UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y CC. Computación
E.T.S.I. Telecomunicación

Programación I

Relación de Ejercicios
y Soluciones

Sonido e Imagen

Contenido

Nota: en la solución a los ejercicios, se ha utilizado el tipo `array` de TR1, que ha sido incorporado a la biblioteca de C++ en el estándar de 2011. Si su biblioteca estándar no contiene la definición del tipo `array`, puede descargarla desde la siguiente dirección: `http://www.lcc.uma.es/%7Evicente/docencia/cpplibs/array_tr1.zip`

Tema 2: Conceptos Básicos de Programación

Práctica 1. Programa, Variables y Expresiones

Laboratorio

1. Estudie el *"Entorno del programación"* a utilizar durante el curso en el documento
   `http://www.lcc.uma.es/%7Evicente/docencia/cppide/ep_anex_entprog_cpp.pdf`
   Además, puede conocer más comandos del *terminal de Linux* en el documento
   `http://www.lcc.uma.es/%7Evicente/docencia/docencia/ep_anex_unix.pdf`

2. El siguiente programa escrito en C++ calcula la cantidad bruta y neta a pagar por un trabajo realizado en función de las horas y días trabajados. Contiene errores, encuéntrelos y corríjalos.

```
#include <iostream>
using namespace std;
const tasa : 25.0;
const PRECIO_HORA = 60.0;
int main()
{
    double horas,dias,total,neto;
    cout << "Introduzca las horas trabajadas: ";
    cin << horas;
    cout << "Introduzca los dias trabajados: ";
    cin >> dias;
    horas*dias*PRECIO_HORA = total;
    neto = total-TASA;
    cout >> "El valor total a pagar es: " >> total >> endl;
    cout << "El valor neto a pagar es: " << NETO << endl;
}
```

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
const double TASA = 25.0;
const double PRECIO_HORA = 60.0;
//-----------------------------------------
int main()
{
    double horas, dias, total, neto;
    cout << "Introduzca las horas trabajadas: ";
    cin >> horas;
    cout << "Introduzca los dias trabajados: ";
    cin >> dias;
    total = horas * dias * PRECIO_HORA;
    neto = total - TASA;
    cout << "El valor total a pagar es: " << total << endl;
    cout << "El valor neto a pagar es: " << neto << endl;
}
```

3. Desarrolle un programa que lea dos números de tipo `int` de teclado y posteriormente los escriba en pantalla. Ejecútelo introduciendo dos números de tipo `int` válidos (por ejemplo `1234` y `5678`). Posteriormente ejecútelo introduciendo por teclado un primer número de tipo `int` (por ejemplo `1234`) e introduciendo por teclado un segundo dato que no pertenezca al tipo `int` (por ejemplo `hola`). Finalmente ejecútelo introduciendo por teclado un primer dato que no pertenezca al tipo `int` (por ejemplo `hola`). Evalúe las diferencias entre ambas ejecuciones del mismo programa.

Solución

```cpp
#include <iostream>
using namespace std;
int main()
{
    int dato1, dato2;
    cout << "Introduzca un número entero: ";
    cin >> dato1;
    cout << "Introduzca otro número entero: ";
    cin >> dato2;
    cout << "El valor del primer número introducido es: " << dato1 << endl;
    cout << "El valor del segundo número introducido es: " << dato2 << endl;
}
//-------------------------------------------------------------------
// Cuando se introduce por teclado un valor adecuado al tipo de la
// variable que se está leyendo, la lectura se hace correctamente.
// Sin embargo, si se introduce por teclado un valor NO adecuado al
// tipo de la variable que se está leyendo, entonces la operación de
// lectura falla, la variable mantiene el valor que tuviese
// anteriormente (en este caso como la variable no ha sido
// inicializada, tendrá un valor INESPECIFICADO), y el flujo de
// entrada (CIN) se pondrá en modo erróneo, por lo que cualquier otra
// operación de entrada que se realice posteriormente también fallará.
```

4. Desarrolle un programa que sólo declare variables de tipo `int`. El programa deberá leer dos números enteros desde el teclado, posteriormente los sumará, almacenando el resultado en una variable, y finalmente escribirá por pantalla el resultado de la suma. Ejecute dicho programa introduciendo como datos de entrada los siguientes números y analice los resultados obtenidos.

   a) -20 y 30.
   b) 20 y -30.
   c) 147483647 y 2000000000.
   d) 200000000 y 2000000000.
   e) 1 y 2147483647.
   f) 1 y 3000000000.

Solución

```cpp
#include <iostream>
using namespace std;
int main()
{
    int dato_1, dato_2;
    cout << "Introduzca el primer número entero: ";
    cin >> dato_1;
    cout << "Introduzca el segundo número entero: ";
    cin >> dato_2;
    int suma = dato_1 + dato_2;
    cout << "El valor resultado es: " << suma << endl;
}
//-------------------------------------------------------------------
// El tipo INT se representa con 32 bits en las máquinas actuales, por
// lo que el menor número que puede representar es el -2147483648
// (-2^31) y el mayor número que puede representar es 2147483647
// (2^31-1). Por lo tanto en los primeros casos, tanto los números
// leídos como el resultado de la operación pueden ser representados
// por el tipo INT, sin embargo en el cuarto y quinto casos el
// resultado de la operación (2200000000 y 2147483648) no puede ser
// representado, y en el sexto caso, incluso el segundo número leído
// (3000000000) tampoco puede ser representado por el tipo INT.
```

5. Desarrolle un programa que lea de teclado una determinada cantidad de *euros*, calcule y escriba su equivalente en *pesetas*, considerando que 1€ son 166.386 *pts*.

Solución

```cpp
#include <iostream>
using namespace std;
//-----------------------------------
const double EUR_PTS = 166.386;
//-----------------------------------
```

```
int main()
{
    cout << "Introduzca la cantidad de euros: ";
    double euros;
    cin >> euros;
    double pts = euros * EUR_PTS;
    cout << euros << " euros son " << pts << " pts" << endl;
}
```

6. Desarrolle un programa que calcule y escriba la media aritmética de 3 números enteros leídos de teclado. Compruebe que la media aritmética de los números `3`, `5` y `8` es `5.33333`

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
const int N_VALORES = 3;
//-----------------------------------------
int main()
{
    cout << "Introduzca 3 números enteros: ";
    int n1, n2, n3;
    cin >> n1 >> n2 >> n3;
    double media = double(n1 + n2 + n3) / double(N_VALORES);
    cout << "Media " << media<< endl;
}
```

7. Desarrolle un programa que lea de teclado una cierta cantidad de segundos y muestre su equivalente en semanas, días, horas, minutos y segundos, según el formato de los siguientes ejemplos:

   - `2178585` segundos equivalen a `[  3] semanas, 4 dias, 05:09:45`
   - `9127145` segundos equivalen a `[ 15] semanas, 0 dias, 15:19:05`

Solución

```
#include <iostream>
#include <iomanip>
using namespace std;
//-----------------------------------------
const int SEG_MIN = 60;
const int MIN_HORA = 60;
const int HORAS_DIA = 24;
const int DIAS_SEMANA = 7;
//-----------------------------------------
const int SEG_HORA = SEG_MIN * MIN_HORA;
const int SEG_DIA = SEG_HORA * HORAS_DIA;
const int SEG_SEMANA = SEG_DIA * DIAS_SEMANA;
//-----------------------------------------
int main()
{
    cout << "Introduzca los segundos: ";
    int segundos_totales;
    cin >> segundos_totales;
    //-------------------------------------
    int semanas = segundos_totales / SEG_SEMANA;
    int resto = segundos_totales % SEG_SEMANA;
    //-------------------
    int dias = resto / SEG_DIA;
    resto = resto % SEG_DIA;
    //-------------------
    int horas = resto / SEG_HORA;
    resto = resto % SEG_HORA;
    //-------------------
    int minutos = resto / SEG_MIN;
    int segundos = resto % SEG_MIN;
    //-------------------------------------
    cout << segundos_totales << " segundos equivalen a "
         << "[" << setfill(' ') << setw(3) << semanas << "] semanas, "
         << dias << " dias "
```

```
        << setfill('0') << setw(2) << horas << ":"
        << setfill('0') << setw(2) << minutos << ":"
        << setfill('0') << setw(2) << segundos << ":"
        << endl;
}
```

Tema 2: Conceptos Básicos de Programación

Práctica 1. Programa, Variables y Expresiones

Ejercicios Complementarios

1. Desarrolle un programa que lea de teclado dos números enteros y los almacene en dos variables de tipo `int`. Posteriormente deberá intercambiar los valores almacenados en dichas variables, y finalmente deberá escribir el valor almacenado en cada una de ellas.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Introduzca dos números naturales: ";
    int dato1, dato2;
    cin >> dato1 >> dato2;
    int aux = dato1;
    dato1 = dato2;
    dato2 = aux;
    cout << "Valor almacenado en Dato1: " << dato1 << endl;
    cout << "Valor almacenado en Dato2: " << dato2 << endl;
}
```

2. Desarrolle un programa que lea de teclado un número entero, y escriba `true` si el número leído es *par*, y `false` en caso contrario.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Introduzca un número natural: ";
    int dato;
    cin >> dato;
    bool es_par = ( dato % 2 == 0 );
    cout << "El número " << dato << " es par ? " << boolalpha << es_par << endl;
}
```

3. Desarrolle un programa que lea de teclado una letra minúscula (supondremos que la entrada de datos es correcta), y escriba la letra mayúscula correspondiente a la letra minúscula leída previamente.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Introduzca una letra minúscula: ";
    char letra_minuscula;
    cin >> letra_minuscula;
    char letra_mayuscula = char( int('A') + (int(letra_minuscula) - int('a')) );
    cout << letra_minuscula << " -> " << letra_mayuscula << endl;
}
```

4. Desarrolle un programa que lea de teclado tres dígitos como caracteres y los almacene en tres variables de tipo `char` (supondremos que la entrada de datos es correcta), y calcule el valor numérico correpondiente a dichos dígitos leídos y lo almacene en una variable de tipo `int`. Finalmente mostrará en pantalla el valor calculado. Por ejemplo, si lee los siguientes tres caracteres `'3'`, `'4'`, y `'5'`, deberá calcular el valor numérico correspondiente `345`.

Solución

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

```
int main()
{
    char d1, d2, d3;
    cout << "Introduzca tres dígitos: ";
    cin >> d1 >> d2 >> d3;
    int num = (int(d1)-int('0'))*100 + (int(d2)-int('0'))*10 + (int(d3)-int('0'));
    cout << "valor númerico: " << num << endl;
}
```

5. Desarrolle un programa que lea de teclado un numero entero de tres dígitos y lo almacene en una variable de tipo `int` (supondremos que la entrada de datos es correcta), y desglose el número leído en los tres dígitos (como caracteres) que lo componen y los almacene en tres variables de tipo `char`. Finalmente mostrará en pantalla el desglose de los dígitos. Por ejemplo, si lee el número `345`, deberá desglosarlo en los siguientes tres caracteres `'3'`, `'4'`, y `'5'`.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    int num;
    cout << "Introduzca un número entero: ";
    cin >> num;
    char d1 = char( ((num / 100) % 10) + int('0') );
    char d2 = char( ((num / 10) % 10) + int('0') );
    char d3 = char( ((num / 1) % 10) + int('0') );
    cout << "Digito 1: " << d1 << endl;
    cout << "Digito 2: " << d2 << endl;
    cout << "Digito 3: " << d3 << endl;
}
```

6. Codifique el siguiente programa, ejecútelo y analice el resultado obtenido.

```
#include <iostream>
using namespace std;
int main()
{
    bool ok = (3.0 * (0.1 / 3.0)) == ((3.0 * 0.1) / 3.0);
    cout << "Resultado de (3.0 * (0.1 / 3.0)) == ((3.0 * 0.1) / 3.0): "
         << boolalpha << ok << endl;
}
```

Solución

```
#include <iostream>
using namespace std;
int main()
{
    bool ok = (3.0 * (0.1 / 3.0)) == ((3.0 * 0.1) / 3.0);
    cout << "Resultado de (3.0 * (0.1 / 3.0)) == ((3.0 * 0.1) / 3.0): "
         << boolalpha << ok << endl;
}
//-------------------------------------------------------------------
// El tipo DOUBLE utiliza una representación INEXACTA y FINITA, por lo
// que se produce pérdida de precisión en las operaciones de coma
// flotante, de tal forma que podría suceder que dos valores que
// matemáticamente son iguales sean computacionalmente ligeramente
// diferentes (un valor decimal muy pequeño) y por lo tanto la
// comparación de igualdad (==) entre números reales no produzca
// los resultados esperados. Por lo tanto, los números reales nunca se
// deberían deberían comparar por igualdad, sino mediante una
// operación similar a: (abs(x-y) < 1e-9), que simplemente comprueba
// si dos números reales están los suficientemente cercanos.
//
// En este ejemplo concreto, el resultado matemático debería ser TRUE,
// pero al realizarse en el ordenador, donde el número de dígitos decimales
// es limitado, la operación 0.1/3.0 produce como resultado 0.033333 periódico
// que al multiplicarse por 3.0 da como resultado de la primera subexpresión
// 0.099999 periódico, mientras que en la segunda subexpresión, 3.0 * 0.1 da
```

```
// como resultado 0.3, que al dividirse entre 3.0 el resultado final es 0.1,
// que es diferente de 0.099999 periódico, por lo que el resultado de la
// comparación final es FALSE
```

7. Codifique el siguiente programa, que lee tres números reales desde el teclado, y posteriormente escribe `true` si el resultado de multiplicar los dos primeros números es igual al tercero (`a * b == c`), y escribe `false` en caso contrario.

```
#include <iostream>
using namespace std;
int main()
{
    double a, b, c;
    cout << "Introduce 3 numeros reales: ";
    cin >> a >> b >> c;
    bool cmp = (a * b == c);
    cout << a << " * " << b << " == " << c << " "
         << boolalpha << cmp
         << endl;
}
```

Ejecute dicho programa introduciendo como datos de entrada los siguientes números y analice los resultados obtenidos. Los siguientes resultados han sido realizados en una máquina de 32 bits, en una máquina de 64 bits, el resultado de la última ejecución es `true` (en vez de `false` como aparece en el enunciado).

- a: 1, b: 1, c: 1 ⇒ `true`
- a: 2, b: 2, c: 4 ⇒ `true`
- a: 3, b: 3, c: 9 ⇒ `true`
- a: 3, b: 0.1, c: 0.3 ⇒ `false`
- a: 2, b: 0.2, c: 0.4 ⇒ `true`
- a: 3, b: 0.3, c: 0.9 ⇒ `false`
- a: 0.1, b: 0.1, c: 0.01 ⇒ `false`
- a: 0.2, b: 0.2, c: 0.04 ⇒ `false`
- a: 0.3, b: 0.3, c: 0.09 ⇒ `false`

Solución

```
#include <iostream>
using namespace std;
int main()
{
    double a, b, c;
    cout << "Introduce 3 números reales: ";
    cin >> a >> b >> c;
    bool cmp = (a * b == c);
    cout << a << " * " << b << " == " << c << " "
         << boolalpha << cmp
         << endl;
}
//-----------------------------------------------------------------
// El tipo DOUBLE utiliza una representación INEXACTA y FINITA, por lo
// que se produce pérdida de precisión en las operaciones de coma
// flotante, de tal forma que podría suceder que dos valores que
// matemáticamente son iguales sean computacionalmente ligeramente
// diferentes (un valor decimal muy pequeño) y por lo tanto la
// comparación de igualdad (==) entre números reales no produzca
// los resultados esperados. Por lo tanto, los números reales nunca se
// deberían deberían comparar por igualdad, sino mediante una
// operación similar a: abs(x-y) < 1e-9
```

Tema 2: Conceptos Básicos de Programación

Práctica 2. Estructuras de Control. Selección

Laboratorio

1. Una empresa maneja códigos numéricos, donde cada código consta de cuatro dígitos:

   - El primer dígito representa a una provincia.
   - Los dos siguientes dígitos indican el número de la operación.
   - El último dígito es un dígito de control.

   Se desea desarrollar un programa que lea de teclado un número entero de cuatro dígitos (el código de provincia es distinto de cero), lo almacene en una variable de tipo entero (`int`), y realice las siguientes acciones:

   - Si el código numérico no tiene 4 dígitos, entonces escribe un mensaje de error.
   - Si el código númerico tiene 4 dígitos, entonces:
     - muestra en pantalla la información (provincia, operación y dígito de control) desglosada.
     - Calcula si el código es correcto comprobando si el valor del dígito de control coincide con el **resto de dividir** entre 10 el resultado de multiplicar el número de operación por el código de la provincia, y mostrará un mensaje adecuado.

Por ejemplo, para el número 32:	Por ejemplo, para el número 7362:	Por ejemplo, para el número 6257:

`Código erróneo`	`Provincia: 7`
	`Número de operación: 36`
	`Dígito de control: 2`
	`Comprobación: correcto`
		`Provincia: 6`
		`Número de operación: 25`
		`Dígito de control: 7`
		`Comprobación: error`

Solución

```
#include <iostream>
using namespace std;
//-------------------------------------
const int LIMITE = 10000;
const int BASE_PROV = 1000;
const int BASE_OP = 10;
//-------------------------------------
int main()
{
    cout << "Introduzca el código numérico de 4 dígitos: ";
    int codigo;
    cin >> codigo;
    if ((codigo >= BASE_PROV) && (codigo < LIMITE)) {
        int provincia = codigo / BASE_PROV;
        int operacion = (codigo % BASE_PROV) / BASE_OP;
        int control = (codigo % BASE_PROV) % BASE_OP;
        //-------------------------------
        cout << "Provincia: " << provincia << endl;
        cout << "Número de operación: " << operacion << endl;
        cout << "Dígito de control: " << control << endl;
        if (control == (operacion * provincia % 10 )) {
            cout << "Comprobación: correcto"<< endl;
        } else {
            cout << "Comprobación: error" << endl;
        }
    } else {
        cout << "Código erróneo" << endl;
    }
    //---------------------------------
}
```

2. Diseñe un programa que lea de teclado cuatro números enteros y escriba en pantalla el mayor de los cuatro.

Solución

```
#include <iostream>
using namespace std;
int main ()
{
   int a, b, c, d;
   cout << "Introduzca cuatro números: ";
   cin >> a >> b >> c >> d;
   int mayor = a;
   if (b > mayor) {
      mayor = b;
   }
   if (c > mayor) {
      mayor = c;
   }
   if (d > mayor) {
      mayor = d;
   }
   cout << "Mayor: " << mayor << endl;
}
```

3. El recibo de la electricidad se elabora de la siguiente forma para una determinada cantidad de Kw consumidos:

   - 1€ de gastos fijos.
   - 0.50€/Kw para los primeros 100 Kw consumidos.
   - 0.35€/Kw para los siguientes 150 Kw consumidos.
   - 0.25€/Kw para el resto de Kw consumidos.

   Elabore un programa que lea de teclado dos números, que representan los dos últimos valores del contador de electricidad (al restarlos obtendremos el consumo en Kw) y calcule e imprima en pantalla el importe total a pagar en función del consumo realizado.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
const double GASTOS_FIJOS = 1.0;
const double PRECIO_1 = 0.50;
const double PRECIO_2 = 0.35;
const double PRECIO_3 = 0.25;
const int UMBRAL_1 = 100;
const int UMBRAL_2 = 150;
//-----------------------------------------
int main()
{
    cout << "Introduzca los dos últimos valores del contador: ";
    int cont_1, cont_2;
    cin >> cont_1 >> cont_2;
    if (cont_1 > cont_2) {
        int aux = cont_1;
        cont_1 = cont_2;
        cont_2 = aux;
    }
    int consumo = cont_2 - cont_1;
    double importe = GASTOS_FIJOS;
    if (consumo <= UMBRAL_1) {
        importe += consumo * PRECIO_1;
    } else {
        importe += UMBRAL_1 * PRECIO_1;
        int resto = consumo - UMBRAL_1;
        if (resto <= UMBRAL_2) {
            importe += resto * PRECIO_2;
        } else {
            importe += UMBRAL_2 * PRECIO_2;
            resto -= UMBRAL_2;
            importe += resto * PRECIO_3;
        }
    }
```

```
    cout << "Consumo: " << consumo << " Kw. "
         << "Importe: " << importe << " Eur"
         << endl;
}
```

4. Codifique un programa que se comporte como una calculadora simple. Para ello deberá tener las siguientes características:

   - Solo efectuará operaciones con dos operandos (binarias).
   - Operaciones permitidas: (+,-,*,/).
   - Se trabajará con operandos enteros.
   - Leerá en primer lugar la operación a realizar, y a continuación los dos operandos numéricos. Si el operador no se corresponde con alguno de los indicados se emitirá un mensaje de error, en otro caso mostrará el resultado de la operación, debiendo tener cuidado con el caso de división por cero. Ejemplo:

     ```
     Operacion : *
     Operando 1 : 24
     Operando 2 : 3
     Resultado : 72
     ```

Solución

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Operación: ";
    char op;
    cin >> op;
    if (!(op == '+' || op == '-' || op == '*' || op == '/')) {
        cout << "ERROR: Operación no válida" << endl;
    } else {
        cout << "Operando 1: ";
        int op1;
        cin >> op1;
        cout << "Operando 2: ";
        int op2;
        cin >> op2;
        int res = 0;
        switch (op) {
        case '+':
            res = op1 + op2;
            break;
        case '-':
            res = op1 - op2;
            break;
        case '*':
            res = op1 * op2;
            break;
        case '/':
            if (op2 != 0) {
                res = op1 / op2;
            } else {
                res = 0;
                cout << "Error, división por cero"<<endl;
            }
            break;
        }
        cout << "Resultado: " << res << endl;
    }
}
```

Tema 2: Conceptos Básicos de Programación

Práctica 2. Estructuras de Control. Selección

Ejercicios Complementarios

1. Codifique un programa que lea de teclado dos números enteros (`x` e `y`) y un carácter (`c`), y escriba `true` si cumplen las siguientes propiedades, y `false` en caso contrario:

   a) `x` $\in \{3, 4, 5, 6, 7\}$

   b) `x` $\in \{1, 2, 3, 7, 8, 9\}$

   c) `x` $\in \{1, 3, 5, 7, 9\}$

   d) `x` $\in \{2, 5, 6, 7, 8, 9\}$

   e) `x` $\in \{3, 4, 6, 8, 9\}$, `y` $\in \{6, 7, 8, 3\}$

   f) Ni `x` ni `y` sean mayores que 10

   g) `x` no sea múltiplo de `y`

   h) `c` es una letra mayúscula

   i) `c` es una letra

   j) `c` es un alfanumérico (letra o dígito)

Solución

```
#include <iostream>
using namespace std;
int main()
{
    int x, y;
    char c;
    cout << "Introduzca dos numeros naturales: ";
    cin >> x >> y;
    cout << "Introduzca un caracter: ";
    cin >> c;
    //-------------------------------------
    bool prop_a = (x >= 3 && x <= 7);
    bool prop_b = (x >= 1 && x <= 3)||(x >= 7 && x <= 9);
    bool prop_c = (x >= 1 && x <= 9)&&(x % 2 == 1);
    bool prop_d = (x == 2)||(x >= 5 && x <= 9);
    bool prop_e = ((x >= 3 && x <= 9 && x != 5 && x != 7)
                  &&((y >= 6 && y <= 8) || y == 3));
    bool prop_f = (x <= 10 && y <= 10);
    bool prop_g = ! (y != 0 && x % y == 0);
    bool prop_h = (c >= 'A' && c <= 'Z');
    bool prop_i = (c >= 'A' && c <= 'Z')||(c >= 'a' && c <= 'z');
    bool prop_j = ((c >= 'A' && c <= 'Z')
                  ||(c >= 'a' && c <= 'z')
                  ||(c >= '0' && c <= '9'));
    //-------------------------------------
    cout << boolalpha;
    cout << "(a) "<<x<<" pertenece a { 3, 4, 5, 6, 7 }: " << prop_a << endl;
    cout << "(b) "<<x<<" pertenece a { 1, 2, 3, 7, 8, 9 }: " << prop_b << endl;
    cout << "(c) "<<x<<" pertenece a { 1, 3, 5, 7, 9 }: " << prop_c << endl;
    cout << "(d) "<<x<<" pertenece a { 2, 5, 6, 7, 8, 9 }: " << prop_d << endl;
    cout << "(e) "<<x<<" pertenece a { 3, 4, 6, 8, 9 }, "
         <<y<<" pertenece a { 6, 7, 8, 3 }: " << prop_e << endl;
    cout << "(f) Ni "<<x<<" ni "<<y<<" sean mayores que 10: " << prop_f <<endl;
    cout << "(g) "<<x<<" no sea multiplo de "<<y<<": " << prop_g << endl;
    cout << "(h) '"<<c<<"' es una letra mayuscula: " << prop_h << endl;
    cout << "(i) '"<<c<<"' es una letra: " << prop_i << endl;
    cout << "(j) '"<<c<<"' es un alfanumerico: " << prop_j << endl;
}
```

2. Diseñe un programa que lea de teclado un número real (n) comprendido entre 0 y 10 e imprima la nota asociada según el siguiente esquema:

$$
\begin{array}{ccl}
n = 10 & \longrightarrow & \text{Matrícula de Honor} \\
9 \leq n < 10 & \longrightarrow & \text{Sobresaliente} \\
7 \leq n < 9 & \longrightarrow & \text{Notable} \\
5 \leq n < 7 & \longrightarrow & \text{Aprobado} \\
0 \leq n < 5 & \longrightarrow & \text{Suspenso} \\
\text{En otro caso} & \longrightarrow & \text{Error}
\end{array}
$$

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    double nota;
    cout << "Introduzca la Nota: ";
    cin >> nota;
    if ( ! ((nota >= 0.0) && (nota <= 10.0))) {
       cout << "Error: 0 <= n <= 10" << endl;
    } else if (nota >= 10.0 - 1e-9) {
       cout << "Matrícula de Honor" << endl;
    } else if (nota >= 9.0) {
       cout << "Sobresaliente" << endl;
    } else if (nota >= 7.0) {
       cout << "Notable" << endl;
    } else if (nota >= 5.0) {
       cout << "Aprobado" << endl;
    } else {
       cout << "Suspenso" << endl;
    }
}
```

Tema 2: Conceptos Básicos de Programación

Práctica 3. Estructuras de Control. Iteración

Laboratorio

1. Diseñe un programa que lea un número entero de teclado y escriba una línea con tantos asteriscos (*) como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
****
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cout << SIMBOLO;
    }
    cout << endl;
}
```

2. Diseñe un programa que lea un número entero de teclado y escriba un cuadrado (relleno) con tantos asteriscos (*) de lado como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
****
****
****
****
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n; ++i) {
            cout << SIMBOLO;
        }
        cout << endl;
    }
}
```

3. Diseñe un programa que lea un número entero de teclado y escriba un cuadrado (hueco) con tantos asteriscos (*) de lado como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
****
*  *
*  *
****
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
```

```
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n; ++i) {
            if ((j==0)||(j==n-1)||(i==0)||(i==n-1)) {
                cout << SIMBOLO;
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}
```

4. Diseñe un programa que lea un número entero de teclado y escriba un triángulo rectángulo (relleno) con tantos asteriscos (*) de base y altura como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
*
**
***
****
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i <= j; ++i) {
            cout << SIMBOLO;
        }
        cout << endl;
    }
}
```

5. Calculador repetitivo. Modifique el problema de la calculadora anterior para que se repita un número indefinido de veces. El calculador dejará de trabajar cuando se introduzca como código de operación el símbolo &. Ejemplo:

```
Operacion : *
Operando 1 : 13
Operando 2 : 10
Resultado : 130
Operacion : u
ERROR!!!!
Operacion : +
Operando 1 : 12
Operando 2 : 3
Resultado : 15
Operacion : &
FIN DEL PROGRAMA
```

Solución

```
#include <iostream>
using namespace std;
int main()
{
    char op;
    cout << "Operación: ";
    cin >> op;
    while (op != '&') {
```

```
        if (!(op == '+' || op == '-' || op == '*' || op == '/')) {
            cout << "ERROR: Operación no válida" << endl;
        } else {
            cout << "Operando 1: ";
            int op1;
            cin >> op1;
            cout << "Operando 2: ";
            int op2;
            cin >> op2;
            int res = 0;
            switch (op) {
            case '+':
                res = op1 + op2;
                break;
            case '-':
                res = op1 - op2;
                break;
            case '*':
                res = op1 * op2;
                break;
            case '/':
                if (op2 != 0) {
                    res = op1 / op2;
                } else {
                    res = 0;
                    cout << "Error, división por cero"<<endl;
                }
                break;
            }
            cout << "Resultado: " << res << endl;
        }
        cout << "Operación: ";
        cin >> op;
    }
}
```

6. Diseñe un programa que lea un número entero de teclado y escriba un mensaje adecuado indicando si el número leído es o no *primo*. Un número mayor que uno (1) es primo si sólo es divisible por él mismo y por la unidad (1).

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    bool es_primo = false;
    int num, divisor;
    cout << "Introduzca un número natural: ";
    cin >> num;
    if (num > 1) {
        divisor = 2;
        while ((num % divisor) != 0) {
            ++divisor;
        }
        es_primo = (divisor == num);
    }
    if (es_primo) {
        cout << "El número " << num << " es primo" << endl;
    } else {
        cout << "El número " << num << " no es primo" << endl;
    }
}
```

7. Diseña un programa que calcule e imprima en pantalla los N primeros términos de la sucesión de Fibonacci, siendo N un número que se introduce por teclado. Los dos primeros números de esta serie son el cero y el uno, y a partir de éstos, cada número de la secuencia se calcula realizando la suma de los dos anteriores. Ej: $0, 1, 1, 2, 3, 5, 8, 13, 21, \cdots$

$$f_n = f_{n-1} + f_{n-2}$$

Solución

```
#include <iostream>
using namespace std;
//-------------------------------------
const int FIB_0 = 0;
const int FIB_1 = 1;
//-------------------------------------
int main()
{
    int n;
    cout << "Introduzca N: ";
    cin >> n;
    //-------------------------------------
    if (n > 0) {
        cout << FIB_0 << ", ";
    }
    if (n > 1) {
        cout << FIB_1 << ", ";
    }
    int fib_n1 = FIB_0;
    int fib_n = FIB_1;
    for (int i = 2; i < n; ++i) {
        int fib_n2 = fib_n1;
        fib_n1 = fib_n;
        fib_n = fib_n1 + fib_n2;
        cout << fib_n << ", ";
    }
    cout << endl;
    //-------------------------------------
}
```

8. Desarrolle un programa que lea una secuencia de números enteros terminada en **0**, y que encuentre y escriba en la pantalla la posición de la primera y de la última ocurrencia del número **12** dentro de la secuencia. Si el número **12** no está en la secuencia, el programa debería escribir **0**. Por ejemplo, si el octavo número de la lista es el único **12**, entonces **8** sería la primera y la última posición de las ocurrencias de **12**.

Solución

```
#include <iostream>
using namespace std;
//-------------------------------------
const int NUMERO = 12;
//-------------------------------------
int main()
{
    int pos = 0;
    int pri_oc = 0;
    int ult_oc = 0;
    int n;
    cout << "Introduzca una secuencia de números terminada en cero (0): ";
    cin >> n;
    while (n != 0) {
        ++pos;
        if (n == NUMERO) {
            if (pri_oc == 0) {
                pri_oc = pos;
            }
            ult_oc = pos;
        }
        cin >> n;
    }
    cout << "Primera ocurrencia: " << pri_oc << endl;
    cout << "Ultima ocurrencia: "  << ult_oc << endl;
}
```

Tema 2: Conceptos Básicos de Programación

Práctica 3. Estructuras de Control. Iteración

Ejercicios Complementarios

1. Diseñe un programa que lea de teclado dos números enteros y escriba en pantalla el resultado de multiplicar ambos números, teniendo en cuenta que no se permite utilizar la operación de multiplicar (*), por lo que se deberá realizar mediante sumas sucesivas. Por ejemplo, $2 \times 3 = 2 + 2 + 2$

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    int m, n;
    cout << "Introduce dos números naturales: ";
    cin >> m >> n;
    int total = 0;
    for (int i = 0; i < n; ++i) {
        total = total + m;
    }
    cout << "Resultado: " << total << endl;
}
```

2. Diseñe un programa que lea de teclado dos números enteros (x e y) y escriba en pantalla el resultado de calcular la potencia de x elevado a y (x^y), mediante multiplicaciones sucesivas. Por ejemplo, $2^3 = 2 \times 2 \times 2$

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    int b, e;
    cout << "Introduce la base y el exponente: ";
    cin >> b >> e;
    int total = 1;
    for (int i = 0; i < e; ++i) {
        total = total * b;
    }
    cout << "Resultado: " << total << endl;
}
```

3. Diseñe un programa que lea de teclado un número entero (x) y escriba en pantalla el resultado de calcular el factorial de x. Por ejemplo, $4! = 1 \times 2 \times 3 \times 4$

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    int x;
    cout << "Introduce un número natural: ";
    cin >> x;
    int total = 1;
    for (int i = 2; i <= x; ++i) {
        total = total * i;
    }
    cout << "Resultado: " << total << endl;
}
```

4. Diseñe un programa que lea de teclado dos números enteros y escriba en pantalla el cociente y resto de la división de ambos números, teniendo en cuenta que no se permite utilizar las operaciones de división y resto (/, %), por lo que se deberá realizar mediante restas sucesivas. Por ejemplo,

$$7 \div 2 \Longrightarrow 7 - 2 = 5 \ ; \ 5 - 2 = 3 \ ; \ 3 - 2 = 1 \Longrightarrow \mathit{Cociente} = 3 \ ; \ \mathit{Resto} = 1$$

Solución

```
#include <iostream>
using namespace std;
int main ()
{
    int dividendo, divisor;
    cout << "Introduce dividendo y divisor: ";
    cin >> dividendo >> divisor;
    if (divisor == 0) {
        cout << "El divisor no puede ser cero" << endl;
    } else {
        int resto = dividendo;
        int cociente = 0;
        while (resto >= divisor) {
            resto -= divisor;
            ++cociente;
        }
        cout << "Cociente: " << cociente << " Resto: " << resto << endl;
    }
}
```

5. Diseñe un programa que lea un número entero de teclado y escriba un triángulo rectángulo (hueco) con tantos asteriscos (*) de base y altura como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
*
**
* *
****
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i <= j; ++i) {
            if ((j==0)||(j==n-1)||(i==0)||(i==j)) {
                cout << SIMBOLO;
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}
```

6. Diseñe un programa que lea un número entero de teclado y escriba un triángulo (relleno) con tantos asteriscos (*) de altura como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
   *
  ***
 *****
*******
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
```

```
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
        }
        for (int i = 0; i < 2*j+1; ++i) {
            cout << SIMBOLO;
        }
        cout << endl;
    }
}
```

7. Diseñe un programa que lea un número entero de teclado y escriba un triángulo (hueco) con tantos asteriscos (*) de altura como indique el número leído. Por ejemplo, para un número leído con valor 4, escribirá:

```
   *
  * *
 *   *
*******
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
        }
        for (int i = 0; i < 2*j+1; ++i) {
            if ((j==0)||(j==n-1)||(i==0)||(i==2*j)) {
                cout << SIMBOLO;
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}
```

8. Diseñe un programa que lea un número entero de teclado y escriba un rombo (relleno) con asteriscos (*), según el siguiente ejemplo para un número leído con valor 4:

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    //-------------------------------
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
```

```
        }
        for (int i = 0; i < 2*j+1; ++i) {
            cout << SIMBOLO;
        }
        cout << endl;
    }
    //-------------------------------
    for (int j = n-2; j >= 0; --j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
        }
        for (int i = 0; i < 2*j+1; ++i) {
            cout << SIMBOLO;
        }
        cout << endl;
    }
    //-------------------------------
}
```

9. Diseñe un programa que lea un número entero de teclado y escriba un rombo (hueco) con asteriscos (*), según el siguiente ejemplo para un número leído con valor 4:

```
   *
  * *
 *   *
*     *
 *   *
  * *
   *
```

Solución

```
#include <iostream>
using namespace std;
const char SIMBOLO = '*';
int main()
{
    int n;
    cout << "Introduzca un número: ";
    cin >> n;
    //-------------------------------
    for (int j = 0; j < n; ++j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
        }
        for (int i = 0; i < 2*j+1; ++i) {
            if ((j==0)||(i==0)||(i==2*j)) {
                cout << SIMBOLO;
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
    //-------------------------------
    for (int j = n-2; j >= 0; --j) {
        for (int i = 0; i < n-j-1; ++i) {
            cout << " ";
        }
        for (int i = 0; i < 2*j+1; ++i) {
            if ((j==0)||(i==0)||(i==2*j)) {
                cout << SIMBOLO;
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
    //-------------------------------
}
```

10. Desarrolle un programa que encuentre el mayor, el menor y la media aritmética de una secuencia números enteros leídos por el teclado donde el número cero **(0)** indica el final de la secuencia. Nota: El número cero no forma parte de la secuencia.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
const int FIN = 0;
//-----------------------------------------
int main()
{
    int num, mayor, menor;
    int suma = 0;
    int cnt = 0;
    cout << "Introduzca una secuencia de números (separados por espacios)" << endl;
    cout << "terminada en cero: " << endl;
    cin >> num;
    mayor = num;     /* INT_MIN */
    menor = num;     /* INT_MAX */
    while (num != FIN) {
        ++cnt;
        suma += num;
        if (num > mayor) {
            mayor = num;
        } else if (num < menor) {
            menor = num;
        }
        cin >> num;
    }
    if (cnt == 0) {
        cout << "Secuencia vacía" << endl;
    } else {
        double media = double(suma) / double(n);
        cout << "Mayor: " << mayor << endl;
        cout << "Menor: " << menor << endl;
        cout << "Media: " << media << endl;
    }
}
```

11. Desarrolle un programa que lea de teclado una secuencia de caracteres (carácter a carácter) hasta leer un punto (`'.'`), y que al final muestre como salida el número de comas (`','`) encontradas, y el número de caracteres leídos. Nota: considere el concepto de *búffer de entrada*.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    int cnt_ct = 0; // cuenta de caracteres
    int cnt_cm = 0; // cuenta de comas
    cout << "Introduzca una frase hasta '.': ";
    char c;
    cin.get(c);      // lee un carácter
    while (c != '.') {
        ++cnt_ct;
        if (c == ',') {
            ++cnt_cm;
        }
        cin.get(c);      // lee un carácter
    }
    cout << "Cuenta de caracteres: " << cnt_ct
         << ". Cuenta de comas: " << cnt_cm << endl;
}
```

12. Desarrolle un programa que determine si la secuencia `"abc"` aparece en una sucesión de caracteres leídos de teclado cuyo final viene dado por un punto (`'.'`). Nota: considere el concepto de *búffer de entrada*.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    bool ok = false;
    char ant2 = ' ';
    char ant1 = ' ';
    char act;
    cout << "Introduzca una frase hasta '.': ";
    cin.get(act);
    while (act != '.') {
        if (ant2 == 'a' && ant1 == 'b' && act == 'c') {
            ok = true;
        }
        ant2 = ant1;
        ant1 = act;
        cin.get(act);
    }
    cout << "La secuencia 'abc' ";
    if (ok) {
        cout << "SI";
    } else {
        cout << "NO";
    }
    cout << " aparece en la frase de entrada" << endl;
}
```

13. Desarrolle un programa para el siguiente juego: el usuario introduce un número como límite inferior, un número como límite superior y piensa un número dentro de ese rango. El objetivo del programa es acertar el número pensado por el usuario. Para ello el programa propone un número y el usuario responde con >, < o =, que significan que el número pensado es mayor, menor o igual respectivamente al numero propuesto por el programa. Si la respuesta es =, entonces el programa terminará satisfactoriamente, en otro caso, si la respuesta es > o < el programa propondrá otro número hasta que finalmente acierte el número pensado por el usuario.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    int lim_inf, lim_sup;
    cout << "Introduce el límite inferior: ";
    cin >> lim_inf;
    cout << "Introduce el límite superior: ";
    cin >> lim_sup;
    if (lim_inf > lim_sup) {
        cout << "Error: límites erróneos" << endl;
    } else {
        cout << "Piense un número dentro del rango ["
             << lim_inf << "-" << lim_sup << "] y pulse ENTER ";
        char cod;
        cin.get(cod); // elimina ENTER del BÚFFER_ENTRADA de lim_sup
        cin.get(cod); // elimina ENTER del BÚFFER_ENTRADA de pausa
        int num;
        do {
            num = (lim_inf + lim_sup) / 2;
            cout << ">> ¿Es el número " << num << " ? ";
            cin >> cod;
            switch (cod) {
            case '<':
                lim_sup = num - 1;
                break;
            case '>':
                lim_inf = num + 1;
                break;
            case '=':
                break;
            default:
                cout << "Código Erróneo"<< endl;
```

```
            }
        } while ((cod != '=')&&(lim_inf <= lim_sup));
        if (cod == '=') {
            cout << "El número pensado es el " << num << endl;
        } else {
            cout << "Has introducido datos erróneos" << endl;
        }
    }
}
```

Tema 2: Conceptos Básicos de Programación

Ejercicios de Autoevaluación

1. Considere la siguiente propiedad descubierta por Nicómano de Gerasa:

   - *Sumando el primer impar, se obtiene el primer cubo;*
   - *Sumando los dos siguientes impares, se obtiene el segundo cubo;*
   - *Sumando los tres siguientes, se obtiene el tercer cubo,*
   - *Etc.*

   Comprobémoslo:

   $$\begin{array}{lcll} 1 & = & 1 & = 1^3 \\ 3+5 & = & 8 & = 2^3 \\ 7+9+11 & = & 27 & = 3^3 \\ 13+15+17+19 & = & 64 & = 4^3 \end{array}$$

   Desarrolle, un programa que lea de teclado un número natural n (> 0), si el número leído anteriormente es menor que 1, entonces entrará en un bucle que muestre un mensaje de error y lea un nuevo número hasta que el número leído n sea mayor que cero.

   Finalmente, el programa mostrará por pantalla todos los cubos, desde 1^3 hasta n^3, tal y como se ha mostrado en el ejemplo anterior para $n = 4$, es decir, mostrando la suma de los números impares que lo generan siguiendo el procedimiento propuesto por Nicómano de Gerasa. Por ejemplo:

   ```
   Introduce un número: 0
   Error: Introduce un número: -1
   Error: Introduce un número: 4

   1 = 1
   3 + 5 = 8
   7 + 9 + 11 = 27
   13 + 15 + 17 + 19 = 64
   ```

Tema 3: Diseño Descendente. Subprogramas

Práctica 4. Subprogramas (I)

Laboratorio

1. Escribe un programa que lea un número entero N por teclado y dibuje un triángulo de asteriscos con altura N. Se deberá definir, al menos, tres subprogramas además del `main`. Por ejemplo si $N = 4$ debería dibujarse:

```
   *
  ***
 *****
*******
```

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
const char SIMBOLO = '*';
//-----------------------------------------
void esc_caracter(int n, char simb)
{
    for (int i = 0; i < n; ++i) {
        cout << simb;
    }
}
//-----------------------------------------
void esc_fila(int f, int nf)
{
    assert(f < nf);
    esc_caracter(nf-f-1, ' ');
    esc_caracter(2*f+1, SIMBOLO);
    cout << endl;
}
//-----------------------------------------
void esc_triangulo(int nf)
{
    for (int f = 0; f < nf; ++f) {
        esc_fila(f, nf);
    }
}
//-----------------------------------------
int main()
{
    cout << "Introduzca numero de filas: ";
    int n_filas;
    cin >> n_filas;
    esc_triangulo(n_filas);
}
```

2. Escribe un programa que lea de teclado dos números enteros y continue leyendo los números mientras el primero sea mayor o igual al segundo. El programa deberá mostrar el resultado de sumar todos los números pares que se encuentren dentro del intervalo especificado por los dos números leídos al principio del programa. Se deberá definir, al menos, un procedimiento para la lectura correcta del intervalo y una función para calcular la suma de los números dentro del intervalo.

Solución

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

void leer_intervalo(int& menor, int& mayor)
{
    do {
        cout << "Introduce el numero menor del intervalo: ";
        cin >> menor;
        cout << "Introduce el numero mayor del intervalo: ";
        cin >> mayor;
    } while (menor >= mayor);
```

```
}

int sumar_pares(int menor, int mayor)
{
    int suma = 0;
    for (int i = menor; i <= mayor; ++i) {
        if (i % 2 == 0) {
            suma += i;
        }
    }
    return suma;
}

int main()
{
    int menor, mayor;
    leer_intervalo(menor, mayor);
    cout << "Suma de intervalo [ " << menor << ", " << mayor << " ]: "
         << sumar_pares(menor, mayor) << endl;
}
```

3. Escribe un programa que lea de teclado el número de filas (mayor que cero y menor que diez, estará en un ciclo mientras el número leído sea incorrecto), y muestre en pantalla una figura como el siguiente ejemplo para un valor de 4.

```
0 1 2 3
1 2 3 0
2 3 0 1
3 0 1 2
```

El programa deberá definir y utilizar el siguiente subprograma:

```
void incremento_circular(int& x, int max);
```

que recibe un primer parámetro de entrada/salida con un valor entero y lo modifica incrementándolo en uno. Sin embargo, si el resultado de incrementar el valor del primer parámetro es mayor o igual al valor del segundo parámetro, entonces el valor del primer parámetro se inicializa a cero.

Solución

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

void leer_nfilas(int& nfilas)
{
    do {
        cout << "Introduce el número de filas: ";
        cin >> nfilas;
    } while ( ! (0 < nfilas && nfilas < 10));
}

void incremento_circular(int& valor, int limite)
{
    // equivalente: valor = (valor + 1) % limite;
    ++valor;
    if (valor >= limite) {
        valor = 0;
    }
}

void mostrar_fila(int f, int nfilas)
{
    int valor = f;
    for (int i = 0; i < nfilas; ++i) {
        cout << valor << " ";
        incremento_circular(valor, nfilas);
    }
    cout << endl;
}
void mostrar_cuadro(int nfilas)
{
    for (int f = 0; f < nfilas; ++f) {
```

```
        mostrar_fila(f, nfilas);
    }
}

int main()
{
    int nfilas;
    leer_nfilas(nfilas);
    mostrar_cuadro(nfilas);
}
```

4. Escribe un programa que calcule el valor de `S` para un número real X $(0 \leq X \leq 1)$ dado por teclado, utilizando la serie de Taylor para calcular el valor de e^x:

$$S = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \frac{X^4}{4!} + \cdots$$

Nota: No se añadirán más sumandos cuando se calcule uno con valor menor que 0.0001.

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
const int LIMITE_FACTORIAL_UNSIGNED = 14;
const double LIMITE = 1e-4;
//-----------------------------------------
double potencia(double base, int exp)
{
    assert(base >= 0 && base <= 1);
    double res = 1;
    for (int i = 0; i < exp; ++i) {
        res *= base;
    }
    return res;
}
//-----------------------------------------
int factorial(int n)
{
    assert(n < LIMITE_FACTORIAL_UNSIGNED);
    int res = 1;
    for (int i = 2; i <= n; ++i) {
        res *= i;
    }
    return res;
}
//-----------------------------------------
double termino(double x, int i)
{
    return potencia(x, i) / double(factorial(i));
}
//-----------------------------------------
double serie(double x)
{
    assert(x >= 0 && x <= 1);
    int i = 0;
    double res = 1;
    double term;
    do {
        ++i;
        term = termino(x, i);
        res += term;
    } while (term >= LIMITE);
    return res;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca el valor de X [0..1]: ";
    double x;
    cin >> x;
    if (! (x >= 0 && x <= 1)) {
```

```
        cout << "Error. Valor de X fuera de rango" << endl;
    } else {
        cout << "Serie: " << serie(x) << endl;
    }
}
```

5. Diseña un programa que calcule e imprima en pantalla los N primeros números primos, siendo N un número que se introduce por teclado.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
bool es_primo(int n)
{
    int i = 2;
    while ((i <= n / 2)&&(n % i != 0)) {
        ++i;
    }
    return (i == n / 2 + 1);
}
//-----------------------------------------
void imprimir_primos(int n)
{
    int cnt = 0;
    int i = 0;
    while (cnt < n) {
        if (es_primo(i)) {
            ++cnt;
            cout << i << ", ";
        }
        ++i;
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    int n;
    cout << "Introduzca N: ";
    cin >> n;
    imprimir_primos(n);
}
```

6. Desarrolle un programa que lea un número **N** por teclado mayor o igual a cero y calcule e imprima el n-ésimo número de la serie de Fibonacci, comenzando la cuenta en cero. Los dos primeros números de esta serie son el cero y el uno, y a partir de éstos, cada número de la secuencia se calcula realizando la suma de los dos anteriores. Ej: $0, 1, 1, 2, 3, 5, 8, 13, 21, \cdots$

$$f_n = f_{n-1} + f_{n-2}$$

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
const int FIB_0 = 0;
const int FIB_1 = 1;
//-----------------------------------------
int fibonacci(int n)
{
    assert(n >= 0);
    int fib_n;
    if (n == 0) {
        fib_n = FIB_0;
    } else {
        int fib_n1 = FIB_0;
        fib_n = FIB_1;
```

```
        for (int i = 2; i <= n; ++i) {
            int fib_n2 = fib_n1;
            fib_n1 = fib_n;
            fib_n = fib_n1 + fib_n2;
        }
    }
    return fib_n;
}
//-------------------------------------
void leer(int& n)
{
    do {
        cout << "Introduzca N: ";
        cin >> n;
    } while (n < 0);
}
//-------------------------------------
int main()
{
    int n;
    leer(n);
    cout << "fibonacci(" << n << ") = " << fibonacci(n) << endl;;
}
//-------------------------------------
```

Tema 3: Diseño Descendente. Subprogramas

Práctica 4. Subprogramas (I)

Ejercicios Complementarios

1. Dadas las siguientes declaraciones en un determinado programa:

```
// - Prototipos --
bool uno (int x, int y);
void dos (int& x, int y);
int tres (int x);
// - Principal ----
int main ()
{
   int a, b, c;
   bool fin;
}
```

¿ Cuáles de las siguientes llamadas a subprogramas en el cuerpo del programa principal son válidas ?

a) `if (uno(a,b)) {/*...*/}`

b) `dos(a, b + 3);`

c) `fin = uno(c, 5);`

d) `fin = dos(c, 5);`

e) `dos(a,tres(a));`

f) `dos(tres(b),c);`

g) `if (tres(a)) {/*...*/}`

h) `b = tres(dos(a,5));`

i) `dos(4, c);`

Solución

```
// Cuales de las siguientes llamadas a subprogramas en el
// cuerpo del programa principal son validas ?
//
//-------------------
// (a) if (uno(a,b)) {/*...*/}
//     Es correcta, ya que [uno] es una función que devuelve [bool],
//     por lo que puede ser utilizada en la condicion de la sentencia
//     [if], y recibe como parametros por valor dos variables [int]
//-------------------
// (b) dos(a, b + 3);
//     Es correcta, ya que [dos] es un procedimiento, por lo que debe
//     ser utilizada como sentencia independiente, recibe como primer
//     parametro por referencia una variable [int], y como segundo
//     parametro por valor una expresión de tipo [int]
//-------------------
// (c) fin = uno(c, 5);
//     Es correcta, ya que [uno] es una función que devuelve [bool],
//     por lo que puede ser utilizada en la asignacion a una variable
//     de tipo [bool], y recibe como parametros por valor una variable
//     y una constante, ambos de tipo [int]
//-------------------
// (d) fin = dos(c, 5);
//     Es erronea, ya que [dos] es un procedimiento, por lo que no puede
//     ser utilizada en la asignacion a ninguna variable.
//-------------------
// (e) dos(a,tres(a));
//     Es correcta, ya que [dos] es un procedimiento, por lo que debe
//     ser utilizada como sentencia independiente, recibe como primer
//     parametro por referencia una variable [int], y como segundo
//     parametro por valor el valor devuelto por la funcion [tres] que
//     es de tipo [int], y la llamada a la funcion [tres] es correcta
//     porque se utiliza su valor [int] como parametro a la llamada a
//     [dos], y recibe como parametro por valor una variable de tipo
//     [int]
//-------------------
// (f) dos(tres(b),c);
//     Es erronea, ya que [dos] es un procedimiento, por lo que debe
//     ser utilizada como sentencia independiente, pero recibe como primer
//     parametro por referencia el valor devuelto por la funcion [tres],
//     que no es adecuada para el paso por referencia (solo es adecuada
//     una variable para el paso por referencia).
//-------------------
```

```
// (g) if (tres(a)) {/*...*/}
//     Es erronea, ya que [tres] es una funcion que devuelve un valor
//     [int], que no es adecuado como condicion de la sentencia [if]
//-------------------
// (h) b = tres(dos(a,5));
//     Es erronea, ya que [tres] es una funcion que devuelve un valor
//     [int], que no es adecuado como asignacion de una variable
//     [bool], ademas recibe como parametro por valor el valor devuelto
//     por el procedimiento [dos] que no devuelve ningun valor.
//-------------------
// (i) dos(4, c);
//     Es erronea, ya que [dos] es un procedimiento, por lo que debe
//     ser utilizada como sentencia independiente, pero recibe como primer
//     parametro por referencia una constante [int], que no es
//     adecuada para el paso por referencia (solo es adecuada una variable
//     para el paso por referencia).
```

2. Escribe un programa que imprima una pirámide de dígitos como la de la figura, tomando como entrada el número de filas de la misma (se supone menor de 10).

```
    1
   121
  12321
 1234321
123454321
```

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
const int MAX_FILAS = 10;
//-----------------------------------------
void esc_caracter(int n, char simb)
{
    for (int i = 0; i < n; ++i) {
        cout << simb;
    }
}
//-----------------------------------------
void esc_ascendente(int n)
{
    for (int i = 1; i <= n; ++i) {
        cout << i;
    }
}
//-----------------------------------------
void esc_descendente(int n)
{
    for (int i = n; i >= 1; --i) {
        cout << i;
    }
}
//-----------------------------------------
void esc_fila(int f, int nf)
{
    assert(f <= nf);
    esc_caracter(nf-f, ' ');
    esc_ascendente(f);
    esc_descendente(f-1);
    cout << endl;
}
//-----------------------------------------
void esc_triangulo(int nf)
{
    for (int f = 1; f <= nf; ++f) {
        esc_fila(f, nf);
    }
}
//-----------------------------------------
int main()
{
```

```
    cout << "Introduzca numero de filas: ";
    int n_filas;
    cin >> n_filas;
    if (n_filas < MAX_FILAS) {
        esc_triangulo(n_filas);
    }
}
```

3. Diseña un programa que lea de teclado un número entero n mayor que cero y muestre las n primeras filas del siguiente triángulo.

```
          1
         232
        34543
       4567654
      567898765
     67890109876
    7890123210987
   890123454321098
  90123456765432109
 0123456789876543210
123456789010987654321
.....................
```

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
void esc_caracter(int n, char simb)
{
    for (int i = 0; i < n; ++i) {
        cout << simb;
    }
}
//-----------------------------------------
void esc_ascendente(int a, int b)
{
    assert(a <= b);
    for (int i = a; i <= b; ++i) {
        cout << (i%10);
    }
}
//-----------------------------------------
void esc_descendente(int a, int b)
{
    for (int i = a; i >= b; --i) {
        cout << (i%10);
    }
}
//-----------------------------------------
void esc_fila(int f, int nf)
{
    assert(f <= nf);
    esc_caracter(nf-f, ' ');
    esc_ascendente(f, 2*f-1);
    esc_descendente(2*f-2, f);
    cout << endl;
}
//-----------------------------------------
void esc_triangulo(int nf)
{
    for (int f = 1; f <= nf; ++f) {
        esc_fila(f, nf);
    }
}
//-----------------------------------------
int main()
{
    cout << "Introduzca numero de filas: ";
    int n_filas;
    cin >> n_filas;
```

```
    esc_triangulo(n_filas);
}
```

4. Escribe un programa que calcule el valor de `S` para un número real X $(0 \leq X \leq 1)$ dado por teclado, utilizando la serie de Taylor para calcular el valor de $\arcsin(x)$:

$$S = X + \frac{1}{2}\frac{X^3}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{X^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{X^7}{7} + \cdots$$

   Nota: No se añadirán más de 7 sumandos.

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
const int LIMITE_FACTORIAL_UNSIGNED = 14;
const int MAX_ITER = 7;
//-----------------------------------------
double potencia(double base, int exp)
{
    assert(base >= 0 && base <= 1);
    double res = 1;
    for (int i = 0; i < exp; ++i) {
        res *= base;
    }
    return res;
}
//-----------------------------------------
int producto(int inicio, int limite)
{
    assert(limite <= LIMITE_FACTORIAL_UNSIGNED);
    int res = 1;
    for (int i = inicio; i < limite; i+=2) {
        res *= i;
    }
    return res;
}
//-----------------------------------------
double termino(double x, int i)
{
    return (double(producto(1, i)) / double(producto(2, i))
            * potencia(x, i) / double(i));
}
//-----------------------------------------
double serie(double x)
{
    assert(x >= 0 && x <= 1);
    double res = x;
    for (int i = 1; i < MAX_ITER; ++i) {
        res += termino(x, 2*i+1);
    }
    return res;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca el valor de X [0..1]: ";
    double x;
    cin >> x;
    if (! (x >= 0 && x <= 1)) {
        cout << "Error. Valor de X fuera de rango" << endl;
    } else {
        double s = serie(x);
        cout << "Serie: " << s << endl;
    }
}
```

5. Diseña un programa que encuentre el primer número perfecto mayor que 28. Un número es perfecto si coincide con la suma de sus divisores (salvo él mismo).

Por ejemplo, 28 es perfecto ya que $28 = 1 + 2 + 4 + 7 + 14$

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
int PRIMER_NUMERO = 29;
//-----------------------------------------
int suma_divisores(int n)
{
    int suma = 0;
    for (int i = 1; i <= n / 2; ++i) {
        if (n % i == 0) {
            suma += i;
        }
    }
    return suma;
}
//-----------------------------------------
inline bool es_perfecto(int n)
{
    return n == suma_divisores(n);
}
//-----------------------------------------
int primer_perfecto()
{
    int i = PRIMER_NUMERO;
    while (! es_perfecto(i)) {
        ++i;
    }
    assert(es_perfecto(i));
    return i;
}
//-----------------------------------------
int main()
{
    cout << "Primer perfecto mayor que " << PRIMER_NUMERO << ": "
         << primer_perfecto() << endl;
}
```

6. Dos números a y b se dice que son amigos si la suma de los divisores de a (salvo él mismo) coincide con b y viceversa. Diseña un programa que tenga como entrada dos números enteros n y m y que muestre en la pantalla todas las parejas de números amigos que existan en el intervalo determinado por n y m.

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
int suma_divisores(int n)
{
    int suma = 0;
    for (int i = 1; i <= n / 2; ++i) {
        if (n % i == 0) {
            suma += i;
        }
    }
    return suma;
}
//-----------------------------------------
inline bool son_amigos(int a, int b)
{
    return ((a == suma_divisores(b))&&(b == suma_divisores(a)));
}
//-----------------------------------------
void ordenar(int& n, int& m)
{
    if (n > m) {
        int aux = n;
```

```
        n = m;
        m = aux;
    }
    assert(n <= m);
}
//-----------------------------------------
void imprimir_amigos(int n, int m)
{
    ordenar(n, m);
    for (int i = n; i < m; ++i) {
        for (int j = i+1; j <= m; ++j) {
            if (son_amigos(i, j)) {
                cout << "Amigos: " << i << ", " << j << endl;
            }
        }
    }
}
//-----------------------------------------
int main()
{
    cout << "Introduzca un intervalo: ";
    int n, m;
    cin >> n >> m;
    imprimir_amigos(n, m);
}
```

Otra solución alternativa

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
int suma_divisores(int n)
{
    int suma = 0;
    for (int i = 1; i <= n / 2; ++i) {
        if (n % i == 0) {
            suma += i;
        }
    }
    return suma;
}
//-----------------------------------------
inline bool son_amigos(int a, int b)
{
    return ((a == suma_divisores(b))&&(b == suma_divisores(a)));
}
//-----------------------------------------
void numero_amigo(int a, int& b, bool& ok)
{
    b = suma_divisores(a);
    ok = (a == suma_divisores(b));
    assert(!ok || son_amigos(a, b));
}
//-----------------------------------------
void ordenar(int& n, int& m)
{
    if (n > m) {
        int aux = n;
        n = m;
        m = aux;
    }
    assert(n <= m);
}
//-----------------------------------------
void imprimir_amigos(int n, int m)
{
    ordenar(n, m);
    for (int i = n; i < m; ++i) {
        bool amigos;
        int j;
        numero_amigo(i, j, amigos);
        if (amigos && (i < j) && (j < m)) {
```

```
            cout << "Amigos: " << i << ", " << j << endl;
        }
    }
}
//-------------------------------------
int main()
{
    cout << "Introduzca un intervalo: ";
    int n, m;
    cin >> n >> m;
    imprimir_amigos(n, m);
}
```

Tema 3: Diseño Descendente. Subprogramas

Práctica 5-1. Buffer de Teclado

Laboratorio

1. Codifique el siguiente programa

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Introduzca un numero: ";
    int dato_1;
    cin >> dato_1;
    cout << "Introduzca otro numero: ";
    int dato_2;
    cin >> dato_2;
    cout << "Los numeros leidos son: " << dato_1 << ' ' << dato_2 << endl;
}
```

Ejecútelo introduciendo el número `12` y pulse `ENTER`, posteriormente introduzca el número `34` y pulse `ENTER`, compruebe la salida del programa.

A continuación, vuélvalo a ejecutar introduciendo el número `12`, pulse `ESPACIO`, número `34` y pulse `ENTER`, compruebe la salida del programa, compárela con la ejecución anterior, evalúe las diferencias y relaciónelas con el concepto de *búffer de entrada*.

Solución

```
//-----------------------------------------
// PRIMERA EJECUCIÓN
//-----------------------------------------
//
// El programa muestra el mensaje [Introduzca un numero: ] entonces la
// sentencia [cin >> dato_1] accede al BUFFER-DE-ENTRADA para leer un
// dato. Como el buffer está vacio, dicha sentencia esperará a que
// haya algún dato en el buffer.  El usuario teclea 12[ENTER] que se
// almacenará en el buffer, por lo que el programa se despierta y
// asignará el valor numerico 12 a la variable [dato_1], eliminándose
// los caracteres 1 y 2 del buffer (el carácter [ENTER] permanece en
// el buffer)
//
// Posteriormente el programa muestra el mensaje [Introduzca otro
// numero: ] entonces la sentencia [cin >> dato_2] accede al
// BUFFER-DE-ENTRADA para leer otro dato. Como el buffer NO está
// vacio, la lectura salta los espacios/ENTER iniciales, por lo que
// eliminará el caracter [ENTER] del buffer (quedaba allí de la
// lectura anterior), por lo que el buffer se queda vacio y por lo
// tanto la sentencia de lectura esperará a que haya algún dato en el
// buffer. El usuario teclea 34[ENTER] que se almacenará nuevamente en
// el buffer, por lo que el programa se despierta y asignará el valor
// numerico 34 a la variable [dato_2], eliminandose los caracteres 3 y
// 4 del buffer (el caracter [ENTER] permanece en el buffer)
//
// Finalmente muestra en pantalla los valores de ambas variables 12 y 34
//
//-----------------------------------------
// SEGUNDA EJECUCIÓN
//-----------------------------------------
//
// El programa muestra el mensaje "Introduzca un numero: " entonces la
// sentencia [cin >> dato_1] accede al BUFFER-DE-ENTRADA para leer un
// dato. Como el buffer está vacio, dicha sentencia esperará a que
// haya algún dato en el buffer. El usuario teclea
// 12[ESPACIO]34[ENTER] que se almacenará en el buffer, por lo que el
// programa se despierta y asignará el valor numerico 12 a la variable
// [dato_1], eliminándose los caracteres 1 y 2 del buffer (los
// caracteres [ESPACIO]34[ENTER] permanecen en el buffer)
//
// Posteriormente el programa muestra el mensaje "Introduzca otro
// numero: " entonces la sentencia [cin >> dato_2] accede al
```

```
// BUFFER-DE-ENTRADA para leer otro dato. Como el buffer NO está
// vacio, la sentencia de lectura no esperará, y por lo tanto, la
// lectura salta los espacios/ENTER iniciales, por lo que eliminará el
// caracter [ESPACIO] del buffer y asignará el valor numérico 34 a la
// variable [dato_2] (quedaban en el buffer de la lectura anterior),
// eliminandose los caracteres 3 y 4 del buffer (el caracter [ENTER]
// permanece en el buffer)
//
// Finalmente muestra en pantalla los valores de ambas variables 12 y 34
//
//-----------------------------------------
```

2. Desarrolle un programa que lea una palabra de cuatro letras por teclado, y posteriormente escriba dicha palabra de manera que cada letra se encuentre codificada sustituyéndola por aquel carácter que le sigue en la tabla de código ASCII. Nota: considere el concepto de *búffer de entrada*.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
// Esta solución NO considera el concepto de 'Buffer de Entrada'
//-----------------------------------------
int main()
{
    cout << "Introduzca una palabra de 4 letras: ";
    char l1, l2, l3, l4;
    cin >> ws;       // salta los espacios iniciales
    cin.get(l1);     // lee un carácter
    cin.get(l2);     // lee un carácter
    cin.get(l3);     // lee un carácter
    cin.get(l4);     // lee un carácter
    char n1 = char(l1 + 1);
    char n2 = char(l2 + 1);
    char n3 = char(l3 + 1);
    char n4 = char(l4 + 1);
    cout << "La palabra [" << l1<<l2<<l3<<l4 << "]"
         << " transformada es [" << n1<<n2<<n3<<n4 <<"]" << endl;
}
```

Solución Alternativa

```
#include <iostream>
using namespace std;
const int NLETRAS = 4;
//-----------------------------------------
// Esta solución considera el concepto de 'Buffer de Entrada'
//-----------------------------------------
int main()
{
    cout << "Introduzca una palabra de 4 letras: ";
    cin >> ws;       // salta los espacios iniciales
    cout << "La palabra transformada es [";
    for (int i = 0; i < NLETRAS; ++i) {
        char l1;
        cin.get(l1);             // lee un carácter
        char n1 = char(l1 + 1); // transforma un carácter
        cout << n1;              // escribe un carácter
    }
    cout << "]" << endl;
}
```

3. Desarrolle un programa que lea una secuencia de dígitos (caracteres) hasta que lea algo distinto de dígito, almacene su valor numérico en una variable entero, y posteriormente muestre dicho valor. Nota: considere el concepto de *búffer de entrada*.

Solución

```
#include <iostream>
```

```
using namespace std;
int main()
{
    cout << "Introduzca un número y pulse ENTER: ";
    char c;
    cin >> ws;
    cin.get(c);          // lee un carácter
    int numero = 0;
    while (c >= '0' && c <= '9') {
        int n = (c - '0');
        numero = numero * 10 + n;
        cin.get(c);      // lee un carácter
    }
    cout << "Valor numérico: " << numero << endl;
}
```

Tema 3: Diseño Descendente. Subprogramas

Práctica 5-1. Buffer de Teclado

Ejercicios Complementarios

1. Desarrolle un programa que lea una palabra formada por letras minúsculas hasta leer `ENTER` (`'\n'`) y a continuación la escriba en mayúsculas. Nota: considere el concepto de *búffer de entrada*.

Solución

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Introduzca una frase hasta 'ENTER': ";
    char c;
    cin.get(c);          // lee un carácter
    while (c != '\n') {
        if (c >= 'a' && c <= 'z') {
            c = char('A' + (c - 'a'));
        }
        cout << c;
        cin.get(c);      // lee un carácter
    }
    cout << endl;
}
```

2. Codifique el siguiente programa

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    cout << "Introduzca un numero: ";
    int num = 123456;
    cin >> num;
    char c = '#';
    cin.get(c);
    cout << "El numero leido es: " << num << endl;
    if (c < ' ') {
        cout << "El separador (cod ASCII) es: "
             << int(c) << endl;
    } else {
        cout << "El separador es: '"
             << c << "'" << endl;
    }
}
```

Ejecútelo para las siguientes entradas de datos, y analice las salidas correspondientes:

- Pulse varios espacios, `ENTER`, varios espacios, el número `9876` y `ENTER`.
- Pulse varios espacios, `ENTER`, varios espacios, el número `9876`, varios espacios y `ENTER`.
- Pulse varios espacios, el número `9876`, la letra `w` y `ENTER`.
- Pulse varios espacios, la letra `x`, el número `9876`, la letra `w` y `ENTER`.

Solución

```
//-----------------------------------------
// NOTA INFORMATIVA:
//-----------------------------------------
// Para simplificar la explicación posterior, se utilizará el símbolo
// [_] para denotar el carácter [ESPACIO] (ASCII 32), y el símbolo [$]
// para denotar el carácter [ENTER] (ASCII 10)
//
//-------------------------------------------------------------------------
// PRIMERA EJECUCIÓN
//-------------------------------------------------------------------------
```

```
// $ ./t2_ej_04
// Introduzca un numero: __$
// ___9876$
// El numero leido es: 9876
// El separador (cod ASCII) es: 10
//-------------------------------------------------------------------------------
// INSTRUCCIÓN          BUFFER              NUM          C          CIN
//-------------------------------------------------------------------------------
//                      [__$___9876$]       [123456]     [#]        [ok]
// cin >> num;
//                      [$]                 [9876]       [#]        [ok]
// cin.get(c);
//                      []                  [9876]       [$]        [ok]
//-------------------------------------------------------------------------------
// SEGUNDA EJECUCIÓN
//-------------------------------------------------------------------------------
// $ ./t2_ej_04
// Introduzca un numero: __$
// ___9876__$
// El numero leido es: 9876
// El separador es: ' '
//-------------------------------------------------------------------------------
// INSTRUCCIÓN          BUFFER              NUM          C          CIN
//-------------------------------------------------------------------------------
//                      [__$___9876__$]     [123456]     [#]        [ok]
// cin >> num;
//                      [__$]               [9876]       [#]        [ok]
// cin.get(c);
//                      [_$]                [9876]       [_]        [ok]
//-------------------------------------------------------------------------------
// TERCERA EJECUCIÓN
//-------------------------------------------------------------------------------
// $ ./t2_ej_04
// Introduzca un numero: ___9876w$
// El numero leido es: 9876
// El separador es: 'w'
//-------------------------------------------------------------------------------
// INSTRUCCIÓN          BUFFER              NUM          C          CIN
//-------------------------------------------------------------------------------
//                      [___9876w$]         [123456]     [#]        [ok]
// cin >> num;
//                      [w$]                [9876]       [#]        [ok]
// cin.get(c);
//                      [$]                 [9876]       [w]        [ok]
//-------------------------------------------------------------------------------
// CUARTA EJECUCIÓN
//-------------------------------------------------------------------------------
// $ ./t2_ej_04
// Introduzca un numero: ___x9876w$
// El numero leido es: 123456
// El caracter es: '#'
//-------------------------------------------------------------------------------
// INSTRUCCIÓN          BUFFER              NUM          C          CIN
//-------------------------------------------------------------------------------
//                      [___x9876w$]        [123456]     [#]        [ok]
// cin >> num;
//                      [x9876w$]           [123456]     [#]        [fail]
// cin.get(c);
//                      [x9876w$]           [123456]     [#]        [fail]
//-------------------------------------------------------------------------------
```

Tema 3: Diseño Descendente. Subprogramas

Práctica 5-2. Subprogramas (II)

Laboratorio

1. Escriba un programa que tome como entrada desde teclado dos números enteros (mayores que cero) N e i, y muestre en pantalla el dígito que ocupa la posición i-ésima del número N. Si i es mayor que el número de dígitos de N, se mostrará en pantalla 0. Por ejemplo, para $N = 25064$ e $i = 2$, el resultado es el dígito 6, y para $i = 6$, el resultado es 0.

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
// El índice del primer dígito es el cero
//-----------------------------------------
int digito(int n, int i)
{
    for (int j = 0; (j < i); ++j) {
        n = n / 10;
    }
    return n % 10;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca número: ";
    int num;
    cin >> num;
    cout << "Introduzca dígito: ";
    int dig;
    cin >> dig;
    if (num <= 0 || dig <= 0) {
        cout << "Error, número o dígito erróneo" << endl;
    } else {
        cout << "Valor del dígito: " << digito(num, dig-1) << endl;
    }
}
```

2. Escribe un programa que acepte como entrada desde teclado un número entero mayor que cero y dé como salida el resultado de sumar dos a dos los dígitos que aparecen en posiciones simétricas respecto al dígito central dentro del número dado como entrada. Por ejemplo :

   - Para el número : 2354869
   - La salida es: $2 + 9 = 11$, $3 + 6 = 9$, $5 + 8 = 13$, 4
   - Para el número : 6582
   - La salida es : $6 + 2 = 8$, $5 + 8 = 13$

Solución

```
#include <iostream>
#include <cassert>
using namespace std;
//-----------------------------------------
int n_digitos(int n)
{
    int i = 1;
    while (n > 9) {
        n = n / 10;
        ++i;
    }
    return i;
}
//-----------------------------------------
int digito(int n, int i)
{
```

```
    for (int j = 0; (j < i); ++j) {
        n = n / 10;
    }
    return n % 10;
}
//-----------------------------------------
void imprimir_suma_digitos(int n)
{
    int nd = n_digitos(n);
    for (int i = 0; i < nd / 2; ++i) {
        int di = digito(n, i);
        int df = digito(n, nd-i-1);
        cout << di << " + " << df << " = " << (di+df) << ", ";
    }
    if (nd%2 != 0) {
        cout << digito(n, nd / 2);
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca número: ";
    int num;
    cin >> num;
    imprimir_suma_digitos(num);
}
```

3. Dada una sucesión, de longitud indeterminada, de caracteres ceros y unos, construir un programa que permita calcular el tamaño de la mayor subsucesión ordenada de menor a mayor. La sucesión se lee desde el teclado, y el final viene dado por el carácter punto (`'.'`). Considere el concepto de *búffer de entrada*. Ejemplos:

   - Para la sucesión de entrada: `001001101.` imprimirá `4`.
   - Para la sucesión de entrada: `0100101111.` imprimirá `5`.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
inline bool es_fin_sec(char ant, char act)
{
    return (act < ant);
}
//-----------------------------------------
void fin_sec_ord(int& cnt, int& mayor)
{
    if (cnt > mayor) {
        mayor = cnt;
    }
    cnt = 0;
}
//-----------------------------------------
void leer(char& ant, char& act)
{
    ant = act;
    cin >> act;
}
//-----------------------------------------
int leer_sucesion()
{
    int mayor = 0;
    int cnt = 0;
    char ant = '0'; // valor ficticio
    char act = '0'; // valor ficticio
    leer(ant, act);
    while (act != '.') {
        if (es_fin_sec(ant, act)) {
            fin_sec_ord(cnt, mayor);
        }
```

```
        ++cnt;
        leer(ant, act);
    }
    fin_sec_ord(cnt, mayor);
    return mayor;
}
//-----------------------------------
int main()
{
    cout << "Introduzca sucesión de ceros y unos hasta punto: ";
    int lng = leer_sucesion();
    cout << "Mayor subsucesión ordenada: " << lng << endl;
}
```

Tema 3: Diseño Descendente. Subprogramas

Práctica 5-2. Subprogramas (II)

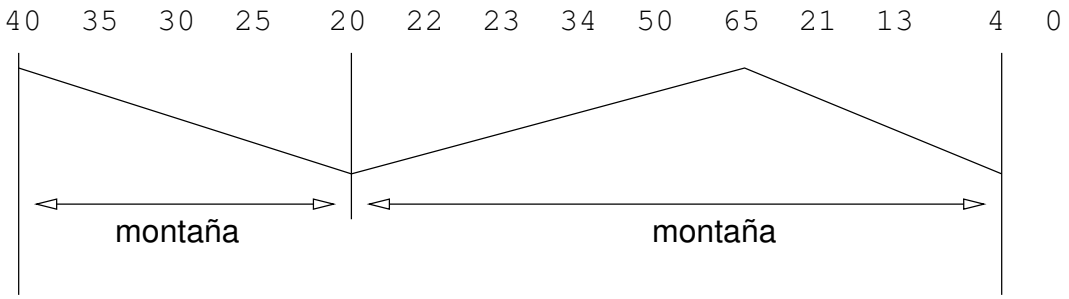
Ejercicios Complementarios

1. Decimos que una sucesión $a_1, a_2, \cdots, a_n$ de enteros forma una montaña, si existe un h tal que : $1 \leq h \leq n$ y además

$$a_1 < ...a_{h-1} < a_h > a_{h+1} > \cdots a_n$$

Definimos la anchura de una montaña como el número de enteros que la forman. Por ejemplo la sucesión $-7, -1, 6, 21, 15$ es una montaña de anchura 5.

Dada una secuencia de números enteros terminada en cero (0) y separados por espacios en blanco, que como mínimo contiene una montaña (suponemos que la secuencia de enteros de entrada es una secuencia correcta de montañas), diseña un programa que calcule la anchura de la montaña más ancha. Nota: el cero (0) terminador no forma parte de la secuencia. Por ejemplo para la secuencia: `40 35 30 25 20 22 23 34 50 65 21 13 4 0` producirá `9` como la mayor anchura de las montañas.



Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
inline bool es_fin_mont(int ant2, int ant1, int act)
{
    return (ant1 < ant2)&&(ant1 < act);
}
//-----------------------------------------
void fin_sec(int& cnt, int& mayor)
{
    if (cnt > mayor) {
        mayor = cnt;
    }
    cnt = 1;
}
//-----------------------------------------
void leer(int& ant2, int& ant1, int& act)
{
    ant2 = ant1;
    ant1 = act;
    cin >> act;
}
//-----------------------------------------
void leer_sucesion(int& mont_mayor)
{
    mont_mayor = 0;
    int mont_cnt = 0;
    int ant2 = 0; // valor ficticio
    int ant1 = 0; // valor ficticio
    int act = 0; // valor ficticio
    leer(ant2, ant1, act);
    while (act != 0) {
        if (es_fin_mont(ant2, ant1, act)) {
            fin_sec(mont_cnt, mont_mayor);
        }
        ++mont_cnt;
        leer(ant2, ant1, act);
```

```
    }
    fin_sec(mont_cnt, mont_mayor);
}
//-----------------------------------------
int main()
{
    cout << "Introduzca sucesión de enteros hasta cero: ";
    int mm;
    leer_sucesion(mm);
    cout << "Mayor Montaña: " << mm << endl;
}
```

2. Decimos que una sucesión $a_1, a_2, \cdots, a_n$ de enteros forma una montaña, si existe un h tal que : $1 \leq h \leq n$ y además

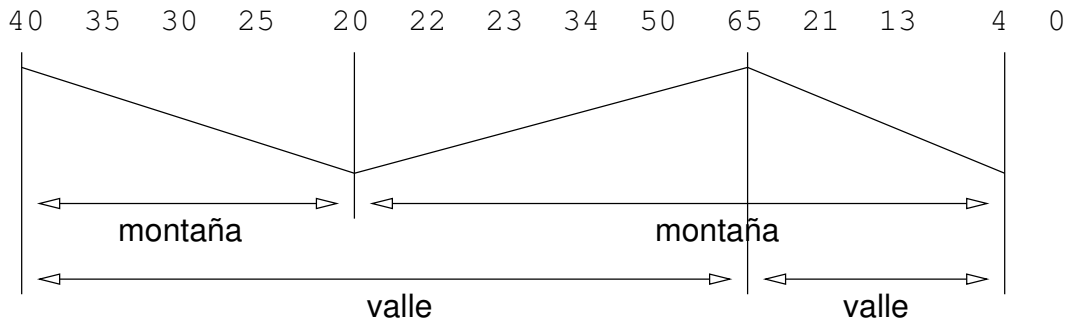
$$a_1 < ...a_{h-1} < a_h > a_{h+1} > \cdots a_n$$

Definimos la anchura de una montaña como el número de enteros que la forman. Por ejemplo la sucesión $-7, -1, 6, 21, 15$ es una montaña de anchura 5. Definimos un valle de la misma forma que una montaña pero cambiando el signo de las desigualdades de la definición anterior:

$$a_1 > ...a_{h-1} > a_h < a_{h+1} < \cdots a_n$$

Por ejemplo $24, 13, 6, 15, 50$ sería un valle.

Dada una secuencia de números enteros terminada en cero (0) y separados por espacios en blanco, que como mínimo contiene una montaña y un valle (suponemos que la secuencia de enteros de entrada es una secuencia correcta de montañas y valles), diseña un programa que calcule la anchura de la montaña más ancha y la anchura del valle más ancho. Nota: el cero (0) terminador no forma parte de la secuencia. Por ejemplo para la secuencia: `40 35 30 25 20 22 23 34 50 65 21 13 4 0` producirá `9` como la mayor anchura de las montañas y `10` como la mayor anchura de los valles.



Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
inline bool es_fin_mont(int ant2, int ant1, int act)
{
    return (ant1 < ant2)&&(ant1 < act);
}
//-----------------------------------------
inline bool es_fin_vall(int ant2, int ant1, int act)
{
    return (ant1 > ant2)&&(ant1 > act);
}
//-----------------------------------------
void fin_sec(int& cnt, int& mayor)
{
    if (cnt > mayor) {
        mayor = cnt;
    }
    cnt = 1;
}
//-----------------------------------------
void leer(int& ant2, int& ant1, int& act)
```

```
{
    ant2 = ant1;
    ant1 = act;
    cin >> act;
}
//-----------------------------------------
void leer_sucesion(int& mont_mayor, int& vall_mayor)
{
    mont_mayor = 0;
    vall_mayor = 0;
    int mont_cnt = 0;
    int vall_cnt = 0;
    int ant2 = 0; // valor ficticio
    int ant1 = 0; // valor ficticio
    int act = 0; // valor ficticio
    leer(ant2, ant1, act);
    while (act != 0) {
        if (es_fin_mont(ant2, ant1, act)) {
            fin_sec(mont_cnt, mont_mayor);
        } else if (es_fin_vall(ant2, ant1, act)) {
            fin_sec(vall_cnt, vall_mayor);
        }
        ++mont_cnt;
        ++vall_cnt;
        leer(ant2, ant1, act);
    }
    fin_sec(mont_cnt, mont_mayor);
    fin_sec(vall_cnt, vall_mayor);
}
//-----------------------------------------
int main()
{
    cout << "Introduzca sucesión de enteros hasta cero: ";
    int mm, vm;
    leer_sucesion(mm, vm);
    cout << "Mayor Montaña: " << mm << endl;
    cout << "Mayor Valle: " << vm << endl;
}
```

Tema 3: Diseño Descendente. Subprogramas

Ejercicios de Autoevaluación

1. Diseñe las funciones que se indican a continuación para calcular la raíz cuadrada de un número **real** k mayor o igual a uno ($k \geq 1$) mediante el *algoritmo babilónico* (nótese que no se puede utilizar la función *pow* ni la función *sqrt* de la biblioteca estándar).

   - `double aprox_inicial(double k)`

     Muchos de los métodos de cálculo para raíces cuadradas requieren un valor inicial. Si el valor inicial está muy lejos de la raíz cuadrada real, el cálculo será muy lento. Por lo tanto es útil tener un cálculo aproximado, que puede ser muy inexacto pero fácil de calcular.

     Diseñe una función que recibe un determinado valor k ($k \geq 1$) de tipo *real* y devuelve la estimación del valor inicial $x_0 \in \mathbb{R}$ según el método descrito a continuación:

     Sea $D \in \mathbb{N}$ el número de dígitos de la parte entera del número k.[1]
     - Si D es impar, entonces $z = (D-1)/2$ y $x_0 = 2 * 10^z$
     - Si D es par, entonces $z = (D-2)/2$ y $x_0 = 6 * 10^z$

     Nótese que es adecuado desarrollar una función para calcular el número de dígitos de un número, así como otra función para calcular la potencia de 10 elevado a z.

   - `double raiz_cuadrada(double k)`

     Diseñe una función que reciba un determinado valor k ($k \geq 1$) de tipo *real* y devuelve el valor de $\sqrt{k}$ según el método Babilónico descrito a continuación:

     A partir del valor de x_0 resultado de invocar a la función `aprox_inicial`, definida anteriormente, sobre el número k, se itera el siguiente cálculo hasta que el valor absoluto de la diferencia entre x_n y x_{n-1} sea menor que un determinado `ERROR_DE_PRECISION` ($= 0.000001$)

     $$x_n = (x_{n-1} + (k/x_{n-1}))/2$$

     Nótese que es un proceso iterativo en el que para calcular el nuevo valor (x_n) se utiliza el valor calculado en la iteración anterior (x_{n-1}). Por ejemplo:

k	=	9876543.21	
D	=	7	
z	=	3	
x_0	=	2000	
x_1	=	3469.14	$x1 = (x0 + k/x0)/2$
x_2	=	3158.06	$x2 = (x1 + k/x1)/2$
x_3	=	3142.73	$x3 = (x2 + k/x2)/2$
x_4	=	3142.7	$x4 = (x3 + k/x3)/2$
x_5	=	3142.7	$x5 = (x4 + k/x4)/2$
NIteraciones	=	5	
resultado	=	3142.7	

k	=	123456.789	
D	=	6	
z	=	2	
x_0	=	600	
x_1	=	402.881	$x1 = (x0 + k/x0)/2$
x_2	=	354.658	$x2 = (x1 + k/x1)/2$
x_3	=	351.379	$x3 = (x2 + k/x2)/2$
x_4	=	351.364	$x4 = (x3 + k/x3)/2$
x_5	=	351.364	$x5 = (x4 + k/x4)/2$
NIteraciones	=	5	
resultado	=	351.364	

   - El programa principal debe leer un número de tipo *real*. Si el número leído es menor que 1, entonces muestra un mensaje de error. En otro caso invoca al subprograma `raiz_cuadrada` para calcular el valor de la raíz cuadrada del número leído, el cual deberá ser mostrado en pantalla. Por ejemplo:

`Introduce un número: 0` `Error`	`Introduce un número: 9876543.21` `Sqrt: 3142.7`	`Introduce un número: 123456.789` `Sqrt: 351.364`

[1] El número de dígitos a la izquierda del punto decimal del número k.

Tema 4: Tipos de Datos Estructurados

Práctica 6. Registros y Strings

Laboratorio

1. Para realizar operaciones con números complejos, podemos definir el siguiente tipo:

```
struct Complejo {
   double real;
   double img;
};
```

Escribe subprogramas que realicen las operaciones de leer, mostrar, suma y resta, (multiplicación y división en casa) de números complejos definidos con el tipo anterior, así como el programa para probar adecuadamente su funcionamiento.

Solución

```
#include <iostream>
using namespace std;
//-----------------------------------------
const double ERROR_PRECISION = 1e-10;
//-----------------------------------------
struct Complejo {
    double real;
    double img;
};
//-----------------------------------------
inline double sq(double x)
{
    return x*x;
}
//-----------------------------------------
bool iguales(double x, double y)
{
    double cmp = x-y;
    return (-ERROR_PRECISION <= cmp)&&(cmp <= ERROR_PRECISION);
}
//-----------------------------------------
Complejo crear(double real, double img)
{
    Complejo res;
    res.real = real;
    res.img =  img;
    return res;
}
//-----------------------------------------
Complejo sumar(const Complejo& c1, const Complejo& c2)
{
    Complejo res;
    res.real = c1.real + c2.real;
    res.img =  c1.img + c2.img;
    return res;
}
//-----------------------------------------
Complejo restar(const Complejo& c1, const Complejo& c2)
{
    Complejo res;
    res.real = c1.real - c2.real;
    res.img =  c1.img - c2.img;
    return res;
}
//-----------------------------------------
Complejo multiplicar(const Complejo& c1, const Complejo& c2)
{
    Complejo res;
    res.real = (c1.real*c2.real) - (c1.img*c2.img);
    res.img =  (c1.real*c2.img) + (c1.img*c2.real);
    return res;
}
//-----------------------------------------
Complejo dividir(const Complejo& c1, const Complejo& c2)
```

```
{
    Complejo res;
    res.real = ((c1.real*c2.real)+(c1.img*c2.img)) / (sq(c2.real)+sq(c2.img));
    res.img =  ((c1.img*c2.real)-(c1.real*c2.img)) / (sq(c2.real)+sq(c2.img));
    return res;
}
//-----------------------------------------
bool iguales(const Complejo& c1, const Complejo& c2)
{
    return iguales(c1.real, c2.real)&&iguales(c1.img, c2.img);
}
//-----------------------------------------
bool distintos(const Complejo& c1, const Complejo& c2)
{
    return ! iguales(c1, c2);
}
//-----------------------------------------
inline void escribir(const Complejo& c)
{
    cout << "( "<<c.real<<", "<<c.img<<" )";
}
//-----------------------------------------
//-----------------------------------------
//-----------------------------------------
void leer(Complejo& c)
{
    cout << "Introduzca un número complejo (real, img): ";
    cin >> c.real >> c.img;
}
//-----------------------------------------
void prueba_suma(const Complejo& c1, const Complejo& c2)
{
    Complejo c0 = sumar(c1, c2);
    escribir(c1);
    cout <<" + ";
    escribir(c2);
    cout <<" = ";
    escribir(c0);
    cout<< endl;
    if (distintos(c1, restar(c0, c2))) {
        cout << "Error en operaciones de suma/resta"<< endl;
    }
}
//-----------------------------------------
void prueba_resta(const Complejo& c1, const Complejo& c2)
{
    Complejo c0 = restar(c1, c2);
    escribir(c1);
    cout <<" - ";
    escribir(c2);
    cout <<" = ";
    escribir(c0);
    cout<< endl;
    if (distintos(c1, sumar(c0, c2))) {
        cout << "Error en operaciones de suma/resta"<< endl;
    }
}
//-----------------------------------------
void prueba_mult(const Complejo& c1, const Complejo& c2)
{
    Complejo c0 = multiplicar(c1, c2);
    escribir(c1);
    cout <<" * ";
    escribir(c2);
    cout <<" = ";
    escribir(c0);
    cout<< endl;
    if (distintos(c1, dividir(c0, c2))) {
        cout << "Error en operaciones de mult/div"<< endl;
    }
}
//-----------------------------------------
void prueba_div(const Complejo& c1, const Complejo& c2)
{
```

```
    Complejo c0 = dividir(c1, c2);
    escribir(c1);
    cout <<" / ";
    escribir(c2);
    cout <<" = ";
    escribir(c0);
    cout<< endl;
    if (distintos(c1, multiplicar(c0, c2))) {
        cout << "Error en operaciones de mult/div"<< endl;
    }
}
//-----------------------------------------
int main()
{
    Complejo c1, c2;
    leer(c1);
    leer(c2);
    //-----------------------------------------
    prueba_suma(c1, c2);
    prueba_resta(c1, c2);
    prueba_mult(c1, c2);
    prueba_div(c1, c2);
    //-----------------------------------------
}
```

2. Diseñe una función para buscar la posición que ocupa un determinado carácter en una cadena de caracteres. Si el carácter a buscar no se encuentra en la cadena, entonces devolverá un valor fuera del rango válido para esa cadena. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Téngase en consideración la evaluación en *cortocircuito* de expresiones lógicas.

```
int buscar (const string& cadena, char car);
```

Solución

```
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
int buscar(const string& cadena, char car)
{
    int i = 0;
    // La evaluación en CORTOCIRCUITO hace que la siguiente expresión
    // sea correcta, ya que se comprueba que el valor de (i) sea
    // correcto antes de que se utilice como índice de la cadena
    while ((i < int(cadena.size())) && (car != cadena[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca una cadena: ";
    string cad;
    getline(cin, cad);
    cout << "Introduzca un carácter a buscar: ";
    char car;
    cin.get(car);
    int i = buscar(cad, car);
    if (i >= int(cad.size())) {
        cout << "El carácter [" << car <<"] no se encuentra en la cadena"<<endl;
    } else {
        cout << "El carácter [" << car <<"] se encuentra en la posición "<<i
             <<" en la cadena" << endl;
    }
    cout << "Cadena: " << cad << endl;
}
```

3. Diseñe una función para buscar la posición que ocupa un patrón dado en una cadena de caracteres. Si el patrón no se encuentra en la cadena, entonces devolverá un valor fuera del rango válido para esa cadena.

Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Téngase en consideración la evaluación en *cortocircuito* de expresiones lógicas.

```
int buscar (const string& cadena, const string& patron);
```

Solución

```
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
int buscar(const string& cadena, const string& patron)
{
    int i = 0;
    // La evaluación en CORTOCIRCUITO hace que la siguiente expresión
    // sea correcta, ya que se comprueba que el valor de (i) sea
    // correcto antes de que se utilice como índice de la subcadena
    while ((i+int(patron.size()) <= int(cadena.size()))
           && (patron != cadena.substr(i, int(patron.size())))) {
        ++i;
    }
    if (i+int(patron.size()) > int(cadena.size())) {
        i = int(cadena.size());
    }
    return i;
}
//-----------------------------------------
int main()
{
    cout << "Introduzca una cadena: ";
    string cad;
    getline(cin, cad);
    cout << "Introduzca un patrón a buscar: ";
    string pat;
    getline(cin, pat);
    int i = buscar(cad, pat);
    if (i >= int(cad.size())) {
        cout << "El patrón [" << pat <<"] no se encuentra en la cadena"<<endl;
    } else {
        cout << "El patrón [" << pat <<"] se encuentra en la posición "<<i
             <<" en la cadena" << endl;
    }
    cout << "Cadena: " << cad << endl;
}
```

4. Diseñe un procedimiento que recibe como primer parámetro una cadena de caracteres, y devuelve en otro parámetro de salida otra nueva cadena de caracteres que contiene exclusivamente todas las consonantes que se encuentran en la cadena de caracteres recibida como primer parámetro. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
void extraer_consonantes (const string& cadena, string& nueva);
```

Solución

```
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
bool es_letra(char c)
{
    return (('a' <= c && c <= 'z')||('A' <= c && c <= 'Z'));
}
//-----------------------------------------
bool es_vocal(char c)
{
    return (c == 'a')||(c == 'e')||(c == 'i')||(c == 'o')||(c == 'u')
         ||(c == 'A')||(c == 'E')||(c == 'I')||(c == 'O')||(c == 'U');
}
//-----------------------------------------
bool es_consonante(char c)
```

```
{
    return es_letra(c) && ! es_vocal(c);
}
//-----------------------------------------
void extraer_consonantes(const string& cadena, string& nueva)
{
    nueva.clear();
    for (unsigned i = 0; i < cadena.size(); ++i) {
        if (es_consonante(cadena[i])) {
            nueva += cadena[i];
        }
    }
}
//-----------------------------------------
int main()
{
    string cadena, nueva;
    cout << "Introduzca una cadena: ";
    getline(cin, cadena);
    extraer_consonantes(cadena, nueva);
    cout << "Cadena original: " << cadena << endl;
    cout << "Cadena nueva: " << nueva << endl;
}
```

5. Diseñe un procedimiento que recibe como parámetro de entrada/salida una cadena de caracteres, y elimina de dicha cadena de caracteres todas las vocales, devolviendo finalmente, por el mismo parámetro, la cadena de caracteres original con las vocales eliminadas. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
void eliminar_vocales (string& cadena);
```

Solución

```
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
bool es_vocal(char c)
{
    return (c == 'a')||(c == 'e')||(c == 'i')||(c == 'o')||(c == 'u')
         ||(c == 'A')||(c == 'E')||(c == 'I')||(c == 'O')||(c == 'U');
}
//-----------------------------------------
void eliminar_vocales(string& cadena)
{
    string nueva;
    for (unsigned i = 0; i < cadena.size(); ++i) {
        if (! es_vocal(cadena[i])) {
            nueva += cadena[i];
        }
    }
    cadena = nueva;
}
//-----------------------------------------
int main()
{
    string cadena;
    cout << "Introduzca una cadena: ";
    getline(cin, cadena);
    cout << "Cadena original: " << cadena << endl;
    eliminar_vocales(cadena);
    cout << "Cadena resultado: " << cadena << endl;
}
```

Solución Alternativa

```
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
```

```
bool es_vocal(char c)
{
    return (c == 'a')||(c == 'e')||(c == 'i')||(c == 'o')||(c == 'u')
         ||(c == 'A')||(c == 'E')||(c == 'I')||(c == 'O')||(c == 'U');
}
//-----------------------------------
void eliminar_vocales(string& cadena)
{
    unsigned j = 0;
    for (unsigned i = 0; i < cadena.size(); ++i) {
        if (! es_vocal(cadena[i])) {
            cadena[j] = cadena[i];
            ++j;
        }
    }
    cadena.resize(j);
}
//-----------------------------------
int main()
{
    string cadena;
    cout << "Introduzca una cadena: ";
    getline(cin, cadena);
    cout << "Cadena original: " << cadena << endl;
    eliminar_vocales(cadena);
    cout << "Cadena resultado: " << cadena << endl;
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 6. Registros y Strings

Ejercicios Complementarios

1. Diseñe un programa que tomando como entrada un texto, realice el listado por pantalla de todas las palabras del texto que comiencen por ciertas iniciales. Dichas iniciales serán las letras que componen la primera palabra del texto.

   - El texto contiene un número **indefinido** de palabras en letras minúsculas y termina con la palabra `fin`.
   - Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos minúsculas).
   - El carácter separador de palabras es el espacio en blanco.

   por ejemplo:

   ```
   Introduce texto:
   hola hoy amanece temprano y la tarde cae en el olvido fin
   Salida:
   hoy amacene la olvido
   ```

   Solución

   ```cpp
   #include <iostream>
   #include <string>
   using namespace std;
   //-----------------------------------------
   const string FIN_SEC = "fin";
   //-----------------------------------------
   int buscar(const string& s, char x)
   {
       int i = 0;
       while ((i < int(s.size())) && (x != s[i])) {
           ++i;
       }
       return i;
   }
   //-----------------------------------------------------------------------------
   void imprimir_palabras(const string& patron)
   {
       string palabra;
       cin >> palabra;
       while (palabra != FIN_SEC){
           if (buscar(patron, palabra[0]) < int(patron.size())) {
               cout << palabra << " ";
           }
           cin >> palabra;
       }
       cout << endl;
   }
   //-----------------------------------------------------------------------------
   int main()
   {
       cout << "Introduzca el texto en minúsculas hasta (fin) con "
            << "el patrón de iniciales al principio." << endl;
       string patron;
       cin >> patron;
       if (patron != FIN_SEC){
           imprimir_palabras(patron);
       }
   }
   ```

2. Se deberá realizar un programa que lea una cadena de caracteres que será utilizada como clave de cifrado, y posteriormente, se leerá un texto de "longitud indefinida" hasta un carácter punto ('.') el cual deberá ser cifrado y mostrado por pantalla. El cifrado (según Vigenère) se realizará de la siguiente manera:

   - Sólo se cifrarán las letras minúsculas.

- Cada carácter de la clave (se supone que sólo esta formada por letras minúsculas) se corresponde con un valor numérico que representa un incremento igual a la distancia alfabética de dicha letra respecto a la letra `'a'`. Por ejemplo al carácter `'a'` de la clave le corresponde un incremento de 0, al carácter `'b'` un incremento de 1, a `'c'` de 2, y así sucesivamente.
- A cada carácter del texto a cifrar se le asocia para realizar el cifrado la letra de la clave correspondiente a su posición (el primer carácter ocupa la posición 0) módulo el número de caracteres de la clave. Nota: módulo significa el resto de la división.
- Cada carácter del texto a cifrar será cifrado mediante un incremento circular (el siguiente a la `'z'` es la `'a'`) correspondiente a la letra de la clave asociada.

Por ejemplo, para la clave `abx` y el texto de entrada `hola y adios.` mostrará como resultado `hpia z xdjls.` como se indica en el siguiente esquema:

```
abxa b xabxa
hola y adios.
-------------
hpia z xdjls.
-------------
|||| | |||||
|||| | ||||+--> s + 0 = s
|||| | |||+---> o + 23 = l
|||| | ||+----> i + 1 = j
|||| | |+-----> d + 0 = d
|||| | +------> a + 23 = x
|||| +--------> y + 1 = z
|||+----------> a + 0 = a
||+-----------> l + 23 = i
|+------------> o + 1 = p
+-------------> h + 0 = h
```

Solución

```
#include <cassert>
#include <iostream>
#include <string>
using namespace std;
//-----------------------------------------
// https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
//-----------------------------------------
bool es_minuscula(char c)
{
    return ((c >= 'a')&&(c <= 'z'));
}
//-----------------------------------------
void incremento_circular(int& i, int limite)
{
    ++i;                    //  i = (i + 1) % limite;
    if (i >= limite) {      //
        i = 0;              //
    }                       //
}
//-----------------------------------------
void inc_circular(int& x, int inc, int min, int max)
{
    assert((x >= min)&&(x <= max) && (inc < max - min + 1));
    x = x + inc;                    // x = ((((x+inc)-min)%(max-min+1))+min);
    if (x > max) {                  //
        x = min + x - max - 1;      //
    }                               //
}
//-----------------------------------------
void cifrar(char& c, char cl)
{
    assert(es_minuscula(c) && es_minuscula(cl));
    const int inc = (cl - 'a');
    int x = c;
    inc_circular(x, inc, int('a'), int('z'));
    c = char(x);
}
//-----------------------------------------
void cifrar_texto(const string& clave)
```

```
{
    assert(int(clave.size()) > 0);
    int i = 0;
    cout << "Texto: ";
    cin >> ws;
    char c;
    cin.get(c);
    cout << "Resultado: ";
    while (c != '.') {
        if (es_minuscula(c)) {
            cifrar(c, clave[i]);
            incremento_circular(i, int(clave.size()));
        }
        cout << c;
        cin.get(c);
    }
    cout << c;
}
//-----------------------------------
int main()
{
    string clave;

    cout << "Clave: ";
    cin >> clave;
    cifrar_texto(clave);
    cout << endl;
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 7. Arrays

Laboratorio

1. Diseñe una función que recibe un array de números enteros (de tamaño 5) como parámetro de entrada, y devuelve `true` si los elementos del array están ordenados ascendentemente, y `false` en otro caso. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
const int MAX = 5;
typedef array<int, MAX> Vector;
bool esta_ordenado (const Vector& v);
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 5;
//-----------------------------------------
typedef array<int, MAX> Vector;
//-----------------------------------------
bool esta_ordenado(const Vector& v)
{
    int i = 0;
    while ((i < int(v.size())-1)&&(v[i] <= v[i+1])) {
        ++i;
    }
    return (i >= int(v.size())-1);
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    cout << "[ ";
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Vector v1;
    leer(v1);
    escribir(v1);
    if (esta_ordenado(v1)) {
        cout << "El array está ordenado"<< endl;
    } else {
        cout << "El array no está ordenado"<< endl;
    }
}
```

2. Diseñe un procedimiento que recibe un array de números enteros (de tamaño 5) como parámetro de entrada/salida, y lo modifica rotando sus elementos una posición a la izquierda. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, el array `{ 1, 2, 3, 4, 5 }`, tras rotar una posición a la izquierda quedaría: `{ 2, 3, 4, 5, 1 }`.

Solución

```
#include <iostream>
```

```cpp
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 5;
//-----------------------------------------
typedef array<int, MAX> Vector;
//-----------------------------------------
void rotar_izq(Vector& v)
{
    int aux = v[0];
    for (int i = 0; i < int(v.size())-1; ++i) {
        v[i] = v[i+1];
    }
    v[int(v.size())-1] = aux;
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    cout << "[ ";
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Vector v1;
    leer(v1);
    escribir(v1);
    rotar_izq(v1);
    escribir(v1);
}
```

3. Diseñe un procedimiento que recibe un array de números enteros (de tamaño 5) como parámetro de entrada/salida, y lo modifica rotando sus elementos una posición a la derecha. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, el array `{ 1, 2, 3, 4, 5 }`, tras rotar una posición a la derecha quedaría: `{ 5, 1, 2, 3, 4 }`.

Solución

```cpp
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 5;
//-----------------------------------------
typedef array<int, MAX> Vector;
//-----------------------------------------
void rotar_dch(Vector& v)
{
    int aux = v[int(v.size())-1];
    for (int i = int(v.size())-1; i > 0; --i) {
        v[i] = v[i-1];
    }
    v[0] = aux;
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
```

```
//-----------------------------------------
void escribir(const Vector& v)
{
    cout << "[ ";
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Vector v1;
    leer(v1);
    escribir(v1);
    rotar_dch(v1);
    escribir(v1);
}
```

4. Diseñe un procedimiento que recibe una *lista* de números enteros (como máximo 20) como parámetro (de entrada/salida), y lo modifica eliminando el primer elemento que sea igual al valor recibido como segundo parámetro (de entrada), considerando que **no** es necesario mantener el orden relativo de los elementos. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, la lista `{ 5, 3, 1, 4, 4, 2, 7, 1 }`, tras eliminar el primer elemento igual al número `4` quedaría: `{ 5, 3, 1, 1, 4, 2, 7 }`.

```
const int MAX = 20;
typedef array<int, MAX> Datos;
struct Lista {
   int nelms;
   Datos elm;
};
void eliminar (Lista& v, int x);
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 20;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
int buscar(const Lista& v, int x)
{
    int i = 0;
    while ((i < v.nelms)&&(x != v.elm[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
void eliminar_pos(Lista& v, int pos)
{
    --v.nelms;
    if (pos < v.nelms) {
        v.elm[pos] = v.elm[v.nelms];
    }
}
//-----------------------------------------
void eliminar(Lista& v, int x)
{
    int j = buscar(v, x);
    if (j < v.nelms) {
        eliminar_pos(v, j);
    }
```

```
}
//-----------------------------------------
void inicializar(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, int x)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
//-----------------------------------------
void leer(Lista& v)
{
    int n;
    inicializar(v);
    cout <<"Introduzca una secuencia de números terminada en cero: " << endl;
    cin >> n;
    while (n != 0) {
        anyadir(v, n);
        cin >> n;
    }
}
//-----------------------------------------
void leer(int& x)
{
    cout <<"Introduzca un número: ";
    cin >> x;
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Lista v1;
    int x;
    leer(v1);
    leer(x);
    cout << "Lista: " ;
    escribir(v1);
    cout << "Eliminar: " << x << endl;
    eliminar(v1, x);
    cout << "Resultado: " ;
    escribir(v1);
}
```

5. Diseñe un procedimiento que recibe una *lista* de números enteros (como máximo 20) como parámetro (de entrada/salida), y lo modifica eliminando el primer elemento que sea igual al valor recibido como segundo parámetro (de entrada), considerando que se debe mantener el orden relativo de los elementos. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, la lista `{ 5, 3, 1, 4, 4, 2, 7, 1 }`, tras eliminar el primer elemento igual al número `4` quedaría: `{ 5, 3, 1, 4, 2, 7, 1 }`.

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 20;
//-----------------------------------------
typedef array<int, MAX> Datos;
```

```
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
int buscar(const Lista& v, int x)
{
    int i = 0;
    while ((i < v.nelms)&&(x != v.elm[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
void eliminar_ord_pos(Lista& v, int pos)
{
    --v.nelms;
    for (int i = pos; i < v.nelms; ++i) {
        v.elm[i] = v.elm[i + 1];
    }
}
//-----------------------------------------
void eliminar_ord(Lista& v, int x)
{
    int j = buscar(v, x);
    if (j < v.nelms) {
        eliminar_ord_pos(v, j);
    }
}
//-----------------------------------------
void inicializar(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, int x)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
//-----------------------------------------
void leer(Lista& v)
{
    int n;
    inicializar(v);
    cout <<"Introduzca una secuencia de números terminada en cero: " << endl;
    cin >> n;
    while (n != 0) {
        anyadir(v, n);
        cin >> n;
    }
}
//-----------------------------------------
void leer(int& x)
{
    cout <<"Introduzca un número: ";
    cin >> x;
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Lista v1;
    int x;
```

```
    leer(v1);
    leer(x);
    cout << "Lista: " ;
    escribir(v1);
    cout << "Eliminar: " << x << endl;
    eliminar_ord(v1, x);
    cout << "Resultado: " ;
    escribir(v1);
}
```

6. Diseñe un procedimiento que recibe una *lista* de números enteros (como máximo 20) como parámetro (de entrada/salida), y lo modifica eliminando todos los números que se encuentren en otra *lista* recibida como segundo parámetro (de entrada), considerando que se debe mantener el orden relativo de los elementos. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, la lista `{ 5, 3, 1, 4, 4, 2, 7, 1 }`, tras eliminar los números que aparecen en `{ 5, 4, 1 }` quedaría: `{ 3, 2, 7 }`.

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 20;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
int buscar(const Lista& v, int x)
{
    int i = 0;
    while ((i < v.nelms)&&(x != v.elm[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
bool encontrado(const Lista& v, int x)
{
    return buscar(v, x) < v.nelms;
}
//-----------------------------------------
void eliminar_numeros(Lista& v, const Lista& ref)
{
    int j = 0;
    for (int i = 0; i < v.nelms; ++i) {
        if ( ! encontrado(ref, v.elm[i]) ) {
            v.elm[j] = v.elm[i];
            ++j;
        }
    }
    v.nelms = j;
}
//-----------------------------------------
void inicializar(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, int x)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
//-----------------------------------------
void leer(Lista& v)
{
```

```
    int n;
    inicializar(v);
    cout <<"Introduzca una secuencia de números terminada en cero: " << endl;
    cin >> n;
    while (n != 0) {
        anyadir(v, n);
        cin >> n;
    }
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Lista v1, ref;
    leer(v1);
    leer(ref);
    cout << "Lista: " ;
    escribir(v1);
    cout << "Referencia: " ;
    escribir(ref);
    eliminar_numeros(v1, ref);
    cout << "Resultado: " ;
    escribir(v1);
}
```

7. Diseñe un programa que lea de teclado una sucesión indefinida de números enteros acabada en 0, y calcule la media de los M **mayores** números **distintos** de la sucesión, siendo M una constante con valor 10. El número de elementos distintos de la sucesión **puede ser menor** que M, en cuyo caso calculará la media de los números distintos introducidos.

Solución

```
#include <iostream>
#include <cassert>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 10;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
void ini(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, int x)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
//-----------------------------------------
int buscar(const Lista& v, int x)
{
    int i = 0;
    while ((i < v.nelms)&&(x != v.elm[i])) {
        ++i;
    }
    return i;
```

```
}
//-----------------------------------------
int buscar_pos_menor(const Lista& v)
{
    int pos_menor = 0;
    for (int i = pos_menor+1; i < v.nelms; ++i) {
        if (v.elm[i] < v.elm[pos_menor]) {
            pos_menor = i;
        }
    }
    return pos_menor;
}
//-----------------------------------------
void insertar(Lista& v, int x)
{
    int p = buscar(v, x);
    if (p == v.nelms) {
        if (v.nelms < int(v.elm.size())) {
            anyadir(v, x);
        } else {
            int i = buscar_pos_menor(v);
            if (x > v.elm[i]) {
                v.elm[i] = x;
            }
        }
    }
}
//-----------------------------------------
double media(const Lista& v)
{
    int suma = 0;
    for (int i = 0; i < v.nelms; ++i) {
        suma += v.elm[i];
    }
    return double(suma) / double(v.nelms);
}
//-----------------------------------------
void leer(Lista& v)
{
    //----------
    ini(v);
    //----------
    cout <<"Introduzca una secuencia de números terminada en cero (0): ";
    int num;
    cin >> num;
    while (num != 0) {
        insertar(v, num);
        cin >> num;
    }
}
//-----------------------------------------
int main()
{
    Lista v;
    leer(v);
    cout << "Media de los "<<v.nelms<<" mayores: " << media(v) <<endl;
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 7. Arrays

Ejercicios Complementarios

1. Se dispone de un array de 10 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. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int NELMS = 5;
typedef array<int, NELMS> Vector;
//-----------------------------------------
int mayor(int a, int b)
{
    int res = a;
    if (b > res) {
        res = b;
    }
    return b;
}
//-----------------------------------------
// Busca un elemento que sea mayor que el mínimo del array
// Al menos hay dos elementos iguales
// Al menos hay dos elementos distintos
int buscar_propiedad(const Vector& v)
{
    int i = 1;
    while (v[0] == v[i]) {
        ++i;
    }
    return mayor(v[0], v[i]);
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<< int(v.size()) <<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << ' ';
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    Vector v;
    leer(v);
    int e = buscar_propiedad(v);
    cout << "El elemento "<< e <<" es mayor que el mínimo de la lista"<< endl;
    cout << "Lista: ";
    escribir(v);
}
```

2. Diseñe una función para buscar la posición que ocupa un elemento dado (`x`) en un array de enteros (de tamaño 5). Si el elemento no se encuentra en el array, entonces devolverá un valor fuera del rango válido para ese array. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Téngase en consideración la evaluación en *cortocircuito* de expresiones lógicas.

```
int buscar (const Vector& v, int x);
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int NELMS = 5;
typedef array<int, NELMS> Vector;
//-----------------------------------------
int buscar(const Vector& v, int x)
{
    int i = 0;
    // La evaluación en CORTOCIRCUITO hace que la siguiente expresión
    // sea correcta, ya que se comprueba que el valor de (i) sea
    // correcto antes de que se utilice como índice del array
    while ((i < int(v.size())) && (x != v[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << ' ';
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    Vector v;
    leer(v);
    cout << "Introduzca elemento a buscar: ";
    int x;
    cin >> x;
    int i = buscar(v, x);
    if (i >= int(v.size())) {
        cout << "El elemento " << x <<" no se encuentra en la lista" << endl;
    } else {
        cout << "El elemento " << x <<" se encuentra en la posición "<<i
             <<" en la lista" << endl;
    }
    cout << "Lista: ";
    escribir(v);
}
```

3. Diseñe una función para buscar la posición del menor elemento de un array de enteros (de tamaño 5). Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
int buscar_pos_menor (const Vector& v);
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int NELMS = 5;
typedef array<int, NELMS> Vector;
//-----------------------------------------
```

```
int buscar_pos_menor(const Vector& v)
{
    int pos_menor = 0;
    for (int i = pos_menor+1; i < int(v.size()); ++i) {
        if (v[i] < v[pos_menor]) {
            pos_menor = i;
        }
    }
    return pos_menor;
}
//-----------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<< int(v.size()) <<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------
void escribir(const Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << ' ';
    }
    cout << endl;
}
//-----------------------------------
int main()
{
    Vector v;
    leer(v);
    int i = buscar_pos_menor(v);
    cout << "El menor elemento se encuentra en la posición "<<i<<" en la lista"
         << endl;
    cout << "Lista: ";
    escribir(v);
}
```

4. Diseñe una función para buscar el menor elemento de un array de enteros (de tamaño 5). Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
int buscar_menor (const Vector& v);
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------
const int NELMS = 5;
typedef array<int, NELMS> Vector;
//-----------------------------------
int buscar_pos_menor(const Vector& v)
{
    int pos_menor = 0;
    for (int i = pos_menor+1; i < int(v.size()); ++i) {
        if (v[i] < v[pos_menor]) {
            pos_menor = i;
        }
    }
    return pos_menor;
}
//-----------------------------------
inline int buscar_menor(const Vector& v)
{
    int pos_menor = buscar_pos_menor(v);
    return v[pos_menor];
}
//-----------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<< int(v.size()) <<" números: ";
```

```
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << ' ';
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    Vector v;
    leer(v);
    int menor = buscar_menor(v);
    cout << "El menor elemento de la lista es "<< menor << endl;
    cout << "Lista: ";
    escribir(v);
}
```

5. Diseña una función booleana que reciba dos arrays de números enteros (de tamaño 5), y devuelva `true` si ambos contienen los mismos elementos y en el mismo orden relativo, suponiendo que el primer elemento sigue al último, y `false` en otro caso. Podemos suponer que cada elemento del array aparece a lo sumo una vez. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
bool iguales_relativos (const Vector& v1, const Vector& v2);
```

Por ejemplo, si la entrada fuese la siguiente, la función devolvería `true`.

```
v1: [ 1, 3, 4, 9, 6 ]
v2: [ 4, 9, 6, 1, 3 ]
```

Solución

```
#include <iostream>
#include <cassert>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 5;
//-----------------------------------------
typedef array<int, MAX> Vector;
//-----------------------------------------
int buscar(const Vector& v, int x)
{
    int i = 0;
    while ((i < int(v.size())) && (x != v[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
bool iguales_relativos(const Vector& v1, const Vector& v2, int i2i)
{
    assert(i2i < int(v2.size()));
    int i2 = i2i;
    int i1 = 0;
    while ((i1 < int(v1.size()))&&(v1[i1] == v2[i2])) {
        i2 = (i2 + 1) % int(v2.size());
        ++i1;
    }
    return (i1 >= int(v1.size()));
}
//-----------------------------------------
bool iguales_relativos(const Vector& v1, const Vector& v2)
{
    int i2 = buscar(v2, v1[0]);
    return (i2 < int(v2.size())) && iguales_relativos(v1, v2, i2);
```

```
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
    }
}
//-----------------------------------------
void escribir(const Vector& v)
{
    cout << "[ ";
    for (int i = 0; i < int(v.size()); ++i) {
        cout << v[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Vector v1, v2;
    leer(v1);
    leer(v2);
    escribir(v1);
    escribir(v2);
    if (iguales_relativos(v1, v2)) {
        cout << "Los vectores son iguales relativos"<< endl;
    } else {
        cout << "Los vectores NO son iguales relativos"<< endl;
    }
}
```

6. Escribe un procedimiento que pueda insertar el valor de una variable `x` en una lista `v` ordenada de forma creciente, de forma que dicha lista continúe estando ordenada. La lista tendrá un número de elementos válidos que podrá ser menor que el número total de elementos del array (de máximo 20 elementos). Si en el momento de la inserción el número de elementos válidos coincide con el número total de elementos del array, el elemento de mayor valor desaparecerá. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
void insertar (Lista& v, int x);
```

Solución

```
#include <iostream>
#include <cassert>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 20;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
// INICIALIZAR LISTA
//-----------------------------------------
void ini(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
// INSERTAR
//-----------------------------------------
int buscar_posicion(const Lista& v, int x)
{
    int i = 0;
    while ((i < v.nelms) && (x > v.elm[i])) {
        ++i;
```

```
    }
    return i;
}
//-----------------------------------------
void abrir_hueco(Lista& v, int pos)
{
    if (v.nelms < int(v.elm.size())) {
        ++v.nelms;
    }
    for (int i = v.nelms-1; i > pos; --i) {
        v.elm[i] = v.elm[i - 1];
    }
}
//-----------------------------------------
void insertar(Lista& v, int x)
{
    int pos = buscar_posicion(v, x);
    if (pos < int(v.elm.size())) {
        abrir_hueco(v, pos);
        v.elm[pos] = x;
    }
}
//-----------------------------------------
// LEER ESCRIBIR
//-----------------------------------------
void leer(Lista& v)
{
    //----------
    ini(v);
    //----------
    cout <<"Introduzca el número de elementos (menor o igual que "
         << int(v.elm.size()) <<") de la lista: ";
    int n_elms;
    cin >> n_elms;
    if (n_elms > int(v.elm.size())) {
        n_elms = int(v.elm.size());
    }
    cout <<"Introduzca "<<n_elms<<" numeros: ";
    for (int i = 0; i < n_elms; ++i) {
        int x;
        cin >> x;
        insertar(v, x);
    }
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
bool esta_ordenado(const Lista& v)
{
    bool ok;
    if (v.nelms == 0) {
        ok = true;
    } else {
        int i = 0;
        while ((i < v.nelms-1)&&(v.elm[i] <= v.elm[i+1])) {
            ++i;
        }
        ok = (i >= v.nelms-1);
    }
    return ok;
}
//-----------------------------------------
int main()
{
    Lista v;
    leer(v);
    if (esta_ordenado(v)) {
```

```
        cout << "Lista ordenada: "<<endl;
        escribir(v);
    } else {
        cout << "Error, la lista no se ha ordenado correctamente"<<endl;
        escribir(v);
    }
}
```

7. Diseña un procedimiento que tome como parámetros de entrada dos listas ordenadas (de máximo 20 elementos) y devuelva en un tercer parámetro de salida el resultado de realizar la mezcla ordenada de ambos, de tal forma que esta lista resultado también esté ordenada. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

```
void mezclar (const Lista& v1, const Lista& v2, Lista& v3);
```

Solución

```
#include <iostream>
#include <cassert>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 20;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
// MEZCLA ORDENADA
//-----------------------------------------
// v1 y v2 están ordenados
void mezcla(const Lista& v1, const Lista& v2, Lista& v3)
{
    v3.nelms = v1.nelms + v2.nelms;
    if (v3.nelms > int(v3.elm.size())) {
        v3.nelms = int(v3.elm.size());
    }
    int i1 = 0;
    int i2 = 0;
    for (int i3 = 0; i3 < v3.nelms; ++i3) {
        if ((i2 >= v2.nelms)||(i1 < v1.nelms && v1.elm[i1] <= v2.elm[i2])) {
            v3.elm[i3] = v1.elm[i1];
            ++i1;
        } else {
            v3.elm[i3] = v2.elm[i2];
            ++i2;
        }
    }
}
//-----------------------------------------
bool esta_ordenado(const Lista& v)
{
    bool ok;
    if (v.nelms == 0) {
        ok = true;
    } else {
        int i = 0;
        while ((i < v.nelms-1)&&(v.elm[i] <= v.elm[i+1])) {
            ++i;
        }
        ok = (i >= v.nelms-1);
    }
    return ok;
}
//-----------------------------------------
// LEER ESCRIBIR
//-----------------------------------------
void leer(Lista& v)
{
    cout <<"Introduzca el número de elementos (menor o igual que "
```

```
          << int(v.elm.size()) <<") del lista: ";
    cin >> v.nelms;
    if (v.nelms > int(v.elm.size())) {
        v.nelms = int(v.elm.size());
    }
    cout <<"Introduzca "<<v.nelms<<" números: ";
    for (int i = 0; i < v.nelms; ++i) {
        cin >> v.elm[i];
    }
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
int main()
{
    Lista v1, v2, v3;
    cout << "Introduzca ORDENADOS los elementos de las listas"<<endl;
    leer(v1);
    leer(v2);
    if (esta_ordenado(v1)&&esta_ordenado(v2)) {
        mezcla(v1, v2, v3);
        cout << "Mezcla ordenada: ";
        escribir(v3);
    } else {
        cout << "Error, las listas no estan ordenadas"<<endl;
    }
}
```

8. Escribe un programa que lea una sucesión de 10 números enteros, encuentre el valor máximo y lo imprima junto con el número de veces que aparece, y las posiciones en que este ocurre. El proceso se repite con el resto de la sucesión hasta que no quede ningún elemento por tratar. Por ejemplo, para la siguiente entrada:

```
7 10 143 10 52 143 72 10 143 7
```

producirá la siguiente salida:

```
143 aparece 3 veces, en posiciones 3 6 9
72 aparece 1 vez, en posicion 7
52 aparece 1 vez, en posicion 5
10 aparece 3 veces, en posiciones 2 4 8
7 aparece 2 veces, en posiciones 1 10
```

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 10;
typedef array<int, MAX> Vector;
typedef array<bool, MAX> Valido;
//-----------------------------------------
void ini(Valido& x)
{
    for (int i = 0; i < int(x.size()); ++i) {
        x[i] = true;
    }
}
//-----------------------------------------
void anular(const Vector& v, Valido& x, int num)
{
    for (int i = 0; i < int(v.size()); ++i) {
        if (v[i] == num) {
            x[i] = false;
        }
    }
```

```
}
//-----------------------------------------
int buscar_primer_valido(const Valido& x)
{
    int i = 0;
    while ((i < int(x.size()))&&(! x[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
int buscar_pos_mayor(const Vector& v, const Valido& x)
{
    int pos_mayor = buscar_primer_valido(x);
    if (pos_mayor < int(v.size())) {
        for (int i = pos_mayor+1; i < int(v.size()); ++i) {
            if ((x[i])&&(v[i] > v[pos_mayor])) {
                pos_mayor = i;
            }
        }
    }
    return pos_mayor;
}
//-----------------------------------------
int frecuencia(const Vector& v, int num)
{
    int cnt = 0;
    for (int i = 0; i < int(v.size()); ++i) {
        if (v[i] == num) {
            ++cnt;
        }
    }
    return cnt;
}
//-----------------------------------------
void escribir_posiciones(const Vector& v, int num)
{
    for (int i = 0; i < int(v.size()); ++i) {
        if (v[i] == num) {
            cout << (i+1) << ' ';
        }
    }
}
//-----------------------------------------
void procesar_mayor(const Vector& v, Valido& x, int num)
{
    int n_veces = frecuencia(v, num);
    cout << num << " aparece " << n_veces;
    if (n_veces == 1) {
        cout << " vez, en posición ";
    } else {
        cout << " veces, en posiciones ";
    }
    escribir_posiciones(v, num);
    anular(v, x, num);
    cout << endl;
}
//-----------------------------------------
void estadisticas(const Vector& v)
{
    Valido x;
    ini(x);
    int i = buscar_pos_mayor(v, x);
    while (i < int(v.size())) {
        procesar_mayor(v, x, v[i]);
        i = buscar_pos_mayor(v, x);
    }
}
//-----------------------------------------
void leer(Vector& v)
{
    cout <<"Introduzca "<<int(v.size())<<" números: ";
    for (int i = 0; i < int(v.size()); ++i) {
        cin >> v[i];
```

```
    }
}
//-----------------------------------------
int main()
{
    Vector v;
    leer(v);
    estadisticas(v);
}
```

9. Un determinado juego de cartas consiste en lo siguiente: consideremos un mazo de N cartas, siendo N un número triangular, esto es, $N = 1 + 2 + 3 + ... + k$ para algún $k \in \mathbb{N}$. Se reparte la totalidad de las N cartas en un número arbitrario de montones, cada uno de ellos con una cantidad arbitraria de cartas.

   El lote de montones se puede reorganizar así: se toma una carta de cada montón (con lo que desaparecerán los montones con una sóla carta), y con todas ellas se forma uno nuevo, que se agrega al final de la lista de montones. Por ejemplo, la operación descrita transforma los montones de 1, 8, 1 y 5 cartas, en otros montones de 7, 4 y 4 respectivamente:

   `[ 1 8 1 5 ]` $\Rightarrow$ `[ 0 7 0 4 4 ]` $\Rightarrow$ `[ 7 4 4 ]`

   El desarrollo del juego consiste en llevar a cabo la reorganización descrita anteriormente cuantas veces sea necesario hasta que haya un montón de 1 carta, otro de 2 cartas..., otro de $k-1$ cartas y, finalmente, otro de k cartas. Por ejemplo, partiendo de la situación `[ 5 7 3 ]`, las reorganizaciones sucesivas evolucionan como sigue:

```
[ 5 7 3 ]
[ 4 6 2 3 ]
[ 3 5 1 2 4 ]
[ 2 4 1 3 5 ]
[ 1 3 2 4 5 ]
[ 2 1 3 4 5 ]
[ 1 2 3 4 5 ]
```

   Realice un programa que lea de teclado el número de montones inicial y el número inicial de cartas de cada montón. El número montones máximo k no podrá ser mayor de 15. El programa comprobará que los valores iniciales de los montones se corresponden con un número triangular (un número N es triangular si existe un $k \in \mathbb{N}$ tal que $N = (1 + k)k/2$, es decir, si $k = \frac{-1+\sqrt{1+8N}}{2} \in \mathbb{N}$), y deberá mostrar por pantalla la evolución de los montones hasta finalizar el juego.

Solución

```
#include <iostream>
#include <cassert>
#include <cmath>
#include <array>
using namespace std;
//-----------------------------------------
const double ERROR_PRECISION = 1e-10;
const int MAX = 15;
//-----------------------------------------
typedef array<int, MAX> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
// LISTA
//-----------------------------------------
void ini(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, int x)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
```

```
//-----------------------------------------
// REORGANIZAR
//-----------------------------------------
void decrementar(Lista& v)
{
    for (int i = 0; i < v.nelms; ++i) {
        v.elm[i] = v.elm[i] - 1;
    }
}
//-----------------------------------------
void compactar(Lista& v)
{
    int j = 0;
    for (int i = 0; i < v.nelms; ++i) {
        if (v.elm[i] > 0) {
            v.elm[j] = v.elm[i];
            ++j;
        }
    }
    v.nelms = j;
}
//-----------------------------------------
void reorganizar(Lista& v)
{
    int nuevo_monton = v.nelms;
    decrementar(v);
    compactar(v);
    anyadir(v, nuevo_monton);
}
//-----------------------------------------
// OTRA POSIBILIDAD
//-----------------------------------------
// void decrementar_compactar(Lista& v)
// {
//  int j = 0;
//  for (int i = 0; i < v.nelms; ++i) {
//      if (v.elm[i] > 1) {
//          v.elm[j] = v.elm[i] - 1;
//          ++j;
//      }
//  }
//  v.nelms = j;
// }
// //-----------------------------------------
// void reorganizar(Lista& v)
// {
//  int nuevo_monton = v.nelms;
//  decrementar_compactar(v);
//  anyadir(v, nuevo_monton);
// }
//-----------------------------------------
int sumar(const Lista& v)
{
    int suma = 0;
    for (int i = 0; i < v.nelms; ++i) {
        suma += v.elm[i];
    }
    return suma;
}
//-----------------------------------------
bool es_triangular(const Lista& v)
{
    int suma = sumar(v);
    double k = (-1+sqrt(1+8*suma)) / 2.0;
    return (k-double(int(k)) < ERROR_PRECISION);
}
//-----------------------------------------
bool es_ascendente(const Lista& v)
{
    int i = 0;
    while ((i < v.nelms)&&(v.elm[i] == (i+1))) {
        ++i;
    }
    return (i >= v.nelms);
```

```
}
//-----------------------------------------
// LEER ESCRIBIR
//-----------------------------------------
void leer(Lista& v)
{
    //----------
    ini(v);
    //----------
    cout <<"Introduzca el número de montones (menor o igual que "
         << int(v.elm.size()) <<"): ";
    int n_elms;
    cin >> n_elms;
    if (n_elms > int(v.elm.size())) {
        n_elms = int(v.elm.size());
    }
    for (int i = 0; i < n_elms; ++i) {
        cout <<"Introduzca el número de cartas del monton "<<(i+1)<<": ";
        int x;
        cin >> x;
        anyadir(v, x);
    }
}
//-----------------------------------------
void escribir(const Lista& v)
{
    cout << "[ ";
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << "]"<< endl;
}
//-----------------------------------------
// JUEGO
//-----------------------------------------
void juego()
{
    Lista v;
    leer(v);
    if (! es_triangular(v)) {
        cout << "Error, el número no es triangular"<<endl;
    } else {
        while (!es_ascendente(v)) {
            escribir(v);
            reorganizar(v);
        }
        escribir(v);
    }
}
//-----------------------------------------
int main()
{
    juego();
}
```

10. Diseñe un programa para contar y mostrar el número total de ocurrencias (frecuencia) de cada letra minúscula en una secuencia de caracteres leída por teclado hasta el carácter punto (`'.'`).

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const int N_LETRAS = ('z' - 'a') + 1;
typedef array<int, N_LETRAS> Cnt;
//-----------------------------------------
// convierte de letra a índice
int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
```

```
        res = (letra - 'a');
    }
    return res;
}
//-----------------------------------------
// convierte de índice a letra
inline char idx_letra(int idx)
{
    return char('a' + idx);
}
//-----------------------------------------
void ini(Cnt& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i] = 0;
    }
}
//-----------------------------------------
void estadisticas_texto(Cnt& cnt)
{
    ini(cnt);
    cout << "Introduzca un texto hasta punto ('.'): ";
    char c;
    cin>>ws;
    cin.get(c);
    while (c != '.') {
        int idx = letra_idx(c);
        if (idx < int(cnt.size())) {
            ++cnt[idx];
        }
        cin.get(c);
    }
}
//-----------------------------------------
void mostrar_estadisticas(const Cnt& cnt)
{
    for (int i = 0; i < int(cnt.size()); ++i) {
        if (i%5 == 0) {
            cout << endl;
        } else {
            cout << "\t";
        }
        cout << idx_letra(i) << ": " << cnt[i];
    }
    cout << endl;
}
//-----------------------------------------
int main()
{
    Cnt cnt;
    estadisticas_texto(cnt);
    mostrar_estadisticas(cnt);
}
```

11. Diseñe un programa que muestra la frecuencia con que aparecen en una lista dada los distintos valores que la pudieran formar. Por ejemplo, si los valores de una lista pueden estar comprendidos entre 0 y 9, y la lista está formada por:

    ```
    6 4 4 1 9 7 5 6 4 2 3 9 5 6 4
    ```

    el programa mostrará:

    ```
    0
    1 *
    2 *
    3 *
    4 ****
    5 **
    6 ***
    7 *
    8
    9 **
    ```

Esto indica que el 0 y el 8 no aparecen ninguna vez, el 1, 2, 3 y 7 aparecen una vez, el 5 y 9 dos veces, etc. Escriba un programa que lea una lista de números comprendidos entre 0 y 9 (la lista acabará cuando se lea un número negativo, y a priori no se puede determinar cuantos números contiene) e imprima por pantalla una gráfica como la anterior.

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 10;
typedef array<int, MAX> Vector;
//-----------------------------------------
void ini(Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i] = 0;
    }
}
//-----------------------------------------
void leer(Vector& v)
{
    ini(v);
    cout << "Introduzca números (hasta negativo): ";
    int num;
    cin>>num;
    while (num >= 0 && num < int(v.size())) {
        ++v[num];
        cin>>num;
    }
}
//-----------------------------------------
void imprimir_asteriscos(int n)
{
    for (int i = 0; i < n; ++i) {
        cout << "*";
    }
}
//-----------------------------------------
void grafica(const Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        cout << i << " ";
        imprimir_asteriscos(v[i]);
        cout << endl;
    }
}
//-----------------------------------------
int main()
{
    Vector v;
    leer(v);
    grafica(v);
}
```

12. Un histograma es una gráfica que muestra la frecuencia con que aparecen en una lista dada los distintos valores que la pudieran formar. Por ejemplo, si los valores de una lista pueden estar comprendidos entre 0 y 9, y la lista está formada por:

```
6 4 4 1 9 7 5 6 4 2 3 9 5 6 4
```

su histograma vertical será:

```
        *
        *   *
        * * *     *
  * * * * * * *   *
------------------
0 1 2 3 4 5 6 7 8 9
```

Esto indica que el 0 y el 8 no aparecen ninguna vez, el 1, 2, 3 y 7 aparecen una vez, el 5 y 9 dos veces, etc. Escriba un programa que lea una lista de números comprendidos entre 0 y 9 (la lista acabará cuando se lea un número negativo, y a priori no se puede determinar cuantos números contiene) e imprima por pantalla un histograma vertical como el anterior.

Solución

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAX = 10;
typedef array<int, MAX> Vector;
//-----------------------------------------
void ini(Vector& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i] = 0;
    }
}
//-----------------------------------------
void leer(Vector& v)
{
    ini(v);
    cout << "Introduzca números (hasta negativo): ";
    int num;
    cin>>num;
    while (num >= 0 && num < int(v.size())) {
        ++v[num];
        cin>>num;
    }
}
//-----------------------------------------
int buscar_mayor(const Vector& v)
{
    int mayor = v[0];
    for (int i = 1; i < int(v.size()); ++i) {
        if (v[i] > mayor) {
            mayor = v[i];
        }
    }
    return mayor;
}
//-----------------------------------------
void imprimir_linea(int n)
{
    for (int i = 0; i < n; ++i) {
        cout << "-";
    }
    cout << endl;
}
//-----------------------------------------
void imprimir_indices(int n)
{
    for (int i = 0; i < n; ++i) {
        cout << i << " ";
    }
    cout << endl;
}
//-----------------------------------------
void imprimir_asteriscos(const Vector& v, int fila)
{
    for (int i = 0; i < int(v.size()); ++i) {
        if (v[i] >= fila) {
            cout << "* ";
        } else {
            cout << "  ";
        }
    }
    cout << endl;
}
//-----------------------------------------
void histograma(const Vector& v)
{
```

```
    int mayor = buscar_mayor(v);
    for (int f = mayor; f > 0; --f) {
        imprimir_asteriscos(v, f);
    }
    imprimir_linea(2*int(v.size())-1);
    imprimir_indices(int(v.size()));
}
//----------------------------------------
int main()
{
    Vector v;
    leer(v);
    histograma(v);
}
```

13. Diseñe un programa que lea por teclado las temperaturas medias de los doce meses del año y calcule la temperatura media anual. El programa deberá utilizar un tipo entero para representar los meses del año y considerar que las temperaturas de los meses se pueden introducir en cualquier orden. Como salida deberá imprimir la temperatura de cada mes y la temperatura media anual. Por ejemplo, para la siguiente entrada:

```
Mes: Marzo
Temperatura: 15.8
Mes: Enero
Temperatura: 12.5
Mes: Febrero
Temperatura: 13.5
...
```

mostrará la siguiente salida:

```
Enero: 12.5
Febrero: 13.5
Marzo: 15.8
...
Temperatura media anual: 17.3
```

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//----------------------------------------
const int N_MESES = 12;
typedef array<string, N_MESES> MesStr;
const MesStr MES_STR = {{
    "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio",
    "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"
}};
//---------
typedef array<double, N_MESES> MesTmp;
typedef array<bool, N_MESES> MesOk;
//----------------------------------------
inline char minuscula(char c)
{
    char res = c;
    if (c >= 'A' && c <= 'Z') {
        res = char('a' + (c - 'A'));
    }
    return res;
}
//----------------------------------------
bool iguales(const string& s1, const string& s2)
{
    bool res = false;
    if (int(s1.size()) == int(s2.size())) {
        int i = 0;
        while ((i < int(s1.size())) && (minuscula(s1[i]) == minuscula(s2[i]))) {
            ++i;
        }
        res = (i >= int(s1.size()));
```

```
    }
    return res;
}
//-----------------------------------------
int buscar(const MesStr& v, const string& x)
{
    int i = 0;
    while ((i < int(v.size())) && ! iguales(x, v[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
inline void escribir_mes(int m)
{
    assert(0 <= m && m < N_MESES);
    cout << MES_STR[int(m)];
}
//-----------------------------------------
inline void leer_mes_simple(int& m)
{
    string str;
    cin >> str;
    m = buscar(MES_STR, str);
}
//-----------------------------------------
void leer_mes(int& m)
{
    cout << "Mes: ";
    leer_mes_simple(m);
    while (m >= N_MESES) {
        cout << "Error. Introduzca Mes: ";
        leer_mes_simple(m);
    }
}
//-----------------------------------------
void leer_tmp(double& tmp)
{
    cout << "Temperatura: ";
    cin >> tmp;
}
//-----------------------------------------
void init(MesOk& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i] = false;
    }
}
//-----------------------------------------
void leer_tmp_anyo(MesTmp& v)
{
    int nmes = 0;
    MesOk mok;
    init(mok);
    while (nmes != int(v.size())) {
        int m;
        leer_mes(m);
        if (mok[m]) {
            cout <<"Error, la temperatura para ese mes ya existe"<<endl;
        } else {
            mok[m] = true;
            ++nmes;
            leer_tmp(v[m]);
        }
    }
}
//-----------------------------------------
double media_tmp_anyo(const MesTmp& v)
{
    double suma = 0;
    for (int i = 0; i < int(v.size()); ++i) {
        suma += v[i];
    }
    return suma / double(v.size());
```

```
}
//-----------------------------------------
void escr_tmp_anyo(const MesTmp& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        escribir_mes(i);
        cout << ": " << v[i] << endl;
    }
}
//-----------------------------------------
int main()
{
    MesTmp vt;
    leer_tmp_anyo(vt);
    escr_tmp_anyo(vt);
    double media = media_tmp_anyo(vt);
    cout << "Temperatura media anual: "<< media<<endl;
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 8. Arrays Multidimensionales

Laboratorio

1. Un array bidimensional `a` de `M` filas y `M` columnas, es simétrico si sus elementos satisfacen la condición $a[i][j] = a[j][i]$ para todo i, j. Escribe una función que determine si un array de ese tipo es simétrico, para un tamaño de `M` igual a `5`. Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

Solución

```
#include <iostream>
#include <iomanip>
#include <array>
using namespace std;
//-----------------------------------------
const int MAXCOL = 5;
const int MAXFIL = MAXCOL;
typedef array<int, MAXCOL> Fila;
typedef array<Fila, MAXFIL> Matriz;
//-----------------------------------------
bool es_simetrica(const Matriz& m)
{
    bool ok = true;
    for (int f = 0; ok && f < int(m.size()); ++f) {
        for (int c = 0; ok && c < f; ++c) {
            ok = m[f][c] == m[c][f];
        }
    }
    return ok;
}
//-----------------------------------------
void escribir(const Matriz& m)
{
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cout << setw(3) << m[f][c] << ' ';
        }
        cout << endl;
    }
}
//-----------------------------------------
void leer(Matriz& m)
{
    cout << "Introduzca " << int(m.size()) << " filas de "
         << int(m[0].size()) << " números" << endl;
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cin >> m[f][c];
        }
    }
}
//-----------------------------------------
int main()
{
    Matriz m;
    leer(m);
    cout << "La matriz"<<endl;
    escribir(m);
    if (es_simetrica(m)) {
        cout <<"SI ";
    } else {
        cout <<"NO ";
    }
    cout << "es simétrica"<<endl;
}
```

2. Escribe una función que encuentre el elemento mayor de un array de dos dimensiones dado (de `5` filas y `7` columnas). Además, diseñe el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento.

Solución

```
#include <iostream>
#include <iomanip>
#include <array>
using namespace std;
//-----------------------------------------
const int MAXCOL = 7;
const int MAXFIL = 5;
typedef array<int, MAXCOL> Fila;
typedef array<Fila, MAXFIL> Matriz;
//-----------------------------------------
int buscar_mayor(const Matriz& m)
{
    int mayor = m[0][0];
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            if (m[f][c] > mayor) {
                mayor = m[f][c];
            }
        }
    }
    return mayor;
}
//-----------------------------------------
void escribir(const Matriz& m)
{
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cout << setw(3) << m[f][c] << ' ';
        }
        cout << endl;
    }
}
//-----------------------------------------
void leer(Matriz& m)
{
    cout << "Introduzca " << int(m.size()) << " filas de "
         << int(m[0].size()) << " números" << endl;
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cin >> m[f][c];
        }
    }
}
//-----------------------------------------
int main()
{
    Matriz m;
    leer(m);
    int mayor = buscar_mayor(m);
    cout << "El número "<< mayor << " es el mayor elemento de la matriz"<<endl;
    escribir(m);
}
```

3. Diseñe un programa que lea dos matrices de números reales de tamaño máximo 10×10 (aunque podrán tener un tamaño menor) y muestre el resultado de multiplicar ambas matrices. Las matrices se almacenarán en estructuras de datos que almacenen las dimensiones actuales de las matrices, y almacenen los datos en arrays de dos dimensiones.

Solución

```
#include <iostream>
#include <array>
#include <cassert>
using namespace std;
// -- Constantes ---------
const int MAX = 10;
// -- Tipos --------------
typedef array<double, MAX> Fila;
typedef array<Fila, MAX> Tabla;
struct Matriz {
   int n_fil;
```

```
    int n_col;
    Tabla datos;
};
//-----------------------------------
void leer_matriz (Matriz& m)
{
    cout << "Dimensiones?: ";
    cin >> m.n_fil >> m.n_col;
    cout << "Escribe valores fila a fila:" << endl;
    for (int f = 0; f < m.n_fil; ++f) {
        for (int c = 0; c < m.n_col; ++c) {
            cin >> m.datos[f][c];
        }
    }
}
//-----------------------------------
void escribir_matriz (const Matriz& m)
{
    for (int f = 0; f < m.n_fil; ++f) {
        for (int c = 0; c < m.n_col; ++c) {
            cout << m.datos[f][c] << " ";
        }
        cout << endl;
    }
}
//-----------------------------------
double suma_fila_por_col (const Matriz& x, const Matriz& y, int f, int c)
{
    assert(x.n_col == y.n_fil); // PRECOND
    double suma = 0.0;
    for (int k = 0; k < x.n_col; ++k) {
        suma += x.datos[f][k] * y.datos[k][c];
    }
    return suma;
}
//-----------------------------------
void mult_matriz (Matriz& m, const Matriz& a, const Matriz& b)
{
    assert(a.n_col == b.n_fil); // PRECOND
    m.n_fil = a.n_fil;
    m.n_col = b.n_col;
    for (int f = 0; f < m.n_fil; ++f) {
        for (int c = 0; c < m.n_col; ++c) {
            m.datos[f][c] = suma_fila_por_col(a, b, f, c);
        }
    }
}
//-----------------------------------
int main ()
{
    Matriz a,b,c;
    leer_matriz(a);
    leer_matriz(b);
    if (a.n_col != b.n_fil) {
        cout << "No se puede multiplicar." << endl;
    } else {
        mult_matriz(c, a, b);
        escribir_matriz(c);
    }
}
```

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

	0	1	$\cdots$	$N-1$	N
0			$\cdots$		
1			$\cdots$		
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$N-1$			$\cdots$		

Una propiedad interesante es que se pueden visitar todas sus casillas haciendo el siguiente recorrido por diagonales. Empezamos por la casilla $(0,0)$ y recorremos la diagonal principal hacia la derecha y hacia abajo hasta llegar a la casilla $(N-1, N-1)$. La siguiente casilla a visitar sería la (N,N) 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 $(0, N)$. Ahora seguimos moviéndonos hacia la derecha y hacia abajo. Pero la siguiente casilla sería la $(1, N+1)$ 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 $(1, 0)$. De nuevo nos movemos hacia la derecha y hacia abajo, hasta alcanzar la casilla $(N-1, N-2)$. La siguiente casilla sería la $(N, N-1)$, pero como nos saldríamos por abajo pasamos a la casilla equivalente de la primera fila $(0, N-1)$. Si se continúa con este proceso se termina visitando todas las casillas del tablero.

Diseñe un procedimiento que dada una constante N (con valor igual a 4) devuelve como parámetro un tablero *N-goro* con sus casillas rellenas con el número correspondiente al momento en que se visitan, así como el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. Por ejemplo, si N es 4, el tablero a devolver sería:

	0	1	2	3	4
0	1	17	13	9	5
1	6	2	18	14	10
2	11	7	3	19	15
3	16	12	8	4	20

Solución

```
#include <iostream>
#include <iomanip>
#include <array>
using namespace std;
//-----------------------------------------
const int MAXFIL = 4;
const int MAXCOL = MAXFIL+1;
typedef array<int, MAXCOL> Fila;
typedef array<Fila, MAXFIL> Matriz;
//-----------------------------------------
void escribir(const Matriz& m)
{
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cout << setw(3) << m[f][c] << ' ';
        }
        cout << endl;
    }
}
//-----------------------------------------
void crear_ngoro(Matriz& m)
{
    int f = 0;
    int c = 0;
    for (int i = 1; i <= int(m.size()*m[0].size()); ++i) {
        m[f][c] = i;
        f = (f+1)%int(m.size());
        c = (c+1)%int(m[0].size());
    }
}
//-----------------------------------------
int main()
{
    Matriz m;
    crear_ngoro(m);
    escribir(m);
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 8. Arrays Multidimensionales

Ejercicios Complementarios

1. Diseñe un programa que tomando como entrada un texto, realice el cálculo de la frecuencia con que aparece cada palabra de dos letras minúsculas en el texto, imprimiendo en pantalla el resultado. Considerando que el número de letras minúsculas es 26 (sin tener en cuenta la letra "ñ").

Solución

```
#include <iostream>
#include <iomanip>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const string FIN_SEC = "fin";
const int N_LETRAS = int('z')-int('a')+1;
typedef array<int, N_LETRAS> FrecLetra;
typedef array<FrecLetra, N_LETRAS> Frec2Letras;
//-----------------------------------------
// convierte de letra a índice
inline int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
        res = (letra - 'a');
    }
    return res;
}
//-----------------------------------------
// convierte de índice a letra
inline char idx_letra(int idx)
{
    return char('a' + idx);
}
//-----------------------------------------
void init(Frec2Letras& fr)
{
    for (int f = 0; f < int(fr.size()); ++f) {
        for (int c = 0; c < int(fr[f].size()); ++c) {
            fr[f][c] = 0;
        }
    }
}
//-----------------------------------------
void inc_frecuencia(const string& s, Frec2Letras& fr)
{
    if (int(s.size()) == 2) {
        int idx_0 = letra_idx(s[0]);
        int idx_1 = letra_idx(s[1]);
        if ((idx_0 < int(fr.size()))&&(idx_1 < int(fr[idx_0].size()))) {
            ++fr[idx_0][idx_1];
        }
    }
}
//-----------------------------------------
void frecuencia_palabras(Frec2Letras& fr)
{
    string palabra;
    init(fr);
    cin >> palabra;
    while (palabra != FIN_SEC){
        inc_frecuencia(palabra, fr);
        cin >> palabra;
    }
}
//-----------------------------------------
// void escribir_frecuencias(const Frec2Letras& fr)
// {
//  cout << "    ";
```

```
//  for (int c = 0; c < int(fr[0].size()); ++c) {
//      cout << setw(2) << idx_letra(c) << " ";
//  }
//  cout << endl;
//  for (int f = 0; f < int(fr.size()); ++f) {
//      cout << " " << idx_letra(f) << ": ";
//      for (int c = 0; c < int(fr[f].size()); ++c) {
//          cout << setw(2) << fr[f][c] << " ";
//      }
//      cout << endl;
//  }
// }
//-----------------------------------------
void escribir_frecuencias(const Frec2Letras& fr)
{
    cout << "Frecuencias de las palabras de 2 letras:"<<endl;
    for (int f = 0; f < int(fr.size()); ++f) {
        for (int c = 0; c < int(fr[f].size()); ++c) {
            if (fr[f][c] > 0) {
                cout <<idx_letra(f)<<idx_letra(c)<<": "<<fr[f][c]<<endl;
            }
        }
    }
}
//-----------------------------------------
int main()
{
    Frec2Letras fr;
    cout << "Introduzca el texto en minúsculas hasta (fin)" << endl;
    frecuencia_palabras(fr);
    escribir_frecuencias(fr);
}
```

2. Se dispone de una determinada zona cuadrangular dividida en celdas (8×8), por algunas de las cuales se puede circular y por otras no. Un ejemplo se muestra en la figura siguiente, en donde las celdas no visitables se marcan con una `X` y las visitables se dejan en blanco.

X	X				X	X	X
		X	X	X	X	X	X
X					X	X	X
X	X	X	X		X	X	X
X	X	X	X		X	X	X
			X				
X	X		X	X	X	X	X
X	X		X	X	X	X	X

Diseñe un procedimiento `recorrido`, el programa y los subprogramas necesarios para probar adecuadamente su funcionamiento. El subprograma `recorrido` toma como parámetros una zona de ese tipo, un valor n y una determinada coordenada (fila y columna de una celda visitable) y determine la celda en la que nos encontraremos tras realizar n movimientos por la zona partiendo de la coordenada dada, teniendo en cuenta:

- Un movimiento consiste en un desplazamiento horizontal o vertical desde una celda a una celda vecina visitable, sin considerar la celda de la que se proviene.
- Cada celda tiene 4 celdas vecinas posibles, teniendo en cuenta sólo vecindad horizontal y vertical (una celda que está en un borde o bordes de la cuadrícula tiene como vecinas a las celdas del borde o bordes opuestos).
- Sólo existirá un camino posible partiendo de la celda dada.
- Si no se puede seguir avanzando sin haber alcanzado los n pasos pedidos, se dará como resultado la celda última en la que nos encontramos.

Así por ejemplo, considerando la zona de la figura anterior, partiendo de la celda $(1, 0)$ y tras realizar 14 movimientos, nos encontraremos en la celda $(5, 2)$. Si por el contrario pretendemos realizar 19 o más movimientos, nos encontraremos en la celda $(0, 4)$.

Solución

```
#include <iostream>
#include <array>
#include <cassert>
using namespace std;
//-----------------------------------------
const int MAXFIL = 8;
const int MAXCOL = MAXFIL;
typedef array<char, MAXCOL> Fila;
typedef array<Fila, MAXFIL> Matriz;
struct Coord {
    int f;
    int c;
};
//-----------------------------------------
const Matriz ZONA = {{
    {{ 'X' , 'X' , ' ' , ' ' , ' ' , 'X' , 'X' , 'X' }},
    {{ ' ' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' , 'X' }},
    {{ 'X' , ' ' , ' ' , ' ' , ' ' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , 'X' , 'X' , ' ' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , 'X' , 'X' , ' ' , 'X' , 'X' , 'X' }},
    {{ ' ' , ' ' , ' ' , 'X' , ' ' , ' ' , ' ' , ' ' }},
    {{ 'X' , 'X' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' }}
}};
//-----------------------------------------
void leer(Coord& c)
{
    cin >> c.f >> c.c;
}
//-----------------------------------------
void escribir(const Coord& c)
{
    cout << "( " << c.f << ", " << c.c << " )";
}
//-----------------------------------------
void mover(const Matriz& m, Coord& c, bool& ok)
{
    int fi = (c.f + 1) % MAXFIL;             // Incremento circular de fila
    int ci = (c.c + 1) % MAXCOL;             // Incremento circular de columna
    int fd = (c.f + MAXFIL - 1) % MAXFIL;    // Decremento circular de fila
    int cd = (c.c + MAXCOL - 1) % MAXCOL;    // Decremento circular de columna
    ok = true;
    if (m[fd][c.c] == ' ') {
        c.f = fd;
    } else if (m[c.f][ci] == ' ') {
        c.c = ci;
    } else if (m[fi][c.c] == ' ') {
        c.f = fi;
    } else if (m[c.f][cd] == ' ') {
        c.c = cd;
    } else {
        ok = false;
    }
}
//-----------------------------------------
void huella(Matriz& m, Coord& c)
{
    assert(m[c.f][c.c] == ' ');
    m[c.f][c.c] = '.';
}
//-----------------------------------------
void recorrido(Matriz& m, int n, Coord& c)
{
    bool ok = true;
    huella(m, c);
    for (int i = 0; ok && i < n; ++i) {
        mover(m, c, ok);
        if (ok) {
            huella(m, c);
        }
    }
}
//-----------------------------------------
```

```
void escribir(const Matriz& m)
{
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cout << m[f][c];
        }
        cout << endl;
    }
}
//-----------------------------------------
void check()
{
    Matriz m = ZONA;
    escribir(m);
    cout << "Introduzca Coordenada de inicio (fila y columna): ";
    Coord c;
    leer(c);
    cout << "Introduzca número de pasos: ";
    int n;
    cin >> n;
    recorrido(m, n, c);
    cout << "Posición final: ";
    escribir(c);
    cout << endl;
    escribir(m);
}
//-----------------------------------------
int main()
{
    check();
}
```

Otra solución alternativa

```
#include <iostream>
#include <array>
using namespace std;
//-----------------------------------------
const int MAXFIL = 8;
const int MAXCOL = MAXFIL;
typedef array<char, MAXCOL> Fila;
typedef array<Fila, MAXFIL> Matriz;
struct Coord {
    int f;
    int c;
};
//-----------------------------------------
const Matriz ZONA = {{
    {{ 'X' , 'X' , ' ' , ' ' , ' ' , 'X' , 'X' , 'X' }},
    {{ ' ' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' , 'X' }},
    {{ 'X' , ' ' , ' ' , ' ' , ' ' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , 'X' , 'X' , ' ' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , 'X' , 'X' , ' ' , 'X' , 'X' , 'X' }},
    {{ ' ' , ' ' , ' ' , 'X' , ' ' , ' ' , ' ' , ' ' }},
    {{ 'X' , 'X' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' }},
    {{ 'X' , 'X' , ' ' , 'X' , 'X' , 'X' , 'X' , 'X' }}
}};
//-----------------------------------------
inline bool iguales(const Coord& c1, const Coord& c2)
{
    return (c1.f == c2.f)&&(c1.c == c2.c);
}
//-----------------------------------------
inline bool distintos(const Coord& c1, const Coord& c2)
{
    return ! iguales(c1, c2);
}
//-----------------------------------------
inline int vabs(int x)
{
    int res;
    if (x < 0) {
        res = int(-x);
```

```
    } else {
        res = int(x);
    }
    return res;
}
//-----------------------------------------
void ini(Coord& c)
{
    c.f = MAXFIL;
    c.c = MAXCOL;
}
//-----------------------------------------
void asg(Coord& c, int ff, int cc)
{
    c.f = ff;
    c.c = cc;
}
//-----------------------------------------
void leer(Coord& c)
{
    int ff, cc;
    cin >> ff >> cc;
    asg(c, ff, cc);
}
//-----------------------------------------
void escribir(const Coord& c)
{
    cout << "( " << c.f << ", " << c.c << " )";
}
//-----------------------------------------
void paso(const Coord& c, int ff, int cc, Coord& n)
{
    n.f = (c.f + MAXFIL + ff) % MAXFIL;
    n.c = (c.c + MAXCOL + cc) % MAXCOL;
}
//-----------------------------------------
inline bool es_visitable(const Matriz& m, const Coord& c)
{
    return (m[c.f][c.c] == ' ');
}
//-----------------------------------------
void mover(const Matriz& m, Coord& ant, Coord& act, bool& ok)
{
    ok = false;
    for (int ff = -1; !ok && ff <= +1; ++ff) {
        for (int cc = -1; !ok && cc <= +1; ++cc) {
            if (vabs(ff)+vabs(cc) == 1) {
                Coord n;
                paso(act, ff, cc, n);
                if (distintos(n, ant) && es_visitable(m, n)) {
                    ok = true;
                    ant = act;
                    act = n;
                }
            }
        }
    }
}
//-----------------------------------------
void recorrido(const Matriz& m, int n, Coord& c)
{
    Coord ant;
    ini(ant);
    bool ok = true;
    for (int i = 0; ok && i < n; ++i) {
        mover(m, ant, c, ok);
    }
}
//-----------------------------------------
void escribir(const Matriz& m)
{
    for (int f = 0; f < int(m.size()); ++f) {
        for (int c = 0; c < int(m[f].size()); ++c) {
            cout << m[f][c];
```

```
        }
        cout << endl;
    }
}
//-------------------------------------
void check()
{
    escribir(ZONA);
    cout << "Introduzca Coordenada de inicio: ";
    Coord c;
    leer(c);
    cout << "Introduzca número de pasos: ";
    int n;
    cin >> n;
    recorrido(ZONA, n, c);
    cout << "Posición final: ";
    escribir(c);
    cout << endl;
}
//-------------------------------------
int main()
{
    check();
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 9. Estructuras de Datos

Laboratorio

1. Diseñe un programa para gestionar una agenda personal, donde la información que se almacena de cada persona es la siguiente: Nombre, y Teléfono. La agenda permitirá realizar las siguientes operaciones:

   a) Añadir los datos de una persona

   b) Acceder a los datos de una persona a partir de su nombre.

   c) Borrar una persona a partir de su nombre.

   d) Modificar los datos de una persona a partir de su nombre.

   e) Listar el contenido completo de la agenda.

   Además, para el trabajo de casa, se debe añadir la siguiente información para cada persona: Dirección, Calle, Número, Piso, Código Postal y Ciudad.

Solución

```
#include <iostream>
#include <string>
#include <cassert>
#include <array>
using namespace std;
// -- Constantes --------
const int MAX_PERSONAS = 50;
// -- Tipos -------------
struct Direccion {
    int num;
    string calle;
    string piso;
    string cp;
    string ciudad;
};
struct Persona {
    string nombre;
    string tel;
    Direccion direccion;
};
// -- Tipos -------------
typedef array<Persona, MAX_PERSONAS> Personas;
struct Agenda {
    int n_pers;
    Personas pers;
};
// Códigos de Error
const int OK = 0;
const int AG_LLENA = 1;
const int NO_ENCONTRADO = 2;
const int YA_EXISTE = 3;
// -- Subalgoritmos -----
void Inicializar (Agenda& ag)
{
    ag.n_pers = 0;
}
// -- Subalgoritmos -----
void Leer_Direccion (Direccion& dir)
{
    cin >> ws;
    getline(cin, dir.calle);
    cin >> dir.num;
    cin >> dir.piso;
    cin >> dir.cp;
    cin >> ws;
    getline(cin, dir.ciudad);
}
//---------------------------
void Leer_Persona (Persona& per)
{
```

```
    cin >> ws;
    getline(cin, per.nombre);
    cin >> per.tel;
    Leer_Direccion(per.direccion);
}
// -- Subalgoritmos -----
void Escribir_Direccion (const Direccion& dir)
{
    cout << dir.calle << ", ";
    cout << dir.num << " ";
    cout << dir.piso << endl;
    cout << dir.cp << " ";
    cout << dir.ciudad << endl;
}
//-----------------------------
void Escribir_Persona (const Persona& per)
{
    cout << per.nombre << endl;
    cout << per.tel << endl;
    Escribir_Direccion(per.direccion);
}
// -- Subalgoritmos -----
// Busca una Persona en la Agenda
// Devuelve su posicion si se encuentra, o bien >= ag.n_pers en otro caso
int Buscar_Persona (const string& nombre, const Agenda& ag)
{
    int i = 0;
    while ((i < ag.n_pers) && (nombre != ag.pers[i].nombre)) {
        ++i;
    }
    return i;
}
//-----------------------------
void Anyadir (Agenda& ag, const Persona& per)
{
    ag.pers[ag.n_pers] = per;
    ++ag.n_pers;
}
//-----------------------------
void Eliminar (Agenda& ag, int pos)
{
    ag.pers[pos] = ag.pers[ag.n_pers - 1];
    --ag.n_pers;
}
// -- Subalgoritmos -----
void Anyadir_Persona (const Persona& per, Agenda& ag, int& ok)
{
    int i = Buscar_Persona(per.nombre, ag);
    if (i < ag.n_pers) {
        ok = YA_EXISTE;
    } else if (ag.n_pers == int(ag.pers.size())) {
        ok = AG_LLENA;
    } else {
        ok = OK;
        Anyadir(ag, per);
    }
}
//-----------------------------
void Borrar_Persona (const string& nombre, Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Eliminar(ag, i);
    }
}
// -- Subalgoritmos -----
void Modificar_Persona (const string& nombre, const Persona& nuevo, Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
```

```
    } else {
        Eliminar(ag, i);
        Anyadir_Persona(nuevo, ag, ok);
    }
}
//-------------------------------
void Imprimir_Persona (const string& nombre, const Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Escribir_Persona(ag.pers[i]);
    }
}
// -- Subalgoritmos -----
void Imprimir_Agenda (const Agenda& ag, int& ok)
{
    for (int i = 0; i < ag.n_pers; ++i) {
        Escribir_Persona(ag.pers[i]);
    }
    ok = OK;
}
//-------------------------------
char Menu ()
{
    char opcion;
    cout << endl;
    cout << "a. - Anadir Persona" << endl;
    cout << "b. - Buscar Persona" << endl;
    cout << "c. - Borrar Persona" << endl;
    cout << "d. - Modificar Persona" << endl;
    cout << "e. - Imprimir Agenda" << endl;
    cout << "x. - Salir" << endl;
    do {
        cout << "Introduzca Opcion: ";
        cin >> opcion;
    } while ( ! (((opcion >= 'a') && (opcion <= 'e')) || (opcion == 'x')));
    return opcion;
}
// -- Subalgoritmos -----
void Escribir_Cod_Error (int cod)
{
    switch (cod) {
    case OK:
        cout << "Operacion correcta" << endl;
        break;
    case AG_LLENA:
        cout << "Agenda llena" << endl;
        break;
    case NO_ENCONTRADO:
        cout << "La persona no se encuentra en la agenda" << endl;
        break;
    case YA_EXISTE:
        cout << "La persona ya se encuentra en la agenda" << endl;
        break;
    }
}
// -- Principal ---------
int main ()
{
    Agenda ag;
    char opcion;
    Persona per;
    string nombre;
    int ok;
    Inicializar(ag);
    do {
        opcion = Menu();
        switch (opcion) {
        case 'a':
            cout << "Introduzca los datos de la Persona" << endl;
            cout << "(nombre, tel, calle, num, piso, cod_postal, ciudad)" << endl;
```

```
            Leer_Persona(per);
            Anyadir_Persona(per, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'b':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            Imprimir_Persona(nombre, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'c':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            Borrar_Persona(nombre, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'd':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            cout << "Nuevos datos de la Persona" << endl;
            cout << "(nombre, tel, calle, num, piso, cod_postal, ciudad)" << endl;
            Leer_Persona(per);
            Modificar_Persona(nombre, per, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'e':
            Imprimir_Agenda(ag, ok);
            Escribir_Cod_Error(ok);
            break;
        }
    } while (opcion != 'x' );
}
```

2. La distancia entre dos palabras en un texto es el número de palabras que aparecen en el texto entre las dos palabras indicadas. Diseñe un programa que lea un texto de longitud INDEFINIDA, en el que sabemos que hay un máximo de 20 palabras distintas, y muestre por pantalla la MÁXIMA distancia entre cada par de palabras repetidas. Por ejemplo, para el siguiente texto de entrada:

   ```
   la casa roja de la esquina es la casa de juan fin
   ```

   mostrará la salida:

   ```
   Distancia entre la: 3
   Distancia entre casa: 6
   Distancia entre de: 5
   ```

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

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const string FIN_SEC = "fin";
struct Distancia {
    string palabra;
    int ult_pos;
    int max_dist;
};
const int MAX_PAL = 20;
typedef array<Distancia, MAX_PAL> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
int buscar(const Lista& v, const string& x)
```

```
{
    int i = 0;
    while ((i < v.nelms) && (x != v.elm[i].palabra)) {
        ++i;
    }
    return i;
}
//-----------------------------------------
void ini(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, const string& x, int pos)
{
    if (v.nelms < int(v.elm.size())) {
        v.elm[v.nelms].palabra = x;
        v.elm[v.nelms].ult_pos = pos;
        v.elm[v.nelms].max_dist = -1;
        ++v.nelms;
    }
}
//-----------------------------------------
void procesar(Lista& v, int pos, const string& palabra)
{
    int i = buscar(v, palabra);
    if (i < v.nelms) {
        int dist = int(pos - v.elm[i].ult_pos) - 1;
        if (dist > v.elm[i].max_dist) {
            v.elm[i].max_dist = dist;
        }
        v.elm[i].ult_pos = pos;
    } else {
        anyadir(v, palabra, pos);
    }
}
//-----------------------------------------
void estadisticas_texto(Lista& v)
{
    int pos = 0;
    ini(v);
    cout << "Introduzca el texto en minúsculas hasta (fin)" << endl;
    string palabra;
    cin >> palabra;
    while (palabra != FIN_SEC){
        ++pos;
        procesar(v, pos, palabra);
        cin >> palabra;
    }
}
//-----------------------------------------
void mostrar_estadisticas(const Lista& v)
{
    for (int i = 0; i < v.nelms; ++i) {
        if (v.elm[i].max_dist >= 0) {
            cout <<"Distancia entre " << v.elm[i].palabra << ": "
                 << v.elm[i].max_dist << endl;
        }
    }
}
//-----------------------------------------
int main()
{
    Lista v;
    estadisticas_texto(v);
    mostrar_estadisticas(v);
}
```

Tema 4: Tipos de Datos Estructurados

Práctica 9. Estructuras de Datos

Ejercicios Complementarios

1. Diseñe un programa 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. Nota: En el texto aparecerán un número máximo de 20 palabras distintas. Por ejemplo, para la entrada:

   ```
   Patron : re
   Texto: creo que iremos a la direccion que nos dieron aunque pienso que
   dicha direccion no es correcta fin
   ```

   mostrará como salida:

   ```
   creo iremos direccion correcta
   ```

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

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const string FIN_SEC = "fin";
const int MAX_PAL = 20;
typedef array<string, MAX_PAL> Datos;
struct Lista {
    int nelms;
    Datos elm;
};
//-----------------------------------------
int buscar(const Lista& v, const string& x)
{
    int i = 0;
    while ((i < v.nelms) && (x != v.elm[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
void ini(Lista& v)
{
    v.nelms = 0;
}
//-----------------------------------------
void anyadir(Lista& v, const string& x)
{
    if ((buscar(v, x) >= v.nelms)&&(v.nelms < int(v.elm.size()))) {
        v.elm[v.nelms] = x;
        ++v.nelms;
    }
}
//-----------------------------------------
int buscar(const string& cadena, const string& patron)
{
    int i = 0;
    while ((i+int(patron.size()) <= int(cadena.size()))
           && (patron != cadena.substr(i, int(patron.size())))) {
        ++i;
    }
    if (i+int(patron.size()) > int(cadena.size())) {
        i = int(cadena.size());
    }
    return i;
}
//-----------------------------------------
```

```
inline bool contiene(const string& palabra, const string& patron)
{
    return buscar(palabra, patron) < int(palabra.size());
}
//-----------------------------------
void leer_palabras(const string& patron, Lista& v)
{
    cout << "Introduzca el texto en minúsculas hasta (fin)" << endl;
    ini(v);
    string palabra;
    cin >> palabra;
    while (palabra != FIN_SEC){
        if (contiene(palabra, patron)) {
            anyadir(v, palabra);
        }
        cin >> palabra;
    }
}
//-----------------------------------
void escribir_palabras(const Lista& v)
{
    for (int i = 0; i < v.nelms; ++i) {
        cout << v.elm[i] << " ";
    }
    cout << endl;
}
//-----------------------------------
int main()
{
    Lista v;
    cout << "Introduzca el patrón en minúsculas: ";
    string patron;
    cin >> patron;
    if (patron != FIN_SEC){
        leer_palabras(patron, v);
        escribir_palabras(v);
    }
}
```

2. 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 programa que lea una secuencia de caracteres terminada en punto (’.’) y muestre por pantalla la máxima distancia entre cada par de letras minúsculas repetidas. Aquellas letras que no se repitan no aparecerán en la salida. Por ejemplo para la secuencia de entrada:

```
abeaddglake.
```

mostrará la siguiente salida:

```
Distancia entre a: 4
Distancia entre d: 0
Distancia entre e: 7
```

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------
const int N_LETRAS = ('z' - 'a') + 1;
struct Estd {
    int max_dist;
    int ult_pos;
};
typedef array<Estd, N_LETRAS> Cnt;
//-----------------------------------
// convierte de letra a índice
int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
        res = (letra - 'a');
```

```
    }
    return res;
}
//-------------------------------------
// convierte de índice a letra
inline char idx_letra(int idx)
{
    return char('a' + idx);
}
//-------------------------------------
void ini(Cnt& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i].max_dist = -1;
        v[i].ult_pos = 0;
    }
}
//-------------------------------------
void procesar(Cnt& cnt, int pos, int idx)
{
    if (idx < int(cnt.size())) {
        if (cnt[idx].ult_pos != 0) {
            int dist = int(pos - cnt[idx].ult_pos) - 1;
            if (dist > cnt[idx].max_dist) {
                cnt[idx].max_dist = dist;
            }
        }
        cnt[idx].ult_pos = pos;
    }
}
//-------------------------------------
void estadisticas_texto(Cnt& cnt)
{
    int pos = 0;
    ini(cnt);
    cout << "Introduzca un texto hasta punto ('.'): ";
    char c;
    cin>>ws;
    cin.get(c);
    while (c != '.') {
        ++pos;
        procesar(cnt, pos, letra_idx(c));
        cin.get(c);
    }
}
//-------------------------------------
void mostrar_estadisticas(const Cnt& cnt)
{
    for (int i = 0; i < int(cnt.size()); ++i) {
        if (cnt[i].max_dist >= 0) {
            cout << "Distancia entre " << idx_letra(i) << ": "
                 << cnt[i].max_dist << endl;
        }
    }
}
//-------------------------------------
int main()
{
    Cnt cnt;
    estadisticas_texto(cnt);
    mostrar_estadisticas(cnt);
}
```

3. Una palabra w es un anagrama de la palabra v, si podemos obtener w cambiando el orden de las letras de v. Por ejemplo, `vaca` lo es de `cava`. Diseña un programa que lea un texto y determine de cuantas palabras leídas es anagrama la primera dentro de dicho texto.

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

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const string FIN_SEC = "fin";
const int N_LETRAS = int('z')-int('a')+1;
typedef array<int, N_LETRAS> Frecuencia;
//-----------------------------------------
// convierte de letra a índice
int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
        res = int(letra) - int('a');
    }
    return res;
}
//-----------------------------------------
void init(Frecuencia& f)
{
    for (int i = 0; i < int(f.size()); ++i) {
        f[i] = 0;
    }
}
//-----------------------------------------
void frecuencia(const string& s, Frecuencia& f)
{
    init(f);
    for (int i = 0; i < int(s.size()); ++i) {
        int idx = letra_idx(s[i]);
        if (idx < int(f.size())) {
            ++f[idx];
        }
    }
}
//-----------------------------------------------------------------------
bool es_anagrama(const string& palabra, const string& patron)
{
    bool res;
    if (int(palabra.size()) != int(patron.size())) {
        res = false;
    } else {
        Frecuencia f1, f2;
        frecuencia(palabra, f1);
        frecuencia(patron, f2);
        res = f1 == f2;
    }
    return res;
}

//-----------------------------------------------------------------------
int contar_anagramas(const string& patron)
{
    int contador = 0;
    string palabra;

    cin >> palabra;
    while (palabra != FIN_SEC){
        if (es_anagrama(palabra, patron)) {
            ++contador;
        }
        cin >> palabra;
    }
    return contador;
}

//-----------------------------------------------------------------------
int main()
{
    string patron;
    int contador = 0;
```

```
    cout << "Introduzca el texto en mínusculas hasta (fin) con "
         << "el anagrama a comprobar al principio." << endl;
    cin >> patron;
    if (patron != FIN_SEC){
        contador = contar_anagramas(patron);
    }
    cout << "En este texto hay " << contador
         << " anagramas como <" << patron
         << ">." << endl;
}
```

4. Sean u y v dos palabras formadas por letras minúsculas. Diremos que u está asociada a v por vocales fantasmas, si u se puede obtener a partir de v después de un cambio de orden de las letras que mantenga inalterado el orden relativo de las consonantes. Por ejemplo, si u contiene `perla` las palabras `parle`, `aperl`, `pearl`, `paerl`, `prale`, $\cdots$ están asociadas a u por vocales fantasmas. En cambio, `lepra` no lo está, ya que no conserva el orden relativo de las consonantes. Diseña un programa que lea desde teclado un texto y calcule el número de palabras leídas asociadas por vocales fantasmas a la primera del mismo.

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

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const string FIN_SEC = "fin";
const int N_LETRAS = int('z')-int('a')+1;
typedef array<int, N_LETRAS> Frecuencia;
//-----------------------------------------
// convierte de letra a índice
inline int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
        res = int(letra) - int('a');
    }
    return res;
}
//-----------------------------------------
inline bool es_minuscula(char letra)
{
    return (letra >= 'a' && letra <= 'z');
}
//-----------------------------------------
inline bool es_vocal(char letra)
{
    return ((letra == 'a')||(letra == 'e')||(letra == 'i')
            ||(letra == 'o')||(letra == 'u'));
}
//-----------------------------------------
inline bool es_consonante(char letra)
{
    return es_minuscula(letra) && ! es_vocal(letra);
}
//-----------------------------------------
void init(Frecuencia& f)
{
    for (int i = 0; i < int(f.size()); ++i) {
        f[i] = 0;
    }
}
//-----------------------------------------
void frecuencia(const string& s, Frecuencia& f)
{
    init(f);
    for (int i = 0; i < int(s.size()); ++i) {
```

```
        int idx = letra_idx(s[i]);
        if (idx < int(f.size())) {
            ++f[idx];
        }
    }
}
//-----------------------------------------
void copiar_consonantes(const string& palabra, string& consonantes)
{
    consonantes = "";
    for (int i = 0; i < int(palabra.size()); ++i) {
        if (es_consonante(palabra[i])) {
            consonantes += palabra[i];
        }
    }
}
//-----------------------------------------------------------------------------
bool asoc_voc_fant(const string& palabra, const string& patron)
{
    bool res;
    if (int(palabra.size()) != int(patron.size())) {
        res = false;
    } else {
        Frecuencia f1, f2;
        frecuencia(palabra, f1);
        frecuencia(patron, f2);
        res = f1 == f2;
        if (res) {
            string c1, c2;
            copiar_consonantes(palabra, c1);
            copiar_consonantes(patron, c2);
            res = c1 == c2;
        }
    }
    return res;
}

//-----------------------------------------------------------------------------
int contar_asoc_voc_fant(const string& patron)
{
    int contador = 0;
    string palabra;

    cin >> palabra;
    while (palabra != FIN_SEC){
        if (asoc_voc_fant(palabra, patron)) {
            ++contador;
        }
        cin >> palabra;
    }
    return contador;
}

//-----------------------------------------------------------------------------
int main()
{
    string patron;
    int contador = 0;

    cout << "Introduzca el texto en minúsculas hasta (fin) con "
         << "el patrón a comprobar al principio." << endl;
    cin >> patron;
    if (patron != FIN_SEC){
        contador = contar_asoc_voc_fant(patron);
    }
    cout << "En este texto hay " << contador
         << " asociaciones por vocales fantasmas con <" << patron
         << ">." << endl;
}
```

5. Diseñe un programa que permita leer una clave y una secuencia de caracteres terminada en punto (.) que representa un mensaje codificado, y lo decodifique según la clave leída. Los signos de puntuación y dígitos que aparezcan en el mensaje deben escribirse como tales. La clave consiste en una sucesión de las 26 letras

minúsculas del alfabeto, las cuales se hacen corresponder la primera letra de la clave con la letra `'a'`, la segunda con la letra `'b'`, etc....Por ejemplo, una entrada de la forma:

```
Introduzca la clave: ixmrklstnuzbowfaqejdcpvhyg [las 26 letras minúsculas]
Introduzca el texto: milk.
```

de tal forma que la letra `'i'` se corresponde con la letra `'a'`, la letra `'x'` se corresponde con la letra `'b'`, la letra `'m'` se corresponde con la letra `'c'`, y así sucesivamente, por lo que el ejemplo anterior debería dar como salida: `cafe`.

Solución

```
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const int N_LETRAS = ('z' - 'a') + 1;
typedef array<char, N_LETRAS> Clave;
//-----------------------------------------
// convierte de letra a índice
int letra_idx(char letra)
{
    int res = N_LETRAS;
    if (letra >= 'a' && letra <= 'z') {
        res = (letra - 'a');
    }
    return res;
}
//-----------------------------------------
// convierte de índice a letra
inline char idx_letra(int idx)
{
    return char('a' + idx);
}
//-----------------------------------------
void init(Clave& c)
{
    for (int i = 0; i < int(c.size()); ++i) {
        c[i] = ' ';
    }
}
//-----------------------------------------
// Ejemplo: ixmrklstnuzbowfaqejdcpvhyg
//-----------------------------------------
void leer_clave(bool& ok, Clave& clave_cifrado, Clave& clave_descifrado)
{
    ok = true;
    init(clave_cifrado);
    init(clave_descifrado);
    cout << "Introduzca la clave: ";
    int i_letra = 0;
    while (ok && i_letra < N_LETRAS) {
        char letra;
        cin >> letra;
        int idx = letra_idx(letra);
        if ((idx < N_LETRAS)&&(clave_descifrado[idx] == ' ')) {
            clave_descifrado[idx] = idx_letra(i_letra);
            clave_cifrado[i_letra] = letra;
            ++i_letra;
        } else {
            ok = false;
        }
    }
    cin.ignore(1000, '\n');
}
//-----------------------------------------
void transformar(const Clave& clave)
{
    cout << "Introduzca el texto: ";
    char c;
    cin >> ws;
    cin.get(c);
    while (c != '.') {
```

```
        if (c >= 'a' && c <= 'z') {
            c = clave[letra_idx(c)];
        }
        cout << c;
        cin.get(c);
    }
    cout << c << endl;
    cin.ignore(1000, '\n');
}
//-----------------------------------
char menu()
{
    char op;
    cout << endl;
    cout << "--------------------"<<endl;
    cout << "c: Cifrar"<< endl;
    cout << "d: Descifrar"<< endl;
    cout << "x: Fin"<< endl;
    cout << "--------------------"<<endl;
    cout << endl;
    do {
        cout << "    Opcion ? ";
        cin >> op;
    } while (op != 'c' && op != 'd' && op != 'x');
    cin.ignore(1000, '\n');
    return op;
}
//-----------------------------------
int main()
{
    bool ok;
    Clave cc, cd;
    leer_clave(ok, cc, cd);
    if (!ok) {
        cout << "clave errónea"<<endl;
    } else {
        char op;
        do {
            op = menu();
            switch (op) {
            case 'c':
                transformar(cc);
                break;
            case 'd':
                transformar(cd);
                break;
            }
        } while (op != 'x');
    }
}
```

6. Un grupo de M soldados (M es una constante) está rodeado por el enemigo y no hay posibilidad de victoria sin refuerzos, pero hay sólo un caballo para escapar y pedir dichos refuerzos. Los soldados llegan a un acuerdo para determinar quien va a escapar y pedir ayuda. Forman un círculo y sacan de un sombrero un número n y un nombre de uno de los soldados que forman el círculo. Empezando por el soldado cuyo nombre se ha sacado cuentan, en el sentido de las agujas del reloj, n soldados y sacan fuera del círculo al soldado encontrado en el lugar n-ésimo. La cuenta empieza de nuevo con el siguiente soldado (el que sigue al eliminado según el sentido de las agujas del reloj). El proceso continúa de forma que cada vez que la cuenta llega a n se saca un soldado del círculo. Una vez que un soldado se saca del círculo ya no se vuelve a contar. El soldado que queda al final es el que coge el caballo y escapa.

   Diseña un subprograma (y programa para prueba) con la siguiente cabecera que realiza el proceso anterior:

   ```
   typedef array<string, M> Soldados;
   int escapa (const Soldados& nom_soldados, int n, const string& pr_sol);
   ```

   El significado de cada parámetro es el siguiente: `nom_soldados` es un array de cadenas de caracteres con los nombres de los M soldados que forman el círculo y en el orden en que se encuentran en el mismo. Al soldado que ocupa la última posición del array le sigue en el círculo (según las agujas del reloj) el que se encuentra en la primera posición del array. El número `n` para realizar la cuenta según el proceso indicado.

`pr_sol` es nombre del soldado que se escoge inicialmente del sombrero. La función devuelve el índice en el array donde se encuentra el soldado elegido para escapar.

Solución

```
#include <cassert>
#include <iostream>
#include <string>
#include <array>
using namespace std;
//-----------------------------------------
const int N_SOLDADOS = 5;
typedef array<string, N_SOLDADOS> Soldados;
typedef array<bool, N_SOLDADOS> Validos;
//-----------------------------------------
void ini_validos(Validos& v)
{
    for (int i = 0; i < int(v.size()); ++i) {
        v[i] = true;
    }
}
//-----------------------------------------
int cnt_validos(const Validos& v)
{
    int cnt = 0;
    for (int i = 0; i < int(v.size()); ++i) {
        if ( v[i] ) {
            ++cnt;
        }
    }
    return cnt;
}
//-----------------------------------------
void siguiente(const Validos& v, int& i)
{
    assert(cnt_validos(v) > 0);
    do {
        i = (i+1)%int(v.size());
    } while ( ! v[i] );
}
//-----------------------------------------
void siguiente_n(const Validos& v, int n, int& i)
{
    for (int k = 0; k < n; ++k) {
        siguiente(v, i);
    }
}
//-----------------------------------------
void liberar(Validos& v, int i)
{
    assert( v[i] );
    v[i] = false;
}
//-----------------------------------------
void liberar_soldado(Validos& v, int n, int& i)
{
    siguiente_n(v, n, i);
    liberar(v, i);
}
//-----------------------------------------
int buscar(const Soldados& s, const string& x)
{
    int i = 0;
    while ((i < int(s.size())) && (x != s[i])) {
        ++i;
    }
    return i;
}
//-----------------------------------------
int escapa(const Soldados& nombres, int n, const string& pr_sol)
{
    Validos v;
    ini_validos(v);
    int i = buscar(nombres, pr_sol);
```

```
    if (i < int(nombres.size())) {
        for (int k = 0; k < int(nombres.size())-1; ++k) {
            liberar_soldado(v, n, i);
            cout << "    Sale: " << nombres[i] << endl;
        }
        siguiente(v, i);
    }
    return i;
}
//-------------------------------------
void leer_nombres(Soldados& nombres)
{
    cout << "Introduzca nombres de los soldados" << endl;
    for (int i = 0; i < int(nombres.size()); ++i) {
        cout << "Nombre " << i << ": ";
        cin >> nombres[i];
    }
}
//-------------------------------------
int leer_numero()
{
    cout << "Introduzca un número: ";
    int num;
    cin >> num;
    return num;
}
//-------------------------------------
void leer_nombre(string& nombre)
{
    cout << "Introduzca el nombre del primer soldado: ";
    cin >> nombre;

}
//-------------------------------------
void circulo()
{
    Soldados nombres;
    leer_nombres(nombres);
    int n = leer_numero();
    string pr_sol;
    leer_nombre(pr_sol);
    int i_esc = escapa(nombres, n, pr_sol);
    if (i_esc < int(nombres.size())) {
        cout << "El soldado que escapa es: " << nombres[i_esc] << endl;
    } else {
        cout << "Error: " << pr_sol << endl;
    }
}
//-------------------------------------
int main()
{
    circulo();
}
```

Tema 4: Tipos de Datos Estructurados

Ejercicios de Autoevaluación

1. Diseñe un procedimiento para reemplazar en un *texto* **todas** las ocurrencias de un *patrón* por otro valor *nuevo*.

   ```
   void reemplazar (string& texto, const string& patron, const string& nuevo);
   ```

   Además, el programa principal deberá leer de teclado (utilizando `getline`) el *texto* de entrada, el *patrón* y el *nuevo* valor a reemplazar, invocará al subprograma definido anteriormente, y finalmente mostrará en pantalla el resultado de la transformación. Por ejemplo, dados los siguientes valores iniciales de los parámetros:

   texto : `hola, la casa es blanca`
   patrón : `la`
   nuevo : `xxx zz`

   mostrará el siguiente resultado:

   texto : `hoxxx zz, xxx zz casa es bxxx zznca`

   Nótese que el subprograma `reemplazar` no debe leer nada de teclado, ni debe escribir nada en pantalla.

2. Desarrolle lo especificado en los siguientes apartados.

   - Defina el tipo `Sopa_Letras` como una matriz de 10×10 caracteres, que representa una plantilla de una **sopa de letras** (véase la figura 4.1).
   - Defina el tipo `Lista` como un *array* de 8 cadenas de caracteres (*strings*) (véase la figura 4.2).
   - Defina dos constantes denominadas `SOPA` y `LISTA`, de los tipos anteriormente mencionados, utilizando para ello los valores proporcionados en las figuras 4.1 y 4.2, aunque el programa deberá funcionar adecuadamente para cualesquiera otros valores.
   - Se debe desarrollar un subprograma denominado `sopa_de_letras` que recibe una sopa de letras y una lista de palabras, y muestre en pantalla cuantes veces aparece cada palabra de la lista en dicha *sopa de letras*. Para ello debe buscar cada palabra de la lista en la plantilla de la *sopa de letras*.

     ```
     void sopa_de_letras(const Sopa_Letras& sopa, const Lista& lista)
     ```

   - Además, se deberán desarrollar los subprogramas necesarios para una adecuada modularización.
   - Téngase en cuenta que cada palabra podría aparecer horizontalmente (hacia la derecha), verticalmente (hacia abajo), o en diagonal (hacia abajo-derecha). Se recomienda tener especial cuidado en los límites de la matriz.
   - Para el ejemplo de las figuras 4.1 y 4.2, el programa debería mostrar el resultado mostrado en la figura 4.3.
   - Desde el programa principal `main` se invocará al subprograma `sopa_de_letras` con las estructuras constantes `SOPA` y `LISTA`.

	0	1	2	3	4	5	6	7	8	9
0	a	a	a	a	a	a	a	c	a	d	i
1	a	a	m	a	l	a	g	a	a	a
2	a	a	a	a	a	c	o	r	d	o	b
3	s	a	a	a	g	a	c	g	d	i
4	a	e	a	a	r	a	a	r	a	a
5	a	a	v	z	x	e	j	a	e	n
6	a	a	a	i	n	a	a	n	a	a
7	e	t	h	d	l	a	a	a	a	l
8	j	a	e	n	d	l	a	d	a	a
9	a	a	a	a	a	a	a	a	a	g

Figura 4.1: Sopa de letras

```
huelva
sevilla
cadiz
cordoba
malaga
jaen
granada
almeria
```

Figura 4.2: Lista de palabras

```
huelva 0
sevilla 1
cadiz 0
cordoba 0
malaga 1
jaen 2
granada 1
almeria 0
```

Figura 4.3: Resultado

3. Se dispone de una superficie donde cada número representa la altura, respecto al nivel del mar, de ese punto en dicha superficie del plano (véase la siguiente figura). Pretendemos conocer el recorrido que seguirá un *balón* de futbol que se encuentra en un determinado punto de la superficie, considerando que el *balón* se desplaza hacia el punto de la superficie que tenga menor altura de entre sus puntos vecinos, hasta que llegue a un punto que no tenga vecinos con menor altura.

	0	1	2	3	4	5	6	7	8	9	10
0	23	27	32	37	42	47	40	33	26	19	12
1	19	23	27	31	35	40	35	30	25	20	16
2	15	18	22	26	30	34	31	28	25	22	20
3	12	15	18	21	24	28	27	26	26	25	25
4	8	10	13	16	19	22	23	24	26	27	29
5	5	7	9	11	13	16	19	23	26	30	34
6	21	21	22	23	24	25	27	29	32	34	37
7	37	36	36	35	35	35	36	37	38	39	41
8	54	52	50	48	46	44	44	44	44	44	44
9	70	66	63	60	57	54	52	51	50	49	48
10	87	82	77	73	68	64	61	59	56	54	52

Para ello, se deben definir los siguientes tipos y subprogramas:

- `Superficie`

  es un *array* de dos dimensiones (de 11 filas por 11 columnas) de números enteros, donde cada número representa la altura, respecto al nivel del mar, de ese punto en dicha superficie del plano.

- `Movimiento`

  es un *array* de dos dimensiones (de 11 filas por 11 columnas) de caracteres, donde cada elemento almacenará un caracter de asterisco (`'*'`) si el balón **sí** ha pasado por ese punto en su recorrido, o el caracter de guión (`'-'`) si el balón **no** ha pasado por ese punto.

- `void mov_balon(const Superficie& sup, int bfil, int bcol, Movimiento& mov)`

  recibe como primer parámetro una matriz de superficie (véase la figura anterior). El segundo y el tercer parámetro representan la fila y la columna, respectivamente, del lugar inicial donde se encuentra el *balón* de futbol. Finalmente el cuarto parámetro, *de salida*, contendrá el recorrido que realiza un balón desde el punto especificado.

  El procedimiento realiza las siguientes operaciones en el orden especificado:

  a) Asigna el caracter guión (`'-'`) a todos los elementos del parámetro de salida *mov*.

  b) Asigna el caracter de asterisco `'*'` al elemento, del parámetro de salida *mov*, de la fila y columna donde se encuentra inicialmente el balón.

  c) Realiza las siguientes operaciones *mientras* sea posible el movimiento del *balón* desde la posición donde se encuentre, hasta que no sea posible mover el balón a una nueva posición, según lo especificado en los puntos 4.1 y 4.2 siguientes.

  1) Sean f, c la fila y la columna respectivamente del lugar donde se encuentra el *balón*.

  2) Se debe buscar, de entre los elementos vecinos que **existan alrededor** del lugar (f, c) donde se encuentra el balón, e incluyendo también el punto (f, c), aquel elemento con menor altura en el parámetro de superficie. Véase recuadro gris de la figura anterior para la fila 9 y columna 5.

  3) En caso de que haya varios elementos menores, se seleccionará el que esté en una fila menor, y en caso de que haya varios, se seleccionará aquel que esté en una columna menor.

  4) Sean r y s la fila y la columna, respectivamente, de este punto seleccionado con menor altura en el parámetro de superficie.

  4.1) Si las coordenadas de la fila r y columna s son distintas de las coordenadas de la fila f y columna c, entonces moverá la posición del balón a la fila r y columna s, y asigna el caracter de asterisco `'*'` al elemento, del parámetro de salida *mov*, de la fila y columna donde se ha movido el balón.

  4.2) En otro caso, el balón no puede realizar ningún movimiento y se debe finalizar la iteración principal especificada en el punto (c).

  d) Finalmente, el recorrido del balón se encuentra marcado por la secuencia de asteriscos en el parámetro de salida *mov*, tal y como se indica en el siguiente ejemplo. Nótese que no se debe mostrar nada por pantalla en este subprograma.

  Por ejemplo, si el balón se encuentra inicialmente en la fila $f = 9$ y columna $c = 5$, entonces se comparan las alturas de los elementos del rectángulo marcado en gris en la figura anterior, y el balón se moverá a la fila $r = 8$ y columna $s = 5$, y así sucesivamente como indica el siguiente esquema (en la parte superior se indican las coordenadas de cada punto, y en la parte inferior se indica su altura):

$$\frac{(9,5)}{54} \rightarrow \frac{(8,5)}{44} \rightarrow \frac{(7,4)}{35} \rightarrow \frac{(6,3)}{23} \rightarrow \frac{(5,2)}{9} \rightarrow \frac{(5,1)}{7} \rightarrow \frac{(5,0)}{5}$$

- El programa principal debe leer de teclado las coordenadas iniciales del balón, invocar al procedimiento `mov_balon` definido anteriormente, pasándole como primer parámetro una matriz de enteros que contiene los valores especificados en la primera figura del ejercicio, y las coordenadas iniciales del balón leídas de teclado anteriormente. El resultado proporcionado por el cuarto parámetro se obtendrá en una variable cuyos valores deberán ser mostrados en la pantalla según el formato mostrado en la siguiente figura para la la fila 9 y la columna 5.

```
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
* * * - - - - - - - -
- - - * - - - - - - -
- - - - * - - - - - -
- - - - - * - - - - -
- - - - - * - - - - -
- - - - - - - - - - -
```

Tema 5: Búsqueda y Ordenación

Ejercicios Complementarios

1. Diseñe un programa para gestionar una agenda personal, donde la información que se almacena de cada persona es la siguiente: Nombre, Teléfono, Dirección, Calle, Número, Piso, Código Postal y Ciudad. La agenda se encontrará **ordenada** según el nombre de la persona, y permitirá realizar las siguientes operaciones:

   a) Añadir los datos de una persona

   b) Acceder a los datos de una persona a partir de su nombre.

   c) Borrar una persona a partir de su nombre.

   d) Modificar los datos de una persona a partir de su nombre.

   e) Listar el contenido completo de la agenda.

Solución

```
#include <iostream>
#include <string>
#include <cassert>
#include <array>
using namespace std;
// -- Constantes --------
const int MAX_PERSONAS = 50;
// -- Tipos --------------
struct Direccion {
    int num;
    string calle;
    string piso;
    string cp;
    string ciudad;
};
struct Persona {
    string nombre;
    string tel;
    Direccion direccion;
};
// -- Tipos --------------
typedef array<Persona, MAX_PERSONAS> Personas;
struct Agenda {
    int n_pers;
    Personas pers;
};
// Códigos de Error
const int OK = 0;
const int AG_LLENA = 1;
const int NO_ENCONTRADO = 2;
const int YA_EXISTE = 3;
// -- Subalgoritmos -----
void Inicializar (Agenda& ag)
{
    ag.n_pers = 0;
}
// -- Subalgoritmos -----
void Leer_Direccion (Direccion& dir)
{
    cin >> ws;
    getline(cin, dir.calle);
    cin >> dir.num;
    cin >> dir.piso;
    cin >> dir.cp;
    cin >> ws;
    getline(cin, dir.ciudad);
}
//-----------------------------
void Leer_Persona (Persona& per)
{
    cin >> ws;
    getline(cin, per.nombre);
    cin >> per.tel;
```

```
    Leer_Direccion(per.direccion);
}
// -- Subalgoritmos -----
void Escribir_Direccion (const Direccion& dir)
{
    cout << dir.calle << ", ";
    cout << dir.num << " ";
    cout << dir.piso << endl;
    cout << dir.cp << " ";
    cout << dir.ciudad << endl;
}
//-----------------------------
void Escribir_Persona (const Persona& per)
{
    cout << per.nombre << endl;
    cout << per.tel << endl;
    Escribir_Direccion(per.direccion);
}
// -- Subalgoritmos -----
// Busca una Persona en la Agenda Ordenada
// Devuelve su posicion si se encuentra, o bien >= ag.n_pers en otro caso
int Buscar_Persona (const string& nombre, const Agenda& ag)
{
    int i = 0;
    int f = ag.n_pers;
    int m = (i + f) / 2;
    while ((i < f) && (nombre != ag.pers[m].nombre)) {
        if (nombre < ag.pers[m].nombre) {
            f = m;
        } else {
            i = m + 1;
        }
        m = (i + f) / 2;
    }
    if (i >= f) {
        m = ag.n_pers;
    }
    return m;
}
// -- Subalgoritmos -----
int Buscar_Posicion (const string& nombre, const Agenda& ag)
{
    int i = 0;
    while ((i < ag.n_pers) && (nombre >= ag.pers[i].nombre)) {
        ++i;
    }
    return i;
}
//-----------------------------
void Abrir_Hueco (Agenda& ag, int pos, const Persona& per)
{
    for (int i = ag.n_pers; i > pos; --i) {
        ag.pers[i] = ag.pers[i - 1];
    }
    ag.pers[pos] = per;
    ++ag.n_pers;
}
//-----------------------------
void Cerrar_Hueco (Agenda& ag, int pos)
{
    --ag.n_pers;
    for (int i = pos; i < ag.n_pers; ++i) {
        ag.pers[i] = ag.pers[i + 1];
    }
}
// -- Subalgoritmos -----
void Anyadir_Persona (const Persona& per, Agenda& ag, int& ok)
{
    int pos = Buscar_Posicion(per.nombre, ag);
    if ((pos < ag.n_pers) && (per.nombre == ag.pers[pos].nombre)) {
        ok = YA_EXISTE;
    } else if (ag.n_pers == int(ag.pers.size())) {
        ok = AG_LLENA;
    } else {
```

```
        ok = OK;
        Abrir_Hueco(ag, pos, per);
    }
}
//-----------------------------
void Borrar_Persona (const string& nombre, Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Cerrar_Hueco(ag, i);
    }
}
// -- Subalgoritmos -----
void Modificar_Persona (const string& nombre, const Persona& nuevo, Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Cerrar_Hueco(ag, i);
        Anyadir_Persona(nuevo, ag, ok);
    }
}
//-----------------------------
void Imprimir_Persona (const string& nombre, const Agenda& ag, int& ok)
{
    int i = Buscar_Persona(nombre, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Escribir_Persona(ag.pers[i]);
    }
}
//-----------------------------
void Imprimir_Agenda (const Agenda& ag, int& ok)
{
    for (int i = 0; i < ag.n_pers; ++i) {
        Escribir_Persona(ag.pers[i]);
    }
    ok = OK;
}
// -- Subalgoritmos -----
char Menu ()
{
    char opcion;
    cout << endl;
    cout << "a. - Anadir Persona" << endl;
    cout << "b. - Buscar Persona" << endl;
    cout << "c. - Borrar Persona" << endl;
    cout << "d. - Modificar Persona" << endl;
    cout << "e. - Imprimir Agenda" << endl;
    cout << "x. - Salir" << endl;
    do {
        cout << "Introduzca Opcion: ";
        cin >> opcion;
    } while ( ! (((opcion >= 'a') && (opcion <= 'e')) || (opcion == 'x')));
    return opcion;
}
// -- Subalgoritmos -----
void Escribir_Cod_Error (int cod)
{
    switch (cod) {
    case OK:
        cout << "Operacion correcta" << endl;
        break;
    case AG_LLENA:
        cout << "Agenda llena" << endl;
        break;
    case NO_ENCONTRADO:
```

```
        cout << "La persona no se encuentra en la agenda" << endl;
        break;
    case YA_EXISTE:
        cout << "La persona ya se encuentra en la agenda" << endl;
        break;
    }
}
// -- Principal ---------
int main ()
{
    Agenda ag;
    char opcion;
    Persona per;
    string nombre;
    int ok;
    Inicializar(ag);
    do {
        opcion = Menu();
        switch (opcion) {
        case 'a':
            cout << "Introduzca los datos de la Persona"<<endl;
            cout << "(nombre, tel, calle, num, piso, cod_postal, ciudad)" << endl;
            Leer_Persona(per);
            Anyadir_Persona(per, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'b':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            Imprimir_Persona(nombre, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'c':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            Borrar_Persona(nombre, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'd':
            cout << "Introduzca Nombre" << endl;
            cin >> nombre;
            cout << "Nuevos datos de la Persona" << endl;
            cout << "(nombre, tel, calle, num, piso, cod_postal, ciudad)" << endl;
            Leer_Persona(per);
            Modificar_Persona(nombre, per, ag, ok);
            Escribir_Cod_Error(ok);
            break;
        case 'e':
            Imprimir_Agenda(ag, ok);
            Escribir_Cod_Error(ok);
            break;
        }
    } while (opcion != 'x' );
}
```

2. Dado una estructura de datos que puede contener hasta un número máximo de 50 personas, donde cada persona contiene su nombre, la fecha de nacimiento y su teléfono:

   a) Diseñe un subprograma para ordenar dicha estructura de datos. El criterio de ordenación es en orden creciente por nombre de la persona, y en caso de nombres iguales, entonces se considera el orden creciente por fecha de nacimiento.

   b) Dada la estructura de datos ordenada según el criterio de ordenación del apartado anterior, diseñe un subprograma para realizar la búsqueda binaria de una persona en la estructura de datos, para ello, el subprograma recibirá tanto el nombre de la persona, como su fecha de nacimiento.

   c) Diseñe el programa principal y los subprogramas necesarios para comprobar el funcionamiento adecuado de los subprogramas anteriores.

Antonio 15/05/1997 952.234.567	...	Lucas 23/11/1989 952.135.246	Lucas 17/03/1992 952.235.711	Lucas 14/08/1992 952.123.456	Lucas 25/08/1992 952.987.654	...	María 13/01/1994 952.567.234

Solución

```
#include <iostream>
#include <iomanip>
#include <string>
#include <cassert>
#include <array>
using namespace std;
// -- Constantes --------
const int MAX_PERSONAS = 50;
// -- Tipos -------------
struct Fecha {
    int dia;
    int mes;
    int anyo;
};
struct Persona {
    string nombre;
    Fecha fnac;
    string tel;
};
struct Clave {
    string nombre;
    Fecha fnac;
};
// -- Tipos -------------
typedef array<Persona, MAX_PERSONAS> Personas;
struct Agenda {
    int n_pers;
    Personas pers;
};
// Códigos de Error
const int OK = 0;
const int AG_LLENA = 1;
const int NO_ENCONTRADO = 2;
const int YA_EXISTE = 3;
// -- Subalgoritmos -----
void Inicializar (Agenda& ag)
{
    ag.n_pers = 0;
}
// -- Subalgoritmos -----
void Leer_Fecha (Fecha& f)
{
    cin >> f.dia;
    cin >> f.mes;
    cin >> f.anyo;
}
//-----------------------------------
void Leer_Clave (Clave& clave)
{
    cin >> ws;
    getline(cin, clave.nombre);
    Leer_Fecha(clave.fnac);
}
//---------------------------
void Leer_Persona (Persona& per)
{
    cin >> ws;
    getline(cin, per.nombre);
    Leer_Fecha(per.fnac);
    cin >> per.tel;
}
//-----------------------------------
void Escribir_Fecha (const Fecha& f)
{
    cout << setfill('0') << setw(2) << f.dia << "/";
    cout << setfill('0') << setw(2) << f.mes << "/";
    cout << setfill('0') << setw(4) << f.anyo << " ";
}
//---------------------------
void Escribir_Persona (const Persona& per)
{
    cout << per.nombre << endl;
    Escribir_Fecha(per.fnac);
```

```
    cout << per.tel << " ";
    cout << endl;
}
//-----------------------------------------
void Crear_Clave (Clave& clave, const Persona& per)
{
    clave.nombre = per.nombre;
    clave.fnac = per.fnac;
}
//-----------------------------------------
// COMPARACIÓN
//-----------------------------------------
inline bool es_igual(const Fecha& f1, const Fecha& f2)
{
    return (f1.anyo == f2.anyo) && (f1.mes == f2.mes) && (f1.dia == f2.dia);
}
inline bool es_distinto (const Fecha& f1, const Fecha& f2)
{
    return ! es_igual(f1, f2);
}
inline bool es_menor (const Fecha& f1, const Fecha& f2)
{
    return ((f1.anyo < f2.anyo)
            || ((f1.anyo == f2.anyo)
                && ((f1.mes < f2.mes)
                    || ((f1.mes == f2.mes) && (f1.dia < f2.dia)))));
}
inline bool es_mayor (const Fecha& f1, const Fecha& f2)
{
    return es_menor(f2, f1);
}
inline bool es_menor_igual (const Fecha& f1, const Fecha& f2)
{
    return ! es_menor(f2, f1);
}
inline bool es_mayor_igual (const Fecha& f1, const Fecha& f2)
{
    return ! es_menor(f1, f2);
}
//-----------------------------------------
inline bool es_igual (const Persona& p1, const Persona& p2)
{
    return (p1.nombre == p2.nombre)
        && es_igual(p1.fnac, p2.fnac)
        && (p1.tel == p2.tel);
}
inline bool es_distinto (const Persona& p1, const Persona& p2)
{
    return ! es_igual(p1, p2);
}
inline bool es_menor (const Persona& p1, const Persona& p2)
{
    return ((p1.nombre < p2.nombre)
            || ((p1.nombre == p2.nombre) &&
                es_menor(p1.fnac, p2.fnac)));
}
inline bool es_mayor (const Persona& p1, const Persona& p2)
{
    return es_menor(p2, p1);
}
inline bool es_menor_igual (const Persona& p1, const Persona& p2)
{
    return ! es_menor(p2, p1);
}
inline bool es_mayor_igual (const Persona& p1, const Persona& p2)
{
    return ! es_menor(p1, p2);
}
//-----------------------------------------
inline bool es_igual (const Clave& p1, const Persona& p2)
{
    return (p1.nombre == p2.nombre) && es_igual(p1.fnac, p2.fnac);
}
inline bool es_distinto (const Clave& p1, const Persona& p2)
```

```
{
    return ! es_igual(p1, p2);
}
inline bool es_menor (const Clave& p1, const Persona& p2)
{
    return ((p1.nombre < p2.nombre)
            || ((p1.nombre == p2.nombre)
                && es_menor(p1.fnac, p2.fnac)));
}
inline bool es_mayor (const Clave& p1, const Persona& p2)
{
    return ((p1.nombre > p2.nombre)
            || ((p1.nombre == p2.nombre) && es_mayor(p1.fnac, p2.fnac)));
}
inline bool es_menor_igual (const Clave& p1, const Persona& p2)
{
    return ! es_mayor(p1, p2);
}
inline bool es_mayor_igual (const Clave& p1, const Persona& p2)
{
    return ! es_menor(p1, p2);
}
//-----------------------------------
// ANYADIR PERSONA SIN ORDENACIÓN
//-----------------------------------
// Busca una Persona en la Agenda
// Devuelve su posición si se encuentra, o bien >= ag.n_pers en otro caso
int Buscar_Persona (const Clave& clave, const Agenda& ag)
{
    int i = 0;
    while ((i < ag.n_pers) && es_distinto(clave, ag.pers[i])) {
        ++i;
    }
    return i;
}
//-------------------------------
void Anyadir (Agenda& ag, const Persona& per)
{
    ag.pers[ag.n_pers] = per;
    ++ag.n_pers;
}
//-----------------------------------
void Anyadir_Persona (const Persona& per, Agenda& ag, int& ok)
{
    Clave clave;
    Crear_Clave(clave, per);
    int i = Buscar_Persona(clave, ag);
    if (i < ag.n_pers) {
        ok = YA_EXISTE;
    } else if (ag.n_pers == int(ag.pers.size())) {
        ok = AG_LLENA;
    } else {
        ok = OK;
        Anyadir(ag, per);
    }
}
//-----------------------------------
// BUSCAR PERSONA CON ORDENACIÓN
//-----------------------------------
// Busca una Persona en la Agenda Ordenada
// Devuelve su posición si se encuentra, o bien >= ag.n_pers en otro caso
int Buscar_Persona_Binaria (const Clave& clave, const Agenda& ag)
{
    int i = 0;
    int f = ag.n_pers;
    int m = (i + f) / 2;
    while ((i < f) && es_distinto(clave, ag.pers[m])) {
        if (es_menor(clave, ag.pers[m])) {
            f = m;
        } else {
            i = m + 1;
        }
        m = (i + f) / 2;
    }
```

```
    if (i >= f) {
        m = ag.n_pers;
    }
    return m;
}
//-----------------------------------
void Imprimir_Persona_Ord (const Clave& clave, const Agenda& ag, int& ok)
{
    int i = Buscar_Persona_Binaria(clave, ag);
    if (i >= ag.n_pers) {
        ok = NO_ENCONTRADO;
    } else {
        ok = OK;
        Escribir_Persona(ag.pers[i]);
    }
}
//-----------------------------------
// ORDENACIÓN
//-----------------------------------
int Buscar_Posicion (const Clave& clave, const Agenda& ag)
{
    int i = 0;
    while ((i < ag.n_pers) && es_mayor_igual(clave, ag.pers[i])) {
        ++i;
    }
    return i;
}
//-----------------------------
void Anyadir_Ord (Agenda& ag, int pos, const Persona& per)
{
    for (int i = ag.n_pers; i > pos; --i) {
        ag.pers[i] = ag.pers[i - 1];
    }
    ag.pers[pos] = per;
    ++ag.n_pers;
}
//-----------------------------------
void Anyadir_Persona_Ord (const Persona& per, Agenda& ag)
{
    Clave clave;
    Crear_Clave(clave, per);
    int pos = Buscar_Posicion(clave, ag);
    Anyadir_Ord(ag, pos, per);
}
//-----------------------------------
void Ordenar_Agenda (Agenda& ag, int& ok)
{
    ok = OK;
    int npers = ag.n_pers;
    ag.n_pers = 1;
    while (ag.n_pers < npers) {
        Persona per = ag.pers[ag.n_pers];
        Anyadir_Persona_Ord(per, ag);
    }
}
//-----------------------------
void Imprimir_Agenda (const Agenda& ag, int& ok)
{
    for (int i = 0; i < ag.n_pers; ++i) {
        Escribir_Persona(ag.pers[i]);
    }
    ok = OK;
}
//-----------------------------------
//-----------------------------------
//-----------------------------------
char Menu (bool ord)
{
    char opcion;
    cout << endl;
    if (ord) {
        cout << "     Estado: Ordenado" << endl;
    } else {
        cout << "     Estado: Desordenado" << endl;
```

```
    }
    cout << "a. - Anadir Persona" << endl;
    cout << "b. - Buscar Persona" << endl;
    cout << "c. - Ordenar" << endl;
    cout << "d. - Imprimir Agenda" << endl;
    cout << "x. - Salir" << endl;
    do {
        cout << "Introduzca Opción: ";
        cin >> opcion;
    } while ( ! (((opcion >= 'a') && (opcion <= 'd')) || (opcion == 'x')));
    return opcion;
}
// -- Subalgoritmos -----
void Escribir_Cod_Error (int cod)
{
    switch (cod) {
    case OK:
        cout << "Operación correcta" << endl;
        break;
    case AG_LLENA:
        cout << "Agenda llena" << endl;
        break;
    case NO_ENCONTRADO:
        cout << "La persona no se encuentra en la agenda" << endl;
        break;
    case YA_EXISTE:
        cout << "La persona ya se encuentra en la agenda" << endl;
        break;
    }
}
// -- Principal ---------
int main ()
{
    bool ordenada = true;
    Agenda ag;
    char opcion;
    Clave clave;
    Persona nuevo;
    int ok;
    Inicializar(ag);
    do {
        opcion = Menu(ordenada);
        switch (opcion) {
        case 'a':
            cout << "Introduzca los datos de la Persona"<<endl;
            cout << "(nombre, dia, mes, anyo, tel)" << endl;
            Leer_Persona(nuevo);
            Anyadir_Persona(nuevo, ag, ok);
            Escribir_Cod_Error(ok);
            ordenada = false;
            break;
        case 'b':
            if (ordenada) {
                cout << "Introduzca Nombre, dia, mes, anyo" << endl;
                Leer_Clave(clave);
                Imprimir_Persona_Ord(clave, ag, ok);
                Escribir_Cod_Error(ok);
            } else {
                cout << "Error, la agenda no esta ordenada"<<endl;
            }
            break;
        case 'c':
            Ordenar_Agenda(ag, ok);
            Escribir_Cod_Error(ok);
            ordenada = true;
            break;
        case 'd':
            Imprimir_Agenda(ag, ok);
            Escribir_Cod_Error(ok);
            break;
        }
    } while (opcion != 'x' );
}
```

Tema 5: Búsqueda y Ordenación

Ejercicios de Autoevaluación

1. Se dispone de dos imágenes del cielo, representadas como un *array* de 10×10 números enteros, donde cada número representa la *magnitud lumínica* de cada punto registrado, de tal forma que las estrellas se representan por un valor numérico mayor que cero, y la oscuridad por un valor igual a cero (véanse las figuras 5.1 y 5.2).

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0
2	0	0	35	0	0	0	12	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	25	0	0	40	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	14	0	0	0	0	0	0	0
7	0	30	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	16	0
9	0	0	0	0	0	0	0	0	0	0

Figura 5.1: Magnitud lumínica (IMG1)

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	13	0	0	0	0
2	0	0	0	0	0	0	0	18	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	40	0	0	0
6	0	0	0	0	35	0	0	0	14	0
7	0	0	0	11	0	0	0	0	0	30
8	0	0	0	0	0	0	25	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Figura 5.2: Magnitud lumínica (IMG2)

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	·	·	·	·	·	·	·
1	·	+	·	·	·	·	·	·	·	·
2	·	·	✳	·	·	+	·	·	·	·
3	·	·	·	·	·	·	·	·	·	·
4	✳	·	·	✳	·	·	·	·	·	·
5	·	·	·	·	·	·	·	·	·	·
6	·	·	+	·	·	·	·	·	·	·
7	·	✳	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	·	·	+	·
9	·	·	·	·	·	·	·	·	·	·

Figura 5.3: Imagen representada (IMG1)

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	+	·	·	·	·
2	·	·	·	·	·	·	·	+	·	·
3	·	·	·	·	·	·	·	·	·	·
4	·	·	·	·	·	·	·	·	·	·
5	·	·	·	·	·	·	✳	·	·	·
6	·	·	·	·	✳	·	·	·	+	·
7	·	·	·	+	·	·	·	·	·	✳
8	·	·	·	·	·	·	✳	·	·	·
9	·	·	·	·	·	·	·	·	·	·

Figura 5.4: Imagen representada (IMG2)

- Defina el tipo `Magnitudes` como una matriz de 10×10 números enteros, que representa las magnitudes lumínicas de una imagen del cielo (véanse las figuras 5.1 y 5.2).
- Defina el tipo `Auxiliar` que proporcione el soporte necesario para resolver el problema, según se describe más adelante (véanse las figuras 5.5 y 5.6).
- Defina dos constantes denominadas `IMG1` y `IMG2`, que representen las magnitudes de dos imágenes del cielo, utilizando para ello los valores proporcionados en las figuras 5.1 y 5.2, aunque el programa deberá funcionar adecuadamente para cualesquiera otros valores.
- Se debe desarrollar un subprograma denominado `constelaciones` que recibe dos arrays de magnitudes lumínicas, y comprueba si la misma constelación aparece en ambos arrays **según el método especificado a continuación**, y devuelve `true` en `ok` en caso afirmativo, y `false` en caso negativo. Además, en el caso afirmativo, también devuelve el desplazamiento de la constelación en la segunda imagen.

```
void constelaciones(const Magnitudes& img1, const Magnitudes& img2,
                    bool& ok, double& desplazamiento)
```

- Además, se deberán desarrollar los subprogramas necesarios para una adecuada modularización.
- El método para determinar si la misma constelación aparece en ambas imágenes es el siguiente (en la siguiente explicación, la coordenada x hace referencia a la columna y la coordenada y hace referencia a la fila de un determinado punto):
  - *a*) Para cada array de magnitudes lumínicas, se realizan las siguientes acciones:
    1) Se seleccionan aquellos puntos (x_i, y_i) cuya magnitud sea mayor que 20 y menor que 50, y se añaden a una estructura de datos (lista de máximo 20 elementos) de tipo **Auxiliar** (véase figura 5.5). Podemos suponer que no habrá más de 20 puntos en ese rango.
    2) Para cada punto seleccionado en el paso anterior (en la estructura auxiliar), se calculan las distancias que lo separan del resto de puntos seleccionados y se añaden a la estructura de datos auxiliar para cada punto (véase la figura 5.6), considerando que la distancia ($d_{ij} \in \mathbb{R}$) entre dos puntos (x_i, y_i) y (x_j, y_j) es:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

3) Se recomienda que para cada punto seleccionado, se ordene el array de distancias a los otros puntos, ya que será más fácil de comparar con otros arrays de distancias (la comparación de las distancias será necesaria en pasos posteriores).

b) Para saber si la misma constelación se halla en ambas imágenes, hay que comprobar que el número de puntos seleccionados es igual en ambas, y además, se debe comprobar que **cada** punto seleccionado de una imagen es igual a algún punto en la otra imagen, tanto en magnitud, como en sus distancias a los otros puntos, pero sin tener en cuenta sus coordenadas (véanse las figuras 5.6 y 5.8).

c) Debido a la pérdida de precisión en la operaciones con números reales, cuando se comparan las distancias entre puntos, consideraremos que dos números reales a y b son iguales si $\text{fabs}(a-b) < 0.0001$.

d) Por simplicidad, supondremos que dentro de la misma constelación, no existen dos puntos que tengan las mismas distancias a los otros puntos de la misma constelación.

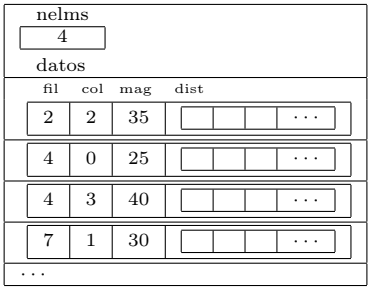


Figura 5.5: Selección de puntos (IMG1)

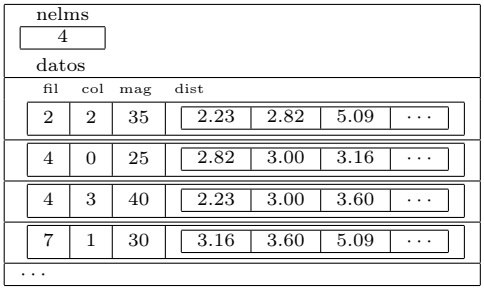


Figura 5.6: Cálculo de distancias (IMG1)

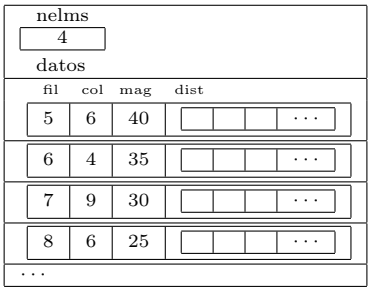


Figura 5.7: Selección de puntos (IMG2)

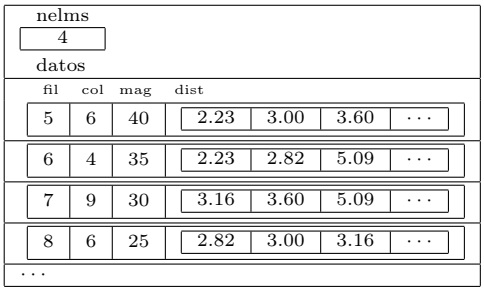


Figura 5.8: Cálculo de distancias (IMG2)

- Una vez que sabemos que la misma constelación se encuentra en ambas imágenes, podemos calcular el desplazamiento de la constelación en la imagen. Para ello, se busca el primer punto (x_0, y_0) seleccionado de la primera imagen en la segunda imagen (x_j, y_j) (utilizando su magnitud y las distancias a los otros puntos, pero sin tener en cuenta sus coordenadas), y una vez localizado, calculando la distancia ($d \in \mathbb{R}$) entre las coordenadas de ambos puntos.

$$d = \sqrt{(x_0 - x_j)^2 + (y_0 - y_j)^2}$$

- Para el ejemplo de las figuras 5.1 y 5.2, el programa debería mostrar de forma afirmativa que la misma constelación aparece en las dos imágenes, con un desplazamiento de 4.4721.
- Desde el programa principal `main` se invocará al subprograma `constelaciones` con las estructuras constantes `IMG1` e `IMG2`, y mostrará los resultados obtenidos.