

FP_Solucion_relac3.pdf



RubenEU



Fundamentos de la Programación



1º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga



**El más PRO del lugar
puedes ser Tú.**

¿Quieres eliminar toda la publi
de tus apuntes?



¡Hazte PRO!

4,95€ / mes



WUOLAH



El más PRO del lugar puedes ser Tú.



¿Quieres eliminar toda la publi de tus apuntes?



¡Fuera Publi!
Concéntrate al máximo



Apuntes a full.
Sin publi y sin gastar coins

Para los amantes de la inmediatez, para los que no desperdician ni un solo segundo de su tiempo o para los que dejan todo para el último día.

Quiero ser PRO

4,95 / mes



UNIVERSIDAD DE MÁLAGA
DPTO. DE LENGUAJES Y C. DE LA COMPUTACIÓN
E.T.S. DE INGENIERÍA INFORMATICA

FUNDAMENTOS DE LA PROGRAMACIÓN

SOLUCIÓN

A

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 números enteros *a* y un número entero *num*, y devuelve *true* si el número *num* está contenido en *a*, y *false* en otro caso.

SOLUCIÓN:

VERSIÓN 1:

```

#include <iostream>

using namespace std;

const unsigned MAX = 10;
typedef int TArray[MAX];

bool esta(int num, const TArray& a) {
    bool encontrado = false;
    unsigned cont = 0;

    while ((cont < MAX) && !encontrado) {
        if (a[cont] == num) {
            encontrado = true;
        } else {

```

```

        cont++;
    }

    return encontrado;
}

int main()
{
    TArray a;
    int num;

    cout << "Introduzca " << MAX << " números enteros: ";
    for (unsigned i = 0; i < MAX; i++) {
        cin >> a[i];
    }
    cout << "Introduzca el numero a buscar: ";
    cin >> num;

    if (esta(num,a)) {
        cout << "El numero " << num << " SI esta en la colección" << endl;
    } else {
        cout << "El numero " << num << " NO esta en la colección" << endl;
    }

    return 0;
}

```

VERSIÓN 2:

```

#include <iostream>

using namespace std;

typedef int TArray[MAX];

bool esta(int num, const TArray& a) {
    unsigned cont = 0;

    while ((cont < MAX) && (a[cont] != num)) {
        cont++;
    }

    return cont < MAX;
}

int main()
{
    TArray a;
    int num;

    cout << "Introduzca " << MAX << " números enteros: ";
    for (unsigned i = 0; i < MAX; i++) {
        cin >> a[i];
    }
    cout << "Introduzca el numero a buscar: ";
    cin >> num;
}

```

```

    if (esta(num,a)) {
        cout << "El numero " << num << " SI esta en la colección" << endl;
    } else {
        cout << "El numero " << num << " NO esta en la colección" << endl;
    }

    return 0;
}

```

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.

SOLUCIÓN:

```

#include <iostream>

using namespace std;

const unsigned MAX = 10; // en lugar de 80 (para probar)

typedef int TArray[MAX];

// el array a contiene al menos dos elementos distintos
int mayorMinimo(const TArray& a) {
    unsigned i = 0;
    int res;

    while ((i < MAX-1) && (a[i] == a[i+1])) {
        i++;
    }

    if (i >= MAX-1) { // robusto
        throw "el array no tiene al menos dos números distintos";
    } else if (a[i] > a[i+1]) {
        res = a[i];
    } else {
        res = a[i+1];
    }

    return res;
}

int main()
{
    TArray a;

    cout << "Introduzca " << MAX << " números enteros: ";
    for (unsigned i = 0; i < MAX; i++) {
        cin >> a[i];
    }
    cout << "Un elemento Mayor que el Mínimo es: "
        << mayorMinimo(a) << endl;

    return 0;
}

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



5. Diseña un algoritmo que permita invertir el contenido de un array. El algoritmo no podrá utilizar arrays auxiliares. Impleméntalo de forma iterativa y de forma recursiva.

Array Original: 24 12 45 90 7 9 15.

Array Invertido: 15 9 7 90 45 12 24.

SOLUCIÓN:

```
#include <iostream>
using namespace std;
const unsigned MAX = 10;
typedef int TArray[MAX];

void intercambiar(int& x, int& y) {
    int temp;

    temp = x;
    x = y;
    y = temp;
}

void invertirIter(TArray& a) {
    for (unsigned i = 0; i <= (MAX-1) / 2; i++) {
        intercambiar(a[i], a[MAX-1-i]);
    }
}

void invertirRecur(TArray& a, unsigned i) {
    if (i <= (MAX-1) / 2) {
        intercambiar(a[i], a[MAX-1-i]);
        invertirRecur(a, i+1);
    }
}

int main()
{
    TArray a;

    cout << "Introduzca " << MAX << " números enteros: ";
    for (unsigned i = 0; i < MAX; i++) {
        cin >> a[i];
    }
    //invertirIter(a);
    invertirRecur(a, 0);
    cout << "Esos números en orden inverso son: ";
    for (unsigned i = 0; i < MAX; i++) {
        cout << a[i] << " ";
    }
}

return 0;
}
```

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.

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

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned MAX = 50;
typedef unsigned TArray[MAX];
struct TResultado {
    TArray digitos; // sucesivos restos
    unsigned nDig; // numero de digitos (restos)
};

void convertir(unsigned numero, unsigned base, TResultado& resultado)
{
    unsigned i;

    i = 0;
    do {
        resultado.digitos[i] = numero % base;
        numero = numero / base;
        i++;
    } while (numero != 0);
    resultado.nDig = i;
}

int main()
{
    unsigned numero, base;
    TResultado resultado;

    cout << "Introduzca un número natural en base 10: ";
    cin >> numero;
    do {
        cout << "Introduzca la base a la que convertirlo (2<=base<=9): ";
        cin >> base;
    } while ((base < 2) || (base > 9));

    convertir(numero, base, resultado);

    cout << "Ese numero en base " << base << " es: ";
}
```

```

        for (int i = resultado.nDig - 1; i >= 0; i--) {
            cout << resultado.digitos[i];
        }
        cout << endl;

    return 0;
}

```

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.

SOLUCIÓN:

```

#include <iostream>

using namespace std;

const unsigned RANGO_MAY = 26; // las 26 letras mayúsculas
typedef unsigned TFrec[RANGO_MAY];

void inicializar(TFrec& freq) {
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        freq[i] = 0;
    }
}

void calcular(TFrec& freq) {
    char c;

    inicializar(freq);

    cout << "Introduzca una secuencia de letras mayúsculas (punto para
terminar): ";
    cin >> c;
    while (c != '.') {
        if (('A' <= c) && (c <= 'Z')) { // robusto
            freq[c - 'A']++;
        }
        cin >> c;
    }
}

void imprimir(const TFrec& freq) {

    cout << "La frecuencia de cada letra es:" << endl;
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        if (freq[i] != 0) {
            cout << char(i + 'A') << ":" << freq[i] << endl;
        }
    }
}

int main()
{
    TFrec freq;

    calcular(freq);
    imprimir(freq);
}

```

```

    return 0;
}

```

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).

SOLUCIÓN

```

#include <iostream>

using namespace std;

const unsigned RANGO_MAY = 26; // las 26 letras mayúsculas
typedef unsigned TFrec[RANGO_MAY];

const unsigned MAX = 10;
typedef char Componentes[MAX];
struct Vector {
    Componentes datos;
    unsigned tam;
};

void inicializar(TFrec& freq);
void calcular(TFrec& freq, const Vector& v);
void encontrarModa(const TFrec& freq, bool& hayModa, char& m);
void moda(const Vector& v, bool& hayModa, char& m);
void leer(Vector& v);

int main()
{
    Vector v;
    bool hayModa;
    char m;

    leer(v);

    moda(v, hayModa, m);

    if (hayModa) {
        cout << "La moda es: " << m << endl;
    } else {
        cout << "NO hay moda\n";
    }
}

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
        return 0;
    }

void inicializar(TFrec& freq) {
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        freq[i] = 0;
    }
}

void calcular(TFrec& freq, const Vector& v) {
    inicializar(freq);
    for (unsigned i = 0; i < v.tam; i++) {
        freq[v.datos[i] - 'A']++;
    }
}

void encontrarModa(const TFrec& freq, bool& hayModa, char& m) {

    unsigned mayor;

    mayor = freq[0]; // frecuencia de la 'A'
    hayModa = true;
    m = 'A';

    for (unsigned i = 1; i < RANGO_MAY; i++) {
        if (freq[i] == mayor) {
            hayModa = false;
        } else if (freq[i] > mayor) {
            mayor = freq[i];
            m = char(i + 'A');
            hayModa = true;
        }
    }
}

void moda(const Vector& v, bool& hayModa, char& m) {

    TFrec freq;

    calcular(freq, v);
    encontrarModa(freq, hayModa, m);
}

void leer(Vector& v) {
    char c;

    cout << "Introduzca una secuencia de letras mayúsculas";
    cout << "(punto para terminar y como máximo " << MAX
        << " letras):\n";

    v.tam = 0;
    cin.get(c);
    while ((c != '.') && (v.tam < MAX)) {
        v.datos[v.tam] = c;
        v.tam++;
        cin.get(c);
    }
}
```



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.

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned MAX = 10;
typedef char Componentes[MAX];
struct Vector {
    Componentes datos;
    unsigned tam;
};

void leer(Vector& v) {
    char c;

    cout << "Introduzca una secuencia de letras mayúsculas";
    cout << "(punto para terminar y como máximo "
        << MAX
        << " letras):\n";

    v.tam = 0;
    cin.get(c);
    while ((c != '.') && (v.tam < MAX)) {
        v.datos[v.tam] = c;
        v.tam++;
        cin.get(c);
    }
}

void incrementoCircular(unsigned& ind, unsigned max) {
    ind = (ind + 1) % max;
/*
    if (ind == max) {
        ind = 0;
    } else {
        ind++;
    }
*/
}

unsigned buscar(char c, const Vector& v) {
    unsigned i = 0;

    while ((i < v.tam) && (c != v.datos[i])) {
        i++;
    }
}
```

```

        return i;
    }

bool comparar(const Vector& v1, unsigned i,
              const Vector& v2, unsigned j) {
    while ((i < v1.tam) && (v1.datos[i] == v2.datos[j])) {
        i++;
        incrementoCircular(j,v2.tam);
    }
    return i >= v1.tam;
}

bool iguales(const Vector& v1, const Vector& v2) {
    bool res;
    unsigned pos;

    if (v1.tam != v2.tam) {
        res = false;
    } else {
        pos = buscar(v1.datos[0],v2);
        if (pos == v2.tam) { // no encontrada
            res = false;
        } else {
            incrementoCircular(pos,v2.tam);
            res = comparar(v1,1,v2,pos);
        }
    }
    return res;
}

int main()
{
    Vector v1, v2;

    leer(v1);
    cin.ignore(10, '\n'); // para saltar el retorno de carro
    leer(v2);

    if (iguales(v1,v2)) {
        cout << "SI son iguales\n";
    } else {
        cout << "NO son iguales\n";
    }

    return 0;
}

```

- 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.

SOLUCIÓN

```
#include <iostream>

using namespace std;
```

```

const unsigned MAX = 10;
typedef char Componentes[MAX];
struct Vector {
    Componentes datos;
    unsigned tam;
};

void leer(Vector& v) {
    char c;

    cout << "Introduzca una secuencia de letras mayúsculas";
    cout << "(punto para terminar y como máximo " << MAX
          << " letras):\n";
}

v.tam = 0;
cin.get(c);
while ((c != '.') && (v.tam < MAX)) {
    v.datos[v.tam] = c;
    v.tam++;
    cin.get(c);
}

void completar(Vector& dest, const Vector& orig, unsigned i) {
    unsigned cont = i;

    while ((cont < orig.tam) && (dest.tam < MAX)) {
        dest.datos[dest.tam] = orig.datos[cont];
        dest.tam++;
        cont++;
    }
}

void mezclar(const Vector& v1, const Vector& v2, Vector& res) {
    unsigned i,j;

    i = 0;
    j = 0;
    res.tam = 0;

    while ((i < v1.tam) && (j < v2.tam) && (res.tam < MAX)) {
        if (v1.datos[i] < v2.datos[j]) {
            res.datos[res.tam] = v1.datos[i];
            i++;
        } else {
            res.datos[res.tam] = v2.datos[j];
            j++;
        }
        res.tam++;
    }

    completar(res,v1,i);
    completar(res,v2,j);
}

int main()
{

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
Vector v1, v2, v3;

leer(v1);
cin.ignore(10, '\n'); // para saltar el retorno de carro
leer(v2);

mezclar(v1,v2,v3);

cout << "La mezcla ordenada es:\n";
for (unsigned i = 0; i < v3.tam; i++) {
    cout << v3.datos[i] << " ";
}

return 0;
}
```

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).

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned DIAS = 31; // suponemos un mes con 31 días
typedef unsigned TFrec[DIAS]; // posición 0 => dia 1

void inicializar(TFrec& freq) {
    for (unsigned i = 0; i < DIAS; i++) {
        freq[i] = 0;
    }
}

void actualizar(TFrec& diasComunes) {
    unsigned dia;

    cin >> dia;
    while (dia != 0) {
        if ((1 <= dia) && (dia <= DIAS)) { // robusto
            diasComunes[dia-1]++;
        }
        cin >> dia;
    }
}

int main()
{
    TFrec diasComunes;
    unsigned alumnos;

    inicializar(diasComunes);
```



```

cout << "Cuántos alumnos va a introducir? ";
cin >> alumnos;

for (unsigned i = 0; i < alumnos; i++) {
    cout << "Introduzca los días preferidos por el alumno "
        << i+1 << " (introduzca un 0 para terminar:\n";
    actualizar(diasComunes);
}

cout << "Los días comunes son: ";
for (unsigned i = 0; i < DIAS; i++) {
    if (diasComunes[i] == alumnos) {
        cout << i+1 << " ";
    }
}
return 0;
}

```

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				
<i>[]</i>					<i>[]</i>				

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned MAX = 5;

typedef int TArray[MAX];

typedef bool TAnalizados[MAX];

void leer(TArray& a) {

    cout << "Introduzca " << MAX << " números enteros: ";
    for (unsigned i = 0; i < MAX; i++) {
        cin >> a[i];
    }
}

void inicializar(TAnalizados a) {
    for (unsigned i = 0; i < MAX; i++) {
        a[i] = false;
    }
}

unsigned primerAnalizable(const TAnalizados& ana) {
```

```

unsigned i = 0;

while ((i < MAX) && (ana[i])) {
    i++;
}

return i;
}

unsigned buscarIndMenor(const TArray& val, const TAnalizados& ana) {
    unsigned indMenor;

    indMenor = primerAnalizable(ana);

    for (unsigned i = indMenor+1; i < MAX; i++) {
        if ((!ana[i]) && (val[i] < val[indMenor])) {
            indMenor = i;
        }
    }

    return indMenor;
}

void calcular(const TArray& val, TArray& ind) {
    TAnalizados analizados;
    unsigned indMenor;

    inicializar(analizados);

    for (unsigned i = 0; i < MAX; i++) {
        indMenor = buscarIndMenor(val, analizados);
        ind[i] = indMenor;
        analizados[indMenor] = true;
    }
}

int main()
{
    TArray valores, indices;

    leer(valores);

    calcular(valores, indices);

    cout << "El orden de los indices es: ";
    for (unsigned i = 0; i < MAX; i++) {
        cout << indices[i] << " ";
    }

    return 0;
}

```

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

```

SOLUCIÓN:

VERSIÓN 1:

// Version 1: Mas sencilla y natural, pero menos eficiente

```

#include <iostream>

using namespace std;

typedef int TArray[MAX];

struct TLista {
    TArray elementos;
    unsigned numElem;
};

bool esta(int elem, const TLista& lista) {
    unsigned cont = 0;

    while ((cont < lista.numElem) && (elem != lista.elementos[cont])) {
        cont++;
    }

    return (cont < lista.numElem);
}

unsigned repeticiones(int elem, const TLista& lista) {
    unsigned res = 0;

    for (unsigned i = 0; i < lista.numElem; i++) {
        if (elem == lista.elementos[i]) {
            res++;
        }
    }

    return res;
}

void criba(unsigned x, const TLista& lista1, TLista& lista2) {
    lista2.numElem = 0;
    for (unsigned i = 0; i < lista1.numElem; i++) {
        if ((repeticiones(lista1.elementos[i], lista1) == x)
            && (!esta(lista1.elementos[i], lista2))) {
            lista2.elementos[lista2.numElem] = lista1.elementos[i];
            lista2.numElem++;
        }
    }
}

int main() {
    TLista lista1, lista2;
    unsigned repeticiones;
    int num;
}

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
cout << "Introduzca la lista original (0 para terminar y como maximo " << MAX <<
" elementos): ";
    lista1.numElem = 0;
    cin >> num;
    while ((lista1.numElem < MAX) && (num != 0)) {
        lista1.elementos[lista1.numElem] = num;
        lista1.numElem++;
        cin >> num;
    }

    cout << "Introduzca el numero de repeticiones para realizar la criba: ";
    cin >> repeticiones;

    criba(repeticiones, lista1, lista2);

    cout << "La lista cribada es: ";
    for (unsigned i = 0; i < lista2.numElem; i++) {
        cout << lista2.elementos[i] << " ";
    }

    return 0;
}
```

VERSIÓN 2:

```
// Version 2: Menos facil de ver, pero mas eficiente

typedef bool TContados[MAX];

void inicializar(TContados contados) {
    for (unsigned i = 0; i < MAX; i++) {
        contados[i] = false;
    }
}

// calcula las repeticiones y marca contados
void repeticiones(int pos, const TLista& lista, TContados& contados, unsigned& rep) {
    rep = 1;
    contados[pos] = true;

    for (unsigned i = pos + 1; i < lista.numElem; i++) {
        if (lista.elementos[pos] == lista.elementos[i]) {
            contados[i] = true;
            rep++;
        }
    }
}

void criba(unsigned x, const TLista& lista1, TLista& lista2) {
    TContados contados;
    unsigned rep;

    inicializar(contados);

    lista2.numElem = 0;
    for (unsigned i = 0; i < lista1.numElem; i++) {
        if (!contados[i]) {
            repeticiones(i, lista1, contados, rep);
            lista2.elementos[lista2.numElem] = lista1.elementos[i];
            lista2.numElem++;
        }
    }
}
```



```

        if (rep == x) {
            lista2.elementos[lista2.numElem] = lista1.elementos[i];
            lista2.numElem++;
        }
    }
}

int main() {
    TLista lista1, lista2;
    unsigned repeticiones;
    int num;

    cout << "Introduzca la lista original (0 para terminar y como maximo " << MAX <<
" elementos): ";
    lista1.numElem = 0;
    cin >> num;
    while ((lista1.numElem < MAX) && (num != 0)) {
        lista1.elementos[lista1.numElem] = num;
        lista1.numElem++;
        cin >> num;
    }

    cout << "Introduzca el numero de repeticiones para realizar la criba: ";
    cin >> repeticiones;

    criba(repeticiones, lista1, lista2);

    cout << "La lista cribada es: ";
    for (unsigned i = 0; i < lista2.numElem; i++) {
        cout << lista2.elementos[i] << " ";
    }

    return 0;
}

```

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

```

SOLUCIÓN:

```

#include <iostream>

using namespace std;

//Definición de constantes y tipos
const unsigned N = 5;
typedef unsigned TVector[N];

//Otra opción es usar abs de <cmath>
unsigned valorAbsoluto(int v) {
    unsigned res;

    if (v < 0) {
        res = -v;
    } else {
        res = v;
    }
    return res;
}

//Otra opción es hacerlo con dos funciones, una para cada sumatorio
int sumaPesos(const TVector &v, int inicio, int fin, unsigned centro)
{
    int suma = 0;

    for(int i = inicio; i <= fin; i++)
    {
        suma = suma + valorAbsoluto(centro-i) * v[i];
    }
    return suma;
}

void centroVector(const TVector &v, bool &existe, unsigned &centro)
{
    existe = false;
    centro = 1;
    while (centro <= N-2 && !existe)
    {
        if (sumaPesos(v,0,centro-1,centro) == sumaPesos(v,centro+1,N-1,centro)) {
            existe = true;
        } else {
            centro++;
        }
    }
}

```

```

int main()
{
    TVector v;
    bool existe;
    unsigned centro;

    cout << "Introduzca " << N << " numeros naturales: " << endl;
    for (unsigned i = 0; i < N; i++){
        cin >> v[i];
    }

    centroVector(v,existe,centro);

    if (existe){
        cout << "El centro de este vector es el indice " << centro << endl;
    }else{
        cout << "Este vector no tiene centro " << endl;
    }
}

```

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.

SOLUCIÓN:

VERSIÓN 1:

// Version 1: los elementos no están ordenados

```

#include <iostream>

using namespace std;

const unsigned MAX = 10;

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

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
unsigned buscar(int num, const Vector& v) {
    unsigned cont = 0;

    while ((cont < v.tam) && (v.numeros[cont] != num)) {
        cont++;
    }

    return cont;
}

// no ordenados
void borrar(int num, Vector& v) {
    unsigned pos = buscar(num,v);

    if (pos < v.tam) {
        v.numeros[pos] = v.numeros[v.tam-1];
        v.tam--;
    }
}

// no ordenados
void insertar(int num, Vector& v) {
    if (v.tam < MAX) {
        v.numeros[v.tam] = num;
        v.tam++;
    }
}

int main() {
    Vector v;
    int valor;

    do {
        cout << "Cuantos numeros va a introducir en el vector (maximo " << MAX <<
    "): ";
        cin >> v.tam;
    } while (v.tam > MAX);

    cout << "Introduzca " << v.tam << " numeros enteros:";
    for (unsigned cont = 0; cont < v.tam; cont++) {
        cin >> v.numeros[cont];
    }

    cout << "Introduzca un numero entero a borrar del vector: ";
    cin >> valor;

    borrar(valor,v);

    cout << "El vector despues de borrar es: ";
    for (unsigned cont = 0; cont < v.tam; cont++) {
        cout << v.numeros[cont] << " ";
    }
    cout << endl;

    cout << "Introduzca un numero entero a insertar en el vector: ";
    cin >> valor;

    insertar(valor,v);
```



```

        cout << "El vector despues de insertar es: ";
        for (unsigned cont = 0; cont < v.tam; cont++) {
            cout << v.numeros[cont] << " ";
        }
        cout << endl;

    return 0;
}

```

VERSIÓN 2:

// Version 1: los elementos sí están ordenados

```

#include <iostream>

using namespace std;

const unsigned MAX = 10;

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

unsigned buscar(int num, const Vector& v) {
    unsigned cont = 0;

    while ((cont < v.tam) && (v.numeros[cont] != num)) {
        cont++;
    }

    return cont;
}

// ordenados
void borrar(int num, Vector& v) {
    unsigned pos = buscar(num,v);

    if (pos < v.tam) {
        for (unsigned cont = pos; cont < v.tam - 1; cont++) {
            v.numeros[cont] = v.numeros[cont+1];
        }
        v.tam--;
    }
}

unsigned posicion(int num, const Vector& v) {
    unsigned cont = 0;

    while ((cont < v.tam) && (num > v.numeros[cont])) {
        cont++;
    }

    return cont;
}

```

```

// ordenados
void insertar(int num, Vector& v) {
    unsigned pos;
    if (v.tam < MAX) {
        pos = posicion(num,v);
        for (unsigned cont = v.tam; cont > pos; cont--) {
            v.numeros[cont] = v.numeros[cont-1];
        }
        v.numeros[pos] = num;
        v.tam++;
    }
}

int main() {
    Vector v;
    int valor;

    do {
        cout << "Cuantos numeros va a introducir en el vector (maximo " << MAX <<
    "): ";
        cin >> v.tam;
    } while (v.tam > MAX);

    cout << "Introduzca " << v.tam << " numeros enteros:" ;
    for (unsigned cont = 0; cont < v.tam; cont++) {
        cin >> v.numeros[cont];
    }

    cout << "Introduzca un numero entero a borrar del vector: ";
    cin >> valor;

    borrar(valor,v);

    cout << "El vector despues de borrar es: ";
    for (unsigned cont = 0; cont < v.tam; cont++) {
        cout << v.numeros[cont] << " ";
    }
    cout << endl;

    cout << "Introduzca un numero entero a insertar en el vector: ";
    cin >> valor;

    insertar(valor,v);

    cout << "El vector despues de insertar es: ";
    for (unsigned cont = 0; cont < v.tam; cont++) {
        cout << v.numeros[cont] << " ";
    }
    cout << endl;

    return 0;
}

```

14. 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

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned RANGO_MAY = 26; // las 26 letras mayúsculas

typedef unsigned TPosiciones[RANGO_MAY];
typedef unsigned TDistancias[RANGO_MAY];
typedef bool TRepeticiones[RANGO_MAY];

struct TDatos {
    TPosiciones pos; // ultima posicion de cada letra
    TDistancias dis; // mayores distancias para cada letra
    TRepeticiones rep; // si se ha repetido o no una letra
};

void inicializar(TDatos& datos) {
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        datos.pos[i] = 0;
        datos.dis[i] = 0;
        datos.rep[i] = false;
    }
}

void escribir(const TDatos& datos) {
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        if (datos.rep[i]) {
            cout << "Distancia entre " << char(i+'A')
                << ": " << datos.dis[i] << endl;
        }
    }
}

int main()
{
    TDatos datos;
    char c;
    unsigned pos, dis;

    inicializar(datos);

    cout << "Introduzca secuencia de mayúsculas (punto para finalizar): ";

    cin.get(c);
    pos = 1;
    while (c != '.') {
        if (('A' <= c) && (c <= 'Z')) { // robusto
            if (datos.pos[c-'A'] == 0) { // primera aparicion
                datos.pos[c-'A'] = pos;
                datos.dis[pos] = 1;
                for (unsigned i = pos+1; i < RANGO_MAY; i++)
                    datos.dis[i] = pos;
            } else
                datos.dis[datos.pos[c-'A']]++;
        }
        cin.get(c);
    }
}
```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
        } else {
            datos.rep[c-'A'] = true;
            dis = pos - datos.pos[c-'A'] - 1;
            datos.pos[c-'A'] = pos;
            if (dis > datos.dis[c-'A']) {
                datos.dis[c-'A'] = dis;
            }
        }
    }
    cin.get(c);
    pos++;
}
escribir(datos);
return 0;
}
```

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.

SOLUCIÓN:

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

#include <iostream>
using namespace std;

const unsigned NUM_FILAS = 3;
const unsigned NUM_COLUMNAS = 4;

typedef int TFilas[NUM_COLUMNAS];
typedef TFilas TMatriz[NUM_FILAS];

void leer(TMatriz& m) {
    cout << "Introduzca la matriz fila a fila (" 
        << NUM_FILAS << " X " << NUM_COLUMNAS << "):\n";
    for (unsigned f = 0; f < NUM_FILAS; f++) {
        for (unsigned c = 0; c < NUM_COLUMNAS; c++) {
            cin >> m[f][c];
        }
    }
}

bool esta(int num, const TMatriz& m) {
    bool encontrado = false;
    unsigned fil,col;

    fil = 0;
    while ((fil < NUM_FILAS) && !encontrado) {
        col = 0;
        while ((col < NUM_COLUMNAS) && !encontrado) {
            if (m[fil][col] == num) {
```



```

        encontrado = true;
    } else {
        col++;
    }
}
if (!encontrado) {
    fil++;
}
}

return encontrado;
}

int main()
{
    TMatriz m;
    int num;

    leer(m);

    cout << "Introduzca el numero entero a buscar: ";
    cin >> num;

    if (esta(num,m)) {
        cout << "El numero " << num << " SI esta en la matriz" << endl;
    } else {
        cout << "El numero " << num << " NO esta en la matriz" << endl;
    }

    return 0;
}

```

16. Diseña un algoritmo para calcular la suma de los elementos de la diagonal principal de una matriz cuadrada.

SOLUCIÓN:

```

#include <iostream>

using namespace std;

const unsigned MAX = 5;

typedef unsigned TFilas[MAX];
typedef TFilas TMatriz[MAX];

void leer(TMatriz& m) {
    cout << "Introduzca la matriz fila a fila (" 
        << MAX << " X " << MAX << "):\n";
    for (unsigned f = 0; f < MAX; f++) {
        for (unsigned c = 0; c < MAX; c++) {
            cin >> m[f][c];
        }
    }
}

unsigned sumaDiagonalPrinc(const TMatriz& m) {
    unsigned res = 0;

```

```

    for (unsigned i = 0; i < MAX; i++) {
        res += m[i][i];
    }
    return res;
}

int main()
{
    TMatriz m;

    leer(m);

    cout << "La suma de la diagonal principal es: "
        << sumaDiagonalPrinc(m);

    return 0;
}

```

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 10x10, muestre en la pantalla las coordenadas de su punto silla si lo tiene o una indicación de que no lo tiene.

SOLUCIÓN

VERSIÓN 1: (máximos y mínimos no estrictos => pueden existir varios puntos de silla (o ninguno)

Version 1.1 No eficiente

```

#include <iostream>

using namespace std;

const unsigned MAX = 10;

typedef int TFilas[MAX];
typedef TFilas TArrayBid[MAX];

struct TMatriz {
    TArrayBid datos;
    unsigned nFil, nCol;
};

void leer(TMatriz& m) {
    do {
        cout << "Introduzca número de filas y columnas (máximo "
            << MAX << " X " << MAX << "): ";
        cin >> m.nFil >> m.nCol;
    } while ((m.nFil < 1) || (m.nFil > MAX) ||
             (m.nCol < 1) || (m.nCol > MAX));

    cout << "Introduzca la matriz fila a fila:\n";

    for (unsigned f = 0; f < m.nFil; f++) {
        for (unsigned c = 0; c < m.nCol; c++) {
            cin >> m.datos[f][c];
        }
    }
}

```

```

        }

    }

int menorFil(unsigned f, const TMatriz& m) {
    int res = m.datos[f][0];

    for (unsigned c = 1; c < m.nCol; c++) {
        if (m.datos[f][c] < res) {
            res = m.datos[f][c];
        }
    }
    return res;
}

int mayorCol(unsigned c, const TMatriz& m) {
    int res = m.datos[0][c];

    for (unsigned f = 1; f < m.nFil; f++) {
        if (m.datos[f][c] > res) {
            res = m.datos[f][c];
        }
    }
    return res;
}

bool esPuntoSilla(unsigned f, unsigned c, const TMatriz& m) {
    return m.datos[f][c] == menorFil(f,m) &&
           m.datos[f][c] == mayorCol(c,m);
}

void mostrarPuntosSilla(const TMatriz& m) {

    cout << "Sus puntos silla son:\n";
    for (unsigned f = 0; f < m.nFil; f++) {
        for (unsigned c = 0; c < m.nCol; c++) {
            if (esPuntoSilla(f,c,m)) {
                cout << "Fila: " << f << ", " << "Columna: "
                     << c << endl;
            }
        }
    }
}

int main()
{
    TMatriz m;

    leer(m);

    mostrarPuntosSilla(m);

    return 0;
}

```

Versión 1.2 Eficiente (para cada iteración del for de las filas calculo el menor y después, se recorre la fila y cada vez que encontremos uno igual al menor se calcula el mayor de la columna)

```

#include <iostream>

using namespace std;

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
const unsigned MAX = 10;

typedef int TFilas[MAX];
typedef TFilas TArrayBid[MAX];

struct TMatriz {
    TArrayBid datos;
    unsigned nFil, nCol;
};

void leer(TMatriz& m) {
    do {
        cout << "Introduzca número de filas y columnas (máximo "
            << MAX << " X " << MAX << ") : ";
        cin >> m.nFil >> m.nCol;
    } while ((m.nFil < 1) || (m.nFil > MAX) ||
             (m.nCol < 1) || (m.nCol > MAX));

    cout << "Introduzca la matriz fila a fila:\n";

    for (unsigned f = 0; f < m.nFil; f++) {
        for (unsigned c = 0; c < m.nCol; c++) {
            cin >> m.datos[f][c];
        }
    }
}

int Menor(const TMatriz& m, unsigned f) {
    int res = m.datos[f][0];

    for (unsigned c = 1; c < m.nCol; c++) {
        if (m.datos[f][c] < res) {
            res = m.datos[f][c];
        }
    }
    return res;
}

int Mayor(const TMatriz& m, unsigned c) {
    int res = m.datos[0][c];

    for (unsigned f = 1; f < m.nFil; f++) {
        if (m.datos[f][c] > res) {
            res = m.datos[f][c];
        }
    }
    return res;
}

void mostrarPuntosSilla(const TMatriz& m) {
    int menor;

    cout << "Sus puntos silla son:\n";
    for (unsigned f = 0; f < m.nFil; f++) {
        menor = Menor(m, f);
        for (unsigned c = 0; c < m.nCol; c++) {
            if (m.datos[f][c] == menor) {
                if (m.datos[f][c] == Mayor(m, c)) {
                    cout << "Fila: " << f << ", " << "Columna: "
                        << c << endl;
                }
            }
        }
    }
}
```



```

        }
    }
}

int main()
{
    TMatriz m;

    leer(m);

    mostrarPuntosSilla(m);

    return 0;
}

```

VERSIÓN 2: (máximos y mínimos estrictos => sólo puede existir un punto de silla (o ninguno)
De la version 1.2 hacerlo sabiendo que tiene que ser un menor estricto y luego un mayor estricto
(algo parecido a la moda)

```

#include <iostream>

using namespace std;

const unsigned MAX = 10;

typedef int TFilas[MAX];
typedef TFilas TArrayBid[MAX];

struct TMatriz {
    TArrayBid datos;
    unsigned nFil, nCol;
};

void leer(TMatriz& m) {

    do {
        cout << "Introduzca número de filas y columnas (máximo "
            << MAX << " X " << MAX << "): ";
        cin >> m.nFil >> m.nCol;
    } while ((m.nFil < 1) || (m.nFil > MAX) ||
        (m.nCol < 1) || (m.nCol > MAX));

    cout << "Introduzca la matriz fila a fila:\n";

    for (unsigned f = 0; f < m.nFil; f++) {
        for (unsigned c = 0; c < m.nCol; c++) {
            cin >> m.datos[f][c];
        }
    }
}

void Menor(const TMatriz& m, unsigned f, unsigned& colMenor, bool& hay) {
    colMenor = 0;
    hay = true;

    for (unsigned c = 1; c < m.nCol; c++) {
        if (m.datos[f][c] < m.datos[f][colMenor]) {
            colMenor = c;
            hay = true;
        }
    }
}

```

```

        } else if (m.datos[f][c] == m.datos[f][colMenor]) {
            hay = false;
        }
    }

void Mayor(const TMatriz& m, unsigned c, unsigned& filMayor, bool& hay) {
    filMayor = 0;
    hay = true;

    for (unsigned f = 1; f < m.nFil; f++) {
        if (m.datos[f][c] > m.datos[filMayor][c]) {
            filMayor = f;
            hay = true;
        } else if (m.datos[f][c] == m.datos[filMayor][c]) {
            hay = false;
        }
    }
}

void mostrarPuntoSilla(const TMatriz& m) {

    unsigned colMenor, filMayor;
    bool hayMenorEstricto, hayMayorEstricto;
    bool encontrado = false;

    unsigned f = 0;
    while (!encontrado && (f < m.nFil)) {
        Menor(m, f, colMenor, hayMenorEstricto);
        if (hayMenorEstricto) {
            Mayor(m, colMenor, filMayor, hayMayorEstricto);
            encontrado = hayMayorEstricto && (filMayor == f);
        }
        f++;
    }

    if (encontrado) {
        cout << "El punto silla es:\nFila: " << filMayor << ", " <<
    "Columna: "
                                            << colMenor << endl;
    } else {
        cout << "No existe punto silla\n";
    }
}

int main()
{
    TMatriz m;

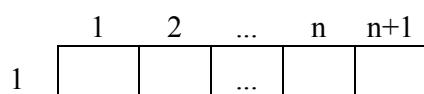
    leer(m);

    mostrarPuntoSilla(m);

    return 0;
}

```

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



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 devuelve como parámetro de salida un tablero N-goro con sus casillas llenas 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

SOLUCIÓN:

```
#include <iostream>

using namespace std;

const unsigned N = 4;

typedef unsigned TFilas[N+1];
typedef TFilas TTablero[N];

void escribir(const TTablero& m) {
    for (unsigned f = 0; f < N; f++) {
        for (unsigned c = 0; c < N+1; c++) {
            cout << m[f][c] << " ";
        }
        cout << endl;
    }
}

void incrementoCircular(unsigned& ind, unsigned max) {
    ind = (ind + 1) % max;
    /*
    if (ind == max) {
        ind = 0;
    } else {
        ind++;
    }
}

```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
*/  
}  
  
void sigCoordenada(unsigned& x, unsigned& y) {  
    incrementoCircular(x,N);  
    incrementoCircular(y,N+1);  
}  
  
void construir(TTablero& m) {  
    unsigned x = 0, y = 0;  
  
    for (unsigned i = 1; i <= N * (N+1); i++) {  
        m[x][y] = i;  
        sigCoordenada(x,y);  
    }  
}  
  
int main()  
{  
    TTablero m;  
  
    construir(m);  
  
    cout << "El tablero N-Goro para N = " << N << " es:\n";  
  
    escribir(m);  
  
    return 0;  
}
```

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 escrutar 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.

SOLUCIÓN:

```
#include <iostream>

using namespace std;

using namespace std;

const unsigned MAX_CARGOS = 15;
const unsigned MAX_PARTIDOS = 10;

struct TPartido {
    string nombre;
    unsigned cargos;
};

typedef TPartido TArray[MAX_PARTIDOS];

struct TResultado {
    TArray partidos;
    unsigned numPartidos;
};

typedef unsigned TMatriz[MAX_PARTIDOS] [MAX_CARGOS];
```

```

struct TDivisores {
    TMatriz valores;
    unsigned numPartidos;
    unsigned numCargos;
};

void leerDatos(TDivisores& divisores, TResultado& resultado) {
    do {
        cout << "Introduzca el Número de Cargos (>= 1 y <= 15): ";
        cin >> divisores.numCargos;
    } while ((divisores.numCargos < 1) || (divisores.numCargos > 15));

    do {
        cout << "Introduzca el Número de Partidos (>= 1 y <= 10): ";
        cin >> divisores.numPartidos;
    } while ((divisores.numPartidos < 1) || (divisores.numPartidos > 10));

    resultado.numPartidos = divisores.numPartidos;

    cout << "Introduzca el Nombre y Número de Votos por Partido:\n";
    for (unsigned i = 0; i < resultado.numPartidos; i++) {
        cout << "Partido " << i+1 << ": ";
        cin >> resultado.partidos[i].nombre >> divisores.valores[i][0];
        resultado.partidos[i].cargos = 0;
    }
}

void calcularDivisores(TDivisores& divisores) {
    for (unsigned fil = 0; fil < divisores.numPartidos; fil++) {
        for (unsigned col = 1; col < divisores.numCargos; col++) {
            divisores.valores[fil][col] = divisores.valores[fil][0] / (col+1);
        }
    }
}

void calcularMayor(const TDivisores& divisores, unsigned& fil, unsigned& col) {
    fil = 0;
    col = 0;
    for (unsigned f = 0; f < divisores.numPartidos; f++) {
        for (unsigned c = 0; c < divisores.numCargos; c++) {
            if (divisores.valores[f][c] > divisores.valores[fil][col]) {
                fil = f;
                col = c;
            }
        }
    }
}

void calcularCargos(TDivisores& divisores, TResultado& resultado) {
    unsigned filMayor,colMayor;
    for (unsigned i = 0; i < divisores.numCargos; i++) {
        calcularMayor(divisores,filMayor,colMayor);
        resultado.partidos[filMayor].cargos++;
        divisores.valores[filMayor][colMayor] = 0;
    }
}

void escribirResultados(const TResultado& resultado) {
    cout << "Los Cargos Electos son:\n";
    for (unsigned i = 0; i < resultado.numPartidos; i++) {
        if (resultado.partidos[i].cargos != 0) {

```

```

        cout << resultado.partidos[i].nombre << " " <<
resultado.partidos[i].cargos;
    }
    cout << endl;
}
}

int main() {
    TDivisores divisores;
    TResultado resultado;

    leerDatos(divisores,resultado);
    calcularDivisores(divisores);
    calcularCargos(divisores,resultado);
    escribirResultados(resultado);

    return 0;
}

```

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		*							
+---+-----+---+---+---+					+---+-----+---+---+---+				

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



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;
}
```

SOLUCIÓN:

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

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

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

void inicializar(Lava& lava) {
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            lava[i][j] = ' ';
        }
    }
}

bool valida(int f, int c) {
    return (0 <= f) && (f < FILAS) && (0 <= c) && (c < COLUMNAS);
}

void desplazar(int fOrig, int cOrig, int fDest, int cDest, const Superficie& sup, Lava& lava) {
    if (valida(fDest,cDest) && (sup[fDest][cDest] < sup[fOrig][cOrig])) {
        lava[fDest][cDest] = 'P';
    }
}
```



```

    }
}

void desplazarVecinos(int f, int c, const Superficie& sup, Lava& lava) {
    desplazar(f,c,f,c-1,sup,lava);
    desplazar(f,c,f,c+1,sup,lava);
    desplazar(f,c,f-1,c,sup,lava);
    desplazar(f,c,f+1,c,sup,lava);
}

void nuevaPosicion(const Lava& lava, int& f, int& c, bool& estable) {
    int i,j;

    estable = true;
    i = 0;
    while ((i < FILAS) && estable) {
        j = 0;
        while ((j < COLUMNAS) && estable) {
            if (lava[i][j] == 'P') {
                f = i;
                c = j;
                estable = false;
            }
            j++;
        }
        i++;
    }
}

// solución iterativa
/*
void flujoDeLava(const Superficie& sup, int fil, int col, Lava& lava) {
    bool estable;
    int f,c;

     inicializar(lava); // todas las casillas a blanco
    f = fil;
    c = col;
    lava[f][c] = 'P'; // pendiente de tratar
    do {
        desplazarVecinos(f,c,sup,lava);
        lava[f][c] = '*';
        nuevaPosicion(lava,f,c,estable);
    } while (!estable);
}
*/
// solución recursiva
void flujoDeLavaRec(const Superficie& sup, int fil, int col, Lava& lava) {
    lava[fil][col] = '*';
    if (valida(fil,col-1) && (sup[fil][col-1] < sup[fil][col])) {
        flujoDeLavaRec(sup,fil,col-1,lava);
    }
    if (valida(fil,col+1) && (sup[fil][col+1] < sup[fil][col])) {
        flujoDeLavaRec(sup,fil,col+1,lava);
    }
    if (valida(fil-1,col) && (sup[fil-1][col] < sup[fil][col])) {
        flujoDeLavaRec(sup,fil-1,col,lava);
    }
}

```

```

    if (valida(fil+1,col) && (sup[fil+1][col] < sup[fil][col])) {
        flujoDeLavaRec(sup,fil+1,col,lava);
    }
}

void flujoDeLava(const Superficie& sup, int fil, int col, Lava& lava) {
    inicializar(lava);
    flujoDeLavaRec(sup,fil,col,lava);
}

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):";
    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)

SOLUCIÓN:

```
#include <iostream>
#include <string>

using namespace std;

const unsigned MAX = 20;

struct TFecha {
    unsigned dia, mes, anyo;
};

struct TProducto {
    unsigned codigo;
    string nombre;
    float precio;
    TFecha caducidad;
};

typedef TProducto TArray[MAX];

struct TFarmacia {
    TArray productos;
    unsigned nProd;
};

void inicializarFarmacia(TFarmacia& f) {
    f.nProd = 0;
}

void leerProducto(TProducto& p) {
    cout << "Datos del producto\n";
    cout << "Código: ";
    cin >> p.codigo;
    cout << "Nombre: ";
    cin >> p.nombre;
    cout << "Precio: ";
    cin >> p.precio;
    cout << "Fecha de caducidad: ";
    cin >> p.caducidad.dia >> p.caducidad.mes >> p.caducidad.anyo;
}

void escribirProducto(const TProducto& p) {
    cout << "Datos del producto\n";
    cout << "Código: " << p.codigo << endl;
    cout << "Nombre: " << p.nombre << endl;
    cout << "Precio: " << p.precio << endl;
    cout << "Fecha de caducidad: ";
    cout << p.caducidad.dia << "/";
    cout << p.caducidad.mes << "/";
    cout << p.caducidad.anyo << endl;
}

void insertarProducto(TFarmacia& f, const TProducto& p) {
    if (f.nProd < MAX) {
        f.productos[f.nProd] = p;
        f.nProd++;
    }
}
```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
}

void buscarProductoCodigo(const TFarmacia& f, unsigned codigo,
                           bool& encontrado, TProducto& p) {
    unsigned i = 0;

    while ((i < f.nProd) && (f.productos[i].codigo != codigo)) {
        i++;
    }

    encontrado = i < f.nProd;
    if (encontrado) {
        p = f.productos[i];
    }
}

void buscarProductoNombre(const TFarmacia& f, string nombre,
                           bool& encontrado, TProducto& p) {
    unsigned i = 0;

    while ((i < f.nProd) && (f.productos[i].nombre != nombre)) {
        i++;
    }

    encontrado = i < f.nProd;
    if (encontrado) {
        p = f.productos[i];
    }
}

void borrarProducto(TFarmacia& f, unsigned codigo) {
    unsigned i = 0;

    while ((i < f.nProd) && (f.productos[i].codigo != codigo)) {
        i++;
    }

    if (i < f.nProd) { // encontrado
        f.productos[i] = f.productos[f.nProd-1];
        f.nProd--;
    }
}

void listarFarmacia(const TFarmacia& f) {
    cout << "La farmacia almacena los siguientes productos:\n";

    for (unsigned i = 0; i < f.nProd; i++) {
        escribirProducto(f.productos[i]);
        cout << endl << endl;
    }
}

int main() {
    // ....
    return 0;
}
```



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.

SOLUCIÓN:

VERSIÓN 1

```
#include <iostream>
#include <string>

using namespace std;

bool estaContenida(char c, const string& s) {
    unsigned i = 0;

    while ((i < s.size()) && (c != s[i])) {
        i++;
    }

    return i < s.size();
}

int main() {
    string primera, pal;

    cout << "Introduzca un texto terminado con la palabra FIN\n";
    cin >> primera;
    if (primera != "FIN") {
        cout << "Las palabras cuya inicial está en la primera palabra del
        texto son:\n";
        cin >> pal;
        while (pal != "FIN") {
            if (estaContenida(pal[0], primera)) {
                cout << pal << " ";
            }
            cin >> pal;
        }
    }
}
```

VERSIÓN 2:

```
#include <iostream>
#include <string>

using namespace std;

const unsigned RANGO_MAY = 26; // las 26 letras mayúsculas
```

```

typedef bool TIniciales[RANGO_MAY];

void inicializar(TIniciales& iniciales) {
    for (unsigned i = 0; i < RANGO_MAY; i++) {
        iniciales[i] = false;
    }
}

void guardar(const string& s, TIniciales& iniciales) {

    inicializar(iniciales);
    for (unsigned i = 0; i < s.size(); i++) {
        iniciales[s[i]-'A'] = true;
    }
}

int main() {
    string primera, pal;
    TIniciales iniciales;

    cout << "Introduzca un texto terminado con la palabra FIN\n";
    cin >> primera;
    if (primera != "FIN") {
        guardar(primera,iniciales);
        cout << "Las palabras cuya inicial está en la primera palabra del
texto son:\n";
        cin >> pal;
        while (pal != "FIN") {
            if (iniciales[pal[0]-'A']) {
                cout << pal << " ";
            }
            cin >> pal;
        }
    }
}

```

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.

SOLUCIÓN:

```
#include <iostream>
#include <string>

using namespace std;

const unsigned MAX_PAL_DIS = 20;

typedef string TPalabras[MAX_PAL_DIS];

struct TDatos {
    TPalabras pal;
    unsigned nPal;
};

bool contiene(const string& pal, const string& patron) {
    unsigned i, tope;
    bool encontrado = false;

    if (pal.size() >= patron.size()) {
        tope = pal.size() - patron.size();
        i = 0;
        while ( (i <= tope) && !encontrado) {
            if (pal[i] == patron[0]) {
                encontrado = pal.substr(i, patron.size()) == patron;
            }
            i++;
        }
    }

    return encontrado;
}

bool esta(const string& pal, const TDatos& datos) {
    unsigned i = 0;

    while ((i < datos.nPal) && (pal != datos.pal[i])) {
        i++;
    }

    return i < datos.nPal;
}

void escribir(const string& patron, const TDatos& datos) {
    cout << "Las palabras que contienen a " << patron << " son:\n";
    for (unsigned i = 0; i < datos.nPal; i++) {
        cout << datos.pal[i] << " ";
    }
}

int main()
{
    TDatos datos;
    string pal, patron;
```

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio.



```
cout << "Introduzca la cadena de caracteres patrón: ";
cin >> patron;
cout << "Introduzca el texto terminado con la palabra FIN\n";
datos.nPal = 0;
cin >> pal;
while (pal != "FIN") {
    if ((contiene(pal, patron)) && (!esta(pal, datos))) {
        datos.pal[datos.nPal] = pal;
        datos.nPal++;
    }
    cin >> pal;
}
escribir(patron, datos);
return 0;
}
```

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.



SOLUCIÓN:

```

#include <iostream>

using namespace std;

const unsigned MAX_PAL_DIST = 20;
const unsigned MAX_REP = 10;

typedef unsigned TPosiciones[MAX_REP];

struct TPalabra {
    string palabra;
    TPosiciones posiciones;
    unsigned nposic;
};

typedef TPalabra TColeccion[MAX_PAL_DIST];

struct TDatos {
    TColeccion palabras;
    unsigned npal;
};

unsigned buscar(const TDatos& datos, const string& pal) {
    unsigned i = 0;

    while ((i < datos.npal) && (datos.palabras[i].palabra != pal)) {
        i++;
    }

    return i;
}

void almacenar(TDatos& datos, const string& pal, unsigned pos) {
    unsigned ind = buscar(datos,pal);

    if (ind >= datos.npal) { // primera aparicion
        datos.palabras[datos.npal].palabra = pal;
        datos.palabras[datos.npal].posiciones[0] = pos;
        datos.palabras[datos.npal].nposic = 1;
        datos.npal++;
    } else { // se repite la palabra
        datos.palabras[ind].posiciones[datos.palabras[ind].nposic] = pos;
        datos.palabras[ind].nposic++;
    }
}

void escribir(const TDatos& datos) {

    for (unsigned i = 0; i < datos.npal; i++) {
        cout << datos.palabras[i].palabra;
        for (unsigned j = 0; j < datos.palabras[i].nposic; j++) {
            cout << " " << datos.palabras[i].posiciones[j] << " ";
        }
        cout << endl;
    }
}

```

```

int main() {
    string palabra;
    unsigned pos;
    TDatos datos;

    datos.npal = 0;

    cout << "Introduzca texto (FIN para terminar)\n";

    cin >> palabra;
    pos = 1;
    while (palabra != "FIN") {
        almacenar(datos,palabra,pos);
        cin >> palabra;
        pos++;
    }

    escribir(datos);

    return 0;
}

```