Type Bentukan

Notasi Algoritmik	Bahasa C

KAMUS
{ Definisi Type }
type namatype :
  < elemen1 : type1,
  elemen2 : type2,
  ... >
 	
{ Deklarasi Variable }
nmvar1 : namatype
nmvar2 : type1 {misal}
	/* KAMUS */
	/* Definisi Type */
	typedef struct {
	  type1 elemen1;
	  type2 elemen2;
	  ...
	} namatype;
	/* Deklarasi Variabel */
	namatype nmvar1;
	type1 nmvar2; /*misal*/

ALGORITMA **{ Akses Elemen }** nmvar2 ← nmvar1.elemen1 nmvar1.elemen2 ← <ekspresi>	**/* ALGORITMA */** **/* Akses Elemen */** nmvar2 = nmvar1.elemen1; nmvar1.elemen2 = <ekspresi>;

Type Bentukan (Contoh)

Notasi Algoritmik	Bahasa C

KAMUS
{ Definisi Type }
type Point :
  < X : integer,
  Y : integer >
 	
{ Deklarasi Variable }
P : Point
Bil : integer {misal}
	/* KAMUS */
	/* Definisi Type */
	typedef struct {
	  int X;
	  int Y;
	} Point;
	/* Deklarasi Variabel */
	Point P;
	int Bil; /* misal */

ALGORITMA **{ Akses Elemen }** Bil ← P.X P.Y ← 20	**/* ALGORITMA */** **/* Akses Elemen */** Bil = P.X; P.Y = 20;

Subprogram

Fungsi

Notasi Algoritmik:

```
function NAMAF (param1 : type1, param2 : type2, …) → type-hasil
{ Spesifikasi fungsi }

KAMUS LOKAL
{ Semua nama yang dipakai dalam algoritma dari fungsi }

ALGORITMA
{ Deretan fungsi algoritmik:
pemberian harga, input, output, analisis kasus, pengulangan }

{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-
hasil }
→ hasil
```

Fungsi

Bahasa C:

```
type-hasil NAMAF (type1 param1, type2 param2, ...)
/* Spesifikasi fungsi */
{
    /* KAMUS LOKAL */
    /* Semua nama yang dipakai dalam algoritma dari
       fungsi */

    /* ALGORITMA */
    /* Deretan fungsi algoritmik:
       pemberian harga, input, output, analisis kasus,
       pengulangan */

    /* Pengiriman harga di akhir fungsi, harus sesuai
       dengan type-hasil */
    return(hasil);
}
```

Pemanggilan Fungsi

Notasi Algoritmik:

```
Program POKOKPERSOALAN
{ Spesifikasi: Input, Proses, Output }
```

```
KAMUS
{semua nama yang dipakai dalam algoritma }
   {Prototype fungsi}
   function NAMAF ([list nama:type input]) → type-hasil
   {Spesifikasi fungsi}
```

```
ALGORITMA
{Deretan instruksi pemberian harga, input, output, analisa kasus,
pengulangan yang memakai fungsi}
{Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi}
   nama ← NAMAF ([list parameter aktual])
   output (NAMAF ([list parameter aktual]))

{Body/Realisasi Fungsi diletakkan setelah program utama}
```

Pemanggilan Fungsi

Bahasa C:

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */
/* Prototype Fungsi */
type-hasil NAMAF ([list <type nama> input]);
/* Spesifikasi Fungsi */
int main () {
    /* KAMUS */
    /* Semua nama yang dipakai dalam algoritma */
    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis
       kasus, pengulangan yang memakai fungsi */
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam
       ekspresi */
    nama = NAMAF ([list parameter aktual]);
    printf ("[format]", NAMAF ([list parameter aktual]));
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam
       ekspresi */

    return 0;
}
/* Body/Realisasi Fungsi diletakkan setalah program utama */
```

Prosedur

Notasi Algoritmik:

```
procedure NAMAP (input nama1 : type1,
                 input/output nama2 : type2,
                 output nama3 : type3)
{ Spesifikasi, Initial State, Final State }
```

```
KAMUS LOKAL
{ Semua nama yang dipakai dalam BADAN PROSEDUR }
```

```
ALGORITMA
{ BADAN PROSEDUR }
{ Deretan instruksi pemberian harga, input, output,
  analisis kasus, pengulangan atau prosedur }
```

Prosedur

Bahasa C:

```
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3)
/* Spesifikasi, Initial State, Final State */
{
    /* KAMUS LOKAL */
    /* Semua nama yang dipakai dalam BADAN PROSEDUR */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output,
       analisis kasus, pengulangan atau prosedur */

}
```

Pemanggilan Prosedur

Notasi Algoritmik:

```
Program POKOKPERSOALAN
{ Spesifikasi: Input, Proses, Output }
KAMUS
{semua nama yang dipakai dalam algoritma }
    {Prototype prosedur}
    procedure NAMAP (input nama1 : type1,
                     input/output nama2 : type2,
                     output nama3 : type3)
    {Spesifikasi prosedur, initial state, final state}
ALGORITMA
    {Deretan instruksi pemberian harga, input, output,
     analisis kasus, pengulangan}
    NAMAP(paramaktual1,paramaktual2,paramaktual3)
{Body/Realisasi Prosedur diletakkan setelah program utama}
```

Pemanggilan Prosedur

Bahasa C:

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */
/* Prototype prosedur */
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3);
/* Spesifikasi prosedur, initial state, final state */
int main () {
    /* KAMUS */
    /* semua nama yang dipakai dalam algoritma */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output,
       analisis kasus, pengulangan */
    NAMAP(paramaktual1,&paramaktual2,&paramaktual3);

    return 0;
}
/* Body/Realisasi Prosedur diletakkan setelah program utama
*/
```

Contoh

- Diktat “Contoh Program Kecil Bahasa C”
  - Prosedur & Fungsi

Type Data Koleksi : Array

Array Statik

Notasi Algoritmik	Bahasa C
{ Deklarasi Variabel } nm_array : array [0..nmax-1] of type-array	**/* Deklarasi Variabel */** type-array nm_array[nmax]; **Catatan: indeks array selalu dimulai dari 0**
{ Cara Mengacu Elemen } nm_array_{indeks}	**{ Cara Mengacu Elemen }** nm_array[indeks]
Contoh: TabInt : array [0..99] of integer $TabInt_i \leftarrow 1$ $X \leftarrow TabInt_{10}$	**Contoh:** int TabInt[100]; TabInt[i] = 1; X = TabInt[10];

Array Dinamik (Bahasa C)

- Array dinamik: alokasi memori untuk array (ukuran array) dapat diatur pada saat runtime
- Deklarasi: `type-array *nm_array;`
- Tahapan:
  1. Deklarasi (definisikan nama)
  2. Alokasi (tentukan ukuran/alokasi memori)
  3. Inisialisasi nilai
  4. Dealokasi (pengembalian memori)

Array Dinamik (Bahasa C) Contoh

1. Deklarasi (definisikan nama)

```
int *TabInt;
```

2. Alokasi (tentukan ukuran/alokasi memori)

```
TabInt = (int *) malloc (100 * sizeof(int));
/* TabInt dialokasi sebesar 100 */
```

3. Inisialisasi nilai

```
*(TabInt + i) = 9; /* pny. efek sama dgn: */
TabInt[i] = 9;
```

4. Dealokasi (pengembalian memori)

```
free(TabInt);
```

Array 2 Dimensi (Statik)

Notasi Algoritmik	Bahasa C
{Deklarasi Array} nm_array : array [0..nmax1-1] of array [0..nmax2-1] of type-array	**/*Deklarasi Array*/** type-array nm_array [nmax1][nmax2];
{Cara mengacu elemen} $nm\_array_{idx1,idx2}$	**/*Cara mengacu elemen*/** nm_array[idx1][idx2]
Contoh: Tab2D : array [0..2] of array [0..3] of integer $Tab2D_{i,j} \leftarrow 9$ $X \leftarrow TabInt_{2,3}$	**Contoh:** int Tab2D[3][4]; Tab2D[i][j] = 9; X = Tab2D[2][3];

Contoh

- Diktat “Contoh Program Kecil Bahasa C”
  - Array

Struktur Komposisi (2)

- Cara-2: **tag** berstruktur komposisi
  - Catatan: tidak terbentuk type baru

```
struct Mhs {
   char nama[20];
   int nim;
   int nilai;
}; /*Mhs adalah tag berupa struct*/
struct Mhs M1, M2; /*variabel berstruktur Mhs*/
```

- Cara-3: **type** berstruktur komposisi
  - Lihat kembali type bentukan

Struktur Komposisi (3)

- Cara-4: **Union**
  - Memungkinkan struktur komposisi mempunyai alternatif type elemen
  - Seluruh elemen union memakai lokasi memori yang sama

```
/* KAMUS */
union IsiSel {
   char *form;
   int nili;
   float nilf;
}; /*IsiSel dapat bertype string, int, float*/
/*variabel berstruktur union*/
union IsiSel sel1, sel2;
/* ALGORITMA */
strcpy(sel1.form,“isinya apa ya”);
printf(“Isi sel1: %s”,sel1);
```

Type Data Pointer (1)

- Type data Pointer berisi alamat mesin
  - Menunjuk kepada nama yang diacu sehingga informasi pada nama dapat diakses
  - Memungkinkan alokasi dinamik → memori baru **dialokasi** berdasarkan kontrol pemrogram, jika sudah tidak dibutuhkan, dapat **didealokasi** → harus hati-hati
  - Dalam bahasa C, nilai variabel bertype pointer dapat dimanipulasi sebagaimana halnya nilai numerik

Type Data Pointer (2)

- Pointer sederhana:

Format Deklarasi:

```
<type> *<nama>;
```

Contoh:

```
int *i;       /*pointer ke integer*/
float *f;     /*pointer ke real*/
char *cc;     /*pointer ke character*/
FILE *FT;     /*handle suatu file*/
int *(T)[10]; /*pointer ke array dg 10 elemen integer*/
int *T[10];   /*pointer ke array dg 10 elemen bertype pointer
                ke integer*/
void *p;      /*pointer ke objek yg tdk diketahui typenya*/
```

Type Data Pointer (3)

- Pointer sederhana:

```
/*Kamus*/
int *iptr;

/*Algoritma*/

/*Alokasi memori untuk 1 buah integer*/
iptr = (int *) malloc (sizeof(int));
/*Inisialisasi nilai yang ditunjuk iptr*/
*iptr = 10;
printf("Nilai yang ditunjuk iptr : %d",*iptr);
free(iptr); /*Dealokasi iptr*/

/*Alokasi kembali dengan ukuran 4 integer*/
/*Menjadi array of integer yang dialokasi dinamik*/
iptr = (int *) malloc (4 * sizeof(int));
/*Mengisi elemen ke-0 dari iptr*/
*(iptr+0) = 999;
printf("Nilai yang ditunjuk iptr : %d", *(iptr+0));
free(iptr); /*Dealokasi iptr*/
```