ALGORITMOS Y PROGRAMACION

QUINTO SEMESTRE

GRUPO 510

ALGORITMO

unidad I: Algoritmo y Programas

1.1 consideraciones acerca de la computabilidad

1.2 Concepto de algoritmo

1.3 datos, tipos de datos y operaciones

1.4 constantes y variables

1.5 funciones internas

1.6 floorchart S.O.B

unidad II: Resolución de problemas con comp. y herramientas de programación

2.1 Resolución de problemas

2.2 Análisis de problemas

2.3 Diseño de algoritmo

2.4 Representación grafica de algoritmos: diagrama de flujo (floor chart)

unidad III: Estructura gral. de un problema

3.1 Concepto de programa

3.2 Partes constitutivas de un programa

3.3 instrucciones y tipos de análisis de problemas modelos

3.4 escritura de algoritmos − Programas

Unidad IV: Introducción a la programación estructurada

4.1 técnicas de prog.

4.2 Prog. Modular

4.3 Prog estructurada

4.4 estructura secuencial

4.5 estructuras selectivas

4.6 estructuras repetitivas

4.7 estructuras repetitivas anidadas

Unidad V: Programas Procedimientos y funciones

5.1 Introducción a los subalgoritmo

5.2 funciones

5.3 procedimientos

5.4 funciones y procedimientos como parámetros

5.5 Recursividad

Programación

Unidad I

- Programa de computadora
- Algoritmos y estructuras de datos
- Programas simples en pascal

Unidad II

2.1 La sintaxis de pascal

2.2 estructuras de un programa en pascal

2.3 Declaraciones

2.4 Los símbolos en pascal

2.4.1 Palabras reservadas

2.4.2 Símbolos reservados

2.4.3 identificadores

2.4.4 comentarios

2.4.5 encabezados

Unidad III

3.1 Tipos de datos

```
Program prog1;
{programa que calcula la suma de dos números}
Uses crt;
Var
N1,N2:interger;
T:real;
Begin
Clrscr;
Writeln(`capruta el primer numero:');
Readln(N1)
Writeln(`dame el segundo numero:');
Readln(N2)
T:=N1+N2;
Writeln(`la suma de dos números es:',T:7:0);
Readln;
End.
```

DIAGRAMA1

```
Program prog2;
{programa que captura un nombre}
Uses crt;
Var
N:string[6]
A:string[5]
Begin
Clrscr;
```

```
Writeln(`captura el nombre:');

Readln(N);

Writeln(`Captura el apellido:');

Readln(A);

Writeln(`tu nombre es:',N,A);

Readln;

End
```

DIAGRAMA 2

```
Program prog3;

{programa que convierte pesos a dólares}

Uses crt;

Const

TC=9.8;

Var

P,D:real;

Begin

Clrscr;

Writeln(`conversión de peso a dólar');

Writeln;

Writeln(`captura la cantidad en pesos:');

Readln(P)

D:=P/TC;

Writeln(`la cantidad en dólar es :',D:6:0);

Readln;

End.
```

DIAGRAMA 3

```
Program Prog4
{programa que calcula el promedio de dos calificaciones}
Uses crt;
Var
C1,c2,p:real;
Begin
Clrscr;
Writeln(`dame tu primera calificación:');
Readln(c1);
Writeln(`dame tu segunda calificación:');
Readln(c2);
P:=(c1+c2)/2;
Writeln(`tu promedio es :',p:7:0);
Readln;
Writeln(`trata de mejorar');
Readln;
End.
```

DIAGRAMA 4

```
Program prog5
{programa de datos personales}
Uses crt;
Var
N,ap,am,d,p:string[50];
E:integer;
Begin
Clrscr;
```

```
Writeln (`dime tu nombre:');

Readln(n);

Writeln (`cual es tu apellido paterno:');

Readln(ap);

Writeln(`cual es tu apellido materno:');

Readln(am);

Writeln(`cual es tu domicilio:');

Readln(d);

Writeln(`cual es tu ocupación:');

Readln(p);

Writeln(`que año naciste:');

Readln(a);

Writeln(`tu eres:',n:,' `,ap:,' `,am);

Writeln(`vives en:',d);

Writeln(`eres:',p);

Writeln(`naciste en el año de:',a);

Readln;

E:=2002−a;

Writeln(`tienes la edad de:',e);

Readln;

End.
```

DIAGRAMA 5

```
Program prog6

{PROGRAMA que captura el sueldo de un empleado}

Uses crt;

Var
```

```
Nde,rfc,d,p,pr:string[50];

Sh,dt,sd,ss,sq:real;

Ht:integer;

Begin

Clrscr;

Writeln(`nombre del empleado:');

Readln(nde);

Writeln(`cual es su rfc:');

Readln(rfc);

Writeln(`cual es su domicilio:');

Readln(d);

Writeln(`cual es su puesto:');

Readln(p);

Writeln(cual es su profesión:');

Readln(pr);

Writeln(`cual es su sueldo por hora:');

Readln(sh);

Writeln(`cuantas horas trabaja al día:');

Readln(ht);

Writeln(`cuantos días trabaja a la semana:');

Readln(dt);

Writeln(`el nombre del empleadoa es:',nde);

Readln;

Writeln(`su rfc es :',rfc);

Readln;

Writeln(su domicilio es :',d);
```

```
Readln;

Sd:=sh*ht;

Writeln(`su sueldo diario es de:',sd:2:0);

Readln;

Ss:=sd*dt;

Writeln(`su sueldo semanal es de:',ss:2:0);

Readln;

Sq:=ss*2;

Writeln(`su sueldo quincenal es de:',sq:2:0);

Readln;

Writeln(`***siga trabajando casi es millonario***');

Readln;

End.
```

DIAGRAMA 6

```
Program prog7

{datos personales};

Uses crt;

Var

N,d,t:string[20]

Begin

Clrscr;

Writeln(`cual es tu nombre:');

Readln(n);

Writeln(`cual es tu domicilio:');

Readln(d);

Writeln(`cual es tu numero telefónico:');
```

```
Readln(t);

Writeln('te llamas:',n);

Readln;

Writeln(vives en:',D);

Readln;

Writeln(`tu numero de teléfono es:',t);

Readln;

End.
```

DIAGRAMA 7

```
Program prog8

{obtener el producto de dos números}

Uses crt;

Var

N1,n2,p:real;

Begin

Clrscr;

Writeln(`cual es el primer numero:');

Readln(n1);

Writeln(`cual es el segundo numero:');

Readln(n2);

P:=n1*n2;

Writeln(`el producto es:',p);

Readln;

End.
```

DIAGRAMA 8

```
Program prog9
```

```
{Conversión de metros a centímetros};
Uses crt;
Var
R,m:real;
Const
Cm=100;
Begin
Clrscr;
Writeln(`cuantos metros:');
Readln(m);
R:=m*cm;
Writeln(`son:',r:4:2);
Readln;
End.
```

DIAGRAMA 9

```
Program prog10
{conversión de millas a kilómetros};
Uses crt;
Var
Mi,r:real;
Const
Km=1.609;
Begin
Clrscr;
Writeln(`cuantas millas son:');
Readln(mi);
```

```
R:=km*mi;

Writeln(`son en km:',r:5:3);

Readln;

End.
```

DIAGRAMA 10

```
Program prog11;

{programa que obtiene la suma de dos exámenes}

Uses crt;

Var

Ex1,ex2,s:real;

Begin

Clrscr;

Writeln('calificación de el examen 1');

Readln(ex1);

Writeln('calificación de el examen 2');

Readln(ex2);

S:=ex1+ex2;

Writeln('la suma de los exámenes es:',s:3:1);

Readln;

If s>=80 then

Begin

Writeln('el alumno esta aceptado');

Readln;

End

Else

Begin
```

```
Writeln('el alumno esta rechazado');

Readln;

End;

End.
```

DIAGRAMA 11

```
Program prog12;

{programa que obtiene la comisión de una persona que vende libros}

Uses crt;

Const

C=20;

D=10;

Var

A,b,x,z:real;

Y:integer;

Begin

Clrscr;

Writeln('el costo de los libros es');

Readln(x);Writeln('los libros vendidos son ');

Readln(y);

Z:=x*y;

Writeln('la ganancia obtenida por los libros vendidos es:', '$',z:3:2);

Readln;

If z>= 1500 then

Begin

Clrscr;

A:=(z*c)/100;
```

```
Writeln('su comisión es de el 20%');

Writeln('su ganancia es');

Writeln('$',a:3:2);

Readln;

End

Else

Begin

Clrscr;

B:=(z*d)/100;

Writeln('su comisión es de el 10%');

Writeln('su ganancia es');

Writeln('$',b:3:2);

Readln;

End;

End.
```

Diagrama 12

```
Program prog13;

{programa que elebora boleta segun caracteristicas del alumno}

Uses crt;

Var

N,s:string;

A,p:real;

Begin

Clrscr;

Writeln('Nombre de el alumno:');

Readln(n);
```

```
Writeln('Semestre:');

Readln(s);

Writeln('Asistencia Total:');

Readln(a);

Writeln('promedio final:');

Readln(p);

If (a>=80) and (p>=7) then

Begin

Clrscr;

Writeln('******SE ELABORA BOLETA******');

Writeln('Nombre:',n);

Writeln('Semestre:',S);

Writeln('Asistencia:',A:3:0,'%');

Writeln('promedio final:',p:3:1);

Readln;

End

Else

Begin

Clrscr;

Writeln('***NO SE ELABORA BOLETA***');

Readln;

End

End.
```

DIAGRAMA 13

```
Program prog14;

{PROGRAMA QUE DESPLIEGA LA POBLACION DE UN ALUMNO SEGUN SUS
CARACTERISTICAS}
```

```
Uses crt;

Var

N:string;

E,p:real;

Begin

Clrscr;

Writeln('Nombre alumno:');

Readln(N);

Writeln('Edad:');

Readln(E);

Writeln('promedio final:');

Readln(p);

If (p>6) and (p<8) and (e>6) and (e<8) then

Begin

CLRSCR;

Writeln('***** POBLACION "A" *********');

Writeln('nombre:',N);

Writeln('edad:',e:3:0);

Writeln('Promedio:',p:3:1);

Readln;

End

Else

If (P=8) and (e=8) then

Begin

CLRSCR;

Writeln('****** POBLACION "B" ********');
```

```
Writeln('Alumno:',N);
Writeln('Edad:',e:3:0);
Writeln('promedio final:',p:3:1);
Readln;
End
Else
Begin
CLRSCR;
Writeln('********** POBLACION "C" **********');
Writeln('Nombre:',N);
Writeln('edad:',e:3:0);
Writeln('Promedio:',p:3:1);
Readln;
End;
End.
```

DIAGRAMA 14

```
Program prog15
{determina si un numero es positivo, negativo o igual a cero};
Uses crt;
Var
N:real;
Begin
Clrscr;
Writeln('Dame un numero:');
Readln(n);
If n>0 then
```

```
Begin
Writeln('El numero es positivo:');
Readln;
End
Else
If n<0 then
Begin
Writeln('el numero es negativo:');
Readln;
End
Else
If n=0 then
Writeln('el numero es cero:');
Readln;
End.
```

DIAGRAMA 15

```
Program prog16
{programa que captura 3 numeros y determina si es mayor menor o igual};
Uses crt;
Var
N1,n2,n3:real;
Begin
Clrscr;
Writeln('Dame el primer numero:');
Readln(n1);
Writeln('dame el segundo numero:');
```

```
Readln(n2);
Writeln('dame el ultimo numero:');
Readln(n3);
If (n1>n2) and (n1>n3) then
Begin
Writeln('el mayor es el primero:',n1:4:2);
Readln;
End
Else
If (n2>n1) and (n2>n3) then
Begin
Writeln('el mayor es el segundo:',n2:4:2);
Readln;
End
Else
If (n3>n1) and (n3>n2) then
Begin
Writeln('el mayor es el tercero:',n3:4:2);
Readln;
End
Else
If (n1=n2) and (n2=n3) then
Begin
Writeln('los numeros son iguales:');
Readln;
End
```

```
Else
If (n1=n2) and (n1>n3) then
Begin
Writeln('el primero y el segundo son iguales y mayores que el tercero');
Readln;
End
Else
If (n1=n3) and (n1>n2) then
Begin
Writeln('el primero y el tercero son iguales y mayores que el segundo numero');
Readln;
End
Else
If (n2=n3) and (n2>n1) then
Begin
Writeln('el segundo y el tercero son iguales y mayores que el primero');
Readln;
End
End.
```

DIAGRAMA 16

```
Program prog17
{programa que suma,resta,multiplica o divide dos numeros};
Uses crt;
Var
N1,n2,s,r,m,d:real;
Op:integer;
```

```
Begin
Clrscr;
Writeln(' * OPCIONES *');
Writeln(' 1 − SUMA');
Writeln(' 2 − RESTA');
Writeln(' 3 − MULTIPLICACION');
Writeln(' 4 − DIVISION');
Writeln(' 5 − SALIDA');
Writeln('SELECCIONE UNA OPCION');
READLN(OP);
CASE OP OF
1:BEGIN;
Writeln(' * SUMA *');
WRITELN('CAPTURA EL PRIMER NUMERO');
READLN(N1);
Writeln('CAPTURA EL SEGUNDO NUMERO');
READLN(N2);
S:=N1+N2;
Writeln('EL RESULTADO ES: ',S:6:2);
READLN;
END;
2:BEGIN;
Writeln(' * RESTA *');
WRITELN('CAPTURA EL PRIMER NUMERO');
READLN(N1);
Writeln('CAPTURA EL SEGUNDO NUMERO');
```

```
READLN(N2);
R:=N1−N2;
Writeln('EL RESULTADO ES: ',R:6:2);
READLN;
END;
3:BEGIN;
Writeln(' * MULTIPLICACION *');
WRITELN('CAPTURA EL PRIMER NUMERO');
READLN(N1);
Writeln('CAPTURA EL SEGUNDO NUMERO');
READLN(N2);
M:=N1*N2;
Writeln('EL RESULTADO ES: ',m:6:2);
READLN;
END;
4:BEGIN;
Writeln(' * DIVISION *');
WRITELN('CAPTURA EL PRIMER NUMERO');
READLN(N1);
Writeln('CAPTURA EL SEGUNDO NUMERO');
READLN(N2);
D:=N1/N2;
Writeln('EL RESULTADO ES: ',D:6:2);
READLN;
END;
5:BEGIN;
```

```
Writeln('*ADIOS*');

READLN;

END;

END;

END.
```

DIAGRAMA 17

```
Program prog18

{programa que calcula el área de un triangulo, cuadrado o circulo};

Uses crt;

Const

Pi=3.141592654;

Var

A,t,c,cu,b,al,r:real;

Op:integer;

Begin

Clrscr;

Writeln(' *menu*');

Writeln(' 1 − triangulo');

Writeln(' 2 − cuadrado');Writeln(' 3 − circulo');

Writeln(' 4 − salida');

Writeln('seleccione una opción');

Readln(op);

Case op of

1:begin

Writeln(' *triangulo*');

Writeln('que tamaño es la base');
```

```
Readln(b);
Writeln('que tamaño es la altura');
Readln(al);
T:=(b*al)/2;
Writeln('el área del triangulo es: ',t:6:2);
Readln;
End;
2:begin
Writeln('cuadrado');
Writeln('que medida tiene uno de sus lados');
Readln(b);
Cu:=b*b;
Writeln('el área del cuadrado es: ',cu:6:2);
Readln;
End;
3:begin
Writeln(' *circulo*');
Writeln('que tamaño es el radio del circulo');
Readln(r);
C:=pi*(r*r);
Writeln('el área del circulo es: ',c:6:2);
Readln;
End;
4:begin
Writeln('*adios*');
Readln
```

```
End;

End;

End.
```

DIAGRAMA 18

```
Program prog19;

{menú de opciones para obtener el sueldo diario de una persona, su edad o la conversión de pesos a dólares}

Uses crt;

Var

S,st,p,c,d:real;

Op,h,a1,a2,e:integer;

Begin

Clrscr;

Writeln('menu de opciones');

Writeln;

Writeln(' 1− sueldo diario de una persona');

Writeln;

Writeln(' 2− edad de una persona');

Writeln;

Writeln(' 3− conversión de pesos a dólares');

Writeln;

Writeln(' 4− salir');

Writeln;

Writeln('seleccione una opción y presione enter');

Readln(op);

Case op of

1:begin
```

```
Clrscr;
Writeln('sueldo diario de una persona');
Writeln;
Writeln('sueldo por horas de trabajo');
Readln(s);
Writeln('horas trabajadas al dia');
Readln(h);
St:=s*h;
Writeln('el sueldo diario de esta persona es:$',st:3:2);
Readln;
End;
2:begin
Clrscr;
Writeln('edad de una persona');
Writeln;
Writeln('año de nacimiento');
Readln(a1);
Writeln('año actual');
Readln(a2);
E:=a2−a1;
Writeln('su edad es:',e,'años');
Readln;
End;
3:begin
Clrscr;
Writeln('conversion de pesos a dolares');
```

```
Writeln;

Writeln('la cantidad en pesos es:');

Readln(p);

Writeln('tipo de cambio actual');

Readln(c);

D:=p/c;

Writeln('la cantidad en dolares es:$',d:3:2);

Readln;

End;

4:begin

Clrscr;

Writeln('adios');

Writeln('precione "enter" para salir');

Readln;

End;

End

End.
```

DIAGRAMA 19

```
Program prog20;

{programa que calcula el cuadrado, la raiz cuadrada y el coseno de un numero}

Uses crt;

Var

N,r,cuad,c:real;

Op:integer;

Resp:char;

Begin
```

```
Repeat;
Clrscr;
Writeln('***menu de opciones***');
Writeln;
Writeln(' 1− cuadrado');
Writeln(' 2−raiz cuadrada');
Writeln(' 3−coseno');
Writeln;
Writeln('***seleccione una opcion y precione "enter"***');
Readln(op);
Case op of
1:begin
Clrscr;
Writeln('***cuadrado***');
Writeln;
Writeln('capturar numero');
Readln(n);
Cuad:=sqr(n);
Writeln('el cuadrado de el numero es:',cuad:3:2);
Readln;
End;
2:begin
Clrscr;
Writeln('***raiz cuadrada***');
Writeln;
Writeln('capturar numero');
```

```
Readln(n);

R:=sqrt(n);

Writeln('la raiz cuadrada de el numero es:',r:3:2);

Readln;

End;

3:begin

Clrscr;

Writeln('***coseno***');

Writeln;

Writeln('capturar numero');

Readln(n);

C:=cos(n);

Writeln('el coseno de el numero es:',c:3:2);

Readln;

End;

End;

Writeln('desea seguir capturando <s/n>');

Readln(resp);

Until(resp <>'s') and (resp <>'s');

End.
```

DIAGRAMA 20

```
Program prog21;

{programa que suma o multiplica}

Uses crt;

Var

N1,n2,s,r:real;
```

```
Resp:char;
Op:integer;
Begin
Clrscr;
Writeln('*********menu**********');
Writeln('1 − suma');
Writeln('2 − multiplicacion');
Writeln('3 − salir');
Writeln;
Writeln('***seleccione una opcion y precione "enter"***');
Readln(op);
Case op of
1:begin
Clrscr;
Writeln('******suma*******');
Writeln('capturar el primer numero');
Readln(n1);
Writeln('capturar el segundo numero ');
Readln(n2);
S:=n1+n2;
Writeln('el resultado de la suma es:',s:3:2);
Readln;
End;
2:begin
Clrscr;
Writeln('**multiplicacion**');
```

```
Writeln('capturar el primer numero');

Readln(n1);

Writeln('capturar el segundo numero');

Readln(n2);

R:=n1*n2;

Writeln('el resultado de la multiplicacion es :',r:3:2);

Readln;

End;

3:begin

Clrscr;

Writeln('adios');

Readln;

End;

Else

Writeln('error');

Readln;

End;

End.
```

DIAGRAMA 21

```
Program prog22;

{descuento a un empleado segun faltas}

Uses crt;

Const

Y=0.10;

Z=0.20;

E=0.15;
```

```
Var

Nom,c:string;

Nuem,op:integer;

D,x,des,ht,dt,sh,sq,sd:real;

Resp:char;

Begin

Repeat

Clrscr;

Writeln('capturar nombre');

Readln(nom);

Writeln('capturar # de empleado');

Readln(nuem);

Writeln('capturar categoria');

Readln(c);

Begin

Clrscr;

Writeln(' ***menu***');

Writeln('1 − de 6 a 8 faltas');

Writeln('2 − de 1 a 5 faltas');

Writeln;

Writeln('***seleccione una opcion y precione "enter"***');

Readln(op);

Case op of

1:begin

Clrscr;

Writeln('capturar numero de horas trabajadas al dia');
```

```
Readln(ht);

Writeln('capturar sueldo por hora ');

Readln(sh);

Writeln('capturar dias trabajados a la quincena');

Readln(dt);

Sq:=sh*ht*dt;

D:=(sq*z);

Sd:=ht*sh;

Des:=sq−d;

Clrscr;

Writeln('***datos***');

Writeln('nombre:');

Writeln(nom);

Writeln('puesto');

Writeln(c);

Writeln('sueldo diario ');

Writeln(sd:3:2);

Writeln('descuento dela quincena');

Writeln(d:3:2);

Writeln('sueldo quincenal');

Writeln(des:3:2);

Readln;

End;

2:begin

Clrscr;

Writeln('capturar numero de horas trabajadas al dia');
```

```
Readln(ht);

Writeln('capturar sueldo por hora ');

Readln(sh);

Writeln('capturar dias trabajados a la quincena');

Readln(dt);

Sq:=sh*ht*dt;

D:=(sq*y);

Sd:=ht*sh;

Des:=sq−d;

Clrscr;

Writeln('***datos***');

Writeln('nombre:');

Writeln(nom);

Writeln('puesto');

Writeln(c);

Writeln('sueldo diario ');

Writeln(sd:3:2);

Writeln('descuento dela quincena');

Writeln(d:3:2);

Writeln('sueldo quincenal');

Writeln(des:3:2);

Readln;

End;

End;

End;

Clrscr;
```

```
Writeln('desea seguir capturando s/n');
Readln(resp);
Until (resp<>'s') and (resp<>'s');
End.
```

DIAGRAMA 22

```
Program prog23;
{programa que realiza descuentos segun el codigo del articulo}
Uses crt;
Var
N,a,b,c,d:string;
P,t,t1:real;
Resp:char;
Begin
Repeat
Clrscr;
Writeln('el nombre de el articulo es:');
Readln(n);
Writeln('el precio de el articulo es:');
Readln(p);
If (p>=5000) then
Begin
Clrscr;
Writeln('el articulo ', n ,' pertenece a el codigo a');
T:=(p*30)/100;
T1:=p−t;
```

```
Writeln('el precio es :',p:3:2);

Writeln('el descuento fue del 30% esto es :',t:3:2);

Writeln('el precio final es :',t1:3:2);

Readln;

End

Else

If (p>=3000) and (p<5000) then

Begin

Clrscr;

Writeln('el articulo ', n ,' pertenece a el codigo "b"' );

T:=(p*20)/100;

T1:=p−t;

Writeln('pertenece al codigo b');

Writeln('',n);

Writeln('el precio es ',p:3:2);

Writeln('el descuento fue del 20% esto es :',t:3:2);

Writeln('el precio final es :',t1:3:2);

Readln;

End

Else

If (p>=2000) and (p<3000) then

Begin

Clrscr;

Writeln('el articulo ', n ,' pertenece a el codigo "c"');

T:=(p*10)/100;

T1:=p−t;
```

```
Writeln('pertenece al codigo c');

Writeln('',n);

Writeln('el precio es :',p:3:2);

Writeln('el descuento fue del 10% esto es :',t:3:2);

Writeln('el precio final es :',t1:3:2);

Readln;

End;

If (p<2000) then

Begin

Clrscr;

Writeln('el articulo ', n ,' pertenece a el codigo "d"');

T:=(p*5)/100;

T1:=p−t;

Writeln('el precio es :',p:3:2);

Writeln('el descuento fue del 5% esto es :',t:3:2);

Writeln('el precio final es :',t1:3:2);

Readln;

End;

Writeln('desea seguir capturando <s/n>');

Readln(resp);

Until (resp<>'s') and (resp<>'s');

Writeln('presione enter para salir');

Readln;

End.

End.
```

DIAGRAMA 23

```
Program prog24;
{programa que calcula el porcentaje de comision de cuatro empleados}
Uses crt;
Var
Nom:string[15];
Vta,total,pv:real;
Cv,ce:integer;
Begin
Clrscr;
Ce:=1;
While ce<=4 do
Begin
Writeln('nombre');
Readln(nom);
Cv:=0;
Total:=0;
Repeat
Writeln('ventas');
Readln(vta);
Total:=total+vta;
Cv:=cv+1;
Until cv=3;
Clrscr;
Pv:=total*0.005;
Clrscr;
Writeln('su nombre es:',nom);
```

```
Writeln('el total de ventas es:',total:2:2);

Writeln('el porcentaje de la venta es:',pv:2:2);

Ce:=ce+1;

End;

Writeln('precione "enter" para salir');

Readln;

End.
```

DIAGRAMA 24

```
Program prog25

{programa que calcula el promedio de 3 calificaciones de 2 alumnos};

Uses crt;

Var

M,s,n:string[15];

X,ca,c:integer;

P,total:real;

Begin

Clrscr;

Ca:=1;

While ca<=2 do

Begin

Writeln('nombre');

Readln(n);

Writeln;

Writeln('semestre que cursa');

Readln(s);

Writeln;
```

```
Writeln('nombre de la materia');

Readln(m);

X:=0;

Total:=0;

Repeat;

Writeln;

Writeln('calificacion');

Readln(c);

Total:=total+c;

X:=x+1;

Until x=3;

Clrscr;

P:=total/3;

Writeln('nombre: ',n);

Writeln;

Writeln('semestre que cursa: ',s);

Writeln;

Writeln('nombre de la materia: ',m);

Writeln;

Writeln('el promedio de calificaciones: ',p:2:2);

Writeln;

Writeln;

Ca:=ca+1;

End;

Writeln('presione "enter" para salir');

Readln;
```

```
End.
```

DIAGRAMA 25

```
Program prog26
{programa que calcula el total de 3 carros};
Uses crt;
Var
Cc:integer;
M,c,mar,p:string[15];
Pre,t,tot,imp:real;
Begin
Clrscr;
Cc:=1;
T:=0;
While cc<=3 do
Begin
Writeln('programa que calcula el precio de 3 carros');
Writeln;
Writeln('modelo del carro');
Readln(m);
Writeln('marca del carro');
Readln(mar);
Writeln('cilindros del carro');
Readln(c);
Writeln('puertas del carro');
Readln(p);
Writeln('precio del auto');
```

```
Readln(pre);

T:=t+pre;

Clrscr;

Cc:=cc+1;

End;

Imp:=t*0.15;

Tot:=t+imp;

Writeln('el subtotal de los 3 carros es:', t:3:2);

Writeln;

Writeln('el impuesto del 15% a los carros es de:', imp:3:2);

Writeln;

Writeln('el total de los carros con impuestos es de :', tot:3:2);

Writeln;

Writeln;

Writeln('presione "enter" para salir');

Readln;

End.
```

DIAGRAMA 26

```
Program prog27

{programa que calcula el precio de 4 peliculas};

Uses crt;

Var

Cp,cc,ca:integer;

P,t,tot,st:real;

Mem,c,np,cla:string[15];

Begin
```

```
Clrscr;
Cc:=1;
While cc<=2 do
Begin
Writeln('nombre del cliente');
Readln(c);
Writeln;
Writeln('numero de membresia');
Readln(mem);
Cp:=0;
T:=0;
Repeat
Writeln;
Writeln('nombre de pelicula');
Readln(np);
Writeln;
Writeln('categoria de la pelicula "1,2,3"');
Readln(ca);
Writeln;
Writeln('clasificacion de la pelicula');
Readln(cla);
Writeln;
Writeln('precio de la pelicula');
Readln(p);
If ca=1 then
Begin
```

```
Tot:=p*0.3;

St:=p−tot;

End;

If ca=2 then

Begin

Tot:=p*0.2;

St:=p−tot;

End;

If ca=3 then

Begin

Tot:=p*0.1;

St:=p−tot;

End;

T:=t+st;

Cp:=cp+1;

Until cp=4;

Clrscr;

Writeln('el total a pagar por 4 peliculas es de:', t:3:2);

Readln;

Cc:=cc+1;

End;

Textcolor(yellow);

Writeln;

Writeln;

Writeln;

Writeln;
```

```
Writeln('el programa ha concluido, presione enter para salir');
Readln;
End.
```

DIAGRAMA 27

```
Program prog28
{programa que calcula la suma y el descuento de 8 articulos};
Uses crt;
Var
Nom:string[15];
Pre,t,tot,des:real;
Ca:integer;
Begin
Clrscr;
Ca:=1;
Tot:=0;
While ca<=8 do
Begin
Writeln;
Writeln('nombre del articulo');
Readln(nom);
Writeln;
Writeln('precio del articulo');
Readln(pre);
Tot:=tot+pre;
Ca:=ca+1;
End;
```

```
If (tot>=2000) then
Begin
Des:=tot*0.1;
T:=tot−des;
Writeln('el total de los articulos es de:', tot:3:2);
Writeln;
Writeln('el descuento del 10% por ser mayor de 2000 es de:', des:3:2);
Writeln;
Writeln('el total a pagar es de:', t:3:2);
Readln;
End;
If (tot<2000) then
Begin
Writeln('el total de los articulos es de:', tot:3:2);
Readln;
End;
End.
```

DIAGRAMA 28

```
Program prog29
{programa que calcula el precio de prendas en una limpiaduria};
Uses crt;
Const
Lun=0.05;
Mar=0.07;
Mie=0.08;
```

```
Jue=0.09;

Vie=0.1;

Pan=30;

Cha=50;

Blu=20;

Cam=25;

Ves=40;

Fal=60;

Sac=80;

Var

T,st,des,pan1,cha1,blu1,cam1,ves1,fal1,sac1:real;

Resp:char;

Op:integer;

Begin

Repeat

Clrscr;

Writeln(' limpiaduria');

Writeln;

Writeln('cantidad de pantalones entregados');

Readln(pan1);

Writeln;

Writeln('cantidad de chammarras entregadas');

Readln(cha1);

Writeln;

Writeln('cantidad de blusas entregadas');

Readln(blu1);
```

```
Writeln;
Writeln('cantidad de camisas entregadas');
Readln(cam1);
Writeln;
Writeln('cantidad de vestidos entregados');
Readln(ves1);
Writeln;
Writeln('cantidad de faldas entregadas');
Readln(fal1);
Writeln;
Writeln('cantidad de sacos entregados');
Readln(sac1);
Clrscr;
Writeln('menu de opciones');
Writeln;
Writeln('1 − lunes');
Writeln('2 − martes');
Writeln('3 − miercoles');
Writeln('4 − jueves');
Writeln('5 − viernes');
Writeln;
Writeln('escoja una opcion');
Readln(op);
Clrscr;
Case op of
1:begin
```

```
St:=(pan1*pan)+(cha1*cha)+(blu1*blu)+(cam1*cam)+(ves1*ves)+(fal1*fal)+(sac1*sac);

Des:=lun*st;

T:=st−des;

Writeln('** lunes **');

Writeln('pantalones entregados:', pan1:2:0);

Writeln('chamarras entregadas:', cha1:2:0);

Writeln('blusas entregadas:', blu1:2:0);

Writeln('camisas entregadas:', cam1:2:0);

Writeln('vestidos entregados:', ves1:2:0);

Writeln('faldas entregadas:', fal1:2:0);

Writeln('sacos entregados:',sac1:2:0);

Writeln;

Writeln('total de articulos:', st:3:2);

Writeln('descuento del dia lunes:', des:3:2);

Writeln('total a pagar:', t:3:2);

Readln;

End;

2:begin

St:=(pan1*pan)+(cha1*cha)+(blu1*blu)+(cam1*cam)+(ves1*ves)+(fal1*fal)+(sac1*sac);

Des:=mar*st;

T:=st−des;

Writeln('** martes **');

Writeln('pantalones entregados:', pan1:2:0);

Writeln('chamarras entregadas:', cha1:2:0);

Writeln('blusas entregadas:', blu1:2:0);

Writeln('camisas entregadas:', cam1:2:0);
```

```
Writeln('vestidos entregados:', ves1:2:0);

Writeln('faldas entregadas:', fal1:2:0);

Writeln('sacos entregados:',sac1:2:0);

Writeln;

Writeln('total de articulos:', st:3:2);

Writeln('descuento del dia martes:', des:3:2);

Writeln('total a pagar:', t:3:2);

Readln;

End;

3:begin

St:=(pan1*pan)+(cha1*cha)+(blu1*blu)+(cam1*cam)+(ves1*ves)+(fal1*fal)+(sac1*sac);

Des:=mie*st;

T:=st−des;

Writeln('** miercoles **');

Writeln('pantalones entregados:', pan1:2:0);

Writeln('chamarras entregadas:', cha1:2:0);

Writeln('blusas entregadas:', blu1:2:0);

Writeln('camisas entregadas:', cam1:2:0);

Writeln('vestidos entregados:', ves1:2:0);

Writeln('faldas entregadas:', fal1:2:0);

Writeln('sacos entregados:',sac1:2:0);

Writeln;

Writeln('total de articulos:', st:3:2);

Writeln('descuento del dia miercoles:', des:3:2);

Writeln('total a pagar:', t:3:2);

Readln;
```

```
End;
4:begin
St:=(pan1*pan)+(cha1*cha)+(blu1*blu)+(cam1*cam)+(ves1*ves)+(fal1*fal)+(sac1*sac);
Des:=jue*st;
T:=st−des;
Writeln('** jueves **');
Writeln('pantalones entregados:', pan1:2:0);
Writeln('chamarras entregadas:', cha1:2:0);
Writeln('blusas entregadas:', blu1:2:0);
Writeln('camisas entregadas:', cam1:2:0);
Writeln('vestidos entregados:', ves1:2:0);
Writeln('faldas entregadas:', fal1:2:0);
Writeln('sacos entregados:',sac1:2:0);
Writeln;
Writeln('total de articulos:', st:3:2);
Writeln('descuento del dia jueves:', des:3:2);
Writeln('total a pagar:', t:3:2);
Readln;
End;
5:begin
St:=(pan1*pan)+(cha1*cha)+(blu1*blu)+(cam1*cam)+(ves1*ves)+(fal1*fal)+(sac1*sac);
Des:=vie*st;
T:=st−des;
Writeln('** viernes **');
Writeln('pantalones entregados:', pan1:2:0);
Writeln('chamarras entregadas:', cha1:2:0);
```

```
Writeln('blusas entregadas:', blu1:2:0);

Writeln('camisas entregadas:', cam1:2:0);

Writeln('vestidos entregados:', ves1:2:0);

Writeln('faldas entregadas:', fal1:2:0);

Writeln('sacos entregados:',sac1:2:0);

Writeln;

Writeln('total de articulos:', st:3:2);

Writeln('descuento del dia viernes:', des:3:2);

Writeln('total a pagar:', t:3:2);

Readln;

End;

End;

Clrscr;

Writeln('desea seguir capturando s/n');

Readln(resp);

Until (resp<>'s') and (resp<>'s');

End.
```

DIAGRAMA 29

```
Program prog30

{programa que calcula el promedio de un alumno con 3 calificaciones};

Uses crt;

Var

Nom:string[15];

Sum,cal,prom:real;

Ccal:integer;
```

```
Begin
Clrscr;
Write('nombre del alumno: ');
Readln(nom);
Sum:=0;
Ccal:=1;
While ccal<=3 do
Begin
Writeln;
Write('calificacion',ccal,': ');
Readln(cal);
Sum:=sum+cal;
Ccal:=ccal+1;
End;
Prom:=sum/3;
Begin
Clrscr;
Writeln('alumno:',nom);
Writeln;
Writeln('el promedio es:',prom:2:2);
Writeln;
Writeln('presione enter para salir');
Readln
End;
End.
```

DIAGRAMA 30

```
PROGRAM PROG31
{PROGRAMA QUE CONVIERTE MINUTOS A SEGUNDOS};
USES CRT;
VAR
min,seg:real;
resp:char;
begin
clrscr;
write('Deseas convertir minutos a segundos: ');
readln(resp);
while (resp='s') or (resp='S') do
begin
Write('Captura los minutos:');
readln(min);
seg:=min*60;
Writeln('',min:2:2,' minutos son ',seg:2:2,' segundos');
writeln;
writeln('Desea seguir capturando');
readln(resp);
end;
end.
```

DIAGRAMA 31

```
PROGRAM PROG32;
{PROGRAMA QUE CAPTURA NOMBRES Y SE SALE EN UN NOMBRE DETERMINADO}
USES CRT;
```

```
VAR
N,NOM,hector:STRING[12];
BEGIN
CLRSCR;
REPEAT
WRITELN('DAME NOMBRE');
READLN(NOM);
N:=NOM;
UNTIL (NOM='HECTOR') OR (NOM='hector');
END.
```

DIAGRAMA 32

```
program prog33
{programa que calcula la raiz cuadrada de un numero};
uses crt;
var
num,i:integer;
raiz:real;
begin
clrscr;
num:=0;
for i:=1 to 10 do
begin
num:=num+1;
raiz:=sqrt(num);
writeln('La raiz cuadrada de',num:3,' es :',raiz:2:2);
readln;
```

```
end;

end.
```

DIAGRAMA 33

```
program prog34

{programa que calcula la tabla del 2};

uses crt;

var

i:integer;

res:real;

begin

clrscr;

for i:=1 to 10 do

begin

res:=2*i;

writeln(`2 x `,i:3,' es :',res:2:2);

readln;

end;

end.
```

DIAGRAMA 34

```
program prog35

{programa que calcula el cuadrado de un numero};

uses crt;

var

num,i:integer;

cua:real;

begin
```

```
clrscr;
num:=0;
for i:=1 to 10 do
num:=num+1;
cua:=sqr(num);
writeln('El cuadrado de',num:3,' es :',cua:2:2);
readln;
end.
```

DIAGRAMA 35

```
program prog36
{programa que despliaga los numeros del 1 al 100};
uses crt;
var
i:integer;
begin
clrscr;
for i:=1 to 100 do
write(' ',i:3);
readln;
end.
```

DIAGRAMA 36

```
program prog37
{programa que calcula la tabla de multiplicar de cualquier numero};
uses crt;
var
num,res:real;
```

```
i:integer;
begin
clrscr;
writeln('Que numero deseas utilizar para calcular su tabla');
readln(num);
for i:=1 to 10 do
begin
res:=num*i;
writeln('',num:2:2,' x',i:2,' es: ',res:2:2);
readln;
end;
end.
```

DIAGRAMA 37

```
program prog38
{reciproco de los numeros 1 al 25};
uses crt;
var
i:integer;
res:real;
begin
clrscr;
for i:=1 to 25 do
begin
res:=1/i;
writeln('1/',i:2,' es igual a:',res:2:3);
readln;
```

```
end;

end.
```

DIAGRAMA 38

```
program prog39

{programa que calcula el cuadrado de veinte numeros apartir de un numero x};

uses crt;

var

i:integer;

num,res:real;

begin

clrscr;

write('Desde que numero deseas empezar: ');

readln(num);

for i:=1 to 20 do

begin

res:=sqr(num);

writeln('el cuadrado de ',num:2:2,' es:', res:2:2);

readln;

num:=num+1;

end;

end.
```

DIAGRAMA 39

```
program prog40

{programa que depliega numeros del 1 − 500};

uses crt;

var
```

```
num,x,y:integer;
begin
clrscr;
x:=1;
y:=1;
for num:=1 to 500 do
begin
gotoxy(x,y);
write(num);
y:=y+1;
if (y=26) then
begin
y:=1;
x:=x+4;
end;
end;
readln;
end.
```

DIAGRAMA 40

```
program prog41;
{programa que captura la tabla de multiplicar del 2 y 3}
uses crt;
var
x,p,i,j:integer;
begin
clrscr;
```

```
x:=2; p:=0;
for i:= 1 to 2 do
begin
for j:=1 to 10 do
begin
p:=x*j;
writeln('',x,' * ',j,'=',p);
end;
x:=x+1;
end;
readln;
end.
```

DIAGRAMA 41

```
program prog42
{programa que calcula el promedio de 4 calificaciones};
uses crt;
var
i:integer;
prom,cal,tot:real;
begin
clrscr;
tot:=0;
for i:=1 to 4 do
begin
writeln('calificacion ',i:1,':');
readln(cal);
```

```
tot:=tot+cal;

end;

prom:=tot/4;

writeln('el promedio de las 4 calificaciones es:',prom:2:2);

readln;

end.
```

DIAGRAMA 42

Algoritmo, en matemáticas, método de resolución de cálculos complicados mediante el uso repetido de otro método de cálculo más sencillo. Ejemplos básicos son los métodos para efectuar operaciones aritméticas (multiplicación, división, obtención de raíces cuadradas...), la obtención del máximo común divisor y del mínimo común múltiplo de un número mediante su descomposición en factores primos, y la división de un polinomio por x − a mediante la regla de Ruffini. En la actualidad, el término algoritmo se aplica a muchos de los métodos de resolución de problemas que emplean una secuencia mecánica de pasos, como en el diseño de un programa de ordenador o computadora. Esta secuencia se puede representar en forma de un diagrama de flujo para que sea más fácil de entender.

En las computadoras con lógica de microordenadores incorporada, esta lógica es un tipo de algoritmo. A medida que los equipos informáticos se hacen más complejos, más y más algoritmos del software toman la forma del llamado hard−software. Esto es, cada vez más, se están convirtiendo en parte de los circuitos básicos de los ordenadores o en módulos auxiliares; también están apareciendo por sí solos en máquinas específicas como las calculadoras de nóminas. En la actualidad, existen muchos algoritmos para diversas aplicaciones y algunos sistemas avanzados, como los algoritmos de inteligencia artificial, llegarán a ser corrientes en el futuro.

Un algoritmo es una serie de operaciones detalladas y no ambiguas, a ejecutar paso a paso y que conducen a la resolución de un problema, en otras palabras es un conjunto de reglas para resolver una cierta clase de problema o forma de escribir el problema.

Consiste en dos partes esenciales:

Una descripción de acciones que deben ser ejecutadas y una descripción de los datos que son manipulados por esas acciones; las acciones se describen mediante, las llamadas sentencias y datos mediante declaraciones y definiciones.

− Características de algoritmos −

a) un algoritmo debe ser preciso e indicar el orden de realización de cada paso

b) un algoritmo debe estar definido, si se sigue un algoritmo 2 veces se debe tener el mismo resultado

c) un algoritmo debe ser finito, si se sigue un algoritmo se debe terminar en algun momento.

Diagrama de flujo de datos: Una técnica de diseño que permite la documentación de un sistema o programa en varios niveles de generalidad.

Diagrama de flujo: Un diagrama que ilustra el flujo de datos, información y trabajo por medio de símbolos especializados que, cuando se conectan por líneas de flujo, reflejan la lógica de un sistema o programa.

Tipo de datos

Los diferentes objetos de información que Pascal utiliza para trabajar se conocen con el nombre de datos. Cada uno de estos objetos tendrá un tipo de datos diferente acorde con su información. Según del tipo de información de que se trate, podría ser un número, una letra, etc.

La asignación de tipos a los datos tiene dos objetivos principales:

Detectar errores de operaciones en programas.

Determinar como ejecutar las operaciones.

Determinar el rango de valores que puede tomar una variable

Reservar en memoria el espacio correcto para cada tipo de datos.

En pascal todos los datos que se vayan a utilizar deben de tener sus tipos declarados previamente.

Clasificación de los tipos de datos

Enteros:

Byte: Números enteros comprendidos entre 0 y 255.

Integer: Enteros entre −32768 y 32767.

Longint: Enteros entre −2.147.483.648 y 2.147.483.647

Word: Enteros positivos entre 0 y 65535.

Reales:

Los tipos de datos reales representan al conjunto de los números reales. Todo número se puede representar como un real, aunque su representación interna no es la misma.

Real: 2.910−39 ... 1.71038 (11−12 cifras)

Single: 1.510−45 ... 3.41038 (7−8 cifras)

Double: 5.010−324 ... 1.710308 (15−16 cifras)

Extended: 1.910−4932 ... 1.1104932 (19−20 cifras)

Comp: −2−63 + 1 ... 263 − 1 (19−20 cifras)

DATOS

NUMERICO CADENA DE CARACTERES LOGICOS

REALES ENTEROS

DECIMALES COMA FLOTANTE

La Estructura de Datos

Es una colección de elementos de datos el medio en que se relacionan unos elementos con otros determina el tipo de estructura de datos.

Un elemento o dato es la cantidad que toma un único elemento que puede cambiar de cuando en cuando (se denomina variable).

Las estructuras de datos mas usuales son cadenas matrices, listas, tablas, árboles, pilas, colas y ficheros.

Símbolos

Entrada/salida: cualquier tipo de introducción de datos en la memoria desde los periféricos, entrada o registro de la información procesada en un periférico salida.

Lecturas tarjeta perforada, indica inicio y final del programa.

Conector (sirve para enlazar 2 partes cualesquiera de un organigrama a través de un conector en la salida y otro en la entrada, se refiere a la conexión en la misma página del diagrama

Símbolo de enlace de comunicaciones, es usado para uní terminales remotas con los sistemas de comunicación.

Línea de flujo: indica el sentido de ejecución de la operación

Línea conectora: sirve de unión entre 2 símbolos.

Terminal (representa el comienzo,<<inicio>>, y el final, <<fin>> de un programa, puede representar también una parada o interrupción programado que sea necesario realizar en un programa.

Proceso (cualquier tipo de operación que pueda originar cambio de valor formato o posición de la información almacenada en la memoria, operaciones aritméticas de transferencia, etc.

Conector (conexión entre 2 puntos del organigrama situado en las paginas diferentes.

Impresora (se utiliza en ocasiones en lugar del símbolo de E/S

Teclado, se utiliza para añadir en lugar del símbolo de E/S

Compilar

Fuente de un programa desde un lenguaje de alto nivel a un código objeto antes de la ejecución del programa. El código objeto es un código máquina ejecutable. De manera más general, compilar suele utilizarse para describir la traducción de cualquier descripción simbólica de alto nivel a un formato simbólico de bajo nivel o legible por una máquina. Un programa que realiza esta función se denomina compilador.

Errores

Las personas tienen una gran capacidad para compensar los errores sufridos por los datos transmitidos. Es posible mantener una conversación entre dos individuos aun cuando sólo llegue intacto un 30% de los datos. Los ordenadores están en el otro extremo del espectro. Un único error de transmisión puede echar por tierra todo un diálogo. Por tal razón, la comprobación y prevención de errores constituye un requisito básico de cualquier tipo de comunicación de datos.

La protección contra los errores suele efectuarse añadiendo bits adicionales a los paquetes que contienen los datos a transferir. Alrededor del 4% de los bits en un paquete de datos se dedican a la detección de errores. El método más sencillo de aprovechar estos bits es fijar un bit de paridad, un único dígito que se coloca para que la suma de una determinada secuencia de bits sea 1 o 0. Es una forma muy eficaz de detectar errores de bits aislados, pero no sirve cuando hay errores que afectan a 2 (o 4) bits.

Normalmente se utilizan otras técnicas más depuradas conocidas como sumas de control. Se fundamentan en complejos cálculos matemáticos y resultan eficaces para detectar diferentes tipos de errores. Más enrevesadas resultan las técnicas de corrección de errores, que suelen precisar un porcentaje mayor de bits, pero que son capaces de corregir realmente errores de transmisión eliminando la necesidad de retransmitir paquetes enteros por culpa de un único bit.

Ejecutar

Realiza una instrucción o pasar un programa en un ordenador

Fases de la programación:

1.− ANALISIS: Cuando se tiene y piensa la idea o problema a ejecutar.

2.− DISEÑO: Se plantea un algoritmo con los pasos a dibujar o realizarse.

3.− IMPLEMENTACION: Los pasos de algoritmo se transforma en un código pascal y se plasman en el programa

4.− PRUEBAS: Se corre el programa y se registra su efectividad.

5.− DEPURACION: después de haber corrido el programa se hacen las correcciones necesarias.

6.− RETROALIMENTACION Y LIBERACION: Después de haber corregido los errores se regresa al paso 4 en caso de haber existido.

Estructura de un programa

Programación de computadoras es el proceso de planificar una secuencia de instrucciones que ha de surgir en una computadora.

Un programa es la secuencia de instrucciones que indica las acciones que ha de ejecutar la computadora.

Se podría considerar la programación como el conjunto de actividades y operaciones realizadas por el personal de informáticos tendientes a influir a la maquina para que puedan realizar las funciones en el algoritmo

Estructura de un programa en pascal

Un programa en lenguaje pascal esta dividido en secciones:

Program_____cabecera del programa

Uses_____ unidades

Label_____ etiquetas

Const_____ declaración de constantes

Type_____ tipos;

Var_____declaración de variables**;**

Procedure____ procedimientos;

Funtion_____ Funciones

Begin

{cuerpo del programa}

end

***Sentencia Program:**

Especifica el nombre del programa, que puede ser un nombre de solo 63 caracteres

EJ: program prog1.

***Sentencia uses:**

Indica que unidades son usadas por el programa la unidad mas utilizada es para pantalla

EJ: Uses crt;

***Sentencia label:**

Generalmente no se utiliza Ej: label A (A es la etiqueta)

***Sentencia Const:**

Se declaran cualquiera de las constantes que se vayan a utilizar siempre se utiliza el =

EJ: IVA=.10

***Sentencia Type:**

En esta se pueden crear y declarar sus propios tipos de datos EJ: Nombre = string[35]

Var N:nombre

***Sentencia Var:**

Se declaran los nombres y tipos de variables que se van a usar en el programa

EJ: var: Num1, num2: integer;

Compilar

Un compilador es un programa que traduce el programa fuente(conjunto de instrucciones de un lenguaje de alto nivel copol y pascal) a programa objeto(instrucciones a un lenguaje maquina que la computadora pueda interpretar y ejecutar)

El compilador efectúa solo la traducción, no ejecuta el programa.

Un intérprete es un programa que procesa los programas escritos en un lenguaje de alto nivel sin embargo esta diseñado de modo que no existe independencia entre la etapa de traducción y cada ejecución.

En cualquier lenguaje de alto nivel en que se escriba un programa este debe ser traducido al lenguaje maquina antes de ser ejecutado.

Esta confección de instrucciones de alto nivel a instrucciones de nivel de maquina se hace por programas del software del sistema denominados compiladores e interpretes.

Estos programas especiales se denominan traductores.

Ejecución

La modificación del programa fuente es muy sencilla en el interprete, mientras que en el compilador además de la modificación con el programa, esto es, será necesario crear un programa compilado, esto significa que para comprobar el funcionamiento de un programa una vez modificado habrá que volver a compilarlo

Las ejecuciones sucesivas de un programa compilado no necesita traducciones del programa fuente. como el interprete no produce un programa objeto se debe realizar el proceso de reducción cada vez que se ejecuta un programa.

LENGUAJES DE BAJO NIVEL

Vistos a muy bajo nivel, los microprocesadores procesan exclusivamente señales electrónicas binarias. Dar una instrucción a un microprocesador supone en realidad enviar series de unos y ceros espaciadas en el tiempo de una forma determinada. Esta secuencia de señales se denomina código máquina. El código representa normalmente datos y números e instrucciones para manipularlos. Un modo más fácil de comprender el código máquina es dando a cada instrucción un mnemónico, como por ejemplo STORE, ADD o JUMP. Esta abstracción da como resultado el ensamblador, un lenguaje de muy bajo nivel que es específico de cada microprocesador.

Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son a menudo difíciles de aprender. Más importante es el hecho de que los programas escritos en un bajo nivel son prácticamente específicos para cada procesador. Si se quiere ejecutar el programa en otra máquina con otra tecnología, será necesario rescribir el programa desde el principio.

LENGUAJES DE ALTO NIVEL

Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de textos. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel

de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Programación orientada a objetos, en informática, un estilo de programación en el que un programa se contempla como un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Una clase define las estructuras de datos y rutinas de un objeto. Un objeto es una instancia de una clase, que se puede usar como una variable en un programa. En algunos lenguajes orientados a objetos, éste responde a mensajes, que son el principal medio de comunicación. En otros lenguajes orientados a objeto se conserva el mecanismo tradicional de llamadas a procedimientos.

Programa fuente: Código del programa original.

Programa objeto: Un programa a nivel de lenguaje máquina que resulta de la compilación de un programa fuente.

PREGUNTAS FRECUENTES

1− ¿Qué es un diagrama de flujo?

Es un cuadro que explica paso por paso la secuencia de un objeto o programa

2− ¿Qué es un algoritmo?

Es una serie de operaciones detallistas y no ambiguas, a ejecutar paso por paso y que conducen a la resolución de un problema, es un conjunto de reglas para resolver un problema

3− ¿Cuáles son las características de los algoritmos?

Un algoritmo debe ser ordenado paso a paso, si se sigue un algoritmo 2 veces debe tener el mismo resultado y debe ser finito

4− ¿Cuáles son las partes esenciales de los algoritmos?

Acciones, las cuales son o deben ser ejecutadas mediante las llamadas sentencias; descripciones de datos que funcionan por medio de declaraciones y definiciones

5− Menciona los tipos de datos y especifique cada uno

Numérico, son los datos los cuales son números reales (decimales o fracciones) y enteros; caracteres, son datos de puras letras; lógicos, es mezcla de números y letras.

6− Cuales son los métodos para la resolución de problemas por computadora

Se utiliza un algoritmo complejo, conduce a la escritura de un programa y su ejecuciones la misma

7− Menciona las partes de un programa

program, uses, label, const, var, type, procedure, function

8− ¿Que es programación?

Conjunto de actividades y operaciones realizados por el personal inform. Tendientes a instruir a la maquina para que pueda realizar las funciones previstas en el algoritmo

9− ¿A que se le llama estructura de datos?

A una colección de datos relacionados entre si por ejemplo: cadena, tablas, árboles, pilas.

10− Definir

Sentencia: la instrucción para realizar uno o mas problemas.

Compilar: traduce todos los códigos fuente de un programa

Interprete: procesa los programas escritos con lenguaje de alto nivel

Declaración: declarar variables o constantes

Ejecutar: realizar una instrucción o realizar un programa

11− Fases para la resolución de un problema

1− Análisis del problema− se tiene una idea de solucionar los problemas

2− Diseño del algoritmo− se escribe paso por paso la solución

3− Codificación− se escribe en el programa

4− Compilación y ejecución − se pone a correr el programa

5− Verificación− se ven si hay algún problema o error y se corrige

6− Depuración− si no corre el programa se vuelve a corregir

7− Documentación− se salva

CICLOS FOR

El ciclo** FOR **repite una sentencia un determinado número de veces que se indica al momento de llamar al ciclo.

Lo que hace FOR es que incrementa una variable en uno desde un valor inicial hasta un valor final ejecutando en cada incremento la sentencia que se quiere repetir. Su sintaxis es:

FOR identificador:= inicio TO fin DO instrucción;

Donde el identificador es la variable que se incrementará, inicio es el primer valor que tendrá dicha variable y

fin es el valor hasta el cual se incrementará la misma; instrucción es la sentencia (sencilla o compuesta) que se ejecutará en cada incremento de la variable.

El siguiente ejemplo escribe los números del 1 al 50 en pantalla. La variable utilizada es "Numero".

```
PROGRAM Ciclo_FOR;
VAR
Numero : Integer;

BEGIN
FOR Numero := 1 to 50 DO
WriteLn(Numero);
END.
```

Una de las limitaciones de los ciclos FOR es que una vez iniciado el ciclo se ejecutará el número de veces predefinido sin posibilidad de agregar o eliminar ciclos.

Es posible hacer que un ciclo cuente hacia atrás, es decir que la variable en lugar de incrementarse se decrementa. Para esto cambiamos la palabra TO por DOWNTO, y colocamos el valor mayor a la izquierda y el menor a la derecha. Ejemplo:

```
PROGRAM Ciclo_FOR_2;
VAR
Numero : Integer;

BEGIN
FOR Numero := 50 DOWNTO 1 DO
WriteLn(Numero);
END.
```

CICLOS WHILE

Los ciclos WHILE ofrecen la ventaja de que la ejecución se realiza mientras se cumpla una condición, por lo tanto es posible controlar el número de repeticiones una vez iniciado el ciclo. Su sintaxis es:

WHILE condición DO instrucción

Donde condición es la condición que se evaluará, mientras ésta sea verdadera se ejecutará la instrucción, que es una sentencia simple o compuesta.

Un programa que escriba los números del 1 al 50, utilizando el ciclo WHILE se vería como sigue:

```
PROGRAM Ciclo_WHILE;
VAR
Numero : Integer;

BEGIN
Numero := 1;
WHILE Numero <= 50 DO
BEGIN
WriteLn (Numero);
Numero := Numero +1;
```

```
END;
END.
```

Al final del programa la variable Numero guardará el valor 51, que fue el valor que no cumplió con la condición establecida en el ciclo WHILE.

CICLOS REPEAT−UNTIL

Este tipo de ciclos es muy parecido a los ciclos WHILE, la diferencia entre ambos es que en WHILE la condición se evalúa al principio del ciclo, en cambio en REPEAT−UNTIL se evalúa al final, lo que significa que en un ciclo REPEAT−UNTIL la sentencia se ejecutará por lo menos una vez, cosa que puede no ocurrir en el ciclo WHILE. Ejemplo:

```
PROGRAM Ciclo_RepeatUntil;
VAR
Numero : Integer;
BEGIN
Numero := 1;
REPEAT
WriteLn (Numero);
Numero := Numero + 1;
UNTIL Numero = 50;
END.
```

Para crear un buen programa es necesario dotarlo con capacidad de decisión con base en las variables o eventos definidos por el programador, para que el programa sea aplicable en un entorno más generalizado y no solo para un problema específico.

SINTAXIS DE PASCAL:

Alternativa doble ::= **if** <Condición> **then** <Sentencia> **else** <Sentencia>

Alternativa múltiple ::= **case** <Expresión de Tipo ordinal> **of** <Caso>{**;** <Caso>} **end**
Alternativa múltiple ::= **case** <Expresión de Tipo ordinal> **of** <Caso>{**;** <Caso>} **else** <Sentencia> **end**

Alternativa simple ::= **if** <Condición> **then** <Sentencia>

Bucle con numero fijo de iteraciones ::= **for** <Sentencia de asignación> **to** <Expresión de Tipo ordinal> **do** <Sentencia>
Bucle con numero fijo de iteraciones ::= **for** <Sentencia de asignación> **downto** <Expresión de Tipo ordinal> **do** <Sentencia>

Bucle con salida al final ::= **repeat** <Sentencia>{**;** <Sentencia>} **until** <Condición>

Bucle con salida al principio ::= **while** <Condición> **do** <Sentencia>

Cabecera de función ::= **function** <identificador de función> **:** <identificador de Tipo elemental> **;**
Cabecera de función ::= **function** <identificador de función> **(** <Parámetros formales por valor>{**;** <Parámetros formales por valor>} **) :** <identificador de Tipo elemental> **;**

Cabecera de procedimiento ::= **procedure** <identificador de procedimiento> **;**
Cabecera de procedimiento ::= **procedure** <identificador de procedimiento> **(** <Parámetros formales>{**;**

<Parámetros formales>} **) ;**

Cabecera de programa ::= **program** <identificador de programa> **;**
Cabecera de programa ::= **program** <identificador de programa> **(** <Lista de dispositivos> **);**

Cabecera de unidad ::= **unit** <identificador de unidad> **;**

Carácter de subrayado ::= **_**

Caso ::= <Rango de literales> **:** <Sentencia>

Comentario ::= **{** {<Carácter>} **}**
Comentario ::= **(*** {<Carácter>} ***)**

Condición ::= <Expresión de tipo boolean>

Conversión de tipo ::= <identificador de Tipo ordinal> **(** <Expresión de Tipo ordinal> **)**

Cuerpo de función ::= <Sentencia compuesta>

Cuerpo de procedimiento ::= <Sentencia compuesta>

Cuerpo de programa ::= <Sentencia compuesta>

Declaración de campos ::= <identificador de campo>{**,** <identificador de campo>} **:** <identificador de Tipo de dato>

Declaración de constante ::= <identificador de constante> **=** <Literal>
Declaración de constante ::= <identificador de constante> **:** <identificador de Tipo elemental> **=** <Literal>

Declaración de función ::= <Cabecera de función><Zona de declaraciones><Cuerpo de función>

Declaración de interfaz ::= <Cabecera de función> | <Cabecera de procedimiento>

Declaración de procedimiento ::= <Cabecera de procedimiento><Zona de declaraciones><Cuerpo de procedimiento>

Declaración de subprograma ::= <Declaración de función> | <Declaración de procedimiento>

Declaración de variables ::= <identificador de variable>{**,** <identificador de variable>} **:** <Tipo de dato>

Definición de tipo ::= <identificador de tipo> **=** <Tipo de dato>

Digito ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

Dispositivo ::= **input** | **output** | <identificador de variable tipo fichero>

Elemento de array ::= <Variable de Tipo array> **[** <Expresión de Tipo índice>{**,** <Expresión de Tipo índice>} **]**

Elemento de estructura ::= <Elemento de array> | <Elemento de registro>

Elemento de registro ::= <Variable de Tipo registro> **.** <identificador de campo>

Estructura alternativa ::= <Alternativa simple> | <Alternativa doble> | <Alternativa múltiple>

Estructura de control ::= <Estructura alternativa> | <Estructura iterativa> | <Estructura with>

Estructura iterativa ::= <Bucle con salida al principio> | <Bucle con salida al final> | <Bucle con numero fijo de iteraciones>

Estructura with ::= **with** <Variable de tipo registro> **do** <Sentencia>

Expresión ::= <Literal> | <identificador de constante> | <Variable> | <Expresión unaria> | <Expresión binaria> | <Llamada a función> | <Conversión de tipo>
Expresión ::= **(** <Expresión> **)**

Expresión binaria ::= <Expresión><Operador binario>Expresión>

Expresión unaria ::= <Operador unario><Expresión>

identificador ::= <Letra>{<Letra> | <Digito>}

Letra ::= <Letra mayúscula> | <Letra minúscula> | <Carácter de subrayado>

Letra mayúscula ::= **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z**

Letra minúscula ::= **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**

Lista de dispositivos ::= <Dispositivo>{<Dispositivo>}

Literal ::= <Literal de tipo entero> | <Literal de tipo real> | <Literal de tipo carácter> | <Literal de tipo boolean> | <Literal de tipo string> | <Literal de tipo conjunto> | <Literal de tipo puntero> | <Literal tipo enumerado>

Literal de tipo boolean ::= **true** | **false**

Literal de tipo carácter ::= **'** <Carácter> **'**
Literal de tipo carácter ::= **#** <Digito>{<Digito>}

Literal de tipo conjunto ::= **[ ]**
Literal de tipo conjunto ::= **[** <Rango de valores> **]**

Literal de tipo entero ::= <Signo><Digito>{<Digito>}

Literal de tipo puntero ::= **nil**

Literal de tipo real ::= <Signo><Digito>{<Digito>} **.** <Digito>{<Digito>}
Literal de tipo real ::= <Signo><Digito>{<Digito>} **e** <Signo><Digito>{<Digito>}
Literal de tipo real ::= <Signo><Digito>{<Digito>} **.** <Digito>{<Digito>} **e** <Signo><Digito>{<Digito>}

Literal de tipo string ::= **'** {<Carácter>} **'**

Literal tipo enumerado ::= <identificador de valor enumerado>

Llamada a función ::= <identificador de función>
Llamada a función ::= <identificador de función> **(** <Parámetros actuales por valor>{**;** <Parámetros actuales por valor>} **)**

Llamada a procedimiento ::= <identificador de procedimiento>
Llamada a procedimiento ::= <identificador de procedimiento> **(** <Parámetros actuales>{**;** <Parámetros actuales>} **)**

Operador aritmético entero ::= **+** | **−** | ***** | **div** | **mod**

Operador aritmético real ::= **+** | **−** | ***** | **/**

Operador binario ::= **+** | **−** | ***** | **div** | **mod** | **/** | **and** | **or** | **=** | **<>** | **<** | **>** | **<=** | **>=** | **in**

Operador de conjunto ::= **+** | **−** | ***** | **=** | **<>** | **<=** | **>=** | **in**

Operador de string ::= **+**

Operador lógico ::= **not** | **and** | **or**

Operador relacional ::= **=** | **<>** | **<** | **>** | **<=** | **>=**

Operador unario ::= **−** | **not**

Parámetros actuales ::= <Parámetros actuales por valor> | <Parámetros actuales por variable>

Parámetros actuales por valor ::= <Expresión>

Parámetros actuales por variable ::= <Variable>

Parámetros formales ::= <Parámetros formales por valor> | <Parámetros formales por variable>

Parámetros formales por valor ::= <identificador de variable>{**,** <identificador de variable>} **:** <identificador de tipo>

Parámetros formales por variable ::= **var** <identificador de variable>{**,** <identificador de variable>} **:** <identificador de tipo>

Programa ::= <Cabecera de programa><Zona de declaración de unidades><Zona de declaraciones><Cuerpo de programa> **.**

Rango de literales ::= <Literal de Tipo ordinal>
Rango de literales ::= <Literal de Tipo ordinal> **..** <Literal de Tipo ordinal>
Rango de literales ::= <Rango de literales>{**,** <Rango de literales>}

Rango de valores ::= <Expresion de Tipo ordinal>
Rango de valores ::= <Expresion de Tipo ordinal> **..** <Expresion de Tipo ordinal>
Rango de valores ::= <Rango de valores>{**,** <Rango de valores>}

Sentencia ::= <Sentencia de asignacion> | <Sentencia compuesta> | <Llamada a procedimiento> | <Estructura de control>

Sentencia compuesta ::= **begin** <Sentencia>{**;** <Sentencia>} **end**

Sentencia de asignación ::= <Variable> **:=** <Expresión>

Signo ::= | **+** | **−**

Tipo array ::= <identificador de tipo array>
Tipo array ::= **array[** <Tipo índice>{**,** <Tipo índice>} **] of** <Tipo de dato>

Tipo conjunto ::= <identificador de tipo conjunto>
Tipo conjunto ::= **set of** <Tipo ordinal>

Tipo de dato ::= <Tipo simple> | <Tipo estructurado>

Tipo elemental ::= <Tipo simple> | <Tipo string>

Tipo entero ::= **integer** | **longint**

Tipo enumerado ::= <identificador de tipo enumerado>
Tipo enumerado ::= **(** <Literal tipo enumerado>{**,** <Literal tipo enumerado>} **)**

Tipo estructurado ::= <Tipo array> | <Tipo registro> | <Tipo fichero> | <Tipo conjunto> | <Tipo string>

Tipo fichero ::= <identificador de tipo fichero>
Tipo fichero ::= **text**
Tipo fichero ::= **file of** <Tipo de dato>

Tipo indice ::= <Tipo ordinal>

Tipo ordinal ::= <Tipo ordinal predefinido> | <Tipo enumerado> | <Tipo subrango>

Tipo ordinal predefinido ::= <Tipo entero> | **char** | **boolean**

Tipo puntero ::= <identificador de tipo puntero>
Tipo puntero ::= ^ <identificador de Tipo de dato>

Tipo real ::= **real**

Tipo registro ::= <identificador de tipo registro>
Tipo registro ::= **record** <Declaración de campos>{**;** <Declaración de campos>} **end**

Tipo simple ::= <Tipo ordinal> | <Tipo real> | <Tipo puntero>

Tipo string ::= <identificador de tipo string>
Tipo string ::= **string[** <Literal de tipo entero> **]**

Tipo subrango ::= <identificador de tipo subrango>
Tipo subrango ::= <Literal de Tipo ordinal> **..** <Literal de Tipo ordinal>

Unidad ::= <Cabecera de unidad><Zona de interfaz><Zona de implementación> **end.**

Variable ::= <identificador de variable> | <Elemento de estructura> | <Variable dinámica>

Variable dinámica ::= <Variable de Tipo puntero> ^

Zona de declaración de constantes ::=
Zona de declaración de constantes ::= **const** <Declaración de constante>{**;** <Declaración de constante>} **;**

Zona de declaración de interfaces ::= <Declaración de interfaz>{**;** <Declaración de interfaz>} **;**

Zona de declaración de subprogramas ::=
Zona de declaración de subprogramas ::= <Declaración de subprograma>{**;** <Declaración de subprograma>} **;**

Zona de declaración de unidades ::=
Zona de declaración de unidades ::= **unit** <identificador de unidad>{**,** <identificador de unidad>} **;**

Zona de declaración de variables ::=
Zona de declaración de variables ::= **var** <Declaración de variables>{**;** <Declaración de variables>} **;**

Zona de declaraciones ::= <Zona de declaración de constantes><Zona de definición de tipos><Zona de declaración de subprogramas><Zona de declaración de variables>

Zona de definición de tipos ::=
Zona de definición de tipos ::= **type** <Definición de tipo>{**;** <Definición de tipo>} **;**

Zona de implementación ::= **implementation** <Zona de declaración de subprogramas>

Zona de interfaz ::= **interface** <Zona de declaracion de unidades><Zona de declaración de constantes><Zona de definición de tipos><Zona de declaración de interfaces>

PROGRAMACIÓN MODULAR

Uno de los métodos más conocidos para resolver un problema es dividirlo en problemas más pequeños, llamados subproblemas. De esta manera, en lugar de resolver una tarea compleja y tediosa, resolvemos otras más sencillas y a partir de ellas llegamos a la solución. Esta técnica se usa mucho en programación ya que programar no es más que resolver problemas, y se le suele llamar diseño descendente, metodología del divide y vencerás o programación top−down.

Es evidente que si esta metodología nos lleva a tratar con subproblemas, entonces también tengamos la necesidad de poder crear y trabajar con subprogramas para resolverlos. A estos subprogramas se les suele llamar módulos, de ahí viene el nombre de programación modular. En Pascal disponemos de dos tipos de módulos: los procedimientos y las funciones.

Programación Estructurada es una técnica en la cual la estructura de un programa, esto es, la interpelación de sus partes realiza tan claramente como es posible mediante el uso de tres estructuras lógicas de control:

a. Secuencia: Sucesión simple de dos o mas operaciones.

b. Selección: bifurcación condicional de una o mas operaciones.

c. Interacción: Repetición de una operación mientras se cumple una condición.

Estos tres tipos de estructuras lógicas de control pueden ser combinados para producir programas que manejen cualquier tarea de procesamiento de información.

Un programa estructurado esta compuesto de segmentos, los cuales puedan estar constituidos por unas pocas instrucciones o por una pagina o más de codificación. Cada segmento tiene solamente una entrada y una salida, estos segmentos, asumiendo que no poseen lazos infinitos y no tienen instrucciones que jamás se ejecuten, se denominan programas propios. Cuando varios programas propios se combinan utilizando las tres estructuras básicas de control mencionadas anteriormente, el resultado es también un programa propio.

Estructura Secuencial

Es aquélla en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. La estructura secuencial tiene una entrada y una salida.

Estructura selectiva:

Se utilizan para tomar decisiones lógicas. En éstas se evalúa una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas. En pseudo código éstas palabras son if, then, else. Las estructuras selectivas pueden ser: − simples, dobles o múltiples.

Estructuras Repetitivas

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan Bucles y se denomina Iteración al hecho de repetir la ejecución de una secuencia de acciones. Entre las estructuras repetitivas se encuentran:

Mientras (while)

Repetir (repeat)

Desde (for)

Estructura Mientras (while)

La estructura repetitiva while, es aquélla en que el cuerpo del bucle se repite mientras se cumple una determinada condición

Estructura Repetir (repeat)

Esta estructura se ejecuta hasta que se cumpla una condición determinada que se comprueba hasta el final del bucle. Se ejecuta al menos una vez.

El bucle repetir−Hasta que se repite mientras el valor de la expresión booleana de la condición sea falsa, justo la opuesta de la sentencia mientras.

Estructura Desde/Para (for)

En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. En estos casos en el que el número de iteraciones es fija, se debe usar la estructura desde o para.

La estructura Desde ejecuta las acciones del cuerpo del bucle un número específico de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.

Recursividad: Propiedad de un lenguaje que permite a un procedimiento llamarse a si mismo para volver a

ejecutar su código en una instancia distinta.

Iteración

Las formas de iteración sirven para ejecutar ciclos repetidamente, dependiendo de que se cumplan ciertas condiciones. Una estructura de control que permite la repetición de una serie determinada de sentencias se denomina bucle1 (lazo o ciclo).

loop

...

end loop;

del que se sale, normalmente, mediante una sentencia "exit when" o con una alternativa que contenga una cláusula "exit".

Tabla de ordenadores de operadores

Operador	Operación	Operandos	Ejemplo	Resultado
+	Suma	real , integer	a + b	suma de a y b
−	Resta	real , integer	a − b	Diferencia de a y b
*	Multiplicación	real , integer	a * b	Producto de a por b
/	División	real , integer	a / b	Cociente de a por b
div	División entera	integer	a div b	Cociente entero de a por b
mod	Módulo	integer	a mod b	Resto de a por b
shl	Desplazamiento a la izquierda		a shl b	Desplazar a la izquierda b bits
shr	Desplazamiento a la derecha		a shr b	Desplazar a la derecha b bits

La diferencia entre las instrucciones **Read** y **ReadLn** consiste en que **Read** permite que la siguiente instrucción continúe leyendo valores en la misma línea; mientras que, con **ReadLn** la siguiente lectura se hará después de que se haya tecleado el carácter de fin de línea.

Cuando se tienen datos de tipo char en una instrucción **Read**, los caracteres blancos y los de fin de línea son considerados en el conteo de los elementos de las cadenas de caracteres mientras no se complete el total especificado para cada una de ellas. Cada fin de línea es tratado como un carácter, pero el valor asignado a la variable será un carácter blanco.

Es aconsejable que cada cadena de caracteres se lea en una instrucción **Read** o **ReadLn** por separado, para evitar el tener que ir contando hasta completar la cantidad exacta de caracteres que forman la cadena (o de lo contrario se tendrán resultados sorpresivos y frustrantes al verificar los datos leídos).

El procedimiento **Write** permite que la siguiente instrucción se realice en la misma línea , mientras que **WriteLn** alimenta una nueva línea, antes de finalizar.

CASE−OF−ELSE

Esta forma es muy útil cuando se tiene que elegir entre más de dos opciones, por lo que le llamaremos forma de selección múltiple.

Dependiendo del valor que tenga la expresión selector, se ejecutarán las instrucciones etiquetadas por constante .

Aquí también los bloques de instrucciones pueden ser reemplazados por instrucciones simples.

Conviene tener presente que no debe escribirse punto y coma antes de la palabra else.

FOR−TO−DO

Cuando se sabe de antemano el número de veces que deberá ejecutarse un ciclo determinado, ésta es la forma más conveniente.

El formato para for−to−do es :

for <contador>:=<expresión.1> to <expresión.2> do

begin

<Instrucciones> ;

end;

Al ejecutarse la sentencia for la primera vez, a contador se le asigna un valor inicial (expresion.1), y a continuación se ejecutan las intrucciones del interior del bucle, enseguida se verifica si el valor final (expresión.2) es mayor que el valor inicial (expresión.1); en caso de no ser así se incrementa contador en uno y se vuelven a ejecutar las instrucciones, hasta que el contador sea mayor que el valor final, en cuyo momento se termina el bucle.

REPEAT−UNTIL

La acción de repeat−until es repetir una serie de instrucciones hasta que se cumpla una determinada condición.

Su formato es :

repeat

<instrucción.1> ;

<instrucción.2> ;

.....

.....

<instrucción.N> ;

until <condición>;

Aquí las palabras repeat y until sirven también como delimitadores de bloque.

WHILE−DO

La estructura repetitiva while(mientras) es aquella en la que el número de iteraciones no se conoce por anticipado y el cuerpo del bucle se ejecuta repetidamente mientras que una condición sea verdadera .

Su formato es :

while <condición> do

begin

<instrucciones>;

end;

Acumulador

Es una variable que nos permite guardar un valor que se incrementa o decrementa en forma no constante durante el proceso, en un momento determinado tendrá un valor y al siguiente tendrá otro valor igual o distinto, por ejemplo cuando se realiza un deposito en el banco, la cantidad depositada no siempre es la misma; unas veces será una cantidad y otras veces distintas. Lo mismo ocurre cuando realizamos algún retiro, pero decrementando la cantidad total.

Bifurcación

Se utilizan para saltar de un lugar a otra zona del programa o código si es que se cumple una cierta condición.

Interruptor

Es un campo de memoria que toman diversos valores a lo largo de la ejecución del programa y que permite comunicar información de una parte a otra del mismo, es decir variar la secuencia de ejecución de un programa, dependiendo el valor de el valor que tenga en cada momento. Los únicos valores que quede tomar es encendido y apagado.

Ruteador

Es un dispositivo diseñado para segmentar la red. Opera en la capa 3 de modelo OSI y permite un control Sowftware sobre los paquetes puede proporcionar seguridad, control, servicio de firewall y un acceso a una WAN (por ejemplo con RSDI, Frame relay, ATM, MODEM).

Al funcionar en una capa mayor que la del switch, el ruteador distingue entre los diferentes protocolos de red tales como IP, IPX, apple talk. Esto le permite hacer una decisión mas inteligente que el switch al reenviar los paquetes permitiendo también unir redes que funcionan con protocolos diferentes.

Es responsable de crear y mantener tablas de ruteo para cada capa del protocolo de red, estas tablas son creadas de forma estática o bien dinámica de esta manera el ruteador extrae de la capa de red la dirección destino y realiza una decisión de envío basado en el contenido de la especificación del protocolo en la tabla de ruteo.

Paquete de datos

Es la unidad básica de transmisión de datos en una red, cada paquete tiene información de un recorrido en la red y la información entra de un dispositivo a otro

Swich

Es un dispositivo diseñado para resolver problemas de rendimiento en la red. Debidos a anchos de banda pequeños y embotellamientos. Opera en la capa 2 del modelo OSI y reenvía los paquetes en base a la dirección MAC

TCP PACKET

Es un protocolo de red independiente del nivel físico y que soporta múltiples sesiones entre múltiples ordenadores. Esta constituido en capas lo que permite adaptarlo a nuevas tecnologías y requerimientos sin necesidad de modificar el conjunto.

ICMP Ping packet

Sirve para indicar errores en el proceso de transmisión de datos como si fuera una ayuda

UDP packet

Provee dos servicios que no incluye la capa IP: provee los números de puerto para distinguir las distintas solicitudes del usuario y una suma para verificar que los datos arriben correctamente las aplicaciones que usan este protocolo deben asegurarse que reciban los data gramas en el orden apropiado

Ping of death

Ataque de tipo dos usado cuando un atacante envia paquetes ping mayores a los 65,536 BYTES permitidos por el protocolo TCP/IP; el ataque provoca que los servidores se congelen o se reinicien.

Router Switch

Sirve para eliminar la necesidad de comprar HUD y para que la red este completamente dedicada teniendo un backbone duplex

ARREGLOS (VECTORES Y MATRICES)

Un arreglo es una posición de memoria que podría considerarse como una plantilla de una hoja de calculo en la que en cada una de las celdas de la plantilla puede haber almacenados distintos valores, los cuales se diferencian unos de otros por su ubicación. En otras palabras un arreglo es una especie de variable que contiene muchos valores pero cada uno con una posición diferente.

Un arreglo puede ser unidimensional o vectorial, bidimensional o matricial, o multidimencional.

VECTORES O ARREGLOS UNIDIMENCIONALES :

Su sintaxis es:

A[i]

Se lee: " A sub i " , donde i es cualquier numero entero positivo

diferente de cero y representa la ubicación de un valor dentro del

vector A.

Gráficamente se vería así :

Ú ¿

³
ÚÄÄÄÂÄÄÄ¿
³

A ³ ³ 5 ³ 7 ³ 2 ³ 9 ³ 3 ³ 8 ³ 6 ³ 1 ³ 4 ³ 0 ³ . ³ . ³ . ³ . ³ . ³ X ³ ³

³
ÀÄÄÄÁÄÄÄÙ
³

³ ³

À i i+1 i+2 i+3 i+4 i+5 i+6 i+7 i+8 i+9 ..... i+N Ù

En este caso vemos por ejemplo que A[7] "A sub 7" tiene almacenado el

valor 4 o escrito de otra forma: A[7] = 4

Y asi mismo:

A[1] = 5

A[2] = 7

A[3] = 2

A[4] = 9

A[5] = 3

: :

: :

A[i+N] = X

Un ejemplo podría ser:

inicio

leer n

para e de 1 hasta n hacer

imprimir "teclee la edad de la persona",e

leer edad

L[e] edad

fin para

para i de 1 hasta n hacer

imprimir "la edad de la persona",i," es: ",L[i]

fin para

fin

Este programa sirve para preguntarle la edad a un numero n de

personas y una vez terminada el ingreso de todos datos, se imprime

una lista con la edad de cada persona.

MATRICES O ARREGLOS BIDIMENCIONALES:

Su sintaxis es:

M[F,C]

Se lee: " A sub F coma C " , donde F y C son dos números cualquiera,

enteros, positivos, diferentes de cero, lo cual da como resultado una

pareja ordenada que representa la ubicación de un valor dentro de

la matriz M.

Donde: El valor de F representa las Filas y C las Columnas

[F,C]

³ ³

³ À Columnas

À Filas

Gráficamente se vería así :

Ú C C+1 C+2 ¿

³ ³

³ ÚÄÄÄÂÄÄÄÂÄÄÄ¿ ³

³ F ³ 5 ³ 7 ³ 2 ³ ³

³ ÃÄÄÄÅÄÄÄÅÄÄÄ´ ³

A ³ F+1 ³ 8 ³ 3 ³ 9 ³ ³

³ ÃÄÄÄÅÄÄÄÅÄÄÄ´ ³

³ F+2 ³ 4 ³ 6 ³ 1 ³ ³

³ ÀÄÄÄÁÄÄÄÁÄÄÄÙ ³

À Ù

En este caso vemos por ejemplo que A[3,2] " A sub 3 coma 2 " tiene

almacenado el valor 6 o escrito de otra forma: A[3,2] = 6

Y así mismo:

A[1,1] = 5

A[1,2] = 7

A[1,3] = 2

A[2,1] = 8

A[2,2] = 3

A[2,3] = 9

A[3,1] = 4

A[3,2] = 6

A[3,3] = 1

Un ejemplo podría ser:

Inicio

imprimir "Teclee el numero de personas que desea registrar"

leer n

para p de 1 hasta n hacer

imprimir "teclee el código de la persona",p

leer codi

imprimir "teclee la edad de la persona",p

leer edad

imprimir "teclee el teléfono de la persona" ,p

leer tele

R[p,1] código

R[p,2] edad

R[p,3] tele

Fin para

para i de 1 hasta n hacer

imprimir "el código de la persona",i," es: ",R[i,1]

imprimir "la edad de la persona",i," es: ",R[i,2]

imprimir "el teléfono de la persona" ,i," es: ",R[i,3]

fin para

fin

Este programa sirve para registrar un numero n de personas con los

datos de código, edad y teléfono y una vez terminada el ingreso de

datos, se imprime una lista con la información de cada persona.

ARREGLOS MULTIDIMENCIONALES :

Los arreglos multidimencionales son aquellos que para determinar la

ubicación de un valor almacenado dentro de ellos, se requiere de mas

de dos términos independientes como en la J y la I en los arreglos

bidimensionales, por lo tanto un arreglo multidimencional podría ser:

T[i,j,k], el cual seria de tres dimensiones o términos independientes

Para esta clase de arreglos no ampliare mas el tema pues no veo necesario tocarlo mas profundamente, pues esto es solo un cursillo de lógica de programación y para aquel que ya esta en capacidad de usar arreglos multidimencionales, este curso ya no será de su necesidad.

Diagrama de flujo

Es un cuadro que explica paso por paso la secuencia de un objeto o programa

Algoritmo

Es una serie de operaciones detallistas y no ambiguas, a ejecutar paso por paso y que conducen a la resolución de un problema, es un conjunto de reglas para resolver un problema

Programación

Conjunto de actividades y operaciones realizados por el personal inform. Tendientes a instruir a la maquina para que pueda realizar las funciones previstas en el algoritmo

Estructura de datos

Es una colección de datos relacionados entre si por ejemplo: cadena, tablas, árboles, pilas.

Sentencia: la instrucción para realizar uno o mas problemas.

Compilar: traduce todos los códigos fuente de un programa

Interprete: procesa los programas escritos con lenguaje de alto nivel

Declaración: declarar variables o constantes

Ejecutar: realizar una instrucción o realizar un programa

Acumulador

Es una variable que nos permite guardar un valor que se incrementa o decrementa en forma no constante durante el proceso, en un momento determinado tendrá un valor y al siguiente tendrá otro valor igual o distinto, por ejemplo cuando se realiza un deposito en el banco, la cantidad depositada no siempre es la misma; unas veces será una cantidad y otras veces distintas. Lo mismo ocurre cuando realizamos algún retiro, pero decrementando la cantidad total.

Bifurcación

Se utilizan para saltar de un lugar a otra zona del programa o código si es que se cumple una cierta condición.

Interruptor

Es un campo de memoria que toman diversos valores a lo largo de la ejecución del programa y que permite comunicar información de una parte a otra del mismo, es decir variar la secuencia de ejecución de un programa, dependiendo el valor de el valor que tenga en cada momento. Los únicos valores que quede tomar es encendido y apagado.

Ruteador

Es un dispositivo diseñado para segmentar la red. Opera en la capa 3 de modelo OSI y permite un control Sowftware sobre los paquetes puede proporcionar seguridad, control, servicio de firewall y un acceso a una WAN (por ejemplo con RSDI, Frame relay, ATM, MODEM).

Al funcionar en una capa mayor que la del switch, el ruteador distingue entre los diferentes protocolos de red tales como IP, IPX, apple talk. Esto le permite hacer una decisión mas inteligente que el switch al reenviar los paquetes permitiendo también unir redes que funcionan con protocolos diferentes.

Es responsable de crear y mantener tablas de ruteo para cada capa del protocolo de red, estas tablas son creadas de forma estática o bien dinámica de esta manera el ruteador extrae de la capa de red la dirección destino y realiza una decisión de envío basado en el contenido de la especificación del protocolo en la tabla de ruteo.

Paquete de datos

Es la unidad básica de transmisión de datos en una red, cada paquete tiene información de un recorrido en la red y la información entra de un dispositivo a otro

Swich

Es un dispositivo diseñado para resolver problemas de rendimiento en la red. Debidos a anchos de banda pequeños y embotellamientos. Opera en la capa 2 del modelo OSI y reenvía los paquetes en base a la dirección MAC

TCP PACKET

Es un protocolo de red independiente del nivel físico y que soporta múltiples sesiones entre múltiples ordenadores. Esta constituido en capas lo que permite adaptarlo a nuevas tecnologías y requerimientos sin necesidad de modificar el conjunto.

ICMP Ping packet

Sirve para indicar errores en el proceso de transmisión de datos como si fuera una ayuda

UDP packet

Provee dos servicios que no incluye la capa IP: provee los números de puerto para distinguir las distintas solicitudes del usuario y una suma para verificar que los datos arriben correctamente las aplicaciones que usan este protocolo deben asegurarse que reciban los data gramas en el orden apropiado

Ping of death

Ataque de tipo dos usado cuando un atacante envia paquetes ping mayores a los 65,536 BYTES permitidos por el protocolo TCP/IP; el ataque provoca que los servidores se congelen o se reinicien.

Router Switch

Sirve para eliminar la necesidad de comprar HUD y para que la red este completamente dedicada teniendo un backbone duplex

BIBLIOGRAFIA

www.laopinion.com/glossary/d.html

http://encarta.msn.es/find/Concise.asp?z=1&pg=2&ti=761574357

http://www.terra.es/personal/ffrrbb/elrincondelpascal

http://www.recursos−as400.com/edep.shtml

Biblioteca Microsoft encarta2002

Enciclopedia Lauruse

Biblioteca Microsoft encarta 2000

119

INICIO

N1,N2

N1,N2:INTEGER

T:REAL

N1,N2

T

T:=N1+N2

FIN

INICIO

N,A:STRING

N,A

N,A

N,A

FIN

INICIO

TC = 9.80

P,D:REAL

P

P

D:=P/TC

D

FIN

INICIO

C1,C2,P:REAL

C1,C2

C1,C2

P:= (C1+C2)/2

P

FIN

INICIO

N,AP,AM,D,P:STRING

A,E:INTEGER

N,AP,AM,D,P,A

N,AP,AM,D,P,A

E:=2002−A

N,AP,AM,D,P,A,E

INICIO

INICIO

NDE,RFC,D,P,PR:STRING

SH,DT,SD,SS,SQ:REAL

HT:INTEGER

NDE,RFC,D,P,PR,SH,DT,HT

NDE,RFC,D,P,PR,SH,DT,HT

SD:=SH*HT

SS:=SD*DT

SQ:=SS*2

NDE,RFC,D,P,PR,SD,SS,SQ

FIN

INICIO

N,D,T:STRING

N,D,T

N,D,T

N,T,D

FIN

INICIO

N1.N2,P:REAL

N1,N2

N1,N2

T:=N1*N2

T

FIN

INICIO

R,M:REAL

CM=100

M

M

R:=M*CM

R

FIN

FIN

R

R:=MI*KM

MI

MI

KM=1.609

R,MI:REAL

INICIO

INICIO

S,EX1,EX2 :REAL

EX1,EX2

EX1,EX2

S:=EX1+EX2

IF S>=80 THEN

RECHAZADO

ACEPTADO

FIN

OP

OP

INICIO

C,T,CM1,V:REAL

V,C

V,C

T:=V*C

T>1500

CM1:=T*(20/100)

CM1:=T*(10/100)

CM1

CM1

FIN

FIN

N,S,A,P

A

SE ELABORA BOLETA

NO SE ELABORA BOLETA

A=>80

P=>7

A

N,S,A,P

N,S,A,P

S,N:INTEGER; A,P:REAL

INICIO

INICIO

N:STRING

E,P:REAL

N,E,P

N,E,P

P>6

P<8

E>6

E<8

E=>8

P=>8

POBLA−CION A

POBLA−CION B

POBLA−CION C

FIN

OP=1

OP=3

OP=2

OP=4

ADIOS

C=PI(SQR(R))

CU=B*AL

T=(B*AL)/2

FIN

NUM ES 0

NUM ES NEG

NUM ES +

N<0

N>0

N

N

N:REAL

INICIO

INICIO

N1,N2,N3:REAL

N1,N2,N3

N1,N2,N3

A

A

N1>N2

N1>N3

N2>N1

N2>N3

N3>N1

N3>N2

N1=N2

N2=N3

N1=N2

N1>N3

N1=N3

N1>N2

N1 ES MAYOR

N2 ES MAYOR

N3 ES MAYOR

SON IGUAL

N1,N2 IGUAL

N1,N3 IGUAL

N2,N3 IGUAL

FIN

ERROR

T

C

CU

ADIOS

FIN

INICIO

A,B:INTEGER S,R,M,D:REAL

OP:CHAR

A,B

A,B

S=SUMA

R=RESTA

M=MULTIPLI

D=DIVISION

OP

OP

A

A

OP=S

OP=R

OP=M

OP=D

S:=A+B

R:=A−B

M:=A*B

D:=A/B

ERROR

S

R

M

D

FIN

1=TRIAN

2=CIRCULO

3=CUADRO

4=SALIR

B,AL,R

B,AL,R

A,T,C,CU,B,AL,R,OP:REAL

INICIO

PI=3.1416

INICIO

SD,SH,HT,TC,P,D,E,A:REAL

1−SUELDO

2−EDAD

3−PESO−DLL

4−SALIR

OP

OP

A

A

OP=1

OP=2

OP=3

OP=4

SH,HT

SH,HT

SD=SH*HT

SD

```
A

A

E=2002−A

E

TC,P

TC,P

DLL=P/TC

DLL

ADIOS

ERROR

FIN

INICIO

N,R,CU,C:REAL

OP,RESP:CHAR

1−CUADRADO

2−RAIZ

3−COSENO

N,OP

N,OP

CASE OP OF

ELSE

CU:=SQR(N)

R:=SQRT(N)

C:=COS(N)

SALIDA

CU
```

R

C

DESEA SEGUIR CAPTURANDO

RESP

RESP

UNTIL R<>'S'

FIN

A

A

INICIO

N1,N2,N3,N4,T:REAL

RESP:CHAR

OP:INTEGER

1−SUMA

2−MULTIPLICAR

3−SALIR

OP

OP

CASE OP OF

ELSE

N1,N2, N3,N4

N1,N2, N3,N4

T:=N1+N2+N3+N4

T

N1,N2, N3,N4

N1,N2, N3,N4

```
T:=N1*N2*N3*N4

T

ADIOS

ERROR

FIN

INICIO

Y=0.10

Z=0.20

E=15

N,C:STRING

NUEM,OP:INTEGER

D,X,DES,HT,SH,SQ,SD:REAL

N,NUEM,C

1− 6 A 8 FALTAS

2− 1 A 5 FALTAS

OP

OP

CASE OP OF

A

A

HT,SH

HT,SH

SQ:=SH*HT*E

D:=SQ*Z

SD:=HT*SH

DES:=SQ−D
```

N,C,SD,DES

N,C,SD,DES

SQ:=SH*HT*E

X:=SQ*Y

SD:=HT*SH

DES:=SQ−X

HT,SH

HT,SH

DESEA SEGUIR CAPTURANDO

UNTIL RESP<>'S'

PRESIONE ENTER

FIN

B

B

INICIO

RESP:CHAR

NOM:STRING

PRE,DES,TOT:REAL

NOM,PRE

NOM,PRE

IF PRE<2000

COD D

DES:=PRE*0.05

TOT:=PRE−DES

DES:=PRE*0.05

TOT:=PRE−DES

COD D

IF PRE<2000

DES:=PRE*0.05

TOT:=PRE−DES

COD D

IF PRE<2000

DES:=PRE*0.05

TOT:=PRE−DES

COD D

IF PRE<2000

ERROR

DESEA CONTINUAR

NOM,PRE,DES,TOT

NOM,PRE,DES,TOT

NOM,PRE,DES,TOT

NOM,PRE,DES,TOT

RESP

RESP

UNTIL RESP='S'

FIN

A

A

INICIO

NOM:STRING

VTA,TOT,PV:REAL

CV,CE:INTEGER

CE:=1

WHILE CE<=4 DO

NOM,VTA

NOM,VTA

UNTIL CV=3

CV:=0

TOT:=0

CV:=CV+1

PV:=TOT*0.05

NOM,TOT,PV

CE:=CE+1

FIN

A

A

A

A

FIN

CA=CA+1

N,S,M,P

P:=TOT/3

TOT=X+TOT

C=C+1

X:=0

TOT:=0

UNTIL C=2

N,M,S,X

N,M,S,X

WHILE CA<=2 DO

CA:=1

N,S,M:STRING

P,TOT:REAL

X,CA,C:INTEGER

INICIO

INICIO

CC:INTEGER

M,C,MAR,P:STING

PRE,T,TOT,IMP:REAL

CC=1

T=0

M,C,MAR,PRE,P

M,C,MAR,PRE,P

T=T+PRE

CC=CC+1

UNTIL CC<=3

A

IMP:=T*0.15

TOT:=T+IMP

T,IMP,TOT

PRESIONE ENTER PARA SALIR

FIN

INICIO

CP,CC,CA:INTEGER

```
P,T,TOT,ST:REAL
MEM,C,NP,CLA
UNTIL
CC<=2
MEM,C
MEM,C
NP,CA,CLA,P
NP,CA,CLA,P
IF CA=1
IF CA=2
IF CA=3
TOT:=P*0.3
ST:=P−TOT
TOT:=P*0.2
ST:=P−TOT
TOT:=P*0.1
ST:=P−TOT
ERROR
T=T+ST
CP=CP+1
UNTIL CP=4
A
A
T
CC=CC+1
UNTIL CC=2
```

```
PROGRAMA HA CONCLUIDO

FIN

B

B

INICIO

NOM:STRING

PRE,T,TOT,DES:REAL

CA:INTEGER

CA:=1

WHILE CA<=8

PRE,NOM

PRE,NOM

TOT=TOT+PRE

CA:=CA+1

IF TOT>=2000 THEN

TOT

DES=TOT*.1

T=TOT−DES

TOT,DES,T

FIN

INICIO

LUN=.05,MAR=.07,MIE=.08,JUE=.09,VIE=.1,PAN=30,
CHA=50,BLU=20,CAM=25,VES=40,FAL=60,SAC=80

T,ST,DES,PAN1,BLU1,CAM1,VES1,SAC1,FAL1:REAL

RESP:CHAR

OP:INTEGER

PAN1,CHA1,BLU1,CAM1,VES1,SAC1,FAL1
```

PAN1,CHA1,BLU1,CAM1,VES1,SAC1,FAL1

ST=(PAN*PAN1)+(CHA*CHA1)+(BLU*BLU1)+(CAM+CAM1)+(VES*VES1)+(FAL*FAL1)+(SAC*SAC1

LUNES − 1

MARTES − 2

MIERCOLES − 3

JUEVES − 4

VIERNES − 5

OP

OP

CASE OP OF

DES=LUN*ST

T=ST−DES

DES=LUN*ST

T=ST−DES

DES=LUN*ST

T=ST−DES

DES=LUN*ST

T=ST−DES

DES=LUN*ST

T=ST−DES

PAN1,CHA1,BLU1,CAM1,VES1,FAL1,SAC1,ST,DES,T

DESEA SEGUIR CAPTURANDO?

FIN

A

A

INICIO

```
NOM:STRING
SUM,CAL,PROM:REAL
CCAL:INTEGER
NOM
NOM
SUM:=0
CCAL:=1
WHILE CCAL<=3 DO
CAL
CAL
SUM:=SUM+CAL
CCAL:=CCAL+1
PROM=SUM/3
NOM,PROM
PRESIONE ENTER PARA SALIR
FIN
INICIO
NIM,SEG:REAL
RESP:CHAR
QUIERE CONVERTIR MINUTOS A SEGUNDOS
WHILE RESP='S'
RESP
RESP
MIN
MIN
SEG=60*MIN
```

DESEA SEGUIR CAPTURANDO

FIN

A

A

B

B

INICIO

N,NOM,HECTOR:STRING

NOM

NOM

N:=NOM

UNTIL NOM:=HECTOR

FIN

A

A

INICIO

NUM,I:INTEGER

R:REAL

NUM:=0

FOR I:=1 TO 10 DO

NUM=NUM+1

R:=SQRT(NUM)

FIN

R

RES

FIN

RES:=2* I

CU

FOR I:=1 TO 10 DO

NUM:=0

NUM,I:INTEGER

RES:REAL

INICIO

FIN

CU:=SQR(NUM)

NUM=NUM+1

FOR I:=1 TO 10 DO

NUM:=0

NUM,I:INTEGER

CU:REAL

INICIO

A

FIN

A

I

FOR I:=1 TO 100 DO

NUM:=0

I:INTEGER

INICIO

R

FIN

R:=NUM* I

```
NUM
FOR I:=1 TO 10 DO
NUM
I:INTEGER
NUM,R:REAL
INICIO
RES
FIN
RES=1/I
I
FOR I:=1 TO 25 DO
RES
I:INTEGER
RES:REAL
INICIO
FIN
NUM=NUM+1
RES:=SQR(NUM)
FOR I:=1 TO 20 DO
IF Y=26 THEN
I:INTEGER
NUM,RES:REAL
INICIO
FIN
Y:=Y+1
GOTO (X,Y); NUM
```

```
FOR I:=1 TO 500 DO

X:=1

Y:=1

NUM,X,Y:INTEGER

INICIO

X:=X+Y

Y:=1

P

FIN

P:=X*J

X:=X+1

I

FOR I:=1 TO 2 DO FOR J:=1 TO 10 DO

X:=2, P:=2

X,P,I,J:INTEGER

INICIO

PROM

FIN

PROM:=TOT/4

TOT:=TOT+CAL

FOR I:=1 TO 4 DO

TOT:=0

I:INTEGER

CAL,TOT:REAL

INICIO

CAL
```

CAL

A

A

Compilador

pascal

Programa fuente en pascal

Computadora

Programa

Objeto

(maquina)

Programa

Objeto

(Maquina)

Resultados del programa pascal

Computadora

Datos necesarios para programa pascal