

Diberikan nama dan nilai mahasiswa, jika nilai tersebut lebih besar atau sama dengan 60 maka mahasiswa tersebut dinyatakan lulus jika nilai lebih kecil dari 50 maka dinyatakan tidak lulus.

Bila nilainya 50 sampai dengan 59, maka harus mengikuti remidi.



Bagaimana Pseudo Code ?

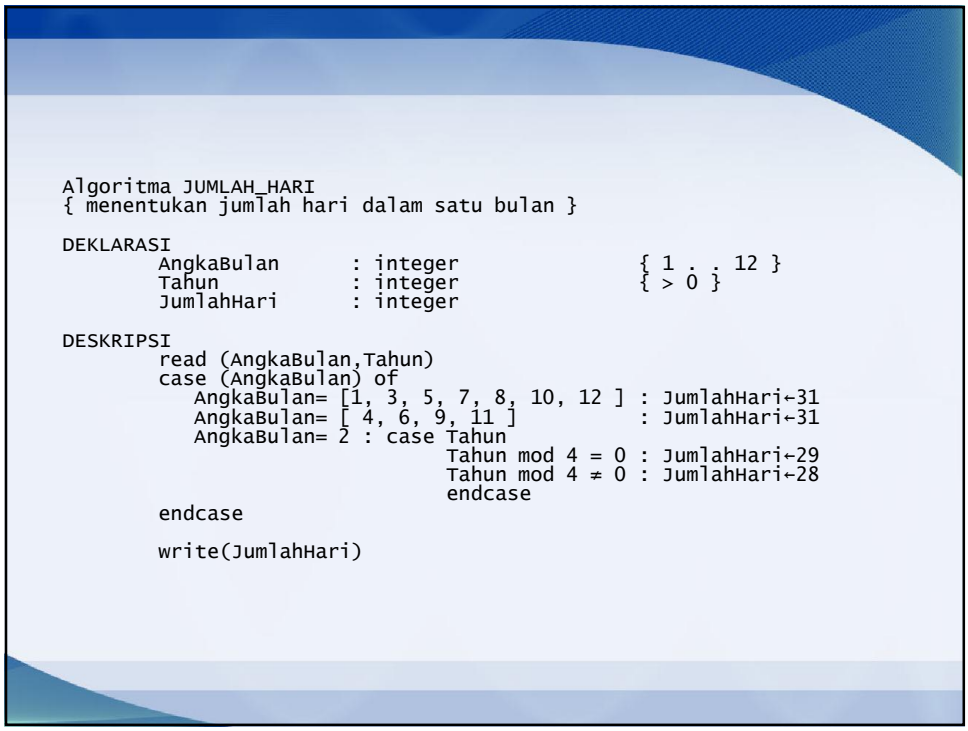
- Pseudo code :

```
algoritma kelulusan
deklarasi
        nilai : integer
        ket : string
deklarasi
        read (nilai)
        Case nilai of
        60 ..100: ('lulus')
        50 .. 59 : ('remidi')
        0 .. 49 : ('tidak lulus')
        endcase
        Write (keterangan)
```

Contoh 2:

- Buatlah algoritma dan program yang membaca angka bulan dan tahun, lalu menuliskan jumlah hari dalam bulan tersebut. Misalnya jika dibaca bulan 8 (agustus), maka jumlah harinya adalah 31.

Bulan	Jumlah hari
1, 3, 5, 7, 8, 10, 12	31
4, 6, 9, 11	30
2	29 (jika tahun kabisat), 28 (jika bukan kabisat)



```
Algoritma JUMLAH_HARI
{ menentukan jumlah hari dalam satu bulan }

DEKLARASI
        AngkaBulan      : integer               { 1 . . 12 }
        Tahun           : integer               { > 0 }
        JumlahHari      : integer

DESKRIPSI
        read (AngkaBulan,Tahun)
        case (AngkaBulan) of
           AngkaBulan= [1, 3, 5, 7, 8, 10, 12 ] : JumlahHari←31
           AngkaBulan= [ 4, 6, 9, 11 ]          : JumlahHari←31
           AngkaBulan= 2 : case Tahun
                           Tahun mod 4 = 0 : JumlahHari←29
                           Tahun mod 4 ≠ 0 : JumlahHari←28
                           endcase
        endcase

        write(JumlahHari)
```

pendahuluan

- Digunakan untuk menjalankan satu atau beberapa pernyataan sebanyak beberapa kali.
- Terdiri dari dua bagian :

1. kondisi pengulangan

   ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan

2. badan pengulangan

   aksi/pernyataan yang harus diulang selama kondisi yang ditentukan untuk pengulangan masih dipenuhi.

1. inisialisasi, yaitu aksi yang dilakukan sebelum pengulangan dilakukan pertama kali
2. terminasi, yaitu aksi yang dilakukan seteleh pengulangan selesai dilaksanakan

Struktur pengulangan
<inisialisasi>
Awal pengulangan
Badan pengulangan
Akhir pengulangan
<terminasi>

Struktur kontrol pengulangan

1. Pernyataan FOR
2. Pernyataan WHILE
3. Pernyataan REPEAT

Pernyataan FOR

- Digunakan untuk menghasilkan pengulangan sejumlah (n) kali yang dispesifikasikan.
- Jumlah pengulangan diketahui (dapat ditentukan) sebelum eksekusi.
- Variabel pencacah
  - Nilainya selalu bertambah setiap kali perulangan dilakukan.
  - Jika nilainya sudah mencapai jumlah yang dispesifikasikan, maka proses perulangan akan berhenti
- Bentuk umum for :
  - Menaik (*ascending*)
  - Menurun (*descending*)

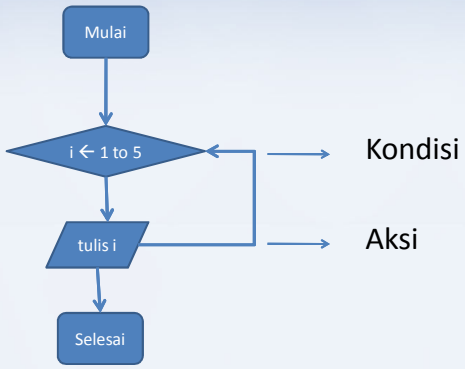
For to do

- Notasi algoritmatik

```
FOR (nama_pencacah ← nilai awal) TO (nilai_akhir) DO
        (aksi/pernyataan)
End For
```

- Ket :

1. Pencacah haruslah dari tipe data integer atau karakter, diinisialisasi dengan nilai_awal, nilai pencacah secara otomatis bertambah satu setiap kali badan pengulangan dimasuki
2. Aksi dapat berupa satu/lebih instruksi yang diulang
3. Nilai_awal harus lebih kecil dari nilai_akhir
4. Jumlah pengulangan yang terjadi = nilai_akhir – nilai_awal + 1



For downto do

- Notasi algoritma

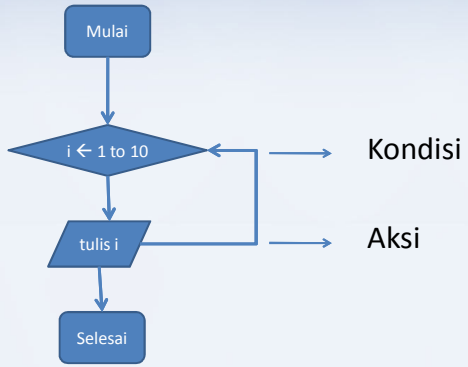
```
FOR (nama_pencacah ← nilai_akhir) DOWNTO (nilai_awal) DO
        (aksi/pernyataan)
End For
```

Contoh

1. Misalkan kita ingin mencetak angka 1 sampai 10, maka algoritmanya :

```
Program Tulis_Nomor
Deklarasi :
    i : integer
Algoritma
    For i ← 1 to 10 do      {kondisi}
      Write (i)             {aksi}
EndFor
```

2. Jika mencetak angka 1 sampai N bagaimana algoritmanya? Apa yang terjadi jika N = 0? N=-1? N = 1?



Pernyataan WHILE

- Perulangan ini dipilih jika kita tidak tahu berapa kali suatu pernyataan akan diulang-ulang.
- Banyak perulangan dilakukan melalui pemeriksaan suatu kondisi tertentu.

  Dengan demikian pemeriksaan kondisi terlebih dahulu dilakukan sebelum perulangan dijalankan.

while do

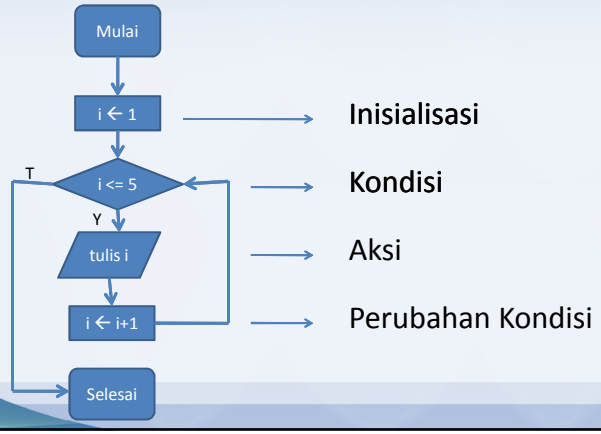
- Syntaks

```
While kondisi_pengulangan do
        Aksi
EndWhile
```

Keterangan :

Aksi akan dilakukan selama kondisi bernilai *true.*
Jika kondisi bernilai *false*, badan pengulangan tidak akan dilaksanakan, yang berarti perulangan selesai.
Yang harus diperhatikan → pengulangan harus berhenti.
Supaya kondisi bernilai *false*,
Di dalam badan pengulangan harus ada instruksi yang mengubah nilai peubah kondisi.

while do (lanjutan)



Inisialisasi

Kondisi

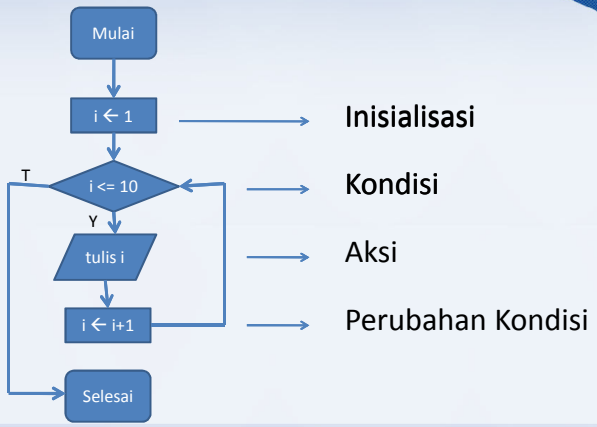
Aksi

Perubahan Kondisi

Contoh

1. Mencetak angka 1 sampai dengan 10

```
Program Tulis_Angka
Deklarasi :
    i : integer
Algoritma
   i ← 1          {inisialisasi}
   while i ≤ 10 do      {kondisi}
    Write (i)           {aksi}
  i ← 1 + 1         {perubahan kondisi}
Endwhile
{i > 10}
```



Inisialisasi

Kondisi

Aksi

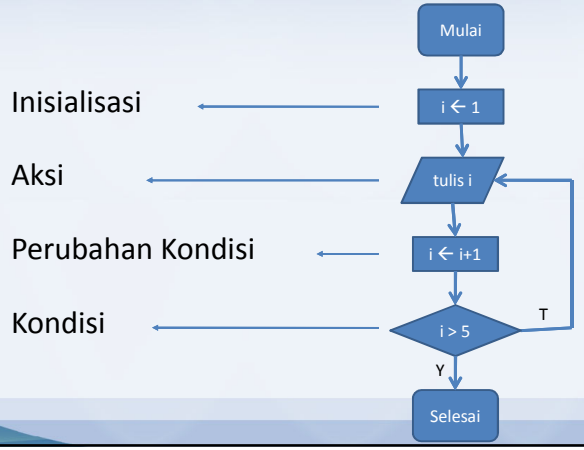
Perubahan Kondisi

Pernyataan repeat

- Bentuk perulangan ini akan melakukan aksi terlebih dahulu (minimal dilakukan satu kali), kemudian baru melakukan pemeriksaan terhadap kondisi.
- Jika kondisi benar maka perulangan masih akan tetap dilakukan.
- Perulangan akan dilakukan sampai kondisi false.
- Tidak mengetahui berapa kali pengulangan akan dikerjakan.
- Di dalam pernyataan, harus ada instruksi yang mengubah nilai kondisi agar pengulangan berhenti.

```
repeat
      aksi
until kondisi
```

repeat until



Contoh

1. Mencetak angka 1 sampai dengan 10

```
Deklarasi :
        i : integer
Algoritma
  i ← 1        {inisialisasi}
  repeat
    Write (i)        {aksi}
    i ← 1 + 1        {perubahan kondisi}
    until i > 10        {kondisi}
```

- FOR digunakan untuk proses pengulangan yang jumlah pengulangannya dapat diketahui
- WHILE fungsinya sama seperti FOR, tetapi WHILE juga dapat digunakan untuk proses yang jumlah pengulangannya tidak dapat ditentukan sebelum eksekusi
- REPEAT fungsinya sama seperti WHILE, dapat menggunakan WHILE ataupun REPEAT untuk masalah-masalah tertentu. Tetapi pada beberapa masalah, pemilihan WHILE atau REPEAT tergantung kepada persoalaanya.

Perbedaan penggunaan REPEAT atau WHILE

- Pada REPEAT, kondisi pengulangan diperiksa pada akhir pengulangan. Jadi instruksi dalam badan pengulangan dilaksanakan dulu, baru pemeriksaan kondisi dilakukan. Konsekuensinya badan pengulangan dilakukan paling sedikit satu kali. → terlebih dahulu memanipulasi objek, baru memeriksa kondisi objek tersebut.
- Pada WHILE, kondisi pengulangan diperiksa di awal pengulangan. Jadi instruksi dalam badan pengulangan hanya dapat dilaksanakan jika pemeriksaan menghasilkan nilai true. (badan pengulangan tidak akan dilaksanakan jika kondisi pengulangan pertama kali bernilai false) → mengharuskan terlebih dahulu pemeriksaan kondisi objek sebelum objek tersebut dimanipulasi

Nested Looping

- Perulangan yang terdiri dari lebih dari satu perulangan.
- Disebut perulangan bersarang (Nested Looping).
- Nested Loop pada For ..... to ..... Do

  Syntax For....to.....do

  Instruksi

  For....to....do

  Instruksi...

  end

  end

  Proses perulangan yang akan diselesaikan terlebih dahulu adalah perulangan yang terletak pada bagian dalam

- Nesterd Loop Pada While .....Do

  Pada prinsip kerjanya nested loop while ...do sama dengan for..to..do dimana proses perulangan yang akan diselesaikan terlebih dahulu adalah perulangan yang terletak bagian dalam

  Syntax :

  While.....do

  Instruksi

  While .....do

  Instruksi

  End

  End

  Proses nested repeat until hampir sama dengan proses yang ada pada nested for dan nested while. Tetapi disini masing-masing perulangan pada repeat ...until satu kali proses pasti akan dilakukan sesuai dengan keterangan yang ada pada perulangan dengan repeat until

Pengertian

- Struktur data statis yang menyimpan sekumpulan elemen yang bertipe sama, dimana setiap elemen memiliki nilai indeks.
- Salah satu tipe data terstruktur, dibutuhkan untuk menyimpan serangkaian elemen bertipe sama.
- Struktur yang dapat diakses secara acak karena semua elemen array dapat diacu secara acak dengan urutan tertentu, yaitu mengetahui nomor urutnya yang disebut indeks.

Ilustasi

- Sebuah asrama mahasiswa, yang setiap kamar mempunyai nomor urut dan dihuni seorang mahasiswa.
- Seorang mahasiswa dapat dibedakan dengan nomor urut kamarnya.
- Asrama mahasiswa seperti array, elemen-elemen asrama bertipe sama, yaitu mahasiswa
- Nomor urut kamar adalah indeksnya.
- Untuk menampilkan nilai array tinggal menyebutkan indeks-nya. Misalkan untuk menampilkan nilai variabel x yang ke 5 dituliskan dengan x(5).

Mendeklarasikan Array

- Mendefinisikan banyaknya elemen larik (ukuran)
- Mendefinisikan tipe elemen larik

Contoh:

1. Sebagai Peubah

   A adalah larik yang berukuran 50 buah elemen bertipe integer, Indeks larik dimulai 1

   Nama adalah larik yang berukuran 10 buah elemen yang bertipe string. Indeks larik dimulai dari 1

   Nilai adalah peubah larik yang berukuran 75 buah elemen yang bertipe real. Indeks larik dimulai dari 0.

   Bagaimana mendefinisikannya???

```
DEKLARASI
a : array [1..100] of integer
Nama : array [1..12] of string
Nilai : array [1..74] of real
```

Translasi notasi:

Pascal

```
Var
  A    : array [1..100] of integer;
  Nama : array [1..12] of string [25];
  Nilai   : array [1..74] of real;
```

Bahasa C

```
Int A [101];
Char Nama [13][25];
Float Nilai [75];
```

2. Sebagai Tipe Bentukan

Larik didefinisikan sebagai nama sebuah tipe baru untuk larik yang bertipe integer. Ukuran larik adalah 100 buah elemen

```
Deklarasi
type Larikint : array [1..100] of integer
A : Larik integer
```

Translasi notasi

Pascal:

```
type Larikint = array [1..100] of integer //nama tipe baru
Var A: Larik int     //A adalah sebuah larik dengan integer 100 elemen
```

Bahasa C

```
typedef int Larikint [101];
Larikint A;
```

3. Mendeklasrasikan ukuran larik sebagai sebuah konstanta misalkan Larikint dideklarasikan sebagai nama sebuah tipe bentukan untuk larik yang bertipe integer. Ukuran maksimum larik adalah 100 elemen. Ukuran maksimum larik dinyatakan sebagai konstanta.

```
Deklarasi
const Nmaks = 100
Type Larikint : array [1..Nmaks] of integer
A : Larikint
```

Transalasi Notasi

Pascal

```
const Nmaks = 100;
Type Larikint : array [1..Nmaks] of integer;
Var A : Larikint;
```

Bahasa C

```
const int Nmaks = 100;
Typedef int Larikint [Nmaks+1]
Larikint A;
```

4. Array bertipe terstruktur

```
Deklarasi
Const Nmaks = 100
Type Mahasiswa : record <NIM : integer,
                         Nama : string,
                         IPK : real
                         >
Type TabMhs : array [1..Nmaks] of mahasiswa
Mhs : TabMhs
```

Translasi Notasi

Pascal

```
Const Nmaks = 100;
Type Mahasiswa : record
              NIM : integer;
              Nama : string;
              IPK : real;
              end;
TabMhs : array [1..Nmaks] of mahasiswa
Var Mhs : TabMhs;
```

Bahasa C

```
Const int Nmaks = 100;
Typedef struct
   {int NIM;
   char Nama [25];
   float IPK;
   } mahasiswa;
Typedef Mahasiswa
    TabMhs [Nmaks+1];
TabMhs Mhs;
```

Contoh penggunaan array adalah pada penyimpanan nilai seorang mahasiswa selama 10 kali mengikuti tes. Ilustrasinya sebagai berikut :

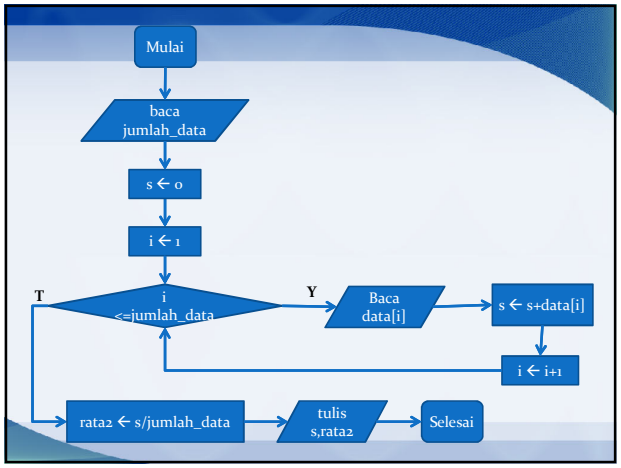
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
86	90	97	100	98	79	76	55	90	100

- Nama variable array di atas adalah A , memiliki 10 elemen. Nilai 1,2,3 ... dst adalah nilai indeks untuk menunjuk elemen tertentu. Range yang digunakan pada array berdimensi satu di atas adalah 1 sampai dengan 10.
- Nilai mahasiswa A pada tes yang ketiga ditunjuk oleh variable A pada indeks ketiga ::: A[3] = 97.

Statement Option Base

- Dalam pemakaian sebuah array kita akan memakai sistem range.
- Contoh pada array nilai di atas (array A) mengindikasikan sebuah array dengan range 1 sampai 10.
- 1 merupakan range nilai awal sedangkan 10 merupakan range nilai akhir.
- Nilai range awal sebuah array, dapat dimulai dengan angka 0 (nol) atau 1 (seperti contoh di atas).

Menentukan jumlah data yang akan dimasukkan, kemudian memasukkan datanya dan menghitung rata-ratanya.



SUB RUTIN

MK. Algoritma dan Struktur Data

Bekti Wulandari, M.Pd.
TE KELAS B
2014

Pendahuluan

- Sub rutin adalah suatu bagian dalam program yang dapat melakukan tugas tertentu. Jadi sub rutin merupakan “program kecil” yang menjadi bagian dari suatu program yang besar.

1. prosedur
2. fungsi

   Perbedaan antara keduanya adalah setelah dipanggil prosedur tidak mengembalikan suatu nilai sedangkan fungsi selalu mengembalikan suatu nilai.

   Prosedur sebagai fungsi tanpa nilai balik. Fungsi tanpa nilai balik ditulis dengan bagian tipe fungsi berupa **void** (void berarti tanpa nilai balik).

PROSEDUR

Prosedur adalah suatu program yang terpisah dalam blok sendiri yang berfungsi sebagai subprogram (program bagian). Prosedur diawali dengan kata cadangan PROCEDURE di dalam bagian deklarasi prosedur. Prosedur dipanggil dan digunakan di dalam blok program lainnya dengan menyebutkan judul prosedurnya.

Procedure adalah sekumpulan instruksi yang dibungkus yang akan dipanggil dalam program utama.

Penulisan fungsi/prosedur sebelum fungsi main

```
tipe_fungsi nama_fungsi(parameter input)
{
Badan Fungsi
}
void nama_prosedur(parameter input,parameter output)
{
Badan Prosedur
}
main()
{
Badan fungsi main
}
```

Penulisan fungsi/prosedur setelah fungsi main

```
tipe_fungsi nama_fungsi(parameter input);
void nama_prosedur(parameter input,parameter output);
main()
   {
         Badan fungsi main
   }
   ..............
tipe_fungsi nama_fungsi(parameter input)
   {
         Badan Fungsi
   }
void nama_prosedur(parameter input,parameter output)
   {
         Badan Prosedur
   }
```

- Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungsi dengan nilai keluaran bertipe integer.
- Untuk fungsi yang memiliki keluaran bertipe bukan integer , maka diperlukan pendefinisian penentu tipe fungsi.
- Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
- Pernyataan yang diberikan untuk memberikan nilai akhir fungsi berupa pernyataan return.
- Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

Punya parameter dan tidak punya parameter

Parameter →nama peubah yang dideklarasikan pada bagian header prosedur.

Parameter dibagi 2, yaitu :

a. Value Parameter (berfungsi sebagai input pada program prosedur)
b. Variabel Parameter (digunakan oleh prosedur untuk berkomunikasi dengan pemanggilnya →sebagai output → untuk pass by reference)

Parameter di prosedur disebut juga formal parameter.

(parameter yang dideklarasikan di bagian header). Untuk pemanggil disebut actual parameter.

PARAMETER DALAM PROSEDUR

Nilai di dalam suatu modul program Pascal sifatnya adalah lokal, artinya hanya dapat digunakan pada modul atau unit program yang bersangkutan saja, tidak dapat digunakan pada modul atau unit program yang lain.

PENGIRIMAN PARAMETER SECARA NILAI

Bila parameter dikirim secara nilai (by value), parameter formal di prosedur akan berisi nilai yang dikirimkan yang kemudian bersifat lokal di prosedur. Bila nilai parameter formal di dalam prosedur tersebut berubah, tidak akan mempengaruhi nilai parameter nyata.

Pengiriman nilai ini merupakan pengiriman searah yang tidak dikirimkan balik dari parameter formal ke parameter nyata. Parameter yang digunakan dengan pengiriman secara nilai ini disebut dengan parameter nilai (value parameter)

Passing Parameter by value

Cara penulisan di parameter formal

double Fak (int X);

Cara penulisan di parameter aktual

A=5;

Hasil = Fak (A);

Isi memori sebelum fungsi dijalankan

Nama Variabel	Isi	Alamat	Keterangan
A	5	100	Lokasi memori untuk fungsi main
Hasil	?	102	
		:	
X	?	500	Lokasi memori untuk fungsi Fak
		:	

PENGIRIMAN PARAMETER SECARA ACUAN

Bila pengiriman parameter secara acuan (by reference), maka perubahan-perubahan yang terjadi pada nilai parameter formal di prosedur akan mempengaruhi nilai parameter nyata.

Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.

Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.

Fungsi yang menerima kiriman alamat ini akan menggunakan alamat yang sama untuk mendapatkan nilai datanya.

Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.

Pengiriman parameter secara referensi adalah pengiriman dua arah , yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.

Pengiriman secara acuan tidak dapat dilakukan untuk suatu ungkapan.

Yang dikirimkan ke fungsi adalah nama dari suatu variabel

– Cara Penulisan di parameter formal

```
void Tukar (int&Bil1, int &Bil2);
```

– Cara Penulisan di parameter aktual

```
A = 5; B = 1;
Tukar (A,B);
```

Isi memori sebelum fungsi dijalankan

Nama Variabel	Isi	Alamat	Keterangan
A	5	100	Lokasi memori untuk fungsi main
B	1	102	
		:	
bantu		500	Lokasi memori untuk fungsi Tukar
		:	

Isi memori setelah fungsi dipanggil

Nama Variabel	Isi	Alamat	Keterangan
A/Bil1	5	100	Lokasi memori untuk fungsi main
B/Bil2	1	102	
		:	
bantu		500	Lokasi memori untuk fungsi Tukar
		:	

Passing Parameter by Pointer

Yang dikirimkan ke fungsi adalah alamat dari suatu variabel

– Cara Penulisan di parameter formal

```
void Tukar (int*Bil1, int *Bil2);
```

– Cara Penulisan di parameter aktual

```
A = 5; B = 1;
Tukar (&A,&B);
```

Isi memori sebelum fungsi dijalankan

Nama Variabel	Isi	Alamat	Keterangan
A	5	100	Lokasi memori untuk fungsi main
B	1	102	
		:	
Bil1	?	500	Lokasi memori untuk fungsi Tukar
Bil2	?	501	
bantu	?	502	
		:	

Isi memori setelah fungsi dipanggil

Nama Variabel	Isi	Alamat	Keterangan
A	5	100	Lokasi memori untuk fungsi main
B	1	102	
		:	
Bil1	100	500	Lokasi memori untuk fungsi Tukar
Bil2	102	501	
bantu	?	502	

Formal VS Actual Parameter

- Actual parameter dan formal parameter harus digunakan secara berurutan.
- Jumlah kedua parameter harus sama.
- Tipe data juga harus sama
- Actual parameter yang berhubungan dengan value parameter dapat mempunyai banyak bentuk.

→ Sub rutin hendaknya jangan memakai variabel global!!!!

```
#include<stdio.h>
int HITUNG(int A, int B);

void main()
{
 int A,B,T;
 A=5; B=2;T=0;
 T = HITUNG(A,B);
 printf("\n %d", T);
}

int HITUNG(int A, int B)
{  int T;
   A = A * 2;
   B = B * 2;
   T= A+B;
   return(T);
}
```

Bagian ini yang disebut : **main** program atau program induk atau disebut juga Fungsi Induk atau Function main Oleh Bahasa C diberi nama **main( )**

Dalam program induk ada instruksi yang memanggil atau menCALL Function lain, baik fungsi yang kita buat sendiri, maupun fungsi pustaka yang disediakan oleh C/C++

Bagian ini memuat fungsi. Fungsi ini fungsi yang kita buat sendiri
Fungsi ini mempunyai :
Nama : HITUNG
Tipe : int

Dalam contoh ini Fungsi HITUNG ditulis dibawah atau sesudah fungsi main Sebuah Fungsi dapat juga ditulis diatas atau sebelum Fungsi main()

Contoh

Mengitun luas dan keliling lingkaran menggunakan fungsi:

```
#include <stdio.h>
float luas_ling(float);
float kel_ling(float);
float phi = 3.14;
main()
{
    float jari,luas, keliling;
    printf ("Masukkan jari-jari =");
    scanf ("%f",&jari);
    luas = luas_ling(jari);
    keliling = kel_ling(jari);
    printf("Luas Lingkaran dengan jari %.of adalah %.2f \n",jari,luas);
    printf("Keliling Lingkaran dengan jari %.of adalah %.2f
    \n",jari,keliling);
}
```

```
float luas_ling(float r)
{
        float  L;
        L = phi * r * r;
        return (L);
}
float kel_ling(float r)
{
        float K;
        K = 2 * phi * r;
        return (K);
}
```

1.

Sebuah fungsi matematika didefinisikan sebagai berikut :

$$F(X) = 3X_1^2 + 6X_2 + 2$$

Dimana nilai X_1 dan X_2 adalah bilangan bulat, buatlah fungsi dari rumus matematika tersebut dan gunakan dalam program utama.

2.

Buat menu untuk pemilihan salah satu dari operasi matematika, yaitu penjumlahan, pengurangan, perkalian, atau pembagian dari dua buah bilangan yang dimasukkan. Pemasukan data dan proses perhitungan dari setiap operasi matematika tersebut dilakukan oleh sub rutin.

Pemilihan dilakukan dengan memasukkan angka pilihan, yaitu :

1 untuk penjumlahan

2 untuk pengurangan

3 untuk perkalian

4 untuk pembagian

0 untuk keluar dari program

Jika pilihan yang dimasukkan tidak 1, 2, 3 atau 4, maka akan muncul pesan kesalahan pemilihan dan perhitungan dapat dilakukan secara berulang kali.