Boletín de Problemas de Metodología y Tecnología de la Programación, IS04

Curso 2006/2007

16 de octubre de 2006

Capítulo 2: Conceptos Básicos

1. Se dispone de dos variables, `x` e `y`. No importa de qué tipo son. Hay que escribir un algoritmo que permita que intercambien sus valores.

2. Sabiendo que `x`, `y` y `z` son variables del mismo tipo, con valores distintos dos a dos, analizar la siguientes secuencias de instrucciones e indicar en cuáles varía el resultado si se invierte el orden de ejecución. Por ejemplo, si varía o no el resultado si se ejecuta la secuencia

$$\begin{array}{l} x = y \\ z = x \end{array} \quad \text{o la inversa} \quad \begin{array}{l} z = x \\ x = y \end{array}$$

$$(a)\ \begin{array}{l} x = y \\ z = y \end{array} \qquad (b)\ \begin{array}{l} x = y \\ y = x \end{array} \qquad (c)\ \begin{array}{l} x = x + y \\ z = x + y \end{array} \qquad (d)\ \begin{array}{l} y = x + y \\ z = x + z \end{array}$$

3. Escribir un algoritmo que, dados tres valores de tipo real, permita calcular su media aritmética.

4. Diseñar un algoritmo que permita pasar una cantidad de tiempo expresada en segundos al formato horas, minutos y segundos.

5. Sabiendo que el precio de venta al público de un artículo engloba su precio de coste y las siguientes repercusiones,

   - Gastos de comercialización, estimados en un 20 % sobre el precio de coste,
   - Gastos Generales, personal, publicidad ... estimados en un 75 % sobre el precio de coste,
   - Impuestos, que repercuten un 1.5 % sobre el precio de coste,
   - Y al resultado del precio obtenido, añadirle un 0.7 % sobre el precio resultante como aportaciones a ONGs,

   escribir un algoritmo que, conociendo el precio de coste, permita determinar el precio de venta al público final.

6. Escribir un algoritmo que permita determinar la solución de una ecuación de primer grado, `ax + b = c`.

7. Escribir un algoritmo que, dado un valor de tipo real, permita calcular el área del cuadrado cuyo lado tiene esa medida.

8. Escribir un algoritmo que, dado un valor de tipo real que representa el área de un círculo, permita calcular el perímetro de su circunferencia.

Capítulo 3: Programación Estructurada

1. Sean a y b dos variables enteras con valores 3 y 5, respectivamente. Indicar si las siguientes condiciones son ciertas o falsas:

```
a + b < 10
!((a + b) <= 10)
a >= 2 && b >= 5
a < 10 || (a > 0  &&  b < 0)
a > 0 && (b < 2 || !(b > 5))
a < 5 && b < 8
!(a >= 5 || b >= 8)
```

2. Sean A, B, C y D variables enteras. Se considera la secuencia

```
if (((A > 0) || (B > C)) && ((D > A) || (D < 5))) {
   A = 0 ;
   D = B + C ;
}
else {
   C = A - B ;
   if (C < 0){
      D = -D ;
   }
   B = 0 ;
}
```

   Determinar los valores finales para las variables, sabiendo las siguientes posibilidades de valores iniciales:

   a) A = 5, B = 3, C = 4, D = 6

   b) A = -1, B = 3, C = 4, D = 3

   c) A = -1, B = -2, C = 4, D = 3

3. Sean X, Y y S variables enteras. Dada la secuencia

```
if (X < 0)
  S = 1 ;
else
    if (Y > 0)
        S = 1 ;
    else
       S = 0 ;
```

   escribir una secuencia equivalente, sin anidamiento de condicionales y sin repetir la instrucción S = 1.

4. Dada la secuencia

```
if (p && q)
     a ;
else
     s ;
```

donde `p` y `q` son predicados elementales y `a` y `s` son acciones, escribir una secuencia equivalente, en la que sólo se evalúe cada vez un predicado elemental.

5. ¿Son equivalentes las tres secuencias siguientes?

`s=0;`
`if (p) {`
`  s=3;`
`}`
`else {`
`  if (!q) {`
`  s=3;`
`  }`
`}`
	`s=0;`
	`if (!(!p&&q)) {`
	`  s=3;`
	`}`
		`s=0;`
		`if (p || !(q)) {`
		`  s=3;`
		`}`

6. A, B y C son tres variables enteras. Escribir una secuencia de instrucciones que determine cuál es la variable de mayor valor y devuelva el resultado en una variable MAX.

7. Dados tres valores enteros A, B y C, escribir una secuencia de instrucciones que los ordene en orden creciente.

8. Las siguientes secuencias hacen lo mismo, pero ¿cuál es mejor y por qué? (nota: los valores de a, b y c son distintos dos a dos.)

`if ((a>b) && (a>c)) {`
`  max = a;`
`}`
`if ((b>a) && (b>c)) {`
`  max = b;`
`}`
`if ((c>a) && (c>b)) {`
`  max = c;`
`}`
	`if ((a>b) && (a>c)) {`
	`  max = a;`
	`}`
	`else {`
	`  if ((b>a) && (b>c)) {`
	`  max = b;`
	`  }`
	`  else {`
	`  if ((c>a) && (c>b)) {`
	`  max = c;`
	`  }`
	`  }`
	`}`

9. Dado el siguiente fragmento de código, reescribirlo utilizando un único condicional, que además esté simplificado al máximo:

```
if (b < 10) {
    if (b >= 5)
        a = a*2;
    if (b < 5) {
       if  (b >= 0)
           a = a*2;
    }
}
```

10. Escribir una secuencia de instrucciones que determine a cuánto asciende la factura de la luz de un abonado. Para ello, se conoce AI, antiguo índice y NI, nuevo índice, que representan los valores leídos en el contador de la luz. El resultado se quiere sobre la variable IMPORTE, que se calcula sabiendo que un abonado

    a) Paga 5 euros por gastos fijos de contrato,

    b) El consumo se determina por tramos: los primeros 100 Kws, a 5 céntimos el Kw; los 150 Kws siguientes, a 3 céntimos el Kw; si el consumo excede de 250 Kws, esa fracción se cobra a 2 céntimos el Kw.

11. Reescribir la siguiente secuencia en forma de un único condicional regido por un predicado compuesto, tan simple como sea posible. Justificar todos los cambios realizados.

```
if (a > b)
    if (b-a > 0)
         a = a + 1;
    else
        if (x > a)
              if (x > b)
                   if ((2*x) > (a+b))
                        x = 0;
                   else
                      if (x < 100)
                            x = 1;
```

12. Simplificar al máximo las siguientes secuencias, justificando las acciones realizadas, sabiendo que las variables X, Y, I y K, son enteras:

```
if (X>0) {
   I=I+2;
   K=0;
   Y=2*X;
}
else {
   I=I+2;
   K=0;
   Y=0;
}
```

```
if (X==3) {
   X=Y;
   Y=0;
   I=X+Y;
}
else {
   X=Y;
   Y=K*2;
   I=X+Y;
}
```

13. Las secuencias siguientes ¿hacen lo mismo?

```
s=0;
x=a;
s=b;
if (x>=c) {
   x=x%c;
}
else {
   x=x+1;
}
```

```
s=b;
if (x>=c) {
   x=a;
   x=x%c;
}
else {
   x=a;
   x=x+1;
}
```

14. ¿Verdadero o Falso (justificar)?: La ejecución de cada una de las dos secuencias siguientes es **exactamente** igual.

```
if (v<V1) {
   <inst_1>;
}
else {
   if (v>V2) {
      <inst_2>;
   }
}
```

```
if (v<V1) {
   <inst_1>;
}
if (v>V2) {
   <inst_2>;
}
```

15. Antes de ejecutarse el siguiente fragmento de código se cumple el predicado

/* j < N `and` acum == 0 */.

Analizar el fragmento e indicar que ocurriría al ejecutarlo:

```
i=j;
while (i<=N) {
   if (j<i) {
      i=i-1;
   }
   else {
      i=i+1;
   }
   acum=acum+i*i;
}
```

16. Simplificar al máximo la secuencia siguiente, justificando adecuadamente todos los cambios realizados:

```
/* x e y son variables reales */
/* n, entero, n=N>1 */
y = x;
for (i=1; i<=n; i=i+1){
   if (i<n) {
      y = i*x;
   }
   else {
      y = 2*x;
   }
}
```

17. Dado el siguiente esquema condicional:

```
if (<cond1>){
  if (<cond2>){
    <inst1>;
  }
  else {
    if (<cond3>){
      <inst2>;
      <inst3>;
    }
  }
}
else {
  if (<cond3>){
    <inst2>;
    <inst3>;
  }
}
```

¿Es posible substituirlo por un único condicional? ¿Por qué? Dar una versión equivalente con el menor número posible de condicionales anidados.

18. Simplificar al máximo el siguiente condicional, sabiendo que `a`, `b`, `c` y `d` son enteros:

```
a = 3;
if (a < 3) {
   b = 2*a;
}
else {
   if (b > c) {
      a = a + b;
      b = b + c;
      c = c + d;
   }
   else {
      a = a + b;
      b = b + c;
      c = 2*c;
      c = c + d;
   }
}
```

19. ¿Verdadero o Falso (justificar)?: "Los dos condicionales siguientes siempre producirán los mismos resultados",

```
if (p) {
   <inst_1>;
}
else {
   <inst_2>;
}
```

```
if (p) {
   <inst_1>;
}
if (!p) {
   <inst_2>;
}
```

20. ¿Verdadero o Falso (justificar)?: "Los dos condicionales siguientes son equivalentes",

```
if (p) {
   for(j=k;j<=q;j++) {
      b[i]=a[j];
      i++;
   }
}
else {
   for(k=j;k<=m;k++) {
      b[i]=a[k];
      i++;
   }
}
```

```
if (p) {
   for(j=k;j<=q;j++) {
      b[i]=a[j];
  }
}
else {
   for(k=j;k<=m;k++) {
      b[i]=a[k];
   }
}
i++;
```

21. Simplificar al máximo el siguiente condicional:

```
if (a>=b) {
   a=a/2;
   if (a>=c) {
      a=a/2;
   }
   else {
      a=a/2;
   }
}
else {
   a=a/2;
}
```

22. Dado el siguiente bucle, en el que `i` y `t` son enteros,

```
/* n es un entero, n>1 */
t=1;
i=0;
while (t<n) {
  t=t*2;
  i=i+1;
}
```

indicar para cada uno de los predicados siguientes si puede ser o no un invariante y por qué:

a) { 2^i<= n && n>0 }

b) { n > 0 && i<=log_2n }

23. Simplificar al máximo la siguiente secuencia de código:

```
/* Pre: j==V1, V1 > 0 */
   int aux, i;
   aux=0;
   i=0;
   while (i<=j) {
      if (i>=j) {
         aux=aux+(i*j);
         i=i+1;
      }
      else {
         aux=i*j;
         i=j;
      }
   }
   printf(" %d\n", aux);
}
```

24. Dada la variable entera NUM, que contiene un valor n ≥ 1, elaborar una secuencia de instrucciones que determine la suma de los n primeros números naturales.

25. Dado un entero i, i > 0, escribir una secuencia de instrucciones que nos devuelva el menor entero n, tal que $2^n > i$.

26. Escribir una secuencia de instrucciones que determine el cociente y el resto de la división entera de dos números enteros A y B. (Nota: mediante restas y sumas).

27. Escribir una secuencia de instrucciones para calcular x^n, siendo n un número entero.

28. Dado un valor $n \geq 1$, elaborar una secuencia de instrucciones que calcule $n!$.

29. Escribir un programa que haga lo siguiente: Se irán leyendo valores reales por teclado y el objetivo es detectar secuencias.

    Una secuencia es una serie de números consecutivos iguales. Por ejemplo, en la siguiente serie hay 6 secuencias:

    `0.25, 0.5, 0.5, 1.75, 0.1, 0.1, 0.1, 0.1, 0.15, 0.8, 0.8, 0.0`

    Cuando se lea el valor `0.0` finaliza la introducción de números y se deberá escribir cuántas secuencias se han detectado.

30. La sucesión de Fibonacci se obtiene de acuerdo a

$$fibonacci(n) = \begin{cases} 1 & si\ n = 1, \\ 1 & si\ n = 2, \\ fibonacci(n-1) + fibonacci(n-2) & si\ n > 2 \end{cases} .$$

Escribir una secuencia de instrucciones que calcule el número de Fibonacci asociado a un entero n.

31. Escribir una secuencia de instrucciones que permita calcular el término n de la sucesión

$$\begin{aligned} &s_0 = 1 \\ &s_1 = 1 \\ &s_2 = 1 \\ &s_n = s_{n-2} + s_{n-3},\ n \geq 3. \end{aligned}$$

Es decir, los tres primeros términos son 1, 1, 1; el siguiente se calcula sumando los términos penúltimo y antepenúltimo. Si, por ejemplo, n=12, los términos a calcular serían: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21,... y habría que devolver el valor 21.

32. Escribir una secuencia de instrucciones que calcule el término k-ésimo de la sucesión

$$\begin{aligned} &T(0) = 1 \\ &T(1) = 1 \\ &T(n) = T(n-1) + (n-1) * T(n-2),\ n \geq 2. \end{aligned}$$

33. Escribir una secuencia de instrucciones que convierta una variable entera en otra, también entera, en la que el entero original aparezca del revés. Por ejemplo, el entero 357 se debe convertir en el 753.

34. Escribir una secuencia de instrucciones que dado un número entero lo reduzca a la suma de sus dígitos, de forma que el resultado sea un número de un sólo dígito. Por ejemplo, dado el número 13674891,

$$13674891 \rightarrow 1+3+6+7+4+8+9+1 = 39 \rightarrow 3+9 = 12 \rightarrow 1+2 = 3$$

el resultado pedido es 3.

35. Escribir una secuencia de instrucciones que calcule la exponencial de un número real `a`, de acuerdo a la serie,

$$e^a = \sum_{n=0}^{\infty} \frac{a^n}{n!} = 1 + a + \frac{a^2}{2} + \frac{a^3}{3!} + \frac{a^4}{4!} + \ldots + \frac{a^n}{n!} + \ldots$$

Aproximar el resultado hasta que para algún `k` se cumpla que $a^k/k! \leq 10^{-5}$.

36. Escribir una secuencia de instrucciones que calcule la exponencial de `a` según la serie del problema 35, pero aproximando hasta que k=20.

37. La función seno se puede calcular de acuerdo a la serie

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

mientras que para calcular la función coseno se utiliza la serie,

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Dado un real, `x`, escribir una secuencia de instrucciones que calcule el seno y el coseno de `x` simultáneamente utilizando estas series y de la forma más eficiente posible. Se considerará una aproximación válida que $|\sin^2 x + \cos^2 x - 1|$ sea menor que 10^{-6}.

38. Escribir una secuencia de instrucciones que permita determinar si un número es perfecto. Un número es perfecto si es resultado de la suma de sus divisores propios; por ejemplo, el 28 tiene como divisores propios al 1, al 2, al 4, al 7 y al 14, y se cumple que

$$28 = 1 + 2 + 4 + 7 + 14.$$

39. Escribir una secuencia de instrucciones que indique si dos números son amigos. Dos números son amigos si la suma de los divisores del primer número (excluido él) es igual al segundo número y viceversa, es decir, la suma de los divisores del segundo número (excluido él) es igual al primer número.

    Por ejemplo, 220 y 284 son amigos. Los divisores de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110, que suman 284 y los divisores de 284 son 1, 2, 4, 71 y 142, que suman 220.

40. Un número odioso es un número que, cuando se pasa a base 2, tiene un número impar de `1`'s. Por ejemplo, el 1, el 2 (`10`), el 4 (`100`), el 7 (`111`), el 8 (`1000`), el 11 (`1011`), ...

    Hay que escribir una secuencia de instrucciones que, dado un valor entero K, devuelva cuántos números odiosos hay entre 1 y K.

41. Se dice que un número es apocalíptico si contiene entre sus dígitos la secuencia 666. Por ejemplo, el 16667234, el 6660, el 34666, el 66666666, el 10266663... El 1346786956, por ejemplo, NO lo es: los '6' no forman secuencia.

    Escribir una secuencia de instrucciones que, dado un número entero, indique si dicho número es o no es apocalíptico.

42. Escribir una secuencia de instrucciones que permita calcular el máximo común divisor de dos números, basándose en el método de Euclides:

$$MCD(u,v) = \begin{cases} u & si\ u = v, \\ MCD(u, v-u) & si\ u < v, \\ MCD(u-v, v & si\ u > v \end{cases} .$$

43. El método de multiplicación de enteros a la rusa se puede enunciar de la siguiente forma: el multiplicando se divide (división entera) entre 2 hasta que el resultado sea 1; y tantas veces como se divide el multiplicando se multiplica por 2 el multiplicador. El resultado se obtiene al sumar todos los números que aparecen en la columna del multiplicador que correspondan a un número impar en la columna del multiplicando. Por ejemplo,

$$45 * 19 = 855$$

45	19
22	38
11	76
5	152
2	304
1	608

$$19 + 76 + 152 + 608 = 855$$

    Escribir una secuencia de instrucciones que permita multiplicar dos números mediante este método.

44. Dados dos objetos variables de tipo entero, `x` e `y`, tales que sus valores son positivos y que el valor de `x` es mayor o igual que el valor de `y`, y dado el bucle,

```
/* x>=y && y>0 */
v=0;
while (y>0) {
   x=x-1;
   y=y-1;
   v=v+1;
}
```

¿cuál de las siguientes expresiones elegirías como candidato a invariante? ¿Por qué?

a) { v == x - y }

b) { x >= y && y >= 0 }

45. Dado el bucle

```
/* P>0 */
k=0;
m=P;
while (m>1) {
   m=m/2; /* División Entera */;
   k=k+1;
}
```

¿cuál de los siguientes predicados será un invariante del bucle? Justificar.

a) $\{2^k <= P \;\&\&\; P > 0\}$

b) $\{k = log_2 P \;\&\&\; P > 0\}$

46. Dado el siguiente bucle, en el que `i`, `n` y `N` son enteros:

```
i=0;
n=N;
while (n>0) {
  i=i+1;
  n=n/10;
}
```

de entre los dos predicados siguientes ¿cuál es un invariante? ¿Por qué?

a) { n == N/(10^i) }

b) { n >= 0 && i==log_{10}N }

47. Dado el siguiente bucle, en el que `resta`, `x` e `y` son enteros:

```
resta=0;
while (x > y) {
  x = x-1;
  resta = resta+1;
};
```

de entre los dos predicados siguientes ¿cuál es un invariante? ¿Por qué?

a) { resta <= | x - y |}

b) { x >= 0 && y >= 0 && resta == x-y }

48. Si podéis, echadle un vistazo a la introducción del libro “The Science of Programming” de David Gries, *Why Use Logic?*.

    Son tres páginas muy sencillas de leer (aunque esté escrito en inglés) y con un ejemplo muy simple, describe como dos programadores van descubriendo la utilidad de incorporar asertos en su código.

Capítulo 4: Introducción al Diseño Descendente

1. Modificar el algoritmo `lecturaCorrecta()` para que sólo admita valores válidos de ángulos de un triángulo: tal y como está definido en el ejemplo, admitiría como valores 180, 0 y 0 (ó 90, 90 y 0) que suman 180.0, pero que no son válidos para definir un triángulo.

2. Modificar el programa `geometriaPlana.c` de forma que el usuario pueda repetir el proceso cuantas veces quiera. Para acabar, dispondrá de una cuarta acción, `finalizar`. Realizar la modificación atendiendo a las acciones no primitivas que se verán afectadas. Volver a especificar todas las acciones no primitivas afectadas.

3. Modificar el programa `geometriaPlana.c` de forma que el usuario pueda calcular un área o bien dando el valor del radio o los lados, o bien indicando los puntos que delimitan radio o lados, entendidos como segmentos. Realizar la modificación atendiendo a las acciones no primitivas que se verán afectadas. Volver a especificar todas las acciones no primitivas afectadas.

4. Enriquecer el módulo `geometria.c` añadiendo las operaciones que permitan calcular el área de un triángulo y la longitud de una circunferencia. Modificar de forma adecuada el fichero `geometria.h` para que las nuevas acciones sean utilizables. Especificar las nuevas acciones.

5. Utilizar el módulo `geometria` de forma que se pueda utilizar en la definición y diseño de un nuevo módulo `geometriaEspacial` que permita calcular el volumen de un cono, el volumen de un cilindro, el volumen de un tetraedro, el volumen de un pirámide de base rectangular y el volumen de un paralelepípedo.

6. Utilizar la acción `maxComDiv()`, para realizar un programa en el que un usuario podrá calcular el máximo común divisor de una serie de números que irá introduciendo por teclado. La serie debe tener al menos dos números y finalizará cuando el número introducido sea 0. Sólo se permitirá leer números positivos (y el 0 final, evidentemente). Especificar el algoritmo.

7. El siguiente algoritmo:

```
int X(int a, int n){
  int y, i;

  y=1;
  for(i=1; i<=n; i++){
     y=y+a*a;
  }
}
```

   ¿calcula $\sum_{i=1}^{n} a^2$? Razona la respuesta.

8. Escribe todo lo que se podría leer en la pantalla del ordenador al ejecutar el siguiente fragmento de programa, sabiendo que al ejecutarlo se dará a `num` el valor `182745`:

```
#include <stdio.h>

int funcionAux(int numero);
int main() {
  int num, res, aux;

  scanf("%d", &num);
```

```
  res = 0;
  aux = 1;
  while ((num/aux)>0) {
    printf("Printf (1): %d\n",funcionAux(num/aux));
    printf("Printf (2): %d\n",num);
    aux = aux * 10;
    res = res + funcionAux(num/aux);
  }
  printf("Printf (3): %d\t %d\n", num, res);
}
int funcionAux(int numero) {
   int res;
   res = 0;
   while (numero>0) {
      if (numero %2==0)
         res=res+numero %10;
      numero=numero/10;
   }
  return res;
}
```

9. Escribir una función o procedimiento (**justificar la elección**) que, dado un valor entero indique si ese valor es o no es una **potencia completa**.

   Se dice que un entero `n` es una *potencia completa* si se cumple que existe un entero `p` que es un **divisor primo** de `n`, y, además, p^2 también es divisor de `n`.

   Por ejemplo, el `81` sería una potencia completa, ya que `3` es divisor de 81, `3` es primo y, además, 3^2 es `9`, que también es divisor de `81`.

   Se debe indicar cuál es la precondición y cuál la postcondición. Además, para considerar correcta la respuesta, el código debe escribirse utilizando de forma adecuada la función, que YA ESTÁ DEFINIDA, `esPrimo`, que responde al prototipo,

   ```
   boole esPrimo(int n);
   /* Pre: n, entero positivo */
   /* Post: devuelve cierto si n es primo,
         falso si no lo es */
   ```

10. Sabiendo que YA ESTÁN DEFINIDAS las funciones `esPrimo` y `elevar`, que responden a los prototipos,

    ```
    boole esPrimo(int n);
    /* Pre: n, entero positivo */
    /* Post: devuelve cierto si n es primo, falso si no lo es */

    int elevar(int k);
    /* Pre: k, entero positivo */
    /* Post: devuelve 2^{k} */
    ```

    y suponiendo YA DEFINIDO el tipo `boole`, se pide lo siguiente:

    a) Escribir la función `numDivPrimos`, que debe responder al prototipo,

    ```
    int numDivPrimos(int m);
    /* Pre: m, entero positivo */
    /* Post: devuelve el numero de divisores primos de m */
    ```

En la definición de esta función DEBE utilizarse correctamente la función `esPrimo`.

b) La α-imagen de n es $\alpha(n) = 2^{T-1}$, siendo T el número de divisores primos de n.
Escribir un programa que lea una secuencia de valores enteros por teclado y, por cada valor leído, calcule su α-imagen y la muestre por pantalla. Cuando se lea el valor `0` finaliza la introducción de datos.
Se DEBE razonar cuáles de entre las funciones definidas en el enunciado y en el apartado anterior son útiles en el desarrollo del programa y USARLAS convenientemente en el diseño del programa.

11. El método de la bisección (ver figura 1) permite encontrar un cero de la función $y = f(x)$, continua en el intervalo $[a, b]$ y tal que $f(a)$ y $f(b)$ tienen distinto signo ($(f(a)f(b) < 0)$.

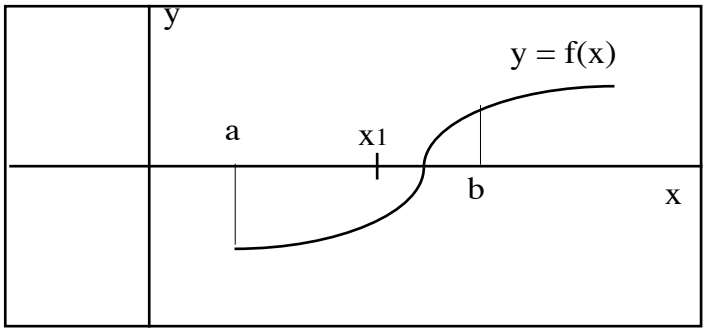


Figura 1: Método de la bisección

El método consiste en dividir el intervalo en dos partes iguales. Si llamamos x1 a la mitad del intervalo, se comprueba el signo de $f(x1)$: si $f(x1)$ tiene el mismo signo que $f(a)$ se continúa la búsqueda en el intervalo $[x1, b]$ y si $f(x1)$ tiene el mismo signo que $f(b)$, la búsqueda se continúa en el intervalo $[a, x1]$. Se repite el proceso para el intervalo resultante, hasta que se llega a que f vale 0 ó que la longitud del intervalo es menor que un cierto valor, epsilon.

Suponiendo que existe una acción no primitiva ya definido,

`float` calculaFuncion (`float` x);

que calcula el valor de la función f en el punto x, escribir un algoritmo que resuelva el cero de una función $f(x)$.

12. Un número entero N se dice que es **curioso** o **automórfico**, cuando N^2 acaba en N. Por ejemplo:

- el 5, $5 \times 5 = 2\mathbf{5}$,
- el 6, $6 \times 6 = 3\mathbf{6}$,
- el 25, $25 \times 25 = 6\mathbf{25}$,
- el 376, $376 \times 376 = 141\mathbf{376}$,
- el 109376, $109376 \times 109376 = 11963\mathbf{109376}$,...

a) Escribir una función que determine si un número es o no es curioso. La cabecera debe ser:

```
boole esCurioso (int n)
```

Hay que indicar la precondición y la postcondición, además del cuerpo de la función.

b) Escribir un programa que use la función del apartado anterior para determinar cuántos números curiosos pares y cuántos números curiosos impares hay entre 1 y un valor entero K, que se leerá de teclado.

13. Diremos que dos números enteros positivos `n` y `m` están **liados** si al pasarlos a binario la cantidad de `1`'s de `n` es igual a la cantidad de `0`'s significativos de `m` y la cantidad de `0`'s significativos de `n` es igual a la cantidad de `1`'s de `m`.

    Por ejemplo, el número 50 (110010) está liado con: el 35 (100011), el 37 (100101), el 38 (100110), el 41 (101001), el 42 (101010), el 44 (101100), el 49 (110001), el 52 (110100) y el 56 (111000).

    a) Escribir una función que determine si dos números están o no liados, indicando su precondición y su postcondición. La cabecera debe ser:

    ```
    boole liados (int n, int m)
    ```

    b) Escribir un programa que use la función del apartado anterior para determinar cuáles son los números liados con uno dado, que se leerá de teclado.

    Nota: Dado `n` ¿cuál es el rango de los posibles candidatos a estar liados con él?

14. ¿Cuál es el dominio de definición del siguiente algoritmo? Justifica la respuesta.

```
char mayuscula(char c) {

  return c-32;
/*Post:devuelve c en mayúsculas*/
}
```

    NOTA: Para transformar 'z' (código ASCII 122) en 'Z' (código ASCII 90), por ejemplo, basta con restarle 32. Los códigos ASCII de las letras mayúsculas van de 65 a 90.

15. El siguiente algoritmo determina si en un número aparece o no un determinado dígito ¿Cuál es su dominio de definición? Justificar la respuesta.

```
boole contieneDigito (int num, int digito) {
  int n;

  n = num;
  while ((n>9) && (n%10!=digito)) {
    n=n/10;
  }

  return (n%10==digito);
}
```

16. Simplificar al máximo la siguiente función y especificarla:

```
int simplFuncion(int y, int z, int v) {
   int i, w, x;

   x = 0;
   if (x > 1) {
      x = 2;
```

```
   }
   else {
      x = 1;
   }
   if (x == 1) {
      x = y;
      x = (x+z)/2;
      w = x ;
      for (i=1;i<=((w)/1)+1);i++) {
          v = v*x+v;
      }
      v = v/w;
   }
   else {
      x = z;
      x = (x+z)/2;
      w =  x;
   }

   return w;
}
```

17. Simplificar al máximo este procedimiento, comentando brevemente todas las simplificaciones realizadas (Nota: N > i > 0).

```
void simplProcedimiento(int N, float b, int *i, float *a) {
    int j,k;
    float w,v;

    j = *i;
    while (j <= N) {
      v = j;
      *a = b + (j-i);
      w = sqrt(((*a)*(*a) + b*b)/2);
      while ((*a < b) || (j == *i)) {
        v = w - b;
        *a = w;
        *i = *i + 1;
      }
      if (v >= 0) {
         for (k=i; k<=j+1; k++) {
             a = a + k;
             *i = j;
         }
      }
      else {
         for (k=i; k<=j+1; k++) {
             v = w + b;
         }
      }
      j = j + 1;
   } /* fin del while (j <= N) */
}
```

18. Dado el siguiente algoritmo, que calcula el resto de la división entera, ¿cuál es su dominio de definición?

```
int resto (int numer, int denom) {
/* Pre: ..... */

  while (numer >= denom)
    numer = numer - denom;
  return numer;
/* Post: devuelve el resto de la división
entera entre numer y denom */
}
```

19. Se pretende escribir una función que resuelva la conjetura de los capicúas: dado un entero positivo, hay que indicar cuántas veces se debe repetir el proceso de sumarlo al valor obtenido al invertir sus dígitos, antes de obtener un resultado capicúa.

    Indicar cuáles son los 4 errores (**no sintácticos**) cometidos al desarrollar la siguiente función, justificando para cada error el porqué:

```
int conjetura (int num) {
/* Pre: num contiene un valor positivo */
   int veces,aux,suma=0;

   veces=0;
   aux=num;
   do {
   /* ... hasta llegar a un capicua */
      while ((aux/10) > 9) {
         suma=(suma*10)+(aux%10);
         aux=aux/10;
      }
      suma=(suma*10)+aux;
      /* suma vale num invertido */
      /* si el número no es capicúa, repito */
      if (num!=suma) {
         veces=veces+1;
         num=num+suma;
      }
   } while (num!=suma);

/* Post: devuelve el número de veces que se */
/* repite el proceso */
}
```

Capítulo 5: Estructuras de Datos Estáticas

1. Definidos

```
int i, aux, n;
int v[10];
```

realizar una traza del siguiente fragmento de código:

```
for (i=0; i<n; i=i+2) {
  aux=v[i];
  v[i]=v[i+1];
  v[i+1]=aux;
}
```

sabiendo que `n=10` y `v=(6, 2, 5, 1, 4, 5, 6, 3, 7, 6)`.

2. Definidos

```
int i, n;
float v[10];
```

realizar una traza del siguiente fragmento de código:

```
for (i=0; i<(n/2); i=i + 1) {
  v[i]=v[n-(i+1)];
  v[n-(i+1)]=v[i+1];
}
```

sabiendo que `n=10` y `v=(6.5,2.0,5.1,4.1,7.5,5.0,6.25,3.1,7.8,6.0)`.

3. Dada la siguiente secuencia de instrucciones:

```
cont=0;
suma=0;
for (i=0; i<n; i=i+1){
    if (v[i]>n1){
        if (v[i]<n2){
            cont=cont+1;
            suma=suma+v[i];
        }
    }
}
```

se pide lo siguiente:

   a) Convertir la secuencia anterior en una función o procedimiento, es decir, darle una cabecera, indicando cuál o cuáles son los parámetros de entrada e indicando cuál o cuáles son los resultados que devuelve.

   b) Establecer la precondición, es decir, describir bajo qué condiciones dicha secuencia será correcta (funciona correctamente y produce algún resultado).

   c) Establecer la postcondición, es decir, indicar qué calcula.

4. Dado el siguiente fragmento de código:

```
blancos = 0;
varios = 0;
contLetra = 0;
i=0;
while (cad[i] != '\0'){
  if ((cad[i] == car)||(cad[i] == car-32)){
    contLetra = contLetra + 1;
  }
  else if (cad[i] == ' '){
    blancos = blancos + 1;
  }
  else {
    varios = varios + 1;
  }
  i = i+1;
}
porcentaje=contLetra/i;
```

a) Convertir la secuencia anterior en una función o procedimiento, es decir, indicando qué objetos deberían ser datos, cuáles resultados y cuáles variables propias del proceso, darle una cabecera, hacer la declaración de variables y reescribir el código para asegurar que el resultado o resultados se devolverán adecuadamente.

b) Establecer la postcondición, es decir, indicar qué calcula.

c) Establecer la precondición, es decir, describir bajo qué condiciones dicha secuencia será correcta (funciona correctamente y produce resultados correctos).

5. ¿Verdadero o Falso (justificar)?:

   "El predicado `(i==0 || i==N || v[i]==c)` es un invariante del siguiente bucle:"

```
int buscar (float v[], int N, float c) {
/*Pre: v=vector d tamaño N, c=elem. a buscar*/
  int i=0;

  while (v[i]!=c && i<N)
    i++;
  return i;
}
/*Post: devuelve índice [0..N-1] de la posi-*/
/*ción de c en v si lo encuentra, o N si no */
```

6. Escribe todo lo que se podría leer en la pantalla del ordenador al ejecutar el siguiente fragmento de programa, sabiendo que:

   - se han definido estos dos tipos

```
typedef struct {
   char nombre[N];
   float s1, s2, s3;
} tSaltador;
```

```
typedef struct {
   char nombre[N];
   float marca;
} tClasificado;
```

   - Es posible que el siguiente procedimiento tenga instrucciones inútiles. Elimina cualquier cosa que te parezca que sobra, justificando cada cambio, de forma que quede una versión lo más simple posible del procedimiento. Y, por supuesto, que siga produciendo el mismo efecto:

```
void chungoChungo(int a[], int n) {
/* Pre: n > 0 */
/* Se supone definido el tipo boole */
   int i,j,k;
   boole chivato;
   i=0;
   while (i<n){
      chivato=falso;
      j=i+1;
      while ((j<n)&& !(chivato)){
         chivato=chivato&&(a[i]==a[j]);
         j=j+1;
      }
      if (j==n){
         a[i]=0;
      }
      else {
         for(k=j;k<n;k=k+1){
            a[k]=a[j]+a[i];
            a[k]=2*a[i];
         }
         if (a[n]<a[i]){
             a[n]=0;
         }
         else{
             a[i]=i;
         }
      }
      i=i+1;
   }
}
```

- el vector `v` es un vector de `TAM=6` elementos de tipo `tSaltador`, cuyos valores son:

nombre	"J.Lino"	"Fiona"	"Yago"	"Niurka"	"Concha"	"Iván"
s1	6.75	0	0	6.38	6.69	6.05
s2	7.1	0	7.06	0	0	6.23
s3	7.32	6.37	7.01	0	6.17	7.08
	0	1	2	3	4	5

- que `v2` es un vector, también de `TAM=6`, y con elementos del tipo `tClasificado`.

```
#include <string.h>
#define N 10
#define TAM 6
#define TOPE 6.5

... /* Las definiciones de tipo */

int main() {
  int indice,i;
  float max;
  tSaltador v[TAM];
  tClasificado v2[TAM];

  ... /* Se asignan los valores de v */

  indice=0;
  for (i=0; i<TAM; i=i+1) {
    if ((v[i].s1>=TOPE) || (v[i].s2>=TOPE) ||
                                (v[i].s3>=TOPE)) {
```

```
      printf("\n%s se clasifica,", v[i].nombre);
      strcpy(v2[indice].nombre,v[i].nombre);
      max=v[i].s1;
      if (v[i].s2 > max) {
        max=v[i].s2;
        printf(" no por el primer salto,");
      }
      if (v[i].s3 > max) {
        max=v[i].s3;
        printf(" ni por el segundo.");
      }
      v2[indice].marca=max;
      indice=indice+1;
    }
  }

  printf ("\n\nClasificados: \n");
  for (i=0; i<indice; i=i+1) {
    printf("\t>>%s, con un mejor salto de
         %5.2f m.\n", v2[i].nombre, v2[i].marca);
  }
}
```

7. Escribe todo lo que se podría leer en la pantalla del ordenador al ejecutar el siguiente fragmento de programa, sabiendo que el vector `v` es un vector de `TAM=6` elementos de tipo `tHalterofilia`,

```
typedef struct {
   char nombre[N];
   int peso;
} tHalterofilia;
```

y que los valores que se asignarán a los elementos de `v` son:

nombre	"Paco"	"Pedro"	"Ana"	"Alicia"	"Pepe"	"Luis"
peso	40	80	45	70	90	42
	0	1	2	3	4	5

```
int main() {
  int indice,i;
  tHalterofilia v[TAM];

  ... /* Se asignan los valores de v */

  indice=0;
  printf("\nEl valor de indice es%d",indice);
  for (i=1; i<TAM; i++) {
     if (v[i].peso > v[indice].peso) {
        indice = i;
        printf("\nEl valor de indice es%d",indice);
     }
  }
```

```
  printf("\nGana %s con una marca de %d kilos.\n",
     v[indice].nombre, v[indice].peso);

  printf ("\nDiferencias entre participantes: \n");
  for (i=0; i<TAM; i++) {
     if (i!=indice) {
        printf("\t>> Entre %s y %s, de %d kilos.\n",
             v[indice].nombre, v[i].nombre,
             (v[indice].peso-v[i].peso));
     }
  }
}
```

8. Escribir un algoritmo que permita sumar dos vectores de `N` elementos.

9. Escribir un algoritmo que calcule la media de los elementos de un vector real de `N` elementos.

10. Escribir un algoritmo que permita obtener el producto escalar de dos vectores.

11. Dado un vector de `N` componentes reales, diseñar un algoritmo que permita obtener su elemento máximo y otro algoritmo que permita obtener su elemento mínimo.

12. Obtener los algoritmos que, para un vector a de `N` componentes, determinen:

    a) El recorrido, r = max(a[i]) - min(a[i]), i=1.. `N`,

    b) El valor medio de los componentes de a, $\tilde{a}$,

    c) La desviación típica,
    $$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(a_i - \tilde{a}\,)^2}{n}}$$

    d) El coeficiente de variación, $\frac{\sigma}{\tilde{a}}$.

13. Alguien ha definido la función `amigos` cuyo prototipo es:

```
boole amigos(int n1, int n2);
/* pre: n1=N1, n2=N2, enteros positivos */
/* post: devuelve cierto si N1 y N2 son amigos, */
/*       falso en caso contrario */
```

Teniendo en cuenta esta definición (además de la habitual para el tipo `boole`), indicar cuáles son los 5 errores (**no sintácticos**) cometidos al desarrollar el siguiente procedimiento, justificando para cada error el porqué:

```
void vectorAmigos (int M, int v[]) {
/* pre: v es un vector de enteros, de tamaño M */
   int suma,i,j;

   for (i=0; i<N; i=i+1) {
      /* Se calcula en suma el valor */
      /* de la suma de divisores de i */
      for (j=1; j<(i/2); j=j+1) {
         if (i%j==0) {
            suma=suma+i;
         }
```

```
      }
      /* El valor almacenado en suma es */
      /* el unico candidato a amigo de i */
      /* Si lo es, se guarda en v[i] y, */
      /* si no, se guarda un cero */

      v[i]=(amigos(i,suma))?0:suma;
   }
/* post: v es un vector en el que v[i] es amigo */
/* de i, si i tiene amigos, 0 en caso contrario */
}
```

14. Dado el siguiente bucle, que trabaja con un vector de `N` reales `v` y un valor real `x`,

```
  ....
  boole iguales=cierto;
  int i=0;

  while ((i<N) && iguales)
    if (v[i]==x)
        iguales=falso;
    else
        i=i+1;
  ....
```

¿es un invariante del bucle el predicado

"`v[k]=x`, para todos los valores de k tales que $0 <= k < i$"?

15. Dado un vector A de caracteres, escribir un algoritmo que indique si la frase almacenada en dicho vector es o no capicúa.

Nota: Se considera que los vectores de caracteres poseen un centinela ('\0') que indica el final de los caracteres válidos del vector.

16. ¿Verdadero o falso? Hay que justificar las respuestas.

a) Las siguientes secuencias hacen exactamente lo mismo.

```
/*Secuencia num. 1*/
i=0;
enc=falso;
while((i<n)&&!(enc)){
  enc=(v[i]==x);
  i=i+1;
}
```

```
/*Secuencia num. 2*/
i=0;
enc=falso;
while((i<n){
  if(!(enc)){
    enc=(v[i]==x);
  }
  i=i+1;
}
```

b) El alumno que escribió la secuencia número 2, obtuvo mejor nota en ese problema que el alumno que escribió la secuencia número 1.

c) La siguiente versión es mucho mejor que las anteriores.

```
/*Secuencia num. 3*/
i=0;
enc=falso;
while((i<n){
  if(!(enc)){
    enc=(v[i]==x);
    i=i+1;
  }
}
```

17. Dado un vector `A` de tipo base carácter y dados dos caracteres `car1` y `car2`, escribir un algoritmo que busque las ocurrencias de `car1` en `A` y las sustituya por `car2`.

18. Dado un vector `A` de tipo base carácter y dado un carácter `c`, escribir un algoritmo que busque las ocurrencias de `c` en `A` y las elimine.

19. Necesitamos una función o procedimiento (**justificar la elección**) para corregir las mayúsculas de un párrafo, de modo que tanto la letra inicial como todas las primeras letras que aparezcan después de un punto estén en mayúsculas (independientemente de los espacios en blanco que pueda haber entre el punto y la letra). Por ejemplo, la corrección de la siguiente cadena

    ``El perro ladra. mi madre canta... quisiera verla. Oigo. no veo nada"

    produciría

    ``El perro ladra. Mi madre canta... Quisiera verla. Oigo. No veo nada"

    a) Implementa una función o procedimiento auxiliar `pasarAMayusculas`. Si se le pasa una letra minúscula, debe devolver su conversión a mayúscula. En otro caso, debe devolver el carácter sin modificar. Recuerda que puedes pasar un carácter `c` de minúscula a mayúscula con la expresión `c-32`.

    b) Utilizando `pasarAMayusculas`, implementa la función o procedimiento `corrigeMayusculas`, que obtenga una versión corregida de una cadena dada.

    c) Escribir un programa que use `corrigeMayusculas` para ir corrigiendo cadenas que se irán leyendo de teclado, finalizando el proceso cuando el usuario teclee una cadena vacía.

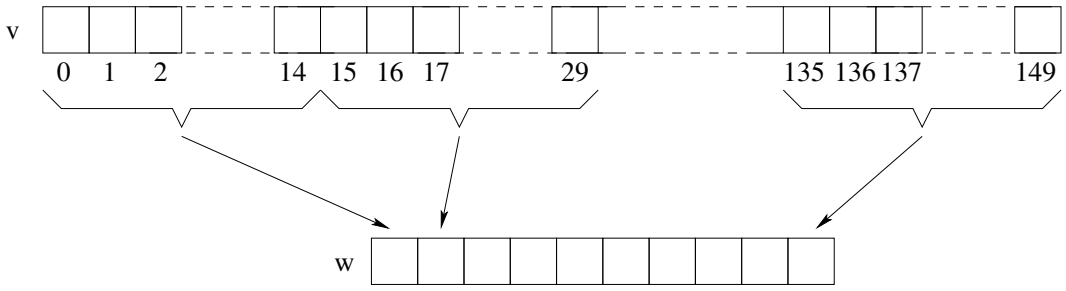
    No olvidéis indicar precondiciones y postcondiciones en todos los apartados.

20. Uno de los métodos más simples para comprimir archivos de música consiste en ir calculando la media aritmética de R valores consecutivos y almacenar el resultado obtenido en otro fichero, también de forma consecutiva. Así, se consigue dividir el tamaño del fichero original entre R. La idea del problema que os proponemos es similar, pero con vectores.

    Se dispone de un vector `v` de `N` números reales, siendo `N` múltiplo de `R`, ambas constantes. Y se quiere obtener otro vector `w` de números reales, de tamaño `M=N/R`. Por ejemplo, si `R`=15 y `N`=150, entonces `M` sería 10.

    ```
    #define R 15    /* Reducción del tamaño */
    #define N 150   /* Tamaño del vector v, múltiplo de R */
    #define M 10    /* Tamaño del vector w, N/R */
    ```

    Se pide escribir una función o procedimiento (hay que justificar la elección) que, dado el vector `v`, obtenga el vector `w`, de modo que cada elemento de `w` sea la media de 15 elementos consecutivos de `v`, tal y como muestra el siguiente esquema:



21. Se sabe que todo número par puede escribirse como la suma de dos números primos. Escribir un algoritmo que, dado un número par, devuelva dos números primos tales que su suma sea el número dado. Indicación: almacenad en un vector los números primos existentes entre el 1 y el número a descomponer.

22. Dados los tipos (N es una constante previamente definida)

```
typedef enum {FALSO, VERDADERO} boole;

typedef boole vecBoole[N];
```

escribir un algoritmo al que se le pasará un valor de tipo entero (positivo)en el parámetro `a`, y devuelva en un vector `b`, de tipo `vecBoole` el valor de `a` codificado en binario, de forma que asimilaremos el valor `VERDADERO` al dígito binario 1 y el valor `FALSO` al dígito binario 0.

Por ejemplo, si el valor de a es 35, sabemos que su representación binaria es el número $100011_{(2)}$. Por lo tanto, en b se debería devolver,

b[0] = `VERDADERO`, b[1] = `VERDADERO`, b[2] = `FALSO`, b[3] = `FALSO`, b[4] = `FALSO`, b[5] = `VERDADERO`, ...

¿Debe cumplir `a` alguna condición especial?

23. Dados los tipos

```
typedef enum {FALSO, VERDADERO} boole;

typedef boole vecBoole[N];
```

donde N es una constante previamente definida, escribir un algoritmo que permita realizar sumas de dos números binarios con acarreo.

Para ello supondremos que nos dan los números como dos vectores del tipo descrito en el problema anterior. Cada bit se almacenará en una posición del vector, sabiendo que el valor boolea `VERDADERO` se corresponde con el 1 y que el valor boole `FALSO`, con el 0. El resultado será otro número binario, por supuesto.

24. Escribir una función o procedimiento (**justificar la elección**) que, dada una cadena que expresa un valor en hexadecimal empezando con los caracteres `"0x"`, devuelva el correspondiente valor entero.

Por ejemplo, dada la cadena `"0x123"` debería devolver el número `291`, y dada la cadena `"0x11af3"` debería devolver el número 72435.

Se supone que la cadena sólo contiene dígitos (`'0'`, `'1'`, ..., `'9'`) y las letras que son dígitos hexadecimales **en minúsculas** (`'a'`, `'b'`, ..., `'f'`), y un `"0x"` al principio.

Se debe indicar cuál es la precondición y cuál la postcondición.

25. Dada la definición de tipo

```
typedef cualquierTipoBase vector[N];
```

y dado, x, de tipo vector, y que está ordenado, escribir:

a) Un algoritmo que permita determinar el número de secuencias que hay en x. Se entiende como secuencia cualquier tramo de vector con valores iguales. Así, si por ejemplo tuviéramos el vector

x = (0, 0, 1, 2, 2, 2, 3, 3, 4, 5, 6, 7, 7, 7, 7, 7, 7, 8, 8, 9, 23)

el número de secuencias es 11; o si tuviéramos,

x = (A, A, A, A, G, H, J, J, J, M, M, M, P, P, Z, Z, Z, Z, Z, Z)

el número de secuencias es 7.

b) Un algoritmo que permita determinar la secuencia más larga en x, indicando para ello su longitud y la posición en la que comienza. En los ejemplos anteriores, en el primer caso tenemos que la secuencia más larga es la de los 7s, que es de longitud 6 y comienza en la posición 12, y en el segundo es la de las Zs, que es de longitud 6 y comienza en la posición 15.

26. Escribir un algoritmo que indique si un vector tiene dos elementos iguales.

27. Escribir un algoritmo que indique cuántos elementos iguales a un valor dado hay en un vector.

28. Dados dos vectores A y B, de tipo base CARACTER, de tamaño N y M respectivamente (N>M), escribir un algoritmo que busque la primera ocurrencia de B en A; es decir, hay que comprobar si B está contenido en A (los elementos de B son los mismos y están en el mismo orden que un subconjunto de elementos de A) e indicar la posición en la que aparece.

29. Se dispone de un vector con la altimetría de una etapa de una carrera ciclista. El vector contiene la altura en metros en cada punto kilométrico: el índice del vector indica el hito kilométrico y el contenido del vector ofrece la altitud del lugar. Por ejemplo:

Índice:	0	1	2	3	...
Contenido:	500	501	200	300	...

Los índices del vector van desde 0 hasta el número de kilómetros de la etapa.

a) Escribir un algoritmo que calcule los kilómetros de subida y los de bajada de la etapa. En el ejemplo, en la etapa hay 2 kilómetros de subida y uno de bajada.

b) Escribir un algoritmo que calcule el desnivel total de subida que deben superar los ciclistas durante la etapa. Si en un kilómetro se sube menos de 10 metros, este desnivel no debe ser considerado e incluido en los cálculos. En el ejemplo anterior, el desnivel es de 100 metros.

c) Escribir un programa que calcule el puerto más largo y su desnivel medio (número de metros de desnivel del puerto * 100 / longitud en metros del puerto). Se define un puerto como una subida continua que no contenga bajadas ni tramos llanos.

30. Escribir un algoritmo que, dados los coeficientes de una ecuación de segundo grado, $ax^2 + bx + c = 0$, permita obtener sus raíces, sean estas reales o imaginarias.

31. Para almacenar los datos de cada participante en una competición de lanzamiento de disco, se ha diseñado la siguiente estructura de datos:

```
#define TAM1 40
#define TAM2 20
```

```
typedef struct {
   char nombre[TAM1];
   char pais[TAM2];
   int dorsal;
   float lanza1, lanza2, lanza3;
} tLanzador;
```

a) escribir una función que, dado un lanzador, calcule la media de sus 3 lanzamientos.

b) escribir una función que, dado un lanzador, devuelva su lanzamiento más largo.

c) escribir un procedimiento que lea por teclado los datos de un lanzador y los devuelva en un parámetro de salida de tipo `tLanzador`.

d) suponiendo que en la competición hay inscritos 50 lanzadores y dada la siguiente definición de tipo,

```
#define N 50

typedef tLanzador tCompeticion[N];
```

hay que escribir una función que devuelva el índice del participante con el mejor lanzamiento.

e) con la misma definición de `tCompeticion`, escribir una función que devuelva el índice del participante que llevaría el "premio a la regularidad", es decir, aquel con mejor media en los tres lanzamientos.

32. Escribir un algoritmo que calcule la media aritmética de los elementos de una matriz de tamaño $m \times n$.

33. Escribir un algoritmo que permita multiplicar una matriz de reales de tamaño $m \times n$ y un vector de reales de tamaño n.

34. Escribir un algoritmo que permita multiplicar dos matrices de dimensiones compatibles, es decir, una matriz de de tamaño $m \times n$ con otra de tamaño $n \times p$.

35. Escribir un algoritmo que permita almacenar una matriz de tamaño $m \times n$ en un vector de p elementos.

36. En una matriz de reales de tamaño $m \times n$, se han almacenado las notas de las n asignaturas de m alumnos de primero. Es decir, en el elemento [i][j] se guarda la nota del alumno i en la asignatura j. Se pide:

a) Un algoritmo que devuelva sobre un vector la nota media de cada alumno en el curso completo (definir el tipo de datos necesario para dicho parámetro de salida).

b) Un algoritmo que devuelva sobre un vector la media de las notas de cada asignatura (definir el tipo de datos necesario para dicho parámetro de salida).

c) Un algoritmo que devuelva los alumnos que han aprobado todas las asignaturas del curso completo (definir el tipo de datos necesario para los parámetros de salida).

37. Una matriz de tamaño $N \times N$ es la matriz identidad de tamaño N cuando todos sus elementos son 0 salvo los de la diagonal, que valen 1. Por ejemplo,

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Escribir una función que determine si una matriz dada de tamaño $N \times N$ es o no es la matriz identidad de tamaño N.

38. Escribe una función o procedimiento que devuelva cierto si, dada una matriz de enteros de `N` filas y `M` columnas, existe al menos una fila cuyos elementos sean todos iguales a uno dado, y falso en caso contrario.

39. Se quieren guardar los elementos no nulos de una matriz cuadrada de reales de tamaño $n \times n$ (definida como tipo `tMatriz`) en un vector de tamaño p (definido como tipo `tVector`). ¿Es correcto el siguiente algoritmo?¿Por qué? (Nota: hay p o menos elementos no nulos en la matriz)

```
void problematico(matriz A, int n, int p, vector v){
     int i, j, k;

     for(k=0; k<p;k++){
          for(i=0;i<n;i++){
               for(j=0;j<n;j++){
                    if (A[i][j]!=0.0)
                         v[k]=A[i][j];
               }
          }
     }
}
```

40. ¿Y este otro?¿Por qué?

```
void otroProblematico(matriz A, int n, int p, vector v){
     int i, j, k;

     k=1;
     for(i=0;i<n;i++){
          for(j=0;j<n;j++){
               if (A[i][j]!=0.0)
                       v[k]=A[i][j];
               k++;
          }
     }
}
```

41. Escribir un algoritmo que permita almacenar los elementos no nulos de una matriz de reales $n \times n$ en un vector, sabiendo que de cada elemento que se almacene, además del valor se quiere guardar el valor de sus índices.

42. Escribir un algoritmo que permita, dada una matriz de reales de tamaño $m \times n$, calcular la suma de su diagonal principal y de su diagonal secundaria. Ojo, que m y n pueden ser valores cualesquiera: iguales, distintos y puede ser que $m < n$ o que $m > n$.

    Por ejemplo, si la matriz fuera,

$$\begin{pmatrix} \mathbf{1,5} & 2{,}2 & 0{,}44 & 5{,}1 \\ 0{,}2 & \mathbf{3,3} & 7{,}5 & 8{,}25 \\ 0{,}33 & 4 & \mathbf{0,99} & 1{,}33 \\ 4{,}25 & 6{,}75 & 1{,}0 & \mathbf{2,1} \\ 0{,}56 & 7{,}25 & 1{,}1 & 33{,}5 \end{pmatrix}$$

el resultado sería 7.89, suma de la diagonal principal, y 20.85, suma de la diagonal secundaria.

43. Se ha definido un tipo `tMatrizChar`, como un array de tipo base char con `N` filas y `M` columnas. En cada elemento se almacena una letra o blanco. Por ejemplo,

H	O	L	A									
A	D	I	O	S								
C	A	R	A	M	B	O	L	A				
E	N	T	R	E	N	A	M	I	E	N	T	O
Y	O											
S	O	B	R	A	D	A	M	E	N	T	E	

Hay que escribir una función o procedimiento (justificar la elección) que procese la matriz y devuelva en un vector el número de elementos válidos en cada fila, es decir, cuántas letras hay antes de encontrar el blanco.

44. Escribir un algoritmo que dada una matriz $n \times n$, construya un vector v de n elementos, de forma que el valor de cada elemento v[i] se obtenga al sumar todos los elementos de la submatriz contenida entre las filas 0 e i y las columnas 0 e i. Por ejemplo, si

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

entonces v[0]=a_{00}, en v[1] estaría la suma de los elementos

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

y, en v[2],

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$$

y así, sucesivamente. El algoritmo sólo se considerará correcto si en el cálculo de cada elemento v[i] se saca provecho del resultado obtenido para el elemento anterior, v[i-1].

45. Suponiendo que se ha definido el tipo `tMatriz` como una matriz de tipo base real de tamaño $M \times N$:

a) Escribir una función o procedimiento (justificar la elección) que desplace los elementos de una fila genérica `f` una posición a la derecha, salvo el último elemento que se colocará como primero.

Por ejemplo,

Valor inicial matriz y `f` es 2:

$$\begin{pmatrix} 1{,}0 & 2{,}0 & 3{,}0 & 4{,}0 & 5{,}0 \\ 1{,}1 & 2{,}2 & 3{,}3 & 4{,}4 & 5{,}5 \\ 0{,}1 & 0{,}2 & 0{,}3 & 0{,}4 & 0{,}5 \\ 1{,}01 & 2{,}02 & 3{,}03 & 4{,}04 & 5{,}05 \end{pmatrix} \rightarrow \begin{pmatrix} 1{,}0 & 2{,}0 & 3{,}0 & 4{,}0 & 5{,}0 \\ 1{,}1 & 2{,}2 & 3{,}3 & 4{,}4 & 5{,}5 \\ \mathbf{0{,}5} & \mathbf{0{,}1} & \mathbf{0{,}2} & \mathbf{0{,}3} & \mathbf{0{,}4} \\ 1{,}01 & 2{,}02 & 3{,}03 & 4{,}04 & 5{,}05 \end{pmatrix}$$

Valor final matriz:

b) Escribir una función o procedimiento (justificar la elección) que utilice la función o procedimiento del apartado anterior, para desplazar una posición a la derecha todas las columnas de una matriz.

Por ejemplo,

Valor inicial matriz:

$$\begin{pmatrix} 1{,}0 & 2{,}0 & 3{,}0 & 4{,}0 & 5{,}0 \\ 1{,}1 & 2{,}2 & 3{,}3 & 4{,}4 & 5{,}5 \\ 0{,}1 & 0{,}2 & 0{,}3 & 0{,}4 & 0{,}5 \\ 1{,}01 & 2{,}02 & 3{,}03 & 4{,}04 & 5{,}05 \end{pmatrix} \rightarrow$$

Valor final matriz:

$$\begin{pmatrix} 5{,}0 & 1{,}0 & 2{,}0 & 3{,}0 & 4{,}0 \\ 5{,}5 & 1{,}1 & 2{,}2 & 3{,}3 & 4{,}4 \\ 0{,}5 & 0{,}1 & 0{,}2 & 0{,}3 & 0{,}4 \\ 5{,}05 & 1{,}01 & 2{,}02 & 3{,}03 & 4{,}04 \end{pmatrix}$$

46. Se necesita escribir una pequeña aplicación para automatizar las reservas de un cine. Para ello, se ha comenzado representando las butacas como una tabla de 2 dimensiones (N filas y M columnas) en la que cada elemento puede valer:

    -1 : ese asiento no está disponible (no se puede vender la entrada porque falta una butaca, o lo que sea)

    0 : ese asiento ya está reservado (está vendida la entrada)

    1 : ese asiento está libre (entrada a la venta)

    Se pide:

    a) Escribir una función o procedimiento que cuente el número total de butacas libres.

    b) Escribir una función o procedimiento que, dado un número de asientos consecutivos que se desea reservar, `nAsientos`, y dada una posición de comienzo, `nFila` y `nColumna`, reserve las `nAsientos` butacas en la fila `nFila` desde la columna `nColumna` en adelante. Además, debe devolver un valor de tipo `boole` que indique si se ha podido realizar la reserva o no se ha podido realizar.

    c) Escribir una función o procedimiento que indique, por cada fila, cuántas butacas libres hay y cuántas butacas no disponibles hay.

47. Se quiere escribir un programa para realizar el proceso de las encuestas del profesorado. La nota media de un profesor para una asignatura se obtendrá procesando una matriz en la que cada una de las *f* filas representa la nota obtenida en cada pregunta de la encuesta y cada una de las *c* columnas la nota otorgada por un alumno.

    Para evitar respuestas subjetivas, la primera pregunta es lo que se llama un "medidor de objetividad". Por ejemplo, la pregunta "El profesor asiste a clase", debería tener una respuesta uniforme, puesto que el profesor acude a clase para todo el mundo o para nadie. Este medidor se utiliza para descartar respuestas subjetivas de la siguiente forma: Se calcula la media de esa pregunta y su desviación típica. Todas aquellas columnas en las que la primera pregunta no esté dentro del rango [media - desviación ... media + desviación], se descartan.

    El algoritmo que permita calcular la media de un profesor, debe desechar las columnas no válidas. Se recuerda que la desviación típica de un vector se calcula como

    $$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(a_i - \tilde{a})^2}{n}}$$

    siendo $\tilde{a}$ el valor de la media.

48. A unas elecciones se presentan *NP* partidos que pretenden repartirse los *NE* escaños del Congreso de los Diputados.

    Escribir un programa que leerá un vector que contenga, por cada partido, el número de votos que ha obtenido y que imprima otro vector que contenga, por cada partido, el número de escaños que ha conseguido.

El método a utilizar para asignar escaños a los partidos es el siguiente (ley de D'Hont simplificada):

- Se construye una tabla de *NE* columnas y *NP* filas.
- La primera columna coincide con los resultados en votos obtenidos por cada partido; las sucesivas, columna j con j=1...*NE-1*, se construyen dividiendo los valores de la primera columna entre (j+1) (división entera).
- Los escaños se reparten atendiendo a las valores máximos de la tabla: el primer escaño al valor más grande, el segundo al siguiente más grande y así sucesivamente hasta repartir los *NE* escaños.