# Diseño Modular Top-Down

Pasar arreglos

# Consideraciones para arreglos

1. Los arreglos de cualquier tipo al ser enviados como parámetros, por definición estos se pasan por referencia.

2. En el llamado de un módulo, cuando se envía un arreglo como parámetro, se omite la palabra reservada dir y solo se envía el nombre del arreglo.

3. En la definición de un módulo, cuando se recibe un arreglo como argumento, solo se agregan [].

# Consideraciones para variables

1. En el llamado de un módulo, cuando se envía una sola variable como parámetro, no se omite la palabra reservada dir.

2. En la definición de un módulo, cuando se recibe una sola variable como argumento, no se omite la palabra reservada var.

# Sintaxis para pasar arreglos

## Definición

tipo nombre(tipo:  d1[])

## Llamado

nomvar ← nombre_módulo(d1)

Coment: d1 es un arreglo, por ejemplo

d1[20]: Entero

# Ejemplo: Pasar una cadena

```
Principal()
Inicio
 Variables:   Caracter: nif[10]
   cadena(Caracter:  cad[])
   nif[]← "hoy”
   cadena(nif)
   EscribirC(nif)
   Regresa()
Fin_Principal
```

```
cadena(Caracter:  cad[])
Inicio
  Variables: i: Entero
  Para(i←1 hasta i<=LenC(cad) Incremento 1)
    cad[i] ← ‘H’
  Fin_Para
  Regresa()
Fin_cadena
```

# Prueba de escritorio

| Principal | | |
|---|---|---|
| Entrada | Variables | Salida |
| nif | nif | |
| h | H | |
| o | H | |
| y | H | H H H |

| cadena | | | |
|---|---|---|---|
| Entrada | Variables | | Salida |
| cad | i | cad | |
| h | 1 | H | |
| o | 2 | H | |
| y | 3 | H | |

A cada iteración de i se va modificando el valor que almacena el arreglo nif

# Ejemplo: Pasar vector

```
vec(Entero:  num[] , Entero: tam)
Inicio
   Variables Entero: i
   Para(i←1 Hasta  tam Incremento 1)
       num[i]← num[i]*2
   Fin_Para
  Regresa()
Fin_vec
```

```
Principal ()
Inicio
   Variables Entero: p,  n[]←{1,2,7,8,9}
    vec(n,5)
    Para(p←1 Hasta 5 Incremento 1)
     Escribir(“los números son”,n[p])
 Fin_Para
 Regresa()
Fin_Principal
```

# Prueba de escritorio

| Principal | | | |
|---|---|---|---|
| Entrada | Variables | | Salida |
| | p | n | n |
| | 1 | 1 | 2 |
| | 2 | 2 | 4 |
| | 3 | 7 | 14 |
| | 4 | 4 | 8 |
| | 5 | 9 | 18 |

| vec | | | | |
|---|---|---|---|---|
| Entrada | Variables | | | Salida |
| num | i | tam | num | |
| 1 | 1 | 5 | 2 | |
| 2 | 2 | 5 | 4 | |
| 7 | 3 | 5 | 14 | |
| 4 | 4 | 5 | 8 | |
| 9 | 5 | 5 | 18 | |

A cada iteración de i se va modificando el valor que almacena el arreglo n

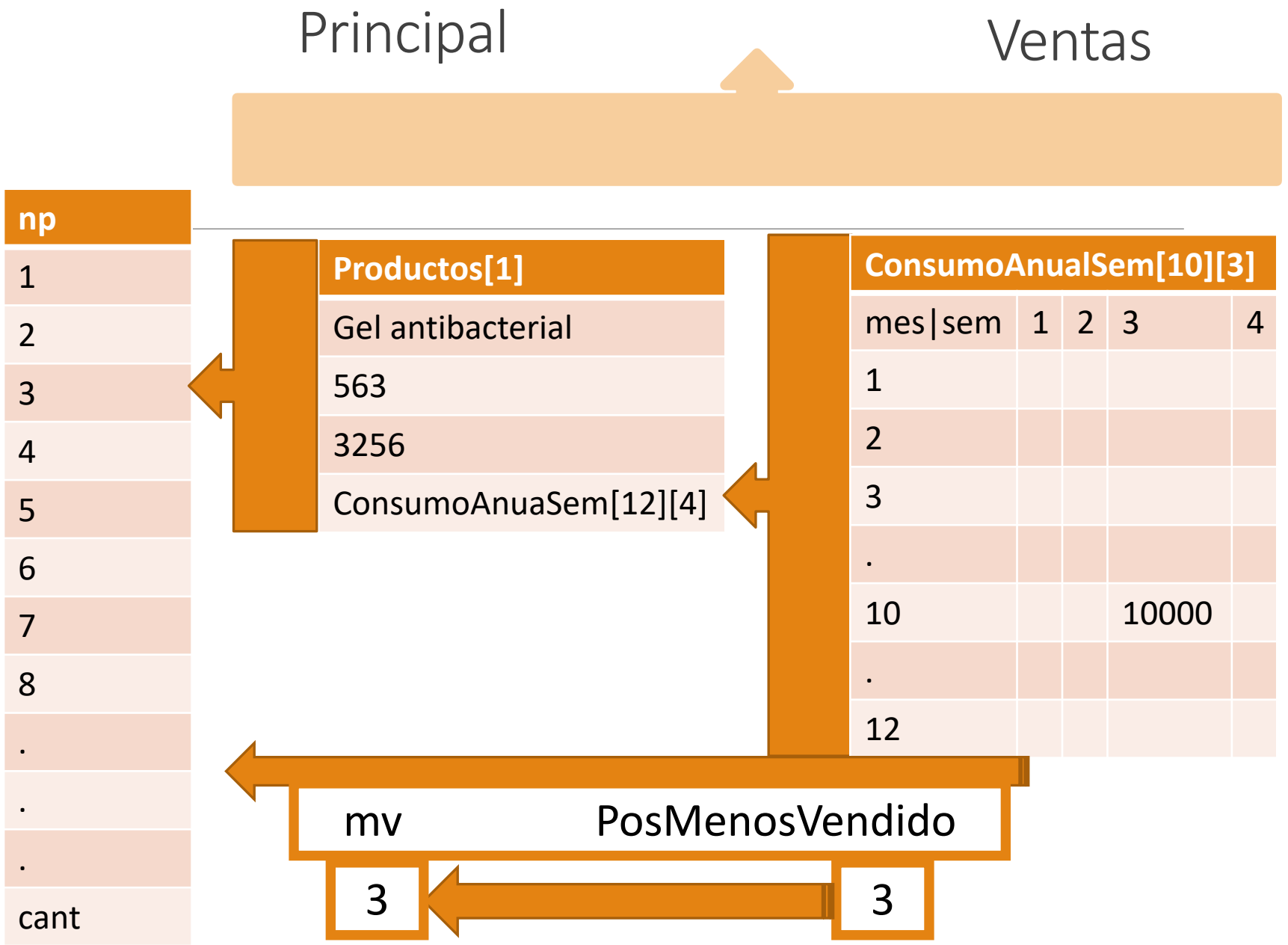
# Ejemplo: Pasar arreglo registros

```
Tipo Registro dato =
    nombre[10]: Caracter
    clave: Entero
    stock: Entero
    ConsumoAnualSem[12][4]:
Entero
 Fin_Registro
```

```
Principal ()
Inicio
  Variables np, cant, mv, mes, sem: Entero
            productos[100]: dato , prod[10][20]: Entero
            Entero Ventas(productos[]: dato, cant: Entero, p[][20])

  Leer(cant)
  Para (np ← 1 hasta cant, incremento 1)
     LeerC (productos[np].nombre)
     Leer (productos[np].clave)
     Leer (productos[np].stock)
        Para (mes ← 1 hasta 12, incremento 1)
           Para (sem ← 1 hasta 4, incremento 1)
              Leer (productos[np]. ConsumoAnualSem[mes][sem])
           Fin_Para
        Fin_Para
    Fin_Para
    mv ← Ventas(productos, cant, prod)
    Regresa()
Fin_Principal
```

```
Entero Ventas(productos[]: datos, cant: Entero, prod[][20])
    Inicio
        Variables: np, PosMenosVendido, MenosVendido: Entero
        MenosVendido ← 0
        PosMenosVendido ← 1
        Para (np ← 1 hasta cant, incremento 1)
           Si (productos[np]. ConsumoAnualSem[10][3] > MenosVendido) AND (productos[np].stock>100
              MenosVendido ← productos[np]. ConsumoAnualSem[10][3]
              PosMenosVendido ← np
           Fin_Si
       Fin_Para
       Regresa(PosMenosVendido)
Fin_Ventas
```


Principal
Ventas
np
1
2
3
4
5
6
7
8
.
.
.
cant
Productos[1]
Gel antibacterial
563
3256
ConsumoAnuaSem[12][4]
ConsumoAnualSem[10][3]
mes|sem 1 2 3 4
1
2
3
.
10 10000
.
12
mv PosMenosVendido
3
3

# Práctica

```
Entero  Facto(Entero: num[], Entero: can, Entero var sufac)
    Inicio
      Variables: Entero: cual, fac, t
Para (cual←1 Hasta can Incremento 1)
          fac←1
          t←num[cual]
          Mientras(t >1)
             fac=fac*t
             t←t-1
          Fin_Mientras
          var sufac←var sufac+fac
 Fin_Para
      Regresa(fac)
   Fin_Facto
```

```
Principal()
Inicio
Variables:  Entero: f, num[50], ele, s←0,
            cuantos
   Leer(cuantos)
   Para (ele←1 Hasta cuantos Incremento 1)
      Leer(num[ele])
   Fin_Para
   f← Facto(num, cuantos, dir s)
   Escribir (“Los resultados son:” f, “y” s)
   Regresa()
Fin_Principal
```

# Preguntas

1. ¿Qué hace el algoritmo?
2. ¿Qué parámetros son pasados por referencia?
1. ¿Qué valores imprime? sí cuantos tiene el valor 5

num tiene 2 5 7 4 3

```
Principal()
Inicio
 Variables: Entero numeros[10], fac[10], pos
 Escribir("Dame 10 enteros")
  Para(pos-> 1 Hasta 10 Paso 1)
     Leer(numeros[pos])
  FinPara
  Calcula_Fac(numeros, fac)
  Para(pos-> 1 Hasta 10 Paso 1)
     Escribir(fac[pos])
  FinPara
 Regresa()
FinPrincipal
```

| |
|---|
| 3 |
| 2 |
| 1 |
| 4 |
| . |
| . |
| . |
| 20 |

| |
|---|
| 3! |
| 2! |
| 1 |
| 4 |
| . |
| . |
| . |
| 20 |

```
Calcula_Fac(numeros[], fac[])     3!=3*2*1
Inicio
 Variables: Entero i, F,k
  Para (i -> 1 Hasta 10 Paso 1)
    F->numeros[i]
    Si (F>1) Entonces
       Para (k->F-1 Hasta 1 Paso -1)
         F-> F*k
       FinPara
        fac[i]->F
     Sino
        fac[i]->1
    Fin_Si
  FinPara
  Regresa()
Fin_Calcula_Fac
```