

BAB III

NOTASI ALGORITMIK

1. PENDAHULUAN

Bab ini memiliki kompetensi dasar untuk Memahami Notasi algoritma dan struktur program bahasa C++. Teks algoritma tidak sama dengan teks program komputer. Tidak ada standarisasi untuk menuliskan teks algoritma, setiap orang dapat saja membuat notasi algoritmiknya sendiri, sedangkan program komputer adalah realisasi teknis dari algoritma dalam notasi bahasa pemrograman.

Agar notasi algoritma mudah ditranslasikan kedalam notasi bahasa pemrograman, maka sebaliknya notasi algoritmik tersebut berupa *pseudo-code* yang berkoresponden dengan bahasa pemrograman secara umum.

2. PENYAJIAN

2.1. Contoh Notasi

Contoh 1 :

Pernyataan dalam notasi deskriptif : **tulis nilai X**

Maka pseudocodenya dalam notasi algoritmik menjadi: **write(X)** atau **output(X)**

Notasi algoritmik dalam bentuk pseudo-code mudah ditranslasikan kedalam notasi bahasa pemrograman. Notasi write di dalam algoritma berkoresponden dengan *printf* dalam bahasa C atau *writeln* dalam bahasa pascal atau *cout* dalam bahasa C++.

Jadi translasi **write(X,Y)** ke dalam masing-masing bahasa tersebut adalah:

Bahasa C	<code>printf("%d ",X);</code>
Bahasa pascal	<code>writeln(X);</code>
Bahasa C++	<code>cout << X;</code>

Contoh 2 :

Pernyataan dalam notasi deskriptif : **isikan nilai X ke dalam min**

Pseudo-code dalam notasi algoritmik menjadi : **min ← X**

Notasi " ← " berarti mengisi (*assign*) peubah (*variable*) min dengan nilai X.

Bahasa C	min = X;
Bahasa Pascal	min := X;
Bahasa C++	min = X;

2.2. Struktur Teks Algoritma

Ada dua buah algoritma sederhana, contoh ilustrasi mengenai teks algoritma

Pertama : Hello world

Program ini pertama kali dikemukakan oleh Brian W. Kernighsn (penemu bahasa C)

Algoritma untuk menulis Hello world:

```

Algoritma Hello_world
{ program untuk mencetak "Hello world"}

DEKLARASI
  {tidak ada}

DESKRIPSI
  Write("Hello world")

```

Kedua : luas segiempat

Yaitu untuk menghitung luas segiempat dengan menggunakan rumus: Luas = panjang x lebar.

Program menghitung luas segiempat adalah:

```

PROGRAM LuasSegi4
  {program untuk menghitung luas segiempat dengan
   diketahui panjang dan lebarnya}

DEKLARASI
  Luas,panjang,lebar :integer

ALGORITMA
  panjang ← 10
  lebar ← 5
  Luas ← panjang * lebar
  write(Luas)

```

Atau :

Algoritma LuasSegi4

{program untuk menghitung luas segiempat dengan diketahui panjang dan lebarnya}

DEKLARASI

Luas, panjang, lebar : integer

DESKRIPSI

panjang \leftarrow 10
 lebar \leftarrow 5
 Luas \leftarrow panjang * lebar
write(Luas)

Atau :

LuasSegi4

{algoritma untuk menghitung luas segiempat dengan diketahui panjang dan lebarnya}

KAMUS

Luas, panjang, lebar : integer

ALGORITMA

panjang \leftarrow 10
 lebar \leftarrow 5
 Luas \leftarrow panjang * lebar
write(Luas)

Dengan melihat contoh sederhana di atas, ada beberapa format penulisan algoritma yang dipakai, tetapi pada intinya sama saja. Maka pada dasarnya algoritma selalu disusun berdasarkan 3 bagian, antara lain:

1. Judul Algoritma

Judul adalah bagian yang terdiri atas nama algoritma dan penjelasan tentang algoritma tersebut. Nama algoritma sebaiknya singkat dan jelas dan algoritmanya juga harus sesuai dengan spesifikasi masalah yang diberikan.

Misalnya contoh algoritma di atas, mempunyai judul:

Algoritma Hello_world

{algoritma untuk mencetak "Hello world"}

2. Deklarasi / Kamus

Digunakan untuk mengumumkan semua nama yang dipakai didalam algoritma beserta propertiesnya (misal type). Nama tersebut dapat berupa nama konstanta, nama peubah, nama tipe, nama procedure dan nama fungsi. Semua nama yang dipakai di dalam algoritma harus dideklarasikan sebelum digunakan.

Misalnya, algoritma Hello_world mengandung bagian deklarasi yang kosong, sedangkan algoritma LuasSegi4 berisi deklarasi nama-nama peubah yang digunakan di dalam algoritma beserta tipenya, yaitu:

DEKLARASI

```
Luas, panjang, lebar : integer
```

Penjelasan : peubah Luas, panjang dan lebar digunakan dalam algoritma nama-nama peubah ini beserta tipenya harus diumumkan di bagian deklarasi sebelum mereka digunakan.

3. Algoritma / Deskripsi

Algoritma adalah bagian inti dari sebuah algoritma dari suatu program. Bagian ini berisi deskripsi langkah-langkah penyelesaian masalah berupa pernyataan-pernyataan, pernyataan tersebut ditulis dengan notasi. Setiap pernyataan dalam algoritma dibaca dari “atas” ke “bawah”

Misalnya:

- notasi write yaitu untuk mencetak data/informasi
- read untuk membaca data

2.3. Format Syntax Algoritma

JUDUL ALGORITMA

Nama Algoritma

DEKLARASI/
KAMUS
{Lokal / Umum}

Type namatipe : tipe [subrange]
Type namatipe : array [min..maks] of tipe
 Namavar : tipe
 Namavar : array [min..maks] of tipe
Const nama = nilai
Procedure namaproc
 (Input/Output[daftar_nama_parameter:tipe])

TUBUH
ALGORITMA/
DESKRIPSI

ALGORITMA

Notasi Assigment
 Notasi Kondisional
 Notasi Pengulangan
 Notasi Pemanggilan

URAIAN
ALGORITMA

Kumpulan algoritma masing-masing Procedure ataupun
 Function yang dipanggil dari Tubuh Algoritma

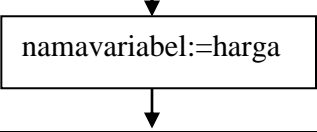
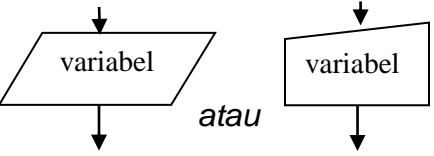
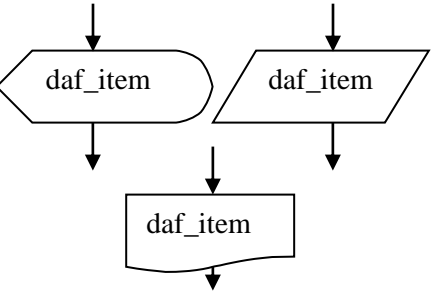
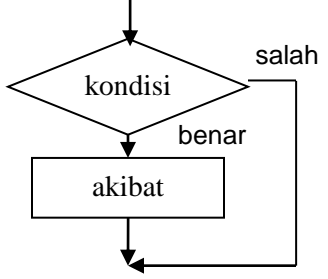
2.4. Translasi Teks Algoritma ke dalam Teks Program

Agar dapat dilaksanakan oleh computer, maka algoritma harus ditranslasikan ke dalam suatu notasi bahasa pemrograman.

Translasi Notasi Kamus pada Algoritma ke Bahasa Pemrograman

Algoritma	Pascal	C dan C++
<i>Kamus/Deklarasi</i>	<i>Deklarasi</i>	<i>Deklarasi</i>
	USES daftar_unit	#include<nama_unit>
<u>CONST</u> namaconst = nilai	CONST namaconst = nilai;	#define namaconst nilai;
<u>TYPE</u> namatipe : <u>tipe</u> data [subrange] <i>Contoh :</i> <u>Type</u> max : <u>integer</u>	TYPE namatipe = tipe;data; <i>Contoh :</i> Type max = integer;	Typedef tipe;data namatipe; <i>Contoh :</i> Typedef int max;
namavar : <u>tipe</u> data [subrange] <i>Contoh :</i> n : <u>integer</u> m : max	VAR namavar : namatipe; <i>Contoh :</i> var n : integer; m : max;	namatipe daftarnamavar; <i>Contoh :</i> int n; max m;

Translasi Notasi Pengendalian Program Pada Algoritma Ke Bahasa Pemrograman

FLOW CHART	ALGORITMA	PASCAL	C	C++
 <pre> graph TD Start(()) --> A[namavariabel:=harga] A --> End(()) </pre>	namavaribel \leftarrow harga	namavar := harga;	namavar = harga;	namavar = harga;
 <pre> graph TD Start(()) --> A[/variabel/] A --> End(()) Start2(()) --> B[variabel] B --> End2(()) </pre> <p>atau</p>	<u>INPUT</u> (daftarvariabel) atau <u>read</u> (daftarvariabel)	read(daftarvariabel); readln(daftarvariabel); readln ;	scanf("%format", &var); getch(varchar); getche():	cin >> var; <i>var dapat berupa :</i> var = getch(); var = getche();
 <pre> graph TD Start(()) --> A[/daf_item/] A --> End(()) Start2(()) --> B[/daf_item/] B --> C[daf_item] C --> End2(()) </pre>	<u>OUTPUT</u> (daftar_item) atau <u>write</u> (daftar_item) <i>Item dapat berupa :</i> Variable, konstanta, string, character, angka, nilai Boolean, dan ekspresi	write(daftar_item); writeln(daftar_item):	printf("string dan %format", var); printf("string"); putch("char"); puts("string");	cout << var; cout << konstanta;
 <pre> graph TD Start(()) --> A{kondisi} A -- benar --> B[akibat] B --> End(()) A -- salah --> End </pre>	<u>IF</u> <kondisi> <u>THEN</u> akibat <u>ENDIF</u>	IF (kondisi) THEN akibat; atau IF (kondisi) THEN Begin akibat; End;	if (kondisi) akibat; atau if (kondisi) { akibat; }	if (kondisi) akibat; atau if (kondisi) { akibat; }
Catatan : akibat bisa berupa pernyataan tunggal (sintaks tanpa tanda kurung) maupun majemuk (dengan tanda kurung)				

FLOW CHART	ALGORITMA	PASCAL	C	C++
<pre> graph TD Start(()) --> Kondisi{kondisi} Kondisi -- Benar --> Akibat1[akibat1] Kondisi -- Salah --> Akibat2[akibat2] Akibat1 --> End1(()) Akibat2 --> End2(()) </pre>	<p><u>IF</u> <kondisi> <u>THEN</u> akibat1 <u>ELSE</u> akibat2 <u>ENDIF</u></p> <p>Catatan : Setelah <i>e/se</i>, dapat bersarang (nested) notasi percabangan lainnya.</p>	<p>IF (kondisi) THEN akibat1; ELSE akibat2</p> <p><i>atau</i></p> <p>IF (kondisi) THEN Begin akibat1; End ELSE Begin akibat2 End;</p>	<p>If (kondisi) akibat1; else akibat2;</p> <p><i>atau</i></p> <p>If (kondisi) { akibat1; } else { akibat2; }</p>	<p>If (kondisi) akibat1; else akibat2;</p> <p><i>atau</i></p> <p>If (kondisi) { akibat1; } else { akibat2; }</p>
<pre> graph TD Start(()) --> Kondisi{kondisi} Kondisi -- harga1 --> Akibat1_1[Akibat1] Kondisi -- harga2 --> Akibat1_2[Akibat1] Kondisi -- harga_n --> Akibat1_3[Akibat1] Kondisi -- else --> AkibatLain[Akibat lain] Akibat1_1 --> Join(()) Akibat1_2 --> Join Akibat1_3 --> Join AkibatLain --> Join Join --> End(()) </pre>	<p><u>DEPEND ON</u> <ekspresi> <ekspresi 1> : akibat 1 <ekspresi 2> : akibat 2 : <ekspresi n> : akibat n <i>atau</i> <u>CASE</u> namavarcase <u>OF</u> expkonstan 1 : akibat1 expkonstan 2 : akibat2 : expkonstan n : akibatn <u>ELSE</u> akibatlain <u>ENDCASE</u></p>	<p>Case varcase Of Nilai1: akibat1; Nilai2: akibat2; : : Nilain: akibatn; Else akibatlain; End;</p>	<p>Switch (ekspresi) { case nilai1: akibat1; break; case nilai2: akibat2; break; : : case nilain: akibatn; break; default:akibatlain; }</p>	<p>Switch (ekspresi) { case nilai1: akibat1; break; case nilai2: akibat2; break; : : case nilain: akibatn; break; default:akibatlain; }</p>

FLOW CHART	ALGORITMA	PASCAL	C	C++
<pre> graph TD Start(()) --> Kondisi{kondisi ulang ?} Kondisi -- salah --> Kondisi Kondisi -- benar --> Daftaraksi[Daftar_aksi] Daftaraksi --> End(()) </pre>	<pre> [inisialisasi] WHILE <kondisi_ulang> DO daftar_aksi {ada aksi thd var kondisi} ENDWHILE </pre>	<pre> [inisialisasi] WHILE (kondisi_ulang) DO Begin daftar_aksi; {ada aksi thd var kondisi} End; </pre>	<pre> [inisialisasi] while (kondisi_ulang) { daftar_aksi; /*ada aksi thd var kondisi*/ } </pre>	<pre> [inisialisasi] while (kondisi_ulang) { daftar_aksi; /*ada aksi thd var kondisi*/ } </pre>
<pre> graph TD Start(()) --> Daftaraksi1[Daftar_aksi] Daftaraksi1 --> Kondisi1{kondisi ulang ?} Kondisi1 -- salah --> Daftaraksi1 Kondisi1 -- benar --> End1(()) </pre> <p><i>repeat-until</i></p> <pre> graph TD Start2(()) --> Daftaraksi2[Daftar_aksi] Daftaraksi2 --> Kondisi2{kondisi ulang ?} Kondisi2 -- salah --> Daftaraksi2 Kondisi2 -- benar --> End2(()) </pre> <p><i>do-while</i></p>	<pre> [inisialisasi] REPEAT Daftar_aksi {ada aksi thd var kondisi} UNTIL <kondisi_stop> </pre>	<pre> [inisialisasi] REPEAT Daftar_aksi; {ada aksi thd var kondisi} UNTIL (kondisi_stop); </pre>	<pre> [inisialisasi] do { daftar_aksi; /*ada aksi thd var kondisi*/ } while (kondisi_ulang); </pre>	<pre> [inisialisasi] do { daftar_aksi; /*ada aksi thd var kondisi*/ } while (kondisi_ulang); </pre>

FLOW CHART	ALGORITMA	PASCAL	C	C++
<pre> graph TD A{{Var ← awal, akhir, step}} -- terpenuhi --> B[daftar_aksi] A -- tidak --> B B --> NextRow </pre>	<p>Namavar <u>TRAVERSAL</u> [awal..akhir] daftar_aksi;</p> <p><i>atau</i></p> <p><u>FOR</u> var ← awal <u>TO/DOWNTO</u> akhir <u>STEP</u> counter <u>DO</u> daftar_aksi;</p>	<pre> { jika awal < akhir } FOR namavar := awal TO akhir DO Begin daftar_aksi; End; { jika akhir < awal } FOR namavar := awal DOWNTO akhir DO begin daftar_aksi End; </pre>	<pre> for(awal,akhir,step) { daftar_aksi; } /*parameter dari fungsi for berupa ungkapan kondisi*/ </pre> <p>catatan : daftar_aksi bisa tunggal bisa majemuk.</p>	<pre> for(awal,akhir,step) { daftar_aksi; } /*parameter dari fungsi for berupa ungkapan kondisi*/ </pre>

Contoh Translasi algoritma kedalam bahasa pemrograman.

Contoh 1 : buatlah algoritma tukar nilai.

ALGORITMA:

```

Algoritma TUKAR
{mempertukarkan nilai A dan B}
DEKLARASI
  A,B: integer
  Temp : integer
DESKRIPSI
  A ← 1
  B ← 2
  Temp ← A      (simpan nilai A kedalam temp)
  A ← B         (simpan nilai B kedalam A)
  B ← temp      (isikan nilai temp kedalam B)
  
```

Bahasa Pascal

```

Program TUKAR_AB;
  { program utama untuk mempertukarkan nilai A dan B. }
  (*DEKLARASI*)
var
  A,B:integer;
  temp : integer;
begin
  A:=1;
  B:=2;
  {cetak nilai X dan Y sebelum pertukaran}
  write('A=',A);
  write('A=',A);
  temp:=A;
  A:=B;
  B:=temp;
  {cetak nilai X dan Y setelah pertukaran}
  write('A=',A);
  write('A=',A);
end;

```

Bahasa C (dapat juga ditulis dengan bahasa C++)

```

/*Program TUKAR_XY untuk mempertukarkan nilai X dan Y.*/
#include<stdio.h>
#include<conio.h>

/*DEKLARASI*/
int A,B,temp;

main()
{
  A=1; B=2;
  /*cetak nilai A dan B sebelum pertukaran*/
  printf("A=%d \n",A);
  printf("B=%d \n",B);

  temp=A;
  A=B;
  B=temp;

  /*cetak nilai A dan B setelah pertukaran*/
  printf("A=%d \n",A);
  printf("B=%d \n",B);
}

```

Bahasa C++

```
#include <iostream.h>
void main ()
{
    int A, B, temp;

    A = 1;
    B = 2;

    /*cetak nilai A dan B sebelum pertukaran*/
    cout << "A = " << A <<endl;
    cout << "B = " << B <<endl;

    temp=A;
    A=B;
    B=temp;

    /*cetak nilai A dan B setelah pertukaran*/
    cout << "A = " << A <<endl;
    cout << "B = " << B <<endl;
}
```

3. PENUTUP

Latihan

1. Buatlah algoritma dan program bahasa C++ untuk menukar 3 buah nilai.
Misal X=10, Y=20 dan Z=30 menjadi X=20, Y=30, Z=10.
2. Tentukan keluaran/output dari algoritma dibawah ini:

```
Algoritma Cetak
Deklarasi
    A, B, X : integer
Deskripsi
    A ← 8
    B ← 6
    X ← A + B
    Output (X);
```

Dan ubahlah menjadi bahasa C++.