

Control-2-FunPro.pdf



pablofa02



Fundamentos de la Programación



1º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Máster

Online en Ciberseguridad

Nº1 en España según El Mundo



Hasta el 46%
de beca



Mejor Máster
según el
Ranking de
EL MUNDO

Para ser el mejor hay que aprender
de los mejores.

IMF
Smart Education
Deloitte.

Infórmate

Que no te escriban poemas de amor
cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

2.- Diseña un procedimiento *ordenar* que realice una ordenación de menor a mayor de una lista de números enteros almacenados en el array unidimensional que recibe como parámetro. Para llevar a cabo dicha ordenación se seguirá el siguiente proceso (método de Selección):

- Buscar el mínimo elemento de la lista e intercambiarlo con el primero.
- Buscar el siguiente mínimo elemento del resto de la lista e intercambiarlo con el segundo.
- Buscar el siguiente mínimo elemento del resto de la lista e intercambiarlo con el tercero.
- Así sucesivamente.

En general: Buscar el mínimo elemento entre una posición i y el final de la lista e intercambiarlo con el elemento de esa posición i . El valor de i comienza en la primera posición y se aumenta de uno en uno.

Importante: sólo se completará el código del procedimiento *ordenar* en el fichero *ejercicio2.cpp* proporcionado en el campus virtual. Se podrán añadir más procedimientos y funciones si son necesarios para la solución de *ordenar*.

La ejecución del código suministrado una vez diseñado el procedimiento *ordenar* será:

```
El array antes de ordenarlo: 2 7 12 89 5 2 9 10  
El array después de ordenarlo: 2 2 5 7 9 10 12 89
```

1.- Diseña una función *mayorPrimo* que recibe como parámetro un array de tamaño TAM (una constante definida) relleno de números naturales y devuelve el mayor número primo almacenado. En caso de que no haya números primos, la función devolverá el valor 0. Nota: el número 1 no se considera primo.

Importante: No debes modificar el código ya proporcionado en el campus virtual (*ejercicio1.cpp*) para probar la función. Debes completar el código de la función *mayorPrimo*. Si para su diseño necesitas más procedimientos o funciones, puedes añadirlos. Si el código no funciona correctamente la puntuación de este problema será de 0 puntos.

La ejecución del código suministrado una vez diseñada la función *mayorPrimo* mostrará por pantalla lo siguiente:

```
El contenido del array 1 de prueba es: 6 4 12 0 8 9 46 15 21 12  
En el array no hay ningún primo  
  
El contenido del array 2 de prueba es: 8 22 3 6 2 7 56 11 5 9  
El mayor primo del array es: 11
```

2.- Un conjunto es una colección de elementos no repetidos. **Se ha definido el tipo Conjunto** que permitirá almacenar hasta un máximo de MAX números naturales, siendo MAX una constante dada. **Se ha diseñado un programa** (*ejercicio2.cpp*, ofrecido en el campus virtual) para trabajar con conjuntos de números naturales. Además de la definición de dicho tipo, aparece el algoritmo principal (*main*) y dos subalgoritmos de lectura/escritura (*leer* y *escribir*). Para **completar el programa**, debes realizar los **dos** siguientes **procedimientos** (se podrán añadir más procedimientos y funciones si son necesarios para la solución):

- *calcularInterseccion* que recibe como parámetros de entrada dos conjuntos y devuelve como parámetro de salida el conjunto resultante de efectuar la intersección entre ambos. La intersección de dos conjuntos es un nuevo conjunto formado por aquellos elementos que pertenecen a ambos conjuntos.
- *calcularUnion* que recibe como parámetros de entrada dos conjuntos y devuelve como parámetro de salida el conjunto resultante de efectuar la unión entre ambos. La unión de dos conjuntos es un nuevo conjunto formado por aquellos elementos que pertenecen a alguno de los dos conjuntos (o a ambos). La operación tendrá otro parámetro de salida de tipo booleano, por el que devolverá true si la operación se ha podido realizar con éxito y false si no se ha podido realizar porque no hay cabida para almacenar todos los elementos que formarían la unión.

Ejemplo 1 de ejecución:

```
Primer conjunto:  
Introduzca el numero de elementos a leer (<= 10): 5  
Introduzca 5 numeros naturales diferentes: 4 2 6 15 9  
  
Segundo conjunto:  
Introduzca el numero de elementos a leer (<= 10): 6  
Introduzca 6 numeros naturales diferentes: 6 3 4 26 1 2  
  
Interseccion:  
Los elementos del conjunto son: 4 2 6  
  
Union:  
Los elementos del conjunto son: 4 2 6 15 9 3 26 1
```

Ejemplo 2 de ejecución:

```
Primer conjunto:  
Introduzca el numero de elementos a leer (<= 10): 5  
Introduzca 5 numeros naturales diferentes: 4 2 6 15 9  
  
Segundo conjunto:  
Introduzca el numero de elementos a leer (<= 10): 8  
Introduzca 8 numeros naturales diferentes: 1 10 2 7 3 8 11 17  
  
Interseccion:  
Los elementos del conjunto son: 2  
Union:  
La Union no se ha podido realizar
```

Que no te escriban poemas de amor cuando terminen la carrera

(a nosotros por suerte nos pasa)



Ayer a las 20:20

Oh Wuolah wuolitah
Tu que eres tan bonita

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

No si antes decirte
Lo mucho que te voy a recordar



Envía un mensaje...



WUOLAH

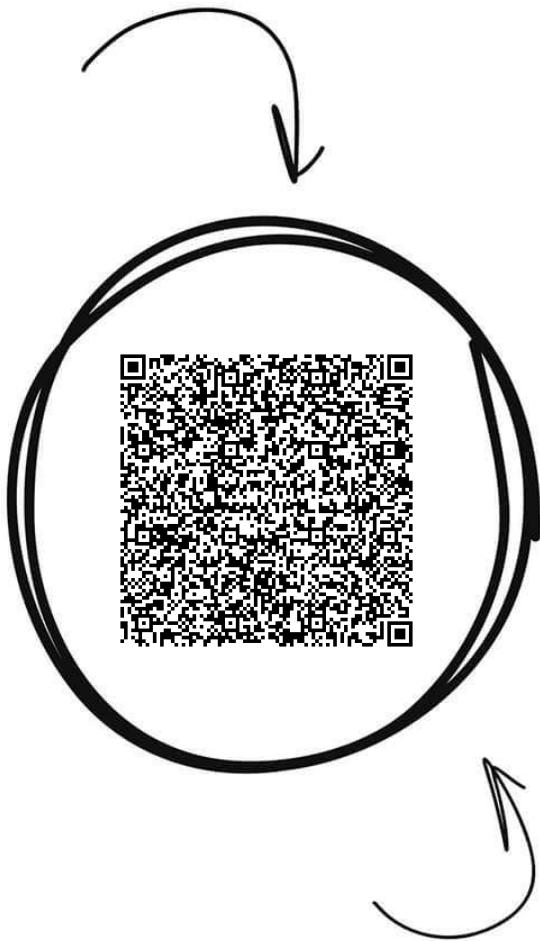


Fundamentos de la Programación



Comparte estos flyers en tu clase y consigue más dinero y recompensas

- 1 Imprime esta hoja
- 2 Recorta por la mitad
- 3 Coloca en un lugar visible para que tus compis puedan escanear y acceder a apuntes
- 4 Llévate dinero por cada descarga de los documentos descargados a través de tu QR



Note bank of the

WUOLAH



- 1.- Diseña una función** *valorDominante* que recibe como parámetro un array completo de tamaño TAM (una constante establecida) de valores naturales y devuelve el valor dominante del mismo o -1 en caso de que dicho valor no exista. El valor dominante de un array de naturales es el elemento que se repite un número de veces mayor que la mitad del tamaño del array.

Importante: sólo se completará el código de la función *valorDominante* en el fichero *ejercicio1.cpp* proporcionado en el campus virtual. Puedes añadir más procedimientos o funciones si lo estimas necesario. No se debe modificar el resto del código proporcionado. La puntuación de este problema será de 1 punto sólo en el caso de que la búsqueda funcione correctamente y se haga de forma que si se encuentra el valor dominante, se detenga la misma. En otro caso la puntuación será de 0 puntos.

La ejecución del código suministrado (tras diseñar la función solicitada) será:

```
El elemento dominante del primer array es: 3  
El elemento dominante del segundo array es: -1  
El elemento dominante del tercer array es: 4
```

- 1.- Diseña una función** *ordenado* que recibe como parámetro un array completo de tamaño TAM (una constante establecida) de valores enteros y devuelve un valor booleano indicando si el array está o no ordenado correctamente de menor a mayor valor.

Importante: sólo se completará el código de la función *ordenado* en el fichero *ejercicio1.cpp* proporcionado en el campus virtual. Puedes añadir más procedimientos o funciones si lo estimas necesario. No se debe modificar el resto del código proporcionado. La puntuación de este problema será de 1 punto sólo en el caso de que la búsqueda funcione correctamente y se haga de forma eficiente. En otro caso la puntuación será de 0 puntos.

La ejecución del código suministrado (tras diseñar la función solicitada) será:

```
El primer array SI esta ordenado  
El segundo array NO esta ordenado
```

- 1.- Diseña un procedimiento** *menorEstricto* con tres parámetros. El primero es de entrada para recibir un array completo de tamaño TAM (una constante establecida) de números enteros. El segundo parámetro es de salida e indicará si se ha encontrado el valor menor estricto (menor único) en el array o no. El tercer parámetro es de salida y será ese valor menor estricto en caso de que exista.

Importante: sólo se completará el código del procedimiento *menorEstricto* en el fichero *ejercicio1.cpp* proporcionado en el campus virtual. Puedes añadir más procedimientos o funciones si lo estimas necesario. No se debe modificar el resto del código proporcionado. La puntuación de este problema será de 1 punto sólo en el caso de que el procedimiento funcione correctamente. En otro caso la puntuación será de 0 puntos.

La ejecución del código suministrado (tras diseñar el procedimiento solicitado) será:

```
El menor estricto del primer array es: 3  
El segundo array no tiene menor estricto  
El menor estricto del tercer array es: 2
```

- 1.- Dado un tipo *array* de 10 números enteros, se debe desarrollar una **función** que devuelva el índice del *array* donde se encuentra el primer elemento (aquel con menor valor de índice) que sea **mayor o igual** que la **media** de los valores almacenados en el array. Se valorará la eficiencia del algoritmo desarrollado.

Por ejemplo, para los números {3, 1, 4, 0, 7, 2, 5, 9, 8, 6} devuelve el índice 4 (que se corresponde con el índice del elemento con valor 7).

Importante: sólo se completará el código del subprograma `buscar_mayig_media` en el fichero `ejercicio1.cpp` proporcionado en el campus virtual. Puedes añadir más subprogramas si lo consideras necesario. No se debe modificar el resto del código proporcionado. La puntuación de este problema será de 1 punto sólo en el caso de que el subprograma funcione correctamente y sea eficiente. En otro caso la puntuación será de 0 puntos.

La ejecución del código suministrado (tras diseñar el procedimiento solicitado) será, para los datos de entrada 3 1 4 0 7 2 5 9 8 6:

```
Introduce 10 números: 3 1 4 0 7 2 5 9 8 6
El primer elemento mayor o igual que la media se encuentra en 4
```

- 2.- **Diseña un algoritmo** que lea de teclado 3 listas (`list1`, `list2`, `list3`) de números enteros. Para cada lista, se leerán valores hasta encontrar un 0 (que actúa como terminador y no forma parte de la lista) o hasta que se hayan leído TAM (una constante establecida, por ejemplo 10) valores. En cada lista, los elementos repetidos se desechan (no cuentan y no se almacenan). Una vez leídas las tres listas, el algoritmo mostrará por pantalla en primer lugar los valores de dichas listas y después los posibles tríos de números (`num1` de la `list1`, `num2` de la `list2`, `num3` de la `list3`) que satisfagan la relación `num1+num2=num3`.

Ejemplo (para TAM = 10):

<p>Entrada:</p> <p>Introduzca Lista 1: 3 -2 3 5 -2 0 Introduzca Lista 2: 6 3 4 18 0 Introduzca Lista 3: 2 4 3 5 8 6 13 9 1 38 7 14 0</p> <p>Salida:</p> <p>Lista 1: 3 -2 5 Lista 2: 6 3 4 18 Lista 3: 2 4 3 5 8 6 13 9 1 38 Los trios de numeros son: 3 6 9 3 3 6 -2 6 4 -2 3 1 -2 4 2 5 3 8 5 4 9</p>
--

Que no te escriban poemas de amor cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirme
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

- 1.- Vamos a trabajar con **listas** de números enteros **de hasta un tamaño máximo de MAX** (una constante cualquiera). **Define** el **tipo de datos TLista** para ello y **diseña** un **procedimiento criba** que recibe como parámetros de entrada una lista de números enteros **list1** de tipo **TLista** y un número natural **x**. El procedimiento devolverá como parámetro de salida otra lista **list2** de tipo **TLista** que contendrá sólo aquellos números de **list1** que están repetidos **x** veces. En la lista **list2** no habrá elementos repetidos.

Ejemplo:

```
El contenido de list1 es: 1 3 4 3 1 3 0 -6 4  
El valor de x es: 2  
El contenido de list2 será: 1 4
```

- 2.- **Diseña** la siguiente **función**:

```
unsigned numOcurrencias(const TNumeros& numeros,  
                        const TPermutacion& permutacion)
```

La función recibe dos parámetros de entrada:

- **numeros**, un array de tipo **TNumeros** de tamaño **TAM1** que contiene números naturales. **Define** previamente el **tipo TNumeros**.
- **permutacion**, un array de tipo **TPermutacion** de tamaño **TAM2**, que contiene también números naturales. **Define** previamente el **tipo TPermutacion**.

La función devuelve el número de ocurrencias de cualquier permutación (mismos elementos aunque puedan estar en orden diferente) de los elementos del array **permutacion** en el array **numeros**, dispuestos de forma consecutiva.

Para calcular las ocurrencias del array **permutacion** en el array **numeros** se hace lo siguiente (para el ejemplo mostrado suponemos que **TAM1** es 11 y **TAM2** es 4 y que la parte sombreada del array **numeros** es lo que se comprueba en cada paso):

- 1) Se “coloca” el array **permutacion** sobre el array **numeros** en la posición 0 de éste y se comprueba si los elementos de **permutacion** son una permutación de la parte del array **numeros** sobre la que está colocado el array **permutacion**. Por ejemplo:

1	4	1	12	permutacion	Sí hay ocurrencia						
12	1	1	4	1	7	14	1	12	12	4	numeros

- 2) Se “desplaza” el array **permutacion** a la posición 1 sobre el array **numeros** y se hace la misma comprobación.

1	4	1	12	permutacion	NO hay ocurrencia						
12	1	1	4	1	7	14	1	12	12	4	numeros

- 3) Se repite el mismo proceso mientras sea posible “colocar” el array **permutacion** sobre el array **numeros**.

Permutacion	1	4	1	12	No hay ocurrencia						
12	1	1	4	1	7	14	1	12	12	4	numeros

Para este ejemplo la función devolvería el valor 1, ya que hay una sola ocurrencia.



NOTAS PARA LA REALIZACIÓN DEL EXAMEN:

- La solución se almacenará en la carpeta **FPGIIAC2**, dentro de **Documentos**. Si la carpeta ya existe, debe borrarse todo su contenido. En otro caso, debe crearse.
- Los nombres de los ficheros con la solución para los ejercicios 1 y 2 serán **ejercicio1.cpp**, **ejercicio2.cpp** respectivamente.
- Al inicio del contenido de cada fichero deberá aparecer un comentario con **el nombre del alumno, titulación, grupo y código del equipo** que se está utilizando (cada dato en una línea diferente).
- **Debe consultarse el documento “Obligaciones y Recomendaciones Estilo de Programación”, disponible en el Campus Virtual de la asignatura, con objeto de tener en cuenta los puntos allí señalados en las soluciones a los ejercicios.**
- Una vez terminado el examen, se subirán los ficheros ***.cpp** a la tarea creada en el **campus virtual** para ello.
- **No está permitido:**
 - Utilizar documentación electrónica o impresa.
 - Intercambiar documentación con otros compañeros.
 - Utilizar soportes de almacenamiento.
 - Utilizar dispositivos electrónicos (móviles, tablets, ...)

1. Escriba un programa que lea un determinado número de valores enteros (máximo 10). El programa habrá de pedir dicho valor hasta que éste sea menor o igual que dicho máximo. Una vez leída esa cantidad de números, debe localizar la posición del primer número primo que haya en esa lista, mostrando el número primo y su posición. A continuación deberá eliminar de la lista dicho elemento. Para la eliminación se ha de mantener el orden relativo de los elementos del array original, esto es, se han de hacer desplazamientos dentro del array y eliminar el hueco. Si el elemento a eliminar estuviera repetido, sólo se eliminará la primera ocurrencia del mismo. En caso de no existir ningún número primo se indicará y no se hará nada más.

Para la resolución del problema utiliza la estructura de datos adecuada para la implementación de listas o arrays incompletos (registro). Recuerda que los elementos en el array han de estar almacenados de forma contigua. Además, es necesaria la utilización de subalgoritmos para la resolución del problema. A continuación, se muestran varios ejemplos de la ejecución del programa:

Introduzca la cantidad de elementos de la lista (MAXIMO 10): 6
Introduzca los 6 elementos: 4 8 9 10 20 15
No hay primos en la lista.

Introduzca la cantidad de elementos de la lista: 11
Introduzca la cantidad de elementos de la lista: 10
Introduzca los 10 elementos: 4 9 14 12 4 8 3 6 5 10
El primer primo es 3 y está en la posición 6
La lista tras la eliminación de 3: 4 9 14 12 4 8 6 5 10

Introduzca la cantidad de elementos de la lista: 6
Introduzca los 6 elementos: 2 3 4 8 6 7
El primer primo es 2 y está en la posición 0
La lista tras la eliminación de 2: 3 4 8 6 7

2. Define un tipo de datos (`TLista`) para poder almacenar una lista de hasta un máximo de MAX (una constante definida) números naturales. El tipo debe definirse de forma que se pueda controlar en todo momento la cantidad de números almacenados.

Diseña un programa que:

Lea por teclado un número natural (`numRot`) que se utilizará más adelante. Después, leerá dos listas de números naturales y las almacenará en dos variables (`list1` y `list2`) del tipo previamente definido `TLista`. Para introducir cada lista, el usuario dará una secuencia de números naturales separados por espacios en blanco y terminará la misma introduciendo un número negativo. Si el usuario da más de MAX números antes de introducir el número negativo, sólo se almacenarán los MAX primeros. Tras leer la primera lista y antes de leer la segunda, se debe invocar a la siguiente instrucción para eliminar de la entrada posibles números sobrantes: `cin.ignore(1000, '\n')`. Tras leer las dos listas, el programa mostrará por pantalla su contenido.

A continuación, el programa modificará las dos listas almacenadas realizando `numRot` rotaciones de sus números según el sentido de las agujas del reloj y teniendo en cuenta que al último número de la lista le sigue el primero. En una rotación, el primer elemento pasa a ser el segundo, el segundo pasa a ser el tercero, ... y el último pasa a ser el primero. Tras realizar estas modificaciones, el programa mostrará por pantalla el contenido de las dos listas.

Por último, el programa modificará el contenido de la `list1`. Para ello, eliminará aquellos elementos de la misma que se encuentren también en la `list2`. Los números que se queden en `list1` deben mantener su orden posicional relativo. Tras esta modificación, el programa mostrará el contenido de ambas listas por pantalla.

Ejemplo de ejecución del programa para una constante definida `MAX = 5`:

```
Introduzca el numero de rotaciones: 6
Introduzca la primera lista: 2 5 1 4 -1
Introduzca la segunda lista: 3 8 7 4 9 8 3 -1
lista1 = 2 5 1 4
lista2 = 3 8 7 4 9
Tras 6 rotaciones:
lista1 = 1 4 2 5
lista2 = 9 3 8 7 4
Tras eliminar los elementos de lista1 que estan en lista2:
lista1 = 1 2 5
lista2 = 9 3 8 7 4
```



- 1.- Diseñe un algoritmo que lea de teclado un número natural M (mayor que 0 y menor o igual que un valor MAX constante conocido). A continuación este algoritmo leerá una sucesión indefinida de números naturales acabada en 0. El algoritmo debe mostrar los M valores mayores de la sucesión, así como la posición en que aparecen cada uno de ellos dentro de la misma. En la sucesión cada número puede aparecer repetido un número máximo MAX REP de veces. Si el número de elementos de la sucesión es menor que M, el algoritmo mostrará todos los valores y sus posiciones.

Introduzca el valor de M: 5

Introduzca una secuencia de números acabada en 0:

2 3 4 7 28 4 5 1 1 1 9 7 4 4 28 2 1 3 3 6 0

Los 5 Mayores y sus posiciones son:

7: 4 12

28: 5 15

9: 11

5: 7

6: 20

Una posible estructura para resolver el problema sería la siguiente:

```
const unsigned MAX=5;
const unsigned MAX REP=4;
typedef unsigned TPosiciones [MAX REP];
struct TNumero{
    unsigned num;
    TPosiciones pos;
    unsigned ocupa;
};
typedef TNumero TSecuencia [MAX];
struct TNumeros{
    TSecuencia numeros;
    unsigned tam;
};
```

Que no te escriban poemas de amor cuando terminen la carrera ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

2. Diseña una función que recibe como parámetros de entrada un array de MAX (una constante definida) números enteros a y un número entero, y devuelve true si el número num está contenido en a y false en otro caso. Si num está en la colección, la búsqueda se detendrá en el momento de encontrarlo. Diseña la función principal (main) para probar el funcionamiento de la función. Para ello, se leerá de teclado una colección de MAX números enteros con los que se llenará el array y también se leerá el número entero a buscar en la colección, se invocará a la función implementada y se mostrará por pantalla una indicación de si el número está o no en la colección. Un ejemplo de ejecución sería (MAX = 10):

```
Introduzca 10 numeros enteros: 4 25 -3 4 2 17 9 5 -7 8
Introduzca el numero a buscar: 2
El numero 2 SI esta en la colección
```

3. Se dispone de un array de MAX (una constante definida) números enteros a, en el que al menos hay dos números que son distintos (es decir, no son todos iguales). Obtenga una función que tomando como parámetro dicho array, devuelva un elemento del array que sea mayor que el mínimo de éste. Diseña la función principal (main) para probar el funcionamiento de la función. Para ello, se leerá de teclado una colección de MAX números enteros con los que se llenará el array, se invocará a la función implementada y se mostrará por pantalla el valor devuelto por la misma. Un ejemplo de ejecución sería (MAX = 10):

```
Introduzca 10 numeros enteros: 3 3 3 3 5 27 1 9 21 32
Un elemento Mayor que el Minimo es: 5
```

4. Diseña un procedimiento que permita invertir el contenido de un array de MAX (una constante definida) números enteros recibido como parámetro. No se podrán utilizar arrays auxiliares. Diseña la función principal (main) para probar el funcionamiento del procedimiento. Para ello, se leerá de teclado una colección de MAX números enteros con los que se llenará el array, se invocará al procedimiento implementado y se mostrará por pantalla el contenido del array modificado.

Ejemplo:

```
Array Original: 24 12 45 90 7 9 15.
Array Invertido: 15 9 7 90 45 12 24.
```

5. Escriba un programa que efectúe la conversión de un número natural en base 10 a otra determinada base, sabiendo que el resultado no sobrepasará los 50 dígitos. El usuario introducirá primero el número en base 10 y después la base a la que convertirlo (el programa debe asegurarse de que la base no sea ni menor de 2 ni mayor de 9)

Nota: Recordemos que para obtener la representación en una base b de un número en decimal, dividimos entre b primero el número y después sucesivamente los diferentes cocientes que se vayan obteniendo hasta que el cociente sea cero. Los diferentes restos obtenidos en esas sucesivas divisiones constituyen la representación en dicha base b (pero en orden inverso a como se han ido calculando). Por ejemplo, para el número decimal 26 en base 2 es 11010.

$$\begin{array}{r} 26 \mid 2 \\ 0 \ 13 \mid 2 \\ 1 \ 6 \mid 2 \\ 0 \ 3 \mid 2 \\ 1 \ 1 \mid 2 \\ 1 \ 0 \end{array}$$

6. Diseña un algoritmo que lea un texto de longitud indefinida formado por letras mayúsculas (que termina con un punto) y muestre por pantalla la frecuencia con la que aparece cada una de las letras del texto. Un ejemplo de ejecución sería:

```
Introduzca una secuencia de letras mayusculas (punto para terminar):
BAACABDPBHBH.
```

La frecuencia de cada letra es:

```
A: 3
B: 4
C: 1
D: 1
H: 2
P: 1
```

7. Suponiendo que los caracteres con los que trabajamos siempre serán letras mayúsculas y dados los siguientes tipos (para una constante MAX cualquiera):

```
typedef array<char, MAX> Componentes;
struct Vector {
    Componentes datos; // array de caracteres
    int tam;           // numero de celdas ocupadas
};
```

- a) La moda de un array de caracteres es el carácter del array que se repite más frecuentemente. Si varios caracteres se repiten con la misma frecuencia máxima, entonces no hay moda. Escribe un procedimiento con tres parámetros. El primero es de entrada para recibir un registro de tipo `Vector` que contiene el array `datos` con `tam` caracteres. El segundo parámetro es de salida e indicará si se ha encontrado la moda en el array o no. El tercer parámetro es de salida y será el carácter que representa la moda (si es que existe). Diseña la función principal (`main`) para probar el funcionamiento del procedimiento. Para llenar el registro, se leerá de teclado una secuencia de letras mayúsculas hasta leer un salto de línea (carácter '`\n`', que actúa como terminador de la secuencia). Si el array `datos` se llena antes de leer el carácter terminador, los caracteres restantes de la entrada (incluido el terminador) se descartarán (leyéndolos y no haciendo nada con ellos). Para la lectura de cada carácter utiliza la instrucción `cin.get()`. Despues se invocará al procedimiento implementado y se mostrará por pantalla el carácter moda en caso de existir o una indicación de que no hay moda. Dos ejemplos de ejecución serían (`MAX = 20`):

```
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): ABACDBAD
```

La moda es: A

```
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): ABACDBD
```

NO hay moda

- b) Diseña una función booleana que, dados dos registros de tipo `Vector` como parámetros, devuelva TRUE si son iguales y FALSE en otro caso. Dos registros son iguales si sus arrays contienen los mismos elementos y en el mismo orden relativo, suponiendo que el primer elemento sigue al último. Por ejemplo, si los arrays de los registros fueran:

```
['A','C','D','F','E']
['D','F','E','A','C']
```

la función devolvería TRUE.

Supón, además, que cada carácter aparece a lo sumo una vez.

Diseña la función principal (`main`) para probar el funcionamiento de la función. Para ello, se leerán de teclado dos secuencias de letras mayúsculas (el carácter '`\n`' actuará como terminador de cada una de ellas) con las que se llenarán dos registros. Para cada secuencia, si el array `datos` del registro correspondiente se llena antes de leer el carácter terminador, los caracteres restantes de la entrada (incluido el terminador) se descartarán (leyéndolos y no haciendo nada con ellos). Para la lectura de cada carácter utiliza la instrucción `cin.get()`. Después se invocará a la función implementada y se mostrará por pantalla una indicación de si los registros son o no iguales. Dos ejemplos de ejecución serían (`MAX = 20`):

```
Introduzca una secuencia de letras mayusculas(punto para terminar y como maximo 20 letras): ACDFE  
Introduzca una secuencia de letras mayusculas(punto para terminar y como maximo 20 letras): DFEAC
```

SI son iguales

```
Introduzca una secuencia de letras mayusculas(punto para terminar y como maximo 20 letras): ACDFE  
Introduzca una secuencia de letras mayusculas(punto para terminar y como maximo 20 letras): DEFAC
```

NO son iguales

- c) Diseña un procedimiento que tome como parámetros de entrada dos registros con los arrays ordenados y devuelva (con un parámetro de salida) el registro resultado de realizar la mezcla ordenada de los dos arrays contenidos en los vectores de entrada. Puede ocurrir que no quepan todos los caracteres en el registro resultado, perdiéndose los mismos. Diseña la función principal (`main`) para probar el funcionamiento del procedimiento. Para ello, se leerán de teclado dos secuencias de letras mayúsculas (el carácter '`\n`' actuará como terminador de cada una de ellas) con las que se llenarán dos registros. Para cada secuencia, si el array `datos` del registro correspondiente se llena antes de leer el carácter terminador, los caracteres restantes de la entrada (incluido el terminador) se descartarán (leyéndolos y no haciendo nada con ellos). Para la lectura de cada carácter utiliza la instrucción `cin.get()`. Después se invocará al procedimiento implementado y se mostrará por pantalla el resultado de la mezcla ordenada. Dos ejemplos de ejecución serían (`MAX = 20`):

```
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): ACHHQ  
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): BCPR
```

La mezcla ordenada es:
ABCCHHPQRS

```
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): ABBBCHPQSTVQ  
Introduzca una secuencia de letras mayúsculas (punto para terminar y como maximo 20 letras): CDFFGPRSST
```

La mezcla ordenada es:
ABBBCCDFFGHPPQRSSTT

8. Los alumnos de informática desean celebrar una comida un día del presente mes en el que puedan acudir todos. Se pide realizar un algoritmo que recoja de cada alumno los días que le vendría bien ir a la comida, e imprima los días concordantes para todos los alumnos o una indicación de que no hay. Los datos se introducirán por teclado. Primero se introducirá el número de alumnos que intervienen. Después, por cada alumno se introducirá una única línea con los números de los días libres separados por espacios (un 0 para terminar). Dos ejemplos de ejecución:

```

Numero de alumnos a introducir: 3
Introduzca los dias preferidos por el alumno 1 (introduzca un 0 para
terminar): 3 5 7 15 20 0
Introduzca los dias preferidos por el alumno 2 (introduzca un 0 para
terminar): 4 6 15 18 20 0
Introduzca los dias preferidos por el alumno 3 (introduzca un 0 para
terminar): 15 20 25 0

```

Los dias comunes son: 15 20

```
Numero de alumnos a introducir: 2
```

```

Introduzca los dias preferidos por el alumno 1 (introduzca un 0 para
terminar): 2 4 7 0
Introduzca los dias preferidos por el alumno 2 (introduzca un 0 para
terminar): 3 8 15 20 0

```

Los dias comunes son: No hay ningun dia comun

9. Diseña un procedimiento que recibe como parámetro de entrada un array de valores enteros val y devuelve como parámetro de salida otro array de enteros ind, de forma que el contenido de cada una de sus celdas es un índice del array val. Los índices estarán almacenados de forma que, si recorremos el array ind de izquierda a derecha, y visitamos las celdas de val, cuyo índice nos vamos encontrando en ind, los valores de val se recorrerán en orden creciente de menor a mayor. El array val no se podrá modificar, ni se puede hacer una copia del mismo. Por ejemplo:

0	1	2	3	4	0	1	2	3	4
10	5	-7	0	12	2	3	1	0	4
val					ind				

Diseña la función principal (main) para probar el funcionamiento del procedimiento. Para ello, se leerá de teclado una colección de MAX (una constante definida) números enteros con los que se llenará el array val, se invocará al procedimiento implementado y se mostrará por pantalla el contenido del array ind.

10. Vamos a trabajar con listas de números enteros de hasta un tamaño máximo de MAX (una constante cualquiera). Define el tipo de datos TLista para ello y diseña un procedimiento criba que recibe como parámetros de entrada una lista de números enteros listal de tipo TLista y un número natural x. El procedimiento devolverá como parámetro de salida otra lista lista2 de tipo TLista que contendrá sólo aquellos números de listal que están repetidos x veces. En la lista lista2 no habrá elementos repetidos. Diseña la función principal (main) para probar el funcionamiento del procedimiento. Para leer la lista de números listal, se le pedirá primero al usuario el número de valores que va a introducir, controlando que éste sea mayor que 0 y menor o igual que MAX. Ejemplo de ejecución (MAX = 10):

```

Cuantos numeros desea introducir (maximo 10): 9
Introduzca 9 numeros: 1 3 4 3 1 3 0 -6 4
Introduzca el numero de repeticiones para realizar la criba: 2

```

La lista cribada es: 1 4

Que no te escriban poemas de amor cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

11. Consideremos un vector V que contiene N valores naturales (array de naturales de tamaño N, siendo N una constante cualquiera). Definimos el *centro* c del vector V como el índice entre 1 y N-2 que verifica la siguiente propiedad:

$$\sum_{i=0}^{c-1} (c-i) * V[i] = \sum_{j=c+1}^{n-1} (j-c) * V[j]$$

Esta propiedad no siempre se verifica; en ese caso, decimos que el vector no tiene centro.

Diseña un procedimiento `centroVector` que reciba como parámetro un vector V y nos devuelva dos valores como parámetros: el primero indica si existe o no el centro del vector, y el segundo, indica el índice en el que se encuentra el centro en caso de existir. Diseña la función principal (`main`) para probar el funcionamiento del procedimiento.

A continuación, se detallan dos ejemplos (suponemos que N = 5):

1º Ejemplo:

El contenido del vector es: **6 2 3 0 1**
El centro de este vector es el índice **1** (casilla donde está el **2**) ya que al calcular los sumatorios:
- Sumatorio izquierda: $(1-0)*V[0] = 1*6 = 6$
- Sumatorio derecha: $(2-1)*V[2]+(3-1)*V[3]+(4-1)*V[4] = 1*3+2*0+3*1 = 6$

2º Ejemplo:

El contenido del vector es: **1 2 1 1 0**
Este vector no tiene centro

12. Definimos el siguiente tipo `Vector` para almacenar números enteros, hasta un máximo de `MAX` (una constante cualquiera):

```
typedef array<int, MAX> TNumeros;
struct Vector {
    TNumeros numeros;      // array de enteros
    int tam;                // numero de celdas ocupadas (contiguas)
};
```

Diseña un procedimiento `borrar`, que recibe como parámetros un registro de tipo `Vector` y un número entero, y elimina dicho número del array del registro (si hay varias ocurrencias del número, se elimina sólo una de ellas; si el número no está, no se hace nada). Diseña otro procedimiento `insertar`, que recibe como parámetros un registro de tipo `Vector` y un número entero, y añade dicho número al array del registro (si el array está lleno no se hace nada). Haz dos versiones de cada procedimiento, una para cuando los elementos del array del registro no están ordenados y otra para cuando sí lo están. Diseña la función principal (`main`) para probar el funcionamiento de los procedimientos. Para llenar el registro, se leerá de teclado una secuencia de números enteros hasta leer el 0 (que actúa como terminador de la secuencia). Si el array `numeros` se llena antes de leer el terminador, los números restantes de la entrada (incluido el terminador) se descartarán (leyéndolos y no haciendo nada con ellos). Después se leerá un valor a borrar de la secuencia introducida anteriormente y se invocará al procedimiento `borrar`, mostrando el contenido del registro posteriormente. Finalmente se leerá un valor a insertar en el array del registro y se mostrará el contenido del mismo tras realizar la invocación al procedimiento `insertar`. Dos ejemplos de ejecución serían los siguientes. El primero para cuando los elementos no están ordenado y el segundo cuando sí lo están. En ambos casos la constante `MAX` es 10:

```
Introduzca una secuencia de numeros enteros (0 para terminar y como maximo  
10 numeros): 2 -4 32 45 6 7 0  
Introduzca un numero entero a borrar del vector: -4  
El vector despues de borrar es: 2 7 32 45 6  
Introduzca un numero entero a insertar en el vector: 8  
El vector despues de insertar es: 2 7 32 45 6 8
```

```
Introduzca una secuencia de numeros enteros (0 para terminar y como maximo  
10 numeros): 3 5 6 14 23 0  
Introduzca un numero entero a borrar del vector: 6  
El vector despues de borrar es: 3 5 14 23  
Introduzca un numero entero a insertar en el vector: 8  
El vector despues de insertar es: 3 5 8 14 23
```

13. La distancia entre dos letras en un texto es el número de letras que aparecen en el texto entre las dos letras indicadas. Diseñe un algoritmo que lea un texto de longitud indefinida formado por letras mayúsculas (que termina con un punto) y muestre por pantalla la máxima distancia entre cada par de letras repetidas. Aquellas letras que no se repitan no aparecerán en la salida.

Por ejemplo:

Texto de entrada: ABEADDGLAKE .

Salida:

```
Distancia entre A: 4  
Distancia entre D: 0  
Distancia entre E: 7
```

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

**Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►**



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

WUOLAH

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito

WUOLAH

si lees esto me debes un besito