

EjercicioTransformacionResueltos...



Duvan02



Metodología de la Programación



1º Grado en Ingeniería Informática



Escuela Superior de Ingeniería Universidad de Cádiz



Estamos de
Aniversario

De la universidad al mercado laboral:

especialízate con los posgrados de EOI y marca la diferencia.





¡UNA HORA UN TRIDENT MÁS Y YA LO TIENES!



Ejercicio 10.

- Topla de parámetros formales x = x,n,i
- carbase: 1:0
- solca = 0
- " DUCCEN: X,n,i-1
- · (ombiz, 3)= x[i] + }
- a) Generalización
- · Funcion sumergida

suma Vect-rec (x,n,i) = x(i)+ suma Vect-rec (x,n,i-1)

· Anbol sintactico (Junción sumergida)

xcir sumalectiec

SumVed-rec

· Aíbol sintáctico con los parámetros de inmersión

Sum SumVect. rec

- · Función inmersora: sumvect Rec (x,n,i, sum) = sum + sumvect_rec(x,n,i)
- · Valores iniciales: Elemento neutro de la suma = O
- · Llamada inicial: sumVectRec(x,n,i,0) = 0 + sumVectrec(x,n,i)
- · Funcion Clamada inicial.

// Cabezera: entero: sum (E Vectix, E entero:n, E entero:i)

1/ Precondition: X: A[1.1] 1 noon OxiEn

Most condición: devuelve el valor Ex[4]

entero: sum (E vect: x, E entero: n, E entero: i)

inicio





```
devolue sum VecTRec (x n,i, o);
    Sin. Juncion
b) Desplegado
   · Siguiendo le definición de Jon-recfinal obsenida en la generalización y
     siquiendo el mismo análisis de casos que en la función recusiva no final,
     diseñar la función inmersora:
     // Cabezera: entero: sumvect Rec ( & Vect: x, & Entero: x, & entero: i, & entero: sum)
     // Piecondición: x= A[1. n] 1 n20 1 Ofien
    // Postcondición: develve el valo. Ex[a]
    - Enlero: sumVect Poc (Evect:x, Enlero:x, Eenloro:i, EEnlero: sum)
        inicio
            si i=o enfonces
                devolver sum +0
            Di no
                 devolves sum + x Ci] + suma Vect-rec (x, n, i-1)
            Jin-si
       din Juncion.
    · Reorganización del caso general:
       (Sum + xci) + suma Vect_101(x,n,i-1)
        Lo único que hemo hecho ha sido reorganizar Por Termino,
    · Función recusiva final ( aun dependiente de la fonción recusiva no final)
    // Cabezora: EnTero: Sum Ved Recl E vect: x, E EnTero: n, E EnTero: i, E EnTero: Sum)
    // Precondición: x= A[1.n] x noo x Osian
    1/ Postcondición: devuelva el valo: Éxtas
    Entero: Sumbect Roc (E Vect: x, E Entero: x, E Entero: 1, E Entero: sum)
      inicio
          si i=0 entonces
                devolver sum +0
               devolver (sum + x [i]) + sumalect rec(x,n,i-1)
          Pin-si
      din- Jurcion
```

```
c) Peogado
   · Sucesores: Suc (sum = sum + x[i]
   · Version Recuisiva Final
     Mabereia: Entero: som Vect Rec (E Vect:x, & Entero:n, & Entero i, & Entero:som)
     / Precondición: x: A[1.n] 1 noo 10 6 i en
     // Postcondición: develve el valor Éxcas
Entero: sum Ved Rec (E Vect: x, E Entero: n, E Entero: , E Entero: sum)
         inicio
              di iso enTonces
                     devolves sum +0
                   devolver sumbect Rec (x,n,i-1, sum + x [i])
               Sin-si
         din-Juncion
Transformación de Junción recursiva no final (RNF) a Iterativa
 //Cabezera: EnTeio: sum Itei ( & Vectix, & Enteio: n, & EnTeio i)
 // Precondicion: X = ACT. n] 1 noo 105 icn
 // Postcondición: devuelve el valor Excas
         Enlew C, res
    inicio
         mientia r(i=o) hacei
               i (- i-1
          din-mientras
          7es ( 0
          mientian Cto hacer
                 c+ (-1
                 ic 1+1
                 rest rei+x[i]
          fin-mientry
          devolver res
    Sin-Junción
```



INA HORA UN TRIDENT MÁS Y YA LO TIENES!



```
STIIIIIRA TUS MOMENTOS
```

```
optimización del código
// Cabezera: Entero sum ITer (E Vect:x E Entero:x, E Entero:i)
// Precondición: x= A[1.n] 1 noo 1 Oxien
// Postwordición: develve el valor Exx[a]
EnTero: Sum Iter (E Vect:x, & EnTero:x, & EnTero:i)
         entero: c, res
    inicio
       c+n-i
       160
     · rest o
       Mientiai C+O hacei
           (-1-1
           i + i+1
           lest les + xCi]
        din mientiai
        devolver res
 Sin-Juncion
 Il Para optimizar la Junion lo que hemo hecho es eliminar el prima, buele, ademais
 hemos inicializado calvalor n-i y i a O, para que la función se ejecute de
 maneia conecta
Transformación de Recursiva Final (RF) a ITerativa.
Mabezera: EnTero: sum (E Vect: x, E EnTero: n E EnTero i)
// Precondición: X = A[1... n] a não A DEIEN
// Postandición: devuelve el valor z x [a]
Entero: sum (E Vect: x & Entero: n & Entero:i)
       devolver sum ITer (x, n, i, o)
  fin- funcion
// Cabezera: Entero: SumIter ( E Vectix, E Entero:n, E entero:i, E Entero sum)
/Precondición: x=A[1.n] A NOO A OSIEN
// Postcondición: devuelve el valos & xca]
```



```
Entero: sum Ita (E Vect: x, E Entero: n, E Entero: i, E Entero: sum)
  inicio
       mienta r (1=0) hacer
          Lijx + muc = muc
          1=1-1
       gin-mientra
     devolves sum + 0
 fin. Junción
Optimización del código
// Cabezeia: EnTeio: sum ITer (E Vect. x, E EnTeio: n, E EnTeio: i E EnTeio: jum)
//Recondición: x=A[1..n] 1 noo 1 osi<n
Most condición: de vuelve el valor Excas
Entero: Dum ITEr (E Vect: x, E Entero:n, E Entero: i, E Entero: Jun)
 inicio
     sum to
     i+ n-1
     mientian o (i=0) hace,
        Ligx + mue = mul
        1=1-1
      gin-mientia
      de Nohre sum + 0
Sin-Juncion
Mara optimizar la función lo que hemor hecho ha sido inicializar los
parametros sum a 0 e i a n-1 para que la función funcione correctamente.
```



deja de imaginar que trabajarás de lo tuyo hazlo posible.



```
Ejercicio 11.
- Tupla de parámetros formales x = x, g, n, i
  caso base (2) = i=n
- solai = xci] yen-i+1]
* Suc(x) = x, y, n, i+1
· (omb(i,j) = x[i],y[n-i+1] + (i11). }
a) Generalización
   · Función sumagida: sumVects-rec(x,y,n,i) = x[i]. y[n-i+1] + (i-1). sumVects-rec(x,y,n,i+1)
   · Aibol sintactico (Junción sumagida)
                         XCI] y[n-i+]
                                     son/eds_rec
   · Arbol sintáctico con los parámetros de inmersión
                                     sumVect, nec
  · Función inmersora: sumbedRec (x,y,n,i,sum,prod) = sum + prod. sumbeds-rec(x,y,n,i)
   · Valores inicialez: Elemento neutro de la suma =0
                        Elemento neutro del moduto = 1
   · Llamada inicial: sumVectRec (x,y,n,i,0,1) = 0+1. sumVectrac (x,y,n,i)
   · Función llamada inicial
    // Cabezera: Entero: Namada ( E Ved: x, E Ved: y, E Entero:n, E Entero:i)
   // Precondición: x: ACI.n] Ay= BCI.n] Alen Al 31 Ano
   // Postcondición: devuelve el valor & al (x cas y cn-x+1)
    Entero: Plamoda ( & Vect: x, & Vect: y, & Entero: n, & Entero: i)
       inicio
```





¡UNA HORA UN TRIDENT MÁS Y YA LO TIENES!



Javolver somVect Rec (x, y, n, i, o, t) Jin-Junción

b) Desplegado

· Siguiendo la dedinición de som Recfinal obtenida en la generalización y siguiendo el mismo análisis de casos que en la gonción reconsiva no dinal, diseñas inmessora;

// Cabezera: EnTero sum VectRec (& Voct. x, & Vect: y, & EnTero:n, & EnTero:i,

E EnTero: sum, E EnTero: prod)

// Postcondición: X = A[1..n] A y = B[1.n] A i ≤ n A i ≥ 1 A n > 0

// Postcondición: de vuelve el valo: \$\frac{\pi}{\pi_1} \frac{\pi_2}{\pi_1} (\pi_2\pi_3) \quad \pi_1 - \pi_1]

EnTero sum\leat Rec (\xi\) Ved: \$\frac{\pi}{\pi_1} \xi\] EnTero: \$\frac{\pi_1}{\pi_1} \xi\] EnTero: \$\frac{\pi_1}{\pi_1} \xi\]

(A Tail \quad \pi_2) \quad \pi_1 \xi\]

(A Tail \quad \pi_2) \quad \pi_1 \quad \pi_2 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_2 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_2 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \pi_2 \quad \pi_1 \quad \pi_2 \quad \quad \pi_2 \quad \pi_2 \quad \pi_2 \quad \pi_2 \quad \pi_2 \qua

E EnTero: prod)

si ion entonces

devolver sum + prod. x[i].y[n-i+1]

Di-No

devolver som + prod (x[i].y[n:i+1]+(i+1). som Vet. rec (x,y,n,i+1))

gin_si

din-funcion

· Reorganización del caso general

(sum + prod. x Ei 3. y En - i + 13) + (i + 1). prod. sum Ved. rec (x, g, n, i + 1)

Se le aplica la propiedad distributiva y se reorganizan los términos, aplicando la propiedad asociativa, de manera similar al caro general.

De esta manera se obtienen los sucesores de los parámetros de inmersión.

· Función recursiva jirol (aux dependiente de la función recursiva no final)

// Cabezesa: EnTero: sumVert Rec (E Vect. x . E Vect. y . E EnTero: n . E EnTero: i, E EnTero: jun, E Entero: piod)

// Recondición: x=A[1...n] A y=B[1..n] A i=n A i=1 A n>0

// Postcondición: devuelve el valor = (x[a].y[n-a+1])

Entero: SumVet Rec (& Vect: x, & Vect: y, & Entero: n, & Entero: j & Entero: jum,

E Entero: prod)



WUOLAH

```
EnTero: sum Vectrec ( & Vect: x , & Vect: y , & Entero: n , & Entero: i,
                           E Entero: Dum, E Entero: prod)
           inicio
                si i=n entoncei
                     devolver sum + prod. x[i].y[n-i=1]
                Di-no
                    devolver (sum + mod, xCi3. y In-i+13) + (i+1) prod sum Vectrec (xy, n, i+1)
                gin.si
          din- funcion
c) Plegado
    · Sucesoses
        Duc (sun) = num + prod.x (i) y [n-i+1)
        suc (prod) = (i+1). prod
    · Version Recorsiva Final
     // Cabezera: EnTero: sum Vect Rec ( & Vect. x , & Vect. y , & EnTero: n , & EnTero: i ,
                   E Entero: sum E Entero: prod)
    / Precondición x = A[1.n] Ay=B[1.n] AiEn Aiz1 A NO
    // Postcondición: devuelve el valor = al (XCa) y cn-a+1)
    EnTaro: SumVectRec (E Vact: x, E Vect: y, E EnTero: n, E EnTero: i, E EnTero: Sum,
    ¿ EnTeio: prod)
       inicio
          si ion entonces
               devolver sum + prod. xtil. y [n-i+1]
               devolves sumVect Rec (x,y,n,i+1, sum+prod.xci) yEn.i+1), (i+1).prod)
           din-si
       Sin. Juncion
Transformación de Función Recursiva No Final (RNF) a ITerativa
 // Cabezera: EnTero: sum lier ( & Vect: x, & Vect: y, & EnTero: n, & EnTero: i)
// Precondición: X= A[1.n] Ay= B[1.n] AIEN AIZI ANDO
// Postcondición: develve el valor & al (x [a] y[n-a-1])
```

```
Entero: sumIter ( & Vect: x , & Vect: y , & Entero: n , & Entero: i)
                Van
                                    EnTero C, 7es
                inicio
                                       mientra mientr
                                                           (+ C+1
                                                             i <- i+1
                                       din-mientrai
                                        7es ( xciz.ycn-i+1)
                                        mientrai C 70 haces
                                                         ( ( -1
                                                           i (- i - 1
                                                           rest X[i] y[n-i+1] + (i+1) res
                                       fin mientral
                                       devolver res
            fin-Junción.
  Optimi ración del código
// Cabezera: Entero: sum Iter (EVect: x, E Vect: y, E Entero: n, & Entero: i)
//Piecondición: x=A[1.n] 1 g=B[1.n] 1 i=n 1 iz1 1 no
// Postcondición: devuelve el valor & al (x (a) g (n-a+1))
EnTero: som ITer ( & Vect: x, & Vect: y, & EnTero: n, & EnTero: i)
             Van
                            Entero c, res
               inicio
                              rest XCiz. yEn-i+1]
                               mientia cto haces
                                                 1es + x 2i3, y 2n-i+1) + (i+1), >0
                               Jin-mientra
                                 devolver res
                gin- Juncion
```



JNA HORA UN TRIDENT MÁS Y YA LO TIENES!

// Para obtimizar el código lo que hemor hecho ha sido eliminar el primer



```
buels e inicializar los valores de nei.
STIIIIIRA TUS MOMENTOS
                                                     Transformación de Recusiva Final (RF) a Iterativa.
                                                      Mabezera: Entero: Mamada (E Vect: x, E Vect: y, E Entero: n, E Entero: i)
                                                     // Precondición: X=A[1.n] A y=B[1.n] A i =n A i >1 A no
                                                     // Postcondición: devuelve el valor & al (xcaj ycn-a+13)
                                                     EnTero: Plamada ( & Vect: x, & Vect: y, & EnTero: n, & EnTero: i)
                                                           devolves sum ITer (x, y, n, i, O, 1)
                                                        fin- Junción
                                                    // Cabezera: Entero: sum Iter ( & Ved:x, & Vect:y, & Entero:n, & Entero: i, & Entero sum,
                                                                 E Entero : nod)
                                                    Precondición: x=A[1.n] A y=B[1.n] AiEn A i 21 A ADO
                                                    // Postcondición: devuelve el valo, & a! (X[a]. y(n-a+1))
                                                    Entero: Dimiter (EVect: x, E Vect: y, E Entero: n, E Entero: j, E Entero: Jum, & Entero: prod)
                                                      inicio
                                                         Mientra 7 (i=n) hacer
                                                           Sum + prod. ACI7. yen-i+1)
                                                           prode (i+1). prod
                                                           i (- i+1
                                                         gin-mientiai
                                                         devolver sum + prod. xcis. gen-i+13
                                                       fin- función
                                                    Optimización del código
                                                    // Cabezona: EnTero: SumIter (E Vect: x, E Vect: y, E EnTero: P, & EnTero: i, E EnTero: sum
                                                                  E Entero: Mod)
                                                   MPrecondicion: X= ACT. n) Ay= BET-n] Aisn Aist Anso
                                                   // Postcondición: devuelve el valos & al (xca).ycn-a+1)
                                                   Entero: Sum ITer (E Vect: x, & Vect: y, E Entero: n, & Entero: E Entero: Sum & Entero: prod)
                                                       inicio
                                                          JUM CO
```

niod (0

mientia rlian) hacei

sum < sum + prod. xci3. y cn-i+1) prod (i+1). prod i + i + 1 fin-mientiai devolves sum + prod xcis ything Sin-Juncion MPara optimizar la función simplemente hemor inicializado a o los parámetros de sum y prod



Ejercicio 12. · Tupla de parámetro, formale: x, y, n, i · (aso base? (x): isn · Sol (x): 1 · Suc (x) = x, y,n, i+1 · (omb(x,8): 6x[i]+6y[i].} a) Generalización. - Función sumergida: prod Sum_rec (x,y,n,i) = (6.xEi) +6.yEij). prod Sum_rec (x,y,n,i+1) - Arbel sintáctico (junción sumagida): 6 x[i] 6 y[i] · Arbol sintáctico con los parámetros de inmersión. · Función inmersora: prod Sum-recf(x,y,n,i,prod) = prod. prod Sum-rec (x,y,ni) - Valores iniciales: Elemento neutro del producto prod:1 - Llamada inicial: prodSum_rec_F(x,y,n,i,1) = 1. prodSum_rec(x,y,n,i) // Cabezera: entero prod (EVectix, Evect; y, E enteroin, E enteroi) // Precondiction: X=A[1...n] A y=B[1...n] A 1 = isn A no // Poscondición: Jevuelve el valor de no 6. (X[a]+y[a])



INA HORA UN TRIDENT MÁS Y YA LO TIENES!



in the second

```
Entero: Junción prod (Evectix, Evectig, E entero: n, E entero: i)
                                                          inicio
                                                             devolver prod Sum-rec_F(x,y,n,i,1)
STIIIIRA TUS MOMENTOS
                                                          fin Juncion
                                                          b) Desplegado
                                                          · Funcion innersona:
                                                         // Cabezera: entero: prodSom-rec_F(Evect:x, Evect:y, Ecntero:n, Eentero:i, Eentero:prod)
                                                         // Precondición: x=A[1...n] A y=D[1,...n] A 1 si En A NO
                                                         // Postcondición: develve el valos 17 6 (x[i]+y[i]):
                                                         Entero: Junción prodsom-rec-F(Eved:x, Eved:y, E entero:n, Eentero: , Eentero: prod)
                                                         inicio
                                                             si isn entoncer
                                                                 devolver 1. prod
                                                                devolves prod. (6xci7+6yci3). prodSunacc (x,y,n,i+1)
                                                             lin-si
                                                         Jin-Juncion
                                                         - Reorganización del caso general:
                                                         Usando las propriedades de la multiplicación se puede reorganizar los Términos
                                                          para obtenes los nuevos sucesores de los parámetros de inmersión.
                                                          6(prod. (xciz+yciz)). prod Sum. rec (x, y, n, i+1)
                                                         - Función recursiva final (aun dependiente de la función recursiva no final)
                                                         1/ Cabezera: entero: mod Sum nec-F(Evetix, Evetiy, Eentero:n, Eentero:i, Eentero: prod)
                                                         //Piecondición: X=A[1..n] A y=B[1..n] A 1 E i En A no
                                                         // Postcondición: devuelve el valo, II 6 (xci] y ci)
                                                         entero: junción prodsom rec. F (Evertix, Evertiy, E entero: n, E entero: i, E entero: prod)
                                                          inicio
```

Si ion entonces

Sin-si din Juncion

devolver 1. prod

devolves 6. prod (xci]+yzi]). prod Sum-rec (x,y,n,i+1)

```
c) Plegado
 -Obtenes los sucesores de los parámetros de inmersión que deben ir en la llamada flecursiva Final
    Suc (prod) = 6 prod. (xci) + geis)
 · Version reconsiva Final:
 // Cabezera: entero prodSum-rect(Evectx, Evecty, Eantero:n, Eentero:i, Eentero:prod)
// Precondición: x=A[1.-n] 1 y= B[1.-n] 1 1 = i = n 1 120
// Postcondición: devuelve el valo 1 6(xci)+yciz)
 Entero: Juncion prod Jum. recté vect x, & vect y, & entero: n, & entero: i, & entero: prod)
 inicio
    si isn enfonces
         devolver 1. prod
   si-no
         devolves prod Sum nec (x,y,n,i+1, 6 prod. (xciz+ yciz))
    gin . si
fin. Junción
- Transformación de Función Recursiva No Final a Iterativa
// Cabezon: entero prodSom. roc (Eved:x, Eved:y, Eentero:n, Eentero:i)
// Recordición: X=A[1...n] A y=B[1...n] A 1= i =n An>o
// Pastcondición: devendre el valos [] 6 (x [i] + y [i])
Entero: Junion prod Sum-rec (Evectix, Evectig, Eentero: 1) Eentero:i)
 Van
      Enlere C, 723
 inicio
     mientia 7 (i>n) haces
      C <- C+1
       i (- i+1
         i - 1-1
        7es ( 6(X[i)+y[i]).7e>
       din mientiai
       devolver res
 Sin-Juncian.
```

```
- OnTimización del código
// Cabezera: entero prodSom-rec (EvecT:x, EvecT:y, E entero:i)
// Precondition x= ACT... n] Ay=BCT...n] A 15cen A n>0
// Postcondición: devuelve el valo: 1 6(xci] + yci)
Entero : Juncian prodsum-rec (Eved: x, Eved: y, E entero: n, & entero: 1)
      entero: c, res
Micio
    itn
    705-1
    mientras c to haces
        CK-C-1
        res ( 6. (x[i]+y[i]). res
         i -1-1
     Jin-miertian
    devolver ses
 fin- Junción
- Transformación de Recupiva Final a ITeraliva.
// Cabezera: entero prod
                             (Evedix, Evedig, Evedin, E entero :1)
// Precondicion: X=A[1...n] A y=B[1...n] A 1 sisn Anso
// Postcondición: devocable el valor de si 6 (xci]+yci)
 Entero: Junction prodSum inicio (Evect: x, Evediy, Einteo:n, Einteo:i)
      devolves
 Sin-Juncion
// Cabezera: entero prod Sum. reall Evection, E voction, E entero : i, E entero in E entero in E entero in E
// Recondición: x=AC1... n3 A y= B[1..n] A 15isn Anso
// Postcondición: deuvelve el valor 1 6 (xciz+yziz)
entero: Juncion prodSum-rec (Eved:x, Eved:y, Exitero:n, Exitero:i, Exitero: prod)
inicio
    mientrani(i>n) hacen
      prod + 6. prod (xci) + yci)
      i (- i+1
```



¡UNA HORA UN TRIDENT MÁS Y YA LO TIENES!



```
gin_mientias
   devolves prod
fin- funcion
- Optimización del código
// (abedera: entro produm. rect(Evect: x, Evect:y, Eentro: r, Eentero: , Eentero: prod)
// Recondición: x=AC1-r3 n y=BC1-r3 n 1 = i = n x noso
// Postcondición: devuelve el valor II 6 (xci) + yci)
entero: juncion prod Sum-rec (E voct: x, Evoct: y, E entero n, E entero i, E entero prod)
    entero : prod
inicio
    prode 1
    mientia r(isn) hacen
      prode 6 prod (xciz+yciz)
      141-11
   Ain mientraj
 devolver prod
din-mientias.
```



WUOLAH

Ejercicio 13. Esquema. · Toplade parametros formales x = x,y,n,i · Caso base? (x) = i=n+1 · Sol(x)=0 'Suc(2) = x, y, n, i+1 · (omb (x, 3) = x [i] . y [n-1+1] + 3. prod Vects_rec a) Generalización Función sumergida = XCII. y[n-i+1] + 3 prod Vects - rec - Aibol sinTácTico= x[i] y[n-it] 3 prodVecto. sec - Arbel sintáctico con los parámetros de inversión = - Función in mersora products_rec (x,y,n,i,sum,prod) = sum + prod. prod vects_rec (x,y,n,i) Valores iniciales = Elemento neutro de la soma =0 Elemento neutro del producto: 1

```
- Clamada inicial: niod Vects-rec-Final (x,y,n,i,0,1) = 0+1 prod Vects-rec (x,y,n,i)
 - Función que realiza la llamado inicial a la función recursiva final (RF):
 // (abezera: entero: pod (E Ved: x, E Vet: y, E entero: n, E entero i)
 // Precondición: X=A[1...n] A y=B[1...n] A i=n A i>1 A n>0
// Postcondición: devodue el valor & 3x-1 (x[a].y[n-a+1])
 entero: Juncion prod(Elect: x, E vect: y, E entero: n, E enteroi)
  Inicio
      devolver prod Vects -rec. Final (x, y, n, i, 0, 1)
  Sin- Juncion
 b) Desplegado
 - Diseñai Juncion inmersora:
// (abezera: entero: prod) vects-rec-final (Evectx, Evect: y, Eentero: n, E entero: i, Eentero: sum, E entero: prod)
// Recondición: x=A[1...n] x y=B[1..n] x i≤n x i≥1 x n>0
// Abdicondición: devuelve el valo, 2 3 (x [a] · y[n-a+1])
entero: función products - rec. Final (Evect:x, Evect:y, & entero:n, E entero:j, E entero:sum, E entero:prod)
 inicio
     si i=n+1 entonce,
        devolver sum + piod. 0
        devolves sum + mod. (xci), ycn-i-1)+3. prodveds-icc (x, yin, i+1))
    din-si
din. Juncion
 - Reorganización del como general =
   Sole aplica la propriedod distributiva y la asociativa para luego reorganizarlo
  de manera similar al caso general para obtener los sucesores de los parómetros
  de inmersion.
  [sum + prod. (xci). y[n.i+1]]+[prod.3]- prodivers-rec (x,y,n,i+1)
Función recursiva final (aun dependiente de la función recursiva no final)
11 (abezera: entero: producti xc. Final (Evet:x Evet: y, Eentero: n. Eentero: i, Eentero: sum, Eentero: prod)
// Precondición. x: A[1.n] A y: B[1.n] A IEN A 121 A NO
// Postcondición: develve el valor = 3 x=1 (xcas.y[n-a+1])
```



desde un curro de verano al trabajo de tu vida.



```
entero: función products-ra-Final (Eved x, Evet; y, E entero; p, E entero: i, E entero: sum, E entero prod)
  inicio
      si i=n+1 enTonces
        devolves sum + prod. O
        devolve, [sum + prod. (x[i].y[n-i+1)]+[prod.5]. prod Vects-rec(xigin,i+1)
      din-si
 fin-función.
 C) Plegado
 - Sucesores = " succsum) = sum + prod. x[i]. y[i-n+1]
                    · suc(prod) = prod. 3
- Versión Recursiva Finals
// Coberena: entero: mod Vects rec. Final (E Vect. x, E Vect. y, E entero: n, E entero: i, E entero: sun, E entero prod
// Recondición: X: All. nJa y: B[1..n] Aisn Aiz1 Anso
// Postcondición: de vuelve de volon & 3ª-1 (xcazy[n-a+1])
entero; función producto nec Final (Evectix, Evectiy, E entero: n, E entero: 1, E entero: 20m, E entero: prod)
inicio
    Si i=n+1 entoncer
          devolver sum + prod. 0
    Ji-no
         devolves prod Vects-rec. Final (x,y,n,i+1,sum+prod. XCI]. y[n-i+1], prod. 3)
    fin-si
Jin. Junción.
- Transformación de una función Recursiva no Final a Iterativa.
// Cabetera: entero: prod Vects_rec_ITor (EVect: x, EVect: y, E entero: n, E entero:i)
// Piccondición: X=A[1...n] n y:B[1...n] n i=n n i=1 n n 20
// Postcondición: develve el valos 2 3ª-1 (x[a] y[n-a+1])
 entero: prodvects_rec_trea (EVect: x, EVect: y, E entero; n, E entero; i)
   Van
       entero c, res
   inicio
    (-0
```

```
mientia 7 (i=n+1) haces
          CE C+1
          i (- i + 1
     din-mientras
     7e> (-0
     mientia, cto hacen
        c - c-1
        ic-1-1
       Tes ( XCi). y [n-i+1]+3. res
     gin_mientrai
  devolver 7es
 Jin Juncion
* Optimización del código:
 // Cabezera: enter prod Vects_nec_ ITer ( E Vect x, E Vect y, E entero n, E entero i)
// Recondición: X=A[1...n] x y=B[1...n] xien xiz1 x nso
// Postcondición: devuelve el valor de \( \sum_{\alpha = 1} 3^{\alpha - 1} (X[\alpha] \, g[n-\alpha + 1])
entero: Junción prod Vects_rec. ITe. (E Vectix, E Vectig, E enteroin, E enteroi)
 Van
      entero cines
 inicio
    (←n-i+1
    i← n+1
    mientras c + 0 haces
        C-C-1
        i - 1-1
        Tes < XCi] y[n-i+1] + 3 res
   Jin-mientias
  devolver res
Sin-Juncion.
```



deja de imaginar que trabajarás de lo tuyo hazlo posible.



```
· Transformación de Recosiva Final a ITerativa:
// Cabezera: entero: prod (Evatix, Evecting, Eenteroin, Eenteroit)
// Piccondición: X=ACL.n) n y=B[1..n] n ien n i=1 1 no
1/ Postondición el valor 2 3ª·i (xcaz. y[n-a+1]
entero: función prod (Exactic, Exedig, Eintero: n, Eintero: 1)
   devolver productioner_ I (x,y,n,i,0,1)
 fin- Juncion
// Cabezcia: entero: proditecturico. I (Evect: x, Evect: y, E entero: n, E entero: i, E entero: son, E entero: prod)
// Precondición: X=A[1...n] 1 y=B[1...n] 1 i=n 1 i=11 n20
// Pastcondición: devoelve el valor à sa-i (xca) y [n-a+1])
Entero: Juncion prod Veda-lec-I (Eved: x, & ved: y, & entero: n, & entero: i, & entero: sum, & entero: prod)
 Inicio
    mientiai 2 (i=n+1) hacei
       Sum + sum + piod xi3 y[n-i+1]
       prod + prod. 3
      14141
    Sin-mientras
    devolver som + prod. o
fin Juncion
*Optimización del código:
// Cabesera: entero: products new [ (E ved: x, E Ved: y, E entero: n, E entero: i, E entero: jum, E entero: prod)
// Precondición: x:ACI..nJn y:B[1.n]n isn n i >1 nn>0
// Poskondicion: devuelve el valo & 30-1 (x [a] . y[n-a=1])
entero: Juncian products reclEbect: x, Eved: y, Eentero: n, Eentero: i, Eentero: jum, Eentero: prod)
Van
     entero: sum, prod
Micio
      JUM FO
      prode 1
     Mientian 7 (1=n+1) hacer
      Some sum + prod. XEIT. yEn-i+17
      prod + prod. 3
     devolves sum + prod. o
 fin-funcion
```