Temas de Intro II

1.- Archivos

2.- String, Enumerado, Subrango y Registro

3.- Punteros y Listas

4.- Listas y Listas de Listas

5.- Recursión

6.- Árboles

Tipo String

Es una secuencia de caracteres (cadena) de longitud variable.

Var Palabra: string (long. Variable con un máximo de 255)

Var Palabra: string[10] (long. Variable con un máximo de 10)

String

Var nombre: string;

….

nombre:= ´Jorge´ ;

Operador:

El operador '+' sirve para concatenar Strings.

Ejemplo:

```
. . .
ApellidoCliente := 'PEREZ';
NombreCliente := ApellidoCliente + ', ' + 'Juan';
Write(NombreCliente);          →      PEREZ, Juan
. . .
```

Operadores relacionales: (< <= > >= = <>)

Algunos ejemplos:

'a' < 'b' → True
'aa' < 'b' → True
'a' < 'ab' → True
'Z' < 'a' → True
'b' < 'ba' → True

Alguna funciones con String….

Length(´abcde´) 5

Copy(´1234567´, 3, 2) ´34´

```
program Ejemplo;
const
   max=5;
type
  palabras= array[1..max] of string;
var
  Apellido, nombre, nomyape: palabras;
  i: integer;
begin
  for i:=1 to max do  begin
     writeln ('Ingrese apellido');
     readln(apellido[i]);
     writeln ('Ingrese nombre');
     readln(nombre[i])
  end;
  for i:=1 to max do     nomyape[i]:= nombre[i] + ' ' + apellido[i];
  for i:=1 to max do     writeln (nomyape[i])
end.
```

¿ Qué hace este programa?

Tipo Enumerado

- Tipo básico de Pascal. (también llamado escalar)

- Permite contener el valor de un identificador dentro de un conjunto de identificadores ordenados.

Ejemplo:
Var Dia : (Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo);

Tipo Enumerado

var Dia : **(Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo);**

0 1 2 3 4 5 6

Dia := Miércoles; {toma valor de Miércoles con orden 2}

Dia := Succ(Dia); {toma el valor de Jueves con orden 3}

Dia := Pred(Dia); {toma el valor de miércoles con orden 2}

Write (Ord(Dia)); {Imprime un valor 2}

Write (Dia); { ERROR!}

Read (Dia); { ERROR!}

Dia := Domingo; {toma valor de Domingo con orden 6}

Dia :=Succ(Dia); { ERROR!}

var Dia : **(Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo);**

0 1 2 3 4 5 6

If (Dia < Sabado) then …… { OK }

For Dia := Lunes to Viernes do { OK }

While Dia < Domingo do begin { OK }

Dia := Succ(Dia);

……

Var Dia : (Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo);

~~DiaHabil : (Lunes, Martes, Miércoles, Jueves, Viernes);~~ **{ ERROR!}**

Mes : (Enero, Febrero, Marzo, Abril, Mayo … , Diciembre);

Dia := Sábado; {OK}

Dia := sábado; {OK}

Mes := Dia; { ERROR ! }

Type Boolean

- ES UN ENUMERATIVO

Type Boolean = (false, true);

La mayor ventaja del tipo ENUMERATIVO es el aumento de la legibilidad del texto del programa.

Ejercicio

Defina un tipo MESES para definir los meses del año.

a) Realice un procedimiento que imprima los meses desde un mes MES1 hasta un MES2

b) Realice una función que retorne verdadero si MES1 es anterior a MES2

c) Realice un procedimiento que a partir del contenido de una variable de tipo MESES, imprima por pantalla la descripción del mes

```
Program MESES;
Type
 nombreMESES = ( ene, feb, mar, abr, may, jun, jul, ago, sep, oct, nov, dic);

 procedure ImprimirNombreMes (indiceMes:nombreMESES);
 begin
 case  indicemes of
   ene:  Writeln('enero');
   feb:  Writeln('febrero');
   mar:  Writeln('marzo');
   abr:  Writeln('abril');
  may:  Writeln('mayo');
   jun:  Writeln('junio');
   jul:  writeln('julio');
   ago:  writeln('agosto');
   sep:   writeln ('septiembre');
   oct:   writeln ('octubre');
   nov:   writeln ('noviembre');
   dic:   writeln ('diciembre');
end;
end;
```

```
procedure ImprimirRangoMeses (Mes1, mes2: nombreMeses);
var
   indiceMes: nombreMESES;
begin
    For indiceMes:= MES1 to mes2 do
       imprimirNombremes(indiceMes);
end;

Function EsAnterior (Mes1, mes2: nombreMeses): boolean;
begin
   if mes1<mes2
   then esanterior:= true
   else esanterior:=false
end;

var
  mes1, mes2: nombreMESES;
begin
mes1:= ene;
mes2:= sep;
…….
end.
```

Tipo Subrango

- Tipo de datos basado en un Tipo Asociado (tipo base) que pueden ser Enumerativo, Char e Integer. Representa un rango de ese tipo, es decir una sucesión de valores consecutivos.

- Las operaciones válidas dependen del Tipo base.

Tipo Subrango

1. Definir el tipo Básico a usar
2. Sintaxis.

TYPE

<tipo subrango> = <cota inferior> **'..'** <cota superior>

Donde las cotas son constantes de igual tipo base y
cotainferior < cotasuperior

Ejemplos:

Notas: 0**..**10 (Tipo Base: Integer)

MesesVerano: ene**..**feb (en base al TYPE enumerativo nombreMESES

primerasLetras: 'a'..'f' (Tipo Base: char)

Ventajas

Un Subrango ofrece legibilidad y resguardo adicional de protección contra asignaciones de valor fuera de rango, y por lo tanto ayuda en la detección de errores

```
var
mesesPrimercuatrimestre: mar..jun;
mesesSegundocuatrimestre:  jul..nov;

begin
   mesesPrimercuatrimestre := abr;
   mesesSegundocuatrimestre := mar;
end.
```

¿Són válidas las sentencias?

```
var
mesesPrimercuatrimestre: mar..jun;
mesesSegundocuatrimestre:  jul..nov;

begin
   mesesPrimercuatrimestre := abr;                    {Ok}
   mesesSegundocuatrimestre := mar;                   { ERROR}
end.
```

	Simple	Estructurado Homogéneo	Estructurado Heterogéneo
Integer, char, Boolean, Real, Enumerado, Subrango	√		
Pila, Fila		√	
Array, File		√	

	Simple	Estructurado Homogéneo	Estructurado Heterogéneo
Integer, char, Boolean, Real, Enumerado, Subrango	√		
Pila, Fila		√	
Array, File		√	
Record			√

Tipo Registro

Tipo de Dato estructurado para almacenar información heterogénea.

Ejemplo:

```
Type  Alumno = record
    nombre:  string[30];
    apellido: string[30];
    DNI: Integer
End;
```

CAMPO

TIPO

IDENTIFICADOR

Cada Elemento (“Campo”) puede ser diferente, por lo tanto para cada uno se brinda su “Identificador” y Tipo.

Ejemplo 1

```
Type Fecha = record
        Dia : Integer;
        Mes : Integer;
        Anio : Integer
End;

Var
FechaVencimiento : Fecha;
Begin
FechaVencimiento := 4/10/2008; { ERROR ! }
FechaVencimiento.Dia := 4; {OK }
FechaVencimiento.Mes := 10; {OK }
FechaVencimiento.Anio := 2008; {OK }
. . .
FechaVencimiento.Dia := FechaVencimiento.Dia + 1; { OK}
```

Ejemplo 2

```
Type Fecha = record
        Dia : 1..31;
        Mes : 1..12;
        Año : 1950..2050
End;

Type TipoFactura = record
        Emision : Fecha;
        Cliente : Integer;
        Importe : Real;
        Impuestos : Real;
        Total : Real;
        Vencimiento : Fecha
End;

Var

unaFactura : TipoFactura;
……
```

SUBRANGO

RECORD

¿Cómo se asignaría el día de Emisión de la variable unaFactura?

Ejemplo 2

```
Type Fecha = record
        Dia : 1..31;
        Mes : 1..12;
        Año : 1950..2050
End;

Type TipoFactura = record
        Emision : Fecha;
        Cliente : Integer;
        Importe : Real;
        Impuestos : Real;
        Total : Real;
        Vencimiento : Fecha
End;

Var

unaFactura : TipoFactura;
……
Begin
```

unaFactura.emisión.Día := 4

Dentro de un registro se pueden utilizar:		
Integer, char, Boolean, Real, Enumerado, String Subrango		SI
Pila, Fila(*)	NO	
Array		SI
File	NO	
Record		SI

UN REGISTRO se puede utilizar dentro de		
Integer, char, Boolean, String, Real, Enumerado, Subrango?	NO	
Pila, Fila (*)	NO	
Record?		SI
Array, **File?**		SI

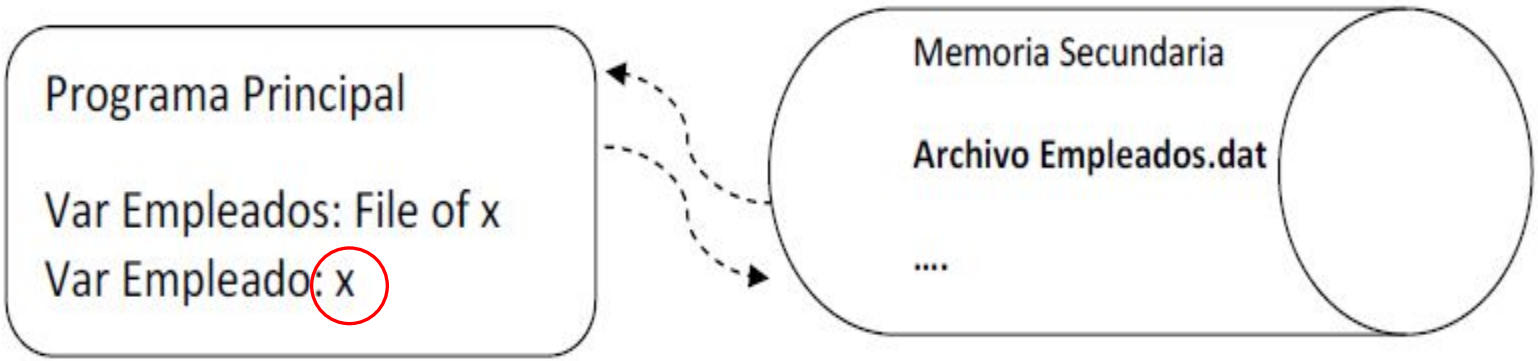
Ejercicio

Crear un archivo desordenado de AlumnosINTRO con datos de 10 alumnos con la siguiente información: DNI, Nro. Libreta, Nombre y Apellido, comisión, PromedioCarrera.

Manejo de Archivos

4) Uso

```
Read (Empleados, Empleado);
```




```
Write (Empleados, Empleado);
```

X ahora es de tipo Record

```
program ArchivoAlumnos;

type
  Alumno= Record
          DNI:   integer;
     NroLibreta: integer;
     NombreApellido: string[30];
     Comision: string [30];
     PromedioCarrera: Real;
     end;

  AlumnosIntro= File of Alumno;
```

```
.....
{procedures}.

var
   alu: alumno;
   Arc_Alu: AlumnosIntro;
   i:integer;

Begin
   { inicialización y apertura}
   assign (Arc_Alu, '/ip2/Apellido-AlumnosIngresantes.dat');
   rewrite (Arc_Alu);
   for i:=1 to 10 do
     begin
       leerAlumno(alu)
       write(Arc_Alu, alu);
     end;
  close(Arc_Alu);
end.
```

```
……..
Procedure leerAlumno(var alu: Alumno):
Begin
        readln(alu.DNI);
        readln(alu.NroLibreta);
        readln(alu.NombreApellido);
        readln(alu.comision);
         readln(alu.PromedioCarrera);
end
……...
```

Ejercicio

Hacer un módulo que dado un DNI imprima todos los datos del alumno si existe en el Archivo AlumnosINTRO

```
program ArchivoAlumnos;

{ módulo que dado un DNI imprima todos los datos del  alumno si
    existe en el Archivo AlumnosINTRO }

type
  Alumno= Record
    DNI: integer;
      NroLibreta: integer;
      NombreApellido: string[30];
      Comision: string [30];
      PromedioCarrera: Real;
 end;
 AlumnosIntro= File of Alumno;

var
  alu: alumno;
  Arc_Alu: AlumnosIntro;
  unDNI:integer;
```

```
.....
Begin

   {inicialización y apertura_ Supongo que existe!!!!!}

   assign (archivo, '/ip2/Apellido-AlumnosIngresantes.dat');
   reset(Arc_Alu);
   Read(Arc_Alu, alu);
   Read (unDNI);
    While (not eof(arc_Alu)) and (alu.DNI <> unDNI) do
        Read(Arc_Alu, alu);
    if alu.DNI = unDNI
           then writeln (alu.DNI) {   Imprimir todos los datos}
      else writeln ('no existe')
end.
```