

PROGRAMACIÓN I

Temas

- ✓ Tipos de datos definidos por el usuario
- ✓ Subrango – Definición - Ejemplos
- ✓ Conjunto – Definición - Ejemplos
- ✓ String – Definición - Ejemplos

Introducción



Tipos de datos definidos por el usuario




Tipos de datos estándares

Hasta ahora
vimos

Tipos de
datos
estándares



- El conjunto de valores de ese tipo
- Las operaciones que se pueden efectuar
- Su representación



Están
definidas y
acotadas por
el lenguaje.

Tipos de datos definidos por el usuario

Un aspecto muy importante en los lenguajes de programación es la **capacidad de especificar y manejar datos no estándar**, indicando valores permitidos, operaciones válidas y su representación interna, en algunos casos.

Tipos de datos definidos por el usuario

Ventajas de contar con Tipos de Datos definidos por el usuario

- **Aumento de la riqueza expresiva del lenguaje**, con mejores posibilidades de abstracción de datos.
- **Mayor seguridad** respecto de las operaciones que se realizan sobre cada clase de datos.
- **Límites preestablecidos sobre los valores posibles** que pueden tomar las variables que corresponden al tipo de dato.

Tipos de datos definidos por el usuario

Un **tipo de dato definido por el usuario** es aquel que no existe en la definición del lenguaje, y el programador es el encargado de su especificación.



¿Cómo definimos un tipo de datos?

Tipos de datos definidos por el usuario

Program ejemplo1;

Type

nuevoTipo = ...;

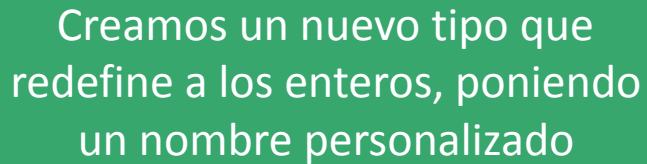
Var

valor1, valor2: nuevoTipo;

Begin

...

End.



Creamos un nuevo tipo que redefina a los enteros, poniendo un nombre personalizado




































Tipos de datos definidos por el usuario

VENTAJAS

Flexibilidad: en el caso de ser necesario modificar la forma en que se representa el dato, sólo se debe modificar una declaración en lugar de un conjunto de declaraciones de variables.

Documentación: se pueden usar como identificador de los tipos, nombres autoexplicativos, facilitando de esta manera el entendimiento y lectura del programa.

SUBRANGOS

REINO ANIMALIA (ANIMALES) >1.000.000 especies	     
PHYLUM CHORDATA (CORDADOS) 40.000 especies	     
CLASE AVES 8.600 especies	     
ORDEN PASERIFORMES (AVES CANORAS) 5.160 especies	     
FAMILIA ESTRILDIDAE 142 especies	     
GENERO POEPHILA 3 especies	  
ESPECIE <i>Poephila acuticauda</i>	
SUBESPECIE <i>Poephila acuticauda hecki</i>	

Motivación

- ¿Cómo hago para representar el mes de nacimiento de una persona?
- ¿Qué tipo de datos utilizo para representar el año de nacimiento de un alumno?

Motivación

var

mes, dia, diaSem, año : integer;

begin

año := 1997;

mes := 5;

dia := 7;

diaSem := 3;

...

end.

¿PROBLEMAS?

Tipos de datos definidos por el usuario- SUBRANGO

CARACTERÍSTICAS

- Consiste en una sucesión de valores de un tipo ordinal tomado como base.
- Existe en la mayoría de los lenguajes.
- Es un tipo de datos simple.
- Es un tipo de datos ordinal.

Tipos de datos definidos por el usuario- SUBRANGO

Program ejemplo2;

Type

años= 1960..1990;

letrasMay= 'A'..'Z';

Var

a: años;

letra: letrasMay;

¿Qué operaciones puedo hacer con a?

¿Qué operaciones puedo hacer
con letra?

Tipos de datos definidos por el usuario- SUBRANGO

Program ejemplo2;

Type

años= 1960..1990;

letrasMay= 'A'..'Z';

Var

a: años;

letra: letrasMay;

Begin

a:= 1965;

a:= 1945; OJO ERROR

read (letra);

if (letra = 'J') then ...

End.

Type

subValores = 23.5 .. 40.4;

OJO ERROR



CONJUNTOS



$$A = \{ \text{black male}, \text{red female}, \text{pink female}, \text{blue male}, \text{dark blue female}, \text{yellow female}, \text{green male}, \text{orange female} \}$$

Tipos de datos definidos por el usuario - CONJUNTO

CARACTERÍSTICAS

- Desde el punto de vista informático un tipo conjunto representará una **colección de datos simples** (además los datos que estarán guardados en el conjunto deben ser de tipo ordinal), **sin repetición y limitada** por la implementación en cada lenguaje o sistema operativo.
- No necesariamente existe en la mayoría de los lenguajes.
- Es un tipo de datos compuesto.
- No es un tipo de datos ordinal.

Tipos de datos definidos por el usuario - CONJUNTO

CARACTERÍSTICAS

- Se pueden tener conjuntos de valores enteros, boolean, y char.
- En la implementación de Pascal el conjunto no puede tener más de 255 elementos (**en la práctica esto no se tendrá en cuenta**).
- No permite operaciones de lectura - escritura.
- Permite las operaciones de asignación, unión, intersección, pertenencia, diferencia

Tipos de datos definidos por el usuario - CONJUNTO

Type identificador = ***set of ...***;

 letras = **set of char**;

Var misLetras: letras;

Begin

 misLetras:= ['a','m','i'];

 ...

End.

Tipos de datos definidos por el usuario - CONJUNTO

ASIGNACIÓN

```
Program uno;  
Type  
    letras = set of char;  
Var  
    letras1,letras2: letras;  
  
Begin  
    letras1:= [ ];  
    letras2:= ['a'..'f'];  
    letras1:= letras2;  
    ...  
End.
```

Tipos de datos definidos por el usuario - CONJUNTO

Program dos;
Type
 conjcar = **set of** char;
Var
 carac1,carac2: conjcar;
Begin
 carac1:= ['E', '7'];
 carac2:= ['a'] + carac1;

End.

UNIÓN

Se representa con el signo + y da como resultado otro conjunto.

En este conjunto resultado aparecen los elementos de los dos conjuntos y aquellos elementos repetidos aparecen una vez.

Tipos de datos definidos por el usuario - CONJUNTO

INTERSECCION

```
Program tres;  
Type  
  conjcar = set of char;  
Var  
  carac1, carac2: conjcar;  
Begin  
  carac1:= ['E', 'a'];  
  carac2:= ['a'] * carac1;  
  .....  
End.
```

Se representa con el signo * y da como resultado otro conjunto.
En el conjunto resultado aparecen solamente los elementos comunes a los dos conjuntos.

Tipos de datos definidos por el usuario - CONJUNTO

DIFERENCIA

```
Program cuatro;  
Type  
    conjcar = set of char;  
Var  
    carac1,carac2: conjcar;  
Begin  
    carac1:= ['E', '9'];  
    carac2:= ['a','E'] - carac1;  
    ....  
End.
```

Se representa con el signo - y da como resultado otro conjunto.

Este conjunto resultado contiene los elementos que están en el primer conjunto y no están en el segundo.

Tipos de datos definidos por el usuario - CONJUNTO

PERTENENCIA

```
Program cinco;  
Type  
    conjcar = set of char;  
  
Var carac1, carac2: conjcar;  
Begin  
    carac1 := ['E', '9'];  
    if ('a' IN carac1)  
    then writeln('El conjunto tiene una vocal');  
    ...  
End.
```

Se representa con el operador **IN** y da como resultado un valor lógico. Esta operación devuelve verdadero si el elemento está en el conjunto y falso en caso contrario.

Tipos de datos definidos por el usuario - CONJUNTO

COMPARACIÓN

```
Program seis;  
Type  
    conjcar = set of char;  
Var carac1, carac2: conjcar;  
Begin  
    carac1:= ['E', '9'];  
    carac2:= ['E', '9', 'F'];  
    if (carac1 <= carac2)  
        then writeln('carac1 está incluido en carac2');  
    ...  
End.
```

Se pueden usar los operadores relacionales para determinar si un conjunto está incluido en otro (\leq), si son distintos (\neq) o iguales ($=$).

EJERCICIOS



$$A = \{ \text{black male}, \text{red female}, \text{pink female}, \text{blue male}, \text{dark blue female}, \text{orange female}, \text{light green male}, \text{light orange female} \}$$

Tipos de datos definidos por el usuario - CONJUNTO

1.- Realice un programa que lea caracteres hasta leer el carácter '@', al finalizar informe la cantidad de **consonantes minúsculas** y la cantidad de **vocales minúsculas** leídas.

Tipos de datos definidos por el usuario - CONJUNTO

Program ej1;

Type

letras = **set of** char;

var

vocales,cons:letras; letra:char;

cantV,cantC:integer;

Begin

cantV:=0;

cantC:=0;

vocales:=['a','e','i','o','u'];

cons:= ['a'..'z'] – vocales;

read (letra);

while (letra <> '@') **do**

begin

if (letra in vocales)

then cantV:= cantV+1

else if(letra in cons)

then cantc:= cantC+1;

read (letra);

end;

writeln (cantV, cantC);

End.

Tipos de datos definidos por el usuario - CONJUNTO

2.- Realice un programa que lea caracteres hasta leer el carácter '@', La secuencia está formada por "palabras", al finalizar informe la cantidad de palabras con al menos 3 vocales minúsculas. Se entiende por "palabra" una sucesión de caracteres sin espacios.

casa

auxilio

resplandor

materia

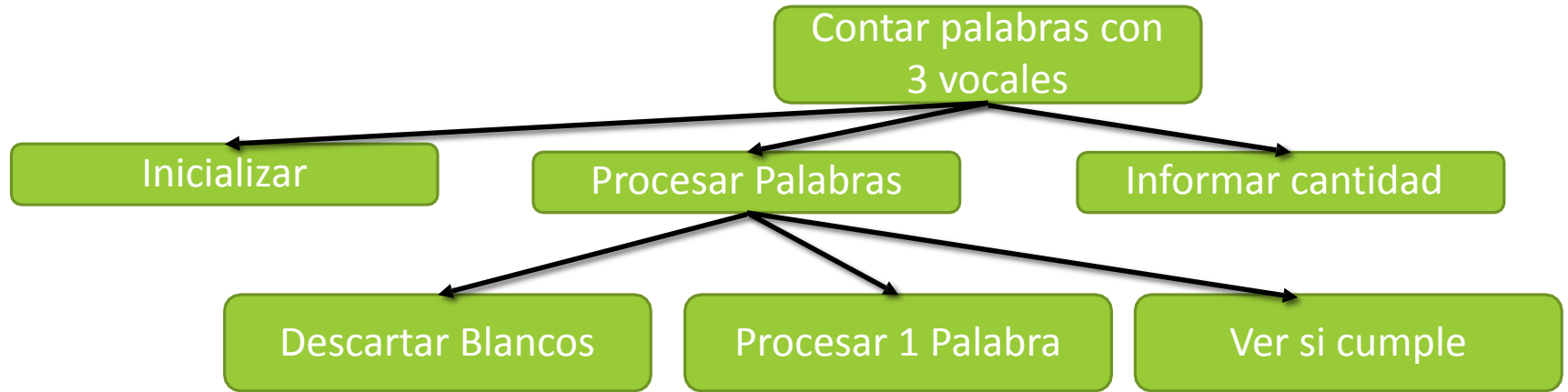
pelea



4

Tipos de datos definidos por el usuario - CONJUNTO

2.- Realice un programa que lea caracteres hasta leer el carácter '@', La secuencia está formada por palabras, al finalizar informe la cantidad de palabras con al menos 3 vocales minúsculas.



Tipos de datos definidos por el usuario-CONJUNTO

Program ej2;

type letras = set of char;

{acá van los módulos}

var vocales:letras; cant,contp:integer; letra:char;

Begin

vocales:=['a', 'e', 'i', 'o', 'u'];

contp:= 0;

read(letra);

while (letra <> '@') **do**

Begin

cant:=0;

descartarBlancos (letra);

procesarPalabra (letra, cant, vocales);

If (cant >= 3) **then** contp:=contp+1;

end;

writeln ('Cantidad de palabras que cumplen: ', contp);

End.

Tipos de datos definidos por el usuario - CONJUNTO

```
Procedure descartarBlancos (var car:char);
```

```
Begin
```

```
  while ( car = ' ') do  
    readln(car);
```

```
End;
```

```
Procedure procesarPalabra (var car: char; var cant:  
integer; vocales:letras);
```

```
Begin
```

```
  while (car <> '@') and (car<>' ')do  
    begin  
      if(car in vocales)then cant:= cant+1;  
      readln (car);  
    end;
```

```
End;
```

```
while (letra <> '@') do Begin
```

```
  cant:=0; descartarBlancos(letra);
```

```
  procesarPalabra (letra, cant);
```

```
  If (cant >= 3) then contp:=contp+1;
```

```
end;
```

```
  writeln ('Cantidad palabras cumplen: ', contp);
```

```
End.
```

Tipos de datos definidos por el usuario - CONJUNTO

3.- Realice un programa que lea caracteres hasta leer el carácter '@', La secuencia está formada por palabras, al finalizar informe la cantidad de palabras con al menos 3 vocales minúsculas distintas.

casa

auxilio

resplandor

materia

pelea



3

Tipos de datos definidos por el usuario - CONJUNTO

```
Program Ej3;  
type  
  conjletras = set of char;  
var  
  vocalesPal, vocales:conjletras;  
  cant, contp:integer; letra:char;  
Begin  
  vocales:=['a','e','i','o','u'];  
  contp:=0;  
  readln(letra);
```

```
  while (letra <> '@') do  
    Begin  
      cant:=0; vocalesPal:=[];  
      descartarBlancos(letra);  
      while (letra <> '@') and (letra<>' ' )do  
        begin  
          if(letra in vocales) and (not (letra in vocalesPal))  
            then begin  
              vocalesPal:=vocalesPal+[letra];  
              cant:= cant+1;  
            end;  
          readln (letra);  
        end;  
      If cant>=3 then contp:=contp+1;  
      Writeln(contp);  
    End;
```

Esto
podría
ser un
módulo

STRING

```
mostrar("hola mundo!");
```

comillas

Mensaje

X

hola mundo!

OK

Tipos de datos definidos por el usuario-STRING

CARACTERÍSTICAS

- Un tipo de dato string es una sucesión de caracteres de un largo determinado, que se almacenan en un área contigua de la memoria.
- Existe en la mayoría de los lenguajes como un tipo predefinido.
- No es un dato simple.

Tipos de datos definidos por el usuario-STRING

CARACTERÍSTICAS

Type

identificador = string[k]; donde k es la longitud máxima del string

identificador = string; se trabaja con 255 caracteres como máximo

STRING- Declaración

Tipos de datos definidos por el usuario-STRING

EJEMPLOS

```
TYPE  cadena1    = string [10];  
      cadena2= string [25];  
      fecha     = string; [8];  
      dia       = string [2];  
  
VAR   h1, h2, h3: cadena1;  
      h4 h5, h6: cadena2;  
      fecha1, fecha2 : fecha;  
      d: dia;
```


STRING- Asignación

Tipos de datos definidos por el usuario-STRING

OPERACIONES

Asignación:

```
Program uno;  
Type cadena= string[8];  
Var c1, c2, c3: cadena;  
Begin  
    c1:= 'casa';  
    c2:= c1;  
    c3:= 'Programacion';  
End.
```

Si cuando se asigna contenido a C1 este supera los 8 caracteres el mismo es truncado (a 8 caracteres en este caso).

STRING- Comparación



Tipos de datos definidos por el usuario-STRING

OPERACIONES

Comparación por igualdad

```
Program uno;  
Type cadena = string[20];  
Var c1, c2: cadena;  
Begin  
  read (c1); read (c2);  
  if (c1 = 'casa') then ...  
  
  while (c2 = c1) do  
    ...  
End.
```

Se evalúa si las longitudes de los strings a comparar son iguales, si es así compara el contenido. Si las longitudes no son iguales no se compara el contenido.

Tipos de datos definidos por el usuario-STRING

Comparación por distinto

Program uno;
Type cadena = string[20];
Var c1, c2: cadena;
Begin
 read (c1); read (c2);
 if (c1 <> 'casa') **then** ...

 while (c2 <> c1) **do**
 ...
End.

Se evalúa si las longitudes de los strings a comparar son iguales, si no es así se devuelve verdadero. Si las longitudes son iguales se compara el contenido.

Tipos de datos definidos por el usuario-STRING

Comparación por menor

Program uno;
Type cadena = string[20];
Var c1, c2: cadena;
Begin
 c1:= 'aula';
 if (c1 <= 'casa') **then** ...
 ...
 while (c2 <> c1) **do**
 ...
End.

OPERACIONES

Se utiliza el orden lexicográfico

Tipos de datos definidos por el usuario-STRING

Comparación por menor

Program uno;
Type cadena = string[20];
Var c1, c2: cadena;
Begin
 c1:= 'aula';
 if (c1 <= 'casa') **then** ...
 ...
 while (c2 <> c1) **do**
 ...
End.

Se utiliza el orden lexicográfico

OPERACIONES

Tipos de datos definidos por el usuario-STRING

■ Entrada/Salida

- El tipo de dato string admite las operaciones Read y Write de Pascal.
- Si la cadena ingresada supera la longitud declarada para el dato string entonces serán descartados los caracteres que se encuentran mas a la derecha.

```
Program uno;
```

```
Type
```

```
    cadena20= string [20];
```

```
    cadena5 = string [5];
```

```
Var
```

```
    cad1: cadena20;
```

```
    cad2: cadena5;
```

```
Begin
```

```
    read (cad1, cad2);
```

```
    write (cad1);
```

```
    write (cad2);
```

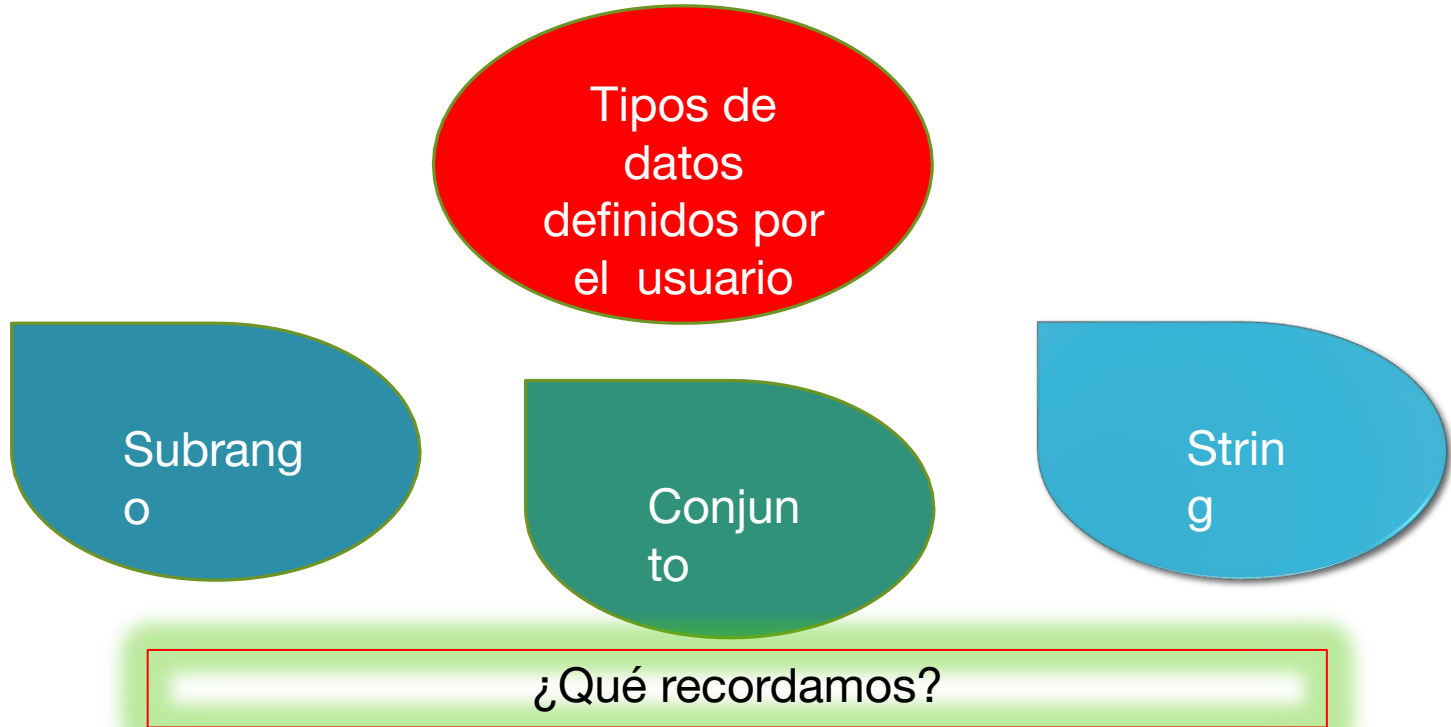
```
End.
```


EJERCICIOS

- 4.- Realizar un programa que lea códigos, nombres y precios de 20 productos de un supermercado. Informar:
- Los nombres de los productos cuyo código tiene igual cantidad de dígitos pares que impares.
 - El precio promedio de los productos.

REPASO

Tipos de datos definidos por el usuario vistos



Tipos de datos definidos por el usuario vistos.

PREGUNTAS

Repaso:

- ✓ ¿Cómo se definen los datos de tipo subrango?
- ✓ ¿Qué operadores pueden utilizarse con datos de tipo subrango?
- ✓ ¿A qué tipos de datos simples pueden aplicarse el concepto de subrango?
- ✓ ¿Cómo es el esquema general de un programa que incorpora la declaración de tipos?