ASPEK PROSEDURAL
DALAM KONTEKS DEKLARATIF

Pendahuluan

Pada bagian ini hendak ditunjukkan bagaimana menyelesaikan sebuah persoalan yang bersifat prosedural dengan pemrograman deklaratif.

Aksioma : **A if B_1 and B_2 and B_3 and ..... B_4**

dapat ditulis sebagai prosedur :
A adalah **procedure head**
B_i adalah **body**

A benar, jika B benar
dengan pengertian A benar, jika B benar

dan dituliskan menjadi
A ← B1, B2, B3, ...Bn

Pada akhirnya diterjemahkan sebagai sekuens/urutan pemanggilan "prosedur" B1, B2, B3, ...Bn dengan parameter yang sesuai.

```
A ← B1,
    B2,
    B3,
     .....
    Bn
```

Prosedur dan Parameter Formal

Parameter formal sebuah prosedur dalam pemrograman prosedural dapat digolongkan menjadi :

- parameter input, yang nilainya harus terdefinisi pada saat pemanggilan prosedur (pada parameter aktual)
- parameter output, yang nilainya tidak perlu terdefinisi pada saat pemanggilan, namun justru terdefinisi berkat pelaksanaan prosedur
- parameter input/output, yang nilainya harus terdefinisi pada saat pemanggilan prosedur, dan nilai tersebut diubah pada pelaksanaan prosedur.

Parameter pada Pemrograman Deklaratif

Parameter adalah argumen dari sebuah predikat.
Pada pernyataan FAKTA atau ATURAN dalam **pemrograman deklaratif murni**, tidak ada masalah, sebab semua parameter adalah parameter masukan, dan evaluasi predikat menghasilkan nilai benar/salah berdasarkan nilai parameter aktual.
Contoh : Predikat Ayah(X,Y), Lelaki(X) pada kasus pohon keluarga.

Pada pernyataan FAKTA/ATURAN yang **bukan predikat murni:**
misalnya pada kasus, seperti Plus(X,Y,Z) : Plus(X,Y,Z) adalah benar, jika Z=X+Y, X dan Y adalah parameter input, sedangkan Z adalah parameter output.

Pada kasus, seperti INCR(X) : INCR X adalah benar, jika X pada akhir pelaksanaan evaluasi predikat adalah berisi succesor X, parameter X adalah sekaligus input dan output.
INCR(X) $\leftarrow$ X is succ(X).
Pada pemrograman deklaratif hal ini **tidak boleh terjadi**. Sebuah parameter hanya dapat menjadi parameter input atau output, dan tidak keduanya.

Eksekusi Passing Parameter dalam PROLOG

Pada pemrograman deklaratif, khususnya PROLOG, passing parameter dilakukan melalui proses unifikasi yang terjadi pada saat mekanisme inferensi dilakukan oleh "mesin PROLOG".

Aspek Prosedural

Yang dimaksudkan dengan aspek prosedural adalah konstruktor yang dibutuhkan dan menjadi dasar pada pemrograman prosedural dan sangat "aksional", yaitu:

- assignment
- input/output
- sekuens
- analisa kasus, yang mengandung aksi
- pengulangan
- file eksternal
- basisdata

Assignment

Assignment adalah merupakan instruksi paling mendasar pada program prosedural, karena dengan instruksi ini suatu "nama" pada tempat penyimpanan diberi nilai, yang selanjutnya dapat dimanipulasi sesuai dengan tujuan program.

Program deklaratif murni TIDAK memerlukan assignment, karena dasarnya bukan manipulasi/kalkulasi isi memori melainkan pencocokan.

Input/Output
Input adalah "aksi" untuk membaca nilai dari suatu "nama" dari suatu piranti masukan. Jika pengisian nilai nama informasi pada assignment dilakukan secara "hard coded" dalam teks program, maka input memungkinkan untuk mengisi nilai nama informasi dari piranti keluaran sehingga program dapat berkelakukan sesuai dengan nilai yang diberikan, tanpa mengubah source code.

Output adalah aksi untuk menuliskan nilai (isi) dari suatu nama informasi ke suatu piranti keluaran seperti layar atau printer. Berkat aksi penulisan inilah isi suatu nama informasi dapat dikomunikasikan ke dunia luar, khususnya pengguna.

Program deklaratif murni TIDAK memerlukan aksi input/output karena pada hakekatnya pemakai melakukan Query, dan Query tersebut akan dicari jawabannya berdasarkan "pencocokan".

Sekuens
Aspek sekuens pada eksekusi program deklaratif dibedakan menjadi :

- aspek sekuens yang dituliskan karena keterbatasan kalkulus orde pertama pada program deklaratif murni. Contoh : pada program- program yang pernah dibahas saat ini.
- aspek sekuens yang benar-benar merupakan sekuens pemanggilan prosedur yang dikehendaki agar dilaksanakan sesuai urut-urutan "pemanggilan". Contoh : pada bagian program yang merealisasikan evaluasi ekspresi aritmatika, dan pada program-program yang mengandung instruksi "aksional" seperti input/output.

Aspek sekuens yang dibahas pada bab ini adalah aspek sekuens prosedural, di mana final state dari suatu "pemanggilan prosedur" akan menjadi initial state pada pemanggilan prosedur berikutnya.

Analisa Kasus
Pada pemrograman prosedural, analisa kasus berarti mendefinisikan kondisi yang mungkin terjadi, dan berdasarkan kondisi yang benar akan dilaksanakan suatu aksi yang sesuai. Lihat contoh pada terjemahan analisa kasus.

Dasar analisa kasus pada pemrograman deklaratif berbeda dengan analisa kasus pada pemrograman prosedural.

Program deklaratif murni mengandung analisa kasus untuk merealisasi suatu aturan yang bersifat "OR". Namun analisa kasus pada program deklaratif murni bukan aksional, artinya berdasarkan kondisi yang dipenuhi maka program bukan melakukan suatu aksi, melainkan melakuan pencocokan terhadap aturan berikutnya yang sesuai dengan kondisi.

Ini menjadi nyata dalam contoh max2(A,B) pada konteks deklaratif dengan pada konsteks prosedural sebagai berikut :

Deklaratif :

Maksimum dua harga versi-1

FAKTA DAN ATURAN

{ max2(X,Y, Z) benar, jika Z adalah maksimum dari X dan Y.

max2(X,Y,X) benar, jika X >= Y

max2(X,Y,Y) benar, jika X <= Y }

{Basis }

```
max2(X,Y,X) ← X >= Y.
max2(X,Y,Y) ← X < Y.
```

Prosedural :

Program MAXAB

{ dibaca nilai X dan Y, menuliskan Y jika X≤ Y, menuliskan Y jika Y > A}

```
Kamus
   X,Y : integer
```

```
Algoritma :
   input (X,Y)
   depend on (a,b)
      X ≤ Y: output (X)
      X < Y :  output (Y)
```

Pengulangan

Program prosedural memerlukan struktur kontrol pengulangan, karena pengulangan adalah struktur kontrol yang sangat dibutuhkan dalam memprogram secara prosedural. Pengulangan merupakan salah satu alat yang ampuh untuk melakukan abstraksi dari beberapa sekuens instruksi, yang memungkinkan program prosedural menjadi singkat penulisannya namun berulang-ulang eksekusinya.

Ada 5 macam bentuk pengulangan (ref : diktat Pemrograman Prosedural), yaitu repeat x times, for, repeat, while dan iterate. Satu bentuk pengulangan dapat ditransformasi menjadi bentuk yang lain. Karena bentuk yang umum tersedia pada bahasa pemrograman hanya empat bentuk yang terakhir, maka pada bagian ini, hanya empat bentuk yaitu : for, repeat, while dan iterate.

Semua bentuk pengulangan mempunyai komponen yang sama, yaitu :

- Kata kunci yang menyatakan Bentuk pengulangan (for, while, repeat, iterate)
- Kondisi berhenti atau kondisi pengulangan
- “Aksi” yang akan diulang
- Inisialisasi, yaitu aksi yang dilakukan sekali saja sebelum pengulangan
- Terminasi, yaitu aksi yang dilakukan sekali saja setelah pengulangan berakhir

Pengulangan pada pemrograman deklaratif sampai dengan contoh-contoh yang diberikan pada bab sebelumnya dapat “dihilangkan” berkat pemanggilan rekursif dari suatu aturan. Program deklaratif murni tidak membutuhkan pengulangan karena pengulangan adalah sangat aksional/prosedural.

Pada akhirnya, aturan rekursif akan dipakai untuk melakukan "pengulangan" dalam konteks deklaratif. Jadi suatu pengulangan akan mempunyai bagian basis (yang menyebabkan berhenti) dan bagian rekurens (yang memberi **efek** pengulangan)
Yang perlu disadari adalah, bahwa walaupun pada teks program tampaknya sama, namun analisa rekurens adalah berbeda cara konstruksinya dengan analisa iteratif.

Namun, seringkali dibutuhkan suatu pengulangan misalnya :

- untuk tidak mengulang-ulang mengetik suatu Query yang sejenis
- jika diperlukan menu yang menawarkan eksekusi beberapa query
- jika harus terjadi "tanya jawab" (dialog) antara program dengan pemakai, dimana program mengajukan pertanyaan, dan menunggu sampai pengguna menjawab, kemudian berdasarkan jawaban itu program akan melakukan pencocokan terhadap aturan sesuai dengan pilihan

File Eksternal
File eksternal adalah media penyimpanan nilai-nilai pada tempat penyimpanan sekunder. Karena menyangkut tempat penyimpanan (memori), maka merupakan aspek prosedural.

Program deklaratif tidak mengenal tempat penyimpanan karena program terdiri dari fakta serta aturan, dan pengguna melakukan Query. Semua fakta dan aturan tersimpan di dalam program, dan biasanya Query diajukan ke program secara interaktif. Maka program deklaratif murni tidak memerlukan file eksternal. Namun seringkali list yang dimanipulasi sebagai objek internal dalam program deklaratif dikehendaki bukan sebagai "konstanta" di dalam program, atau jika merupakan "hasil" masih dapat diperoleh kembali karena "disimpan". Maka beberapa program deklaratif memerlukan file eksternal untuk tempat penyimpanan sehingga dapat diakses kembali.

SARAN UNTUK PEMAKAIAN ASPEK PROSEDURAL DALAM KONTEKS DEKLARATIF

1. Tuliskanlah bagian yang murni deklaratif sebagai "inti/nukleus" program.
2. Pisahkan aspek prosedural yang diperlukan (biasanya untuk interaksi, dialog dan penyimpanan nilai) dalam suatu aturan yang sebenarnya men-simulasi "prosedur" dalam pemrograman prosedural. Setiap aturan dan fakta yang ada pada program deklaratif harus jelas, untuk kelompok deklaratif murni atau yang diperlukan untuk bagian prosedural.
3. Pertimbangkan dengan baik, "bobot" deklaratif dibandingkan dengan prosedural. Jika ternyata kelas persoalan lebih berat ke arah prosedural, tuliskanlah dalam bahasa pemrograman prosedural.

Tidak disarankan untuk menuliskan sebuah program deklaratif dengan "cara berpikir prosedural" seperti pada contoh evaluasi ekspresi aritmatika yang diberikan pada bab ini.

Biasanya, instruksi yang bersifat prosedural sangat spesifik terhadap pemroses bahasanya. Pada diktat ini, diberikan kasus translasi dari konstruktor prosedural ke program deklaratif, langsung dalam bahasa Turbo Prolog.

ASPEK PROSEDURAL
DALAM BAHASA PROLOG

Pada konteks bahasa PROLOG (khususnya: Turbo Prolog) juga terdapat beberapa struktur kontrol yang merupakan struktur kontrol pada program prosedural, yaitu :

1. Assignment
2. Input/Output
3. Sekuens
4. Kondisional
5. Pengulangan
6. External Sequential File
7. Basis Data

1. Assignment

a. Assignment tipe dasar

Sintaks :

a. Jenis eksplisit, dengan operator = :

```
<nama-variabel bebas> = <nama-variabel-terikat>
```

b. Jenis implisit, terjadi pada saat unifikasi.

Contoh :

Notasi Algoritmik	Notasi Turbo Prolog

Kamus
  i,j : integer
  cc,bc : character
  found, yes : boolean
 	
Algoritma
  i ← 5
  j ← i
  cc ← 'W'
  bc ← cc
  Found ← false
  Yes ← Found
	Domains
	Predicates
	  assign_i(integer,integer)
	  assign_c(char,char)
	  assign_s(symbol,symbol
	Goal /* query internal */
	  go
	Clauses
	  go :-
	  assign_i(5,I),
	  assign_i(I,J),
	  assign_c('W',CC),
	  assign_c(CC,BC),
	  assign_s(false,Found),
	  assign_s(Found,Yes).
	
	  assign_i(A,B) :- B = A.
	  assign_c(A,B) :- B = A.
	  assign_s(A,B) :- B = A.

Catatan :

1. Predikat **go** hanya diperlukan untuk melakukan eksekusi dari seluruh deretan sekuens sehingga menjadi satu kesatuan program.
2. Perhatikan bahwa tanda "=" pada program deklaratif bukan instruksi assignment, melainkan hanya "mencocokkan" bahwa B diinstansiasi sama dengan A.

b. Assignment tipe terstruktur

Sintaks :

a. tipe tanpa nama, diimplementasikan dengan list

```
[<nama-elemen-1>,...,<nama-elemen-n>]
```

b. tipe dengan nama, diimplementasikan dengan functor

```
<nama-tipe>(<nama-elemen-1>,...,<nama-elemen-n>)
```

Contoh :

Notasi Algoritmik

```
Kamus
  type date :
        <d:integer[0..31],
         m:integer[0..12],
         y:integer[0..99]>
  type mhs :
        <nim:integer,
         nama:string[20],
         nilai:real>
  now, nextday : date
  ali, badu : mhs



Algoritma
  now.d ← 1
  now.m ← 2
  now.y ← 96
  nextday ← now

  ali.nim ← 12345
  ali.nama ← 'Ali SS'
  ali.nilai ← 75.8
  badu ← ali
```

Notasi Turbo Prolog

```
Domains
  d,m,y = integer
  ddate = date(d,m,y)
  nim = integer
  nama = string
  nilai = real
  dmhs = mhs(nim,nama,nilai)
Predicates
  assign_d(ddate,ddate)
  assign_m(dmhs,dmhs)
Goal /* query internal */
  go
Clauses
  go :-
    assign_d(date(1,2,96),Now),


    assign_d(Now,Nextday),

    assign_m(mhs(12345,
             'Ali SS',75.8),Ali),

    assign_m(Ali,Badu).

  assign_d(A,B) :- B = A.
  assign_m(A,B) :- B = A.
```

2. Input/Output Secara Interaktif

Sintaks : (predikat sistem)

```
read(X) benar, jika hasil pembacaan (sebanyak satu kali) dari
      'current input device' dapat berunifikasi dengan X.
write(X) benar, jika sistem selesai melakukan penulisan X ke
      'current output device' sebanyak satu kali.
nl menyatakan newline, untuk menuliskan pada baris baru.
```

Contoh :

Notasi Algoritmik	Notasi Turbo Prolog

Kamus
  a, b : integer
 	
Algoritma
  input(a)
  input(b)
  output(a)
  output(b)
	Domains
	Predicates
	  go
	Goal /* query internal */
	  go
	Clauses
	  go :-
	  read(A),
	  read(B),
	  write(A), nl,
	  write(B), nl.

3. Sekuens

Dengan cara menuliskan predikat-predikat berurutan pada bagian kanan dari aturan.

Sintaks :

```
<predikat-1>,
<predikat-2>,
...,
<predikat-n>.
```

Contoh :

Notasi Algoritmik	Notasi Turbo Prolog

KALI
 	
Kamus
  a, b : integer
  c : integer
 	
Algoritma
  input(a)
  input(b)
  c ← a * b
  output(c)
	KALI
	
	Domains
	Predicates
	  go
	Goal /* query internal */
	  go
	Clauses
	  go :- read(A),
	  read(B),
	  C = A * B,
	  write(C), nl.

4. **Kondisional**

Dengan cara menuliskan kondisi-kondisi secara implist dalam argumen predikat; predikat dapat berbentuk fakta maupun aturan.

Sintaks :

```
<nama-predikat>(<kasus-1>).
<nama-predikat>(<kasus-2>).
...
<nama-predikat>(<kasus-n>).
```

Contoh :

Notasi Algoritmik	Notasi Turbo Prolog

IF1
{ realisasi tanpa procedure }
Kamus
  a : integer
 	
Algoritma
  input(a)
  if a > 0 then
  output('Positif')
	IF1
	
	Domains
	  ddata = integer
	Predicates
	  suhu(ddata)
	Goal /* internal */
	  go
	Clauses
	  go :- read(A),
	  A > 0,
	  write('Positif'), nl.

IF2
{ realisasi dengan procedure }
Kamus
  a : integer
 	
Algoritma
  input(a)
  suhu(a)
 	
procedure suhu (input A:integer)
{ menuliskan 'Positif' jika A positif }
Algoritma
  if a > 0 then
  output('Positif')
	IF2
	
	Domains
	  ddata = integer
	Predicates
	  suhu(ddata)
	Goal /* internal */
	  go
	Clauses
	  go :- read(A),
	  suhu(A).
	
	  suhu(A) :- A > 0,
	  write('Positif'), nl.

IF-ELSE
 	
Kamus
  a : integer
 	
Algoritma
  input(a)
  if a > 0 then
  output('Positif')
  else
  output('Negatif')
	IF-ELSE
	
	Domains
	  ddata = integer
	Predicates
	  suhu(ddata)
	Goal /* internal */
	  go
	Clauses
	  go :- read(A),
	  suhu(A).
	
	  suhu(A) :- A > 0,
	  write('Positif'), nl.
	  suhu(A) :- A <= 0,
	  write('Negatif'), nl.

(lanjutan Kondisional)

Notasi Algoritmik	Notasi Turbo Prolog

IF-BANYAK
 	
Kamus
  a : integer
 	
Algoritma
  input(a)
  if a > 0 then
  output('Positif')
  else
  if a = 0 then
  output('Nol')
  else
  output('Negatif')
	IF-BANYAK
	
	Domains
	  ddata = integer
	Predicates
	  suhu(ddata)
	Goal /* internal */
	  go
	Clauses
	  go :- read(A),
	  suhu(A).
	
	  suhu(A) :- A > 0,
	  write('Positif').
	  suhu(0) :- write('Nol').
	  suhu(A) :- A < 0,
	  write('Negatif').

Contoh : pada analisa kasus masalah penentuan nilai maksimum antara dua harga.

Notasi Algoritmik	Notasi Turbo Prolog

MAKSIMUM 2 HARGA
 	
Kamus
  x,y : integer
 	
Algoritma
  input(x)
  input(y)
  depend on (x,y)
  x ≤ y : output(x)
  x < y : output(y)
	MAKSIMUM 2 HARGA
	
	Domains
	  ddata = integer
	Predicates
	  max2(ddata,ddata,ddata)
	Goal /* internal */
	  go
	Clauses
	  go :- read(X),
	  read(Y),
	  max2(X,Y,NMax),
	  write(NMax).
	
	  max2(X,Y,X) :- X >= Y.
	  max2(X,Y,Y) :- X < Y.

5. Pengulangan

Pada bagian ini akan diberikan "terjemahan" dalam bahasa Prolog dari bentuk pengulangan yang sering dipakai pada pemrograman prosedural, yaitu for, repeat, while, iterate, dengan mendefinisikan predikat yang menyatakan:

- Bentuk pengulangan (for, while, repeat, iterate)
- Kondisi berhenti atau kondisi pengulangan
- "Aksi" yang akan diulang

Perhatikanlah bahwa rekurens dipakai sebagai sarana untuk mengulang.

PENGULANGAN

FAKTA DAN ATURAN

```
/* POLA UMUM */
/* Kondisi STOP : Basis */
   pengulangan :- aturan-stop.
/* Kondisi PENGULANGAN : REKURENS */
   pengulangan :- aturan-ulang,
                  aksiyang diulang,
                  perubahan pengontrol pengulangan,
                  pengulangan.
```

```
/* REPEAT */
/* for */
/* bentuk umum : */
/* for I in [Awal..Akhir]
      Aksi
*/
/* Kondisi berhenti */
   for(I,Akhir) :- I> Akhir.
/* Kondisi pengulangan */
   for(I,Akhir) :- I <= Akhir, Aksi, for(I,Akhir).
```

```
/* REPEAT */
/* bentuk umum : */
/* repeat
     Aksi
    until (kondisiberhenti)
*/
/* contoh jika kondisi berhenti tergantung I dan Akhir */
/* Kondisi pengulangan */
   repeat(I,Akhir) :- Aksi, until(I,Akhir).
   until(I,Akhir) :- I<= Akhir, repeat(I,Akhir).
/* Kondisi berhenti */
   until(I,Akhir) :- I> Akhir.
```

```
/* WHILE */
/* bentuk umum : */
/* while (kondisi) do   */
/*     Aksi                                                  */
/* contoh jika kondisi berhenti tergantung I dan Akhir */
/* Kondisi berhenti */
   while (I,Akhir) :- I> Akhir.
/* Kondisi pengulangan */
   while (I,Akhir) :- I <= Akhir, Aksi, while(I,Akhir).
```

```
/* ITERATE */
/* bentuk umum : */
/* Iterate
     Aksi1
   Stop : kondisiberhenti
     Aksi2
*/
/* contoh jika kondisi berhenti tergantung I dan Akhir */
/* Kondisi pengulangan */
   iterate (I,Akhir) :- Aksi1, kondisi (I,Akhir).
   kondisi(I,Akhir) :- I <> Akhir,
                       Aksi2, iterate(I,Akhir)
/* Kondisi berhenti */
   kondisi (I,Akhir) :- I= Akhir.
```

Contoh aplikasi :

Notasi Algoritmik	Notasi Turbo Prolog

FOR
 	
Kamus
  i : integer
 	
Algoritma
  i traversal [0..4]
  output(i)
	FOR
	
	Domains
	  dindeks = integer
	Predicates
	  for(dindeks,dindeks)
	  forAksi(dindeks,dindeks)
	Goal /* internal */
	  for(0,4)
	Clauses
	  for(Awal,Akhir) :-
	  I = Awal,
	  forAksi(I,Akhir).
	
	  forAksi(I,Akhir) :-
	  I <= Akhir,
	  write(I), nl,
	  I1 = I + 1,
	  forAksi(I1,Akhir).
	  forAksi(I,Akhir) :-
	  I > Akhir.

REPEAT-1
 	
Kamus
  i : integer
 	
Algoritma
  i ← 1
  repeat
  output(i)
  i ← i+1
  until i > 10
	REPEAT-1
	
	Domains
	  dindeks = integer
	Predicates
	  repeat(dindeks,dindeks,dindeks)
	  repeatAksi(dindeks,dindeks,
	  dindeks)
	  until(dindeks,dindeks,dindeks)
	Goal /* internal */
	  repeat(1,1,10)
	Clauses
	  repeat(Awal,Next,Akhir) :-
	  I = Awal,
	  repeatAksi(I,Next,Akhir).
	
	  repeatAksi(I,Next,Akhir) :-
	  write(I), nl,
	  I1 = I + Next,
	  until(I1,Next,Akhir).
	
	  /* kondisi ulang */
	  until(I,Next,Akhir) :-
	  I <= Akhir,
	  repeatAksi(I,Next,Akhir).
	  /* kondisi berhenti */
	  until(I,_,Akhir) :-
	  I > Akhir.

(lanjutan Pengulangan)

Notasi Algoritmik	Notasi Turbo Prolog

REPEAT-2
 	
Kamus
  i : integer
 	
Algoritma
  repeat
  input(i)
  print(i)
  until i = 999
	REPEAT-2
	
	Domains
	Predicates
	  go
	  repeat
	Goal /* internal */
	  go
	Clauses
	  go :- repeat,
	  read(I),
	  write(I), nl,
	  I = 999.
	
	  repeat.
	  repeat :- repeat.

WHILE-1
 	
Kamus
  i : integer
 	
Algoritma
  i ← 1
  while i <= 10 do
  output(i)
  i ← i+1
	WHILE-1
	
	Domains
	  dindeks = integer
	Predicates
	  while(dindeks,dindeks,dindeks)
	  whileAksi(dindeks,dindeks,
	  dindeks)
	Goal /* internal */
	  while(1,1,10)
	Clauses
	  while(Awal,Next,Akhir) :-
	  I = Awal,
	  whileAksi(I,Next,Akhir).
	
	  whileAksi(I,Next,Akhir) :-
	  I <= Akhir,
	  write(I), nl,
	  I1 = I + Next,
	  whileAksi(I1,Next,Akhir).
	  whileAksi(I,_,Akhir) :-
	  I > Akhir.

(lanjutan Pengulangan)

Notasi Algoritmik	Notasi Turbo Prolog

WHILE-2
 	
Kamus
  i : integer
 	
Algoritma
  input(i)
  while i <> 999 do
  output(i)
  input(i)
	WHILE-2
	
	Domains
	  ddata = integer
	Predicates
	  while(ddata)
	  whileAksi(ddata,ddata)
	Goal /* internal */
	  while(999)
	Clauses
	  while(Akhir) :-
	  read(I),
	  whileAksi(I,Akhir).
	
	  whileAksi(I,Akhir) :-
	  I <> 999,
	  write(I1), nl,
	  read(I1),
	  whileAksi(I1,Akhir).
	  whileAksi(Akhir,Akhir).