**Function**

Blok pada function hampir sama dengan blok pada procedure, hanya pada function harus dideklarasikan dengan tipe dari function tersebut yang merupakan tipe hasil dari function itu sendiri. Sehingga dikatakan function dapat mengembalikan nilai.

Sintaks :

FUNCTION identifier(daftar parameter) : type;

**I. Parameter Nilai dalam function**

Parameter dalam function dapat dikirimkan secara nilai atau secara acuan.

Penulisan judul function yang menggunakan parameter secara Nilai adalah : Function besar(a,b : real) : real;

Contoh :

program penggunaan\_parameter\_nilai;

uses crt;

function besar(a,b :real) : real;

begin

if a>b then besar:=a else besar:=b;

end;

{modul utama}

var

nil1,nil2 : real;

begin

write('bilangan 1=');readln(nil1); write('bilangan 2=');readln(nil2); writeln('bilangan terbesar =',besar(nil1,nil2):6:2); readln;

end.

**II. Function dengan parameter acuan**

Penulisan judul function dengan menggunakan parameter secara acuan adalah sama

dengan procedure yaitu ditambah Var pada deklarasi parameter. Dengan demikian nilai parameter acuan ini dapat digunakan sebagai hasil balik.

Sintaks :

FUNCTION jumlah(var a,b : integer) : integer; Contoh :

program pengiriman\_parameter\_secara\_acuan; function kali(var bil1,bil2,jumlah : integer) : integer; begin

kali:=bil1\*bil2; jumlah:=bil1+bil2; end;

var

x,y,z : integer;

begin

write('bilangan 1=');readln(x);

write('bilangan 2=');readln(y); writeln(x:3,'\*',y:3,' = ',kali(x,y,z):5); writeln(x:3,'+',y:3,' = ',z);

readln;

end.

**III. Function tanpa parameter**

Suatu function tanpa parameter berarti nilai balik yang akan dihasilkan merupakan

nilai yang sudah pasti. Jika pada function dengan parameter, parameternya digunakan untuk input pada function dan function akan memberikan hasil balik sesuai dengan parameter yang diberikan sehingga bisa diatur dari program pemanggil. Sedang pada function tanpa parameter hasil dari function tidak dapat diatur. Sehingga function tanpa parameter jarang digunakan.

Contoh :

function tiga : integer;

begin tiga:=3; end; begin

writeln(tiga);

end;

Jadi hasil :

3

Function type string untuk membuat garis, ini juga merupakan contoh function tanpa parameter.

uses crt;

function garis : string;

begin

garis:='----------';

end;

{modul utama} begin writeln(garis); writeln('pascal'); writeln(garis);

readln;

end.

Contoh :

program pangkat\_dgn\_function;

uses crt;

function pangkat(bil :real; pang: integer) : real;

var

hasil : real;

i : integer;

begin

hasil := 1;

for i:= 1 to pang do

begin

hasil:= hasil\*bil; end; pangkat:=hasil; end;

var

hitung,bil : real; pang : integer; begin

write('bilangan =');readln(bil); write('pangkat=');readln(pang); hitung:= 2\*pangkat(2,3);

writeln(bil:5:2,' pangkat',pang:5,' = ',pangkat(bil,pang):6:2);

writeln('2 \* (2 pangkat 3) =',hitung:6:2);

readln;

end.

**IV. Rekursi pada function**

Rekursi adalah dimana suatu function memanggil dirinya sendiri. Proses dapat dilihat

pada contoh berikut. Dimana fungsi faktor dipanggil oleh dirinya sendiri. Contoh :

program function\_memanggil\_funnction\_yg\_lain;

uses crt;

function faktor(bilangan : integer) : real;

begin

if bilangan=0 then faktor:=1

else

faktor:=faktor(bilangan-1)\*bilangan;

end;

var

n : integer;

begin

write('berapa faktorial =');readln(n);

writeln(N:5,' faktorial =',faktor(n):9:0);

readln;

end.

**File Teks**

Pascal mempunyai dua macam file. File teks dan file binary. Bagian ini membicarakan file teks. Fiel teks tidak mempunyai besar yang eteap. Untuk menandai akhir suatu file, komputer menempatkan karakter khusus end-of-file (<eof>) setelah karakter yang paling akhir. Untuk menandai akhir suatu baris, komputer menampatkan karakter khusus end- of-line pada akhir baris.

Dalam perogram yang interaktif biasanya kita menuliskan nilai sentinel untuk menandai akhir suatu baris atau file. Sebagai contoh, pecahan program semacam ini digunakan membaca suatu nama (kumpulan karakter) dengan nilai sentinel titik.

Read(nama); While(nama<>’.’) Do

Read(nama);

Dalam file teks, untuk mengetes apakah baris sudah berganti, kita bias menggunakan fungsi eoln. Berikut ini pecahan program diatas yang ditulis deng fungsi eoln.

While not EoLn Do

Read(nama);

Untuk mengetes apakah akhir suatu file, kita bias menggunakan fungsi eof seperti berikut ini.

While not Eof(InfileData) Do

Begin

End;

While Not EoL Do

Read(Nama); ReadLn(Gaji);

InfileData diatas merupakan nama file yang bertipe teks. Program diatas membaca variable-variabel Nama dan gaji dalam File Infile. Apabila akhir baris dtidak ditemui, nilai Eoln berarti false yang berarti program membaca variable nama. Setelah akhir baris ditemui, nilai EoLn menjadi true dan program membaca variable berikutnya yaitu Gaji. Setelah akhir file ditemui, nilai Eof menjadi true dan program keluar dari loop.

*Membuat File Teks*

File teks bias dibuat melalui beberapa cara. Kalau kita ad DOS, maka dengan cara yang

termudah adalah dengan menggunakan DOS tersebut. Perintah yang digunakan adalah sebagai berikut :

A:\> Edit <namafile>

Dos editor kemudian muncul, dan kita bias mengetik angka-angka atau huruf yang akan disimpan sebagai file teks.

Apabila kita mempunyai pascal Editor, file teks dapat dibuat dengan menggunakan editor pascal. Bentuk editor tersebut mirip dengan DOS Editor. Untuk mengaktifkan menu, kita bisa menekan tombol F10. File teks bisa disimpan dengan menu Save atau Save As.

*Deklarasi File Teks*

Seperti variable-variabel lain dalam pascal, file teks juga harus dideklarasikan terlebih

dahulu sebelum digunakan. Berikut ini adalah deklarasi file teks bernama InfileData.

Program ProsesFile (InfileData,OutFIle)

Var

InfileData, Outfile : text;

InfileData dan OutFile dideklarasikan sebagi file teks. Dalam judul program keduanya harus dituliskan. Apabila program juga akan menggunakan keyboard (sebagai input) dan monitor (sebagai Output), maka judul program dituliskan sebagai berikut :

Program ProsesFile (InfileData,Input,OutFile,Output);

Var

InfileData,Outfile : Text;

*Pernyataan Reset*

Pernyataan Reset digunakan untuk menyiapkan suatu file teks untuk dibaca oleh

program. File teks siap untuk diproses dengan pernyataan berikut : Reset(InfileData);

Dengan pernyataan Reset, pointer digeser ke permulaan file Teks. Karakter pertama dalam suatu fiel akan diproses sesudah pernyataan Reset. Sebelum data dibaca, operasi Reset harus dilakukan , apabila tidak program akan gagal menjalankan tugasnya (error akan muncul).

*Pernyataan Rewrite*

Untuk menyiapkan Output (file teks yang akan menampung Output program kita), kita

harus menuliskan pernyataan seperti berikut ini : Rewrite(OutFile);

Pernyataan diatas menyiapkan file OutFile untuk menampung hasil pemrosesan. Kija tidak ada file OutFile sebelumnya, OutFile akan diciptakan. Apabila

sebelumnya ada file OutFile, pointer akan ditempatkan pada awal File dan semua isi

OutFile yang lama akan terhapus oleh hasil pemrosesan yang terbaru.

*Pernyataan Close*

Pernyataan Close dipakai untuk menutup file-file yang dibuka dan dipakai dalam suatu

program. Program yang menggunakan operasi Output-Input (O/I) biasanya lebih lambat, karena program tersebut dengan menggunakan jasa DOS berhubungan dengan aspek Fisik dari disket.

Pascal menyediakan memori untuk menampung atau menuliskan data ke file. Ketika

program menuliskan data atau membaca data, program membaca atau menuliskan data ke file buffer., bukannya langsung ke file eksternal secara langsung. Pascal secara periodic memindahkan data tersebut dari file buffer ke file eksternal. Apabila kita tidak menuliskan pernyataan close, proses pemindahan data tidak akan sempurna, dengan

akibat ada data yang hilang. Tidak disimpan dalam file. Dengan cara semacam itu, program yang melibatkan operasi I/O akan diproses lebih cepat daripada apabila program langsung memanggil file eksternal.

Penulisan pernyataan Close adalah sebagai berikut : Close(InfileData);

Contoh program :

Program hasilPrinter (InfileData,Output): Var

Begin

InfileData : text;

I : Integer;

AssiGn(InfielData,’PRN’); Rewrite(InfileData) WriteLn(InfileData,’Bilangan dari 10 ke 10); WriteLn;

For I := 1 to 10 Do WriteLn(InfileData,I); WriteLn(InfileData,chr(12));

Close(InfileData); End.

Hasil dari program diatas :

1

2

3

4

5

6

7

8

9

10

**File Binary**

Sebagai alternatif penulisan file teks, pascal memungkinkan kita menuliskan file dengan menggunakan kata(konstruktor) file seperti berikut ini :

Type

Var

DeretAngka = File of Integer;

IntData : DeretAngka; Angka : Integer;

File inData disebut juga sebagai File Binary. File Binary adalah file dimana representasi internal dari tiap-tiap komponen secara langsung. Misalkan nilai variable angka adalah

244, pernyataan : Write(InData,Angka);

Mengkopi representasi binary internal variable angka dari memori ke file InData. Misalkan file OutData bertipe teks, pernyataan berikut ini :

Write(OutData, angka:4);

Akan menuliskan niali variable angka ke file OutData dengan empat Bytes. Komputer pertama harus mengubah representasi binary dari memori ke string ‘244’ dan kemudian menuliskan kode bnary unutk karakter blank(‘ ‘) ,2,4 dan 4 ke OutFile.

Sebaliknya apabila angka ‘244’ mau ditampilkan dilayar monitor, komputer akan

mengkopi representasi binary dair blank (‘ ‘),2,4 dan 4 kemudian menuliskan ke teks string ‘244’ yang kemudian ditampilkan dilayar monitor. Proses semacam ini memakan waktu lebih lama dibandingkan kalau langsung mengkopi representasi binary internal ke disk.

Bentuk Umum dari file binary(sering juga disebut typed filed), adalah sebagai berikut : Var

InFile : File of <tipe>;

Dimana tipe bis merupakan tipe dasar file seperi Integer, Char, bahkan suatu record, dan bias juga suatu string.

Berikut deklarasi file binary : Type

String10 = string[10];

RecMhs = record Nama : string[10]; IP : Real;

End;

Var

InChar : File of char; InMhs : File of recMhs; InItgr : File of Integer; InStrng : File of String[10];

Contoh program : Program Bin01(input,OutFile); Var

Begin

OutFile : File of Integer; Angka,Jumlah : Integer;

AssiGn(OutFile,’a:\latihan\outline.txt’); Rewrite(OutFile);

WriteLn;

WriteLn(‘Berapa angka yang akan dimasukkan : ‘);ReadLn(Jumlah); For Angka := 1 to Jumlah Do

Write(OutFile,Angka); Reset(OutFIle);

For Angka := 1 to Jumlah Do

Begin

End; ReadLn; End.

Read(OutFile, Angka); Write(Output,Angka);

**Pointer**

Pada bab terdahulu (dalam hal Array), masih menyisakan beberapa kelemahan, antara lain adalah penggunaan space (ruang) yang tetap. Kita harus tahu jumlah maksimum yang akan kita alokasikan untuk record tersebut, baru kita menentukan space yang kita butuhkan.

Pascal menyediakan fasilitas untuk mengatasi masalh static data structure seperti digambarkan diatas dengan menggunakan dynamic data structure. Berbeda dengan static data structure, dynamic data structure, struktur data bias berkembang atau berkurang sesuai dengan kebutuhan. Dengan dynamic data structure kita tidak perlu menentukan jumlah record maksimum dan tidak perlu membuang space yang tidak terpakai apabila jumlah record lebih kecil dibandingkan kapasitas yang disediakan.

Sebagai gambaran array record dan linked record, adalah sebagai berikut :

Misalkan kita mempunyai data yang tersusun berdasarkan urutan alphabet adalah sebagai berikut :

Pertama (6) Kedua (7)

|  |  |  |  |
| --- | --- | --- | --- |
| Nama | (1) Djoko | 5.000 | 5 |
| Nama | (2) Koen | 7.000 | -1 |
| Nama | (3) Laudiah | 3.000 | 4 |
| Nama | (4) Boby | 4.000 | 1 |
| Nama | (5) Inonk | 6.000 | 2 |
| Nama | (6) Inem | 8.000 | 3 |
| Nama | (7) …. | …. | …. |
| Nama | (8) …. | …. | …. |
| . |  |  |  |
| . |  |  |  |
| . |  |  |  |
| . |  |  |  |
| Nama | (50) .. | … | 50 |

Data diatas bisa dibaca sebagia berikut :

Pertama menunjuk ke posisi (6) yang merupakan urutan pertama. Kemudian Inem menunjuk kearah berkutnya (Laudiah) yang terletak pada baris ketiga. Laudiah menunjuk keposisi berikutnya daalam urutan yaitu Boby yang berada pada urutan ke-4. Posisi terakhir, yaitu Koen menunjuk pada posisi -1 yang berarti daftar sudah berakhir.

Data diatas dapat dideklarasikan sebagai berikut : Type

Pembeli : Record

Nama : Packed ArraY [1..10] of char; Uang, Link : IntegeR;

Var

End;

Daftar : ArraY [1..50] of Pembeli; Pertama, kosong : Integer;

Apabila kita ingin menambah data baru, maka kita bis menyelipkan data tersebut diantara daftar yang ada sekarang. Misalkan kita ingin memasukkan nama baru mahmud kedalam data diatas.

Data Tipe Pointer

Kita akan membicarakan data dengan tipe pointer. Data bisa kita simpan pada variable semacam ini. Sebagai contoh, dibawah ini penulisan deklarasi variable dengan tipe pointer.

Type

Var

JumlahReal = ^Real;

Jumlah : JumlahReal;

Jumlah dideklarasikan sebagai variable pointer dengan tipe JumlahReal. Kita bisa menyimpan alamat memori variable tipe real pada jumlah. ^Real dibaca sebagi petunjuk (pointer) ke Real.

Dengan Pointer, record daftar pembeli seperti dimuka bisa ditulis sebagi berikut :

Type

Pembeli = Record

Nama : Packed ArraY [1..10] of char; Uang : Integer;

Link : ^Pembeli;

Var

End;

Daftar, Daftar1,daftar2 : ^Pembeli;

Perlu diingat bahwa variable pointer ini hanya mengandung nilai yang menunjuk pada alamat memori yang menunjuk pada record dengan tipe pembeli.

Pernyataan New digunakan untuk mengaktifkan daftar.

New(daftar);

Pernyataan tersebut mengalokasikan pada daftar. Alamat memori kemudian disimpan pada variable daftar. Gambar berikut menjelaskan proses tersebut secara grafis.

Daftar Daftar

Sebelum New ?

Sesudah New 3142 → ….. …… …… Daftar^Nama

Daftar^Uang

Daftar^link

Dengan pernyataan New, sel memory dengan alamat 3142 dialokasikan ke daftat. Penugasan semacam dibawah ini akan mengisi nilai untuk daftar^Nama dan Daftar^Uang. Daftar^link belum diisi nilainya.

Daftar^nama : Sari;

Daftar^uang : 10000;

Kita juga bisa mengkopikan isi daftar ke daftar1 yang mempunyai tipe data yang sama.

Daftar1 := Daftar;

Daftar dengan daftar1 mempunyai alamat memory yang sama, dan dengan demikian menunjuk pada isi yang sama. Gambar berikut ini menjelaskan proses tersebut secara grafis.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nama | Uang | Link |
| Daftar | Sari | 10000 | ? |

Sesudah daftar dikopikan kedalam daftar1, adalah sebagai berikut :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nama | Uang | Link |
| Daftar  Daftar1 | Sari | 10000 | ? |

**Menghubungkan variable pointer**

Misalkan kita mempunyai 3 variabel pointer (sering juga disebut Node), seperti yang ada

|  |  |  |  |
| --- | --- | --- | --- |
| dibawah ini : | Nama | Uang | Link |
| Daftar1 | Sari | 10000 | ? |
| (2311) |  |  |  |
| Daftar2 | Ahmad | 11000 | ? |
| (3110) |  |  |  |
| Daftar3 | Djoko | 6000 | ? |
| (1100) |  |  |  |

Angka dalam kurung adalah menunjuk alamat memori variable diatas. Variabel diatas belum dihubunkan satu dengan yang lain. VAriabel diatas dihubungkan dengan pernyataan sebagai berikut :

1. Daftar1^link := daftar2;

2. Daftar2^link := daftar3;

3. Daftar3^link := Nil;

sesudah penugasan pernyataan-pernyataan diatas, variable diatas bisa digambarkan sebagai berikut :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nama | Uang | Link |
| daftar1 | Sari | 10000 | \* |
| (2311) |  |  |  |

daftar2 Ahmad 11000 \* (3110)

daftar3 Djoko 6000 \* (1100)

Nil menunjukkan bahwa daftar3 merupakan akhir dari daftar pembeli diatas. Dengan menggunakan alamat-alamat memori, proses diatas akan nampak jelas :

|  |  |  |  |
| --- | --- | --- | --- |
| Pertama (3110) |  | | |
| Daftar | Nama  Sari | Uang  10000 | Link  Nil |
| (2311) |  |  |  |
| daftar  (3110) | ahmad | 11000 | 1100 |
| daftar  (1100) | Djoko | 6000 | 2311 |

variable pertama mempunyai alamat memori yang menunjuk pada alamat 3110.

Pertama^Nama = ahmad

Pertama^Uang = 11000

Pertama^Link = 1100

Pernyataan daftar := pertama^link (atau dalam hal ini daftar := 1100), membuat program bergerak turun ke memory sel 1100

Daftar^Nama = Djoko

Daftar^uang = 6000

Daftar^link = 2311

Pernyataan daftar := daftar^link (daftar := 2311) membuar program bergerak turun ke memory sel 2311

Daftar^Nama = Sari

Daftar^Uang = 10000

Daftar^Link = Nil

Nil menunjukkan bahwa nilai ini merupakan tanda akhir list (daftar) Contoh :

Berikut adalah program menciptakan linked data, mencari data dan menghilangkan data

Program Link1 (Input,Output); Uses crt;

Type

Pointer = ^Cell; Cell = Record

Value : Integer; Next : pointer;

Var

End;

Last, belakang, Q, P : pointer; Angka, Nomor : Integer; Jawab : Char; Found : Boolean;

Procedure printlist; Begin

Last := Belakang;

While last <> nil Do

Begin

End;

End;

WriteLn(last^.Value); Last := Last^.next;

Procedure Look; Begin

WriteLn(‘Masukkan Angka yang akan dicari : ‘);ReadLn(angka); Last := Belakang;

Found := false;

While (last <> nil) and (found <> true) Do

Begin

If last^.value = angka then

Begin

Found := True; WriteLn(‘Angka Ditemukan’); WriteLn(Last^.value);

Found := True;

Else

End;

End;

End;

Last := Last^.next;

Procedure Delete; Begin

WriteLn(‘Angka lain dihapus ???’);ReadLn(angka); Last := Belakang;

Found := False;

While (last <> Nil) and (found <> true) Do

Begin

Last := Last^.next;

If Q^.value = angka then

Begin

Else

End;

Dispose(Q); Found := True;

If last ^.value = angka then

Begin

End;

End;

End;

WriteLn(last^.value, ‘ ‘ , ditemukan dan dihapus’)l

Q^.next := last^.next; Dispose(last);

Found := true;

Begin

Belakang := Nil; Nomor := 1;

writeLn(‘nomor 1 : ‘, nomor); Jawab := ‘y’;

While (jawab <> ‘t’) Do

Begin

Writeln(‘Masukkan Angka : ‘);ReadLn(angka); New(last);

Last^.Value := angka; Last^.next := belakang; Belakang := last;

Nomor := nomor + 1;

printList; Look; Delete; printList;

End;

WriteLn(‘Nomor : ‘, nomor); WriteLn(‘Terus (y/t ???? ‘); Readln(jawab);

ReadLn(jawab); End.

Hasil dari program diatas, adalah : Masukkan angka : 3

Terus(y/t) ???? y

Masukkan angka : 10

Terus(y/t) ???? y

Masukkan angka : 7

Terus(y/t) ???? t

7

10

3

Masukkan angka yang akan dicari : 10

Angka ditemukan

Angka ingin dihapus : 7

7 ditemukan dan dihapus

10

3