



UNIVERSIDAD DE MALAGA
DPTO. DE LENGUAJES Y C. DE LA COMPUTACION
E.T.S. DE INGENIERIA INFORMATICA

FUNDAMENTOS DE LA PROGRAMACIÓN

TERCERA RELACIÓN DE PROBLEMAS

1. Dadas las siguientes declaraciones:

Constantes

```
const unsigned TAMLISTA = 50000;  
const unsigned TAMSUB = 20;
```

Tipos

```
typedef string TLista[TAMLISTA];  
typedef string TSublista[TAMSUB];  
enum TElemento {Hardware, Software};  
struct TregistroLista{  
    TLista Lista;  
    TSublista Sublista;  
};  
struct TRegCatalogo {  
    string nombre_elem;  
    TElemento elemento;  
    TregistroLista subcatalogo;  
};  
typedef TRegCatalogo TCatalogo[TAMSUB];
```

Variables

```
TCatalogo catalogo;  
TRegCatalogo un_catalogo;  
TregistroLista una_lista;  
int cont;  
string un_nombre;
```

¿ Cuales de las siguientes sentencias son válidas? (Asume que las variables válidas tienen ya valores definidos)

- a) if (TRegCatalogo.elemento == Hardware) {
 cont++;
}
- b) catalogo[1].subcatalogo.Lista[2] = un_catalogo;
- c) catalogo[5].TregistroLista = una_lista;
- d) un_catalogo.nombre_elem[2] = una_lista.Lista[2];
- e) catalogo[1].Lista.Lista[1][2] = un_nombre[2];
- f) catalogo[19].Lista[1] = un_nombre;
- g) if (catalogo[19].elemento == Software) {
 catalogo[10].TregistroLista.Sublista[3][9] = 'A';
}
- h) un_catalogo = catalogo[5];
- i) un_catalogo.subcatalogo = una_lista;

2. Dadas las siguientes declaraciones:

Constantes

```
const unsigned MAX = 10;
```

Tipos

```
typedef unsigned TArray[MAX];
```

Variables

```
TArray a;  
unsigned x, k;
```

donde el array *a* tiene los valores: 6, 3, 9, 7, 1, 8, 10, 2, 4 y 5.

¿Cuánto vale *x* cuando se ejecuta cada uno de los siguientes segmentos de código?

<pre>x = 0; for (unsigned i = 0; i < MAX; i++) { x += a[i]; }</pre>	<pre>x = a[0]; for (unsigned i = 1; i < MAX; i++) { if (x < a[i]) { x = a[i]; } }</pre>
<pre>k = 0; for (unsigned i = 1; i < MAX; i++) { if (a[k] < a[i]) { k = i; } } x = a[k];</pre>	

3. Diseña una función que recibe como parámetros de entrada un array de MAX (una constante definida) números enteros *a* y un número entero *num*, y devuelve *true* si el número *num* está contenido en *a*, y *false* en otro caso.
4. Se dispone de un array de 80 números enteros en el que al menos hay dos números que son iguales y dos que son distintos. Obtenga una función que tomando como parámetro dicho array, devuelva un elemento del array que sea mayor que el mínimo de éste.
5. Diseña un procedimiento que permita invertir el contenido de un array recibido como parámetro. No se podrán utilizar arrays auxiliares. Impleméntalo de forma iterativa y de forma recursiva.
Ejemplo:
Array Original: 24 12 45 90 7 9 15.
Array Invertido: 15 9 7 90 45 12 24.
6. 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.

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

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

```
typedef char Componentes[MAX];
struct Vector {
    Componentes datos;    // array de caracteres
    unsigned 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).

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 vectores son iguales si contienen los mismos elementos y en el mismo orden relativo, suponiendo que el primer elemento sigue al último. Por ejemplo, si los arrays 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.

c) Diseña un procedimiento que tome como parámetros de entrada dos vectores con los arrays ordenados y devuelva (con un parámetro de salida) el vector resultado de realizar la mezcla ordenada de los dos arrays contenidos en los vectores de entrada.

9. 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 las fechas concordantes para todos los alumnos. 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).

10. 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				

11. 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 `lista1` 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 `lista1` que están repetidos `x` veces. En la lista `lista2` no habrá elementos repetidos.

Ejemplo:

El contenido de `lista1` es: 1 3 4 3 1 3 0 -6 4
 El valor de `x` es: 2
 El contenido de `lista2` será: 1 4

12. 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.

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

1^{er} 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

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

```
typedef int TNumeros[MAX];
struct Vector {
    TNumeros numeros;    // array de enteros
    unsigned tam;        // numero de celdas ocupadas
};
```

Diseña un procedimiento `borrar`, que recibe como parámetros un vector y un número entero, y elimina dicho número del vector (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 vector y un número entero, y añade dicho número al vector (si el vector está lleno no se hace nada). Haz dos versiones de cada procedimiento. Una para cuando los elementos del vector no están ordenados y otra para cuando sí lo están.

14. La distancia entre dos letras en un texto es el número de letras que aparecen en el texto entre las dos letras indicadas. Diseña 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

15. Diseña una función que recibe como parámetros de entrada una matriz de números enteros `m` y un número entero `num`, y devuelve `true` si el número `num` está contenido en `m`, y `false` en otro caso.
16. Diseña una función para calcular la suma de los elementos de la diagonal principal de una matriz cuadrada.
17. Una matriz tiene un punto silla en una de sus componentes, si ese componente es el mayor estricto de su columna y el menor estricto de su fila. Diseña un algoritmo que recogiendo de teclado los componentes de una matriz cuadrada de enteros de hasta un máximo de 10×10 , muestre en la pantalla las coordenadas de su punto silla si lo tiene o una indicación de que no lo tiene.

18. Un tablero `n`-goro es un tablero con $n \times (n+1)$ casillas de la forma:

	1	2	...	n	n+1
1			...		
2			...		
...
n			...		

Una propiedad interesante es que se pueden visitar todas sus casillas haciendo el siguiente recorrido por diagonales. Empezamos por la casilla (1,1) y recorremos la diagonal principal hacia la derecha y hacia abajo hasta llegar a la casilla (n,n). La siguiente casilla a visitar sería la (n+1,n+1) que no existe porque nos saldríamos del tablero por abajo. En estos casos siempre se pasa a la casilla equivalente en la primera fila, es decir, la (1,n+1). Ahora seguimos moviéndonos hacia la derecha y hacia abajo. Pero la siguiente casilla sería la (2,n+2) que no existe porque nos hemos salido por la derecha. En estos casos se continúa por la casilla equivalente de la primera columna, es decir, la (2,1). De nuevo nos movemos hacia la derecha y hacia abajo, hasta alcanzar la casilla (n,n-1). La siguiente casilla sería la (n+1,n), pero como nos saldríamos por abajo pasamos a la casilla equivalente de la primera fila (1,n). Si se continúa con este proceso se termina visitando todas las casillas del tablero.

Diseñe un procedimiento que dada una constante N devuelva como parámetro de salida un tablero N-goro con sus casillas rellenas con el número correspondiente al momento en que se visitan. Por ejemplo, si N es 4, el tablero a devolver sería:

	1	2	3	4	5
1	1	17	13	9	5
2	6	2	18	14	10
3	11	7	3	19	15
4	16	12	8	4	20

19. Se desea crear una aplicación que permita asignar cargos electos a diversos partidos en unas elecciones, considerando el sistema conocido como ley de D'Hondt. El sistema de D'Hondt es una fórmula electoral, creada por Victor d'Hondt, que permite obtener el número de cargos electos asignados a las candidaturas, en proporción a los votos conseguidos. El procedimiento consiste en, tras escutar todos los votos, calcular una serie de divisores para cada partido (o lista) político. La fórmula de los divisores es V/N , donde V representa el número total de votos recibidos por el partido, y N representa cada uno de los números enteros de 1 hasta el número de cargos electos de la circunscripción objeto de escrutinio. Una vez realizadas las divisiones de los votos de cada candidatura por cada uno de los divisores desde 1 hasta N , la asignación de cargos electos se hace ordenando los cocientes de las divisiones de mayor a menor y asignando a cada uno un escaño hasta que éstos se agoten. Por ejemplo, para una distribución de votos como la siguiente:

	Partido A	Partido B	Partido C	Partido D	Partido E
Votos	340.000	280.000	160.000	60.000	15.000

Y un número de cargos electos a distribuir de 7, los divisores para cada partido político serían los siguientes:

	1	2	3	4	5	6	7
A	$V_A/1=340.000$	$V_A/2=170.000$	$V_A/3=113.333$	$V_A/4=85.000$	$V_A/5=68.000$	56.667	48.571
B	$V_B/1=280.000$	$V_B/2=140.000$	$V_B/3=93.333$	$V_B/4=70.000$	$V_B/5=56.000$	46.667	40.000
C	$V_C/1=160.000$	$V_C/2=80.000$	$V_C/3=53.333$	$V_C/4=40.000$	$V_C/5=32.000$	26.667	22.857
D	$V_D/1=60.000$	$V_D/2=30.000$	$V_D/3=20.000$	$V_D/4=15.000$	$V_D/5=12.000$	10.000	8.571

E	$V_E/1=15.000$	$V_E/2=7.500$	$V_E/3=5.000$	$V_E/4=3.750$	$V_E/5=3.000$	2.500	2.143
---	----------------	---------------	---------------	---------------	---------------	-------	-------

Las celdas sombreadas se corresponden con los 7 cocientes más grandes, y por lo tanto el resultado sería el siguiente:

	Partido A	Partido B	Partido C
Cargos	3	3	1

- Implementa un programa que tenga una entrada de datos como la siguiente:

```

Introduzca el Número de Cargos: 7
Introduzca el Número de Partidos: 5
Introduzca el Nombre y Número de Votos por Partido:
Partido 1: PP 340.000
Partido 2: PSOE 280.000
Partido 3: IU 160.000
Partido 4: UPyD 60.000
Partido 5: VERDES 15.0000

```

Y que aplicando la Ley D'Hondt muestre un resultado como el siguiente:

```

Los Cargos Electos son:
PP 3
PSOE 3
IU 1

```

No olvides aplicar Diseño Descendente a la hora de diseñar el programa. Podemos suponer que el máximo número de Cargos será 15 y el de Partidos 10.

20. Un importante vulcanólogo, con objeto de estudiar las consecuencias destructoras de un volcán en una determinada área, pretende desarrollar un sistema informático que le ayude en su tarea.

Para ello, nos pide que realicemos un procedimiento con la siguiente cabecera:

```
void flujoDeLava(const Superficie& sup, int fil, int col, Lava& lava)
```

El procedimiento recibe una matriz (sup) que representa el plano de la zona a estudiar, donde cada elemento contiene un número que representa la altura de ese punto respecto al nivel del mar. Así mismo, recibe un punto (fil y col) de dicho plano donde surge el cráter de un volcán. El procedimiento predecirá el recorrido que realiza la lava, y lo representará en la matriz de salida (lava), donde el asterisco (*) representa que la lava ha pasado por ese punto, y el espacio en blanco (' ') representa que la lava no ha pasado por dicho punto.

El flujo de lava se desplaza según el siguiente esquema a partir del cráter:

- Desde un determinado punto, siempre se mueve hacia los puntos circundantes que se encuentran a menor altura (los puntos circundantes de uno dado serán el superior, inferior, izquierdo y derecho) (la estructura no se considera circular).
- Así sucesivamente se repite el proceso para todos los puntos donde haya alcanzado el flujo de lava.

Ejemplo: Cráter del Volcán: fil = 2, col = 2

	sup						lava				
	0	1	2	3	4		0	1	2	3	4
0	4	7	8	7	6		*				
1	5	6	9	9	7		*	*			
2	7	7	8	8	7			*	*		
3	5	6	8	6	8		*	*			
4	4	6	6	9	5		*				

Pruebe dicho procedimiento usando el siguiente algoritmo:

```
#include <iostream>
using namespace std;

const int FILAS = 5;
const int COLUMNAS = 5;

typedef int Superficie[FILAS][COLUMNAS];
typedef char Lava[FILAS][COLUMNAS];

int main() {
    Superficie sup;
    Lava lava;
    int fil,col;

    cout << "Introduzca superficie (matriz de naturales " << FILAS
    << "x" << COLUMNAS << "):\n";
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            cin >> sup[i][j];
        }
    }
    cout << "Introduzca punto de crater (fila y columna):\n";
    cin >> fil >> col;

    flujoDeLava(sup,fil,col,lava);

    cout << "El recorrido de la lava es:\n";
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            cout << lava[i][j];
        }
        cout << endl;
    }
    return 0;
}
```

21. Una farmacia desea almacenar sus productos (TProducto) en una estructura. De cada producto hay que almacenar la siguiente información: código (unsigned) , nombre (string), precio (float), fecha de caducidad (definir un tipo registro para la fecha). Diseña la estructura de datos (TFarmacia) para almacenar hasta un máximo de MAX (una constante) productos y realiza los siguientes subalgoritmos:

- void LeerProducto(TProducto& p)
- void EscribirProducto(const TProducto& p)
- void InicializarFarmacia(TFarmacia& f)
- void InsertarProducto(TFarmacia& f, const TProducto& p)
- void BorrarProducto(TFarmacia& f, unsigned codigo)

- `void BuscarProductoCodigo(const TFarmacia& f, unsigned codigo, bool& encontrado, TProducto& p)`
- `void BuscarProductoNombre(const TFarmacia& f, string nombre, bool& encontrado, TProducto& p)`
- `void ListarFarmacia(const TFarmacia& f)`

22. Diseñe un algoritmo lea de teclado un texto y muestre un listado por pantalla de todas las palabras del texto que comiencen por ciertas iniciales. Dichas iniciales serán las letras que formen la primera palabra del texto.

NOTAS :

- El texto contiene un número indefinido de palabras.
- El texto termina con la palabra FIN.
- Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos mayúsculas).
- El carácter separador de palabras es el espacio en blanco.

23. Diseñe un algoritmo que lea de teclado un patrón (una cadena de caracteres) y un texto, y dé como resultado las palabras del texto que contengan a dicho patrón. En la salida no habrá palabras repetidas.

Ejemplo:

Entrada:

Patrón : RE

Texto: CREO QUE IREMOS A LA DIRECCION QUE NOS DIERON AUNQUE PIENSO
QUE DICHA DIRECCION NO ES CORRECTA FIN

Salida:

CREO IREMOS DIRECCION CORRECTA

NOTAS :

- El texto contiene un número indefinido de palabras.
- El texto termina con la palabra FIN.
- Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos mayúsculas).
- El carácter separador de palabras es el espacio en blanco.
- En el texto aparecerán un número máximo MAX_PAL_DIST (una constante) de palabras distintas.

24. Diseñe un algoritmo que lea de teclado un texto, y dé como resultado las palabras del texto junto con las posiciones en las que aparece dicha palabra dentro del texto. En la salida no habrá palabras repetidas.

Ejemplo:

Entrada:

Introduzca texto (FIN para terminar):

CREO QUE IREMOS A MI CASA PRIMERO Y QUE DESPUES IREMOS A LA
CASA QUE QUIERAS FIN

Salida:

CREO 1

QUE 2 9 15
IREMOS 3 11
A 4 12
MI 5
CASA 6 14
PRIMERO 7
Y 8
DESPUES 10
LA 13
QUIERAS 16

NOTAS :

- El texto contiene un número indefinido de palabras.
- El texto termina con la palabra FIN.
- Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos mayúsculas).
- El carácter separador de palabras es el espacio en blanco.
- En el texto aparecerán un número máximo MAX_PAL_DIST (una constante conocida) de palabras distintas.
- Cada palabra en el texto aparecerá repetida un número máximo MAX_REP (una constante conocida) de veces.