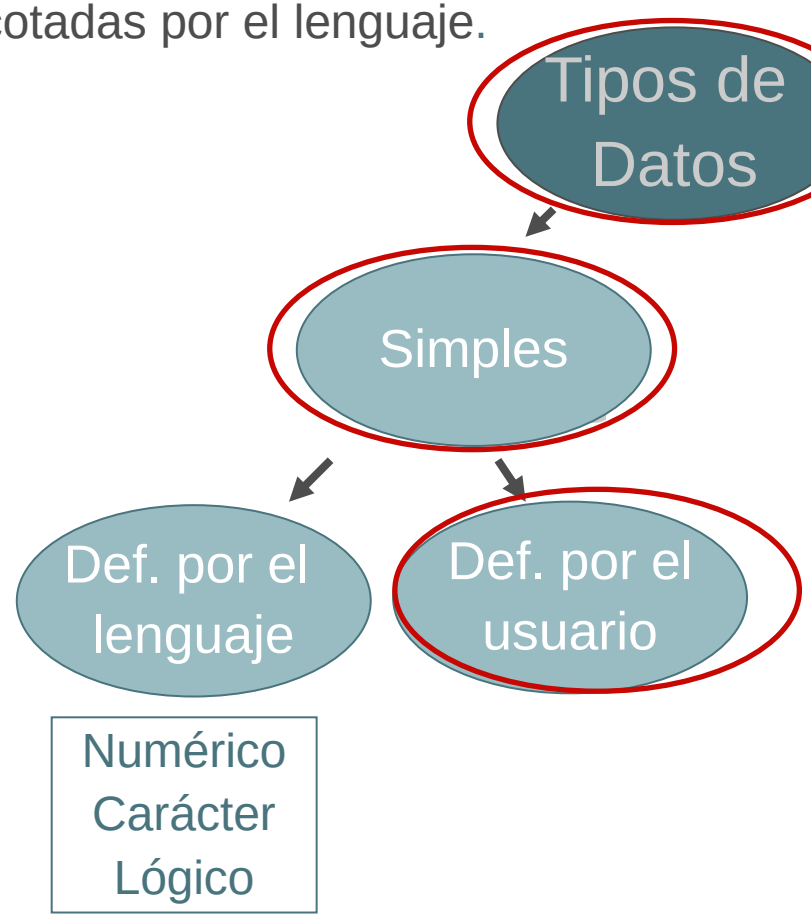
TEMAS de la CLASE

1. Tipos de datos definidos por el usuario
2. Ejercitación

Clasificación de Tipos De datos

Hasta aquí presentamos **los tipos de datos simples** - que son aquellos que toman un único valor, en un momento determinado, entre todos los permitidos para ese tipo - **y definidos por el lenguaje o estándar** - esto significa que el conjunto de valores de ese tipo, las operaciones que se pueden efectuar y su representación están definidas y acotadas por el lenguaje.



Tipos de datos definidos por el usuario

Recordemos que

- Un DATO en nuestras soluciones se utiliza para representar un objeto del mundo real.

- El tipo de dato se caracteriza por:
  - ✓ Un conjunto de valores o estados posibles.
  - ✓ Un conjunto de operaciones permitidas.
  - ✓ Una representación interna

- Puede surgir la necesidad de representar objetos del mundo real que utilicen tipos de datos diferentes a los estándar.

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.

Esto permite:

- 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.

- Establecer límites sobre los valores posibles que pueden tomar las variables que corresponden al tipo de dato.

Tipos de datos definidos por el usuario

¿Qué ventajas tiene DECLARAR tipos?

- **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:** El uso de nombres autoexplicativos como identificadores de los tipos, facilita el entendimiento y lectura del programa.

- **Seguridad:** Se reducen los errores por el uso de operaciones inadecuadas del dato a manejar, y se pueden obtener programas más confiables.

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.

Sintéticamente entonces un Tipo significa una clase de datos que tiene asociado:

- Un rango de valores posibles.
- Una forma de representación.
- Un conjunto de operaciones permitidas.
- Un conjunto de condiciones de valores permitidos que se pueden verificar.

Tipos de datos definidos por el usuario

En Pascal, los tipos deben ser declarados antes de ser usados.

La declaración de tipos se hace a través de la palabra clave TYPE de la siguiente forma:

```
TYPE identificador = tipo;
```

Nombre con que se conocerá al tipo de dato en el programa.

Puede ser un tipo estándar o alguno de los tipos de datos definidos por el usuario.

Esquema general de un programa que usa tipos definidos por el usuario

```
Program nombre;
Const
   ...
Type
   ...
Var
   ...
Begin       { Cuerpo del programa}
    ...
    ...  {Instrucciones ejecutables}
End.
```

Zona de Declaración de constantes

Zona de Declaración de tipos

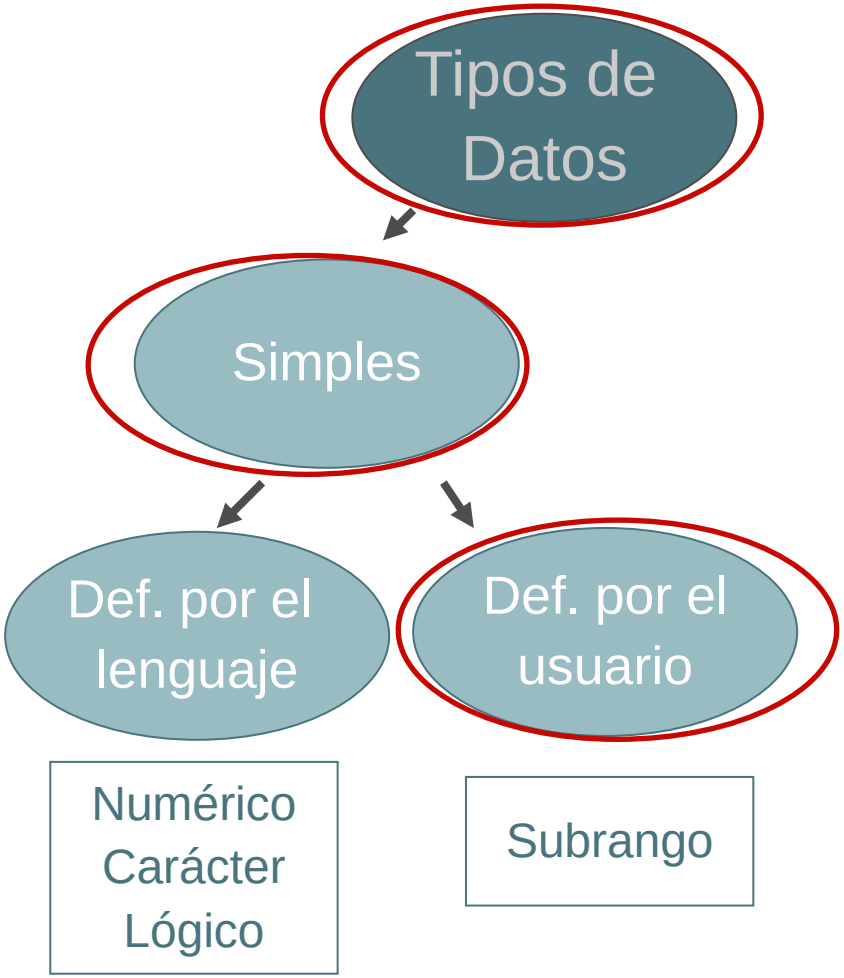
Zona de Declaración de variables

Zona de Instrucciones ejecutables

Clasificación de los Tipos de Datos

Recordemos la clasificación de los tipos de datos ya vista…

Ahora vamos a comenzar a trabajar con un tipo de dato **simple y definido por el usuario**.



Tipo de dato definido por el usuario: SUBRANGO

Un tipo de dato **subrango** es un tipo simple y ordinal que consiste de una sucesión de valores extraídos de un tipo ordinal base.

Los tipos base permitidos para definir subrangos son:

- ✓ Enteros
- ✓ Caracteres

La ocupación en memoria estará condicionada por el tipo base.

Tipo de dato definido por el usuario: SUBRANGO

Para declarar un tipo **SUBRANGO** se deben especificar los valores inicial y final de la sucesión, separados por dos puntos seguidos:

```
Type  identificador  =  valor inicial .. valor final;
```

Tipo de dato definido por el usuario: SUBRANGO

Pensemos…

¿Cual es el tipo base?

¿Cual es el rango de valores?

Edades de personas

Entre 0 y 130

Letras Minúsculas

Entre 'a' y 'z'

Días del mes

Entre 1 y 31

Meses

Entre 1 y 12

Letras Mayúsculas

Entre 'A' y 'Z'

Notas

Entre 0 y 10

Tipo de dato definido por el usuario: SUBRANGO

```
Program nombre;
Const
  fin = 10;
Type
  notas = 0 .. Fin;
  mayusculas = ‘A’ .. ‘Z’;
Var
  letra: mayusculas;
  miNota: notas;
Begin      { Cuerpo del programa}
   ...
   ...     {Instrucciones ejecutables}
End.
```

Tipo SUBRANGO - Operaciones

- Las operaciones de un tipo de dato subrango se heredan del tipo base.

```
Program ejemplo;
Const  minimo = 1;
       maximo = 500;

Type
   rango  =  minimo .. maximo;
   meses = 1 .. 12;
   minúsculas = ‘a’ .. ’z’;

Var
    dato1, dato2 : rango;
    descanso : meses;
    letra1, letra2: minúsculas;

Begin
    ...
    dato1 := dato1 Div dato2;
    descanso:= 1;
    Read (letra1);
    if (letra1 < letra2) then
    ...
End.
```

Tipo SUBRANGO - Ventajas

¿Por qué son útiles los tipos subrango?

- Facilitan el chequeo de posibles errores, pues permite que el lenguaje verifique si los valores asignados se encuentran dentro del rango establecido.

- Ayudan al mantenimiento del programa.

Ejercicio 1: Realizar un programa que lea edades de 20 alumnos de la facultad e informe el promedio de edades y la edad más grande.

Algoritmo

```
Inicializar suma edades
Repetir 20 veces
    Leer edad
    Sumar edad a suma edades
    si edad supera a edad máxima
            Actualizar edad máxima con edad
Mostrar edad máxima
Calcular promedio
Mostrar promedio
```

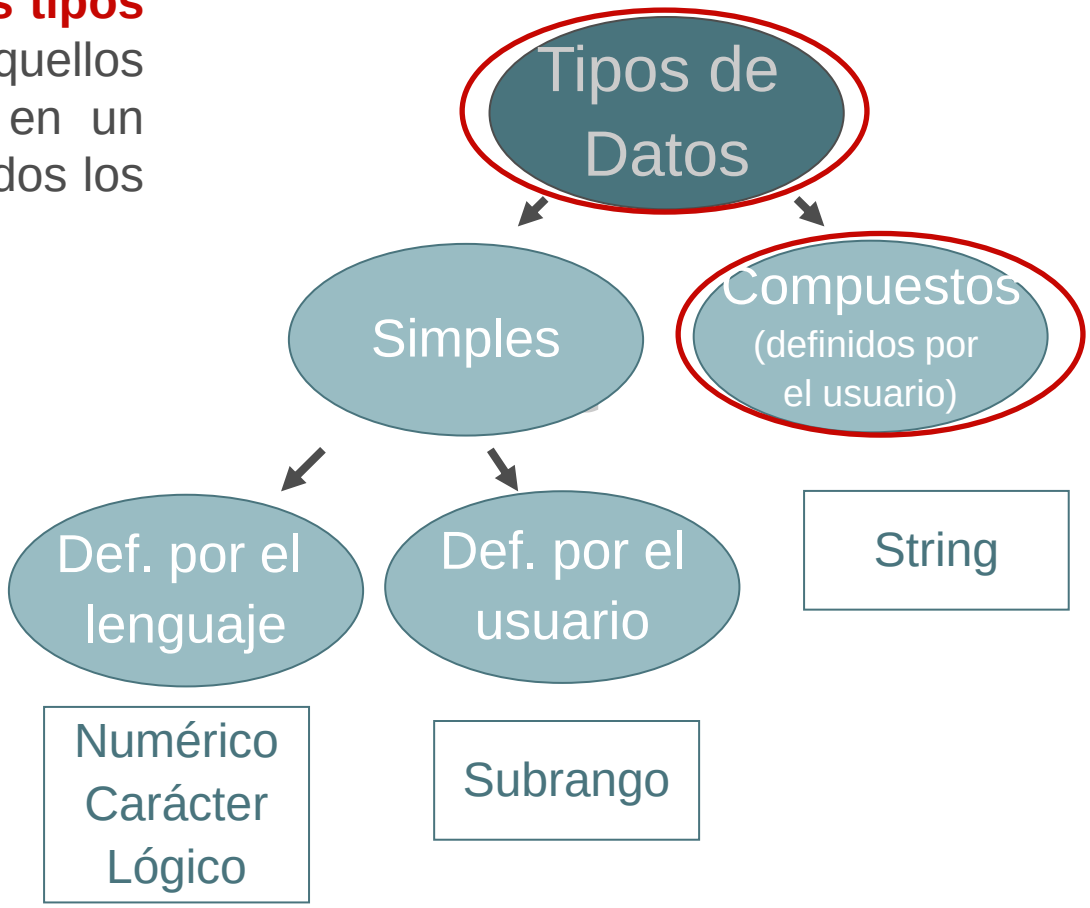
Analicemos los datos...

Ejercicio 1: Realizar un programa que lea edades de 20 alumnos de la facultad e informe el promedio de edades y la edad más grande.

```
program Ejercicio1;
Const cant = 20;
Type edades = 16..100;
var i, sumaEdades: integer;
    edad, maxEdad: edades;
    promedio: real;
begin
      sumaEdades:= 0;
      maxEdad:= 16;
      for i:= 1 to cant do
      begin
          read (edad);
          sumaEdades:= sumaEdades + edad;
          if (edad > maxEdad) then maxEdad:= edad;
      end;
      writeln ('La edad mas grande leida es: ', maxEdad);
      promedio:= sumaEdades / cant;
      writeln ('El promedio de las edades leidas es: ', promedio);
end.
```

Clasificación de Tipos De datos

- **Hasta aquí presentamos los tipos de datos simples** que son aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.



- **Ahora comenzaremos a trabajar con los tipos de datos compuestos** que son aquellos que pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos.

Tipo de dato definido por el usuario: STRING

Un tipo de dato string es una sucesión de caracteres de longitud determinada.

```
TYPE identificador = string [ longitud ];
```

- Longitud es el número máximo de caracteres que puede contener el dato.
- En PASCAL cuando no se especifica la longitud ese identificador podrá contener como máximo 255 caracteres.
- La cantidad de memoria que utiliza una variable de tipo string está determinada por su longitud más un byte que almacena la cantidad real de caracteres que contiene la secuencia. Recordar que cada carácter ocupa 1 byte.

Tipo de dato definido por el usuario: STRING

```
Program nombre;
Type
  cadena10 = string [10];
  cadena25 = string [25];
  fecha    = string [8];

Var
   nom1, nom2, nom3 :  cadena10;
   apellido :  cadena25;
   fecha1, fecha2 : fecha;

Begin      { Cuerpo del programa}
   ...
   ...     {Instrucciones ejecutables}
End.
```

Tipo de dato STRING: Operaciones

Las operaciones permitidas son:

- **Asignación (:=)**
- **Entrada/Salida (Read / write)**
- **De relación (>,<,=,…)**

Tipo de dato STRING: Operaciones

- **Asignación**

➢ Para asignar valor a una variable de tipo de dato string se hace igual que en una variable de tipo carácter (:=)

➢ Si se le asigna mayor cantidad de caracteres que lo declarado como longitud máxima, los últimos a partir de esa longitud se pierden y se dice que la hilera de caracteres “se trunca”.

```
Program uno;
Type
   cadena20= string [20];
   cadena5 = string [5];
Var
 cad1: cadena20;
 cad2: cadena5;

Begin
  cad1:= ‘buenos días!’;
  cad2:= cad1;
End.
```

Tipo de dato STRING: Operaciones

- **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.
```

Tipo de dato STRING: Operaciones

- **De Relación (=, <>, <=, =>)**

➢ Si las cadenas que se comparan son de igual longitud y contienen los mismos símbolos, en el mismo orden, el resultado de la operación es verdadero.

➢ Estos operadores realizan la comparación carácter por carácter. Si tienen distinta longitud el resultado de la comparación es falso.

```
Program uno;
Type
   cadena20= string [20];
   cadena5 = string [5];
Var
 cad1: cadena20;
 cad2: cadena5;
Begin
  cad1:= ‘buenos días!’;
  cad2:= ‘ggg´;
  if (cad1 = cad2) then…
  ...
End.
```

Ejercicio 2: Realizar un programa que lea nombres y DNI de personas hasta leer el nombre 'Ana', que debe procesarse. Se debe informar la cantidad personas cuyo DNI es par y el nombre de cada persona cuyo DNI es impar.

Algoritmo

Analicemos los datos...

```
Inicializar cantidad de pares
Repetir
        Leer nombre
        Leer dni
        Si dni es par
                incrementar cantidad de pares
        sino
                informar nombre
Hasta que nombre sea Ana
Informar cantidad de pares
```

Ejercicio 2: Realizar un programa que lea nombres y DNI de personas hasta leer el nombre 'Ana', que debe procesarse. Se debe informar la cantidad personas cuyo DNI es par y el nombre de cada persona cuyo DNI es impar.

```
Program Ejercicio2;
Type cadena20 = string [20];

var DNI, cantPares: integer;
    nombre: cadena20;

Begin
  cantPares:= 0;
  repeat
    readln (nombre);
    readln (DNI);
    if (DNI mod 2 = 0) then cantPares:= cantPares + 1
                       else writeln ('Persona con DNI impar: ',nombre);
  until (nombre = 'Ana');
  writeln ('La cantidad de personas cuyo DNI es par es ', cantPares);
end.
```

Ejercicio 3: Se leen nombres y edades de los alumnos que cursan CADP en el aula 11. Implementar un programa que informe la cantidad total de alumnos leidos y además informe el nombre y la edad del alumno con más edad. El ingreso de la información finaliza cuando se lee el nombre 'ZZZ'.

Algoritmo

Analicemos los datos...

Inicializar cantidad de alumnos

Mientras haya alumnos para procesar

    Leo el nombre y la edad del alumno

    Actualizar cantidad de alumnos

    Si (edad > edad máxima) entonces

        actualizo la edad máxima

        guardo el nombre como nombre "máximo"

Informo la cantidad de alumnos leidos

Informo el nombre y edad del alumno con edad máxima

EJERCITACION CLASE 3

Un centro de deportes quiere procesar la información de sus clientes y de los 4 tipos de actividades que ofrece: 1 (Musculación), 2 (Spinning), 3 (Cross Fit) y 4 (Libre).

Para ello, se debe leer y guardar el precio mensual de cada actividad.

Además, se debe leer para cada cliente el apellido y nombre, la edad y el número de actividad elegida (1..4). La lectura finaliza luego de procesar al cliente con nombre y apellido ‘JuanRiz’.

Se pide, informar para cada cliente, el monto a pagar y la cantidad de clientes con edad superior a 70 años.

Se sabe que cada cliente elige una sola actividad.