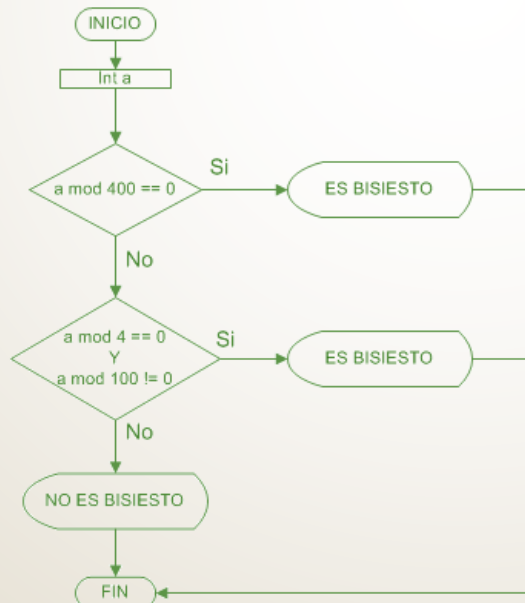


Programación Estructurada

UNIDAD VI: REGISTROS



Índice

- Definición de registros
- Declaración de registros
- Acceso a los campos de un registro (calificación)
- Anidamiento de registros
- Sentencia *WITH*
- Operaciones sobre registros
 - Asignación, Lectura y Escritura
- Arreglos de registros

Definición (1)

- ¿Cómo representar las entidades del mundo real teniendo en cuenta sus características?
- En programación, los REGISTROS se usan para definir un conjunto de datos relacionados como una única estructura.

EMPLEADO

Nombre
Fecha de Nac.
Cargo
Salario
...



PRODUCTO

Código
Descripción
Precio
Stock
...

Definición (2)

- Un *registro* es una estructura de datos compuesta que agrupa, en una única estructura, datos de diferentes tipos (reales, lógicos, caracteres, etc.) que tienen alguna conexión lógica.
- Características:
 - un registro es una estructura **heterogénea** que puede contener datos de distinto tipo,
 - los componentes de un registro se denominan **campos**, éstos pueden ser accedidos en forma individual a través de identificadores y
 - un registro es una estructura **estática** ya que el espacio de memoria que ocupa es fijo.

Declaración (1)

TIPOS

`t_registro=REGISTRO`

Tipo registro

`campo_1: tipo_dato`

`campo_2: tipo_dato`

`...`

`campo_n: tipo_dato`

*Campos
de
registro*

FIN_REGISTRO

VARIABLES

`nombre_variable:t_registro`

Variable tipo registro

Declaración (2)

TIPOS

t_producto=REGISTRO

*Identificador
de campo*

codigo: entero

descripcion: cadena

precio: real

stock: entero

*Tipo de dato
del campo*

FIN_REGISTRO

VARIABLES

articulo: t_producto

*Registro artículo
de tipo t_producto*

Calificación

- Para **acceder** a los campos de un registro se debe indicar el **nombre del registro y del campo** que se desea referenciar. Esto se denomina **calificar** el campo. Por ejemplo:

articulo . precio

- Entre el **registro (artículo)** y el **campo (precio)** se indica el **operador punto**, también conocido como designador o **selector de campo**.

Anidamiento de registros (1)

- Los registros son tipos estructurados que permiten el **anidamiento**.
- El anidamiento permite que **un campo de registro sea a su vez otro registro**.
- Un registro con uno o más campos de tipo registro se llama **registro jerárquico o anidado**.



Anidamiento de registros (2)

TIPOS

`t_fecha=REGISTRO`

`dia: entero`

`mes: entero`

`anio: entero`

`FIN_REGISTRO`

`t_persona=REGISTRO`

`legajo: entero`

`nombre: cadena`

`fecha_nac: t_fecha`

`FIN_REGISTRO`

*Se pueden
definir campos
de tipo registro*

VARIABLES

`empleado: t_persona`

Anidamiento de registros (3)

- El **acceso** a los campos de un registro anidado se realiza mediante una **calificación sucesiva** de campos tipo registro. Por ejemplo:

Registro empleado **empleado**. *Registro fecha nacimiento* **fecha_nac**. dia

- Especificaciones INCORRECTAS de la jerarquía anterior:

 t_persona.fecha_nac.dia

empleado.t_fecha.dia 

t_persona.t_fecha.dia 

Operaciones sobre registros (1)

- Cada campo de registro puede operarse según el tipo de dato que le corresponda.
- **Asignación** entre registros del mismo tipo. Si *vendedor* y *empleado* son *t_persona* es válido:

vendedor ← **empleado**

- No pueden compararse registros completos, la **comparación** se realiza **campo por campo**.

SI **empleado** < > **vendedor** ENTONCES
 ESCRIBIR "DISTINTOS"
SINO
 ESCRIBIR "IGUALES"
FIN_SI



SI **empleado.legajo** < > **vendedor.legajo** ENTONCES
 ESCRIBIR "DISTINTO"
SINO
 ESCRIBIR "IGUALES"
FIN_SI



Operaciones sobre registros (2)

- Las operaciones LEER y ESCRIBIR sólo pueden ejecutarse sobre **campos individuales**.

~~LEER empleado~~ (INCORRECTO)

LEER empleado.legajo (CORRECTO)

~~ESCRIBIR vendedor~~ (INCORRECTO)

ESCRIBIR vendedor.nombre (CORRECTO)

Sentencia WITH (1)

- Los registros anidados de varios niveles pueden hacer que el acceso a los campos a través de la calificación sea tediosa y ardua.
- El lenguaje Pascal cuenta con la sentencia *WITH* que permite especificar el nombre de un registro una sola vez y acceder directamente a sus campos.

Sentencia WITH (2)

- En Pascal

```
WITH nombre_variable_registro DO  
    BEGIN  
        ACCIONES  
        ...  
    END
```

- En pseudocódigo

```
CON nombre_variable_registro HACER  
    ACCIONES  
    ...  
FIN-CON
```


Sentencia WITH (3)

- Acceso a campos de registro sin usar WITH

PROCEDIMIENTO Alta_Emp (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.fecha_nac.dia

ESCRIBIR "Ingrese mes de nacimiento:"

LEER empleado_sucursal.fecha_nac.mes

ESCRIBIR "Ingrese año de nacimiento:"

LEER empleado_sucursal.fecha_nac.anio

FIN

Sentencia WITH (4)

- Acceso a campos de registro usando WITH

```
PROCEDIMIENTO Alta_Emp (E/S empleado_sucursal:t_persona)
```

```
INICIO
```

```
    CON empleado_sucursal HACER
```

```
    INICIO
```

```
        ESCRIBIR 'Ingrese legajo del empleado:'
```

```
        LEER legajo
```

```
        ESCRIBIR 'Ingrese nombre del empleado:'
```

```
        LEER nombre
```

```
        CON fecha_nac HACER
```

```
        INICIO
```

```
            ESCRIBIR 'Ingrese día de nacimiento:'
```

```
            LEER dia
```

```
            ESCRIBIR 'Ingrese mes de nacimiento:'
```

```
            LEER mes
```

```
            ESCRIBIR 'Ingrese año de nacimiento:'
```

```
            LEER anio
```

```
        FIN
```

```
    FIN
```

```
FIN
```

Arreglos de registros (1)

- En general, los registros se agrupan en conjuntos conocidos como arreglos de registros.

Posición 1	Posición 2	Posición 3	...	Posición 99	Posición 100
código descripción precio stock	código descripción precio stock	código descripción precio stock	...	código descripción precio stock	código descripción precio stock

inventario (variable de tipo t_prods)

Las operaciones de asignación, lectura/escritura, recorrido, actualización, ordenación, búsqueda, intercalación para arreglos son aplicables (con ligeras modificaciones) a arreglos de registros.

**inventario[99].codigo
inventario[99].descripcion
inventario[99].precio
inventario[99].stock**

Arreglos de registros (2)

➤ Un conjunto

CONSTANTES

MAXPROD=1

TIPOS

t_product

t_prods=A

VARIABLES

inventari

```
...
const int MAXPROD=100;
...
typedef char tcad[30];
typedef struct tproducto {
    int codigo;
    tcad descripcion;
    float precio;
    int stock;

};

typedef tproducto tprods[MAXPROD];
...
main()
{ tprods inventario;
    ...
}
```

Arreglos de registros (3)

- Agregar un producto:

```
void agregar(tprods articulos, int &ocupado, tproducto nuevo)
{ if (ocupado==MAXPROD-1)
    cout << "ARREGLO COMPLETO" << endl;
  else
  { ocupado++;
    articulos[ocupado]=nuevo;
  }
}
```

FIN

Arreglos de registros (4)

- Buscar un producto

FUNCIÓN `busq_sec(E articulos:t_prods, E ocupado:entero, E buscado:entero):lógico`

```
bool busq_sec(tprods articulos, int ocupado, int buscado)
{ int i;
  bool existe=false;
  i=0;
  while (i<=ocupado && existe==false)
  { if (articulos[i].codigo==buscado)
      existe=true;
    else
      i++;
  }
  return existe;
}
```

FIN

Arreglos de registros (5)

```

PROCEDIMIENTO selección(E/S arts:t_prods, E ocup:entero)
VARIABLES
    i
INICIO
    PARA i
        PARA
            S
                if (arts[i].codigo > arts[k].codigo)
                    cambio(arts[i], arts[k]);
        }
    FIN
FIN_PA
FIN
PROCEDIMIENTO
VARIABLES
    aux:
INICIO
    au
    r1
    r2 ← aux
FIN

```

```

void seleccion (tprods arts, int ocup)
{ int i,k;
  for(i=0;i<ocup;i++)
    for(k=i+1;k<=ocup;k++)
      if (arts[i].codigo > arts[k].codigo)
        cambio(arts[i], arts[k]);
}

void cambio (tproducto &r1, tproducto &r2)
{ tproducto aux;
  aux=r1;
  r1=r2;
  r2=aux;
}

```

Bibliografía

- Sznajdleder, Pablo Augusto. Algoritmos a fondo. Alfaomega. 2012.
- López Román, Leobardo. Programación estructurada y orientada a objetos. Alfaomega. 2011.
- De Giusti *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta, 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Joyanes Aguilar, Luis. Programación en Turbo Pascal. Mc Graw Hill. 1990.