

► **1** Compara los programas `sumatorio.py` y `sumatorio.c`. Analiza sus semejanzas y diferencias. ¿Qué función desempeñan las llaves en `sumatorio.c`? ¿Qué función crees que desempeñan las líneas 6 y 7 del programa C? ¿A qué elemento de Python se parecen las dos primeras líneas de `sumatorio.c`? ¿Qué similitudes y diferencias aprecias entre las estructuras de control de Python y C? ¿Cómo crees que se interpreta el bucle `for` del programa C? ¿Por qué algunas líneas de `sumatorio.c` finalizan en punto y coma y otras no? ¿Qué diferencias ves entre los comentarios Python y los comentarios C?

► **2** Traduce a C este programa Python.

```
1 a = int(raw_input('Dame el primer número: '))
2 b = int(raw_input('Dame el segundo número: '))
3
4 if a >= b:
5     maximo = a
6 else:
7     maximo = b
8
9 print 'El máximo es', maximo
```

► **3** Traduce a C este programa Python.

```
1 n = int(raw_input('Dame un número: '))
2 m = int(raw_input('Dame otro número: '))
3
4 if n * m == 100:
5     print 'El producto de %d * %d es igual a 100' % (n, m)
6 else:
7     print 'El producto de %d * %d es distinto de 100' % (n, m)
```

► **4** Traduce a C este programa Python.

```
1 from math import sqrt
2
3 x1 = float(raw_input("Punto 1, coordenada x: "))
4 y1 = float(raw_input("Punto 1, coordenada y: "))
5 x2 = float(raw_input("Punto 2, coordenada x: "))
6 y2 = float(raw_input("Punto 2, coordenada y: "))
7 dx = x2 - x1
8 dy = y2 - y1
9 distancia = sqrt(dx**2 + dy**2)
10 print 'La distancia entre los puntos es: ', distancia
```

► **5** Traduce a C este programa Python.

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 if a != 0:
5     x = -b/a
6     print 'Solución: ', x
7 else:
8     if b != 0:
9         print 'La ecuación no tiene solución.'
10    else:
11        print 'La ecuación tiene infinitas soluciones.'
```

► **6** Traduce a C este programa Python.

```
1 from math import log
2
3 x = 1.0
4 while x < 10.0:
5     print x, '\t', log(x)
6     x = x + 1.0
```

► **7** Traduce a C este programa Python.

```
1 n = 1
2 while n < 6:
3     i = 1
4     while i < 6:
```

```

5     print n*i, '\t',
6         i = i + 1
7     print
8     n = n + 1

```

- 8 Traduce a C este programa Python.

```

1 from math import pi
2
3 opcion = 0
4 while opcion != 4:
5     print 'Escoge una opción:_'
6     print '1) Calcular el diámetro.'
7     print '2) Calcular el perímetro.'
8     print '3) Calcular el área.'
9     print '4) Salir.'
10    opcion = int(raw_input('Teclea 1, 2, 3 o 4 y pulsa el retorno de carro:'))
11
12    radio = float(raw_input('Dame el radio de un círculo:'))
13
14    if opcion == 1:
15        diametro = 2 * radio
16        print 'El diámetro es', diametro
17    elif opcion == 2:
18        perimetro = 2 * pi * radio
19        print 'El perímetro es', perimetro
20    elif opcion == 3:
21        area = pi * radio ** 2
22        print 'El área es', area
23    elif opcion < 0 or opcion > 4:
24        print 'Sólo hay cuatro opciones: 1, 2, 3 o 4. Tú has tecleado', opcion

```

- 9 Este programa C incorrecto tiene varios errores que ya puedes detectar. Indica cuáles son:

```

1 #include <stdio.h>
2
3 int a, b;
4
5 scanf("%d", &a); scanf("%d", &b)
6 while (a <= b):
7     scanf("%d", &a)
8     scanf("%d", &b)
9 printf("%d,%d\n", a, b);

```

- 10 Indenta «correctamente» este programa C.

```

1 #include <stdio.h>
2 int main(void)
3 {
4     int a, b;
5     scanf("%d", &a);
6     scanf("%d", &b);
7     while(a > b) {
8         scanf("%d", &a);
9         scanf("%d", &b);
10    }
11    printf("%d,%d\n", a, b);
12    return 0;
13 }

```

- 11 Haciendo pruebas durante el desarrollo de un programa hemos decidido comentar una línea del programa para que, de momento, no sea compilada. El programa nos queda así:

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b, i, j;

```

```

6
7  scanf("%d", &a);
8  scanf("%d", &b);
9  i = a;
10 j = 1;
11 while (i <= b) {
12     /* printf("%d %d\n", i, j); */
13     j *= 2;
14     i += 1;
15 }
16 printf("%d\n", j);
17 return 0;
18 }

```

Compilamos el programa y el compilador no detecta error alguno. Ahora decidimos comentar el bucle **while** completo, así que añadimos un nuevo par de marcas de comentario (líneas 11 y 17):

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b, i, j;
6
7     scanf("%d", &a);
8     scanf("%d", &b);
9     i = a;
10    j = 1;
11    /*
12     while (i <= b) {
13         /* printf("%d %d\n", i, j); */
14         j *= 2;
15         i += 1;
16     }
17     */
18    printf("%d\n", j);
19    return 0;
20 }
21 }

```

Al compilar nuevamente el programa aparecen mensajes de error. ¿Por qué?

► **12** ¿Da problemas este otro programa con comentarios?

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b, i, j;
6
7     scanf("%d", &a);
8     scanf("%d", &b);
9     i = a;
10    j = 1;
11    /*
12     while (i <= b) {
13         // printf("%d %d\n", i, j);
14         j *= 2;
15         i += 1;
16     }
17     */
18    printf("%d\n", j);
19    return 0;
20 }
21 }

```

► **13** ¿Cómo se interpreta esta sentencia?

```

1 i_=_x_/_*y*/z++
2 ;

```

► 14 Traduce a cadenas C las siguientes cadenas Python:

1. "una_cadena"
2. 'una_cadena'
3. "una_\ncadena\""
4. 'una_"cadena"'
5. 'una_\ncadena\''
6. "una_cadena_que_ocupa\ndos_líneas"
7. "una_cadena_que_\nno_ocupa_dos_líneas"


► 15 ¿Que mostrará por pantalla esta llamada a *printf* suponiendo que *a* es de tipo entero y vale 10?

```
printf("%d-%d\n", a+1, 2+2);
```


► 16 ¿Qué muestra por pantalla cada uno de estos programas?

a)  `ascii1.c` `ascii1.c`

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     char i;
6     for (i='A'; i<='Z'; i++)
7         printf("%c", i);
8     printf("\n");
9     return 0;
10 }
```

b)  `ascii2.c` `ascii2.c`

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     char i;
6     for (i=65; i<=90; i++)
7         printf("%c", i);
8     printf("\n");
9     return 0;
10 }
```

c)  `ascii3.c` `ascii3.c`

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i;
6     for (i='A'; i<='Z'; i++)
7         printf("%d_", i);
8     printf("\n");
9     return 0;
10 }
```

d)  `ascii4.c` `ascii4.c`

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i;
6     for (i='A'; i<='Z'; i++)
7         printf("%d-%c_", i, i);
8     printf("\n");
9     return 0;
10 }
```

e) `ascii5.c`

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     char i;
6     for (i='A'; i<='z'; i++) // Ojo: la z es minúscula.
7         printf("%d_", (int) i);
8     printf("\n");
9     return 0;
10 }
```

► **17** Diseña un programa que muestre la tabla ASCII desde su elemento de código numérico 32 hasta el de código numérico 126. En la tabla se mostrarán los códigos ASCII, además de las respectivas representaciones como caracteres de sus elementos. Aquí tienes las primeras y últimas líneas de la tabla que debes mostrar (debes hacer que tu programa muestre la información exactamente como se muestra aquí):

Decimal	Carácter
32	
33	!
34	"
35	#
36	\$
37	%
...	...
124	
125	}
126	~

► **18** Interpreta el significado de la sentencia $a = a + b$.

► **19** Sean a , b y c tres variables de tipo `int` cuyos valores actuales son 0, 1 y 2, respectivamente. ¿Qué valor tiene cada variable tras ejecutar esta secuencia de asignaciones?

```

1 a = b++ - c--;
2 a += --b;
3 c *= a + b;
4 a = b | c;
5 b = (a > 0) ? ++a : ++c;
6 b <<= a = 2;
7 c >>= a == 2;
8 a += a = b + c;
```

► **20** ¿Qué hace este programa?

`ternario.c`

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b, c, r;
6
7     printf("Dame un valor entero: "); scanf("%d", &a);
8     printf("Dame otro valor entero: "); scanf("%d", &b);
9     printf("Y uno más: "); scanf("%d", &c);
10
11     r = (a < b) ? ( (a < c) ? a : c ) : ( (b < c) ? b : c );
12
13     printf("Resultado: %d\n", r);
14
15     return 0;
16 }
```

- ▶ **21** Haz un programa que solicite el valor de x y muestre por pantalla el resultado de evaluar $x^4 - x^2 + 1$. (Recuerda que en C no hay operador de exponenciación.)
- ▶ **22** Diseña un programa C que solicite la longitud del lado de un cuadrado y muestre por pantalla su perímetro y su área.
- ▶ **23** Diseña un programa C que solicite la longitud de los dos lados de un rectángulo y muestre por pantalla su perímetro y su área.
- ▶ **24** Este programa C es problemático:

```

un_misterio.c un_misterio.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b;
6
7     a = 2147483647;
8     b = a + a;
9     printf("%d\n", a);
10    printf("%d\n", b);
11    return 0;
12 }

```

Al compilarlo y ejecutarlo hemos obtenido la siguiente salida por pantalla:

```

2147483647
-2

```

¿Qué ha ocurrido?

- ▶ **25** Diseña un programa C que solicite el radio r de una circunferencia y muestre por pantalla su perímetro ($2\pi r$) y su área (πr^2).
- ▶ **26** Si a es una variable de tipo **char** con el valor 127, ¿qué vale $\sim a$? ¿Y qué vale $!a$? Y si a es una variable de tipo **unsigned int** con el valor 2147483647, ¿qué vale $\sim a$? ¿Y qué vale $!a$?
- ▶ **27** ¿Qué resulta de evaluar cada una de estas dos expresiones?
 - a) $1 \ \&\& \ !\ !\ !\ (0 \ || \ 1) \ || \ !\ (0 \ || \ 1)$
 - b) $1 \ \& \ \sim\sim\sim(0 \ | \ 1) \ | \ \sim(0 \ | \ 1)$
- ▶ **28** ¿Por qué si a es una variable entera $a / 2$ proporciona el mismo resultado que $a \gg 1$? ¿Con qué operación de bits puedes calcular $a * 2$? ¿Y $a / 32$? ¿Y $a * 128$?
- ▶ **29** ¿Qué hace este programa?

```

swap.c swap.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     unsigned char a, b;
6     printf("Introduce el valor de a (entre 0 y 255): "); scanf("%hhu", &a);
7     printf("Introduce el valor de b (entre 0 y 255): "); scanf("%hhu", &b);
8
9     a ^= b;
10    b ^= a;
11    a ^= b;
12
13    printf("Valor de a: %hhu\n", a);
14    printf("Valor de b: %hhu\n", b);
15
16    return 0;
17 }

```

- ▶ **30** ¿Qué mostrará por pantalla este programa?

```

otra_conversion.delicada.c
otra_conversion.delicada.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a, b;
6     char c, d;
7     unsigned char e, f;
8
9     a = 384;
10    b = 256;
11    c = a;
12    d = b;
13    e = a;
14    f = b;
15    printf("%hhd_%hhd\n", c, d);
16    printf("%hhu_%hhu\n", e, f);
17
18    return 0;
19 }

```

► **31** ¿Qué valor se almacena en las variables i (de tipo **int**) y x (de tipo **float**) tras ejecutar cada una de estas sentencias?

- | | | | |
|-----------------|-------------------|---------------------|-----------------|
| a) $i = 2;$ | c) $i = 2 / 4;$ | e) $x = 2.0 / 4.0;$ | g) $x = 2 / 4;$ |
| b) $i = 1 / 2;$ | d) $i = 2.0 / 4;$ | f) $x = 2.0 / 4;$ | h) $x = 1 / 2;$ |

► **32** ¿Qué valor se almacena en las variables i (de tipo **int**) y x (de tipo **float**) tras ejecutar estas sentencias?

- | | | |
|--|----------------------------------|----------------------------------|
| a) $i = (\text{float}) 2;$ | e) $x = 2.0 / (\text{int}) 4.0;$ | i) $x = (\text{float}) (1 / 2);$ |
| b) $i = 1 / (\text{float}) 2;$ | f) $x = (\text{int}) 2.0 / 4;$ | j) $x = 1 / (\text{float}) 2;$ |
| c) $i = (\text{int}) (2 / 4);$ | g) $x = (\text{int}) (2.0 / 4);$ | |
| d) $i = (\text{int}) 2. / (\text{float}) 4;$ | h) $x = 2 / (\text{float}) 4;$ | |

► **33** ¿Qué valor tiene cada identificador de este tipo enumerado?

```
enum { Primera='a', Segunda, Tercera, Penultima='y', Ultima };
```

(No te hemos explicado qué hace la segunda asignación. Comprueba que la explicación que das es correcta con un programa que muestre por pantalla el valor de cada identificador.)

► **34** Diseña un programa C que solicite la longitud de los tres lados de un triángulo (a , b y c) y muestre por pantalla su perímetro y su área ($\sqrt{s(s-a)(s-b)(s-c)}$, donde $s = (a + b + c)/2$).

Compila y ejecuta el programa.

► **35** Diseña un programa C que solicite el radio r de una circunferencia y muestre por pantalla su perímetro ($2\pi r$) y su área (πr^2). Utiliza la aproximación a π predefinida en la biblioteca matemática.

Compila y ejecuta el programa.

► **36** Diseña un programa C que pida por teclado un número entero y diga si es par o impar.

► **37** Diseña un programa que lea dos números enteros y muestre por pantalla, de estos tres mensajes, el que convenga:

- «El segundo es el cuadrado exacto del primero.»,
- «El segundo es menor que el cuadrado del primero.»,
- «El segundo es mayor que el cuadrado del primero.».

► **38** También en C es problemática la división por 0. Haz un programa C que resuelva la ecuación $ax + b = 0$ solicitando por teclado el valor de a y b (ambos de tipo **float**). El programa detectará si la ecuación no tiene solución o si tiene infinitas soluciones y, en cualquiera de los dos casos, mostrará el pertinente aviso.

► **39** Diseña un programa que solucione ecuaciones de segundo grado. El programa detectará y tratará por separado las siguientes situaciones:

- la ecuación tiene dos soluciones reales;

- la ecuación tiene una única solución real;
- la ecuación no tiene solución real;
- la ecuación tiene infinitas soluciones.

► **40** Realiza un programa que proporcione el desglose en billetes y monedas de una cantidad exacta de euros. Hay billetes de 500, 200, 100, 50, 20, 10 y 5 euros y monedas de 1 y 2 euros.

Por ejemplo, si deseamos conocer el desglose de 434 euros, el programa mostrará por pantalla el siguiente resultado:

```
2 billetes de 200 euros.
1 billete de 20 euros.
1 billete de 10 euros.
2 monedas de 2 euros.
```

Observa que la palabra «billete» (y «moneda») concuerda en número con la cantidad de billetes (o monedas) y que si no hay piezas de un determinado tipo (en el ejemplo, de 1 euro), no muestra el mensaje correspondiente.

► **41** Diseña un programa C que lea un carácter cualquiera desde el teclado, y muestre el mensaje «Es una MAYÚSCULA.» cuando el carácter sea una letra mayúscula y el mensaje «Es una MINÚSCULA.» cuando sea una minúscula. En cualquier otro caso, no mostrará mensaje alguno. (Considera únicamente letras del alfabeto inglés.)

► **42** Diseña un programa que lea cinco números enteros por teclado y determine cuál de los cuatro últimos números es más cercano al primero.

(Por ejemplo, si el usuario introduce los números 2, 6, 4, 1 y 10, el programa responderá que el número más cercano al 2 es el 1.)

► **43** Diseña un programa que, dado un número entero, determine si éste es el doble de un número impar. (Ejemplo: 14 es el doble de 7, que es impar.)

► **44** Escribe un programa que muestre un menú en pantalla con dos opciones: «saludar» y «salir». El programa pedirá al usuario una opción y, si es válida, ejecutará su acción asociada. Mientras no se seleccione la opción «salir», el menú reaparecerá y se solicitará nuevamente una opción. Implementa el programa haciendo uso únicamente de bucles **do-while**.

► **45** Haz un programa que pida un número entero de teclado distinto de 1. A continuación, el programa generará una secuencia de números enteros cuyo primer número es el que hemos leído y que sigue estas reglas:

- si el último número es par, el siguiente resulta de dividir a éste por la mitad;
- si el último número es impar, el siguiente resulta de multiplicarlo por 3 y añadirle 1.

Todos los números se irán mostrando por pantalla conforme se vayan generando. El proceso se repetirá hasta que el número generado sea igual a 1. Utiliza un bucle **do-while**.

► **46** Implementa el programa de cálculo de x^n (para x y n entero) con un bucle **for**.

► **47** Implementa un programa que dado un número de tipo **int**, leído por teclado, se asegure de que sólo contiene ceros y unos y muestre su valor en pantalla si lo interpretamos como un número binario. Si el usuario introduce, por ejemplo, el número 1101, el programa mostrará el valor 13. Caso de que el usuario introduzca un número formado por números de valor diferente, indica al usuario que no puedes proporcionar el valor de su interpretación como número binario.

► **48** Haz un programa que solicite un número entero y muestre su factorial. Utiliza un entero de tipo **long long** para el resultado. Debes usar un bucle **for**.

► **49** El número de combinaciones de n elementos tomados de m en m es:

$$C_n^m = \binom{n}{m} = \frac{n!}{(n-m)!m!}$$

Diseña un programa que pida el valor de n y m y calcule C_n^m . (Ten en cuenta que n ha de ser mayor o igual que m .)

(Puedes comprobar la validez de tu programa introduciendo los valores $n = 15$ y $m = 10$: el resultado es 3003.)

► **50** ¿Qué muestra por pantalla este programa?

```
desplazamientos.c
desplazamientos.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a = 127, b = 1024, c, i;
6
```

```

7  c = a ^ b;
8
9  printf("%d\n", c);
10
11 a = 2147483647;
12 for (i = 0; i < 8*sizeof(a); i++) {
13     printf("%d", ((c & a) != 0) ? 1 : 0);
14     a >>= 1;
15 }
16 printf("\n");
17
18 a = 1;
19 for (i = 0; i < 8*sizeof(a); i++) {
20     if ((c & a) != 0) c >>= 1;
21     else c <<= 1;
22     a <<= 1;
23 }
24
25 a = 2147483647;
26 for (i = 0; i < 8*sizeof(a); i++) {
27     printf("%d", ((c & a) != 0) ? 1 : 0);
28     a >>= 1;
29 }
30 printf("\n");
31 return 0;
32 }

```

► **51** Cuando no era corriente el uso de terminales gráficos de alta resolución era común representar gráficas de funciones con el terminal de caracteres. Por ejemplo, un periodo de la función seno tiene este aspecto al representarse en un terminal de caracteres (cada punto es un asterisco):



Haz un programa C que muestre la función seno utilizando un bucle que recorre el periodo 2π en 24 pasos (es decir, representándolo con 24 líneas).

► **52** Modifica el programa para que muestre las funciones seno (con asteriscos) y coseno (con sumas) simultáneamente.

► **53** Diseña un programa C que muestre el valor de 2^n para todo n entre 0 y un valor entero proporcionado por teclado.

► **54** Haz un programa que pida al usuario una cantidad de euros, una tasa de interés y un número de años y muestre por pantalla en cuánto se habrá convertido el capital inicial transcurridos esos años si cada año se aplica la tasa de interés introducida.

Recuerda que un capital C a un interés del x por cien durante n años se convierte en $C \cdot (1 + x/100)^n$.

(Prueba tu programa sabiendo que 10 000 euros al 4.5% de interés anual se convierten en 24 117.14 euros al cabo de 20 años.)

► **55** Un vector en un espacio tridimensional es una tripleta de valores reales (x, y, z) . Deseamos confeccionar un programa que permita operar con dos vectores. El usuario verá en pantalla un menú con las siguientes opciones:

- 1) Introducir el primer vector
- 2) Introducir el segundo vector
- 3) Calcular la suma
- 4) Calcular la diferencia
- 5) Calcular el producto vectorial
- 6) Calcular el producto escalar

- 7) Calcular el ángulo (en grados) entre ellos
- 8) Calcular la longitud
- 9) Finalizar

Tras la ejecución de cada una de las acciones del menú éste reaparecerá en pantalla, a menos que la opción escogida sea la número 9. Si el usuario escoge una opción diferente, el programa advertirá al usuario de su error y el menú reaparecerá.

Las opciones 4 y 5 pueden proporcionar resultados distintos en función del orden de los operandos, así que, si se escoge cualquiera de ellas, aparecerá un nuevo menú que permita seleccionar el orden de los operandos. Por ejemplo, la opción 4 mostrará el siguiente menú:

- 1) Primer vector menos segundo vector
- 2) Segundo vector menos primer vector

Nuevamente, si el usuario se equivoca, se le advertirá del error y se le permitirá corregirlo.

La opción 8 del menú principal conducirá también a un submenú para que el usuario decida sobre qué vector se aplica el cálculo de longitud.

Puede que necesites que te refresquemos la memoria sobre los cálculos a realizar. Quizá la siguiente tabla te sea de ayuda:

Operación	Cálculo
Suma: $(x_1, y_1, z_1) + (x_2, y_2, z_2)$	$(x_1 + x_2, y_1 + y_2, z_1 + z_2)$
Diferencia: $(x_1, y_1, z_1) - (x_2, y_2, z_2)$	$(x_1 - x_2, y_1 - y_2, z_1 - z_2)$
Producto escalar: $(x_1, y_1, z_1) \cdot (x_2, y_2, z_2)$	$x_1x_2 + y_1y_2 + z_1z_2$
Producto vectorial: $(x_1, y_1, z_1) \times (x_2, y_2, z_2)$	$(y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2)$
Ángulo entre (x_1, y_1, z_1) y (x_2, y_2, z_2)	$\frac{180}{\pi} \cdot \arccos\left(\frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2}\sqrt{x_2^2 + y_2^2 + z_2^2}}\right)$
Longitud de (x, y, z)	$\sqrt{x^2 + y^2 + z^2}$

Ten en cuenta que tu programa debe contemplar toda posible situación excepcional: divisiones por cero, raíces con argumento negativo, etc..

► 56 ¿Qué muestra por pantalla este programa?

```

continue.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i;
6
7     i = 0;
8     while (i < 10) {
9         if (i % 2 == 0) {
10            i++;
11            continue;
12        }
13        printf("%d\n", i);
14        i++;
15    }
16
17    for (i = 0; i < 10; i++) {
18        if (i % 2 != 0)
19            continue;
20        printf("%d\n", i);
21    }
22    return 0;
23 }

```

► 57 Traduce a C este programa Python.

```

1 car = raw_input('Dame un carácter: ')
2 if "a" <= car.lower() <= "z" or car == "_":
3     print "Este carácter es válido en un identificador en Python."
4 else:
5     if not (car < "0" or "9" < car):

```

```

6     print "Un dígito es válido en un identificador en Python.",
7     print "siempre que no sea el primer carácter."
8     else:
9     print "Carácter no válido para formar un identificador en Python."

```

► **58** Traduce a C este programa Python.

```

1  from math import pi
2  radio = float(raw_input('Dame el radio de un círculo:'))
3  opcion = ''
4  while opcion != 'a' and opcion != 'b' and opcion != 'c':
5      print 'Escoge una opción:'
6      print 'a) Calcular el diámetro.'
7      print 'b) Calcular el perímetro.'
8      print 'c) Calcular el área.'
9      opcion = raw_input('Teclea a, b o c y pulsa el retorno de carro:')
10 if opcion == 'a':
11     diametro = 2 * radio
12     print 'El diámetro es', diametro
13 elif opcion == 'b':
14     perimetro = 2 * pi * radio
15     print 'El perímetro es', perimetro
16 elif opcion == 'c':
17     area = pi * radio ** 2
18     print 'El área es', area
19 else:
20     print 'Sólo hay tres opciones: a, b o c. Tú has tecleado', opcion

```

► **59** Traduce a C este programa Python.

```

1  anyo = int(raw_input('Dame un año:'))
2  if anyo % 4 == 0 and (anyo % 100 != 0 or anyo % 400 == 0):
3      print 'El año', anyo, 'es bisiesto.'
4  else:
5      print 'El año', anyo, 'no es bisiesto.'

```

► **60** Traduce a C este programa Python.

```

1  limite = int(raw_input('Dame un número:'))
2
3  for num in range(1, limite+1):
4      creo_que_es_primo = 1
5      for divisor in range(2, num):
6          if num % divisor == 0:
7              creo_que_es_primo = 0
8              break
9      if creo_que_es_primo == 1:
10         print num

```

► **61** Escribe un programa que solicite dos enteros n y m asegurándose de que m sea mayor o igual que n . A continuación, muestra por pantalla el valor de $\sum_{i=n}^m 1/i$.

► **62** Escribe un programa que solicite un número entero y muestre todos los números primos entre 1 y dicho número.

► **63** Haz un programa que calcule el máximo común divisor (mcd) de dos enteros positivos. El mcd es el número más grande que divide exactamente a ambos números.

► **64** Haz un programa que calcule el máximo común divisor (mcd) de tres enteros positivos.

► **65** Haz un programa que vaya leyendo números y mostrándolos por pantalla hasta que el usuario introduzca un número negativo. En ese momento, el programa acabará mostrando un mensaje de despedida.

► **66** Haz un programa que vaya leyendo números hasta que el usuario introduzca un número negativo. En ese momento, el programa mostrará por pantalla el número mayor de cuantos ha visto.

► **67** Declara e inicializa un vector de 100 elementos de modo que los componentes de índice par valgan 0 y los de índice impar valgan 1.

► **68** Escribe un programa C que almacene en un vector los 50 primeros números de Fibonacci. Una vez calculados, el programa los mostrará por pantalla en orden inverso.

- **69** Escribe un programa C que almacene en un vector los 50 primeros números de Fibonacci. Una vez calculados, el programa pedirá al usuario que introduzca un número y dirá si es o no es uno de los 50 primeros números de Fibonacci.
- **70** Puedes ahorrar tiempo de ejecución haciendo que i tome valores entre 2 y la raíz cuadrada de N . Modifica el programa y comprueba que obtienes el mismo resultado.
- **71** ¿Contiene en cada instante la variable *frecuencia* el verdadero valor de la frecuencia de aparición de un valor? Si no es así, ¿qué contiene? ¿Afecta eso al cálculo efectuado? ¿Por qué?
- **72** Esta nueva versión del programa presenta la ventaja adicional de no fijar un límite máximo a la edad de las personas. El programa resulta, así, de aplicación más general. ¿Son todas ventajas? ¿Ves algún aspecto negativo? Reflexiona sobre la velocidad de ejecución del programa comparada con la del programa que consume más memoria.
- **73** Fíjate en la línea 44 del programa y compárala con las líneas 44 y 45 de su versión anterior. ¿Es correcto ese cambio? ¿Lo sería este otro?:

```
44     if (frecuencia++ > frecuencia_moda) {
```

- **74** Cuando el número de edades es par no hay elemento central en el vector ordenado, así que estamos escogiendo la mediana como uno cualquiera de los elementos «centrales». Utiliza una definición alternativa de edad mediana que considera que su valor es la media de las dos edades que ocupan las posiciones más próximas al centro.
- **75** Modifica el ejercicio anterior para que, caso de haber dos o más valores con la máxima frecuencia de aparición, se muestren todos por pantalla al solicitar la moda.
- **76** Modifica el programa anterior para que permita efectuar cálculos con hasta 100 personas.
- **77** Modifica el programa del ejercicio anterior para que muestre, además, cuántas edades hay entre 0 y 9 años, entre 10 y 19, entre 20 y 29, etc. Considera que ninguna edad es igual o superior a 150.
Ejemplo: si el usuario introduce las siguientes edades correspondientes a 12 personas:

```
10 23 15 18 20 18 57 12 29 31 78 28
```

el programa mostrará (además de la media, desviación típica, moda y mediana), la siguiente tabla:

```
0 - 9: 0
10 - 19: 5
20 - 29: 4
30 - 39: 1
40 - 49: 0
50 - 59: 1
60 - 69: 0
70 - 79: 1
80 - 89: 0
90 - 99: 0
100 - 109: 0
110 - 119: 0
120 - 129: 0
130 - 139: 0
140 - 149: 0
```

- **78** Modifica el programa para que muestre un histograma de edades. La tabla anterior se mostrará ahora como este histograma:

```
0 - 9:
10 - 19: *****
20 - 29: ****
30 - 39: *
40 - 49:
50 - 59: *
60 - 69:
70 - 79: *
80 - 89:
90 - 99:
100 - 109:
110 - 119:
120 - 129:
130 - 139:
140 - 149:
```

Como puedes ver, cada asterisco representa la edad de una persona.

► **79** Modifica el programa anterior para que el primer y último rangos de edades mostrados en el histograma correspondan a tramos de edades en los que hay al menos una persona. El histograma mostrado antes aparecerá ahora así:

```
10 - 19: *****
20 - 29: *****
30 - 39: *
40 - 49:
50 - 59: *
60 - 69:
70 - 79: *
```

► **80** Modifica el programa del ejercicio anterior para que muestre el mismo histograma de esta otra forma:

```
|##### | | | | | | |
|##### |##### | | | | | |
|##### |##### | | | | | |
|##### |##### | | | | | |
|##### |##### |##### | |##### | |##### |
+-----+-----+-----+-----+-----+-----+
| 10 - 19 | 20 - 29 | 30 - 39 | 40 - 49 | 50 - 59 | 60 - 69 | 70 - 79 |
```

► **81** Modifica el programa anterior para que no se muestren los coeficientes nulos.

► **82** Tras efectuar los cambios propuestos en el ejercicio anterior no aparecerá nada por pantalla cuando todos los valores del polinomio sean nulos. Modifica el programa para que, en tal caso, se muestre por pantalla 0.000000.

► **83** Tras efectuar los cambios propuestos en los ejercicios anteriores, el polinomio empieza con un molesto signo positivo cuando s_0 es nulo. Corrige el programa para que el primer término del polinomio no sea precedido por el carácter +.

► **84** Cuando un coeficiente es negativo, por ejemplo -1 , el programa anterior muestra su correspondiente término en pantalla así: $+ -1.000 x^1$. Modifica el programa anterior para que un término con coeficiente negativo como el del ejemplo se muestre así: $- 1.000000 x^1$.

► **85** ¿Entiendes por qué hemos reservado $2 * TALLA_POLINOMIO - 1$ elementos para m y no $2 * TALLA_POLINOMIO$?

► **86** El programa que hemos diseñado es ineficiente. Si, por ejemplo, trabajamos con polinomios de grado 5, sigue operando con los coeficientes correspondientes a x^6, x^7, \dots, x^{10} , que son nulos. Modifica el programa para que, con la ayuda de variables enteras, recuerde el grado de los polinomios $p(x)$ y $q(x)$ en sendas variables $talla_p$ y $talla_q$ y use esta información en los cálculos de modo que se opere únicamente con los coeficientes de los términos de grado menor o igual que el grado del polinomio.

► **87** Diseña un programa que pida el valor de 10 números enteros *distintos* y los almacene en un vector. Si se da el caso, el programa advertirá al usuario, tan pronto sea posible, si introduce un número repetido y solicitará nuevamente el número hasta que sea diferente de todos los anteriores. A continuación, el programa mostrará los 10 números por pantalla.

► **88** En una estación meteorológica registramos la temperatura (en grados centígrados) cada hora durante una semana. Almacenamos el resultado en un vector de 168 componentes (que es el resultado del producto 7×24). Diseña un programa que lea los datos por teclado y muestre:

- La máxima y mínima temperaturas de la semana.
- La máxima y mínima temperaturas de cada día.
- La temperatura media de la semana.
- La temperatura media de cada día.
- El número de días en los que la temperatura media fue superior a 30 grados.
- El día y hora en que se registró la mayor temperatura.

► **89** La cabecera `stdlib.h` incluye la declaración de funciones para generar números aleatorios. La función `rand`, que no tiene parámetros, devuelve un entero positivo aleatorio. Si deseas generar números aleatorios entre 0 y un valor dado N , puedes evaluar `rand() % (N+1)`. Cuando ejecutas un programa que usa `rand`, la semilla del generador de números aleatorios es siempre la misma, así que acabas obteniendo la misma secuencia de números aleatorios. Puedes cambiar la semilla del generador de números aleatorios pasándole a la función `srand` un número entero sin signo.

Usa el generador de números aleatorios para inicializar un vector de 10 elementos con números enteros entre 0 y 4. Muestra por pantalla el resultado. Detecta y muestra, a continuación, el tamaño de la sucesión más larga de números consecutivos iguales.

(Ejemplo: si los números generados son 0 4 3 3 2 1 3 2 2 2, el tramo más largo formado por números iguales es de talla 3 (los tres doses al final de la secuencia), así que por pantalla aparecerá el valor 3.)

- **90** Modifica el ejercicio anterior para que trabaje con un vector de 100 elementos.
- **91** Genera un vector con 20 números aleatorios entre 0 y 100 y muestra por pantalla el vector resultante y la secuencia de números crecientes consecutivos más larga.
(Ejemplo: la secuencia 1 33 73 85 87 93 99 es la secuencia creciente más larga en la serie de números 87 45 34 12 1 33 73 85 87 93 99 0 100 65 32 17 29 16 12 0.)
- **92** Escribe un programa C que ejecute 1000 veces el cálculo de la longitud de la secuencia más larga sobre diferentes secuencias aleatorias (ver ejercicio anterior) y que muestre la longitud media y desviación típica de dichas secuencias.
- **93** Genera 100 números aleatorios entre 0 y 1000 y almacénalos en un vector. Determina a continuación qué números aparecen más de una vez.
- **94** Genera 100 números aleatorios entre 0 y 10 y almacénalos en un vector. Determina a continuación cuál es el número que aparece más veces.
- **95** Diseña un programa C que almacene en un vector los 100 primeros números primos.
- **96** Diseña un programa C que lea y almacene en un vector 10 números enteros asegurándose de que sean positivos. A continuación, el programa pedirá que se introduzca una serie de números enteros y nos dirá si cada uno de ellos está o no en el vector. El programa finaliza cuando el usuario introduce un número negativo.
- **97** Diseña un programa C que lea y almacene en un vector 10 números enteros asegurándose de que sean positivos. A continuación, el programa pedirá que se introduzca una serie de números enteros y nos dirá si cada uno de ellos está o no en el vector. El programa finaliza cuando el usuario introduce un número negativo.
Debes ordenar por el método de la burbuja el vector de 10 elementos tan pronto se conocen sus valores. Cuando debas averiguar si un número está o no en el vector, utiliza el algoritmo de búsqueda dicotómica.
- **98** ¿Qué problema presenta esta otra versión del mismo programa?

```

lee_vector.1.c
lee_vector.c
1 #include <stdio.h>
2
3 #define TALLA 5
4
5 int main(void)
6 {
7     int a[TALLA], i;
8
9     for (i = 0; i < TALLA; i++)
10        printf("Introduce el valor de a[%d]:", i); scanf("%d", a[i]);
11
12    return 0;
13 }
```

- **99** ¿Es válida esta otra forma de leer una cadena? Pruébala en tu ordenador.

```

1 #include <stdio.h>
2
3 #define MAXLON 80
4
5 int main(void)
6 {
7     char cadena[MAXLON+1];
8
9     scanf("%s", &cadena[0]);
10    printf("La cadena leída es %s.\n", cadena);
11
12    return 0;
13 }
```

- **100** ¿Qué problema presenta esta otra versión del mismo programa?

```

1 #define MAXLON 10
2
3 int main(void)
4 {
5     char original[MAXLON+1] = "cadena";
6     char copia[MAXLON+1];
7     int i;
```

```

8
9  for (i = 0; i <= MAXLON; i++) {
10     if (copia[i] == '\0')
11         break;
12     else
13         copia[i] = original[i];
14 }
15
16 return 0;
17 }

```

► **101** Diseña un programa que lea una cadena y copie en otra una versión encriptada. La encriptación convertirá cada letra (del alfabeto inglés) en la que le sigue en la tabla ASCII (excepto en el caso de las letras «z» y «Z», que serán sustituidas por «a» y «A», respectivamente.) No uses la función *strcpy*.

► **102** Diseña un programa que lea una cadena que posiblemente contenga letras mayúsculas y copie en otra una versión de la misma cuyas letras sean todas minúsculas. No uses la función *strcpy*.

► **103** Diseña un programa que lea una cadena que posiblemente contenga letras mayúsculas y copie en otra una versión de la misma cuyas letras sean todas minúsculas. Usa la función *strcpy* para obtener un duplicado de la cadena y, después, recorre la copia para ir sustituyendo en ella las letras mayúsculas por sus correspondientes minúsculas.

► **104** Diseña un programa que lea una cadena y la invierta.

► **105** Diseña un programa que lea una *palabra* y determine si es o no es palíndromo.

► **106** Diseña un programa que lea una *frase* y determine si es o no es palíndromo. Recuerda que los espacios en blanco y los signos de puntuación no se deben tener en cuenta a la hora de determinar si la frase es palíndromo.

► **107** Escribe un programa C que lea dos cadenas y muestre el índice del carácter de la primera cadena en el que empieza, por primera vez, la segunda cadena. Si la segunda cadena no está contenida en la primera, el programa nos lo hará saber.

(Ejemplo: si la primera cadena es "un_ejercicio_de_ejemplo" y la segunda es "eje", el programa mostrará el valor 3.)

► **108** Escribe un programa C que lea dos cadenas y muestre el índice del carácter de la primera cadena en el que empieza por *última* vez una aparición de la segunda cadena. Si la segunda cadena no está contenida en la primera, el programa nos lo hará saber.

(Ejemplo: si la primera cadena es "un_ejercicio_de_ejemplo" y la segunda es "eje", el programa mostrará el valor 16.)

► **109** Escribe un programa que lea una línea y haga una copia de ella eliminando los espacios en blanco que haya al principio y al final de la misma.

► **110** Escribe un programa que lea repetidamente líneas con el nombre completo de una persona. Para cada persona, guardará temporalmente en una cadena sus iniciales (las letras con mayúsculas) separadas por puntos y espacios en blanco y mostrará el resultado en pantalla. El programa finalizará cuando el usuario escriba una línea en blanco.

► **111** Diseña un programa C que lea un entero n y una cadena a y muestre por pantalla el valor (en base 10) de la cadena a si se interpreta como un número en base n . El valor de n debe estar comprendido entre 2 y 16. Si la cadena a contiene un carácter que no corresponde a un dígito en base n , notificará el error y no efectuará cálculo alguno.

Ejemplos:

- si a es "ff" y n es 16, se mostrará el valor 255;
- si a es "f0" y n es 15, se notificará un error: «f no es un dígito en base 15»;
- si a es "1111" y n es 2, se mostrará el valor 15.

► **112** Diseña un programa C que lea una línea y muestre por pantalla el número de palabras que hay en ella.

► **113** Escribe un programa C que lea el nombre y los dos apellidos de una persona en tres cadenas. A continuación, el programa formará una sólo cadena en la que aparezcan el nombre y los apellidos separados por espacios en blanco.

► **114** Escribe un programa C que lea un verbo regular de la primera conjugación y lo muestre por pantalla conjugado en presente de indicativo. Por ejemplo, si lee el texto **programar**, mostrará por pantalla:

```

yo programo
tú programas
él programa
nosotros programamos
vosotros programáis
ellos programan

```

- ▶ **115** Diseña un programa C que lea dos cadenas y, si la primera es *menor o igual* que la segunda, imprima el texto «menor o igual».
- ▶ **116** ¿Qué valor devolverá la llamada `strcmp("21", "112")`?
- ▶ **117** Escribe un programa que lea dos cadenas, *a* y *b* (con capacidad para 80 caracteres), y muestre por pantalla `-1` si *a* es menor que *b*, `0` si *a* es igual que *b*, y `1` si *a* es mayor que *b*. Está prohibido que utilices la función `strcmp`.
- ▶ **118** ¿Qué problema presenta este programa?

```

1 #include <stdio.h>
2 #include <ctype.h>
3
4 int main(void)
5 {
6     char b[2] = "a";
7
8     if (isalpha(b))
9         printf("Es una letra\n");
10    else
11        printf("No es una letra\n");
12
13    return 0;
14 }

```

- ▶ **119** ¿Qué almacena en la cadena *a* la siguiente sentencia?

```
sprintf(a, "%d-%c-%d%s", 1, 48, 2, "si");
```

- ▶ **120** Escribe un programa que pida el nombre y los dos apellidos de una persona. Cada uno de esos tres datos debe almacenarse en una variable independiente. A continuación, el programa creará y mostrará una nueva cadena con los dos apellidos y el nombre (separado de los apellidos por una coma). Por ejemplo, Juan Pérez López dará lugar a la cadena "PérezLópez, Juan".
- ▶ **121** Modifica `normaliza.c` para que elimine, si los hay, los blancos inicial y final de la cadena normalizada.
- ▶ **122** Haz un programa que lea una frase y construya una cadena que sólo contenga sus letras minúsculas o mayúsculas en el mismo orden con que aparecen en la frase.
- ▶ **123** Haz un programa que lea una frase y construya una cadena que sólo contenga sus letras minúsculas o mayúsculas en el mismo orden con que aparecen en la frase, pero sin repetir ninguna.
- ▶ **124** Lee un texto por teclado (con un máximo de 1000 caracteres) y muestra por pantalla la frecuencia de aparición de cada una de las letras del alfabeto (considera únicamente letras del alfabeto inglés), sin distinguir entre letras mayúsculas y minúsculas (una aparición de la letra *e* y otra de la letra *E* cuentan como dos ocurrencias de la letra *e*).
- ▶ **125** Este programa es incorrecto. ¿Por qué? Aun siendo incorrecto, produce cierta salida por pantalla. ¿Qué muestra?

```

matriz_mal.c
matriz_mal.c
1 #include <stdio.h>
2
3 #define TALLA 3
4
5 int main(void)
6 {
7     int a[TALLA][TALLA];
8     int i, j;
9
10    for (i=0; i<TALLA; i++)
11        for (j=0; j<TALLA; j++)
12            a[i][j] = 10*i+j;
13
14    for (j=0; j<TALLA*TALLA; j++)
15        printf("%d\n", a[0][j]);
16
17    return 0;
18 }

```

- ▶ **126** En una estación meteorológica registramos la temperatura (en grados centígrados) cada hora durante una semana. Almacenamos el resultado en una matriz de 7×24 (cada fila de la matriz contiene las 24 mediciones de un día). Diseña un programa que lea los datos por teclado y muestre:

- La máxima y mínima temperaturas de la semana.
- La máxima y mínima temperaturas de cada día.
- La temperatura media de la semana.
- La temperatura media de cada día.
- El número de días en los que la temperatura media fue superior a 30 grados.

► **127** Representamos diez ciudades con números del 0 al 9. Cuando hay carretera que une directamente a dos ciudades i y j , almacenamos su distancia en kilómetros en la celda $d[i][j]$ de una matriz de 10×10 enteros. Si no hay carretera entre ambas ciudades, el valor almacenado en su celda de d es cero. Nos suministran un vector en el que se describe un trayecto que pasa por las 10 ciudades. Determina si se trata de un trayecto válido (las dos ciudades de todo par consecutivo están unidas por un tramo de carretera) y, en tal caso, devuelve el número de kilómetros del trayecto. Si el trayecto no es válido, indícalo con un mensaje por pantalla.

La matriz de distancias deberás inicializarla explícitamente al declararla. El vector con el recorrido de ciudades deberás leerlo de teclado.

► **128** Diseña un programa que lea los elementos de una matriz de 4×5 flotantes y genere un vector de talla 4 en el que cada elemento contenga el sumatorio de los elementos de cada fila. El programa debe mostrar la matriz original y el vector en este formato (evidentemente, los valores deben ser los que correspondan a lo introducido por el usuario):

	0	1	2	3	4	Suma
0	[+27.33	+22.22	+10.00	+0.00	-22.22]	-> +37.33
1	[+5.00	+0.00	-1.50	+2.50	+10.00]	-> +16.00
2	[+3.45	+2.33	-4.56	+12.56	+12.01]	-> +25.79
3	[+1.02	+2.22	+12.70	+34.00	+12.00]	-> +61.94
4	[-2.00	-56.20	+3.30	+2.00	+1.00]	-> -51.90

► **129** Extiende el programa de calculadora matricial para efectuar las siguientes operaciones:

- Producto de una matriz por un escalar. (La matriz resultante tiene la misma dimensión que la original y cada elemento se obtiene multiplicando el escalar por la celda correspondiente de la matriz original.)
- Transpuesta de una matriz. (La transpuesta de una matriz de $n \times m$ es una matriz de $m \times n$ en la que el elemento de la fila i y columna j tiene el mismo valor que el que ocupa la celda de la fila j y columna i en la matriz original.)

► **130** Una matriz tiene un valle si el valor de una de sus celdas es menor que el de cualquiera de sus 8 celdas vecinas. Diseña un programa que lea una matriz (el usuario te indicará de cuántas filas y columnas) y nos diga si la matriz tiene un valle o no. En caso afirmativo, nos mostrará en pantalla las coordenadas de *todos* los valles, sus valores y el de sus celdas vecinas.

La matriz debe tener un número de filas y columnas mayor o igual que 3 y menor o igual que 10. Las casillas que no tienen 8 vecinos no se consideran candidatas a ser valle (pues no tienen 8 vecinos).

Aquí tienes un ejemplo de la salida esperada para esta matriz de 4×5 :

$$\begin{pmatrix} 1 & 2 & 9 & 5 & 5 \\ 3 & 2 & 9 & 4 & 5 \\ 6 & 1 & 8 & 7 & 6 \\ 6 & 3 & 8 & 0 & 9 \end{pmatrix}$$

```

Valle en fila 2 columna 4:
9 5 5
9 4 5
8 7 6
Valle en fila 3 columna 2:
3 2 9
6 1 8
6 3 8

```

(Observa que al usuario se le muestran filas y columnas numeradas desde 1, y no desde 0.)

► **131** Modifica el programa del ejercicio anterior para que considere candidato a valle a cualquier celda de la matriz. Si una celda tiene menos de 8 vecinos, se considera que la celda es valle si su valor es menor que el de todos ellos.

Para la misma matriz del ejemplo del ejercicio anterior se obtendría esta salida:

```

Valle en fila 1 columna 1:
x x x
x 1 2
x 3 2
Valle en fila 2 columna 4:
9 5 5
9 4 5
8 7 6
Valle en fila 3 columna 2:
3 2 9
6 1 8
6 3 8
Valle en fila 4 columna 4:
8 7 6
8 0 9
x x x

```

► **132** Un programador, al copiar el programa, ha sustituido la línea que reza así:

```
while (i<lonfrase && !isalpha(frase[i])) i++; // Saltarse las no-letras iniciales.
```

por esta otra:

```
while (frase[i] != '\0' && !isalpha(frase[i])) i++; // Saltarse las no-letras iniciales.
```

¿Es correcto el programa resultante? ¿Por qué?

► **133** Un programador, al copiar el programa, ha sustituido la línea que reza así:

```
while (i<lonfrase && !isalpha(frase[i])) i++; // Saltarse las no-letras iniciales.
```

por esta otra:

```
while (frase[i] && !isalpha(frase[i])) i++; // Saltarse las no-letras iniciales.
```

¿Es correcto el programa resultante? ¿Por qué?

► **134** Un programador, al copiar el programa, ha sustituido la línea que reza así:

```
while (i<lonfrase && isalpha(frase[i])) palabra[palabras][j++] = frase[i++];
```

por esta otra:

```
while (isalpha(frase[i])) palabra[palabras][j++] = frase[i++];
```

¿Es correcto el programa resultante? ¿Por qué?

► **135** Un programador, al copiar el programa, ha sustituido la línea que reza así:

```
while (i<lonfrase && !isalpha(frase[i])) i++; // Saltarse las no-letras iniciales.
```

por esta otra:

```
while (!isalpha(frase[i])) palabra[palabras][j++] = frase[i++];
```

¿Es correcto el programa resultante? ¿Por qué?

► **136** Escribe un programa C que lea un texto (de longitud menor que 1000) y obtenga un vector de cadenas en el que cada elemento es una palabra distinta del texto (con un máximo de 500 palabras). Muestra el contenido del vector por pantalla.

► **137** Modifica el programa del ejercicio anterior para que el vector de palabras se muestre en pantalla ordenado alfabéticamente. Deberás utilizar el método de la burbuja para ordenar el vector.

► **138** Representamos la baraja de cartas con un vector de cadenas. Los palos son "oros", "copas", "espadas" y "bastos". Las cartas con números entre 2 y 9 se describen con el texto "número_de_palo" (ejemplo: "2_de_oros", "6_de_copas"). Los ases se describen con la cadena "as_de_palo", las sotas con "sota_de_palo", los caballos con "caballo_de_palo" y los reyes con "rey_de_palo".

Escribe un programa que genere la descripción de las 40 cartas de la baraja. Usa bucles siempre que puedas y compón las diferentes partes de cada descripción con *strcat* o *sprintf*. A continuación, baraja las cartas utilizando para ello el generador de números aleatorios y muestra el resultado por pantalla.

► **146** Diseña un programa C que gestione una agenda telefónica. Cada entrada de la agenda contiene el nombre de una persona y *hasta* 10 números de teléfono. El programa permitirá añadir nuevas entradas a la agenda y nuevos teléfonos a una entrada ya existente. El menú del programa permitirá, además, borrar entradas de la agenda, borrar números de teléfono concretos de una entrada y efectuar búsquedas por las primeras letras del nombre. (Si, por ejemplo, tu agenda contiene entradas para «José Martínez», «Josefa Pérez» y «Jaime Primero», una búsqueda por «Jos» mostrará a las dos primeras personas y una búsqueda por «J» las mostrará a todas.)

► **147** Define una función que reciba un **int** y devuelva su cuadrado.

► **148** Define una función que reciba un **float** y devuelva su cuadrado.

► **149** Define una función que reciba dos **float** y devuelva 1 («cierto») si el primero es menor que el segundo y 0 («falso») en caso contrario.

► **150** Define una función que calcule el volumen de una esfera a partir de su radio r . (Recuerda que el volumen de una esfera de radio r es $4/3\pi r^3$.)

► **151** El seno hiperbólico de x es

$$\sinh = \frac{e^x - e^{-x}}{2}.$$

Diseña una función C que efectúe el calculo de senos hiperbólicos. (Recuerda que e^x se puede calcular con la función `exp`, disponible incluyendo `math.h` y enlazando el programa ejecutable con la librería matemática.)

► **152** Diseña una función que devuelva «cierto» (el valor 1) si el año que se le suministra como argumento es bisiesto, y «falso» (el valor 0) en caso contrario.

► **153** La distancia de un punto (x_0, y_0) a una recta $Ax + By + C = 0$ viene dada por

$$d = \frac{Ax_0 + By_0 + C}{\sqrt{A^2 + B^2}}.$$

Diseña una función que reciba los valores que definen una recta y los valores que definen un punto y devuelva la distancia del punto a la recta.

► **154** Define una función que, dada una letra minúscula del alfabeto inglés, devuelva su correspondiente letra mayúscula. Si el carácter recibido como dato no es una letra minúscula, la función la devolverá inalterada.

► **155** ¿Qué error encuentras en esta función?

```

1 int minimo (int a, b, c)
2 {
3     if (a <= b && a <= c)
4         return a;
5     if (b <= a && b <= c)
6         return b;
7     return c;
8 }
```

► **156** Diseña una función que calcule el factorial de un entero n .

► **157** Diseña una función que calcule x^n , para n entero y x de tipo **float**. (Recuerda que si n es negativo, x^n es el resultado de multiplicar $1/x$ por sí mismo $-n$ veces.)

► **158** El valor de la función e^x puede aproximarse con el desarrollo de Taylor:

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

Diseña una función que aproxime el valor de e^x usando n términos del desarrollo de Taylor, siendo n un número entero positivo. (Puedes usar, si te conviene, la función de exponenciación del último ejercicio para calcular los distintos valores de x^i , aunque hay formas más eficientes de calcular $x/1!$, $x^2/2!$, $x^3/3!$, \dots , ¿sabes cómo? Plantéate cómo generar un término de la forma $x^i/i!$ a partir de un término de la forma $x^{i-1}/(i-1)!$.)

► **159** El valor de la función coseno puede aproximarse con el desarrollo de Taylor:

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Diseña una función que aproxime el coseno de un valor x usando n términos del desarrollo de Taylor, siendo n un número entero positivo.

- **160** Diseña una función que diga si un número es perfecto o no. Si el número es perfecto, devolverá «cierto» (el valor 1) y si no, devolverá «falso» (el valor 0). Un número es perfecto si es igual a la suma de todos sus divisores (excepto él mismo).
- **161** Diseña una función que diga si un número entero es o no es capicúa.
- **162** ¿Qué muestra por pantalla este programa?

```

contador_global.c
contador_global.c
1 #include <stdio.h>
2
3 int contador; // Variable global.
4
5 void fija(int a)
6 {
7     contador = a;
8 }
9
10 int decrementa(int a)
11 {
12     contador -= a;
13     return contador;
14 }
15
16 void muestra(int contador)
17 {
18     printf("[%d]\n", contador);
19 }
20
21 void cuenta_atras(int a)
22 {
23     int contador;
24     for (contador=a; contador >=0; contador--)
25         printf("%d ", contador);
26     printf("\n");
27 }
28
29 int main(void) {
30     int i;
31
32     contador = 10;
33     i = 1;
34     while (contador >= 0) {
35         muestra(contador);
36         cuenta_atras(contador);
37         muestra(i);
38         decrementa(i);
39         i *= 2;
40     }
41 }

```

- **163** El programa `dato.c` siempre genera la misma secuencia de números aleatorios. Para evitarlo, debes proporcionar una semilla diferente con cada ejecución del programa. El valor de la semilla se suministra como único argumento de la función `srand`. Modifica `dato.c` para que solicite al usuario la introducción del valor semilla.
- **164** Diseña una función que lea por teclado un entero positivo y devuelva el valor leído. Si el usuario introduce un número negativo, la función advertirá del error por pantalla y leerá nuevamente el número cuantas veces sea menester.
- **165** Diseña un procedimiento que reciba un entero n y muestre por pantalla n asteriscos seguidos con un salto de línea al final.
- **166** Diseña un procedimiento que, dado un valor de n , dibuje con asteriscos un triángulo rectángulo cuyos catetos midan n caracteres. Si n es 5, por ejemplo, el procedimiento mostrará por pantalla este texto:

```

1 *
2 **
3 ***
4 ****
5 *****

```

Puedes usar, si te conviene, el procedimiento desarrollado en el ejercicio anterior.

► **167** Diseña un procedimiento que reciba un número entero entre 0 y 99 y muestre por pantalla su transcripción escrita. Si le suministramos, por ejemplo, el valor 31, mostrará el texto «treinta y uno».

► **168** Estudia este programa y muestra gráficamente el contenido de la memoria cuando se van a ejecutar por primera vez las líneas 24, 14 y 5.

```

suma_cuadrados.c
suma_cuadrados.c
1 #include <stdio.h>
2
3 int cuadrado(int i)
4 {
5     return i * i;
6 }
7
8 int sumatorio(int a, int b)
9 {
10    int i, s;
11
12    s = 0;
13    for (i=a; i<=b; i++)
14        s += cuadrado(i);
15    return s;
16 }
17
18 int main(void)
19 {
20    int i, j;
21
22    i = 10;
23    j = 20;
24    printf("%d\n", sumatorio(i, j));
25    return 0;
26 }

```

► **169** Este programa muestra por pantalla los 10 primeros números primos. La función *siguiente* genera cada vez un número primo distinto. Gracias a la variable global *ultimoprimo* la función «recuerda» cuál fue el último número primo generado. Haz una traza paso a paso del programa (hasta que haya generado los 4 primeros primos). Muestra el estado de la pila y el de la zona de variables globales en los instantes en que se llama a la función *siguienteprimo* y cuando ésta devuelve su resultado

```

diez_primos.c
diez_primos.c
1 #include <stdio.h>
2
3 int ultimoprimo = 0;
4
5 int siguienteprimo(void)
6 {
7     int esprimo, i;
8
9     do {
10        ultimoprimo++;
11        esprimo = 1;
12        for (i=2; i<ultimoprimo/2; i++)
13            if (ultimoprimo % i == 0) {
14                esprimo = 0;
15                break;
16            }
17    } while (!esprimo);
18    return ultimoprimo;
19 }
20
21 int main(void)
22 {
23    int i;
24
25    printf("Los 10 primeros números primos\n");
26    for (i=0; i<10; i++)

```

```

27     printf("%d\n", siguienteprimo());
28     return 0;
29 }

```

► **170** Diseña un programa C que manipule polinomios de grado menor o igual que 10. Un polinomio se representará con un vector de **float** de tamaño 11. Si p es un vector que representa un polinomio, $p[i]$ es el coeficiente del término de grado i . Diseña un procedimiento *suma* con el siguiente perfil:

```
void suma(float p[], float q[], float r[])
```

El procedimiento modificará r para que contenga el resultado de sumar los polinomios p y q .

- **171** Diseña una función que, dada una cadena y un carácter, diga cuántas veces aparece el carácter en la cadena.
- **172** Diseña un procedimiento *ordena* que ordene un vector de enteros. El procedimiento recibirá como parámetros un vector de enteros y un entero que indique el tamaño del vector.
- **173** Diseña una función que devuelva el máximo de un vector de enteros. El tamaño del vector se suministrará como parámetro adicional.
- **174** Diseña una función que diga si un vector de enteros es o no es palíndromo (devolviendo 1 o 0, respectivamente). El tamaño del vector se suministrará como parámetro adicional.
- **175** Diseña una función que reciba dos vectores de enteros de idéntica talla y diga si son iguales o no. El tamaño de los dos vectores se suministrará como parámetro adicional.
- **176** Diseña un *procedimiento* que reciba un vector de enteros y muestre todos sus componentes en pantalla. Cada componente se representará separado del siguiente con una coma. El último elemento irá seguido de un salto de línea. La talla del vector se indicará con un parámetro adicional.
- **177** Diseña un *procedimiento* que reciba un vector de **float** y muestre todos sus componentes en pantalla. Cada componente se representará separado del siguiente con una coma. Cada 6 componentes aparecerá un salto de línea. La talla del vector se indicará con un parámetro adicional.
- **178** Diseña un procedimiento que modifique el valor del parámetro de tipo **float** para que valga la inversa de su valor cuando éste sea distinto de cero. Si el número es cero, el procedimiento dejará intacto el valor del parámetro.
Si a vale 2.0, por ejemplo, *inversa(&a)* hará que a valga 0.5.
- **179** Diseña un procedimiento que intercambie el valor de dos números enteros.
Si a y b valen 1 y 2, respectivamente, la llamada *intercambia(&a, &b)* hará que a pase a valer 2 y b pase a valer 1.
- **180** Diseña un procedimiento que intercambie el valor de dos números **float**.
- **181** Diseña un procedimiento que asigne a todos los elementos de un vector de enteros un valor determinado. El procedimiento recibirá tres datos: el vector, su número de elementos y el valor que que asignamos a todos los elementos del vector.
- **182** Diseña un procedimiento que intercambie el contenido completo de dos vectores de enteros de igual talla. La talla se debe suministrar como parámetro.
- **183** Diseña un procedimiento que asigne a un entero la suma de los elementos de un vector de enteros. Tanto el entero (su dirección) como el vector se suministrarán como parámetros.
- **184** Diseña una función que calcule la inversa de *calcula_xy*, es decir, que obtenga el valor del radio y del ángulo a partir de x e y .
- **185** Diseña una función que reciba dos números enteros a y b y devuelva, simultáneamente, el menor y el mayor de ambos. La función tendrá esta cabecera:

```
1 void minimax (int a, int b, int * min, int * max)
```

► **186** Diseña una función que reciba un vector de enteros, su talla y un valor de tipo entero al que denominamos *buscado*. La función devolverá (mediante return) el valor 1 si *buscado* tiene el mismo valor que algún elemento del vector y 0 en caso contrario. La función devolverá, además, la distancia entre *buscado* y el elemento más próximo a él.

La cabecera de la función ha de ser similar a ésta:

```
1 int busca(int vector[], int talla, int buscado, int * distancia)
```

Te ponemos un par de ejemplos para que veas qué debe hacer la función.

```

1 #include <stdio.h>
2
3 #define TALLA 6
4
5 // Define aquí la función
6 ...
7
8 int main(void)
9 {
10  int v[TALLA], distancia, encontrado, buscado, i;
11
12  for (i=0; i<TALLA; i++) {
13      printf("Introduce el elemento%d:\n", i);
14      scanf("%d", &v[i]);
15  }
16
17  printf("¿Qué valor busco?:\n");
18  scanf("%d", &buscado);
19
20  encontrado = busca(v, TALLA, buscado, &distancia);
21  if (encontrado)
22      printf("Encontré el valor%d.\n", buscado);
23  else
24      printf("No está. El elemento más próximo está a distancia%d.\n", distancia);
25
26  printf("¿Qué valor busco ahora?:\n");
27  scanf("%d", &buscado);
28
29  encontrado = busca(v, TALLA, buscado, &distancia);
30  if (encontrado)
31      printf("Encontré el valor%d.\n", buscado);
32  else
33      printf("No está. El elemento más próximo está a distancia%d.\n", distancia);
34
35  return 0;
36 }

```

Al ejecutar el programa obtenemos esta salida por pantalla:

```

Introduce el elemento: 0 ↵
Introduce el elemento: 5 ↵
Introduce el elemento: 10 ↵
Introduce el elemento: 15 ↵
Introduce el elemento: 20 ↵
Introduce el elemento: 25 ↵
¿Qué valor busco?: 5 ↵
Encontré el valor 5.
¿Qué valor busco ahora?: 17 ↵
No está. El elemento más próximo está a distancia 2.

```

- **187** Modifica la función del ejercicio anterior para que, además de la distancia al elemento más próximo, devuelva el valor del elemento más próximo.
- **188** Modifica la función del ejercicio anterior para que, además de la distancia al elemento más próximo y el elemento más próximo, devuelva el valor de su índice.
- **189** Este ejercicio y los siguientes de este bloque tienen por objeto construir una serie de funciones que permitan efectuar transformaciones afines sobre puntos en el plano. Los puntos serán variables de tipo **struct** *Punto*, que definimos así:

```

1 struct Punto {
2     float x, y;
3 };

```

Diseña un procedimiento *muestra_punto* que muestre por pantalla un punto. Un punto *p* tal que *p.x* vale 2.0 y *p.y* vale 0.2 se mostrará en pantalla así: (2.000000, 0.200000). El procedimiento *muestra_punto* recibirá un punto *por valor*.

Diseña a continuación un procedimiento que permita leer por teclado un punto. El procedimiento recibirá *por referencia* el punto en el que se almacenarán los valores leídos.

► **190** La operación de traslación permite desplazar un punto de coordenadas (x, y) a $(x+a, y+b)$, siendo el desplazamiento (a, b) un vector (que representamos con otro punto). Implementa una función que reciba dos parámetros de tipo punto y modifique el primero de modo que se traslade lo que indique el vector.

► **191** La operación de escalado transforma un punto (x, y) en otro (ax, ay) , donde a es un factor de escala (real). Implementa una función que escale un punto de acuerdo con el factor de escala a que se suministre como parámetro (un **float**).

► **192** Si rotamos un punto (x, y) una cantidad de θ *radianes* alrededor del origen, obtenemos el punto

$$(x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta).$$

Define una función que rote un punto la cantidad de *grados* que se especifique.

► **193** La rotación de un punto (x, y) una cantidad de θ *radianes* alrededor de un punto (a, b) se puede efectuar con una traslación con el vector $(-a, -b)$, una rotación de θ *radianes* con respecto al origen y una nueva traslación con el vector (a, b) . Diseña una función que permita trasladar un punto un número dado de grados alrededor de otro punto.

► **194** Diseña una función que diga si dos puntos son iguales.

► **195** Hemos definido un tipo registro para representar complejos así:

```
1 struct Complejo {
2     float real;
3     float imag;
4 };
```

Diseña e implementa los siguientes procedimientos para su manipulación:

- leer un complejo de teclado;
- mostrar un complejo por pantalla;
- el módulo de un complejo ($|a + bi| = \sqrt{a^2 + b^2}$);
- el opuesto de un complejo ($-(a + bi) = -a - bi$);
- el conjugado de un complejo ($\overline{a + bi} = a - bi$);
- la suma de dos complejos ($(a + bi) + (c + di) = (a + c) + (b + d)i$);
- la diferencia de dos complejos ($(a + bi) - (c + di) = (a - c) + (b - d)i$);
- el producto de dos complejos ($(a + bi) \cdot (c + di) = (ac - bd) + (ad + bc)i$);
- la división de dos complejos ($\frac{a+bi}{c+di} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$).

► **196** Define un tipo registro y una serie de funciones para representar y manipular fechas. Una fecha consta de un día, un mes y un año. Debes implementar funciones que permitan:

- mostrar una fecha por pantalla con formato *dd/mm/aaaa* (por ejemplo, el 7 de junio de 2001 se muestra así: 07/06/2001);
- mostrar una fecha por pantalla como texto (por ejemplo, el 7 de junio de 2001 se muestra así: 7 de junio de 2001);
- leer una fecha por teclado;
- averiguar si una fecha cae en año bisiesto;
- averiguar si una fecha es anterior, igual o posterior a otra, devolviendo los valores -1 , 0 o 1 respectivamente,
- comprobar si una fecha existe (por ejemplo, el 29 de febrero de 2002 no existe);
- calcular la diferencia de días entre dos fechas.

► **197** Vamos a diseñar un programa capaz de jugar al tres en raya. El tablero se representará con una matriz de 3×3 . Las casillas serán caracteres. El espacio en blanco representará una casilla vacía; el carácter 'o' representará una casilla ocupada con un círculo y el carácter 'x' representará una casilla marcada con una cruz.

- Diseña una función que muestre por pantalla el tablero.
- Diseña una función que detecte si el tablero está lleno.
- Diseña una función que detecte si algún jugador consiguió hacer tres en raya.



► **199** Implementa el juego del buscaminas. El juego del buscaminas se juega en un tablero de dimensiones dadas. Cada casilla del tablero puede contener una bomba o estar vacía. Las bombas se ubican aleatoriamente. El usuario debe descubrir todas las casillas que no contienen bomba. Con cada jugada, el usuario descubre una casilla (a partir de sus coordenadas, un par de letras). Si la casilla contiene una bomba, la partida finaliza con la derrota del usuario. Si la casilla está libre, el usuario es informado de cuántas bombas hay en las (como mucho) 8 casillas vecinas.

Este tablero representa, en un terminal, el estado actual de una partida sobre un tablero de 8×8 :

```

 abcdefgh
a 00001___
b 00112___
c 222_____
d _____
e ____3___
f _____
g 1_111111
h __100000

```

Las casillas con un punto no han sido descubiertas aún. Las casillas con un número han sido descubiertas y sus casillas vecinas contienen tantas bombas como se indica en el número. Por ejemplo, la casilla de coordenadas ('e', 'e') tiene 3 bombas en la vecindad y la casilla de coordenadas ('b', 'a'), ninguna.

Implementa un programa que permita seleccionar el nivel de dificultad y, una vez escogido, genere un tablero y permita jugar con él al jugador.

Los niveles de dificultad son:

- fácil: tablero de 8×8 con 10 bombas.
- medio: tablero de 15×15 con 40 bombas.
- difícil: tablero de 20×20 con 100 bombas.

Debes diseñar funciones para desempeñar cada una de las acciones básicas de una partida:

- dado un tablero y las coordenadas de una casilla, indicar si contiene bomba o no,
- dado un tablero y las coordenadas de una casilla, devolver el número de bombas vecinas,
- dado un tablero y las coordenadas de una casilla, modificar el tablero para indicar que la casilla en cuestión ya ha sido descubierta,
- dado un tablero, mostrar su contenido en pantalla,
- etc.

► **200** Vuelve a implementar las funciones de manipulación de puntos en el plano (ejercicios 189–194) para que no modifiquen el valor del registro `struct Punto` que se suministra como parámetro. En su lugar, devolverán el punto resultante como valor de retorno de la llamada a función.

► **201** Implementa nuevamente las funciones del ejercicio 195, pero devolviendo un nuevo complejo con el resultado de operar con el complejo o complejos que se suministran como parámetros.

► **202** Reescribe el programa para que no se use un variable de tipo `struct GrupoNaufragos` como almacén del grupo de naufragos, sino una matriz paralela a la matriz `espacio`.

Cada naufrago se representará con un '*' mientras permanezca perdido, y con un 'X' cuando haya sido descubierto.

► **203** Siempre que usamos `rand` en miniGalaxis calculamos un par de números aleatorios. Hemos definido un nuevo tipo y una función:

```

1 struct Casilla {
2     int fila, column;
3 };
4
5 struct Casilla casilla_al_azar(void)
6 {
7     struct Casilla casilla;

```

```

8
9  casilla.fila = rand() % FILAS;
10 casilla.columna = rand() % COLUMNAS;
11 return casilla;
12 }

```

Y proponemos usarlos así:

```

1 void pon_naufragos(struct GrupoNaufragos * grupoNaufragos, int cantidad)
2     /* Situa aleatoriamente cantidad náufragos en la estructura grupoNaufragos. */
3 {
4     int fila, columna, ya_hay_uno_ahi, i;
5     struct Casilla una_casilla;
6
7     grupoNaufragos->cantidad = 0;
8     while (grupoNaufragos->cantidad != cantidad) {
9         una_casilla = casilla_al_azar();
10        ya_hay_uno_ahi = 0;
11        for (i=0; i<grupoNaufragos->cantidad; i++)
12            if (una_casilla.fila == grupoNaufragos->naufrago[i].fila &&
13                una_casilla.columna == grupoNaufragos->naufrago[i].columna) {
14                ya_hay_uno_ahi = 1;
15                break;
16            }
17        if (!ya_hay_uno_ahi) {
18            grupoNaufragos->naufrago[grupoNaufragos->cantidad].fila = una_casilla.fila;
19            grupoNaufragos->naufrago[grupoNaufragos->cantidad].columna = una_casilla.columna;
20            grupoNaufragos->naufrago[grupoNaufragos->cantidad].encontrado = 0;
21            grupoNaufragos->cantidad++;
22        }
23    }
24 }

```

¿Es correcto el programa con estos cambios?

► **204** Como siempre que usamos *rand* calculamos un par de números aleatorios, hemos modificado el programa de este modo:

```

1 struct Naufrago naufrago_al_azar(void)
2 {
3     struct Naufrago naufrago;
4
5     naufrago.fila = rand() % FILAS;
6     naufrago.columna = rand() % COLUMNAS;
7     naufrago.encontrado = 0;
8     return naufrago;
9 }
10
11 void pon_naufragos(struct GrupoNaufragos * grupoNaufragos, int cantidad)
12     /* Situa aleatoriamente cantidad náufragos en la estructura grupoNaufragos. */
13 {
14     int fila, columna, ya_hay_uno_ahi, i;
15     struct Naufrago un_naufrago;
16
17     grupoNaufragos->cantidad = 0;
18     while (grupoNaufragos->cantidad != cantidad) {
19         un_naufrago = naufrago_al_azar();
20         ya_hay_uno_ahi = 0;
21         for (i=0; i<grupoNaufragos->cantidad; i++)
22             if (un_naufrago.fila == grupoNaufragos->naufrago[i].fila &&
23                 un_naufrago.columna == grupoNaufragos->naufrago[i].columna) {
24                 ya_hay_uno_ahi = 1;
25                 break;
26             }
27         if (!ya_hay_uno_ahi) {
28             grupoNaufragos->naufrago[grupoNaufragos->cantidad] = un_naufrago;
29             grupoNaufragos->cantidad++;
30         }

```

```

31 }
32 }

```

¿Es correcto el programa con estos cambios?

- ▶ **205** Modifica el juego para que el usuario pueda escoger el nivel de dificultad. El usuario escogerá el número de naufragos perdidos (con un máximo de 20) y el número de sondas disponibles.
- ▶ **206** Hemos construido una versión simplificada de Galaxis. El juego original sólo se diferencia de éste en las direcciones exploradas por la sonda: así como las sondas de miniGalaxis exploran 4 direcciones, las de Galaxis exploran 8. Te mostramos el resultado de lanzar nuestra primera sonda en las coordenadas 4J de un tablero de juego Galaxis:

```

  ABCDEFGHIJKLMNOPQRST
0 +++++.+++ .+++ .+++++
1 ++++++.++.++ .+++++
2 ++++++.+.+.+++++
3 ++++++. . .+++++
4 .....1.....
5 ++++++. . .+++++
6 ++++++.+.+.+++++
7 +++++.++.++ .+++++
8 +++++.+++ .+++ .+++++

```

Implementa el juego Galaxis.

- ▶ **207** Diseña una función que calcule recursivamente x^n . La variable x será de tipo **float** y n de tipo **int**.
- ▶ **208** Diseña una función recursiva que calcule el n -ésimo número de Fibonacci.
- ▶ **209** Diseña una función recursiva para calcular el número combinatorio n sobre m sabiendo que

$$\binom{n}{n} = 1,$$

$$\binom{n}{0} = 1,$$

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}.$$

- ▶ **210** Diseña un procedimiento recursivo llamado *muestra_bin* que reciba un número entero positivo y muestre por pantalla su codificación en binario. Por ejemplo, si llamamos a *muestra_bin*(5), por pantalla aparecerá el texto «101».
- ▶ **211** Dibuja el estado de la pila cuando se llega al caso base en la llamada recursiva *impar*(7).
- ▶ **212** Diseña una macro que calcule la tangente de una cantidad de radianes. Puedes usar las funciones *sin* y *cos* de `math.h`, pero ninguna otra.
- ▶ **213** Diseña una macro que devuelva el mínimo de dos números, sin importar si son enteros o flotantes.
- ▶ **214** Diseña una macro que calcule el valor absoluto de un número, sin importar si es entero o flotante.
- ▶ **215** Diseña una macro que decremente una variable entera si y sólo si es positiva. La macro devolverá el valor ya decrementado o inalterado, según convenga.
- ▶ **216** ¿Puedes usar la función *integra* para calcular la integral definida de la función matemática $\sin(x)$? ¿Cómo?
- ▶ **217** Diseña una función C capaz de calcular

$$\sum_{i=a}^b f(i),$$

siendo f una función matemática cualquiera que recibe un entero y devuelve un entero.

- ▶ **218** Diseña una función C capaz de calcular

$$\sum_{i=a}^b \sum_{j=c}^d f(i, j),$$

siendo f una función matemática cualquiera que recibe dos enteros y devuelve un entero.

- ▶ **219** Diseña una función que seleccione todos los números positivos de un vector de enteros. La función recibirá el vector original y un parámetro con su longitud y devolverá dos datos: un puntero al nuevo vector de enteros positivos y su longitud. El puntero se devolverá como valor de retorno de la función, y la longitud mediante un parámetro adicional (un entero pasado por referencia).

► **220** Desarrolla una función que seleccione todos los números de un vector de **float** mayores que un valor dado. Diseña un programa que llame correctamente a la función y muestre por pantalla el resultado.

► **221** Escribe un programa que lea por teclado un vector de **float** cuyo tamaño se solicitará previamente al usuario. Una vez leídos los componentes del vector, el programa copiará sus valores en otro vector distinto que ordenará con el método de la burbuja. Recuerda liberar toda memoria dinámica solicitada antes de finalizar el programa.

► **222** Escribe una función que lea por teclado un vector de **float** cuyo tamaño se solicitará previamente al usuario. Escribe, además, una función que reciba un vector como el leído en la función anterior y devuelva una copia suya con los mismos valores, pero ordenados de menor a mayor (usa el método de ordenación de la burbuja o cualquier otro que conozcas).

Diseña un programa que haga uso de ambas funciones. Recuerda que debes liberar toda memoria dinámica solicitada antes de finalizar la ejecución del programa.

► **223** Escribe una función que reciba un vector de enteros y devuelva otro con sus n mayores valores, siendo n un número menor o igual que la talla del vector original.

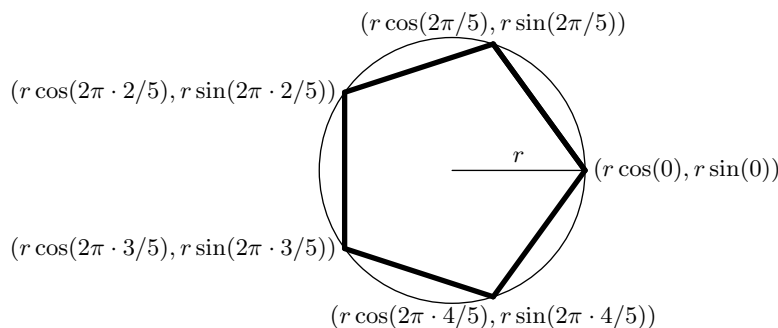
► **224** Escribe una función que reciba un vector de enteros y un valor n . Si n es menor o igual que la talla del vector, la función devolverá el un vector con las n primeras celdas del vector original. En caso contrario, devolverá un vector de n elementos con un copia del contenido del original y con valores nulos hasta completarlo.

► **225** ¿Funciona esta otra implementación de *perimetro_poligono*?

```

1 float perimetro_poligono(struct Poligono pol)
2 {
3     int i;
4     float perim = 0.0;
5
6     for (i=1; i<pol.puntos+1; i++)
7         perim +=
8             sqrt((pol.p[i%pol.puntos].x - pol.p[i-1].x) * (pol.p[i%pol.puntos].x - pol.p[i-1].x) +
9                 (pol.p[i%pol.puntos].y - pol.p[i-1].y) * (pol.p[i%pol.puntos].y - pol.p[i-1].y));
10    return perim;
11 }
```

► **226** Diseña una función que cree un polígono regular de n lados inscrito en una circunferencia de radio r . Esta figura muestra un pentágono inscrito en una circunferencia de radio r y las coordenadas de cada uno de sus vértices:



Utiliza la función para crear polígonos regulares de talla 3, 4, 5, 6, ... inscritos en una circunferencia de radio 1. Calcula a continuación el perímetro de los sucesivos polígonos y comprueba si dicho valor se aproxima a 2π .

► **227** Diseña un programa que permita manipular polinomios de cualquier grado. Un polinomio se representará con el siguiente tipo de registro:

```

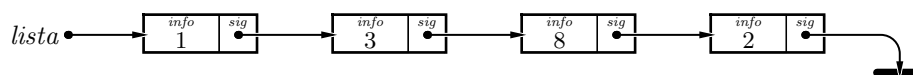
1 struct Polinomio {
2     float * p;
3     int grado;
4 };
```

Como puedes ver, el campo p es un puntero a **float**, o sea, un vector dinámico de **float**. Diseña y utiliza funciones que hagan lo siguiente:

- Leer un polinomio por teclado. Se pedirá el grado del polinomio y, tras reservar memoria suficiente para sus coeficientes, se pedirá también el valor de cada uno de ellos.
- Evaluar un polinomio $p(x)$ para un valor dado de x .
- Sumar dos polinomios. Ten en cuenta que cada uno de ellos puede ser de diferente grado y el resultado tendrá, en principio, grado igual que el mayor grado de los operandos. (Hay excepciones; piensa cuáles.)

- Multiplicar dos polinomios.
- ▶ **228** Diseña un programa que solicite la talla de una serie de valores enteros y dichos valores. El programa ordenará a continuación los valores mediante el procedimiento *mergesort*. (Ten en cuenta que el vector auxiliar que necesita *merge* debe tener capacidad para el mismo número de elementos que el vector original.)
- ▶ **229** Diseña una función que lea una cadena y construya otra con una copia invertida de la primera. La segunda cadena reservará sólo la memoria que necesite.
- ▶ **230** Diseña una función que lea una cadena y construya otra que contenga un ejemplar de cada carácter de la primera. Por ejemplo, si la primera cadena es "este_ejemplo", la segunda será "est_jmplo". Ten en cuenta que la segunda cadena debe ocupar la menor cantidad de memoria posible.
- ▶ **231** Implementa una función que reciba una cadena y devuelva una copia invertida. (Ten en cuenta que la talla de la cadena puede conocerse con *strlen*, así que no es necesario que suministres la talla explícitamente ni que devuelvas la talla de la memoria solicitada con un parámetro pasado por referencia.)
Escribe un programa que solicite varias palabras a un usuario y muestre el resultado de invertir cada una de ellas.
- ▶ **232** Diseña una *función* que reciba un número de filas y un número de columnas y devuelva una matriz dinámica de enteros con *filas* × *columnas* elementos.
- ▶ **233** Diseña un *procedimiento* que reciba un puntero a una matriz dinámica (sin memoria asignada), un número de filas y un número de columnas y devuelva, mediante el primer parámetro, una matriz dinámica de enteros con *filas* × *columnas* elementos.
- ▶ **234** En *muestra_matriz* hemos pasado la matriz *mat* por valor. ¿Cuántos bytes se copiarán en pila con cada llamada?
- ▶ **235** Diseña una nueva versión de *muestra_matriz* en la que *mat* se pase por referencia. ¿Cuántos bytes se copiarán en pila con cada llamada?
- ▶ **236** Diseña una función que sume dos matrices.
- ▶ **237** Pasar estructuras por valor puede ser ineficiente, pues se debe obtener una copia en pila de la estructura completa (en el caso de las matrices, cada variable de tipo **struct** *Matriz* ocupa 12 bytes —un puntero y dos enteros—, cuando una referencia supone la copia de sólo 4 bytes). Modifica la función que multiplica dos matrices para que sus dos parámetros se pasen por referencia.
- ▶ **238** Diseña una función que encuentre, si lo hay, un *punto de silla* en una matriz. Un punto de silla es un elemento de la matriz que es o bien el máximo de su fila y el mínimo de su columna a la vez, o bien el mínimo de su fila y el máximo de su columna a la vez. La función devolverá cierto o falso dependiendo de si hay algún punto de silla. Si lo hay, el valor del primer punto de silla encontrado se devolverá como valor de un parámetro pasado por referencia.
- ▶ **239** Diseña una función que devuelva cierto (valor 1) o falso (valor 0) en función de si una palabra pertenece o no a un diccionario.
- ▶ **240** Diseña una función que borre una palabra del diccionario.
- ▶ **241** Diseña una función que muestre por pantalla todas la palabras del diccionario que empiezan por un prefijo dado (una cadena).
- ▶ **242** Diseña una función que muestre por pantalla todas la palabras del diccionario que acaban con un sufijo dado (una cadena).
- ▶ **243** ¿Cuántas comparaciones se hacen en el peor de los casos en la búsqueda dicotómica de una palabra cualquiera en un diccionario con 8 palabras? ¿Y en un diccionario con 16 palabras? ¿Y en uno con 32? ¿Y en uno con 1024? ¿Y en uno con 1048576? (Nota: el valor 1048576 es igual a 2^{20} .)
- ▶ **244** Al insertar una nueva palabra en un diccionario hemos de comprobar si existía previamente y, si es una palabra nueva, averiguar en qué posición hay que insertarla. En la última versión presentada, esa búsqueda se efectúa recorriendo el diccionario palabra a palabra. Modifícala para que esa búsqueda sea dicotómica.
- ▶ **245** Diseña una función que funda dos diccionarios ordenados en uno sólo (también ordenado) que se devolverá como resultado. La fusión se hará de modo que las palabras que están repetidas en ambos diccionarios aparezcan una sola vez en el diccionario final.
- ▶ **246** Modifica *anyadir_telefono_a_entrada* para que compruebe si el teléfono ya había sido dado de alta. En tal caso, la función dejará intacta la lista de teléfonos de esa entrada.
- ▶ **247** Diseña una función que permita eliminar una entrada de la agenda a partir del nombre de una persona.
- ▶ **248** La agenda, tal y como la hemos implementado, está desordenada. Modifica el programa para que esté siempre ordenada.

► **249** Hemos diseñado un método (que mejoraremos en el siguiente apartado) que permite insertar elementos por el final de una lista y hemos necesitado un bucle. ¿Hará falta un bucle para insertar un elemento por delante en una lista cualquiera? ¿Cómo harías para convertir la última lista en esta otra?:



► **250** ¿Funcionará correctamente *longitud_lista* cuando le pasamos una lista vacía?

► **251** Diseña una función que reciba una lista de enteros con enlace simple y devuelva el valor de su elemento máximo. Si la lista está vacía, se devolverá el valor 0.

► **252** Diseña una función que reciba una lista de enteros con enlace simple y devuelva su media. Si la lista está vacía, se devolverá el valor 0.

► **253** Diseña un procedimiento que muestre el contenido de una lista al estilo Python. Por ejemplo, la lista de la última figura se mostrará como `[3, 8, 2]`. Fíjate en que la coma sólo aparece separando a los diferentes valores, no después de todos los números.

► **254** Diseña un procedimiento que muestre el contenido de una lista como se indica en el siguiente ejemplo. La lista formada por los valores 3, 8 y 2 se representará así:

->[3]->[8]->[2]->|

(La barra vertical representa a NULL.)

► **255** ¿Seguro que el bucle de *borraCola* funciona correctamente *siempre*? Piensa si hace lo correcto cuando se le pasa una lista formada por un solo elemento.

► **256** ¿Funcionan correctamente las funciones que hemos definido antes (cálculo de la longitud, inserción por cabeza y por cola y borrado de cabeza) cuando se suministra una lista compuesta por un único elemento?

► **257** ¿Funciona correctamente *pertenece* cuando se suministra NULL como valor de *lista*, es decir, cuando se suministra una lista vacía? ¿Y cuando se suministra una lista con un único elemento?

► **258** ¿Funciona *borraPrimeraOcurriencia* cuando ningún nodo de la lista contiene el valor buscado?

► **259** ¿Funciona correctamente en los siguientes casos?

- lista vacía;
- lista con un sólo elemento que no coincide en valor con el buscado;
- lista con un sólo elemento que coincide en valor con el buscado.

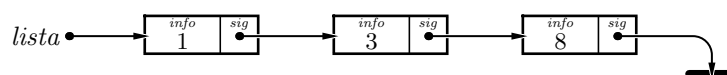
Si no es así, corrige la función.

► **260** ¿Funciona *borraValor* con listas vacías? ¿Y con listas de un sólo elemento? ¿Y con una lista en la que todos los elementos coinciden en valor con el entero que buscamos? Si falla en alguno de estos casos, corrige la función.

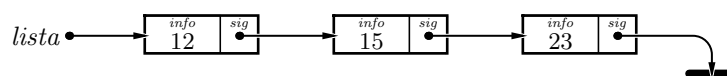
► **261** Modifica la función para que, si nos pasan un número de posición mayor que el número de elementos de la lista, no se realice inserción alguna.

► **262** Haz una traza de la inserción del valor 7 con *insertaEnOrden* en cada una de estas listas:

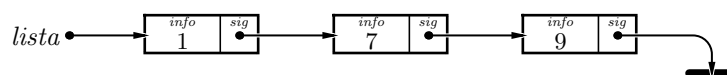
a)



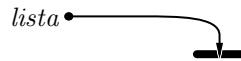
b)



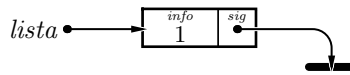
c)



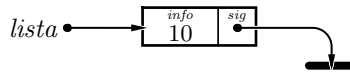
d)



e)



f)



- ▶ **263** Diseña una función de inserción ordenada en lista que inserte un nuevo nodo si y sólo si no había ningún otro con el mismo valor.
- ▶ **264** Determinar la pertenencia de un valor a una lista ordenada no requiere que recorras siempre toda la lista. Diseña una función que determine la pertenencia a una lista ordenada efectuando el menor número posible de comparaciones y desplazamientos sobre la lista.
- ▶ **265** Implementa una función que ordene una lista cualquiera mediante el método de la burbuja.
- ▶ **266** Diseña una función que diga, devolviendo el valor 1 o el valor 0, si una lista está ordenada o desordenada.
- ▶ **267** Diseña una función que añada a una lista una copia de otra lista.
- ▶ **268** Diseña una función que devuelva una lista con los elementos de otra lista que sean mayores que un valor dado.
- ▶ **269** Diseña una función que devuelva una lista con los elementos comunes a otras dos listas.
- ▶ **270** Diseña una función que devuelva una lista que es una copia *invertida* de otra lista.
- ▶ **271** Diseña una función que devuelva un «corte» de la lista. Se proporcionarán como parámetros dos enteros i y j y se devolverá una lista con una copia de los nodos que ocupan las posiciones i a $j - 1$, ambas incluidas.
- ▶ **272** Diseña una función que elimine un «corte» de la lista. Se proporcionarán como parámetros dos enteros i y j y se eliminarán los nodos que ocupan las posiciones i a $j - 1$, ambas incluidas.
- ▶ **273** Diseña una función que determine si un número pertenece o no a una lista con punteros a cabeza y cola.
- ▶ **274** Diseña una función que elimine el primer nodo con un valor dado en una lista con punteros a cabeza y cola.
- ▶ **275** Diseña una función que elimine todos los nodos con un valor dado en una lista con punteros a cabeza y cola.
- ▶ **276** Diseña una función que devuelva el elemento que ocupa la posición n en una lista con puntero a cabeza y cola. (La cabeza ocupa la posición 0.) La función devolverá como valor de retorno 1 o 0 para, respectivamente, indicar si la operación se pudo completar con éxito o si fracasó. La operación no se puede completar con éxito si n es negativo o si n es mayor o igual que la talla de la lista. El valor del nodo se devolverá en un parámetro pasado por referencia.
- ▶ **277** Diseña una función que devuelva un «corte» de la lista. Se recibirán dos índices i y j y se devolverá una nueva lista con punteros a cabeza y cola con una copia de los nodos que van del que ocupa la posición i al que ocupa la posición $j - 1$, ambos incluidos. La lista devuelta tendrá punteros a cabeza y cola.
- ▶ **278** Diseña una función de inserción ordenada en una lista ordenada con punteros a cabeza y cola.
- ▶ **279** Diseña una función que devuelva el menor valor de una lista ordenada con punteros a cabeza y cola.
- ▶ **280** Diseña una función que devuelva el mayor valor de una lista ordenada con punteros a cabeza y cola.
- ▶ **281** Diseña una función que añada a una lista con punteros a cabeza y cola una copia de otra lista con punteros a cabeza y cola.
- ▶ **282** Diseña una función que inserte un nuevo nodo al final de una lista doblemente enlazada.
- ▶ **283** Diseña una función que borre la cabeza de una lista doblemente enlazada. Presta especial atención al caso en el que la lista consta de un sólo elemento.
- ▶ **284** Reescribe la función de inserción en una posición dada para que no efectúe llamadas a la función *inserta_por_cabeza*.
- ▶ **285** Reescribe la función de inserción en una posición dada para que no efectúe llamadas a la función *inserta_por cola*. ¿Es más eficiente la nueva versión? ¿Por qué?

► **286** ¿Qué ocurriría si las últimas líneas de la función fueran éstas?:

```

1  ...
2  nuevo->sig = aux;
3  nuevo->ant = aux->ant;
4  aux->ant = nuevo;
5  aux->ant->sig = nuevo;
6  }
7  return lista;
8  }
```

¿Es correcta ahora la función? Haz una traza con un caso concreto.

► **287** Diseña una función que permita efectuar la inserción ordenada de un elemento en una lista con enlace doble que está ordenada.

► **288** Diseña una función que permita concatenar dos listas doblemente enlazadas. La función recibirá las dos listas y devolverá una lista nueva con una copia de la primera seguida de una copia de la segunda.

► **289** Diseña una función que devuelva una copia *invertida* de una lista doblemente enlazada.

► **290** Diseña una función que calcule la longitud de una lista doblemente enlazada con punteros a cabeza y cola.

► **291** Diseña una función que permita insertar un nuevo nodo en cabeza.

► **292** Diseña una función que permita insertar un nuevo nodo en cola.

► **293** Diseña una función que permita borrar el nodo de cabeza.

► **294** Diseña una función que elimine el primer elemento de la lista con un valor dado.

► **295** Diseña una función que elimine todos los elementos de la lista con un valor dado.

► **296** Diseña una función que inserte un nodo en una posición determinada que se indica por su índice.

► **297** Diseña una función que inserte ordenadamente en una lista ordenada.

► **298** Diseña una función que muestre por pantalla el contenido de una lista, mostrando el valor de cada celda en una línea. Los elementos se mostrarán en el mismo orden con el que aparecen en la lista.

► **299** Diseña una función que muestre por pantalla el contenido de una lista, mostrando el valor de cada celda en un línea. Los elementos se mostrarán en orden inverso.

► **300** Diseña una función que devuelva una copia *invertida* de una lista doblemente enlazada con puntero a cabeza y cola.

► **301** Rellena una tabla similar a la anterior para estas otras operaciones:

- a) Insertar ordenadamente en una lista ordenada.
- b) Insertar en una posición concreta.
- c) Buscar un elemento en una lista ordenada.
- d) Buscar el elemento de valor mínimo en una lista ordenada.
- e) Buscar el elemento de valor máximo en una lista ordenada.
- f) Unir dos listas ordenadas de modo que el resultado esté ordenado.
- g) Mostrar el contenido de una lista por pantalla.
- h) Mostrar el contenido de una lista en orden inverso por pantalla.

► **302** Vamos a montar una pila con listas. La pila es una estructura de datos en la que sólo podemos efectuar las siguientes operaciones:

- insertar un elemento en la cima,
- eliminar el elemento de la cima,
- consultar el valor del elemento de la cima.

¿Qué tipo de lista te parece más adecuado para implementar una pila? ¿Por qué?

► **303** Vamos a montar una cola con listas. La cola es una estructura de datos en la que sólo podemos efectuar las siguientes operaciones:

- insertar un elemento al final de la cola,
- eliminar el elemento que hay al principio de la cola,
- consultar el valor del elemento que hay al principio de la cola.

¿Qué tipo de lista te parece más adecuado para construir una cola? ¿Por qué?

► **304** La verdad es que insertar las canciones por la cabeza es el método menos indicado, pues cuando se recorra la lista para mostrarlas por pantalla aparecerán en orden inverso a aquél con el que fueron introducidas. Modifica *anyade_cancion* para que las canciones se inserten por la cola.

► **305** Y ya que sugerimos que insertes canciones por cola, modifica las estructuras necesarias para que la lista de canciones se gestione con una lista de registros con puntero a cabeza y cola.

► **306** Modifica *anyade_disco* para que los discos estén siempre ordenados alfabéticamente por intérprete y, para cada intérprete, por valor creciente del año de edición.

► **307** Modifica el programa para que se almacene la duración de cada canción (en segundos) junto al título de la misma.

► **308** La función de búsqueda de discos por intérprete se detiene al encontrar el primer disco de un intérprete dado. Modifica la función para que devuelva una lista con una copia de todos los discos de un intérprete. Usa esa lista para mostrar su contenido por pantalla con *muestra_coleccion* y elimínala una vez hayas mostrado su contenido.

► **309** Diseña una aplicación para la gestión de libros de una biblioteca. Debes mantener dos listas: una lista de libros y otra de socios. De cada socio recordamos el nombre, el DNI y el teléfono. De cada libro mantenemos los siguientes datos: título, autor, ISBN, código de la biblioteca (una cadena con 10 caracteres) y estado. El estado es un puntero que, cuando vale NULL, indica que el libro está disponible y, en caso contrario, apunta al socio al que se ha prestado el libro.

El programa debe permitir dar de alta y baja libros y socios, así como efectuar el préstamo de un libro a un socio y gestionar su devolución. Ten en cuenta que no es posible dar de baja a un socio que posee un libro en préstamo ni dar de baja un libro prestado.

► **310** ¿Qué ocupa en un fichero de texto cada uno de estos datos?

- | | | |
|-------|----------|----------------|
| a) 1 | d) -15 | g) -32768 |
| b) 0 | e) 128 | h) 2147483647 |
| c) 12 | f) 32767 | i) -2147483648 |

¿Y cuánto ocupa cada uno de ellos si los almacenamos en un fichero binario como valores de tipo **int**?

► **311** ¿Cómo se interpreta esta secuencia de bytes en cada uno de los siguientes supuestos?

00000000|00000000|00000000|00001100

- a) Como cuatro datos de tipo **char**.
- b) Como cuatro datos de tipo **unsigned char**.
- c) Como un dato de tipo **int**.
- d) Como un dato de tipo **unsigned int**.

► **312** Diseña un programa que añada al fichero *primos.txt* los 100 siguientes números primos. El programa leerá el contenido actual del fichero para averiguar cuál es el último primo del fichero. A continuación, abrirá el fichero en modo adición ("a") y añadirá 100 nuevos primos. Si ejecutásemos una vez *genera_primos* y, a continuación, dos veces el nuevo programa, el fichero acabaría conteniendo los 1200 primeros primos.

► **313** Diseña un programa que lea de teclado una frase y escriba un fichero de texto llamado *palabras.txt* en el que cada palabra de la frase ocupa una línea.

► **314** Diseña un programa que lea de teclado una frase y escriba un fichero de texto llamado *letras.txt* en el que cada línea contenga un carácter de la frase.

► **315** Modifica el programa *miniGalaxis* para que gestione una lista de records. Un fichero de texto, llamado *minigalaxis.records* almacenará el nombre y número de movimientos de los 5 mejores jugadores de todos los tiempos (los que completaron el juego usando el menor número de sondas).

► **316** Disponemos de dos ficheros: uno contiene un diccionario y el otro, un texto. El diccionario está ordenado alfabéticamente y contiene una palabra en cada línea. Diseña un programa que lea el diccionario en un vector de cadenas y lo utilice para detectar errores en el texto. El programa mostrará por pantalla las palabras del texto que no están en el diccionario, indicando los números de línea en que aparecen.

Supondremos que el diccionario contiene, a lo sumo, 1000 palabras y que la palabra más larga (tanto en el diccionario como en el texto) ocupa 30 caracteres.

(Si quieres usar un diccionario real como el descrito y trabajas en Unix, encontrarás uno en inglés en `/usr/share/dict/words` o `/usr/dict/words`. Puedes averiguar el número de palabras que contiene con el comando `wc` de Unix.)

► **317** Modifica el programa del ejercicio anterior para que el número de palabras del vector que las almacena se ajuste automáticamente al tamaño del diccionario. Tendrás que usar memoria dinámica.

Si usas un vector de palabras, puedes efectuar dos pasadas de lectura en el fichero que contiene el diccionario: una para contar el número de palabras y saber así cuánta memoria es necesaria y otra para cargar la lista de palabras en un vector dinámico. Naturalmente, antes de la segunda lectura deberás haber reservado la memoria necesaria.

Una alternativa a leer dos veces el fichero consiste en usar *realloc* juiciosamente: reserva inicialmente espacio para, digamos, 1000 palabras; si el diccionario contiene un número de palabras mayor que el que cabe en el espacio de memoria reservada, duplica la capacidad del vector de palabras (cuantas veces sea preciso si el problema se da más de una vez).

Otra posibilidad es usar una lista simplemente enlazada, pues puedes crearla con una primera lectura. Sin embargo, no es recomendable que sigas esta estrategia, pues no podrás efectuar una búsqueda dicotómica a la hora de determinar si una palabra está incluida o no en el diccionario.

► **318** Hemos escrito este programa para probar nuestra comprensión de *fgets* y *fputs* (presta atención también a los blancos, que se muestran con el carácter `□`):

```

1 #include <stdio.h>
2 #include <string.h>
3
4 #define MAXLON 100
5
6 int main (void)
7 {
8     FILE * f;
9     char s[MAXLON+1];
10    char * aux;
11
12    f = fopen("prueba.txt", "w");
13    fputs("si", f);
14    fputs("no\n", f);
15    fclose(f);
16
17    f = fopen("prueba.txt", "r");
18    aux = fgets(s, MAXLON, f);
19    printf("%s□%s\n", aux, s);
20    aux = fgets(s, MAXLON, f);
21    printf("%s□%s\n", aux, s);
22    fclose(f);
23
24    return 0;
25 }
```

Primera cuestión: ¿Cuántos bytes ocupa el fichero *prueba.txt*?

Al ejecutarlo, obtenemos este resultado en pantalla:

```

sino
□sino
(null)□sino
```

Segunda cuestión: ¿Puedes explicar con detalle qué ha ocurrido? (El texto «(null)» es escrito automáticamente por *printf* cuando se le pasa como cadena un puntero a NULL.)

► **319** La gestión de ficheros mediante su carga previa en memoria puede resultar problemática al trabajar con grandes volúmenes de información. Modifica el programa de la agenda para que no cargue los datos en memoria. Todas las operaciones (añadir datos y consultar) se efectuarán gestionando directamente ficheros.

► **320** Modifica el programa propuesto en el ejercicio anterior para que sea posible borrar entradas de la agenda. (Una posible solución pasa por trabajar con dos ficheros, uno original y uno para copias, de modo que borrar una información sea equivalente a no escribirla en la copia.)

► **321** Modifica el programa de la agenda para que se pueda mantener más de un teléfono asociado a una persona. El formato del fichero pasa a ser el siguiente:

- Una línea que empieza por la letra N contiene el nombre de una persona.
- Una línea que empieza por la letra D contiene la dirección de la persona nombre cuyo nombre acaba de aparecer.
- Una línea que empieza por la letra T contiene un número de teléfono asociado a la persona cuyo nombre apareció más recientemente en el fichero.

Ten en cuenta que no se puede asociar más de una dirección a una persona (y si eso ocurre en el fichero, debes notificar la existencia de un error), pero sí más de un teléfono. Además, puede haber líneas en blanco (o formadas únicamente por espacios en blanco) en el fichero. He aquí un ejemplo de fichero con el nuevo formato:

```

                                agenda.txt
1 N_Juan_Gil
2 D_Ronda_Mijares,_1220
3 T_964_123456
4
5 N_Ana_García
6 D_Plaza_del_Sol,_13
7 T_964-872777
8 T_964-872778
9
10
11 N_Pepe_Pérez
12 D_Calle_de_Arriba,_1
13 T_964_263_263
14 T_964_163_163
15 T_96_2663_663

```

► **322** En un fichero *matriz.mat* almacenamos los datos de una matriz de enteros con el siguiente formato:

- La primera línea contiene el número de filas y columnas.
- Cada una de las restantes líneas contiene tantos enteros (separados por espacios) como indica el número de columnas. Hay tantas líneas de este estilo como filas tiene la matriz.

Este ejemplo define una matriz de 3×4 con el formato indicado:

```

                                matriz.txt
1 3_4
2 1_0_3_4
3 0_-1_12_-1
4 3_0_99_-3

```

Escribe un programa que lea *matriz.mat* efectuando las reservas de memoria dinámica que corresponda y muestre por pantalla, una vez cerrado el fichero, el contenido de la matriz.

► **323** Modifica el programa del ejercicio anterior para que, si hay menos líneas con valores de filas que filas declaradas en la primera línea, se rellene el restante número de filas con valores nulos.

Aquí tienes un ejemplo de fichero con menos filas que las declaradas:

```

                                matriz_incompleta.txt
1 3_4
2 1_0_3_4

```

► **324** Diseña un programa que facilite la gestión de una biblioteca. El programa permitirá prestar libros. De cada libro se registrará al menos el título y el autor. En cualquier instante se podrá volcar el estado de la biblioteca a un fichero y cargarlo de él.

Conviene que la biblioteca sea una lista de nodos, cada uno de los cuales representa un libro. Uno de los campos del libro podría ser una cadena con el nombre del prestatario. Si dicho nombre es la cadena vacía, se entenderá que el libro está disponible.

► **325** ¿Qué aparecerá en pantalla si mostramos con el comando `cat` el contenido del fichero binario `otraprueba.dat` generado en este programa?:

```

otra_prueba.c
otra_prueba.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     FILE * fp;
6     int i, v[26];
7
8     fp = fopen("otra_prueba.dat", "wb");
9     for (i=97; i<123; i++)
10        v[i-97] = i;
11    fwrite(v, sizeof(int), 26, fp);
12    fclose(fp);
13
14    return 0;
15 }
```

(Una pista: el valor ASCII del carácter 'a' es 97.)

¿Y qué aparecerá si lo visualizas con el comando `od -c` (la opción `-c` indica que se desea ver el fichero carácter a carácter e interpretado como secuencia de caracteres).

► **326** Diseña un programa que genere un fichero binario `primos.dat` con los 1000 primeros números primos.

► **327** Diseña un programa que añada al fichero binario `primos.dat` (ver ejercicio anterior) los 100 siguientes números primos. El programa leerá el contenido actual del fichero para averiguar cuál es el último primo conocido. A continuación, abrirá el fichero en modo adición y añadirá 100 nuevos primos. Si ejecutásemos dos veces el programa, el fichero acabaría conteniendo los 1200 primeros primos.

► **328** Los dos programas anteriores suponen que hay diez puntos en el fichero `puntos.dat`. Modifícalo para que procesen tantos puntos como haya en el fichero.

► **329** Implementa un programa que genere un fichero llamado `puntos.dat` con 10 elementos del tipo `struct Punto`. Las coordenadas de cada punto se generarán aleatoriamente en el rango $[-10, 10]$. Usa el último programa para generar el fichero `puntos2.dat`. Comprueba que contiene el valor absoluto de los valores de `puntos.dat`. Si es necesario, diseña un nuevo programa que muestre por pantalla el contenido de un fichero de puntos cuyo nombre suministra por teclado el usuario.

► **330** Diseña una función de nombre `rebobina` que recibe un `FILE *` y nos ubica al inicio del mismo.

► **331** Diseña una función que reciba un `FILE *` (ya abierto) y nos diga el número de bytes que ocupa. Al final, la función debe dejar el cursor de lectura/escritura en el mismo lugar en el que estaba cuando se la llamó.

► **332** Diseña un programa que calcule y muestre por pantalla el máximo y el mínimo de los valores de un fichero binario de enteros.

► **333** Diseña un programa que calcule el máximo de los enteros de un fichero binario y lo intercambie por el que ocupa la última posición.

► **334** Nos pasan un fichero binario `dobles.dat` con una cantidad indeterminada de números de tipo `float`. Sabemos, eso sí, que los números están ordenados de menor a mayor. Diseña un programa que pida al usuario un número y determine si está o no está en el fichero.

En una primera versión, implementa una búsqueda secuencial que se detenga tan pronto estés seguro de que el número buscado está o no. El programa, en su versión final, deberá efectuar la búsqueda dicotómicamente (en un capítulo anterior se ha explicado qué es una búsqueda dicotómica).

► **335** Los dos programas anteriores pueden plantear problemas cuando trabajan con palabras que tiene 80 caracteres más el terminador. ¿Qué problemas? ¿Cómo los solucionarías?

► **336** Diseña un programa que lea una serie de valores enteros y los vaya escribiendo en un fichero hasta que el usuario introduzca el valor `-1` (que no se escribirá en el fichero). Tu programa debe, a continuación, determinar si la secuencia de números introducida en el fichero es palíndroma.

► **337** Deseamos gestionar una colección de cómics. De cada cómic anotamos los siguientes datos:

- Superhéroe: una cadena de hasta 20 caracteres.
- Título: una cadena de hasta 200 caracteres.
- Número: un entero.

- Año: un entero.
- Editorial: una cadena de hasta 30 caracteres.
- Sinopsis: una cadena de hasta 1000 caracteres.

El programa permitirá:

1. Dar de alta un cómic.
2. Consultar la ficha completa de un cómic dado el superhéroe y el número del episodio.
3. Ver un listado por superhéroe que muestre el título de todas sus historias.
4. Ver un listado por año que muestre el superhéroe y título de todas sus historias.

Diseña un programa que gestione la base de datos teniendo en cuenta que no queremos cargarla en memoria cada vez que ejecutamos el programa, sino gestionarla directamente sobre disco.

► **338** ¿Qué pasa si el usuario escribe la siguiente secuencia de caracteres como datos de entrada en la ejecución del programa?

2	0		3	.	1	2	3		4	5		5	\n
---	---	--	---	---	---	---	---	--	---	---	--	---	----

► **339** ¿Qué pasa si el usuario escribe la siguiente secuencia de caracteres como datos de entrada en la ejecución del programa?

2	0		3	.	1	2	3	\n	4	5		5	\n
---	---	--	---	---	---	---	---	----	---	---	--	---	----

► **340** ¿Qué pasa si el usuario escribe la siguiente secuencia de caracteres como datos de entrada en la ejecución del programa?

2	0				2		4	5			x	\n
---	---	--	--	--	---	--	---	---	--	--	---	----

► **341** ¿Qué pasa si el usuario escribe la siguiente secuencia de caracteres como datos de entrada en la ejecución del programa?

6		x		2	\n
---	--	---	--	---	----

(Prueba este ejercicio con el ordenador.)