PROGRAMACIÓN ESTRUCTURADA

Trabajo Práctico N° 12

Registros y Arreglos

12

Apellido y Nombre: ..... Fecha: ...../...../.....

CONCEPTOS A TENER EN CUENTA

REGISTROS

Un registro es una estructura de datos compuesta que permite agrupar datos de diferentes clases (reales, lógicos, caracteres, etc.) que tienen alguna conexión lógica en una única estructura. En otras palabras, un registro es un conjunto de valores, con tres características básicas:

- Los valores pueden ser de distinto tipo, un registro es una estructura heterogénea.
- Los valores almacenados en un registro se denominan campos, y cada uno de ellos tiene un identificador; los campos son nombrados individualmente, como variables ordinarias.
- El almacenamiento (memoria) ocupado por un registro es fijo; por esto, un registro es una estructura estática.

La definición de registros en programación permite representar entidades del mundo real en soluciones basadas en computadora. Así, la información acerca de los empleados de una empresa (nombre, fecha de nacimiento, cargo, salario, etc.) pueden ser almacenados en una única estructura que permita manejar como un todo un conjunto de datos relacionados.

Declaración de Registros

Un registro se declara identificando al tipo como un registro (`t_registro=registro`) y luego especificando el nombre y tipo de los campos individuales (`campo_1:tipo_dato`). Esta lista de campos sigue las reglas generales de declaración de variables, y se encuentra entre las palabras reservadas *registro* y *fin_registro*. Los campos pueden ser de cualquier tipo, datos simples o estructurados (arreglos, registros).

El formato general de declaración de registros es:

```
TIPOS
t_registro=registro
            campo_1: tipo_dato
            campo_2: tipo_dato
            …
            campo_n: tipo_dato
          fin_registro
VARIABLES
          nombre_variable:t_registro
```

Obsérvese que una vez declarado el tipo registro, se pueden definir variables de ese tipo para utilizarlas en el programa. El siguiente ejemplo ilustra la declaración de un registro que almacena la información acerca de un producto.

```
TIPOS
t_producto=registro
            cod_producto: entero
            marca_producto: cadena
            precio_producto: real
            descripción_producto: cadena
          fin_registro
VARIABLES
          mercadería: t_producto
```

Acceso a los campos de un registro

Para acceder a los campos de un registro es necesario especificar tanto el nombre de la variable de tipo registro como el del campo que se desea referenciar. Esto se denomina calificar el campo. Por ejemplo, si se quiere almacenar un valor en el campo *precio_producto* de la variable *mercaderia* (ejemplo anterior) se procede como sigue:

```
mercaderia.precio_producto←12.36
```

Puede observarse que entre el registro *mercaderia* y el campo *precio_producto* aparece un punto. Este símbolo se denomina designador o selector de campo.

Anidamiento de registros

Las variables estructuradas, como los registros, pueden estar anidadas (una dentro de otra). Es decir, un campo de un registro puede, a su vez, ser otro registro. Un registro con uno o más campos de tipo registro se llama registro jerárquico o anidado.

El siguiente ejemplo ilustra un anidamiento de registro de 2 niveles:

```
TIPOS
t_fecha=registro
            día: entero
            mes: entero
            anio: entero
          fin_registro

t_persona=registro
            legajo: entero
            nombre: cadena
            f_nacimiento: t_fecha
          fin_registro
VARIABLES
          empleado: t_persona
```

En la declaración precedente se especifican los registros *t_fecha* y *t_persona*. Nótese que en la declaración de *t_persona* el campo *f_nacimiento* es de tipo *t_fecha*, es decir, que este campo es también un registro.

La manera de acceder a los campos esencialmente no cambia, sin embargo, es necesario utilizar doble calificación para referenciar los campos *día*, *mes* y *anio*. Por ejemplo, si se quiere almacenar un valor en el campo *día* del registro f_nacimiento, que es campo de la variable *empleado* se procede como sigue:

```
empleado.f_nacimiento.dia←28
```

Observe que se destacaron en negritas y cursiva (negritas para el primer nivel y cursiva para el segundo) los dos niveles de registro presentes en esta definición. Los siguientes son ejemplos de especificaciones INCORRECTAS de esta misma jerarquía (en negritas se indican los errores):

```
t_persona.f_nacimiento.dia←28
   empleado.t_fecha.dia←28
  t_persona.t_fecha.dia←28
```

Operaciones sobre registros

Dado que los campos de un registro son variables de algún tipo de dato, las operaciones posibles sobre un campo son las permitidas para el tipo de dato correspondiente.

Además de las operaciones sobre cada campo, existe una que puede realizarse sobre un registro completo, la *ASIGNACIÓN*. Esto es posible si las variables utilizadas en la operación son del mismo tipo de registro. Por ejemplo, si la variable *vendedor* y la variable *empleado* son del tipo *t_persona*, la siguiente operación de asignación es válida.

```
vendedor←empleado
```

En este caso, el valor de cada campo de *empleado* se copia en cada campo de *vendedor*.

No pueden realizarse comparaciones entre registros completos, es decir, que dos variables del mismo tipo de registro no pueden ser comparadas utilizando los operadores relacionales. Para determinar si dos registros son iguales es necesario realizar la comparación campo por campo.

Sobre las variables de tipo registro no se pueden aplicar directamente las operaciones *LEER* y *ESCRIBIR*; éstas deben ejecutarse sobre campos individuales. Por ejemplo, NO ES CORRECTA la sentencia

```
ESCRIBIR vendedor
```

pero si lo es la sentencia

```
ESCRIBIR vendedor.nombre
```

La sentencia WITH

Cuando se trabaja con registros, hay ocasiones en que el acceso a los campos a través de la calificación suele ser tediosa (por ejemplo, asignaciones con nombres de registro demasiado largos). Para evitar esto, el lenguaje Pascal provee la sentencia *WITH* que permite que un registro sea nombrado una vez, y luego sea accedido directamente. El formato general de la sentencia *WITH* es:

```
WITH nombre_variable_registro DO
BEGIN
        …
END
o en pseudocódigo
CON nombre_variable_registro HACER
        …
FIN_CON
```

El siguiente ejemplo ilustra el uso de la sentencia *WITH*:

```
Sin utilizar la sentencia WITH
PROCEDIMIENTO CARGAR-EMPLEADO (E/S empleado_sucursal:t_persona)
INICIO
      ESCRIBIR "Ingrese legajo del empleado:"
      LEER empleado_sucursal.legajo
      ESCRIBIR "Ingrese nombre del empleado:"
      LEER empleado_sucursal.nombre
      ESCRIBIR "Ingrese día de nacimiento:"
      LEER empleado_sucursal.f_nacimiento.dia
      ESCRIBIR "Ingrese mes de nacimiento:"
      LEER empleado_sucursal.f_nacimiento.mes
      ESCRIBIR "Ingrese año de nacimiento:"
      LEER empleado_sucursal.f_nacimiento.anio
FIN
Utilizando la sentencia WITH (CON)
PROCEDIMIENTO CARGAR-EMPLEADO (E/S empleado_sucursal:t_persona)
INICIO
     CON empleado_sucursal HACER
          ESCRIBIR "Ingrese legajo del empleado:"
          LEER legajo
          ESCRIBIR "Ingrese nombre del empleado:"
          LEER nombre
          CON f_nacimiento HACER
               ESCRIBIR "Ingrese día de nacimiento:"
               LEER dia
               ESCRIBIR "Ingrese mes de nacimiento:"
               LEER mes
               ESCRIBIR "Ingrese año de nacimiento:"
               LEER anio
          FIN_CON
     FIN_CON
FIN
```

Arreglos de Registros

En la práctica no es tan común el uso de registros simples. En general, los registros se agrupan en conjuntos conocidos como arreglos de registro. Por ejemplo, la siguiente declaración permite representar 100 productos:

```
CONSTANTES
MAXPROD=100
TIPOS
t_producto=registro
            cod_producto: entero
            marca_producto: cadena
            precio_producto: real
            descripción_producto: cadena
          fin_registro

t_stock=arreglo [1..MAXPROD] de t_producto
VARIABLES
          inventario: t_stock
```

Posición 1	Posición 2	Posición 3	...	Posición 99	Posición 100
cod_producto marca_producto precio_producto descripción_producto	cod_producto marca_producto precio_producto descripción_producto	cod_producto marca_producto precio_producto descripción_producto	...	cod_producto marca_producto precio_producto descripción_producto	cod_producto marca_producto precio_producto descripción_producto

inventario (variable de tipo t_stock)

Observe que cada una de las posiciones del arreglo *inventario* es un registro de tipo *t_producto*. En este caso si se quiere asignar un valor al campo *precio_producto* del registro que ocupa la posición 3 del arreglo *inventario* se procede como sigue:

```
inventario[3].precio-producto←69.50
```

Todas las operaciones (asignación, lectura/escritura, recorrido, actualización, ordenación, búsqueda, intercalación) vistas para arreglos son aplicables (con ligeras modificaciones) a arreglos de registros. Por ejemplo, a continuación, se presenta el algoritmo Borrar modificado para trabajar sobre el arreglo de productos definido previamente.

```
PROCEDIMIENTO BORRAR (E/S productos: t_stock, E/S ocuprod: entero, E codigoprod: entero)
variables
        i: entero
        encontrado: logico
inicio
        si ocuprod=0 entonces
             escribir "NO EXISTEN PRODUCTOS EN STOCK"
        sino
             i←1
             encontrado<-FALSO
             mientras i<=ocuprod Y NO encontrado hacer
                  si productos[i].cod_producto=codigoprod entonces
                       encontrado←VERDADERO
                  sino
                       i←i+1
                  fin_si
             fin_mientras
             si encontrado=VERDADERO entonces
                  mientras i<ocuprod hacer
                       productos[i]←productos[i+1]
                       i←i+1
                  fin_mientras
                  ocuprod←ocuprod-1
             sino
                  escribir "EL PRODUCTO NO EXISTE"
             fin_si
        fin_si
fin
```

EJERCICIOS RESUELTOS

1. El encargado del depósito de una empresa de artículos electrónicos necesita almacenar información de inventario acerca de los 400 tipos de productos que se comercializan. La información de interés para el encargado es la siguiente: código del producto, descripción, marca, precio unitario, cantidad (stock), fecha de elaboración y proveedor. Escriba un programa que permita gestionar esta información, y que presente al encargado del depósito, un menú con las siguientes opciones: 1-Agregar productos, 2-Listar productos, 3- Buscar un producto determinado, 4-Salir.

```
PROGRAMA EMPRESA
CONSTANTES
    MAXPROD=400
TIPOS
    t_producto=registro
            codigo:entero
            descrip:cadena
            marca:cadena
            precio:real
            cantidad:entero
            f_elab:t_fecha
            proveedor:cadena
    fin_registro

t_fecha=registro
            dia:entero
            mes:entero
            anio:entero
        fin-registro

t_deposito=arreglo [1..MAXPROD] de t_producto


VARIABLES
    almacen:t_deposito
    articulo:t_producto
    ocupado, opcion, codigo_prod:entero
```

```
PROCEDIMIENTO CARGAR_PRODUCTO (E/S p:t_producto)
INICIO
         escribir "Ingrese código del producto:"
         leer p.codigo
         escribir "Ingrese descrip del producto:"
         leer p.descrip
         escribir "Ingrese marca del producto:"
         leer p.marca
         escribir "Ingrese precio del producto:"
         leer p.precio
         escribir "Ingrese cantidad del producto:"
         leer p.cantidad
         escribir "Ingrese fecha de elaboración del producto:"
         escribir "Ingrese día:"
         leer p.f_elab.dia
         escribir "Ingrese mes:"
         leer p.f_elab.mes
         escribir "Ingrese año:"
         leer p.f_elab.anio
         escribir "Ingrese proveedor del producto:"
         leer p.proveedor
FIN

PROCEDIMENTO AGREGAR_PRODUCTOS (E/S prods:t_deposito, E/S ocup:entero, E nuevo:t_producto)
INICIO
     si ocup < MAXPROD entonces
          ocup←ocup+1
          prods[ocup]←nuevo
     sino
          escribir "No se pueden agregar más productos"
     fin-si
FIN

PROCEDIMIENTO MOSTRAR_PRODUCTO(E p:t_producto)
INICIO
       escribir p.codigo
       escribir p.descrip
       escribir p.marca
       escribir p.precio
       escribir p.cantidad
       escribir p.f_elab.dia
       escribir p.f_elab.mes
       escribir p.f_elab.anio
       escribir p.proveedor
FIN

PROCEDIMIENTO LISTAR_PRODUCTOS (E/S prods:t_deposito, E ocup:entero)
VARIABLES
       i:entero
INICIO
     para i desde 1 hasta ocup hacer
        mostrar_producto(prods[i])
     fin_para
FIN

PROCEDIMIENTO BUSCAR_PRODUCTO(E/S prods:t_deposito, E ocup: entero, E buscado:entero)
VARIABLES
       i:entero
       encontrado:lógico
INICIO
     i<-1
     encontrado<-FALSO
     mientras i<=ocup Y no encontrado hacer
          si prods[i].codigo=buscado entonces
             encontrado<-VERDADERO
          sino
             i<-i+1
          fin-si
     fin-mientras
     si encontrado=VERDADERO entonces
          mostrar_producto(prods[i])
     sino
        escribir "El producto no existe o el código es incorrecto"
     fin-si
FIN
```

```
INICIO
     ocupado<-0
     repetir
          escribir "1-Agregar Productos"
          escribir "2-Listar Productos"
          escribir "3-Buscar un Producto"
          escribir "4-Salir"
          escribir "Ingrese opcion:"
          leer opcion
          según opcion hacer
               1: cargar_producto(articulo)
                  agregar_productos(almacen,ocupado,articulo)
               2: listar_productos(almacen,ocupado)
               3: escribir "Ingrese código del producto a buscar:"
                  Leer código_prod
                  buscar_producto(almacen,ocupado,código_prod)
               4: escribir "Fin del Programa…"
          de otro modo
               escribir "Opción Incorrecta"
          fin_segun
     hasta_que opcion=4
FIN
```

EJERCICIOS A RESOLVER

1. Dada la siguiente definición de datos de la entidad *usuario*, diseñe las operaciones *cargar_usuario* y *mostrar_usuario*.

```
PROGRAMA gestion_usuario
TIPOS
      t_usuario=REGISTRO
                  id_usuario:entero
                  apellido:cadena
                  nombre:cadena
                  f_nacimiento:cadena
                  dni:entero
                  contacto:cadena
                  videos_publicados:entero
                  suscriptores:entero
               FIN_REGISTRO
VARIABLES
      user:t_usuario
```

2. Considerando que la definición de la entidad *usuario* (ejercicio 1) se modificó como se muestra a continuación, adapte las operaciones *cargar_usuario* y *mostrar_ usuario* a esta nueva definición.

```
PROGRAMA gestión_usuario
TIPOS
t_fecha=REGISTRO
          dia:entero
          mes:entero
          anio:entero
        FIN_REGISTRO
t_contacto=REGISTRO
            correo:cadena
            telefono:cadena
        FIN_REGISTRO
t_actividad=REGISTRO
            videos_publicados:entero
            suscriptores:entero
        FIN_REGISTRO
t_usuario=REGISTRO
          id_usuario:entero
          apellido:cadena
          nombre:cadena
          f_nacimiento:tfecha
          dni:entero
          contacto:tcontacto
          actividad:t_actividad
       FIN_REGISTRO
VARIABLES
     user:t_usuario
```

3. Analice el siguiente módulo, escriba la definición de datos de la entidad representada y diseñe el módulo de carga.

```
PROCEDIMIENTO mostrar_movil (E a:t_movil)
INICIO
     ESCRIBIR "Marca: ", a.marca
     ESCRIBIR "Modelo: ", a.modelo
     ESCRIBIR "IMEI: ", a.codigo
     ESCRIBIR "Procesador: ", a.hardware.procesador
     ESCRIBIR "RAM: ", a.hardware.memoria
     ESCRIBIR "Almacenamiento: ", a.hardware.interno
     SI a.hardware.nfc=VERDADERO ENTONCES
        ESCRIBIR "Posee chip NFC"
     FIN_SI
FIN
```

4. Consigne la declaración de tipos y variables necesaria para almacenar información acerca de un usuario de *Instagram*: apellido, nombre, fecha de nacimiento (día, mes, año), nombre de usuario, contacto (email, teléfono, Facebook), cantidad de historias publicadas, cantidad de seguidores y cantidad de usuarios seguidos. Además, diseñe los módulos necesarios para cargar, modificar y mostrar estos datos.

5. Modifique la definición del ítem anterior de modo que sea posible gestionar un máximo de 800 usuarios. Además, diseñe los módulos necesarios para agregar un nuevo usuario, modificar los datos de un usuario y listar todos los usuarios registrados.

6. Modifique los siguientes algoritmos para adaptarlos a la declaración de tipos y variables anterior

    a) Insertar un nuevo usuario (en orden decreciente, por email)
    b) Buscar un usuario por email (búsqueda binaria)
    c) Ordenar por cantidad de seguidores (de forma ascendente aplicando el algoritmo de Selección)
    d) Eliminar un usuario (identificado por nombre de usuario)

7. Consigne la declaración de tipos y variables necesaria para almacenar la siguiente información acerca del catálogo de películas de *Netflix* (considere 3000 títulos): título, director, género, duración (horas, minutos, segundos) y ficha técnica (año de estreno, productora, país). Además, diseñe los módulos para:

    a) Agregar películas
    b) Ordenar películas por año de estreno (ordenación Burbuja)
    c) Determinar las películas de menor y mayor duración.
    d) Mostrar las películas que correspondan a un género, año de estreno y país especificados por el usuario.

8. El responsable del depósito de una farmacia desea registrar información acerca de los 400 medicamentos que se mantienen en stock. Por cada medicamento deben almacenarse los siguientes datos: id de medicamento, nombre, descripción, presentación, precio unitario, laboratorio (nombre, dirección, teléfono) y stock (cantidad de unidades). En virtud de lo enunciado se pide:

    a) Consigne la declaración de tipos y variables que represente la situación planteada. Considere que el conjunto de medicamentos debe implementarse mediante un arreglo de DOS dimensiones.
    b) Diseñe un procedimiento/función que liste los medicamentos (nombre, presentación y precio unitario) cuyo laboratorio corresponda a uno especificado por el usuario. Indique cuántos medicamentos se encontraron.
    c) Diseñe un procedimiento/función que muestre el medicamento con el menor stock.
    d) Diseñe un procedimiento/función que borrar un medicamento.

9. Una aplicación de enseñanza de idiomas mantiene registro de sus usuarios y de los logros en su proceso de aprendizaje. Para ello, la aplicación debe almacenar la siguiente información: id de usuario, apellido y nombre, nombre de usuario, correo electrónico, número de teléfono, fecha de nacimiento (día, mes, año), cantidad de lecciones aprobadas, nivel actual, cantidad de días de entrenamiento y cantidad de amigos (en la red educativa de la aplicación). En virtud de lo enunciado se pide:

a) Consigne la declaración de tipos y variables que represente la situación planteada.
b) Diseñe un procedimiento/función que muestre los usuarios (nombre de usuario, lecciones aprobadas y nivel actual) que tienen menos de un año utilizando la aplicación.
c) Diseñe un procedimiento/función que muestre el usuario (apellido y nombre, cantidad de días de entrenamiento) con la máxima cantidad de lecciones aprobadas.
d) Diseñe un procedimiento/función que liste los usuarios (apellido y nombre, correo y nivel actual) que aún no tengan amigos.

10. El sistema de gestión del personal de la UNJu registra información acerca del personal docente y no docente que se desempeña en sus 4 facultades. Respecto al personal se almacena la siguiente información: legajo, apellido, nombre, fecha de alta (día, mes, año), domicilio (calle, número, barrio), cargo e id de facultad. En cuanto a las facultades se registra: id de facultad, nombre, domicilio (calle, número, barrio) y teléfono. En virtud de esto se solicita:

a) Consigne la declaración de tipos y variables que represente la situación planteada.
b) Diseñe un procedimiento/función que liste, por cada facultad, los docentes que cumplen funciones en ellas. Considere que el atributo *cargo* indica si se trata de un docente o no docente.
c) Diseñe un procedimiento/función que muestre, por cada facultad, la cantidad de docentes y la cantidad de no docentes que cumplen funciones en ellas. Considere que el atributo *cargo* indica si se trata de un docente o no docente.
d) Diseñe un procedimiento/función que liste los no docentes de mayor antigüedad. Considere que por cada no docente listado debe mostrarse la facultad a la que pertenece. Para el cálculo de antigüedad tome como referencia el año 2023.