



Fundamentos de Programación.

Guión de Prácticas.

Curso 2017/2018

"Lo que tenemos que aprender a hacer, lo aprendemos haciéndolo".
Aristóteles



"In theory, there is no difference between theory and practice. But, in practice, there is".
Jan L. A. van de Snepscheut



"The gap between theory and practice is not as wide in theory as it is in practice".



"Theory is when you know something, but it doesn't work. Practice is when something works, but you don't know why. Programmers combine theory and practice: Nothing works and they don't know why".



Sobre el guión de prácticas

El guión está dividido en sesiones. En cada sesión se plantean una serie de problemas de programación a resolver. En la semana número i se publicará la **Sesión i** . En dicha sesión se especifica la lista de problemas que el alumno tiene que resolver.

Las soluciones de los ejercicios deberán ser subidas a la plataforma **PRADO**, en el plazo establecido (consulte el fichero de información de la asignatura). El alumno subirá un fichero zip que contendrá los ficheros con extensión `cpp` correspondientes a las soluciones de los ejercicios.

La defensa de la sesión i se hará la semana siguiente (semana $i + 1$), durante las horas de prácticas. El profesor llamará aleatoriamente a los alumnos para que defiendan dichos ejercicios (a veces explicándolos a sus compañeros). Simultáneamente a la defensa, todos los alumnos tendrán que ir realizando una serie de actividades que vienen descritas en este guión. Dichas actividades no se entregarán al profesor. Terminada la defensa, el profesor explicará los ejercicios a todos los alumnos.

Los problemas a resolver en cada sesión están incluidos en las *Relaciones de Problemas*. Hay una relación de problemas por cada tema de la asignatura. Los problemas que hay que entregar son de dos tipos:

1. **Obligatorios**: Todos los alumnos deben resolver estos problemas.

Si se realizan correctamente estos ejercicios, el alumno podrá sacar hasta un 9 (sobre 10) en la nota de prácticas.

2. **Opcionales**: Su entrega no es obligatoria.

Si se realizan correctamente estos ejercicios, el alumno podrá sacar hasta un 10 (sobre 10) en la nota de prácticas. Para poder optar a la Matrícula de Honor es necesario realizar todos los ejercicios opcionales de todas las sesiones.

Para la realización de estas prácticas, se utilizará el entorno de programación Orwell Dev C++. En la página **3** se encuentran las instrucciones para su instalación en nuestra casa. En cualquier caso, el alumno puede instalar en su casa cualquier otro compilador.

Muy importante:

- La resolución de los problemas y actividades puede hacerse en grupo, pero la defensa durante las horas de prácticas es individual.
- Es muy importante que la asignatura se lleve al día para poder realizar los ejercicios propuestos en estos guiones.

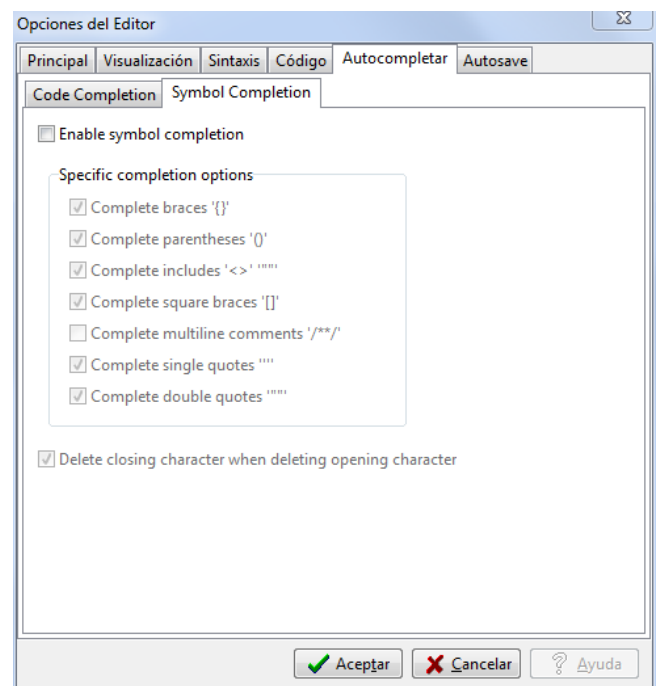
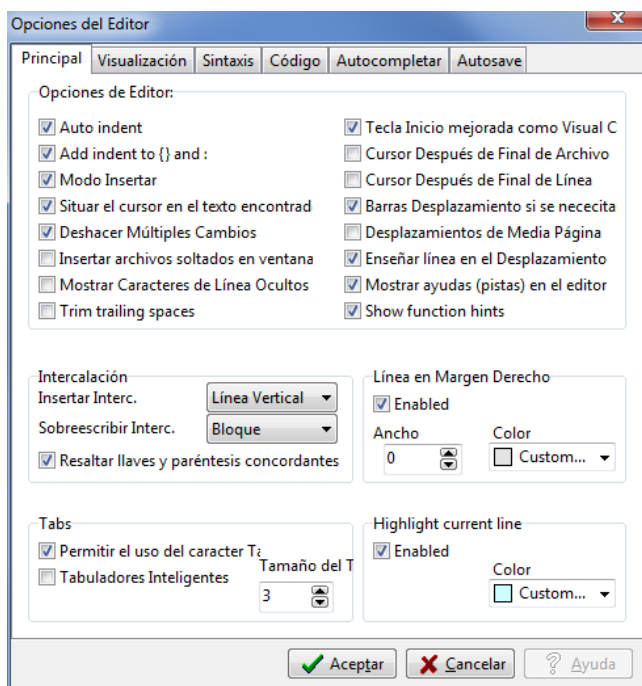
Instalación de Orwell Dev C++ en nuestra casa

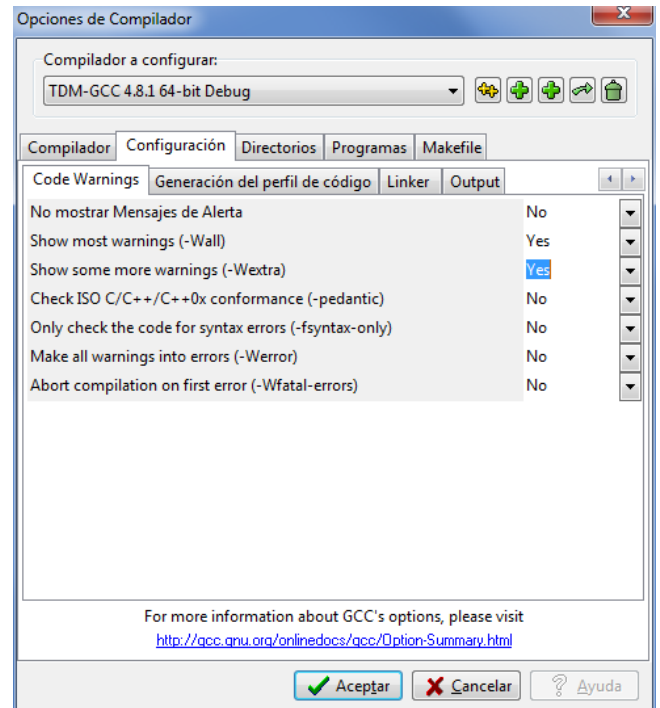
El entorno de desarrollo que usaremos será Orwell Dev C++. Puede descargarse desde la página:

http://sourceforge.net/projects/orwelldevcpp/?source=typ_redirect

Cuando lo instalemos en nuestra casa, configurar las siguientes opciones:

```
Herramientas -> Opciones del Compilador
  Compilador a configurar: TDM-GCC ... Debug
  Configuración -> Code Warnings. Marcar los siguientes:
    Show most warnings
    Show some more warnings
  Configuración -> Linker.
    Generar información de Debug: Yes
    Generación de código -> Language standard (-std) -> ISO C++11
Herramientas -> Opciones del editor
  -> Principal
    Desmarcar Tabuladores inteligentes
    Tamaño del tabulador: 3
  -> Autocompletar -> Symbol completion
    Desmarcar Enable Symbol completion
```





Preparar y acceder a la consola del sistema

La consola de Windows (la ventana con fondo negro que aparece al ejecutar el comando `cmd.exe`, o bien la que sale al ejecutar un programa en Dev C++) no está preparada por defecto para mostrar adecuadamente caracteres latinos como los acentos. Por ejemplo, al ejecutar la sentencia de C++

```
cout << "Atención"
```

saldrá en la consola un mensaje en la forma

```
Atenci3/4n
```

Si se desea ver correctamente dichos caracteres, tenemos dos alternativas:

1. Incluir la siguiente sentencia al inicio de nuestro programa (.cpp), al empezar el `main`:

```
setlocale(LC_ALL, "spanish");
```

2. En Windows, seguir los siguientes pasos:

- a) Cambiar la fuente de la consola a una que acepte caracteres Unicode. En la versión de XP de las aulas ya se ha realizado dicho cambio. En nuestra casa, tendremos que hacer lo siguiente:

Inicio -> Ejecutar -> cmd

Una vez que se muestre la consola, hacemos click con la derecha y seleccionamos Predeterminados. Seleccionamos la fuente Lucida Console y aceptamos.

- b) Si queremos que la consola siempre cargue la tabla de caracteres latinos, debemos modificar el registro de Windows. Lo abrimos desde

Inicio->Ejecutar->regedit

Nos situamos en la clave

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
\Control\Nls\CodePage

y cambiamos el valor que hubiese dentro de OEMCP y ACP por el de 1252. Esta es una página muy parecida a la 8859-1 referenciada en los apuntes (Tema I)

Esta es la forma recomendada y la que se ha usado en las aulas de prácticas. Requiere reiniciar el ordenador.

Muy Importante: Si se usa otra tabla (distinta a 1252), el sistema operativo podría incluso no arrancar.

Tabla resumen de accesos directos usados en Orwell Dev C++

Tab	Tabula una línea o un bloque
Shift Tab	Quita tabulación a una línea o un bloque
Ctrl Barra Espaciadora	Ayuda autocompletación del código
F9	Compilar
F10	Ejecutar
F11	Compilar y Ejecutar
F5	Depurar
	Empieza la depuración
F7	Siguiente paso
	Ejecución paso a paso sin entrar en los métodos o funciones
F8	Avanzar paso a paso
	Ejecución paso a paso entrando en los métodos o funciones

Sesión 0

Esta sesión consta de:

1. *Actividades a realizar en casa.* Son las tareas que el alumno debe realizar en su casa durante la semana del 13 al 17 de Septiembre. Esta es la única sesión en la que el alumno no tendrá que realizar ninguna entrega.
2. *Actividades a realizar en las aulas de ordenadores.* Son las actividades que se realizarán con la ayuda del profesor durante la primera hora de las clases de prácticas de la semana del 18 al 22 de Septiembre.

► **Actividades a realizar en casa**

Actividad: Conseguir login y password.

El alumno debe registrarse electrónicamente como alumno de la Universidad, tal y como se indica en el fichero de información general de la asignatura. De esta forma, obtendremos un login y un password que habrá que introducir al arrancar los ordenadores en las aulas de prácticas. La cuenta tarda 48 horas en activarse, por lo que el registro debe realizarse al menos dos días antes de la primera sesión de prácticas.

Actividad: Instalación de Orwell Dev C++.

Durante la semana del 13 al 17 de Septiembre, el alumno debe instalar en su casa el compilador Orwell Dev C++. Consulte la sección de Instalación (página 3) de este guión.

Actividad: Preparar la clase de prácticas de la semana del 18 al 22 de Septiembre.

Realice una lectura rápida de las actividades a realizar durante las horas de prácticas en las aulas de ordenadores (ver páginas siguientes). Estas actividades se desarrollarán durante la semana del 18 al 22 de Septiembre.

Actividades de Ampliación

Lea el artículo de Norvig: *Aprende a programar en diez años*

<http://loro.sourceforge.net/notes/21-dias.html>

sobre la dificultad del aprendizaje de una disciplina como la Programación.



► **Actividades a realizar en las aulas de ordenadores**

Estas son las actividades que se realizarán, con la ayuda del profesor, durante las clases de prácticas de la semana del 18 al 22 de Septiembre.

El Entorno de Programación. Compilación de Programas

Arranque del Sistema Operativo

Para poder arrancar el SO en las aulas de ordenadores, es necesario obtener el login y password indicados en las actividades a realizar en casa.

En la casilla etiquetada como Código, introduciremos `fp`. Al arrancar el SO, aparecerá una instalación básica de Windows con el compilador Orwell Dev C++. Todo lo que escribamos en la unidad C: se perderá al apagar el ordenador.

Por ello, el alumno dispone de un directorio de trabajo en la unidad lógica U:, cuyos contenidos permanecerán durante todo el curso académico. En cualquier caso, es recomendable no saturar el espacio usado ya que, en caso contrario, el compilador podría no funcionar.

El alumno deberá crear el directorio U:\FP dentro de su unidad U:

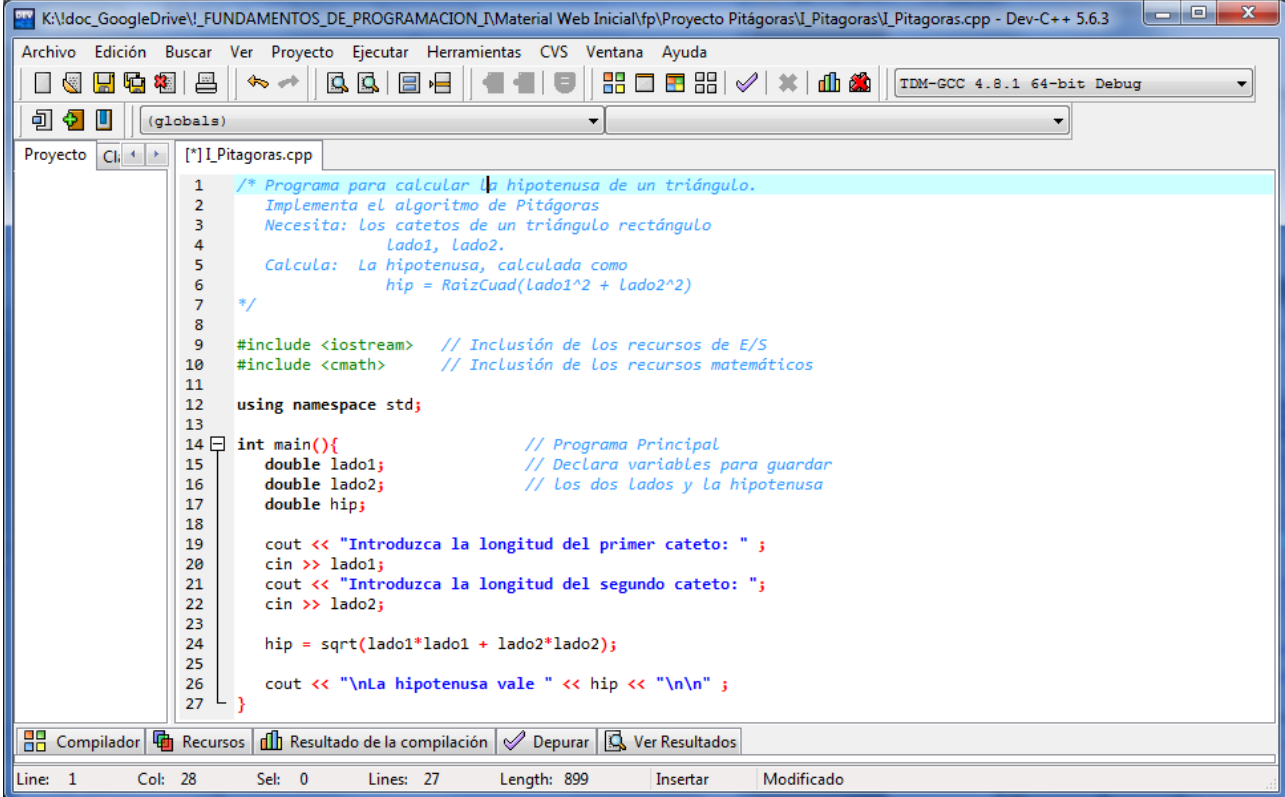
Para acceder a la unidad U: desde nuestras casas, debemos usar cualquier programa de ftp que use el protocolo ssh, como por ejemplo filezilla o winscp. Instalamos este programa en nuestra casa y simplemente nos conectamos a `turing.ugr.es` con nuestras credenciales.

El primer programa

Copiando el código fuente

Descargue el fichero http://decsai.ugr.es/jccubero/FP/I_Pitagoras.cpp y cópielo en su carpeta local (dentro de U:\FP).

Desde el Explorador de Windows, haga doble click sobre el fichero `I_Pitagoras.cpp`. Debe aparecer una ventana como la de la figura 1



The screenshot shows the Dev-C++ 5.6.3 IDE interface. The title bar indicates the file path: K:\doc_GoogleDrive\FUNDAMENTOS_DE_PROGRAMACION\Material Web Inicial\fp\Proyecto Pitágoras\Pitagoras\Pitagoras.cpp - Dev-C++ 5.6.3. The menu bar includes Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Herramientas, CVS, Ventana, and Ayuda. The toolbar contains icons for file operations, compilation, and execution. The compiler is set to TDM-GCC 4.8.1 64-bit Debug. The project explorer on the left shows a project named 'Pitagoras' with a file 'I_Pitagoras.cpp'. The main editor displays the following C++ code:

```
1  /* Programa para calcular la hipotenusa de un triángulo.
2  Implementa el algoritmo de Pitágoras
3  Necesita: Los catetos de un triángulo rectángulo
4  lado1, lado2.
5  Calcula: La hipotenusa, calculada como
6  hip = RaizCuad(lado1^2 + lado2^2)
7  */
8
9  #include <iostream>    // Inclusión de los recursos de E/S
10 #include <cmath>      // Inclusión de los recursos matemáticos
11
12 using namespace std;
13
14 int main(){           // Programa Principal
15     double lado1;      // Declara variables para guardar
16     double lado2;      // Los dos lados y la hipotenusa
17     double hip;
18
19     cout << "Introduzca la longitud del primer cateto: ";
20     cin >> lado1;
21     cout << "Introduzca la longitud del segundo cateto: ";
22     cin >> lado2;
23
24     hip = sqrt(lado1*lado1 + lado2*lado2);
25
26     cout << "\nLa hipotenusa vale " << hip << "\n\n";
27 }
```

The status bar at the bottom shows: Line: 1, Col: 28, Sel: 0, Lines: 27, Length: 899, Insertar, and Modificado.

Figura 1: Programa que implementa el algoritmo de Pitágoras

Algunas consideraciones con respecto a la escritura de código en C++ (ver figura 2)

- Es bueno que, desde el principio se incluyan comentarios indicando el objetivo del programa y resaltando los aspectos más importantes de la implementación.
- Es muy importante una correcta tabulación de los programas. Por ahora, incluiremos todas las sentencias del programa principal con una tabulación. Sólo es necesario incluir la primera; el resto las pone automáticamente el entorno, al pasar a la siguiente línea.
- Para facilitar la lectura del código fuente, se deben usar espacios en blanco para separar las variables en la línea en la que van declaradas, así como antes y después del símbolo = en una sentencia de asignación. Dejad también un espacio en blanco antes y después de << y >> en las sentencias que contienen una llamada a cout y cin respectivamente.

```
int main(){
    double lado1;
    double lado2;
    double hip;

    cout << "Introduzca la longitud del primer cateto: " ;
    cin >> lado1;
    cout << "Introduzca la longitud del segundo cateto: ";
    cin >> lado2;

    hip = sqrt(lado1*lado1 - lado2*lado2);

    cout << "\nLa hipotenusa vale " << hip << "\n\n" ;
    system("pause");
}
```

Declaraciones

Entradas datos

Computos

Salida Resultados

líneas en blanco

espacios en blanco

Comentarios separados visualmente del código

// Programa Principal
// Declara variables para guardar
// los dos lados y la hipotenusa

Figura 2: Escritura de código

No respetar las normas de escritura de código baja puntos en todos los exámenes y prácticas de la asignatura

IMPORTANT

Compilación

Una vez cargado el programa, pasamos a comprobar si las sentencias escritas son sintácticamente correctas, es decir, pasamos a *compilar* el programa. Para ello pulsamos F9, o bien sobre el icono .

Para que el proceso de compilación se realice de forma correcta y se obtenga el programa ejecutable, es necesario que el código fuente no contenga errores sintácticos. Si aparecen errores, es necesario volver a la fase de edición, guardar de nuevo el código fuente y repetir la fase de compilación.

Como resultado de la fase de compilación, en la parte de abajo del entorno debe aparecer un mensaje del tipo:

```
Compilation succeeded
```

Una vez compilado el programa, habremos obtenido el fichero `I_Pitagoras.exe`. Para ejecutarlo desde el entorno basta pulsar sobre F10. Si se quiere, ambos pasos (compilación y ejecución) pueden realizarse pulsando sobre F11. Debe aparecer una ventana de comandos del Sistema, en la que se estará ejecutando el programa. La ejecución del programa se detendrá en aquellos puntos del mismo donde se requiera la interacción del usuario para poder proseguir, es decir, en las operaciones de entrada de datos a través del dispositivo estándar de entrada. En este ejemplo, sería en las dos operaciones `cin`. En el resto de los casos, la ejecución del programa continuará hasta el final. La introducción de datos mediante la sentencia `cin` se hace siempre de la misma manera; primero se introduce el valor que se desee y al terminar se pulsa la tecla RETURN.

Introduzca ahora los valores pedidos en el ejemplo de Pitágoras y compruebe la respuesta del programa.

Como hemos indicado anteriormente, en la fase de generación del ejecutable se ha creado un fichero en el Sistema que se llama igual que nuestro fichero pero sustituyendo la extensión `"cpp"` por `"exe"`, es decir, `I_Pitagoras.exe`. Este fichero se encuentra en el mismo directorio que el del fichero `cpp`. Para mostrar que el fichero generado es independiente del entorno de programación, haga lo siguiente:

1. Cierre Orwell Dev C++.
2. Abra una ventana de Mi PC.
3. Sitúese en la carpeta que contiene el ejecutable.
4. Haga doble click sobre el fichero `I_Pitagoras.exe`.

Prueba del programa

Uno podría pensar que una vez que consigo un fichero ejecutable a partir de mi código fuente, el problema está terminado. Sin embargo esto no es así. Tras el proceso de compilado se

requiere una fase de prueba. Dicha fase intenta probar que el algoritmo planteado resuelve el problema propuesto. Para llevar a cabo esta fase, es necesario ejecutar el programa y verificar que los resultados que obtiene son los esperados.

Ahora que podemos ver el resultado obtenido por el programa implementado, verifiquemos mediante el siguiente conjunto de pruebas que el programa funciona de forma correcta.

lado1	lado2	hip
3	4	5
1	5	5.099
2.7	4.3	5.077
1.25	2.75	3.02

Una vez que el algoritmo supera la fase de prueba, podemos considerar que se ha concluido con la fase inicial del desarrollo del software.

Nota:

C++11 incorpora en la biblioteca `cmath` una función específica para calcular la hipotenusa, llamada `hypot`. Si lo desea, también puede comparar los resultados de su algoritmo con el ofrecido por dicha función

Introducción a la corrección de errores

Los errores de compilación

Ya hemos visto los pasos necesarios para construir un fichero ejecutable a partir del código fuente. El paso central de este proceso era la fase de compilación. En esta parte de este guión de prácticas aprenderemos a corregir los errores más comunes que impiden una compilación exitosa del fichero fuente.

Cargue el fichero `I_Pitagoras.cpp`. Quítele una 'u' a alguna aparición de `cout`. Intente compilar. Podemos observar que la compilación no se ha realizado con éxito. Cuando esto sucede, en la parte inferior de la ventana principal aparecen los errores que se han encontrado. Aparece una descripción del error, así como otra información, como el número de línea en la que se produjo. Los pasos que debemos seguir para la corrección son los siguientes:

1. Ir a la primera fila de la lista de errores.
2. **Leer el mensaje de error e intentar entenderlo.**
3. Hacer doble click sobre esa fila con el ratón. Esto nos posiciona sobre la línea en el fichero fuente donde el compilador detectó el error.

4. Comprobar la sintaxis de la sentencia que aparece en esa línea. Si se detecta el error, corregirlo. Si no se detecta el error mirar en la línea anterior, comprobar la sintaxis y repetir el proceso hasta encontrar el error.
5. Después de corregir el posible error, guardamos de nuevo el archivo y volvemos a compilar. Esto lo hacemos aunque aparezcan más errores en la ventana. La razón es que es posible que el resto de los errores sean consecuencia del primer error.
6. Si después de corregir el error aparecen nuevos errores, volver a repetir el proceso desde el paso 1.

A veces, el compilador no indica la línea exacta en la que se produce el error, sino alguna posterior. Para comprobarlo, haced lo siguiente:

- Comente la línea de cabecera `#include <iostream>` desde el principio. El compilador no reconocerá las apariciones de `cin` o `cout`.
- Quite un punto y coma al final de alguna sentencia. Dará el error en la línea siguiente.

Para familiarizarnos con los errores más frecuentes y su corrección vamos a realizar el siguiente proceso: a partir del código fuente del ejemplo `I_Pitagoras.cpp`, iremos introduciendo deliberadamente errores para conocer los mensajes que nos aparecen. A continuación se muestran algunos errores posibles. No deben introducirse todos ellos a la vez, sino que han de probarse por separado.

1. Cambie algún punto y coma por cualquier otro símbolo
2. Cambie `double` por `dpuble`
3. Cambie la línea `using namespace std;` por `using namespace STD;`
4. Ponga en lugar de `iostream`, el nombre `iotream`.
5. Borre alguno de los paréntesis de la declaración de la función `main`
6. Introduzca algún identificador incorrecto, como por ejemplo `cour`
7. Use una variable no declarada. Por ejemplo, en la definición de variables cambiad el nombre a la variable `lado1` por el identificador `lado11`.
8. Borre alguna de las dobles comillas en una constante de cadena de caracteres, tanto las comillas iniciales como las finales.
9. Borre alguna de las llaves que delimitan el inicio y final del programa.
10. Borre la línea `using namespace std;` (basta con comentarla con `//`)

11. Cambie un comentario iniciado con `//`, cambiando las barras anteriores por las siguientes `\`
12. Cambie la aparición de `<<` en `cout` por las flechas cambiadas, es decir, `>>`. Haced lo mismo con `cin`.
13. Suprima todo el `main`. No hace falta borrar el código, basta con comentarlo.

Además de los errores, el compilador puede generar *avisos*. Estos se muestran como **Warning** en la misma ventana de la lista de errores. Estas advertencias indican que algún código puede generar problemas durante la ejecución. Por ejemplo, al usar una variable que todavía no tiene un valor asignado, al intentar asignar un entero *grande* a un entero *chico*, etc. Sin embargo, no son errores de compilación, por lo que es posible generar el programa ejecutable correspondiente.

Los errores lógicos y en tiempo de ejecución

Aunque el programa compile, esto no significa que sea correcto. Puede producirse una excepción durante la ejecución, de forma que el programa terminará bruscamente (típico error en Windows de *Exception Violation Address*) o, lo que es peor, dará una salida que no es correcta (error lógico).

Sobre el programa `I_Pitagoras.cpp`, haga lo siguiente:

- Cambie la sentencia
`sqrt(lado1*lado1 + lado2*lado2)` por:
`sqrt(lado1*lado2 * lado2*lado2)`
Ejecute introduciendo los lados 2 y 3. El resultado no es correcto, pero no se produce ningún error de compilación ni en ejecución. Es un error lógico.
- Para mostrar un error de ejecución, declare tres variables **ENTERAS** (**tipo** `int`) `resultado`, `numerador` y `denominador`. Asígnele cero a `denominador` y 7 a `numerador`. Asígnele a `resultado` la división de `numerador` entre `denominador`. Imprima el resultado. Al ejecutar el programa, se produce una excepción o error de ejecución al intentar dividir un entero entre cero.

Creación de un programa nuevo

En esta sección vamos a empezar a crear nuestros propios programas desde Orwell Dev C++. El primer ejemplo que vamos a implementar corresponde al ejercicio 2 sobre la Ley de Ohm, de la relación de problemas I.

Para crear un programa nuevo, abrimos Orwell Dev C++ y elegimos

Archivo->Nuevo Código Fuente (Ctrl-N)

Para cambiar el nombre asignado por defecto, seleccionamos Archivo -> Guardar Como. Nos vamos a la carpeta U:\FP e introducimos el nombre I_Voltaje.

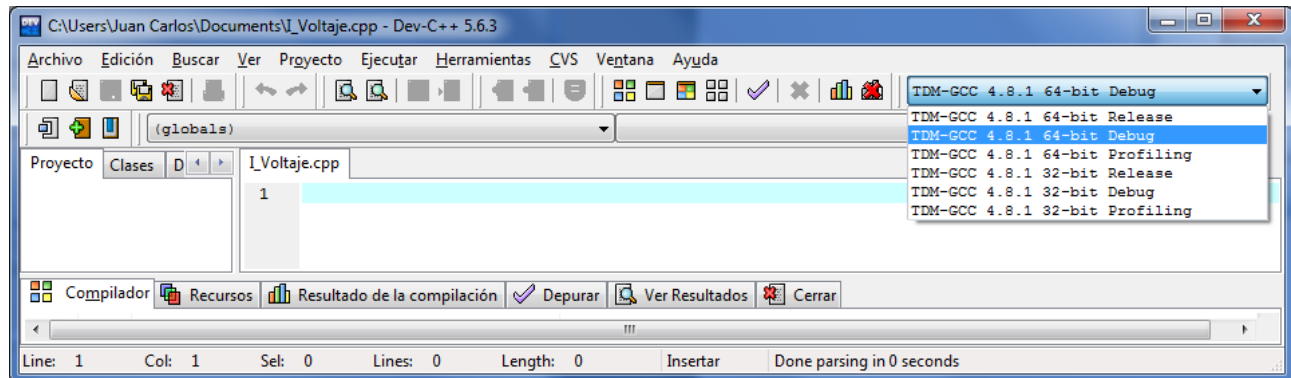


Figura 3: Creación de un programa nuevo

Confirme que en la esquina superior derecha está seleccionada la opción de compilación

TDM-GCC ... Debug

Ya estamos en condiciones de resolver el problema pedido. Escribimos el código en la ventana de edición. Habrá que leer desde teclado los valores de intensidad y resistencia y el programa imprimirá en pantalla el voltaje correspondiente. Recordad que compilamos con F9 y ejecutamos con F10, o directamente ambas acciones con F11.

Nota. Cuando tenemos varias variables en el código, podemos empezar a escribir el nombre de alguna de ellas y antes de terminar, pulsar Ctr-Barra espaciadora. La ayuda nos mostrará los identificadores disponibles que empiecen por las letras tecleadas.

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Cree un fichero cpp por cada uno de los ejercicios. Puede ir guardándolos en su unidad U: para poder acceder a ellos desde su casa. Estos ficheros no tiene que entregarlos en este momento. Los tendrá que entregar en PRADO para defenderlos en la semana del 25 al 29 de Septiembre. La fecha límite para entregarlos es el Lunes de la semana del 25 al 29 de Septiembre a las 9:30h.

Sesión 1

Tipos básicos y operadores

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

1 [Asignaciones secuenciales salario]

http://decsai.ugr.es/jccubero/FP/I_AsignacionesSecuenciales.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la relación I. Recuerde que debe subir a **PRADO** las soluciones de estos ejercicios, tal y como se explica en la página 2. Los alumnos defenderán su entrega durante las clases de prácticas de la semana del 25 al 29 de Septiembre.

Recuerde que debe subir un fichero llamado `s1.zip` que incluya todos los ficheros `cpp` (pero **no** los `.exe`). Nombre como quiera cada uno de los ficheros `cpp` pero no use acentos ni espacios en blanco y ponga al final del nombre el número del ejercicio.

- **Obligatorios:**

- 3 [Cálculo de π a partir del arco-seno]
- 4 [Conversión de grados a radianes]
- 5 [Tarifa aérea según km]
- 6 [Tarifa aérea con descuento]
- 7 [Interés bancario]
- 8 [Segundos entre dos instantes]

- **Opcionales:**

- 9 [Decimal redondeado]

► **Actividades a realizar en las aulas de ordenadores**

El profesor irá corrigiendo individualmente (a algunos alumnos elegidos aleatoriamente) los ejercicios indicados en la página anterior. Mientras tanto, los alumnos restantes deben intentar resolver los ejercicios de la próxima sesión de prácticas.

Sesión 2

► Actividades a realizar en casa

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

11 [Sistema métrico]

http://decsai.ugr.es/jccubero/FP/I_ConversionSistemaMetrico.cpp

12 [Circunferencia]

http://decsai.ugr.es/jccubero/FP/I_Circunferencia.cpp

Actividad: Resolución de problemas.

- **Obligatorios:**

- 10 [Intercambiar variables]

- 13 [Uso de constantes]

- 14 [Distancia Euclídea]

- 15 [Coordenadas geográficas]

- 16 [Gaussiana]

- **Opcionales:**

- 17 [Intercambiar tres variables]

- 18 [Desplaza entero dentro de un intervalo]

Actividades de Ampliación

Recuerde los conceptos de combinación y permutación, que irán apareciendo recurrentemente a lo largo de la carrera. Consulte, por ejemplo, la siguiente web, para una introducción básica a los conceptos:

<http://www.disfrutalasmaticas.com/combinatoria/combinaciones-permutaciones.html>

Si por ejemplo queremos ver las posibles combinaciones (con repetición e importando el orden) de dos elementos (0 y 1) en 4 posiciones de memoria (4 bits) obtenemos un total de $2^4 = 16$ posibilidades: 0000, 0001, 0010, ..., 1111. Ejecute el siguiente applet para ver las combinaciones resultantes:

<http://dm.udc.es/elearning/Applets/Combinatoria/index.html>



► **Actividades a realizar en las aulas de ordenadores**

El profesor irá corrigiendo individualmente (a algunos alumnos elegidos aleatoriamente) los ejercicios indicados en la página anterior. Mientras tanto, los alumnos restantes deben intentar resolver los ejercicios de la próxima sesión de prácticas.

Sesión 3

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

19 [Media y desviación]

http://decsai.ugr.es/jccubero/FP/I_Media.cpp

21 [Índice de mayúscula]

http://decsai.ugr.es/jccubero/FP/I_IndiceMayuscula.cpp

25 [Pasar de carácter a entero]

http://decsai.ugr.es/jccubero/FP/I_Caracter_a_Entero.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la relación de Problemas I:

- **Obligatorios:**

20 [Aproximación del valor de π]

22 [Descuento tarifa aérea mezclando tipos]

23 [Trunca decimales]

24 [Pasar de mayúscula a minúscula]

26 [Expresiones lógicas]

27 [Elección tipo de dato]

- **Opcionales:**

28 [Precisión y desbordamiento]

29 [Codificación de caracteres con algoritmo de rotación]

Actividades de Ampliación

Hojea la página

<http://catless.ncl.ac.uk/Risks>

que publica periódicamente casos reales en los que un mal desarrollo del software ha tenido implicaciones importantes en la sociedad.



► **Actividades a realizar en las aulas de ordenadores**

Redireccionando las entradas de cin

Cada vez que se ejecuta `cin`, se lee un dato desde el periférico por defecto. Nosotros lo hemos hecho desde el teclado, introduciendo un dato y a continuación un ENTER. Otra forma alternativa es introducir un dato y luego un separador (uno o varios espacios en blanco o un tabulador). Esto es posible gracias a la existencia de un buffer intermedio que canaliza el flujo de datos entre el programa y la consola. Para más detalle, consulte el final del Tema 1, disponible en [PRADO](#).

Para comprobarlo, copie localmente el fichero http://decsai.ugr.es/jccubero/FP/II_cin.cpp. El programa simplemente lee un entero y dos caracteres y los imprime en pantalla. Siempre hemos introducido cada valor y a continuación ENTER. Ahora lo hacemos de otra forma alternativa: introducimos los datos separados por espacios en blanco y pulsamos ENTER al final (una sola vez).

Para comprobar el correcto funcionamiento de nuestros programas, tendremos que ejecutarlos en repetidas ocasiones usando distintos valores de entrada. Este proceso lo repetiremos hasta que no detectemos fallos. Para no tener que introducir los valores pedidos uno a uno, podemos recurrir a un simple copy-paste. Para comprobarlo, cree un fichero de texto con un entero y dos caracteres. Separe estos tres datos con varios espacios en blanco. Seleccione con el ratón los tres y cópielos al portapapeles (Click derecho-Copiar). Ejecute el programa y cuando aparezca la consola del sistema haga click derecho sobre la ventana y seleccione Editar-Pegar. También puede usar la típica combinación de teclas Ctrl-C para copiar y Ctrl-V para pegar.

Otra alternativa es ejecutar el fichero `.exe` desde el sistema operativo y redirigir la entrada de datos al fichero que contiene los datos. Para poder leer los datos del fichero, basta con ejecutar el programa `.exe` desde una consola del sistema y especificar que la entrada de datos será desde un fichero a través del símbolo de redireccionamiento `<` (no ha de confundirse con el token `<<` que aparecía en una instrucción `cout` de C++) Hay que destacar que este redireccionamiento de la entrada lo estamos haciendo en la llamada al ejecutable desde la consola del sistema operativo¹. Para probarlo, descargue el fichero http://decsai.ugr.es/jccubero/FP/II_cin_datos_entrada.txt y cópielo dentro de la misma carpeta en la que tiene el programa `II_cin.exe`. Abrimos dicha carpeta desde el explorador y seleccionamos con el click derecha del ratón "Abrir Símbolo del Sistema"². Introducimos la instrucción siguiente:

¹También pueden leerse datos de un fichero desde dentro del propio código fuente del programa, pero esto se verá en el segundo cuatrimestre

²Para poder lanzar una consola desde el explorador de Windows, en nuestra casa, o bien instalamos un programa que permita abrir una consola en el directorio actual, como por ejemplo *Open Command Prompt Shell Extension* disponible en <http://code.kliu.org/cmdopen/> o bien abrimos un símbolo del sistema (Inicio->Ejecutar->cmd) y vamos cambiando de directorio con la orden `cd`

```
II_cin.exe < II_cin_datos_entrada.txt
```

Ahora, cada vez que se ejecute una instrucción `cin` en el programa, se leerá un valor de los presentes en el fichero de texto.

Cuando ejecutemos el programa, cada ejecución de `cin` leerá un dato desde el fichero indicado, saltándose todos los espacios en blanco y tabuladores que hubiese previamente. Cuando llegue al final del fichero, cualquier entrada de datos posterior que realicemos dará un *fallo*.

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 4

Estructura condicional

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

6 [Convertir mayúscula en minúscula]

http://decsai.ugr.es/jccubero/FP/II_Mayuscula.cpp

7 [Se dividen]

http://decsai.ugr.es/jccubero/FP/II_SeDividen.cpp

9 [Pasar de mayúscula a minúscula y viceversa]

http://decsai.ugr.es/jccubero/FP/II_MayusculaMinusculaViceversa.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la relación II.

Importante: En estos ejercicios se permite mezclar E/S con cálculos dentro del mismo condicional

- **Obligatorios:**

- 1 [Mismo signo -multiplicando-]

- 2 [Codificación de caracteres con algoritmo de rotación. Condicional simple]

- 3 [Tarifa aérea]

- 4 [Operadores lógicos]

- 5 [Coordenadas geográficas (distancia)]

- 8 [Tarifa aérea con descuentos]

- **Opcionales:**

- 10 [Comparación de dos instantes]

- 11 [Multiautovía]

► Actividades a realizar en las aulas de ordenadores

En esta sesión empezaremos a trabajar en el aula con las estructuras condicionales. Es muy importante poner atención a la tabulación correcta de las sentencias, tal y como se indica en las transparencias. Recordad que una tabulación incorrecta supondrá bajar puntos en la primera prueba práctica que se realizará dentro de algunas semanas.

El entorno de compilación incluirá automáticamente los tabuladores cuando iniciemos una estructura condicional (lo mismo ocurrirá cuando veamos las estructuras repetitivas). En cualquier caso, si modificamos el código y añadimos/suprimimos estructuras anidadas (`if` dentro de otro `if` o `else`) podemos seleccionar el texto del código deseado y pulsar la tecla de tabulación para añadir margen o `Shift`+tabulación para quitarlo.

Como base para esta práctica vamos a emplear el ejercicio de la subida salarial visto en clase de teoría. Por simplicidad, usamos la versión que usa literales en el código³

http://decsai.ugr.es/jccubero/FP/II_actualizacion_salarial_con_literales.cpp

En primer lugar, copiamos en nuestra carpeta local el anterior fichero.

Fuerce los siguientes errores en tiempo de compilación, para ver los mensajes de error ofrecidos por el compilador:

- Suprima los paréntesis de alguna de las expresiones lógicas de las sentencias condicionales.
- Quite la llave abierta `{` del condicional de la expresión `edad >= 45 && salario_base < 1300`
- Quite la llave cerrada `}` de la anterior sentencia condicional.
- ¿Qué pasaría si quitásemos las llaves de la sentencia correspondiente a la condición `(experiencia > 2)`?

Depuración

" If debugging is the process of removing bugs, then programming must be the process of putting them in. Edsger Dijkstra (1930/2002) "



Un depurador de programas (*debugger* en inglés) permite ir ejecutando un programa sentencia a sentencia (ejecución paso a paso). Además, nos permite ver en cualquier momento

³Recuerde que siempre debemos evitar usar literales en las expresiones. La versión de este programa que usa constantes se encuentra en: http://decsai.ugr.es/jccubero/FP/II_actualizacion_salarial.cpp

el valor de las variables usadas por el programa. El uso de un depurador facilita la localización de errores lógicos en nuestros programas, que de otra forma resultarían bastante difíciles de localizar directamente en el código.

"Debuggers don't remove bugs. They only show them in slow motion".

Para poder realizar tareas de depuración en Dev C++ debemos asegurarnos que estamos usando un perfil del compilador con las opciones de depuración habilitadas. Si cuando configuramos el compilador seleccionamos **Herramientas | Opciones del Compilador | Compilador a configurar: Debug** nuestro entorno estará preparado para depurar programas.

Si no fuera así, al intentar depurar el programa, Dev C++ nos mostrará la ventana de la figura 4.

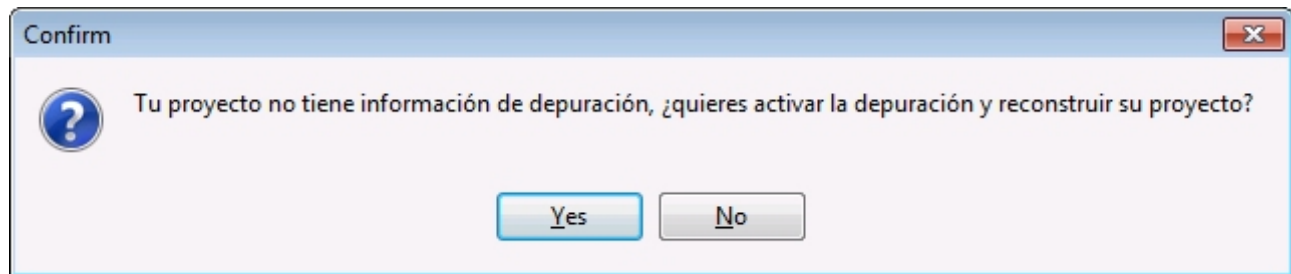


Figura 4: Ventana emergente que aparece cuando la configuración actual del compilador no permite tareas de depuración

La idea básica en la depuración es ir ejecutando el código *línea a línea* para ver posibles fallos del programa. Para eso, debemos dar los siguientes pasos:

1. Establecer una línea del programa en la que queremos que se pare la ejecución. Lo haremos introduciendo un **punto de ruptura** o (*breakpoint*) en dicha línea. Si sospechamos dónde puede estar el error, situaremos el punto de ruptura en dicha línea. En caso contrario, lo situaremos:
 - a) al principio del programa, si no sabemos exactamente dónde falla el programa, o
 - b) al principio del bloque de instrucciones del que desconfiamos, siempre y cuando tengamos confianza en todas las instrucciones que se ejecutan antes.

Para establecer un punto de ruptura podemos mover el ratón en la parte más a la izquierda de una línea de código (o sobre el número de línea) y pulsar el botón izquierdo del ratón en esa posición. La instrucción correspondiente queda marcada en rojo. Si en esa línea ya había un punto de ruptura, entonces será eliminado. También podemos colocar el cursor sobre la instrucción y con el menú contextual (botón derecho del ratón) seleccionar **Añadir/Quitar Punto de Ruptura** o simplemente,

pulsar **F4**. Para eliminar un punto de ruptura, se realiza la misma operación que para incluirlo, sobre la instrucción que actualmente lo tiene.

Coloque ahora un punto de ruptura sobre la línea que contiene la primera sentencia condicional `if` (figura 5).

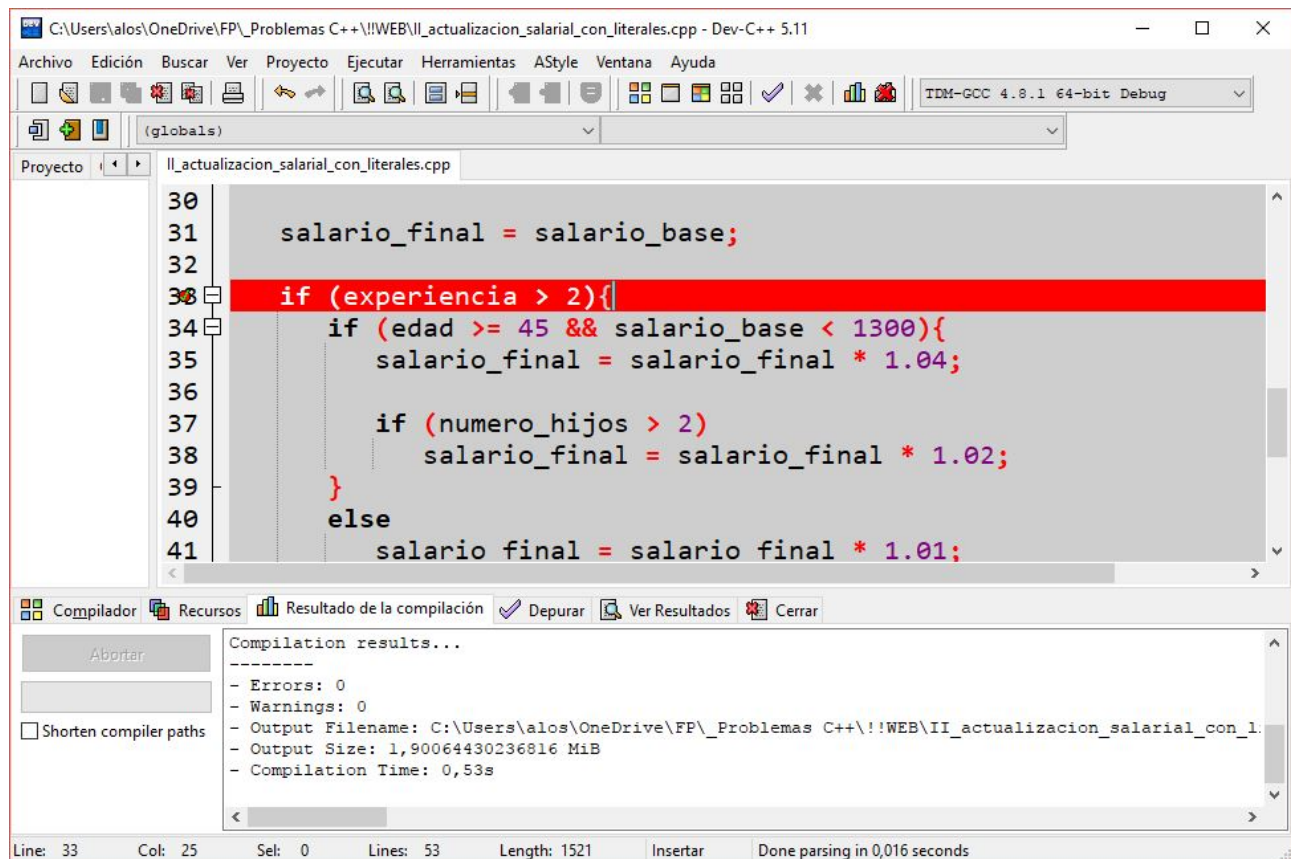


Figura 5: Se ha activado un punto de ruptura

2. Comenzar la depuración:

- pulsar **F5**,
- pulsar sobre el icono
- seleccionar en el menú Ejecutar | Depurar, ó
- en la zona inferior, pestaña Depurar, pulsar el botón **Depurar** (figura 6)

Muy importante: Si se escoge *ejecutar* en lugar de *depurar*, el programa se ejecuta normalmente, sin detenerse en los puntos de ruptura.

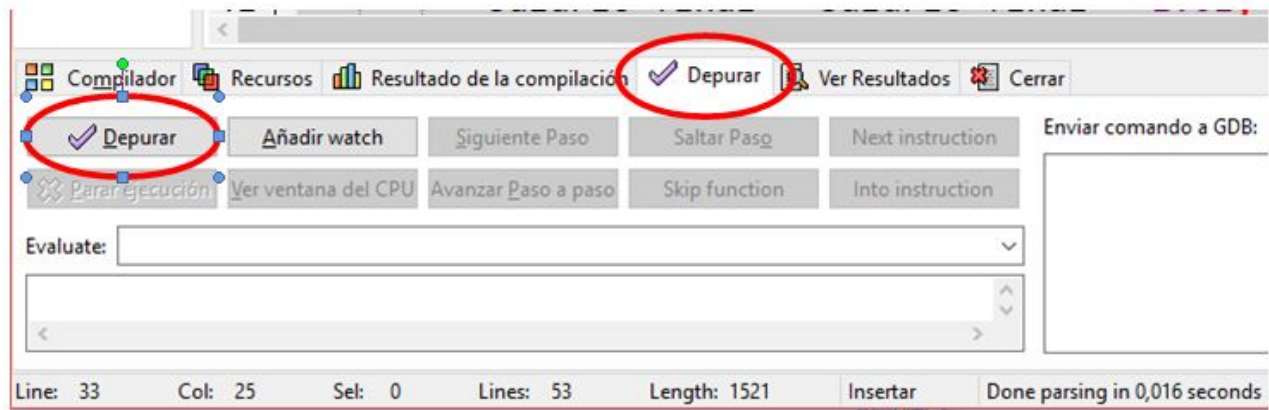


Figura 6: Inicio del proceso de depuración

Al iniciarse la depuración se ejecutan todas las sentencias hasta alcanzar el primer punto de ruptura. Llegado a este punto, la ejecución se interrumpe (queda “en espera”) y se muestra en azul (figura 7) la línea que se va a ejecutar a continuación (en este caso, la que contiene el punto de interrupción).

Ahora podemos escoger entre varias alternativas, todas ellas accesibles en la zona inferior (pestaña **Depurar**) pulsando el botón correspondiente (ver figura 7):

- **Parar ejecución**: Detener la depuración (y ejecución) del programa.
- **Siguiente Paso (F7)**: Ejecuta la siguiente instrucción. Si se trata de una llamada a una función, la ejecuta y continúa con la siguiente instrucción, sin entrar a ejecutar las instrucciones internas de la función. Las funciones se verán dentro de dos semanas.
- **Avanzar Paso a paso (F8)**: Ejecuta la siguiente instrucción. Si se trata de una llamada a una función, entra en la función y ejecuta la primera instrucción de la función, continuando la depuración dentro de la función.
- **Saltar Paso**: Ejecuta todas las instrucciones hasta encontrar un nuevo punto de ruptura, o llegar al final del programa.

La posibilidad de ver el valor de los datos que gestiona el programa durante su ejecución hace que sea más sencilla y productiva la tarea de la depuración.

La manera más sencilla de comprobar el valor que tiene una variable es colocar el cursor sobre el nombre de la variable y esperar un instante. Veremos un globo que nos muestra el nombre y valor de la variable. El inconveniente es que al mover el ratón desaparece el globo, y cuando queramos inspeccionar nuevamente el valor de la variable debemos repetir la operación.

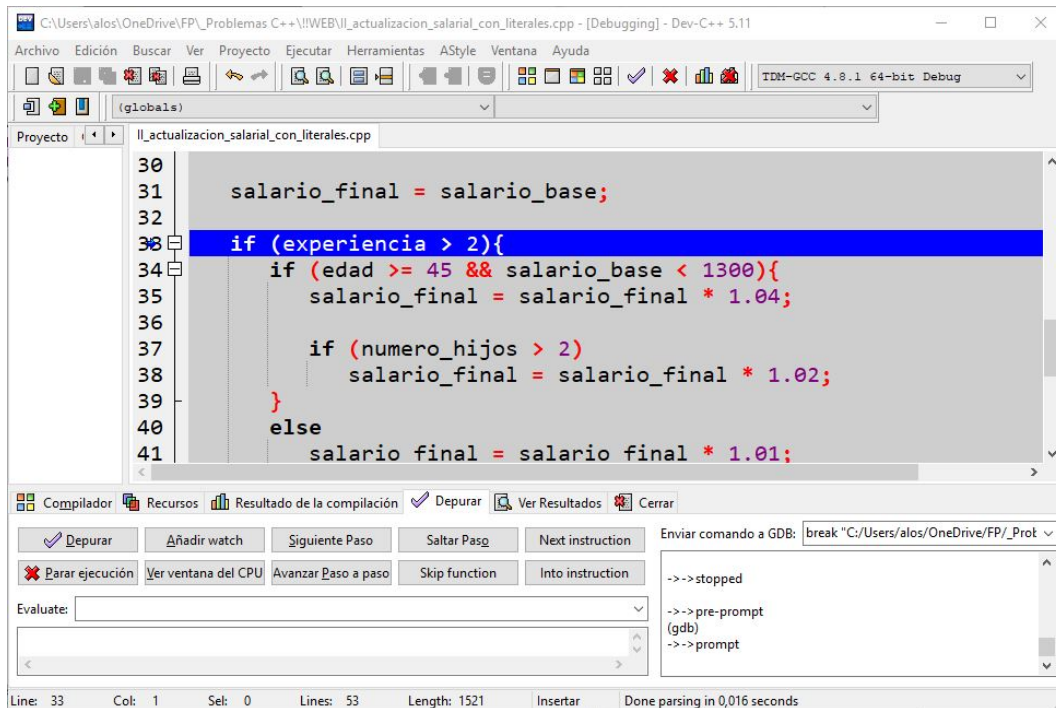


Figura 7: Inicio del proceso de depuración

Podemos mantener variables permanentemente monitorizadas. Aparecerán en el Explorador de Proyectos/Clases (seleccionar la pestaña Depurar).

Para añadir una variable podemos:

1. colocar el cursor sobre la variable y con el menú contextual (botón derecho del ratón) seleccionar **Añadir watch**. Aparecerá una ventana con el nombre de la variable preseleccionado. Al seleccionar OK aparece la información de esa variable en el Explorador de Proyectos/Clases (figura 8)
2. abrir el menú contextual (botón derecho del ratón) en cualquier lugar del editor, seleccionar **Añadir watch** y escribir el nombre de la variable,
3. abrir el menú contextual en el Explorador de Proyectos/Clases (pestaña Depurar), seleccionar **Añadir watch** y escribir el nombre de la variable,
4. pulsar el botón **Añadir watch** en la zona inferior (pestaña Depurar) y escribir el nombre de la variable.

Conforme se ejecuta el programa podremos ver cómo cambian los valores de las variables monitorizadas.

También podríamos, incluso, modificar su valor directamente pinchando con el botón derecho sobre la variable y seleccionando **Modificar Valor**.

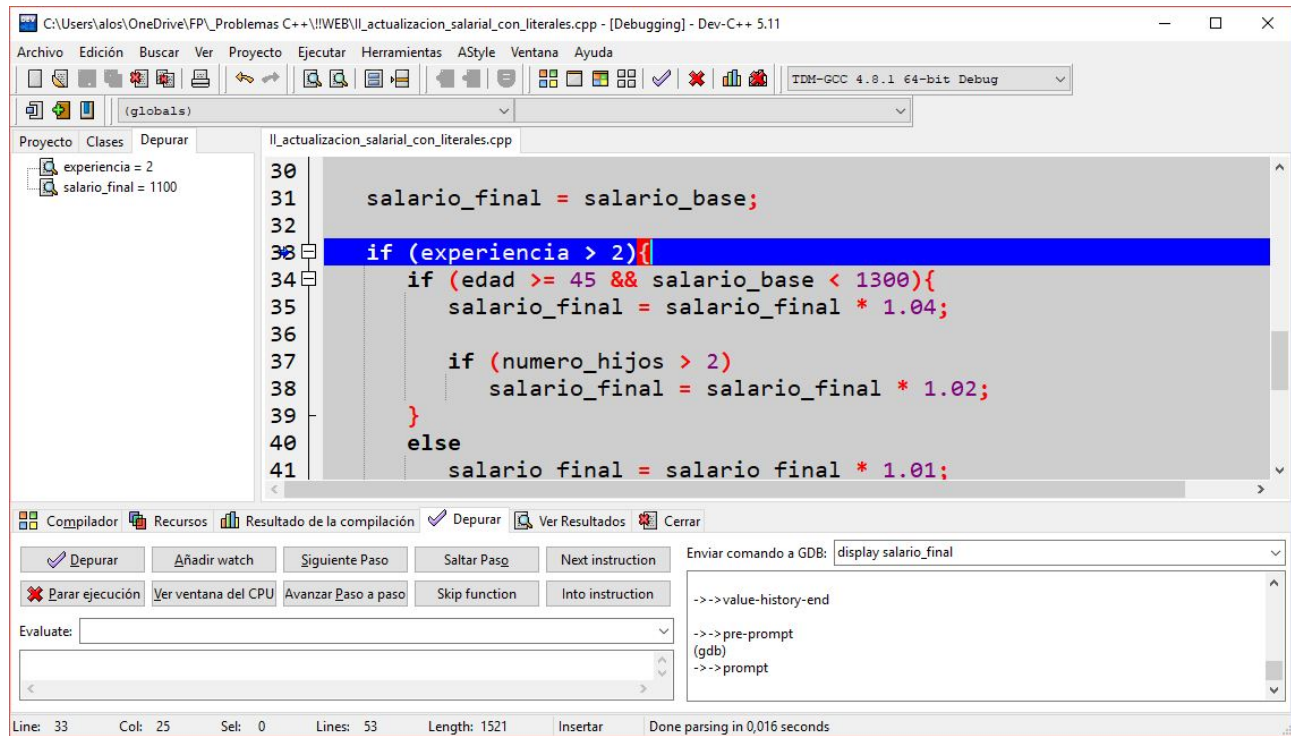


Figura 8: Añadiendo una variable para su inspección permanente

Otras dos opciones accesibles desde el Explorador de Proyectos/Clases (pestaña Depurar), son Quitar watch para eliminar una variable y Clear All para eliminarlas todas,

Observación final: El depurador ayuda a encontrar errores al permitir ejecutar las sentencias paso a paso y así comprobar por donde va el flujo de control y ver cómo van cambiando las variables. En cualquier caso, nunca nos cansaremos de repetir que el mejor programador es el que piensa la solución en papel, antes de escribir una sola línea de código en el entorno de programación.

"When your code does not behave as expected, do not use the debugger, think".



Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 5

Estructura Repetitiva. Bucles `while`

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

13 [Mismo signo separando E/S y C]

http://decsai.ugr.es/jccubero/FP/II_MismoSignoMultiplicando.cpp

16 [Divisores de un entero]

http://decsai.ugr.es/jccubero/FP/II_Divisores.cpp

17 [Mínimo de varios valores]

http://decsai.ugr.es/jccubero/FP/II_Min.cpp

Actividad: Resolución de problemas.

- **Obligatorios:**

Ejercicios sobre condicionales:

12 [Velocidad imputada]

14 [Mismo signo con condicionales]

15 [Mayúscula a minúscula y viceversa usando un enumerado]

Ejercicios sobre bucles:

18 [Tarifa aérea con filtro de entrada de datos]

19 [Valores de la Gaussiana]

- **Opcionales:**

20 [Número desgarrable]

► **Actividades a realizar en las aulas de ordenadores**

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 6

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

26 [RLE]

http://decsai.ugr.es/jccubero/FP/II_RLE.cpp

27 [Número Narcisista]

http://decsai.ugr.es/jccubero/FP/II_Narcisista.cpp

► **Actividades a realizar en casa**

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la Relación de Problemas II.

Importante: En estos ejercicios se permiten mezclar E/S con cálculos dentro del mismo bucle (ya que todavía no se conocen herramientas para no hacerlo)

- **Obligatorios:**

- 22 [Velocidad imputada -lectura en bucle-]

- 23 [Aproximación de π por Gregory-Leibniz]

- 24 [Aproximación de π por Wallis]

- **Opcionales:**

- 25 [Aproximación de π por Madhava sin usar pow]

► **Actividades a realizar en las aulas de ordenadores**

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 7

Estructura Repetitiva: bucles for y bucles anidados

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

28 [Parejas de caracteres]

http://decsai.ugr.es/jccubero/FP/II_Parejas.cpp

29 [Pirámide] y 30 [Cuadrado]

http://decsai.ugr.es/jccubero/FP/II_PiramideCuadrado.cpp

31 [Gaussiana con un menú]

http://decsai.ugr.es/jccubero/FP/II_Gaussiana.cpp

Actividad: Resolución de problemas.

Importante: En estos ejercicios se permiten mezclar E/S con cálculos dentro del mismo bucle (ya que todavía no se conocen herramientas para no hacerlo)

- **Obligatorios:**

- 32 [Tarifa aérea: múltiples billetes]

- 33 [Bits to char]

- 34 [Cuadro de límites de velocidad]

- **Opcionales:**

- 35 [Número feliz]

► **Actividades a realizar en las aulas de ordenadores**

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 8

Vectores y Matrices

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

1 [Palíndromo e Invierte]

http://decsai.ugr.es/jccubero/FP/III_palindromo_invierte.cpp

5 [Elimina ocurrencias de una componente -versión ineficiente-]

http://decsai.ugr.es/jccubero/FP/III_elimina_ocurrencias_ineficiente.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes ejercicios de la relación de problemas III:

- **Obligatorios:**

2 [Comprobación Fecha]

3 [Máximo desnivel]

4 [Sistema de D'Hondt]

7 [Sustituir carácter por vector (con vector auxiliar)]

8 [Sustituir carácter por vector (versión ineficiente)]

- **Opcionales:**

9 [Sustituir carácter por vector (versión eficiente)]

► **Actividades a realizar en las aulas de ordenadores**

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 9

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

10 [Login]

http://decsai.ugr.es/jccubero/FP/III_login.cpp

12 [Top k (versión ineficiente)]

http://decsai.ugr.es/jccubero/FP/III_topk_ineficiente.cpp

14 [Traspuesta]

http://decsai.ugr.es/jccubero/FP/III_traspuesta.cpp

15 [Producto de matrices]

http://decsai.ugr.es/jccubero/FP/III_producto_matrices.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes ejercicios de la relación de problemas III:

- **Obligatorios:**

6 [Elimina ocurrencias de una componente -versión eficiente-]

11 [Elimina varios]

16 [Límites de velocidad -en una matriz-]

17 [Máximo de los mínimos]

- **Opcionales:**

13 [Top k (versión eficiente)]

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

► **Actividades a realizar en las aulas de ordenadores**

Sesión 10

Funciones

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

1 [Errores en funciones]

http://decsai.ugr.es/jccubero/FP/IV_errores_funciones.cpp

2 [Lee entero en rango]

http://decsai.ugr.es/jccubero/FP/IV_lee_int_rango.cpp

4 [Máximo de 3 valores]

http://decsai.ugr.es/jccubero/FP/IV_max.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes ejercicios de la relación de problemas III:

- **Obligatorios:**

3 [Lee entero mayor o igual que otro]

5 [Potencia entera]

6 [Diferencia entre instantes]

7 [Decimal redondeado]

8 [Elimina últimos]

9 [double to string]

- **Opcionales:**

10 [double to string mejorado]

► **Actividades a realizar en las aulas de ordenadores**

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 11

Clases

► Actividades a realizar en casa

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

11 [Recta]

http://decsai.ugr.es/jccubero/FP/IV_Recta.cpp

12 [Cronómetro]

http://decsai.ugr.es/jccubero/FP/IV_Cronometro.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes ejercicios de la relación de problemas III:

- *Obligatorios:*

- 13 [Generador aleatorio]

- 14 [Coordenadas geográficas]

- 15 [Dinero]

- 16 [Instante]

- 17 [Formateador de doubles]

- *Opcionales:*

- 18 [Calificación final]

► Actividades a realizar en las aulas de ordenadores

Empiece a trabajar con los ejercicios incluidos en la próxima sesión.

Sesión 12

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios:

19 [Palíndromo e invierte dentro de la clase `SecuenciaCaracteres`]

21 [Elimina ocurrencias eficiente]

24 [Moda]

La solución de los tres ejercicios se encuentra en:

http://decsai.ugr.es/jccubero/FP/IV_SecuenciaCaracteres.cpp

Actividad: Resolución de problemas.

Resuelva los siguientes ejercicios de la relación de problemas IV:

- *Obligatorios:*

- 20 [Elimina ocurrencias ineficiente]

- 25 [Tabla de temperaturas]

- 26 [Túnel]

- *Opcionales:*

- 22 [Elimina repetidos ineficiente]

- 23 [Elimina repetidos eficiente]



Fundamentos de Programación.

Relaciones de Problemas.

© CopyRight: Juan Carlos Cubero. Universidad de Granada.

Sugerencias: por favor, enviar un e-mail a JC.Cubero@decsai.ugr.es

RELACIÓN DE PROBLEMAS I. Introducción a C++

1. [Asignaciones secuenciales salario] Indique cuál sería el resultado de las siguientes operaciones:

```
int salario_base;
int salario_final;
int incremento;

salario_base = 1000;
incremento = 200;

salario_final = salario_base;
salario_final = salario_final + incremento;

salario_base = 3500;

cout << "\nSalario base: " << salario_base;
cout << "\nSalario final: " << salario_final;
```

Responda razonadamente a la siguiente pregunta: ¿El hecho de realizar la asignación `salario_final = salario_base;` hace que ambas variables estén ligadas durante todo el programa y que cualquier modificación posterior de `salario_base` afecte a `salario_final`?

*Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión.
Dificultad Baja.*

2. [Ley de Ohm] Cree un programa que pida un valor de intensidad y resistencia e imprima el voltaje correspondiente, según la *Ley de Ohm*:

$\text{voltaje} = \text{intensidad} * \text{resistencia}$

*Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión.
Dificultad Baja.*

3. [Cálculo de π a partir del arco-seno] Si dividimos la longitud de cualquier circunferencia por el doble de su radio, se obtiene siempre el mismo número real, a saber 3.1415927... Este número es bien conocido y se designa por la letra griega π . Es irracional y por tanto tiene infinitas cifras decimales. Hay diversos métodos para obtener este valor, algunos mejores que otros. En posteriores ejercicios veremos algunos de ellos. En la página web https://en.wikipedia.org/wiki/Chronology_of_computation_of_%CF%80 se puede consultar los últimos logros en la obtención de nuevas cifras decimales de π .

Una forma de obtener π es a través de la función arco-seno, ya que $\frac{\pi}{6}$ es el ángulo en radianes de un seno de 0.5:

$$\frac{\pi}{6} = \text{arco-seno}(0.5)$$

Construir un programa que imprima el valor de π calculado a partir de la anterior fórmula.

Recuerde que la función arco-seno está implementada en la función `asin` de la biblioteca `cmath`.

La salida debe ser 3.14159

Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión. Dificultad Baja.

4. [Conversión de grados a radianes] Queremos transformar una cantidad g dada en grados a radianes. Para ello, basta aplicar la siguiente fórmula:

$$r = g \frac{\pi}{180}$$

Construya un programa que lea dos números reales que representarán dos grados. Debe calcular los radianes correspondientes e imprimir el resultado en pantalla.

Ejemplo de entrada: 20 90
— Salida correcta: 0.349066 1.5708
Ejemplo de entrada: 180 0
— Salida correcta: 3.14159 0

Para representar π puede usar el literal 3.14159 o bien la fórmula indicada en el ejercicio 3 [Cálculo de π a partir del arco-seno] .

Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión. Dificultad Baja.

5. [Tarifa aérea según km] Una compañía aérea establece el precio del billete como sigue: en primer lugar se fija una tarifa base de 150 euros, la misma para todos los destinos. A continuación se suman 10 céntimos por cada kilómetro de distancia al destino.

Cree un programa que lea el número de kilómetros al destino y calcule el precio final del billete. Utilice el tipo de dato `double` para todas las variables del programa.

Ejemplo de entrada: 120 — Salida correcta: 162
Ejemplo de entrada: 200 — Salida correcta: 170
Ejemplo de entrada: 201 — Salida correcta: 170.1
Ejemplo de entrada: 452 — Salida correcta: 195.2

Finalidad: Trabajar con expresiones numéricas. Dificultad Baja.

6. [Tarifa aérea con descuento] Una compañía aérea quiere aplicar una política de descuentos al precio final de un billete. Construya un programa que lea el precio del billete y aplique e imprima el resultado de aplicar los siguientes descuentos (de forma independiente):

- a) Un descuento del 4%
- b) Un descuento del 2%

Utilice el tipo de dato `double` para todas las variables del programa.

Ejemplo de entrada: 300 — Salida correcta: 288 294

Ejemplo de entrada: 500 — Salida correcta: 480 490

Ejemplo de entrada: 0 — Salida correcta: 0 0

Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión. Dificultad Baja.

7. [Interés bancario] Un banco presenta la siguiente oferta. Si se deposita una cantidad de euros dada por la variable `capital` durante un año a plazo fijo, se dará un interés dado por la variable `interes`. Realice un programa que lea una cantidad `capital` y un interés `interes` desde teclado. A continuación, el programa debe calcular en una variable `total` el dinero que se tendrá al cabo de un año, aplicando la fórmula de abajo e imprimirá el resultado en pantalla.

$$\text{total} = \text{capital} + \text{capital} * \frac{\text{interes}}{100}$$

Utilice el tipo de dato `double` para todas las variables. Supondremos que el usuario introduce el interés como un valor real entre 0 y 100, es decir, un interés del 5.4 % se introducirá como 5.4. También supondremos que lo introduce correctamente, es decir, que sólo introducirá valores entre 0 y 100.

Observe que para implementar la fórmula anterior, debemos usar el operador de división que en C++ es `/`, por lo que nos quedaría:

```
total = capital + capital * interes / 100;
```

Es importante destacar que el compilador primero evaluará la expresión de la parte derecha de la anterior asignación (usando el valor que tuviese la variable `capital`) y a continuación ejecutará la asignación, escribiendo el valor resultante de la expresión dentro de la variable `total`.

En la asignación que calcula la variable `total`, ¿se podría sustituir dicha variable por `capital`? es decir:

```
capital = capital + capital * interes / 100;
```

Analice las ventajas o inconvenientes de hacerlo así.

Ejemplo de entrada: 300 5.4 — Salida correcta: 316.2

Ejemplo de entrada: 300 0 — Salida correcta: 300

Ejemplo de entrada: 0 5.4 — Salida correcta: 0

Finalidad: Resolver un problema real sencillo, usando varias sentencias. Dificultad Baja.

8. [Segundos entre dos instantes] Calcule el número de segundos que hay entre dos instantes del mismo día.

Cada instante se caracteriza por la hora (entre 0 y 23), minuto (entre 0 y 59) y segundo (entre 0 y 59).

El programa leerá la hora, minuto y segundo del instante inicial y la hora, minuto y segundo del instante final (supondremos que los valores introducidos son correctos) y mostrará el número de segundos entre ambos instantes. Si el segundo instante es menor que el primero, se devolverá el número de segundos en negativo.

Ejemplo de entrada: 9 12 9 10 34 55 — Salida correcta: 4966

Ejemplo de entrada: 10 34 55 9 12 9 — Salida correcta: -4966

Ejemplo de entrada: 10 34 55 10 34 55 — Salida correcta: 0

Finalidad: Trabajar con expresiones numéricas y algoritmos. Dificultad Baja.

9. [Decimal redondeado] En `cmath` está definida la función `round` que permite redondear un real al entero más próximo. Por ejemplo:

`round(3.6)` devuelve 4

`round(3.5)` devuelve 4

`round(3.1)` devuelve 3

`round(3.49)` devuelve 3

Construya un programa para redondear un número real a cualquier cifra decimal. Por ejemplo:

El resultado de redondear 3.49 a la primera cifra decimal es 3.5

El resultado de redondear 3.49 a la segunda cifra decimal es 3.49

El resultado de redondear 3.496 a la segunda cifra decimal es 3.5

El programa principal leerá el número real y la posición de la cifra decimal e imprimirá el resultado. Puede usar la misma función `round` y la función `pow`, ambas incluidas en `cmath`. La función `pow` eleva un número a otro; por ejemplo, para elevar 3.5 a 7, basta usar la expresión `pow(3.5, 7)`

Ejemplo de entrada: 3.49 1 — Salida correcta: 3.5

Ejemplo de entrada: 3.49 2 — Salida correcta: 3.49

Ejemplo de entrada: 3.496 2 — Salida correcta: 3.5

Finalidad: Practicar la construcción de expresiones matemáticas. Dificultad Baja.

10. **[Intercambiar variables]** Queremos construir un programa que simule un juego inspirado en el de los *triles* (del que procede el nombre de *trilero*). Suponemos que hay dos participantes y cada uno tiene una caja etiquetada con su nombre. Dentro de cada caja hay una cantidad de dinero y el objetivo es intercambiar las cantidades que hay dentro. Por ahora, sólo se pide construir un programa que haga lo siguiente:

- Debe leer desde teclado dos variables `caja_izda` y `caja_dcha`.
- A continuación debe intercambiar sus valores y finalmente, mostrarlos en pantalla.

Observe que se desea intercambiar el contenido de las variables, de forma que `caja_izda` pasa a contener lo que tenía `caja_dcha` y viceversa. El siguiente código no es válido ya que simplemente engaña al usuario pero las cajas no se quedan modificadas:

```
cout << "La caja izquierda vale " << caja_dcha << "\n";  
cout << "La caja derecha vale " << caja_izda;
```

Estaríamos tentados a escribir el siguiente código:

```
caja_izda = caja_dcha;  
caja_dcha = caja_izda;
```

pero no funciona correctamente ¿Por qué?

Proponga una solución e impleméntela.

Finalidad: Entender cómo funciona la asignación entre variables. Dificultad Baja.

11. **[Sistema métrico]** Realice un programa que nos pida una longitud cualquiera dada en metros. El programa deberá calcular e imprimir en pantalla el equivalente de dicha longitud en pulgadas, pies, yardas y millas. Para el cálculo, utilice la siguiente tabla de conversión del sistema métrico:

1 pulgada= 25,4 milímetros
1 pie = 30,48 centímetros
1 yarda = 0,9144 metros
1 milla = 1609,344 metros
1 milla marina = 1852 metros

Ejemplo de entrada: 1 — — Salida correcta: 39.3701 3.28084 1.09361 0.00062

Finalidad: Plantear la solución de un ejercicio básico como es el de una conversión. Dificultad Baja.

12. **[Circunferencia]** Cree un programa que nos pida la longitud del radio y calcule el área del círculo y la longitud de la circunferencia correspondientes. Finalmente, el programa mostrará en pantalla los resultados. Recuerde que:

$$\text{área círculo} = \pi r^2 \quad \text{longitud circunferencia} = 2\pi r$$

En primera instancia, use como π el valor 3.1416. A continuación cambie el valor por 3.1415927, recompile y ejecute.

Ejemplo de entrada: 3 — Salida correcta: 28.274 18.849

Ejemplo de entrada: 0 — Salida correcta: 0 0

Finalidad: Uso de constantes. Dificultad Baja.

13. **[Uso de constantes]** Re-escriba las soluciones de los ejercicios 3 **[Cálculo de π a partir del arco-seno]**, 4 **[Conversión de grados a radianes]**, 5 **[Tarifa aérea según km]** y 6 **[Tarifa aérea con descuento]** usando constantes donde estime necesario.

Finalidad: Uso de constantes. Dificultad Baja.

14. **[Distancia Euclídea]** Cree un programa que lea las coordenadas de dos puntos $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$ y calcule la distancia euclídea entre ellos:

$$d(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Debe calcular el cuadrado sin usar ninguna función de la biblioteca `cmath`. Para la raíz cuadrada, sí puede usar la función `sqrt`.

Ejemplo de entrada: 2.1 3.2 0.5 1.6 — Salida correcta: 2.26274

Ejemplo de entrada: 2.1 3.2 2.1 3.2 — Salida correcta: 0

Finalidad: Trabajar con expresiones numéricas y con variables para no repetir cálculos. Dificultad Baja.

15. **[Coordenadas geográficas]** En ejercicios posteriores, trabajaremos con un conjunto de coordenadas geográficas y necesitaremos realizar ciertas operaciones trigonométricas. En este ejercicio, vamos a implementar una parte de dichas operaciones.

Se pide construir un programa que lea cuatro variables reales `grados_lat1`, `grados_lon1`, `grados_lat2`, `grados_lon2` representando las coordenadas (en grados) de longitud y latitud de dos puntos en el plano y calcule el valor de la variable a aplicando la siguiente fórmula:

$$a = \sin^2 \left(\frac{1}{2}(\text{lat}_2 - \text{lat}_1) \right) + \cos(\text{lat}_1) \cos(\text{lat}_2) \sin^2 \left(\frac{1}{2}(\text{lon}_2 - \text{lon}_1) \right)$$

Los valores de lat_i y lon_i son los valores correspondientes a `grados_lati`, `grados_loni`, pero expresados en radianes, por lo que tendrá que utilizar la conversión vista en el ejercicio 4 **[Conversión de grados a radianes]**

RELACIÓN DE PROBLEMAS I. Introducción a C++

Tenga en cuenta que si escribe la expresión $1/2$ el resultado es cero ya que, al ser los operandos de tipo entero, el operador $/$ es la división entera.

Ejemplo de entrada: 37.18817 -3.60667 41.89193 12.51133

— Salida correcta: 0.0133395

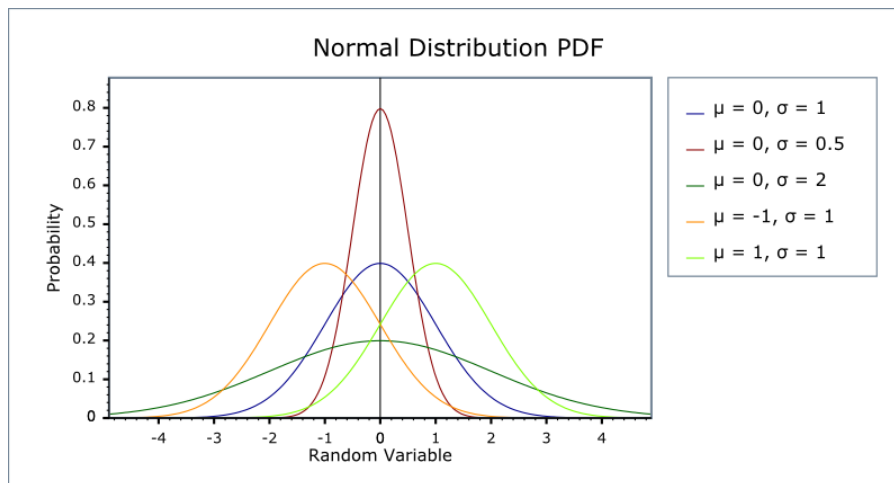
Los datos anteriores corresponden a la latitud y longitud de Granada y Roma respectivamente.

Finalidad: Trabajar con expresiones numéricas y con variables para no repetir cálculos. Dificultad Baja.

16. **[Gaussiana]** La función gaussiana es muy importante en Estadística. Es una función real de variable real que depende de dos parámetros μ y σ . El primero (μ) se conoce como *esperanza* o *media* y el segundo (σ) como *desviación típica* (*mean* y *standard deviation* en inglés). Su definición viene dada por la siguiente expresión:

$$\text{gaussiana}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-0.5 \left(\frac{x - \mu}{\sigma}\right)^2\right)}$$

En la gráfica de abajo pueden verse algunos ejemplos de esta función con distintos parámetros.



Realice un programa que lea los coeficientes reales μ y σ de una función gaussiana. A continuación el programa leerá un valor de abscisa x y se imprimirá el valor que toma la función en x

Para realizar las operaciones indicadas, debe utilizar las siguientes funciones de `cmath`:

- Para elevar el número e a un valor cualquiera, use la función `exp`. Por ejemplo, para calcular e^8 debería usar la siguiente expresión:

`exp(8)`

- Para calcular la raíz cuadrada, use `sqrt`.
- Para elevar un número a otro, utilice la función `pow` en la siguiente forma:

`pow(base, exponente)`

En nuestro caso, la base es $\frac{x - \mu}{\sigma}$ y el exponente 2.

Una vez resuelto el ejercicio usando la función `pow`, resuélvalo de otra forma en la que no necesite usar dicha función.

Compruebe que los resultados son correctos, usando cualquiera de las calculadoras disponibles en:

<http://danielsoper.com/statcalc3/calc.aspx?id=54>

<https://www.easycalculation.com/statistics/normal-pdf.php>

Ejemplo de entrada: 12 5 2.5 — Salida correcta: 0.01312316

Ejemplo de entrada: 0 1 0 — Salida correcta: 0.39894228

Finalidad: Trabajar con expresiones numéricas más complejas. Dificultad Media.

17. **[Intercambiar tres variables]** Se quiere generalizar el ejercicio 10 que intercambiaba el valor de dos variables al caso de tres variables. Construya un programa que declare las variables `x`, `y`, `z`, lea su valor desde teclado e intercambien entre sí sus valores de forma que el valor de `x` pasa a `y`, el de `y` pasa a `z` y el valor de `z` pasa a `x` (se pueden declarar variables auxiliares aunque se pide que se use el menor número posible).

Ejemplo de entrada: 7 4 5 — Salida correcta: 5 7 4

Finalidad: Mostrar la importancia en el orden de las asignaciones. Dificultad Media.

18. **[Desplaza entero dentro de un intervalo]** Queremos construir una expresión numérica que desplace un entero un número de posiciones, pero lo mantenga dentro de un intervalo. Por ejemplo, si el intervalo fijado es `[65, 90]`, el desplazamiento es de 3 unidades y el entero a desplazar es el 70, el resultado sería 73. Si el entero fuese el 89 y el desplazamiento 3, el resultado sería 92. Al no estar el 92 dentro del intervalo, se realiza un *ciclo* de forma que el 91 se transformaría en el 65 y el 92 en el 66.

Se pide construir un programa que lea dos enteros `minimo` y `maximo` que determinarán un intervalo `[minimo, maximo]`. Supondremos que el usuario introduce como `maximo` un valor mayor o igual que `minimo`. A continuación el programa leerá un valor entero `desplazamiento` (supondremos que el usuario introduce un valor entre 0 y `maximo - minimo`). Finalmente, el programa leerá un entero `a_desplazar` (supondremos que el usuario introduce un número entre `minimo` y `maximo`) le sumará el valor `desplazamiento` y lo convertirá en un entero dentro del intervalo `[minimo, maximo]` tal y como se ha descrito anteriormente.

Ejemplo de entrada: 65 90 3 70 — Salida correcta: 73

Ejemplo de entrada: 65 90 3 89 — Salida correcta: 66

Ejemplo de entrada: 65 90 3 90 — Salida correcta: 67

Finalidad: Trabajar con los operadores enteros. Dificultad Media.

19. **[Media y desviación]** Escriba un algoritmo para calcular la media aritmética muestral y la desviación estándar (o típica) muestral de las alturas de tres personas ($n=3$). Estos valores serán reales (de tipo `double`). La fórmula general para un valor arbitrario de n es:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i, \quad S = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{X})^2}$$

\bar{X} representa la media aritmética y S la desviación típica muestral. Para resolver este problema es necesario usar la función `sqrt` (raíz cuadrada) que se encuentra en `cmath`.

Estas medidas se utilizan mucho en Estadística para tener una idea de la distribución de datos. La media (mean en inglés) nos da una idea del valor central y la desviación típica (standard deviation) nos da una idea de la dispersión de éstos. Ejecutad el programa con varios valores y comprobad que el resultado es correcto utilizando una calculadora científica o cualquier calculadora online como por ejemplo la disponible en <http://www.disfrutalasmaticas.com/datos/desviacion-estandar-calculadora.html>

Ejemplo de entrada: 160 170 180

— Salida correcta: Media: 170 Desviación: 8.16497

Ejemplo de entrada: 170 170 170

— Salida correcta: Media: 170 Desviación: 0

Finalidad: Trabajar con expresiones numéricas y con variables para no repetir cálculos. Dificultad Baja.

20. **[Aproximación del valor de π]** Escriba un programa que muestre en pantalla el resultado de las siguientes expresiones numéricas que constituyen una aproximación al valor de π . La primera es del año 1800 antes de Cristo, la segunda (también de la misma era) es una aproximación introducida por los matemáticos mesopotámicos y la tercera es del siglo II (introducida por Claudio Ptolomeo)

$$\pi \approx \frac{256}{81} \quad \pi \approx 3 + \frac{1}{8} \quad \pi \approx \frac{377}{120}$$

El resultado debe ser 3.16049, 3.125 y 3.14167

Finalidad: Mezclar tipos enteros y reales. Dificultad Baja.

21. **[Índice de mayúscula]** Realice un programa que lea una mayúscula desde teclado sobre una variable de tipo `char`. A continuación, el programa imprimirá el 0 si se ha introducido el carácter A, el 1 si era la B, el 2 si era la C y así sucesivamente. Supondremos que el usuario introduce siempre un carácter mayúscula.

Ejemplo de entrada: C — Salida correcta: 2

Finalidad: Entender el tipo de dato `char`. Dificultad Baja.

22. [Descuento tarifa aérea mezclando tipos] Recupere la solución del ejercicio 6 [Tarifa aérea con descuento]. En vez de fijar los dos descuentos del 4% y del 2% dentro del código, se quiere leer dichos valores con `cin`. Utilice el tipo de dato `int` para leer las dos variables correspondientes a los dos descuentos y el tipo `double` para el precio del billete.

Ejemplo de entrada: 300.4 4 2 — Salida correcta: 288.384 294.392

Ejemplo de entrada: 500 4 2 — Salida correcta: 480 490

Ejemplo de entrada: 0 4 2 — Salida correcta: 0 0

Finalidad: Mezclar tipos de datos numéricos en una expresión. Dificultad Baja.

23. [Trunca decimales] Se quiere construir un programa que lea un número real r y un número entero n y trunque r a tantas cifras decimales como indique n . La única función que puede usar de `cmath` es `pow`.

Ejemplo de entrada: 1.2349 2 — Salida correcta: 1.23

Ejemplo de entrada: 1.237 2 — Salida correcta: 1.23

Ejemplo de entrada: 1.237 1 — Salida correcta: 1.2

Ejemplo de entrada: 1.237 0 — Salida correcta: 1

Finalidad: Practicar el truncamiento de un real asignado a un entero. Dificultad Baja.

24. [Pasar de mayúscula a minúscula] Construya un programa que lea un carácter (supondremos que el usuario introduce una mayúscula), lo pase a minúscula y lo imprima en pantalla. Hágalo sin usar las funciones `toupper` ni `tolower` declaradas en `cctype`. Para ello, debe considerarse la relación que hay en C++ entre los tipos enteros y caracteres.

Ejemplo de entrada: D — Salida correcta: d

Finalidad: Entender la equivalencia de C++ entre tipos enteros y de carácter. Dificultad Baja.

25. [Pasar de carácter a entero] Supongamos el siguiente código:

```
int entero;
char character;

cin >> character;
entero = character;
```

Supongamos que ejecutamos el código e introducimos el 7 desde teclado. El programa está leyendo una variable de tipo `char`. Por lo tanto, el símbolo 7 se interpreta como un carácter y es como si hiciésemos la siguiente asignación:

```
character = '7';
entero = character;
```

por lo que la variable `character` almacenará internamente el valor 55 (el orden en la tabla ASCII del carácter '7'). Lo mismo ocurre con la variable `entero`, que pasa a contener 55.

Sin embargo, queremos construir un programa para asignarle a la variable `entero` el número 7 asociado al dígito representado en la variable `character`, es decir, el 7 y no el 55. ¿Cómo se le ocurre hacerlo? El programa también imprimirá en pantalla el resultado.

Nota. La comilla simple para representar un literal de carácter es la que hay en el teclado del ordenador debajo de la interrogación ?. Esta comilla hay que ponerla en el código pero no en la entrada del carácter desde teclado.

Ejemplo de entrada: 7 (cin de un char) — Salida correcta: 7 (cout de un int)

Finalidad: Entender la equivalencia de C++ entre tipos enteros y de carácter. Dificultad Baja.

26. [Expresiones lógicas] Escriba una expresión lógica que sea verdadera si una variable de tipo carácter llamada `letra` es una letra minúscula y falso en otro caso.

Escriba una expresión lógica que sea verdadera si una variable de tipo entero llamada `edad` es menor de 18 o mayor de 65.

Escriba una expresión lógica que sea verdadera si una variable de tipo entero llamada `adivine` está entre 1 y 100.

Escriba una expresión lógica que sea verdadera si un año es bisiesto. Los años bisiestos son aquellos que o bien son divisibles por 4 pero no por 100, o bien son divisibles por 400.

Escriba un programa que lea las variables `letra`, `edad`, `adivine` y `anio`, calcule el valor de las expresiones lógicas anteriores e imprima el resultado. Debe almacenarse el resultado de las expresiones lógicas en variables de tipo `bool`.

Tenga en cuenta que cuando se imprime por pantalla (con `cout`) una expresión lógica que es `true`, se imprime 1. Si es `false`, se imprime un 0. En el tema 2 veremos la razón.

Ejemplo de entrada: a 30 0 2017 — Salida correcta: 1 0 0 0

Ejemplo de entrada: A 17 30 2000 — Salida correcta: 0 1 1 1

Finalidad: Empezar a trabajar con expresiones lógicas, muy usadas en el tema 2. Dificultad Baja.

27. [Elección tipo de dato] Indique qué tipo de dato usaría para representar:

- Edad de una persona
- Producto interior bruto de un país. Consultad:
[http://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_PIB_\(nominal\)](http://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_PIB_(nominal))
- La cualidad de que un número entero sea primo o no.

- Estado civil (casado, soltero, separado, viudo)
- Sexo de una persona (hombre o mujer exclusivamente)

Finalidad: Saber elegir adecuadamente un tipo de dato, atendiendo a la información que se quiere representar. Dificultad Media.

28. **[Precisión y desbordamiento]** Indique si se produce un problema de precisión o de desbordamiento en los siguientes ejemplos y diga cuál sería el resultado final de las operaciones.

Nota. Si se desea ver el contenido de una variable real con `cout`, es necesario que antes de hacerlo, se establezca el número de decimales que se quieren mostrar en pantalla. Para ello, basta ejecutar la sentencia `cout.precision(numero_digitos);` al inicio del programa. Hay que destacar que al trabajar con reales en coma flotante (`double`, `float`, etc) siempre debemos asumir que el valor almacenado es sólo una representación aproximada.

- a)

```
int chico, chico1, chico2;
chico1 = 1234567;
chico2 = 1234567;
chico = chico1 * chico2;
```
- b)

```
long grande;
int chico1, chico2;
chico1 = 1234567;
chico2 = 1234567;
grande = chico1 * chico2;
```
- c)

```
double resultado, real1, real2;
real1 = 123.1;
real2 = 124.2;
resultado = real1 * real2;
```
- d)

```
double resultado, real1, real2;
real1 = 123456789.1;
real2 = 123456789.2;
resultado = real1 * real2;
```
- e)

```
double real, otro_real;
real = 2e34;
otro_real = real + 1;
otro_real = otro_real - real;
```
- f)

```
double real, otro_real;
real = 1e+300;
otro_real = 1e+200;
otro_real = otro_real * real;
```

```
g)    float chico;  
       double grande;  
       grande = 2e+150;  
       chico = grande;
```

Finalidad: Entender los problemas de desbordamiento y precisión. Dificultad Media.

29. [Codificación de caracteres con algoritmo de rotación] Para intercambiar mensajes de forma privada, se utilizan distintos algoritmos que codifican/descodifican una cadena de caracteres. Uno de los más sencillos y que fue utilizado por Julio César durante la época del Imperio Romano es el de rotación.

Consiste en seleccionar una `clave` (un número entero), y desplazar las letras del alfabeto tantas posiciones como indica la clave.

Trabajaremos únicamente con las letras mayúsculas.

Se considera una representación *circular* del alfabeto, de tal forma que el carácter que sigue a 'Z' es 'A'. Por ejemplo, si `clave=4`, entonces la 'A' se reemplaza por la 'E' y la 'Z' por la 'D'. Utilizando `clave=0` la secuencia cifrada es igual a la original.

Construya un programa que lea un entero representando la clave y un carácter (supondremos que se introduce correctamente una letra mayúscula del alfabeto inglés). El programa codificará el carácter según la clave introducida y decodificará este segundo carácter para comprobar que se obtiene de nuevo el carácter original. El programa imprimirá por pantalla dichos caracteres.

Para resolver este ejercicio sólo se pueden usar las herramientas de programación vistas en el primer tema. Se recomienda que revise la solución del ejercicio 18 [Desplaza entero dentro de un intervalo] de esta Relación de Problemas.

Ejemplo de entrada: 4 A — Salida correcta: E A

Ejemplo de entrada: 4 Y — Salida correcta: C Y

Finalidad: Expresiones con caracteres y enteros. Dificultad Media.

Ejercicios complementarios

30. **[Subir sueldo]** Construya un programa para leer el valor de una variable `salario_base` de tipo `double`, la incremente un 2% e imprima el resultado en pantalla. Para realizar este cómputo, multiplique por 1.02 el valor original. Para resolver este ejercicio tiene varias alternativas:
- a) Directamente hacer el cómputo `1.02 * salario_base` dentro de la sentencia `cout`
 - b) Introducir una variable `salario_final`, asignarle la expresión anterior y mostrar su contenido en la sentencia `cout`
 - c) Modificar la variable original `salario_base` con el resultado de incrementarla un 2%.

Indique qué alternativa elige y justifíquela.

Ejemplo de entrada: 30 — Salida correcta: 30.6

Ejemplo de entrada: 0 — Salida correcta: 0

Finalidad: Ejemplo básico de asignación a una variable del resultado de una expresión. Dificultad Baja.

31. **[Dos subidas de sueldo]** Recupere la solución del ejercicio 30 **[Subir sueldo]**. Además de mostrar el salario con la subida del 2% se quiere mostrar el salario resultante de subirle otro 3% adicional. Esta segunda subida se realizará sobre el resultado de haber aplicado la primera subida. El programa debe mostrar los salarios resultantes (el resultante de la subida del 2% y el resultante de las dos subidas consecutivas del 2% y del 3%).

Ejemplo de entrada: 30 — Salida correcta: 30.6 31.518

Ejemplo de entrada: 0 — Salida correcta: 0 0

Finalidad: Trabajar con expresiones numéricas y con variables para no repetir cálculos. Dificultad Baja.

32. **[Consumo combustible]** Escriba un programa que calcule el consumo de gasolina. Pedirá la distancia recorrida (en kms), los litros de gasolina consumidos y los litros que quedan en el depósito. El programa debe informar el consumo en *km/litro*, los *litros/100 km* y cuántos kilómetros de autonomía le restan con ese nivel de consumo. Utilice nombres de variables significativos.

Finalidad: Resolver un problema real sencillo, usando varias sentencias. Dificultad Baja.

33. **[Métricas atletismo]** En atletismo se expresa la rapidez de un atleta en términos de ritmo (*minutos y segundos por kilómetro*) más que en unidades de velocidad (*kilómetros por hora*).

Escriba dos programas para convertir entre estas dos medidas:

- a) El primero leerá el ritmo (minutos y segundos, por separado) y mostrará la velocidad (kilómetros por hora).
- b) El segundo leerá la velocidad (kilómetros por hora) y mostrará el ritmo (minutos y segundos).

Finalidad: Trabajar con expresiones numéricas y con variables de diferentes tipos. Dificultad Baja.

34. **[Reparto de ganancias]** Las ganancias de un determinado producto se reparten entre el diseñador y los tres fabricantes del mismo. Diseñe un programa que pida la ganancia total de la empresa (los ingresos realizados con la venta del producto) y diga cuánto cobran cada uno de ellos, sabiendo que el diseñador cobra el doble que cada uno de los fabricantes. El dato de entrada será la ganancia total a repartir. Utilice el tipo `double` para todas las variables.
Importante: No repita cálculos ya realizados.

Finalidad: Entender la importancia de no repetir cálculos para evitar errores de programación. Dificultad Baja.

35. **[Reparto de ganancias mezclando tipos]** Realice el ejercicio del reparto de la ganancia de un producto, pero cambiando el tipo de dato de la ganancia total a `int` (el resto de variables siguen siendo `double`)

Finalidad: Trabajar con expresiones numéricas que involucren distintos tipos de datos. Dificultad Baja.

36. Construya un programa que lea desde teclado tres variables correspondientes a un número de horas, minutos y segundos, respectivamente. A continuación, el programa debe calcular las horas, minutos y segundos dentro de su rango correspondiente. Por ejemplo, dadas 312 horas, 119 minutos y 1291 segundos, debería dar como resultado 13 días, 2 horas, 20 minutos y 31 segundos. El programa no calculará meses, años, etc. sino que se quedará en los días.

Como consejo, utilice el operador `/` que cuando trabaja sobre datos enteros, obtiene la división entera. Para calcular el resto de la división entera, use el operador `%`.

Ejemplo de entrada: 312 119 1291 — Salida correcta: 13 2 20 31

Finalidad: Trabajar con expresiones numéricas y con variables para no repetir cálculos. Dificultad Media.

37. **[Pinta dígitos]** Escriba un programa que lea un valor entero e imprima en pantalla cada uno de sus dígitos separados por dos espacios en blanco. Supondremos que el usuario introduce siempre un entero de cinco dígitos, como por ejemplo 35197. En este caso, la salida sería:

3 5 1 9 7

Finalidad: Ejemplo de asignación acumulada.

Dificultad Media.

38. **[Población]** Los estudios poblacionales utilizan los conceptos de tasa de natalidad, mortalidad, etc. Al igual que un porcentaje representa una razón del total por cada cien (tanto por ciento), la tasa es una razón del total por cada mil (tanto por mil). Así pues una tasa de natalidad de 32, por ejemplo, significa que hay 32 nacimientos por cada 1000 habitantes.

Escriba un programa que calcule la estimación de la población de un territorio después de tres años. Para ello, el programa debe leer la población inicial, la tasa de natalidad, la de mortalidad y la tasa de migración. Ésta última es la diferencia entre los que se van y los que vienen, por lo que puede ser o bien positiva o bien negativa.

Suponga que todos los datos son enteros.

Tenga en cuenta que una vez calculada la población que habrá el siguiente año, las tasas se vuelven a aplicar sobre la población así obtenida, y así sucesivamente, tantos años como estemos interesados.

Ejemplo de entrada: 1375570814 32 12 7 — Salida correcta: 1490027497

Finalidad: Ejemplo básico de asignación acumulada y uso de tipos numéricos distintos.
Dificultad Baja.

39. Razone sobre la falsedad o no de las siguientes afirmaciones:

- a) 'c' es una expresión de caracteres.
- b) 4 < 3 es una expresión numérica.
- c) (4 + 3) < 5 es una expresión numérica.
- d) cout << a; da como salida la escritura en pantalla de una a.
- e) ¿Qué realiza cin >> cte, siendo cte una constante entera?

Finalidad: Distinguir entre expresiones de distinto tipo de dato. Dificultad Baja.

40. Dadas las variables count = 0, limit = 10, x = 2, y = 7, calcule el valor de las siguientes expresiones lógicas

```
count == 0 && limit < 20
limit > 20 || count < 5
!(count == 12)
count == 1 && x < y
!( (count < 10 || x < y) && count >= 0 )
(count > 5 && y == 7) || (count <= 0 && limit == 5*x)
!( limit != 10 && x > y )
```

41. El precio final de un automóvil para un comprador es la suma total del costo del vehículo, del porcentaje de ganancia de dicho vendedor y del I.V.A. Diseñe un algoritmo para obtener el precio final de un automóvil sabiendo que el porcentaje de ganancia de este vendedor es del 20 % y el I.V.A. aplicable es del 16 %.

Dificultad Baja.

42. Cree un programa que lea un valor de temperatura expresada en grados Celsius y la transforme en grados Fahrenheit. Para ello, debe considerar la fórmula siguiente:

$$\text{Grados Fahrenheit} = (\text{Grados Celsius} * 180 / 100) + 32$$

Buscad en Internet el por qué de dicha fórmula.

Dificultad Baja.

43. Declare las variables necesarias y traduzca las siguientes fórmulas a expresiones válidas del lenguaje C++.

a) $\frac{1 + \frac{x^2}{y}}{\frac{x^3}{1+y}}$

b) $\frac{1 + \frac{1}{3} \sin h - \frac{1}{7} \cos h}{2h}$

c) $\sqrt{1 + \left(\frac{e^x}{x^2}\right)^2}$

Algunas funciones de `cmath`

$\text{sen}(x) \rightarrow \sin(x)$

$\text{cos}(x) \rightarrow \cos(x)$

$x^y \rightarrow \text{pow}(x, y)$

$\ln(x) \rightarrow \log(x)$

$e^x \rightarrow \exp(x)$

Dificultad Baja.

44. Dos locomotoras parten de puntos distintos avanzando en dirección contraria sobre la misma vía. Se pide redactar un programa para conocer las distancias que habrán recorrido ambas locomotoras antes de que choquen teniendo en cuenta que la primera locomotora viaja a una velocidad constante V_1 , que la segunda viaja a una velocidad constante V_2 , la fórmula que relaciona velocidad, espacio y tiempo ($s = v t$) y que el momento en que se producirá el choque viene dado por la fórmula

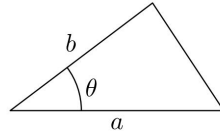
$$t = \frac{D}{V_1 + V_2}$$

dónde D es la distancia que separa los puntos iniciales de partida. Los datos de entrada al programa serán D , V_1 y V_2 .

Dificultad Baja.

45. El área A de un triángulo se puede calcular a partir del valor de dos de sus lados, a y b , y del ángulo θ que éstos forman entre sí con la fórmula $A = \frac{1}{2}ab \sin(\theta)$. Construya

un programa que pida al usuario el valor de los dos lados (en centímetros), el ángulo que éstos forman (en grados), y muestre el valor del área.



Tened en cuenta que el argumento de la función `sin` va en radianes por lo que habrá que transformar los grados del ángulo en radianes (recordad que 360 grados son 2π radianes).

Dificultad Baja.

46. Los compiladores utilizan siempre el mismo número de bits para representar un tipo de dato entero (este número puede variar de un compilador a otro). Por ejemplo, 32 bits para un `int`. Pero, realmente, no se necesitan 32 bits para representar el 6, por ejemplo, ya que bastarían 3 bits:

$$6 = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 \equiv 110$$

Se pide crear un programa que lea un entero n , y calcule el mínimo número de dígitos que se necesitan para su representación. Para simplificar los cálculos, suponed que sólo queremos representar valores enteros positivos (incluido el cero). Consejo: se necesitará usar el logaritmo en base 2 y obtener la parte entera de un real (se obtiene tras el truncamiento que se produce al asignar un real a un entero)

Dificultad Media.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Ejercicios sobre condicionales

1. [Mismo signo -multiplicando-] Construya un programa que lea dos datos a y b de tipo `int` y nos diga si tienen el mismo signo. Se considera que el cero no tiene signo por lo que cualquier número (incluido el cero) tiene un signo distinto del cero.

Para ello, basta comprobar si la multiplicación de a y b es positiva o no.

Ejemplo de entrada: 0 0 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 0 3 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 3 0 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 3 4 — Salida correcta: Tienen el mismo signo

Ejemplo de entrada: -3 -4 — Salida correcta: Tienen el mismo signo

Ejemplo de entrada: 3 -4 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: -3 4 — Salida correcta: No tienen el mismo signo

Finalidad: Trabajar con estructuras condicionales simples. Dificultad Baja.

2. [Codificación de caracteres con algoritmo de rotación. Condicional simple] Reutilice la solución del ejercicio 29 [Codificación de caracteres con algoritmo de rotación] y cambie la implementación para realizar la codificación y decodificación del carácter utilizando un condicional simple en vez de una expresión tan compleja como la que se utilizó en dicho ejercicio. En definitiva, en este ejercicio no puede usar el operador módulo %, aunque sí puede usar condicionales simples.

Ejemplo de entrada: 4 A — Salida correcta: E A

Ejemplo de entrada: 4 Y — Salida correcta: C Y

Finalidad: Plantear una estructura condicional simple. Actualizar una variable según una condición. Dificultad Baja.

3. [Tarifa aérea] Retome la solución del ejercicio 5 [Tarifa aérea según km] de la Relación de Problemas I. La forma de calcular la tarifa final del billete cambia ahora de la forma siguiente: la tarifa base sigue siendo de 150 euros, la misma para todos los destinos. Ahora bien, si el destino está a menos de 300 kilómetros, el precio final es la tarifa base. Para destinos a más de 300 Km, se suman 10 céntimos por cada kilómetro de distancia al destino (a partir del Km 300).

Cree un programa para que lea el número de kilómetros al destino y calcule el precio final del billete.

Ejemplo de entrada: 120 — Salida correcta: 150

Ejemplo de entrada: 300 — Salida correcta: 150

Ejemplo de entrada: 301 — Salida correcta: 150.1

Ejemplo de entrada: 452 — Salida correcta: 165.2

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Finalidad: Plantear una estructura condicional simple. Actualizar una variable según una condición. Dificultad Baja.

4. [Operadores lógicos] En este ejercicio no hace falta construir ningún programa. Debe crear un fichero de texto (con extensión `cpp` o `txt`) explicando qué problemas observa en los siguientes condicionales:

```
a)   char tipo_radar;
      cin >> tipo_radar;

      if (tipo_radar == 'F' && tipo_radar == 'f')
          .....

b)   double velocidad;
      cin >> velocidad;

      if (velocidad > 100 && velocidad < 70)
          cout << "\nVelocidad fuera del rango";

c)   double velocidad;
      cin >> velocidad;

      if (velocidad > 100 || velocidad > 110)
          cout << "Velocidad excesiva";
```

Finalidad: Entender cómo funcionan los operadores lógicos. Dificultad Baja.

5. [Coordenadas geográficas (distancia)] Se quiere calcular la distancia entre dos puntos de la Tierra. Cada punto vendrá determinado por sus coordenadas geográficas, dadas por tres datos reales: su longitud, latitud y altura. Lo vamos a hacer de dos formas:

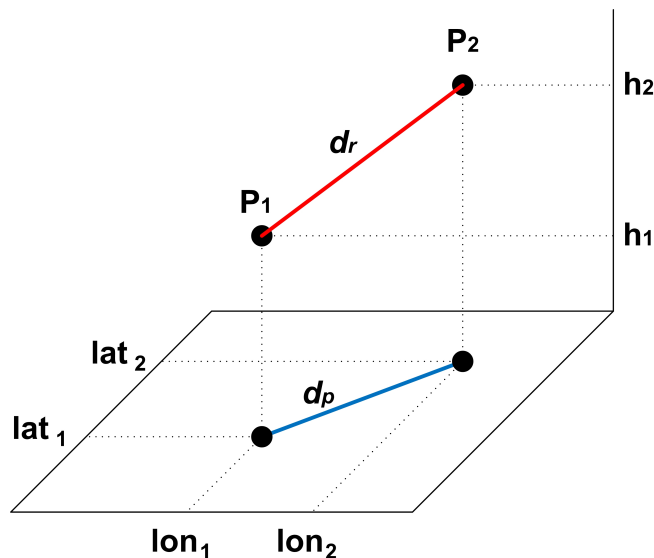
- a) Sin tener en cuenta la altura. La distancia *sobre plano* entre dos puntos se calcula como la longitud del segmento d_p de la figura y se puede calcular con la llamada *fórmula del Haversine*:

- I) Se calcula $\Delta_{lon} = lon_2 - lon_1$ y $\Delta_{lat} = lat_2 - lat_1$
- II) Se calcula $a = \sin^2\left(\frac{1}{2} \Delta_{lat}\right) + \cos(lat_1) \cos(lat_2) \sin^2\left(\frac{1}{2} \Delta_{lon}\right)$
- III) Se calcula $c = 2 \arcsin(\min(1, \sqrt{a}))$ donde $\min(1, \sqrt{a})$ representa el mínimo entre 1 y \sqrt{a} .
- IV) Finalmente, la distancia será $d_p = R c$, donde $R = 6372797.560856$ metros es la longitud media del radio terrestre.

Tenga en cuenta lo siguiente:

- I) Los datos de latitud y longitud en las fórmulas, vienen expresados en radianes.

- II) La función arco-seno (\arcsin) viene ya implementada en la biblioteca `cmath` con el nombre `asin`
 - III) `mín` representa el mínimo y debe calcularlo en el programa usando una estructura condicional simple.
- b) Teniendo en cuenta la altura se obtiene la distancia *real* entre los dos puntos. Dicha distancia es la longitud del segmento d_r en la figura. Para calcular dicho valor, basta observar que se puede formar un triángulo rectángulo a partir de los segmentos d_p , d_r y la diferencia de las alturas (representadas por h_1 y h_2 en el gráfico).



Construya un programa que lea las coordenadas del primer punto (latitud, longitud y altura), las del segundo e imprima en pantalla la distancia sobre plano y la distancia real. Supondremos que las coordenadas de los puntos se introducen en grados, por lo que habrá que pasarlas a radianes para poder aplicar las fórmulas anteriores.

Para resolver este problema puede recuperar la solución del ejercicio 15 [Coordenadas geográficas] de la Relación de Problemas I.

Ejemplo de entrada: 37.18817 -3.60667 738 41.89193 12.51133 52

— Salida correcta: 1475367.200551 1475367.36003946

Los datos anteriores corresponden a la latitud, longitud y altura de Granada y Roma respectivamente.

Finalidad: Plantear una estructura condicional sencilla. Dificultad Baja.

6. [Convertir mayúscula en minúscula] Se quiere leer un carácter `letra_original` desde teclado, y comprobar con una estructura condicional si es una letra mayúscula. En dicho caso, hay que calcular la minúscula correspondiente almacenando el resultado en una variable llamada `letra_convertida`. En el caso

RELACIÓN DE PROBLEMAS II. Estructuras de Control

de que no sea una mayúscula, le asignaremos a `letra_convertida` el valor que tenga `letra_original`. Finalmente, imprimiremos en pantalla el valor de `letra_convertida`. No pueden usarse las funciones `tolower` ni `toupper` de la biblioteca `cctype`.

Ejemplo de entrada: D — Salida correcta: d

Ejemplo de entrada: d — Salida correcta: d

Ejemplo de entrada: ! — Salida correcta: !

Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. Dificultad Baja.

7. **[Se dividen]** Realice un programa que lea dos valores enteros desde teclado y diga si cualquiera de ellos divide o no (de forma entera) al otro. En este problema no hace falta decir quién divide a quién. Supondremos que los valores leídos desde teclado son ambos distintos de cero.

Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. Dificultad Baja.

8. **[Tarifa aérea con descuentos]** Recupere la solución del ejercicio 3 **[Tarifa aérea]** de esta Relación de Problemas. Una vez se ha obtenido el precio final del billete se quiere aplicar un descuento en función del número de kilómetros del trayecto y del número de puntos de la tarjeta de fidelización del cliente que ha comprado el billete. En concreto:

a) Si el trayecto es mayor de 700 km, se aplica un descuento del 2 %

b) Si el número de puntos es mayor de 100, se aplica un descuento del 3 % y si es mayor de 200, se aplica un descuento del 4 %.

Los descuentos por longitud del trayecto y el correspondiente al número de puntos, son acumulables. En cualquier caso, ambos se aplican sobre el precio del billete (es decir, un descuento no se aplica sobre el resultado de haber aplicado previamente el otro descuento).

Construya un programa que lea el número de kilómetros del trayecto y el número de puntos del cliente e imprima en pantalla el precio final del billete con los descuentos aplicados, en su caso.

Ejemplo de entrada: 200 90 — Salida correcta: 150

Ejemplo de entrada: 200 120 — Salida correcta: 145.5

Ejemplo de entrada: 200 250 — Salida correcta: 144

Ejemplo de entrada: 650 90 — Salida correcta: 185

Ejemplo de entrada: 650 120 — Salida correcta: 179.45

Ejemplo de entrada: 800 90 — Salida correcta: 196

Ejemplo de entrada: 800 120 — Salida correcta: 190

Ejemplo de entrada: 800 250 — Salida correcta: 188

Finalidad: Planteamiento de una estructura condicional secuencial. Dificultad Baja.

9. **[Pasar de mayúscula a minúscula y viceversa]** Queremos modificar el ejercicio 6 para leer un carácter `letra_original` desde teclado y hacer lo siguiente:

- Si es una letra mayúscula, almacenaremos en la variable `letra_convertida` la correspondiente letra minúscula.
- Si es una letra minúscula, almacenaremos en la variable `letra_convertida` la correspondiente letra mayúscula.
- Si es un carácter no alfabético, almacenaremos el mismo carácter en la variable `letra_convertida`

El programa debe imprimir en pantalla el valor de `letra_convertida` e indicar si la letra introducida era una minúscula, mayúscula o no era una carácter alfabético. No pueden usarse las funciones `tolower` ni `toupper` de la biblioteca `cctype`.

Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

10. **[Comparación de dos instantes]** Construya un programa que lea dos instantes de tiempo (según se define en el ejercicio 8 **[Segundos entre dos instantes]** de la Relación de Problemas I) y nos diga si el primero es anterior al segundo. Hágalo de dos formas:

- a) Calculando los segundos que hay de diferencia entre ambos instantes, tal y como se hizo en el ejercicio 8 **[Segundos entre dos instantes]** de la Relación de Problemas I
- b) Sin calcular el número de segundos. Por lo tanto, tendrá que comparar apropiadamente los valores de la hora, minuto y segundo de cada instante.

Ejemplo de entrada: 9 12 9 10 34 55

— Salida correcta: El primero es anterior

Ejemplo de entrada: 10 34 55 9 12 9

— Salida correcta: El primero no es anterior

Ejemplo de entrada: 10 34 55 10 34 55

— Salida correcta: El primero no es anterior

Finalidad: Planteamiento de una estructura condicional anidada. Dificultad Baja.

11. **[Multa autovía]** La Dirección General de Tráfico publica en su página web las sanciones a aplicar por infracción de velocidad. Puede consultarse la tabla en https://sede.dgt.gob.es/Galerias/tramites-y-multas/alguna-multa/consulta-de-sanciones-por-exceso-velocidad/cuadro_velocidad.pdf

En este ejercicio queremos determinar la sanción a aplicar en una autovía, cuyo límite de velocidad es 120. En la siguiente tabla se muestra la velocidad del vehículo y la sanción correspondiente (número de puntos del carnet de conducir que se restan y la multa en euros)

121 -> 0, 100
151 -> 2, 300
171 -> 4, 400
181 -> 6, 500
191 -> 6, 600

Escriba un programa que lea la velocidad del vehículo e imprima en pantalla la sanción correspondiente (número de puntos a detraer y multa en euros)

Finalidad: Planteamiento de una estructura condicional anidada. Dificultad Baja.

12. [Velocidad imputada] En el ejercicio 11 [Multa autovía] de esta Relación de Problemas, se usaba la velocidad del vehículo para establecer la sanción correspondiente en una autovía. Dicha velocidad es captada por unos aparatos electrónicos, los conocidos radares o (*cinemómetros*), que pueden ser fijos o móviles. Estos aparatos presentan un margen de error que hay que aplicar a la velocidad captada por el radar y vienen especificados en la orden ITC/3123/2010, de 26 de noviembre
https://www.boe.es/diario_boe/txt.php?id=BOE-A-2010-18556

a) En el caso de un radar fijo:

- I) Si la velocidad captada por el radar es menor o igual que 100 km/h, el margen de error es de ± 5 km/h.
- II) En caso contrario, el margen de error es de un 5%

b) En el caso de un radar móvil:

- I) Si la velocidad captada por el radar es menor o igual que 100 km/h, el margen de error es de 7 km/h.
- II) En caso contrario, el margen de error es de un 7%

Los márgenes de error se aplican sobre la velocidad captada y da como resultado la velocidad imputada. Por ejemplo, si la velocidad captada es de 95 km/h, la velocidad imputada sería de $95 - 5 = 90$ km/h en el caso de un radar fijo y de $95 - 7 = 88$ km/h en el caso de un radar móvil.

Si la velocidad captada es, por ejemplo, de 104 km/h, la velocidad imputada sería de $104 - 5\% \text{ de } 104 = 104 - 5.2 = 98.8$ km/h en el caso de un radar fijo y de $104 - 7\% \text{ de } 104 = 104 - 7.28 = 96.72$ km/h en el caso de un radar móvil.

Construya un programa que lea desde teclado un carácter que indique el tipo de radar ('F' para fijo y cualquier otra letra para móvil), la velocidad captada, e imprima la velocidad imputada.

Ejemplo de entrada: F 95 — Salida correcta: 90

Ejemplo de entrada: M 95 — Salida correcta: 88

Ejemplo de entrada: F 104 — Salida correcta: 98.8

Ejemplo de entrada: M 104 — Salida correcta: 96.72

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Finalidad: Plantear una estructura condicional anidada y actualizar variables en función de ciertas condiciones. Dificultad Baja.

13. [Mismo signo separando E/S y C] Recupere la solución del ejercicio 1 [Mismo signo -multiplicando-] de esta Relación de Problemas. Resuelva el mismo problema de ver si dos enteros tienen el mismo signo pero sin mezclar Entradas/Salidas y Cálculos. Para ello, use una variable lógica `mismo_signo` y asígnele el valor correspondiente de la expresión lógica.

Finalidad: Separación de E/S y C. Dificultad Baja.

14. [Mismo signo con condicionales] Recupere la solución del ejercicio 13 [Mismo signo separando E/S y C] de esta Relación de Problemas. Resuelva el mismo problema de ver si dos enteros tienen el mismo signo sin utilizar la operación de multiplicación. Para ello, se pide que implemente todas y cada una de las siguientes posibilidades:

- Enumerando todas las posibles situaciones, es decir, los dos positivos, los dos negativos, el primero positivo y el segundo negativo o el primero negativo y el segundo positivo. En esta versión, sí puede (y debe) usar los operadores lógicos `&&`, `||`, `!`.
- Utilizar una estructura condicional anidada en la que sólo se permiten expresiones lógicas simples, es decir, no se pueden usar los operadores lógicos `&&`, `||`, `!`.
- Sin utilizar estructuras condicionales. Debe construir una única expresión lógica compuesta y asignarle su valor a una variable lógica. En esta versión, sí puede (y debe) usar los operadores lógicos.

Ejemplo de entrada: 0 0 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 0 3 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 3 0 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: 3 4 — Salida correcta: Tienen el mismo signo

Ejemplo de entrada: -3 -4 — Salida correcta: Tienen el mismo signo

Ejemplo de entrada: 3 -4 — Salida correcta: No tienen el mismo signo

Ejemplo de entrada: -3 4 — Salida correcta: No tienen el mismo signo

Finalidad: Trabajar con estructuras condicionales anidadas y expresiones lógicas compuestas. Dificultad Media.

15. [Mayúscula a minúscula y viceversa usando un enumerado] Modifique la solución al ejercicio 9 [Pasar de mayúscula a minúscula y viceversa] para que, dependiendo de cómo era la letra introducida, imprima en pantalla alguno de los siguientes mensajes:

- La letra era una mayúscula. Una vez convertida es ...
- La letra era una minúscula. Una vez convertida es ...
- El carácter no era una letra.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Hágalo separando entradas y salidas de los cálculos. Para ello, utilice una variable de tipo enumerado que represente las opciones de que un carácter sea una mayúscula, una minúscula o un carácter no alfabético.

Finalidad: Separación de E/S y C. Usar el tipo enumerado para detectar cuándo se produce una situación determinada. Dificultad Media.

Ejercicios sobre bucles

16. [Divisores de un entero] Realice un programa que lea desde teclado un entero `tope` e imprima en pantalla todos sus divisores propios⁴. Para obtener los divisores, basta recorrer todos los enteros menores que el valor introducido y comprobar si lo dividen. A continuación, mejorar el ejercicio obligando al usuario a introducir un entero positivo, usando un filtro con un bucle post test (`do while`).

Finalidad: Plantear un ejemplo sencillo de bucle y de filtro de entrada de datos. Dificultad Baja.

17. [Mínimo de varios valores] Realice un programa que lea enteros desde teclado y calcule cuántos se han introducido y cual es el mínimo de dichos valores (pueden ser positivos o negativos). Se dejará de leer datos cuando el usuario introduzca el valor 0. Realice la lectura de los enteros dentro de un bucle sobre una única variable llamada `dato`. Es importante controlar los casos extremos, como por ejemplo, que el primer valor leído fuese ya el terminador de entrada (en este caso, el cero).

Ejemplo de entrada: 0 — Salida correcta: 0

Ejemplo de entrada: 1 3 -1 2 0 — Salida correcta: -1

Ejemplo de entrada: 1 3 1 2 0 — Salida correcta: 1

Finalidad: Destacar la importancia de las inicializaciones antes de entrar al bucle. Ejemplo de lectura anticipada. Dificultad Baja.

18. [Tarifa aérea con filtro de entrada de datos] Recupere la solución del ejercicio 8 [Tarifa aérea con descuentos] de esta Relación de Problemas. Modifíquelo para obligar al usuario a introducir valores correctos. Por lo tanto, debe usar un filtro de entrada de datos para que el número de kilómetros sea positivo y el número de puntos sea un valor entre 0 y 400 (máximo número posible de puntos).

Ejemplo de entrada: -2 -5 200 1300 -450 250 — Salida correcta: 144

Ejemplo de entrada: 650 1300 -450 90 — Salida correcta: 185

Finalidad: Uso de bucles en filtros de entrada de datos. Dificultad Baja.

19. [Valores de la Gaussiana] Recupere la solución del ejercicio 16 (función gaussiana) de la relación de problemas I. Se pide construir un programa para imprimir el resultado de aplicar dicha función a varios valores de abscisas.

En primer lugar, se leerán los parámetros que definen la función, es decir, la esperanza y la desviación. La esperanza puede ser cualquier valor, pero para leer el valor de desviación debe utilizar un bucle y obligar a que sea mayor o igual que cero.

A continuación el programa pedirá un valor mínimo, un valor máximo y un incremento. El valor máximo ha de leerse con un filtro de entrada obligando a que

⁴Los divisores propios de un entero n son los divisores de n distintos de 1 y el mismo n

sea mayor que mínimo. El programa mostrará el valor de la función gaussiana en todos los valores de x (la abscisa) entre mínimo y máximo a saltos de incremento, es decir, mínimo, mínimo + incremento, mínimo + 2*incremento, ..., hasta llegar, como mucho, a máximo.

```
12      <- Media
5       <- Desviación
2       <- Mínimo
3       <- Máximo
0.5     <- Incremento
```

Ejemplo de entrada: 12 5 2 3 0.5

— Salida correcta: 0.0107982 0.0131232 0.01579

Ejemplo de entrada: 12 -5 5 2 3 0.5

— Salida correcta: 0.0107982 0.0131232 0.01579

Ejemplo de entrada: 12 -5 5 2 1 0 3 0.5

— Salida correcta: 0.0107982 0.0131232 0.01579

Finalidad: Ejemplo básico de bucle. Dificultad Baja.

20. **[Número desgarrable]** Un número entero n se dice que es *desgarrable* si al dividirlo en dos partes cualesquiera izda y dcha, el cuadrado de la suma de ambas partes es igual a n . Por ejemplo, 88209 es desgarrable ya que $(88 + 209)^2 = 88209$; 81 también lo es ya que $81 = (8 + 1)^2$. Cree un programa que lea un entero n e indique si es o no desgarrable.

Finalidad: Ejercitar los bucles con condiciones de salida complejas. Dificultad Media.

21. **[Factorial y Potencia]** Calcule mediante un programa en C++ la función potencia x^n , y la función factorial $n!$ con n un valor entero y x un valor real. No pueden usarse las funciones definidas en `cmath`, por lo que tendrá que implementar los cálculos con los bucles necesarios.

El factorial de un entero n se define de la forma siguiente:

$$0! = 1$$

$$n! = 1 \times 2 \times 3 \times \cdots \times n, \quad \forall n \geq 1$$

Escriba un programa de prueba que lea un número entero n obligando a que esté en el intervalo $[1, 20]$. A continuación lea un valor real x y calcule e imprima en pantalla el factorial de n y la potencia de x elevado a n .

Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

22. **[Velocidad imputada -lectura en bucle-]** Recupere la solución del ejercicio 12 **[Velocidad imputada]** de esta Relación de Problemas. Modifique el programa principal para que vaya leyendo los siguientes datos dentro de un bucle: en primer lugar, se leerá

la matrícula del vehículo (en un dato de tipo `string`), a continuación el tipo de radar (dato de tipo `char`) y finalmente la velocidad del vehículo (en un dato de tipo `double`).

Supondremos que el carácter 'F' está asociado al radar fijo y el carácter 'M' al radar móvil. Supondremos que el usuario introduce un carácter correcto.

El programa calculará las velocidades imputadas según se indicó en el ejercicio 12 [Velocidad imputada] e imprimirá en pantalla la matrícula de aquel vehículo que tenga máxima velocidad imputada (también imprimirá dicha velocidad)

La lectura de los datos terminará cuando la matrícula del vehículo sea la cadena "#"

Ejemplo de entrada: 4312BHG F 95 8342DFW F 104 4598IJG M 95 #
— Salida correcta: 8342DFW 98.8

Finalidad: Trabajar con la lectura anticipada. Dificultad Baja.

23. [Aproximación de π por Gregory-Leibniz] En el siglo XVII el matemático alemán Gottfried Leibniz y el matemático escocés James Gregory introdujeron una forma de calcular π a través de una serie, es decir, de una suma de términos:

$$\frac{\pi}{4} = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 1 - \frac{1}{2*1+1} + \frac{1}{2*2+1} - \frac{1}{2*3+1} + \dots$$

Esta es una serie infinita, pues realiza la suma de infinitos términos. Como en Programación no podemos realizar un número infinito de operaciones, habrá que parar en un índice dado, llamémosle `tope`, obteniendo por tanto una aproximación al valor de π . Usaremos el símbolo \approx para denotar esta aproximación:

$$\begin{aligned} \frac{\pi}{4} &\approx \sum_{i=0}^{i=\text{tope}} \frac{(-1)^i}{2i+1} = 1 - \frac{1}{2*1+1} + \frac{1}{2*2+1} - \frac{1}{2*3+1} + \dots + \frac{(-1)^{\text{tope}}}{2*\text{tope}+1} \\ &= 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^{\text{tope}}}{2*\text{tope}+1} \end{aligned}$$

Construya un programa que lea el valor `tope` obligando a que esté entre 1 y cien mil, calcule la aproximación de π mediante la anterior serie e imprima el resultado en pantalla.

Resuelva este problema de dos formas distintas:

- Usando la función `pow` (potencia) de `cmath` para implementar $(-1)^i$
- Sin usar la función `pow`. Observe que el valor de $(-1)^i$ es 1 para los valores pares de i y -1 para los impares

Recuerde que, para visualizar 15 cifras decimales, por ejemplo, debe incluir la sentencia `cout.precision(15)`; antes de realizar la salida en pantalla.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Ejemplo de entrada: 1000 — Salida correcta: 3.14259165433954

Ejemplo de entrada: 100000 — Salida correcta: 3.14160265348972

Ampliación:

Es fácil demostrar que el valor absoluto de cada sumando se va haciendo cada vez más pequeño, por lo que cuantos más sumandos incluyamos, mejor será la aproximación. En cualquier caso, la serie *converge* al valor correcto de forma muy lenta. Habría que incluir cinco mil millones de términos para obtener 10 cifras decimales exactas en el cómputo de π .

Finalidad: Bucles simples. Acumulación de sumas. Dificultad Baja.

24. [Aproximación de π por Wallis] Otra aproximación de π introducida en el siglo XVII por el matemático inglés John Wallis viene dada por:

$$\frac{\pi}{2} \approx \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \dots$$

Construya un programa que lea el valor `tope` obligando a que esté entre 1 y cien mil, calcule la aproximación de π mediante la anterior fórmula (multiplicando un total de `tope` fracciones) e imprima el resultado en pantalla.

Debe resolver este problema de dos formas distintas, a saber:

- Observe que el numerador y el denominador varían de forma alternativa (aunque ambos de la misma forma, a saltos de 2). Cuando a uno le toca cambiar, el otro permanece igual. Este comportamiento se puede controlar con una única variable de tipo de dato `bool`.
- Otra forma de implementar los cambios en el numerador y denominador es observando que en cada iteración, el numerador es el denominador de la iteración anterior más 1 y el denominador es el numerador de la iteración anterior más 1.

Ejemplo de entrada: 1000 — Salida correcta: 3.1400238186006

Ejemplo de entrada: 100000 — Salida correcta: 3.14157694582286

Finalidad: Bucles simples. Acumulación de productos. Usar un `bool` fijado en la iteración anterior. Dificultad Media.

25. [Aproximación de π por Madhava sin usar `pow`] En el siglo XIV el matemático indio Madhava of Sangamagrama calculó el arco tangente a través de un desarrollo de Taylor (este tipo de desarrollos se ve en la asignatura de Cálculo)

$$\arctan(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{2i+1}}{2i+1}$$

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Usando como x el valor 1, obtenemos la serie de Leibniz vista en el ejercicio 23:

$$\frac{\pi}{4} = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1}$$

Usando como x el valor $\frac{1}{\sqrt{3}}$, obtenemos:

$$\frac{\pi}{6} = \sum_{i=0}^{\infty} \frac{(-1)^i \left(\frac{1}{\sqrt{3}}\right)^{2i+1}}{2i+1}$$

Por lo tanto, podemos usar la siguiente aproximación:

$$\frac{\pi}{6} \approx \sum_{i=0}^{\text{tope}} \frac{(-1)^i \left(\frac{1}{\sqrt{3}}\right)^{2i+1}}{2i+1}$$

Construya un programa que lea el valor `tope` obligando a que esté entre 1 y cien mil, calcule la aproximación de π mediante la anterior serie e imprima el resultado en pantalla.

Importante: En la implementación de esta solución NO puede usar la función `pow` ni ningún condicional `if`.

Ejemplo de entrada: 1000 — Salida correcta: 3.14159265358979

Ejemplo de entrada: 100000 — Salida correcta: 3.14159265358979

Finalidad: Bucles simples. Aprovechar cálculos de la iteración anterior. Dificultad Media.

26. **[RLE]** El método RLE (Run Length Encoding) codifica una secuencia de datos formada por series de valores idénticos consecutivos como una secuencia de parejas de números (valor de la secuencia y número de veces que se repite). Esta codificación es un mecanismo de compresión de datos (zip) sin pérdidas. Se aplica, por ejemplo, para comprimir los ficheros de imágenes en las que hay zonas con los mismos datos (fondo blanco, por ejemplo). Realice un programa que lea una secuencia de números naturales terminada con un número negativo y la codifique mediante el método RLE.

Entrada:	1 1 1 2 2 2 2 2 3 3 3 3 3 5 -1
	(tres veces 1, cinco veces 2, seis veces 3, una vez 5)
Salida:	3 1 5 2 6 3 1 5

Finalidad: Controlar en una iteración lo que ha pasado en la anterior. Dificultad Media.

27. **[Número Narcisista]** Un número entero de n dígitos se dice que es **narcisista** si se puede obtener como la suma de las potencias n -ésimas de cada uno de sus dígitos. Por ejemplo 153 y 8208 son números narcisistas porque $153 = 1^3 + 5^3 + 3^3$ (153 tiene

3 dígitos) y $8208 = 8^4 + 2^4 + 0^4 + 8^4$ (8208 tiene 4 dígitos). Construya un programa que, dado un número entero positivo, nos indique si el número es o no narcisista.

Finalidad: Ejercitar los bucles. Dificultad Media.

28. **[Parejas de caracteres]** Escriba un programa que lea cuatro valores de tipo `char` (`min_izda`, `max_izda`, `min_dcha`, `max_dcha`) e imprima las parejas que pueden formarse con un elemento del conjunto `{min_izda ... max_izda}` y otro elemento del conjunto `{min_dcha ... max_dcha}`. Por ejemplo, si `min_izda = b`, `max_izda = d`, `min_dcha = j`, `max_dcha = m`, el programa debe imprimir las parejas que pueden formarse con un elemento de `{b c d}` y otro elemento de `{j k l m}`, es decir:

```
bj bk bl bm
cj ck cl cm
dj dk dl dm
```

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

29. **[Pirámide]** Cree un programa que ofrezca en pantalla la siguiente salida:

```
1 2 3 4 5 6
2 3 4 5 6
3 4 5 6
4 5 6
5 6
6
```

Modifique la solución para que se lea desde teclado el valor inicial y el número de filas a imprimir. En el ejemplo anterior el valor inicial era 1 y se imprimía un total de 6 filas.

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

30. **[Cuadrado]** Cree un programa que ofrezca en pantalla la siguiente salida:

```
1 2 3 4 5 6
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10
6 7 8 9 10 11
```

Modifique la solución para que se lea desde teclado el valor inicial y el número de filas a imprimir. En el ejemplo anterior el valor inicial era 1 y se imprimía un total de 6 filas.

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

31. [Gaussiana con un menú] Recupere la solución del ejercicio 16 (función gaussiana) de la relación de problemas I. Se pide crear un menú principal para que el usuario pueda elegir las siguientes opciones:

Introducir parámetros de la función (esperanza y desviación)
Salir del programa

Si el usuario elige la opción de salir, el programa terminará; si elige la opción de introducir los parámetros, el programa leerá los dos parámetros (esperanza y desviación). La media puede ser un valor cualquiera, pero la desviación ha de ser un número positivo. A continuación, el programa presentará un menú con las siguientes opciones:

Introducir rango de valores de abscisas
Volver al menú anterior (el menú principal)

Si el usuario elige volver al menú anterior, el programa debe presentar el primer menú (el de la introducción de los parámetros) Si el usuario elige introducir los valores de abscisas, el programa le pedirá un valor mínimo, un valor máximo (ha de ser mayor que mínimo) y un incremento y mostrará el valor de la función gaussiana en todos los valores de x (la abscisa) entre mínimo y máximo a saltos de incremento, es decir, mínimo, mínimo + incremento, mínimo + 2*incremento, ..., hasta llegar, como mucho, a máximo. Después de mostrar los valores de la función, el programa volverá al menú de introducción del rango de valores de abscisas.

Ejemplo de entrada: P 12 5 R 11 13 0.5 V S (Se han elegido las letras P para introducir parámetros, R para introducir el rango, V para volver del menú secundario al principal y S para salir del programa)

— Salida correcta:

```
f(11)=0.0782085
f(11.5)=0.0793905
f(12)=0.0797885
f(12.5)=0.0793905
f(13)=0.0782085
```

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

32. [Tarifa aérea: múltiples billetes] Recupere la solución del ejercicio 18 [Tarifa aérea con filtro de entrada de datos] disponible en:

http://decsai.ugr.es/jccubero/FP/II_TarifaAereaFiltroEntrada.cpp

Modifique dicha solución para que el programa calcule la tarifa final de una serie de billetes. Para ello, cada vez que se introduzcan los datos de un nuevo billete, el usuario introducirá el carácter 'N'. La entrada de datos finaliza con el carácter '#'. Si se introduce un carácter distinto, el programa pedirá un nuevo carácter.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

En este ejercicio no podrá separar las E/S de los cálculos, por lo que dentro del mismo bucle tendrá que realizar tanto los cálculos como las E/S.

Ejemplo de entrada:

```
J K N -2 -5 200      1300 -450 250      P N 650      1300 -450 90 #
```

— Salida correcta: 144 185

Observe que los filtros de entrada de datos han *rechazado* los datos
J K -2 -5 1300 -450 P 1300 -450

Finalidad: Bucles anidados. Lectura de datos con terminador. Dificultad Baja.

33. **[Bits to char]** (*Examen Julio 2017*) Implemente un programa que lea *bits* (ceros y unos) desde teclado, hasta que se introduzca un valor negativo. Si se introduce un positivo distinto de 0 y 1, el programa lo descartará y volverá a leer un valor.

Cada 8 valores de *bits* leídos, el programa calculará el número entero que representa y lo transformará en el carácter (*char*) correspondiente. Debe tener en cuenta que el primer *bit* leído es el más significativo.

Si no es posible completar el último bloque con 8 bits (porque se haya introducido un negativo antes del octavo bit), se descartarán todos los bits de ese último bloque incompleto.

Si el carácter obtenido corresponde a una letra -mayúscula o minúscula- lo mostrará por pantalla. Una vez terminada la entrada de datos, el programa mostrará el porcentaje de letras y otros símbolos leídos.

Ejemplo de entrada:

```
0 1 0 0 1 3 4 1 1 1 2 0 1 1 0 1 0 1 1 9 0 0 1 7 0 0 0 0 1 1 0 -1
```

— Salida correcta:

Ok

Letras: 66.67%

Otros: 33.33%

Donde las correspondencias son 01001111 → 0, 01101011 → k, y 00100001 → !.
Los *bits* finales 10 se descartan ya que no se ha completado un bloque de ocho.

Finalidad: Bucles anidados y lectura de datos. Dificultad Media.

34. **[Cuadro de límites de velocidad]** La Dirección General de Tráfico publica en su página web la sanción a aplicar cuando se sobrepasa el límite de velocidad establecido en una vía. La sanción consta de una multa económica y una detracción del número de puntos del carnet del infractor.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

El siguiente cuadro está publicado en

[https://sede.dgt.gob.es/Galerias/tramites-y-multas/
alguna-multa/consulta-de-sanciones-por-exceso-velocidad/
cuadro_velocidad.pdf](https://sede.dgt.gob.es/Galerias/tramites-y-multas/alguna-multa/consulta-de-sanciones-por-exceso-velocidad/cuadro_velocidad.pdf)



Cuadro para excesos de velocidad (Fecha de entrada en vigor 25/05/2010)

Límite		30	40	50	60	70	80	90	100	110	120	Multa	Puntos
Exceso de velocidad	Grave	31 50	41 60	51 70	61 90	71 100	81 110	91 120	101 130	111 140	121 150	100	-
		51 60	61 70	71 80	91 110	101 120	111 130	121 140	131 150	141 160	151 170	300	2
		61 70	71 80	81 90	111 120	121 130	131 140	141 150	151 160	161 170	171 180	400	4
		71 80	81 90	91 100	121 130	131 140	141 150	151 160	161 170	171 180	181 190	500	6
	Muy Grave	81	91	101	131	141	151	161	171	181	191	600	6

Escriba un programa en C++ para imprimir los límites inferiores indicados dentro del cuadro. Es decir, el programa debe imprimir lo siguiente:

```
31 41 51 61 71 81 91 101 111 121
51 61 71 91 101 111 121 131 141 151
61 71 81 111 121 131 141 151 161 171
71 81 91 121 131 141 151 161 171 181
81 91 101 131 141 151 161 171 181 191
```

Debe usar un bucle anidado en otro y establecer los incrementos convenientes. Por ejemplo, el incremento entre dos columnas consecutivas de una misma fila es de 10, salvo el incremento que hay entre la tercera y cuarta columnas que es de 10 para la primera fila, de 20 para la segunda y de 30 para la tercera fila y siguientes.

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

35. [Número feliz] Se dice que un número natural es feliz si cumple que si sumamos los cuadrados de sus dígitos y seguimos el proceso con los resultados obtenidos, finalmente obtenemos uno (1) como resultado. Por ejemplo, el número 203 es un número feliz ya que $2^2 + 0^2 + 3^2 = 13 \rightarrow 1^2 + 3^2 = 10 \rightarrow 1^2 + 0^2 = 1$.

Se dice que un número es feliz de grado k si se ha podido demostrar que es feliz en un máximo de k iteraciones. Se entiende que una iteración se produce cada vez que

RELACIÓN DE PROBLEMAS II. Estructuras de Control

se elevan al cuadrado los dígitos del valor actual y se suman. En el ejemplo anterior, 203 es un número feliz para cualquier grado mayor o igual que 3.

En general, si un número es feliz de grado k' también es feliz para cualquier grado $k \geq k'$

Escriba un programa que lea un valor entero n y un valor k . Si el número es feliz para un grado k' menor o igual que k , el programa parará y mostrará dicho grado k' . En caso contrario, el programa nos dirá que no es feliz para cualquier grado menor o igual que k .

Ejemplo de entrada: 13 2

— Salida correcta: Es feliz para cualquier grado ≥ 2

Ejemplo de entrada: 13 5

— Salida correcta: Es feliz para cualquier grado ≥ 2

Ejemplo de entrada: 13 1

— Salida correcta: No es feliz para cualquier grado ≤ 1

Ejemplo de entrada: 203 1

— Salida correcta: No es feliz para cualquier grado ≤ 1

Ejemplo de entrada: 203 7

— Salida correcta: Es feliz para cualquier grado ≥ 3

Ejemplo de entrada: 102 6

— Salida correcta: No es feliz para cualquier grado ≤ 6

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

Ejercicios complementarios de condicionales

36. [Media y desviación típica] Amplíe el ejercicio 19 de la relación de problemas I, para que, una vez calculada la media y la desviación, el programa imprima por cada uno de los valores introducidos previamente, si está por encima o por debajo de la media. Por ejemplo:

```
33 es menor que su media
48 es mayor o igual que su media
.....
```

Nota. Los valores introducidos son enteros, pero la media y la desviación son reales.

Finalidad: Plantear un ejemplo básico con varias estructuras condicionales dobles consecutivas. *Dificultad Baja.*

37. [Año bisiesto] Cree un programa que lea el número de un año e indique si es bisiesto o no. Un año es bisiesto si es múltiplo de cuatro, pero no de cien. Excepción a la regla anterior son los múltiplos de cuatrocientos que siempre son bisiestos. Por ejemplo, son bisiestos: 1600, 1996, 2000, 2004. No son bisiestos: 1700, 1800, 1900, 1998, 2002.

Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. *Dificultad Baja.*

38. Queremos gestionar la nómina de los empleados de un centro de atención telefónica. Construya un programa que lea el salario por hora (dato de tipo real) de un empleado, el número de horas trabajadas durante el mes actual (dato de tipo entero) el número de casos resueltos de forma satisfactoria (dato de tipo entero) y el grado medio de satisfacción de los usuarios de los servicios telefónicos con el empleado en cuestión (real entre 0 y 5).

Se quiere aplicar una subida salarial en función de varios factores. En ejercicios sucesivos se irán planteando distintas posibilidades. La primera que se quiere implementar es la siguiente:

Se aplicará una subida del 4% a los empleados que han resuelto más de 30 casos.

```
Más de 30 casos resueltos:                +4%
```

Imprima el salario final en pantalla.

Ejemplo de entrada: 8.5 150 32 5 — Salida correcta: 1326

Ejemplo de entrada: 7.5 130 24 3 — Salida correcta: 975

Finalidad: Plantear una estructura condicional de actualización de una variable. *Dificultad Baja.*

39. Recupere la solución del ejercicio 38 sobre el cómputo de la nómina de los trabajadores de un centro de atención telefónica. Implemente ahora el siguiente criterio para la subida salarial. Se aplicará una subida del 4% a los empleados que han resuelto más de 30 casos y una subida del 2% si el grado de satisfacción media de los usuarios es mayor o igual que 4.0. Ambas subidas son compatibles, es decir, si un trabajador cumple las dos condiciones, se le aplicarán ambas subidas.

Resuelva este ejercicio considerando que la nueva subida del 2% se realiza sobre el salario inicial y no sobre el resultado de haber aplicado, en su caso, la otra subida del 4%.

Más de 30 casos resueltos:	+4%
Grado de satisfacción ≥ 4 :	+2%

Ejemplo de entrada: 8.5 150 32 5 — Salida correcta: 1351.5

Ejemplo de entrada: 8.5 150 29 5 — Salida correcta: 1300.5

Ejemplo de entrada: 7.5 130 24 3 — Salida correcta: 975

Finalidad: Plantear estructuras condicionales consecutivas. Dificultad Baja.

40. Escriba un programa en C++ para que lea tres enteros desde teclado y nos diga si están ordenados (da igual si es de forma ascendente o descendente) o no lo están. Por ejemplo, la sucesión de números 3, 6, 9 estaría ordenada así como la serie 13, 2, 1 pero no lo estaría la serie 3, 9, 5.

Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. Dificultad Baja.

41. [Tres valores ordenados separando E/S y C con un enumerado] Modifique el ejercicio 40 para que el programa nos diga si los tres valores leídos están ordenados de forma ascendente, ordenados de forma descendente o no están ordenados. Para resolver este problema, debe usar una variable de tipo enumerado.

Finalidad: Separación de E/S y C. Usar el tipo enumerado para detectar cuándo se produce una situación determinada. Dificultad Baja.

42. Modifique la solución del ejercicio 39 para que ambas subidas salariales sean excluyentes, es decir, si se aplica una, no se aplicará la otra. En el caso de que ambas sean aplicables, debe aplicarse la subida más ventajosa para el trabajador, es decir, la del 4%.

De forma exclusiva:

Más de 30 casos resueltos:	+4%
Grado de satisfacción ≥ 4 :	+2%

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Ejemplo de entrada: 8.5 150 32 5 — Salida correcta:

Ejemplo de entrada: 8.5 150 29 5 — Salida correcta:

Ejemplo de entrada: 7.5 130 24 3 — Salida correcta:

Finalidad: Plantear estructuras condicionales anidadas. Dificultad Baja.

43. La tabla para el cálculo del precio a pagar en los parkings de Madrid para el 2015 es la siguiente:

Si permanece más de 660 minutos se paga una única tarifa de 31.55 euros

Desde el minuto 0 al 30: 0.0412 euros cada minuto

Desde el minuto 31 al 90: 0.0370 euros cada minuto

Desde el minuto 91 al 120: 0.0311 euros cada minuto

Desde el minuto 121 al 660: 0.0305 euros cada minuto

Dado un tiempo de entrada (hora, minuto y segundo) y un tiempo de salida, construya un programa que calcule la tarifa final a cobrar. Para calcular el número de minutos entre los dos instantes de tiempo, puede utilizar la solución del ejercicio 8 [Segundos entre dos instantes] de la Relación de Problemas I.

Ejemplo de entrada: 2 1 30 2 1 29 — Salida correcta: -1

Ejemplo de entrada: 2 1 30 2 1 31 — Salida correcta: 0

Ejemplo de entrada: 2 1 30 2 2 31 — Salida correcta: 0.0412

Ejemplo de entrada: 2 1 30 2 41 31 — Salida correcta: 1.606

Ejemplo de entrada: 2 1 30 3 41 31 — Salida correcta: 3.767

Ejemplo de entrada: 2 1 30 5 41 31 — Salida correcta: 7.439

Ejemplo de entrada: 2 1 30 23 1 1 — Salida correcta: 31.55

Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

44. Modifique la solución del ejercicio 39 para que también aplique una subida del 3% a los que han resuelto entre 20 y 30 casos:

Entre 20 y 30 casos resueltos: +3%

Más de 30 casos resueltos: +4%

Grado de satisfacción ≥ 4 : +2%

Ejemplo de entrada: 8.5 150 32 5 — Salida correcta: 1351.5

Ejemplo de entrada: 7.5 130 24 3 — Salida correcta: 1004.25

Ejemplo de entrada: 7.5 130 24 4 — Salida correcta: 1023.75

Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

45. Construya un programa para calcular el importe total a facturar de un pedido. El programa leerá el número de unidades vendidas y el precio de venta de cada unidad. Si la cantidad vendida es mayor de 100 unidades, se le aplica un descuento del 3%. Por otra parte, si el precio final de la venta es mayor de 700 euros, se aplica un descuento

del 2 %. Ambos descuentos son acumulables. Obtenga el importe final e imprímalo en pantalla.

Vamos a cambiar el criterio de los descuentos. Supondremos que sólo se aplicará el descuento del 2 % (por una venta mayor de 700 euros) cuando se hayan vendido más de 100 unidades, es decir, para ventas de menos de 100 unidades no se aplica el descuento del 2 % aunque el importe sea mayor de 700 euros.

Cambiar el programa para incorporar este nuevo criterio.

Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

46. Modifique la solución del ejercicio 40 (valores ordenados) para que no se mezclen E/S y C (entradas/salidas y cálculos) dentro de la misma estructura condicional.

Finalidad: Diseñar programas que separen Entradas/Salidas y cálculos. Dificultad Baja.

47. Modifique la solución del ejercicio 37 (año bisiesto) para que no se mezclen E/S y C (entradas/salidas y cálculos) dentro de la misma estructura condicional.

Finalidad: Diseñar programas que separen Entradas/Salidas y cálculos. Dificultad Baja.

48. Cree un programa que lea los datos fiscales de una persona, reajuste su renta bruta según el criterio que se indica posteriormente e imprima su renta neta final.

- La renta bruta es la cantidad de dinero íntegra que el trabajador gana.
- La retención fiscal es el tanto por ciento que el gobierno *se queda*.
- La renta neta es la cantidad que le queda al trabajador después de quitarle el porcentaje de retención fiscal, es decir:

$$\text{Renta_neta} = \text{Renta_bruta} - \text{Renta_bruta} * \text{Retención final} / 100$$

Los datos a leer son:

- Si la persona es un trabajador autónomo o no
- Si es pensionista o no
- Estado civil
- Renta bruta (total de ingresos obtenidos)
- Retención inicial a aplicar.

La retención inicial se va a modificar ahora atendiendo al siguiente criterio:

- Se baja 3 puntos la retención fiscal a los autónomos, es decir, si la retención inicial era de un 15 %, por ejemplo, la retención final a aplicar será de un 12 % (por lo que la renta neta final será mayor)
- Para los no autónomos:

- Se sube un punto la retención fiscal a todos los pensionistas, es decir, si la retención inicial era de un 13 %, por ejemplo, la retención final a aplicar será de un 14 % (por lo que la renta neta final será menor)
- Al resto de trabajadores (no autónomo y no pensionista) se le aplica a todos una primera subida lineal de dos puntos en la retención inicial. Una vez hecha esta subida, se le aplica (sobre el resultado anterior) las siguientes subidas **adicionales**, dependiendo de su estado civil y niveles de ingresos:
 - o Se sube otros dos puntos la retención fiscal si la renta bruta es menor de 20.000 euros
 - o Se sube otros 2.5 puntos la retención fiscal a los casados con renta bruta superior a 20.000 euros
 - o Se sube otros tres puntos la retención fiscal a los solteros con renta bruta superior a 20.000 euros

Una vez calculada la retención final, habrá que aplicarla sobre la renta bruta para así obtener la renta final del trabajador.

Finalidad: Plantear una estructura condicional anidada. Dificultad Media.

Ejercicios complementarios de bucles

49. Se pide leer un carácter desde teclado, obligando al usuario a que sea una letra mayúscula. Para ello, habrá que usar una estructura repetitiva `while`, de forma que si el usuario introduce un carácter que no sea una letra mayúscula, se le volverá a pedir otro carácter. Calcule la minúscula correspondiente e imprímala en pantalla. No pueden usarse las funciones `tolower` ni `toupper` de la biblioteca `cctype`.

Finalidad: Trabajar con bucles con condiciones compuestas. Dificultad Baja.

50. Se pide leer dos enteros `min` y `max` que representarán un rango de valores `[min,max]`. El primer valor a leer, `min`, debe ser un número positivo y el segundo valor `max`, debe ser mayor que `min`. El programa irá leyendo estos dos valores hasta que el usuario los introduzca correctamente.

Una vez leídos ambos valores, el programa pedirá otro entero nuevo obligando a que esté dentro del intervalo `[min, max]`. Si el usuario introduce más de 3 valores fuera del rango, el bucle terminará y se mostrará en pantalla un mensaje indicando que superó el número de intentos máximo. En caso contrario, es decir, el usuario introduce un valor en el rango pedido, el bucle también terminará y se mostrará en pantalla el resultado de calcular `nuevo - min y max - nuevo`.

Ejemplo de entrada: -5 -6 -7 -1 5 3 4 2 8 7

— Salida correcta: 2 1

Ejemplo de entrada: -5 -6 -7 -1 5 3 4 2 8 4 9 7

— Salida correcta: 2 1

Ejemplo de entrada: -5 -6 -7 -1 5 3 4 2 8 4 9 10

— Salida correcta: Número de intentos sobrepasado

Finalidad: Trabajar con bucles con condiciones compuestas en filtros de entrada de datos. Dificultad Media.

51. **[Diferencia entre instantes con filtro de entrada]** Amplíe el ejercicio 8 **[Segundos entre dos instantes]** de la Relación de Problemas I de manera que se obligue al usuario a que el segundo instante introducido sea posterior al primero. Suponemos que ambos instantes corresponden al mismo día. Para comparar los instantes puede usar la solución del ejercicio 10 **[Comparación de dos instantes]** de esta Relación de Problemas.

Finalidad: Trabajar con condicionales complejos y filtros de entradas de datos. Reutilizar código ya escrito y verificado. Dificultad Media.

52. Modifiquemos el ejercicio 7 del capital y los intereses de la primera relación. Supongamos ahora que se quiere reinvertir todo el dinero obtenido (el original C más los intereses producidos) en otro plazo fijo a un año y así, sucesivamente. Construya un programa para que lea el capital, el interés y un número de años N , y calcule e imprima todo el dinero obtenido durante cada uno de los N años, suponiendo que todo lo ganado (incluido el capital original C) se reinvierte a plazo fijo durante el siguiente año. El programa debe mostrar una salida del tipo:

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Total en el año número 1 = 240
Total en el año número 2 = 288
Total en el año número 3 = 345.6
.....

Finalidad: Usar una variable acumuladora dentro del cuerpo de un bucle (aparecerá a la izquierda y a la derecha de una asignación). Dificultad Baja.

53. Sobre el mismo ejercicio del capital y los intereses, construya un programa para calcular cuántos años han de pasar hasta llegar a doblar, como mínimo, el capital inicial. Los datos que han de leerse desde teclado son el capital inicial y el interés anual.

Finalidad: Usar la variable acumuladora en la misma condición del bucle. Dificultad Baja.

54. Realice un programa que lea dos secuencias de enteros desde teclado y nos diga si todos los valores de la primera secuencia son mayores que todos los valores de la segunda secuencia.

Realice la lectura de los enteros dentro de sendos bucles sobre una única variable llamada dato. El final de cada secuencia viene marcado cuando se lee el 0.

Finalidad: Ejercitar el uso de bucles. Dificultad Baja.

55. Amplie el ejercicio 38 (Población) de la relación de problemas I.

Esta nueva versión del programa, además de los datos ya pedidos en dicho ejercicio, se le pedirá al usuario que introduzca un número de años (será el último dato leído). Debe leer cada dato con un filtro conveniente. Por ejemplo, las tasas de natalidad, mortalidad y emigración deben ser enteros entre 0 y 1000, mientras que la población inicial debe ser un entero positivo.

El programa debe calcular e imprimir el número total de habitantes transcurridos dichos años.

Además, el programa también calculará el número de años que tienen que pasar hasta que haya, como mínimo, el doble de la población inicial. Imprima dicho número de años, junto con la población que habrá pasado ese tiempo.

Por ejemplo, para la siguiente entrada

```
1375570814  <- Población inicial
32           <- Tasa de natalidad
12           <- Tasa de mortalidad
7           <- Tasa de migración
3           <- Número de años
```

el programa debe devolver lo siguiente:

RELACIÓN DE PROBLEMAS II. Estructuras de Control

1490027497 <- Número de habitantes pasados 3 años
27 <- Años que han de pasar hasta doblar la población
2824131580 <- Población transcurridos 27 años

Ejemplo de entrada: 1375570814 32 12 7 3
— Salida correcta: 1490027497 27 2824131580

Finalidad: Ejemplo básico de asignación acumulada. Dificultad Baja.

56. Una empresa que tiene tres sucursales decide llevar la contabilidad de las ventas de sus productos a lo largo de una semana. Para ello registra cada venta con tres números, el identificador de la sucursal (1, 2 o 3), el código del producto codificado como un carácter (a, b ó c) y el número de unidades vendidas. Diseñar un programa que lea desde el teclado una serie de registros compuestos por `sucursal`, `producto`, `unidades` y diga cuál es la sucursal que más productos ha vendido. La serie de datos termina cuando la sucursal introducida vale -1. Por ejemplo, con la serie de datos

```
2 a 20
1 b 10
1 b 4
3 c 40
1 a 1
2 b 15
1 a 1
1 c 2
2 b 6
-1
```

Se puede ver que la sucursal que más productos ha vendido es la número 2 con 41 unidades totales. Para comprobar que el programa funciona correctamente, cread un fichero de texto y re-dirigid la entrada a dicho fichero.

Finalidad: Ver un bucle en el que se leen varios datos en cada iteración, pero sólo uno de ellos se usa como terminador de la entrada. Dificultad Media.

57. Se quiere construir un programa para leer los datos necesarios del ejercicio 44 de la subida salarial.

Supondremos que sólo hay tres empleados y que están identificados con un código (1, 2 y 3). Además, el salario por hora es el mismo para todos los empleados. Éste será el primer valor que se leerá (de tipo `double`) Después de haber leído este dato, se leerán los datos de los casos atendidos por los empleados en el siguiente orden: en primer lugar, el código del empleado, a continuación el número de segundos que ha durado la atención telefónica, en tercer lugar un 1 si el caso se resolvió de forma

RELACIÓN DE PROBLEMAS II. Estructuras de Control

satisfactoria y un 0 en caso contrario; finalmente, un valor entero entre 0 y 5 con el grado de satisfacción del usuario.

Cuando nos encontremos el terminador -1 como primer dato (código del empleado) se detendrá la introducción de datos. Supondremos que siempre se introduce al menos el primer valor (el salario), pudiendo ser ya el siguiente dato leído el terminador.

```
7.5      <- Salario de 7.5 euros por hora
2 124 1 3 <- Empleado 2, 124'', resuelto,    grado sat: 3
1 32 0 0  <- Empleado 1, 32'', no resuelto, grado sat: 0
2 26 0 2  <- Empleado 2, 26'', no resuelto, grado sat: 2
-1        <- Fin de entrada de datos
```

El programa debe imprimir el número total de casos introducidos (3 en el ejemplo anterior) y el código del empleado con mayor grado de satisfacción medio (también imprimirá dicho grado medio). En el ejemplo anterior, sería el empleado 2 con un nivel medio de satisfacción de 2.5 (observe que el grado medio se calcula en relación al número total de casos atendidos y no sobre los casos resueltos).

Observe que, en este ejercicio, no se están teniendo en cuenta los datos referentes al tiempo de cada caso y si fue resuelto o no, pero hay que leer todos los datos para llegar a los que sí nos interesan.

Ejemplo de entrada: 7.5 2 124 1 3 1 32 0 0 2 26 0 2 -1

— Salida correcta: 3 2 2.5

Ejemplo de entrada: 7.5 -1

— Salida correcta: No se introdujo ningún caso

Finalidad: Plantear un bucle de lectura de datos. Dificultad Baja.

58. Construya un programa que calcule cuándo se produjo la mayor secuencia de días consecutivos con temperaturas crecientes. El programa leerá una secuencia de reales representando temperaturas, hasta llegar al -1 y debe calcular la subsecuencia de números ordenada, de menor a mayor, de mayor longitud.

El programa nos debe decir la posición donde comienza la subsecuencia y su longitud. Por ejemplo, ante la entrada siguiente:

```
17.2 17.3 16.2 16.4 17.1 19.2 18.9 -1.0
```

el programa nos debe indicar que la mayor subsecuencia empieza en la posición 3 (en el 16.2) y tiene longitud 4 (termina en 19.2)

Puede suponer que siempre se introducirá al menos un valor de temperatura.

Ejemplo de entrada: 17.2 17.3 16.2 16.4 17.1 19.2 18.9 -1

— Salida correcta: 3 4

Ejemplo de entrada: 17.2 17.3 16.2 16.4 17.1 19.2 -1

RELACIÓN DE PROBLEMAS II. Estructuras de Control

— Salida correcta: 3 4

Ejemplo de entrada: 17.2 17.3 -1 — Salida correcta: 1 2

Ejemplo de entrada: 17.2 15.3 -1 — Salida correcta: 1 1

Ejemplo de entrada: 17.2 -1 — Salida correcta: 1 1

Finalidad: Trabajar con bucles que comparan un valor actual con otro anterior. Dificultad Media.

59. En el ejercicio 37 de la Relación de Problemas I se pedía escribir un programa que leyese un valor entero de tres dígitos e imprimiese los dígitos separados por un espacio en blanco. Haga lo mismo pero para un número entero arbitrario. Por ejemplo, si el número es 3519, la salida sería:

3 5 1 9

En este ejercicio se pueden mezclar entradas y salidas con cálculos.

Finalidad: Trabajar con bucles que recorren los dígitos de un número. Dificultad Media.

60. El algoritmo de la multiplicación rusa es una forma distinta de calcular la multiplicación de dos números enteros $n * m$. Para ello este algoritmo va calculando el doble del multiplicador m y la mitad (sin decimales) del multiplicando n hasta que n tome el valor 1 y suma todos aquellos multiplicadores cuyos multiplicandos sean impares. Por ejemplo, para multiplicar 37 y 12 se harían las siguientes iteraciones

Iteración	Multiplicando	Multiplicador
1	37	12
2	18	24
3	9	48
4	4	96
5	2	192
6	1	384

Con lo que el resultado de multiplicar 37 y 12 sería la suma de los multiplicadores correspondientes a los multiplicandos impares (en negrita), es decir $37 * 12 = 12 + 48 + 384 = 444$

Cree un programa para leer dos enteros n y m y calcule su producto utilizando este algoritmo. No puede utilizarse en ningún momento el operador producto $*$.

Dificultad Media.

61. Amplíe el ejercicio 37 (año bisiesto). El programa pedirá los valores de dos años obligando a que estén entre el año cero y 2100. A continuación, el programa mostrará todos los años bisiestos comprendidos entre los años anteriores.

Finalidad: Practicar con filtros y ciclos básicos. Practicar con algoritmos más elaborados y eficientes. Reutilizar código ya escrito y verificado. Dificultad Media.

62. Todo lo que se puede hacer con un bucle `while` se puede hacer con un `do while`. Lo mismo ocurre al revés. Sin embargo, cada bucle se usa de forma natural en ciertas situaciones. El no hacerlo, nos obligará a escribir más código y éste será más difícil de entender. Para comprobarlo, haced lo siguiente:

- a) Modifique la solución del ejercicio 16 de forma que el filtro de entrada usado para leer la variable `tope`, se haga con un bucle pre-test `while`.
- b) Modifique la solución del ejercicio 52 sustituyendo el bucle `while` por un `do while`. Observad que debemos considerar el caso en el que el número de años leído fuese cero.

Finalidad: Enfatizar la necesidad de saber elegir entre un bucle pre-test o un bucle post-test. Dificultad Media.

63. Calcule mediante un programa en C++ el combinatorio $\binom{n}{m}$ con n, m valores enteros. No pueden usarse las funciones de la biblioteca `cmath`.

El combinatorio de n sobre m (con $n \geq m$) es un número entero que se define como sigue:

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

Finalidad: Trabajar con bucles controlados por contador. Dificultad Media.

64. Construya un programa que lea un valor T y calcule la siguiente sumatoria:

$$\sum_{i=1}^{i=T} i! = \sum_{i=1}^{i=T} \left(\prod_{j=1}^{j=i} j \right)$$

Por ejemplo, para $T = 4$, la operación a realizar es:

$$1! + 2! + 3! + 4!$$

es decir:

$$1 + (1 * 2) + (1 * 2 * 3) + (1 * 2 * 3 * 4)$$

Ejemplo de entrada: 3 — Salida correcta: 9

Ejemplo de entrada: 4 — Salida correcta: 33

Ejemplo de entrada: 6 — Salida correcta: 873

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

65. Resuelva el ejercicio 64 sin utilizar bucles anidados, es decir, debe usar un único bucle.

Finalidad: Aprovechar en una iteración los cálculos hechos en la iteración anterior. Dificultad Media.

66. Diremos que un número entero positivo es secuenciable si se puede generar como suma de números consecutivos (al menos dos). Por ejemplo, $6 = 1+2+3$, $15 = 7+8$. Esta descomposición no tiene por qué ser única. Por ejemplo, $15 = 7+8 = 4+5+6 = 1+2+3+4+5$. Escriba un programa que lea un entero $n \geq 1$ e imprima todas las descomposiciones posibles. En este ejercicio puede mezclar operaciones de E/S y C dentro del mismo bucle.

Como curiosidad, los únicos números con 0 descomposiciones son las potencias de 2.

Ejemplo de entrada: 6 — Salida correcta: 1 2 3

Ejemplo de entrada: 15 — Salida correcta: 7 8 / 4 5 6 / 1 2 3 4 5

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

67. (*Examen Septiembre 2014*) ¿Cuántas veces aparece el dígito 9 en todos los números que hay entre el 1 y el 100? Por ejemplo, el 9 aparece una vez en los números 19 y 92 mientras que aparece dos veces en el 99. Pretendemos diseñar un algoritmo que responda a esta sencilla pregunta, pero de forma suficientemente generalizada. Para ello, se pide construir un programa que lea una cifra (entre 1 y 9), dos enteros \min y \max y calcule el número de apariciones del dígito cifra en los números contenidos en el intervalo cerrado $[\min, \max]$.

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

68. Supongamos una serie numérica cuyo término general es:

$$a_i = a_1 r^{i-1}$$

Es decir, la serie la forman los siguientes términos:

$$a_1 = a_1$$

$$a_2 = a_1 r$$

$$a_3 = a_1 r^2$$

$$a_4 = a_1 r^3$$

...

Se pide crear un programa que lea desde teclado r , el primer elemento a_1 y el tope k y calcule la suma de los primeros k valores de la serie, es decir:

$$\sum_{i=1}^{i=k} a_i$$

Se proponen dos alternativas:

- a) Realice la suma de la serie usando la función `pow` para el cómputo de cada término a_i . Los argumentos de `pow` no pueden ser ambos enteros, por lo que forzaremos a que la base (por ejemplo) sea `double`, multiplicando por `1.0`.

- b) Si analizamos la expresión algebraica de la serie numérica, nos damos cuenta que es una *progresión geométrica* ya que cada término de la serie queda definido por la siguiente expresión:

$$a_{i+1} = a_i * r$$

Es decir, una progresión geométrica es una secuencia de elementos en la que cada uno de ellos se obtiene multiplicando el anterior por una constante denominada razón o factor de la progresión.

Cree el programa pedido usando esta fórmula. NO puede utilizarse la función `pow`.

¿Qué solución es preferible en términos de eficiencia?

Finalidad: Trabajar con bucles que aprovechan cálculos realizados en la iteración anterior. Dificultad Baja.

69. Reescribid la solución a los ejercicios 16 (divisores) y 52 (interés) usando un bucle `for`

Finalidad: Familiarizarnos con la sintaxis de los bucles `for`. Dificultad Baja.

70. Diseñar un programa para calcular la suma de los 100 primeros términos de la sucesión siguiente:

$$a_i = \frac{(-1)^i(i^2 - 1)}{2i}$$

No puede usarse la función `pow`. Hacedlo calculando explícitamente, en cada iteración, el valor $(-1)^i$ (usad un bucle `for`). Posteriormente, resolvedlo calculando dicho valor a partir del calculado en la iteración anterior, es decir, $(-1)^{i-1}$.

Finalidad: Enfatizar la conveniencia de aprovechar cálculos realizados en la iteración anterior. Dificultad Media.

71. Se pide diseñar un programa para jugar a adivinar un número entre 1 y 100. El juego tiene que dar pistas de si el número introducido por el jugador está por encima o por debajo del número introducido. Como reglas de parada se consideran los siguientes dos casos:

a) se ha acertado b) se decide abandonar el juego (decida cómo quiere especificar esta opción)

Para poder generar números aleatorios en un rango determinado será necesario incluir las siguientes instrucciones:

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



```
int main(){
    const int MIN = 1, MAX = 100;
    const NUM_VALORES = MAX-MIN + 1;           // rango
    int incognita;                             // número generado
    time_t tiempo;

    // Inicialización de la secuencia:
    srand(time(&tiempo));

    // Generación de un número aleatorio incognita:
    // MIN <= incognita <= MAX
    incognita = (rand() \% NUM_VALORES) + MIN;
```

La sentencia `srand(time(&tiempo))` debe ejecutarse una única vez al principio del programa y sirve para inicializar la secuencia de números aleatorios. Posteriormente, cada vez que se ejecute la sentencia `incognita = (rand() \% NUM_VALORES) + MIN;` se obtendrá un valor aleatorio (pseudoaleatorio).

Realizar el mismo ejercicio pero permitiendo jugar tantas veces como lo desee el jugador.

Dificultad Media.

72. Sobre la solución del ejercicio 52 de esta relación de problemas, se pide lo siguiente. Supondremos que sólo pueden introducirse intereses enteros (1, 2, 3, etc). Se pide calcular el capital obtenido al término de cada año, pero realizando los cálculos para todos los tipos de interés enteros menores o iguales que el introducido (en pasos de 1). Por ejemplo, si el usuario introduce un interés igual a 5 y un número de años igual a 3, hay que mostrar el capital ganado al término de cada uno de los tres años a un interés del 1 %, a continuación, lo mismo para un interés del 2 % y así sucesivamente hasta llegar al 5 %. El programa debe mostrar una salida del tipo:

Cálculos realizados al 1%:

```
Dinero obtenido en el año número 1 = 2020
Dinero obtenido en el año número 2 = 2040.2
Dinero obtenido en el año número 3 = 2060.6
```

Cálculos realizados al 2%:

```
Dinero obtenido en el año número 1 = 2040
Dinero obtenido en el año número 2 = 2080.8
Dinero obtenido en el año número 3 = 2122.42
.....
```

Finalidad: Empezar a trabajar con bucles anidados. Dificultad Baja.

73. Implemente un programa que sea capaz de “dibujar” rectángulos utilizando un símbolo (un carácter) dado. El usuario ingresará el símbolo *simb*, la altura *M* y el ancho *N* del rectángulo. Por ejemplo, siendo *simb* = *, *M* = 3 y *N* = 5, el dibujo tendría la siguiente forma:

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

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

74. Implemente un programa que sea capaz de “dibujar” pinos utilizando asteriscos “*”. El usuario ingresara el ancho de la base del pino (podemos asumir que es un número impar). Supongamos que se ingresa 7, entonces el dibujo tendrá la siguiente forma:

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

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

75. Realizar un programa para calcular los valores de la función:

$$f(x) = \sqrt{\frac{3x + x^2}{1 - x^2}}$$

para valores de *x* enteros en el rango [-3..3].

Dificultad Baja.

76. Realizar un programa para calcular los valores de la función:

$$f(x, y) = \frac{\sqrt{x}}{y^2 - 1}$$

para los valores de (*x*, *y*) con *x* = -50, -48, ..., 48, 50 y = -40, -39, ..., 39, 40, es decir queremos mostrar en pantalla los valores de la función en los puntos

(-50, 40), (-50, -39), ... (-50, 40), (-48, 40), (-48, -39), ... (50, 40)

Dificultad Baja.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

77. Diseñar un programa que presente una tabla de grados C a grados Fahrenheit ($F=9/5C+32$) desde los 0 grados a los 300, con incremento de 20 en 20 grados.

Dificultad Baja.

78. Diseñar un programa que lea caracteres desde la entrada y los muestre en pantalla, hasta que se pulsa el '.' y diga cuántos separadores se han leído (espacios en blanco ' ', tabuladores '\t' y caracteres de nueva línea '\n').

Dificultad Baja.

79. Realizar un programa para calcular la suma de los términos de la serie

$$1 - 1/2 + 1/4 - 1/6 + 1/8 - 1/10 + \dots - 1/(2n - 1) + 1/(2n)$$

para un valor n dado.

Dificultad Baja.

80. Se decide informatizar el acta de un partido de baloncesto para saber qué equipo es el ganador del partido. El acta contiene una serie de anotaciones formadas por una pareja de números cada una, con el dorsal del jugador y el número de puntos conseguidos teniendo en cuenta que la última anotación es un valor -1. Por ejemplo

1 2 4 1 4 1 2 3 6 2 3 2 5 2 5 1 1 3 -1

El programa deberá indicar si ha ganado el equipo 1 (con los dorsales 1, 2 y 3) o el equipo 2 (dorsales 4, 5 y 6) o han empatado.

Por ejemplo, con la entrada anterior, gana el equipo 1.

Dificultad Baja.

81. La Unión Europea ha decidido premiar al país que más toneladas de hortalizas exporte a lo largo del año. Se dispone de un registro de transacciones comerciales en el que aparecen tres valores en cada apunte. El primer valor es el indicativo del país (E: España, F: Francia y A: Alemania), el segundo valor es un indicativo de la hortaliza que se ha vendido en una transacción (T: Tomate, P: Patata, E: Espinaca) y el tercer valor indica las toneladas que se han vendido en esa transacción. Diseñar un programa que lea desde el teclado este registro, el cual termina siempre al leer un país con indicativo '@', y que diga qué país es el que más hortalizas exporta y las toneladas que exporta.

Por ejemplo, con la entrada

E T 10 E T 4 E P 1 E P 1 E E 2 F T 15 F T 6 F P 20 A E 40 @

el país que más vende es Francia con un total de 41 toneladas.

Dificultad Baja.

82. Se pide leer dos enteros sabiendo que el primero no tiene un tamaño fijo y que el segundo siempre es un entero de dos dígitos. Se pide comprobar si el segundo está

RELACIÓN DE PROBLEMAS II. Estructuras de Control

contenido en el primero. Entendemos que está contenido si los dos dígitos del segundo entero están en el primer entero de forma consecutiva y en el mismo orden. Por ejemplo, 89 está contenido en 7890, en 7789 y en 8977 pero no en 7980.

Dificultad Media.

83. Se dice que un número es triangular si se puede poner como la suma de los primeros m valores enteros, para algún valor de m . Por ejemplo, 6 es triangular ya que $6 = 1 + 2 + 3$. Se pide construir un programa que obtenga todos los números triangulares que hay menores que un entero t ope introducido desde teclado.

Dificultad Baja.

84. Escriba un programa que lea por teclado un número entero positivo t ope y muestre por pantalla el factorial de los t primeros números enteros. Recuerda que el factorial de un número entero positivo n es igual al producto de los enteros positivos del 1 al n .

Dificultad Baja.

85. Construya un programa para comprobar si las letras de una palabra se encuentran dentro de otro conjunto de palabras. Los datos se leen desde un fichero de la forma siguiente: el fichero contiene, en primer lugar un total de 3 letras que forman la palabra a buscar, por ejemplo f e o. Siempre habrá, exactamente, tres letras. A continuación, el fichero contiene el conjunto de palabras en el que vamos a buscar. El final de cada palabra viene determinado por la aparición del carácter '@', y el final del fichero por el carácter '#'. La búsqueda tendrá las siguientes restricciones:

- Deben encontrarse las tres letras
- Debe respetarse el orden de aparición. Es decir, si por ejemplo encontramos la 'f' en la segunda palabra, la siguiente letra a buscar 'e' debe estar en una palabra posterior a la segunda.
- Una vez encontremos una letra en una palabra, ya no buscaremos más letras en dicha palabra.
- No nos planteamos una búsqueda barajando todas las posibilidades, en el sentido de que una vez encontrada una letra, no volveremos a buscarla de nuevo.

Entrada:	f e o	
	h o l a @	
	m o f e t a @	<- f
	c o f i a @	
	c e r r o @	<- e
	p e r a @	
	c o s a @	<- o
	h o y @	
	#	

En este caso, sí se encuentra.

Dificultad Media.

86. Un número perfecto es aquel que es igual a la suma de todos sus divisores positivos excepto él mismo. El primer número perfecto es el 6 ya que sus divisores son 1, 2 y 3 y $6=1+2+3$. Escribir un programa que muestre el mayor número perfecto que sea menor a un número dado por el usuario.

Dificultad Media.

87. Escribir un programa que encuentre dos enteros n y m mayores que 1 que verifiquen lo siguiente:

$$\sum_{i=1}^m i^2 = n^2$$

Dificultad Media.

88. En matemáticas, la **sucesión de Fibonacci** (a veces mal llamada *serie* de Fibonacci) es la siguiente sucesión infinita de números naturales:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

La sucesión comienza con los números 1 y 1, y a partir de éstos, cada término puede calcularse como la suma de los dos anteriores. A los elementos de esta sucesión se les llama *números de Fibonacci*.

El número de Fibonacci de orden n , al que llamaremos f_n se puede definir mediante la siguiente relación de recurrencia:

- $f_n = f_{n-1} + f_{n-2}$ para $n > 2$
- $f_1 = f_2 = 1$

Esta sucesión fue descrita en Europa por Leonardo de Pisa, matemático italiano del siglo XIII también conocido como Fibonacci. Tiene numerosas aplicaciones en ciencias de la computación, matemáticas y teoría de juegos. También aparece en diversas configuraciones biológicas.

Escribir un programa que calcule el número de Fibonacci de orden n , donde n es un valor introducido por el usuario. A continuación, el programa solicitará un nuevo valor, k , y mostrará todos los números de Fibonacci $f_1, f_2, f_3, \dots, f_k$.

Finalidad: Trabajar con bucles controlados por contador. Dificultad Media.

89. El número áureo se conoce desde la Antigüedad griega y aparece en muchos temas de la geometría clásica. La forma más sencilla de definirlo es como el único número positivo ϕ que cumple que $\phi^2 - \phi = 1$ y por consiguiente su valor es $\phi = \frac{1 + \sqrt{5}}{2}$.

Se pueden construir aproximaciones al número áureo mediante la fórmula $a_n = \frac{f_{n+1}}{f_n}$ siendo f_n el número de Fibonacci de orden n (ver problema 88).

La sucesión de valores así calculada proporciona, alternativamente, valores superiores e inferiores a ϕ , siendo cada vez más cercanos a éste, y por lo tanto la diferencia entre a_n y ϕ es cada vez más pequeña conforme n se hace mayor.

Escribir un programa que calcule el menor valor de n que hace que la aproximación dada por a_n difiera en menos de δ del número ϕ , sabiendo que $n \geq 1$.

La entrada del programa será el valor de δ y la salida el valor de n . Por ejemplo, para un valor de $\delta = 0.1$ el valor de salida es $n = 4$

Dificultad Media.

90. Una *sucesión alícuota* es una sucesión iterativa en la que cada término es la suma de los divisores propios del término anterior. La sucesión alícuota que comienza con el entero positivo k puede ser definida formalmente mediante la función divisor σ_1 de la siguiente manera:

$$\begin{aligned}s_0 &= k \\ s_n &= \sigma_1(s_{n-1}) - s_{n-1}\end{aligned}$$

Por ejemplo, la sucesión alícuota de 10 es 10, 8, 7, 1, 0 porque:

$$\begin{aligned}\sigma_1(10) - 10 &= 5 + 2 + 1 = 8 \\ \sigma_1(8) - 8 &= 4 + 2 + 1 = 7 \\ \sigma_1(7) - 7 &= 1 \\ \sigma_1(1) - 1 &= 0\end{aligned}$$

Aunque muchas sucesiones alícuotas terminan en cero, otras pueden no terminar y producir una sucesión alícuota periódica de período 1, 2 o más. Está demostrado que si en una sucesión alícuota aparece un *número perfecto* (como el 6) se produce una sucesión infinita de período 1. Un *número amigable* produce una sucesión infinita de período 2 (como el 220 ó 284).

Escribir un programa que lea un número natural menor que 1000 y muestre su sucesión alícuota. Hay que tener en cuenta que en ocasiones se pueden producir sucesiones infinitas, por lo que en estos casos habrá que detectarlas e imprimir puntos suspensivos cuando el período se repita. Solo hay que considerar períodos infinitos de longitud 2 como máximo. Por ejemplo; para el número 6, se imprimiría: 6, 6, ...; y para el número 220, se imprimiría: 220, 284, 220, 284, ...

Finalidad: Practicar los bucles anidados y controlar las condiciones de parada a partir de lo sucedido en iteraciones pasadas.. Dificultad Media.

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

En todos los ejercicios, tenga en cuenta lo siguiente:

- Siempre ha de diseñar una batería de pruebas, que intente garantizar que la solución propuesta funcione correctamente en cualquier situación extrema.
- Recuerde lo visto en las transparencias del primer tema. Si queremos leer datos de tipo `char`, para poder leer un espacio en blanco **no** puede emplear `cin >> character`, sino `character = cin.get()`. Cada vez que se ejecute `cin.get()` el compilador lee un carácter (incluido el espacio en blanco, el tabulador y el retorno de carro `'\n'`) desde la entrada de datos por defecto. En definitiva, el bucle de lectura de datos será del tipo:

```
character = cin.get();

while (character != TERMINADOR){
    .....
    character = cin.get();
}
```

Cuando la entrada de datos es desde un fichero de texto, cada ejecución de `cin.get()` lee directamente un carácter (incluidos los espacios en blanco, tabuladores y retornos de carro) y el programa pasa a la siguiente sentencia.

Si fuese desde el teclado, habría que esperar a que el usuario introdujese el retorno de carro para que los datos pasasen al buffer y una vez ahí, se ejecutarían automáticamente todos los `cin.get()` (recordad lo visto al final del primer tema) En el caso de que, por ejemplo, quisiéramos parar la lectura cuando se hubiesen introducido más de un número tope de caracteres, no podríamos hacerlo con la lectura desde teclado ya que los datos no pasan al buffer hasta que no se pulse el retorno de carro.

- Algunos ejercicios usan un vector muy grande. Para que el vector con dicho texto quepa en la pila, debe hacer los siguientes cambios:
 - Declare una constante de tamaño del vector que permita almacenar todos los datos.

```
const int TAMANIO = 2500000;
<tipo de dato> vector[TAMANIO];
```

Si se prefiere, para que quede más claro el número de datos reservados en memoria, puede usar una constante `double` en formato científico y dejar que C++ haga el casting a `int` automáticamente:

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

```
const int TAMANIO = 25e+5; // int = double  
<tipo de dato> vector[TAMANIO];
```

- Aumente el tamaño de la pila asignada por el sistema operativo al programa generado por el compilador. Para ello, debe seleccionar desde DevC++:

Herramientas -> Opciones del Compilador ->

Señale la opción

Añadir los siguientes comandos al llamar al compilador
y en la caja de texto introduzca lo siguiente (todo seguido sin espacios en blanco):

```
-Wl,--stack,n
```

donde *n* será un número entero que indicará el número de bytes que va a permitir almacenar en la pila. Por ejemplo, si necesita un vector de 25e+5 datos de tipo `double`, necesitará un mínimo de $25e+5 * 8$ bytes. Reserve algo más para otras variables del programa, por lo que un valor de *n* adecuado en este caso sería de $2.1e+7$. Así pues, pondríamos lo siguiente:

```
-Wl,--stack,21000000
```

Tenga en cuenta que aquí no puede usar la notación científica.

Problemas sobre vectores

1. **[Palíndromo e Invierte]** Tenga en cuenta la observación al inicio de esta relación de problemas sobre la lectura de los caracteres (ver página **RP-III.1**). Para poder leer caracteres, incluyendo los espacios en blanco, hay que usar `caracter = cin.get()`, en vez de `cin >> caracter`.

Implemente algoritmos para realizar las siguientes tareas:

- a) Comprobar si el vector es un palíndromo, es decir, que se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, {'a', 'b', 'b', 'a'} sería un palíndromo, pero {'a', 'c', 'b', 'a'} no lo sería. Si la secuencia tiene un número impar de componentes, la que ocupa la posición central es descartada, por lo que {'a', 'b', 'j', 'b', 'a'} sería un palíndromo.
- b) Invertir el vector. Si éste contenía, por ejemplo, los caracteres {'m', 'u', 'n', 'd', 'o'}, después de llamar al método se quedará con {'o', 'd', 'n', 'u', 'm'}.

Construya un programa principal y declare un vector de caracteres de tamaño 100. Lea las componentes considerando como terminador el carácter # (éste no forma parte de la secuencia) y que no se introduzcan más de 100 caracteres. A continuación, el programa debe determinar si la secuencia es un palíndromo. En caso negativo, debe invertirla y mostrar el resultado en pantalla.

Ejemplo de entrada: a# — Salida correcta: Es un palíndromo

Ejemplo de entrada: abcba# — Salida correcta: Es un palíndromo

Ejemplo de entrada: abccba# — Salida correcta: Es un palíndromo

Ejemplo de entrada: abcdab#

— Salida correcta: No es un palíndromo. Secuencia invertida: abdcba

Finalidad: Recorrer las componentes de un vector. Dificultad Baja.

2. **[Comprobación Fecha]** Se quiere construir un programa que compruebe si una fecha es correcta. El programa pedirá los datos de día mes y año e indicará si la fecha correspondiente es correcta o no. El programa pedirá los datos de varias fechas, hasta que el usuario introduzca cualquier valor negativo en el dato del día.

Para resolver este problema, debe tener en cuenta lo visto sobre los años bisiestos en el ejercicio 26 **[Expresiones lógicas]** de la Relación de Problemas I así como los días que trae cada mes (Noviembre 30, Diciembre 31, etc) Se aconseja usar un vector con tantas componentes como meses hay (12).

Separe la parte de cálculos con las Entradas/Salidas del programa.

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

Ejemplo de entrada: 10 12 1967 10 15 1967 10 1 2015 -1
— Salida correcta: SI NO SI

Finalidad: Usar vectores auxiliares para realizar recorridos de datos. Dificultad Baja.

3. [Máximo desnivel] Supongamos un vector de enteros relativo a un conjunto de alturas de un track GPS. Cada entero representa la altura de la posición GPS en un instante dado. Se quiere calcular lo siguiente:

- Máximo desnivel (máxima diferencia en valor absoluto) entre dos alturas consecutivas.

Por ejemplo, si el vector es 1 3 2 4 7 5, las diferencias son +2 -1 +2 +3 -2, siendo 3 la máxima en valor absoluto. Si el vector fuese 1 3 2 4 1 2, las diferencias serían +2 -1 +2 -3 +1, siendo 3 la máxima en valor absoluto.

- Desnivel acumulado positivo.

La idea es ir buscando puntos consecutivos en los que la altura vaya aumentando. Cuando se encuentre una altura menor, se para y se acumula la suma de las alturas anteriores al desnivel acumulado.

Por ejemplo, si el vector es 1 3 2 4 7 5, las diferencias son +2 -1 +2 +3 -2, por lo que el desnivel acumulado positivo será 7 (la suma de +2 +2 +3)

Construir un programa que, en primer lugar, lea un número entero que indique el número de valores de altura que se van a introducir. A continuación el programa leerá dichos valores y los almacenará en un vector. El programa calculará el máximo desnivel y el desnivel acumulado positivo y los imprimirá en pantalla.

Ejemplo de entrada: 6 1 3 2 4 7 5

— Salida correcta: 3 7

Ejemplo de entrada: 2 4 1

— Salida correcta: 3 0

Ejemplo de entrada: 1 7

— Salida correcta: No hay datos suficientes

Finalidad: Recorridos sobre un vector. Dificultad Baja.

4. [Sistema de D'Hondt] ([Examen Febrero 2017](#)) El sistema D'Hondt es el método que se utiliza en España para asignar los escaños del Congreso de los Diputados. Se quiere construir un programa que lea el número total de escaños a distribuir, el número de partidos que han participado en las elecciones y los votos obtenidos por cada uno de ellos. El programa mostrará cuántos escaños obtuvo cada partido.

La asignación de los escaños se hace a través de un proceso iterativo en el que en cada iteración se asigna un escaño a un partido y así hasta llegar al número total de

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

escaños a repartir. En una determinada iteración, un partido se llevará un escaño si tiene el mayor cociente de D'Hondt, definido éste como sigue:

$$\text{Cociente de D'Hondt} = V_i / (S_i + 1)$$

dónde V_i es el número total de votos obtenidos en las elecciones por el partido i y S_i es el número de escaños asignados hasta esa iteración al partido i .

Abajo se muestra un ejemplo de este proceso iterativo 5 escaños a repartir y 4 partidos. Un asterisco (*) nos indica que el partido correspondiente (la columna) se lleva el escaño a asignar en esa iteración (la fila).

Votos	Partido A $V_A = 340000$	Partido B $V_B = 280000$	Partido C $V_C = 160000$	Partido D $V_D = 60000$
Escaño 1	(340000/1 =) 340000*	(280000/1 =) 280000	(160000/1 =) 160000	(60000/1 =) 60000
Escaño 2	(340000/2 =) 170000	(280000/1 =) 280000*	(160000/1 =) 160000	(60000/1 =) 60000
Escaño 3	(340000/2 =) 170000*	(280000/2 =) 140000	(160000/1 =) 160000	(60000/1 =) 60000
Escaño 4	(340000/3 =) 113333	(280000/2 =) 140000	(160000/1 =) 160000*	(60000/1 =) 60000
Escaño 5	(340000/3 =) 113333	(280000/2 =) 140000*	(160000/2 =) 80000	(60000/1 =) 60000
E. asignados	2	2	1	0

Ejemplo de entrada: 7 5 340000 280000 160000 60000 15000

— Salida correcta: 3 3 1 0 0

Finalidad: Recorrido de las componentes de un vector y usar vectores auxiliares. Dificultad Baja.

5. [Elimina ocurrencias de una componente -versión ineficiente-] Se desea eliminar todas las apariciones de un determinado carácter `a_borrar`, dentro de un vector de caracteres. Por ejemplo, después de eliminar el carácter 'o' del vector {'S', 'o', 'Y', ' ', 'y', 'o'}, éste debe quedarse con {'S', 'Y', ' ', 'y'}.

Un primer algoritmo para resolver este problema sería el siguiente (en ejercicios posteriores se verán métodos más eficientes):

Recorrer todas las componentes del vector

Si la componente es igual al carácter `a_borrar`, eliminarla
(desplazando hacia la izda las componentes que hay a su dcha)

A la hora de implementar el anterior código, debe prestar especial atención cuando hay dos caracteres a borrar consecutivos.

Construya un programa que lea caracteres hasta que se introduzca # lo que formará el vector. A continuación lea el carácter a eliminar. El programa imprimirá el vector resultante después de eliminar todas las apariciones de dicho carácter.

Ejemplo de entrada: maaaovaiala#a

— Salida correcta: movil

Ejemplo de entrada: aaaaaa#a

— — Salida correcta: *vector vacío*

Finalidad: Recorrido de un vector eliminando componentes. Dificultad Baja.

6. [Elimina ocurrencias de una componente -versión eficiente-] El algoritmo del ejercicio 5 [Elimina ocurrencias de una componente -versión ineficiente-] es muy ineficiente ya que requiere trasladar muchas veces muchas posiciones (usa dos bucles anidados). Para comprobarlo, recupere el texto

http://decsai.ugr.es/jccubero/FP/Quijote_con_ruido.txt

En él aparecen numerosas ocurrencias del carácter '~' (código ASCII 126) como ruido en el texto y se quieren eliminar. Este fichero contiene los datos en el orden esperado (`vector#caracter_a_borrar`) para poder redirigir directamente la entrada de datos del programa. El vector contendrá el texto del Quijote. Al ser un vector muy grande, recuerde compilar el programa con las instrucciones detalladas en la página **RP-III.1**.

Si aplica el algoritmo ineficiente, la ejecución puede tardar más de **12 minutos** (recuerde compilar el programa con las instrucciones detalladas en la página **RP-III.1**)

Para resolver eficientemente este problema se propone utilizar dos variables, `posicion_lectura` y `posicion_escritura` que nos vayan indicando, en cada momento, la componente que se está leyendo y el sitio dónde tiene que escribirse. Por ejemplo, supongamos que en un determinado momento la variable `posicion_lectura` vale 6 y `posicion_escritura` 3. Si la componente en la posición 6 es el carácter a borrar, simplemente avanzaremos `posicion_lectura`. En caso contrario, la colocaremos en la posición 3 y avanzaremos una posición ambas variables.

Implemente este algoritmo y observe la diferencia de tiempo al ejecutarlo sobre el Quijote, ya que ahora el tiempo de ejecución es de unos **8 milisegundos**.

*Finalidad: Modificar un vector a través de dos **apuntadores**. Dificultad Media.*

7. [Sustituir carácter por vector (con vector auxiliar)] Dado un vector de caracteres, queremos sustituir todas las apariciones de un carácter y poner en su lugar el contenido de otro vector.

Por ejemplo, si tenemos el vector `[u n o a d o s a a]`, el resultado de sustituir las apariciones del carácter 'a' por el nuevo vector `[T T U]` sería `[u n o T T U d o s T T U T T U]`

Resolveremos este problema de varias formas a lo largo de esta Relación de Problemas. En este ejercicio, se construirá un tercer vector sustituido con el resultado pedido.

Construya un programa que lea caracteres hasta que se introduzca # lo que formará el primer vector (`v`). A continuación lea el conjunto de caracteres que sustituirán al

carácter `a_borrar` (también con terminador `#`). Estos caracteres formarán el vector nuevo. Finalmente el programa leerá el carácter (`a_borrar`) a sustituir

El programa construirá e imprimirá en pantalla un tercer vector sustituido que contendrá los caracteres de `v` pero reemplazando todas las apariciones de `a_borrar` por los caracteres del vector nuevo. Si el vector nuevo contuviese el carácter `a_borrar`, dichas apariciones no se eliminan, tal y como puede apreciarse en el último ejemplo que aparece al final de este enunciado.

La lectura de los caracteres la puede hacer o bien con `cin`, en cuyo caso no se leerán los separadores, o bien con `cin.get()` en cuyo caso sí se leerán. En los siguientes ejemplos, se ha usado `cin` (por lo que los espacios en blanco son omitidos):. La lectura de los dos vectores es muy similar, pero por ahora no tenemos herramientas para no duplicar el código resultante.

Ejemplo de entrada: `u n o a d o s a a # T T U # a`

— Salida correcta: `u n o T T U d o s T T U T T U`

Ejemplo de entrada: `u n o a d o s a a # T a U # a`

— Salida correcta: `u n o T a U d o s T a U T a U`

Finalidad: Trabajar con vectores auxiliares. Dificultad Baja.

8. [Sustituir carácter por vector (versión ineficiente)] Resuelva el ejercicio 7 [Sustituir carácter por vector (con vector auxiliar)] sin utilizar el vector auxiliar sustituido. Por lo tanto habrá que modificar el mismo vector de partida `v` con el resultado de sustituir todas las apariciones del carácter `a_borrar` por el otro vector de caracteres nuevo. Para ello, implemente el siguiente algoritmo:

Algoritmo para sustituir las apariciones de "`a_borrar`" dentro de "`v`", por un vector "`nuevo`"

Recorrer las componentes `i` de `v`

Si `v[i] == a_borrar`, elimina la componente `i`

(desplazando a la izda las componentes que hay a su dcha)

Insertar todas y cada una de las componentes de nuevo

(desplazando a la dcha las componentes que hay

a partir de la posición de inserción)

Se recomienda usar un bucle `while` para recorrer el vector principal `v`, ya que conforme se vayan introduciendo componentes, va cambiando el número de componentes utilizadas de `v`.

Finalidad: Recorrido de un vector eliminando e insertando componentes. Dificultad Media.

9. [Sustituir carácter por vector (versión eficiente)] (*Examen Febrero 2017*) Ejecute el programa pedido en el ejercicio 8 [Sustituir carácter por vector (versión ineficiente)] con el fichero del Quijote que se encuentra en:

http://decsai.ugr.es/jccubero/FP/Quijote_replace_blanco.txt

Este fichero contiene los datos en el orden esperado (`v#nuevo#a_borrar`) para poder redirigir directamente la entrada de datos del programa. El vector `v` contendrá el texto del Quijote. Al ser un vector muy grande, recuerde compilar el programa con las instrucciones detalladas en la página [RP-III.1](#). El vector `nuevo` está formado por tres espacios en blanco y el carácter `a_borrar` es el espacio en blanco. Así pues, lo que pretendemos en este ejemplo, es crear un nuevo texto del Quijote con más espacio entre las palabras (3 espacios en vez de 1). Como queremos leer los espacios en blanco, la lectura de los caracteres debe hacerse con `cin.get()`.

Si ejecuta la solución planteada en el ejercicio 8 [Sustituir carácter por vector (versión ineficiente)] el programa tardará unos *45 minutos* (sobre un i7 con 16Gb de RAM).

Por lo tanto, se pide implementar un algoritmo más eficiente que no necesite realizar tantos desplazamientos de las componentes del vector. Para ello, se recomienda seguir una aproximación distinta. La idea es ir colocando directamente cada componente en el sitio que le corresponde. Por ejemplo, si el vector `v` sólo contuviera una única ocurrencia de `a_borrar` (en la posición `pos_a_borrar`) y el vector `nuevo` contuviese 4 caracteres, bastaría colocar todas las componentes de `v` que hubiese después de `pos_a_borrar`, un total de 3 posiciones por encima. Una vez hecho este desplazamiento, bastaría colocar las 4 componentes de `nuevo`, a partir de `pos_a_borrar`.

En el caso general de que haya más de una ocurrencia de `a_borrar` dentro de `v`, se recomienda que realice un primer recorrido sobre todo el vector `v` para contar el número de ocurrencias de `a_borrar`.

Para implementar esta idea no necesitará dos bucles anidados sino que podrá realizar la tarea pedida con un único bucle. Si lo hace correctamente, el tiempo de ejecución baja drásticamente a unos *0.2 segundos*.

En este ejercicio no puede usar ningún vector auxiliar.

Finalidad: Trabajar eficientemente con vectores. Dificultad Media.

10. [Login] Se está diseñando un sistema web que recolecta datos personales de un usuario y, en un momento dado, debe sugerirle un nombre de usuario (login). Dicho login estará basado en el nombre y los apellidos; en concreto estará formado por los N primeros caracteres de cada nombre y apellido (en minúsculas, unidos y sin espacios en blanco). Por ejemplo, si el nombre es "Antonio Francisco Molina Ortega" y $N = 2$, el nombre de usuario sugerido será "anfrmoor".

Debe tener en cuenta que el número de palabras que forman el nombre y los apellidos puede ser cualquiera. Además, si N es mayor que alguna de las palabras que aparecen en el nombre, se incluirá la palabra completa.

Por ejemplo, si el nombre es "Ana CAMPOS de la Blanca" y $N=4$, entonces la sugerencia será "anacampdelablan" (observe que se pueden utilizar varios espacios en blanco para separar palabras).

Construya un programa que lea caracteres con `cin.get()` hasta llegar al terminador `'>'` y los vaya introduciendo en una variable `a_codificar` de tipo de dato `string`. Para ello debe usar `a_codificar.push_back(caracter)` tal y como se indica en las transparencias de clase. A continuación, el programa leerá el valor de N .

El programa debe construir otra variable codificada de tipo de dato `string` con el login correspondiente e imprimirlo en pantalla (también debe usar `codificada.push_back(caracter)`).

Ejemplo de entrada: Codillo Ificante Carlos>3

— Salida correcta: codificar

Finalidad: Recorrido de las componentes de un string, controlando qué ha ocurrido anteriormente. Dificultad Media.

11. **[Elimina varios]** Se quiere eliminar un conjunto de posiciones `a_borrar` de un vector `v` de enteros. Por ejemplo, si el vector contiene `-3 1 15 6 4 -1 8`, después de eliminar el conjunto de posiciones dado por `2 5 3`, el vector se quedará con `-3 1 4 8`. El conjunto de posiciones puede venir dado en cualquier orden.

Observe que una posibilidad sería sustituir los enteros a borrar por un entero especial, por ejemplo 0 y luego pasarle un algoritmo que eliminase todas las ocurrencias de 0. Sin embargo, debemos evitar esta técnica ya que no podemos presuponer que tenemos la posibilidad de elegir un entero especial en todas las situaciones posibles.

Se recomienda implementar el siguiente algoritmo:

Ordenar `a_borrar` (el vector de posiciones)

Utilizar dos índices: `pos_escritura` y `pos_lectura` que marquen las posiciones de lectura y escritura en el vector `v`

Recorrer con `pos_lectura` los caracteres del vector `v`
Si el carácter actual no está en una posición a borrar,
colocarlo en `pos_escritura`.

Construya un programa principal que lea el número de componentes a rellenar del vector `v` y a continuación los valores de éste. A continuación lea el tamaño del vector `a_borrar` y sus componentes. El programa borrará las componentes indicadas y mostrará el resultado en pantalla.

Ejemplo de entrada: 7 -3 1 15 6 4 -1 8 3 2 5 3

— Salida correcta: -3 1 4 8

Finalidad: Ordenación de un vector y eliminar eficientemente. Dificultad Media.

12. **[Top k (versión ineficiente)]** Se dispone de una serie de datos enteros positivos y se quiere calcular los k mayores, ordenados de mayor a menor. Construya un programa que vaya leyendo datos desde teclado hasta que se introduzca -1. A continuación lea el número k y aplique el siguiente algoritmo:

Vector original: `v`

Vector que contendrá los `k` mayores valores: `topk`

Copiar `v` en `topk`

Ordenar `topk` de MAYOR a MENOR <-- Atención!!!

(se recomienda modificar el algoritmo de ordenación por inserción)

Seleccionar los `k` primeros elementos de `topk`

Finalmente, imprima los `k` primeros valores del vector `topk` en pantalla.

Ejemplo de entrada: 2 0 3 2 12 -1 2

-- Salida correcta: 12 3

Finalidad: Ordenación de un vector y usar vectores auxiliares. Dificultad Baja.

13. [Top k (versión eficiente)] (Examen Febrero 2015) Ejecute la solución del ejercicio 12 [Top k (versión ineficiente)] con el siguiente fichero (con cien mil valores) como entrada de datos. Tenga en cuenta las modificaciones que tiene que hacer en el entorno de compilación para poder trabajar con vectores grandes (ver página RP-III.1 al inicio de esta Relación de Problemas)

http://decsai.ugr.es/jccubero/FP/datos_topk1e5.txt

La salida debe ser:

999954038 999948623 999919447 999912553 999909127

En un ordenador i7 tardará unos 6 segundos en ejecutarse. Si lo hacemos con el siguiente fichero (con un millón de datos):

http://decsai.ugr.es/jccubero/FP/datos_topk1e6.txt

cuya salida debe ser:

999999875 999999069 999997647 999997024 999995114

tardará unos **9 minutos!**.

Por lo tanto, se pide resolver el ejercicio 12 [Top k (versión ineficiente)] usando un algoritmo más eficiente. Para ello, observe que no es necesario ordenar todo el vector. Basta con ordenar al principio `k` valores y posteriormente, ir insertando de forma ordenada el resto de valores del vector.

El nuevo algoritmo ha de ser capaz de ordenar los cien mil datos en medio segundo y tardar no más de cuatro segundos en ordenar el fichero con un millón de datos.

Finalidad: Ordenación de un vector. Dificultad Media.

Problemas sobre matrices

14. [Traspuesta] Lea desde teclado dos variables `util_filas` y `util_columnas` y lea los datos de una matriz de reales de tamaño `util_filas` x `util_columnas`. Sobre dicha matriz, se pide lo siguiente:

Calcular la traspuesta de la matriz, almacenando el resultado en otra matriz y mostrar el resultado.

Ejemplo de entrada:

```
3 4
9   7   4   5
2   18  2   12
7   9   1   5
```

— — Salida correcta:

```
9   2   7
7   18  9
4   2   1
5   12  5
```

Finalidad: Recorrido de las componentes de una matriz. Dificultad Baja.

15. [Producto de matrices] Lea los datos de una matriz de reales tal y como se indica en el ejercicio 14 [Traspuesta] de esta relación de problemas.

Si la matriz que se ha introducido es $n \times m$, por ejemplo, ahora se procederá a leer los datos de una segunda matriz $m \times k$. Por lo tanto, en primer lugar se lee el entero k y a continuación los valores de esta segunda matriz.

Multiplique ambas matrices y muestre el resultado en pantalla.

Ejemplo de entrada:

```
3 4
9   7   4   5
2   18  2   12
7   9   1   5
2
1   2
3   4
5   6
7   8
```

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

— — Salida correcta:

30	46
56	76
34	50

Finalidad: Recorrido de las componentes de una matriz. Dificultad Baja.

16. [Límites de velocidad -en una matriz-] Recupere la solución del ejercicio 34 [Cuadro de límites de velocidad] de la Relación de Problemas II. Modifique el programa para separar E/S y C. Para ello, defina una matriz y guarde los valores de los límites en dicha matriz. Una vez se tenga la matriz de límites, imprímala en pantalla.

Finalidad: Recorrido de las componentes de una matriz. Dificultad Baja.

17. [Máximo de los mínimos] (Examen Septiembre 2011) Lea los datos de una matriz de reales tal y como se indica en el ejercicio 14 [Traspuesta] de esta relación de problemas.

Calcule la posición de aquel elemento que sea el mayor de entre los mínimos de cada fila. Por ejemplo, dada la matriz M (3×4),

9	7	4	5
2	18	2	12
7	9	1	5

el máximo entre 4, 2 y 1 (los mínimos de cada fila) es 4 y se encuentra en la posición (0, 2).

Finalidad: Recorrido de las componentes de una matriz. Dificultad Media.

Ejercicios complementarios

18. [Contiene de forma cíclica] (*Examen Febrero 2013. Doble grado*) Se quiere ver si un vector de caracteres grande contiene todos los caracteres de otro vector peque en el mismo orden (no tienen que estar consecutivos) y de forma cíclica. Para que se cumpla este criterio se debe satisfacer que todos los caracteres de peque estén en grande, en el mismo orden aunque no de forma consecutiva. Además, si durante la búsqueda se ha llegado al final del vector grande, se debe proseguir la búsqueda por el inicio de grande, pero sin sobrepasar la posición en la que hubo la primera concordancia. Por ejemplo, los siguientes vectores contienen a peque = {abcd}

```
x a y o b c p d g
c p d g x a y o b
y o b c p d g x a
```

Hay que destacar que la primera letra de peque a buscar en grande podría estar en cualquier sitio. Por ejemplo, si grande = bcada y peque = abcd, podemos ver que a partir de la primera a de grande no podemos encontrar peque de forma cíclica ya que al ciclar, no podemos seguir buscando más allá de la aparición de la primera a. Sin embargo, si empezamos a buscar a partir de la segunda aparición del carácter a de grande, podemos ver que sí encontramos todos los caracteres.

Finalidad: Recorrido de las componentes de un vector. Dificultad Media.

19. [Series ascendentes] (*Examen Septiembre Doble Grado 2013*) Se quiere calcular el número de series ascendentes de un vector de caracteres. Por ejemplo, el vector ttuvghtwwbde tiene 3 series ascendentes que son ttuv, ghtww, bde.

Construya un programa que lea un conjunto de caracteres con terminador #, calcule e imprima el número de series ascendentes.

Ejemplo de entrada: ttuvghtwwbde# — Salida correcta: 3

Ejemplo de entrada: gfed# — Salida correcta: 4

Ejemplo de entrada: # — Salida correcta: 0

Finalidad: Recorrido sobre un vector procesando dos componentes en cada iteración. Dificultad Media.

20. [Elimina un bloque (versión ineficiente)] Sobre un vector de tipo char, se quiere eliminar todos los caracteres que haya entre dos posiciones. Por ejemplo, después de eliminar el bloque que hay entre las posiciones 1 y 3 del vector {'S', 'o', 'Y', ' ', 'y', 'o'}, ésta debe quedarse con {'S', 'y', 'o'}.

Un primer algoritmo para resolver este problema sería el siguiente (en ejercicios posteriores se verán métodos más eficientes):

Para borrar el bloque entre izda y dcha:

```
Recorrer cada componente -i- del vector
entre las posiciones izda y dcha
  Eliminar dicha componente -i-
```

Para eliminar una posición hay que desplazar hacia la izquierda todas las componentes que hay a su derecha.

Construya un programa que lea caracteres hasta el terminador #. A continuación lea dos enteros que representen las posiciones izquierda (≥ 0) y derecha (entre izquierda y la última componente del vector) e imprima el vector resultante de quitar el bloque de caracteres.

Ejemplo de entrada: abcdefg# 1 3 — Salida correcta: aefg

Ejemplo de entrada: abcdefg# 1 0 -1 15 3 — Salida correcta: aefg

Ejemplo de entrada: abcdefg# 0 5 — Salida correcta: g

Finalidad: Recorrido sencillo de un vector con dos bucles anidados. Dificultad Baja.

21. [Elimina un bloque (versión eficiente)] Resuelva el ejercicio 20 [Elimina un bloque (versión ineficiente)] pero de una forma eficiente. Para ello, implemente el siguiente algoritmo:

Para borrar el bloque entre izda y dcha:

```
Calcular num_a_borrar como dcha - izda + 1

Recorrer las componentes -i- del vector
entre las posiciones dcha+1 hasta el final
  Colocar la componente -i- en la posición
  i - num_a_borrar
```

Este algoritmo resuelve el problema con un único bucle mientras que la versión ineficiente recurría a dos bucles anidados.

Finalidad: Recorrido sencillo de un vector con dos bucles anidados. Dificultad Baja.

22. [k mayores que otro valor] ([Examen Septiembre 2016](#)) Se dispone de un conjunto de reales positivos y se quiere construir otro vector `mayores_que` que contenga los k primeros valores que son mayores o iguales que otro valor de referencia. Los valores que se obtengan como resultado deben estar ordenados de menor a mayor.

Por ejemplo, si el vector original es

{5.2 2.5 7.3 4.2 3.1 4.9}

y el valor de referencia es 4.3, el vector `mayores_que` debe quedar así:

{4.9 5.2 7.3}

Construya un programa que vaya leyendo datos desde teclado e introdúzcalos en un vector. La entrada de datos termina con el -1. A continuación lea el real de referencia y el entero k . Calcule el vector `mayores_que` e imprímalo en pantalla.

Aplique el siguiente algoritmo:

Copiar el vector original en "mayores_que"

Ordenar el vector "mayores_que" de menor a mayor
(utilizar el algoritmo de ordenación por inserción)

Seleccionar los k primeros que sean mayores que la referencia

Utilice también el fichero de datos

http://decsai.ugr.es/jccubero/FP/mayores_que.txt

La salida correcta se encuentra en el fichero

http://decsai.ugr.es/jccubero/FP/mayores_que_solucion.txt

Finalidad: Ordenación de un vector. Dificultad Baja.

23. [**k mayores que otro valor, eficiente**] Modifique la solución del ejercicio 22 usando un algoritmo más eficiente. Observe que no hace falta ordenar todo el vector, sino únicamente considerar los datos que son mayores que la referencia.

Aplique el siguiente algoritmo:

Recorrer las componentes del vector original
Si es mayor que la referencia, insertar dicho
valor de forma ordenada en el vector "mayores_que"

El vector "mayores_que" siempre tendrá,
como mucho, k componentes

Mientras que la versión vista en el ejercicio 22 tardaba varios segundos, esta nueva versión tarda menos de un segundo.

Finalidad: Recorrido sobre un vector, insertando componentes. Dificultad Media.

24. [**Descodifica**] (*Examen Febrero 2016*) Dado un vector de caracteres que contiene un mensaje cifrado, se pide construir otro vector nuevo con el mensaje descifrado. La forma de descifrado consiste en coger la primera y última letra de cada palabra. Las palabras están separadas por uno o más espacios en blanco o el final del vector. Si la última palabra no tiene espacios en blanco a su derecha, se coge sólo el primer carácter.

Por ejemplo, si denotamos el inicio y final de la secuencia con un corchete, entonces:

[Hidrógeno limpia] se descodificaría como [Hola]

Reserve memoria para trabajar con un máximo de 1000 caracteres.

Para leer el mensaje cifrado debe leer caracteres con `cin.get()` (repase lo visto al inicio de esta relación de problemas) hasta que el usuario introduzca el carácter #. A continuación, el programa mostrará la cadena descodificada.

Ejemplo de entrada: [Hidrógeno limpia] — Salida correcta: [Hola]

Ejemplo de entrada: [Hidrógeno limpia] — Salida correcta: [Hol]

Ejemplo de entrada: [Hidrógeno] — Salida correcta: [H]

Ejemplo de entrada: [Hidrógeno] — Salida correcta: [Ho]

Ejemplo de entrada: [H] — Salida correcta: [H]

Ejemplo de entrada: [H] — Salida correcta: [H]

Finalidad: Recorrido de las componentes de un vector, controlando qué ha ocurrido anteriormente. Dificultad Media.

25. [Moda] Se quiere calcular la moda de un vector de caracteres, es decir, el carácter que más veces se repite. Por ejemplo, si el vector fuese

```
{ 'l', 'o', 's', ' ', 'd', 'o', 's', ' ', 'c', 'o', 'f', 'r', 'e', 's' }
```

los caracteres que más se repiten son 'o' y 's' con un total de 3 apariciones. La moda sería cualquiera de ellos, por ejemplo, el primero encontrado 'o'.

Para almacenar de forma conjunta el carácter y la frecuencia usaremos el siguiente struct:

```
struct FrecuenciaCaracter{
    char caracter;
    int  frecuencia;
}
```

El campo `caracter` contendrá el carácter en cuestión ('o') y en el campo `frecuencia` el conteo de la moda (3).

Construya un programa que lea caracteres con `cin.get()` (repase lo visto al inicio de esta relación de problemas) hasta que el usuario introduzca el carácter #. Almacene todos los valores en un vector de caracteres. A continuación, calcule la moda y muéstrela en pantalla junto con su frecuencia.

Para calcular la moda, se recomienda que use un vector auxiliar en el que almacene los caracteres que ya se han procesado en algún momento anterior.

Utilice el texto del Quijote sin espacios en blanco disponible en:

http://decsai.ugr.es/jccubero/FP/Quijote_sin_espacios.txt

La moda es la letra 'e' con un total de 217900 apariciones.

Finalidad: Recorridos sobre un vector. Dificultad Media.

26. [Cuenta Mayúsculas] Construya un programa que vaya leyendo caracteres hasta que se encuentre un punto '.' y cuente el número de veces que aparece cada una de las letras mayúsculas. Imprimir el resultado.

Una posibilidad sería declarar un vector `contador_mayusculas` con tantas componentes como letras mayúsculas hay ('Z' - 'A' + 1) y conforme se va leyendo cada carácter, ejecutar lo siguiente:

```
cin >> letra;

if (letra == 'A')
    contador_mayusculas[0] = contador_mayusculas[0] + 1;
else if (letra == 'B')
    contador_mayusculas[1] = contador_mayusculas[1] + 1;
else if (letra == 'C')
    contador_mayusculas[2] = contador_mayusculas[2] + 1;
else ....
```

Sin embargo, este código es muy redundante. Como solución se propone calcular de forma directa el índice entero que le corresponde a cada mayúscula, de forma que todos los if-else anteriores los podamos resumir en una **única** sentencia del tipo:

```
contador_mayusculas[indice] = contador_mayusculas[indice] + 1;
```

Hacedlo, declarando el vector directamente dentro del `main`.

Finalidad: Acceder a las componentes de un vector con unos índices que representen algo. Dificultad Baja.

27. [Frecuencias vocales] (*Examen Septiembre 2016*) Queremos calcular la frecuencia absoluta de las vocales (sin acentuar) presentes en un texto y mostrar el resultado en forma de un diagrama de barras en el que cada barra corresponde a una vocal y la altura representa la frecuencia de aparición de la correspondiente vocal.

Un ejemplo de diagrama de barras sería el siguiente:

```
6          *
5      *   *
4      * * *
3 * * * * *
2 * * * * *
1 * * * * *
   a e i o u
```

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

Cada aparición de una vocal la representaremos con un asterisco *. Por ejemplo, la barra correspondiente a la vocal i tiene un total de tres asteriscos ya que dicha vocal ha aparecido tres veces en el texto.

Defina una matriz de caracteres para representar las barras. Se emplea el carácter '*' para indicar que la casilla está ocupada y el carácter ' ' (espacio en blanco) para indicar que la casilla está libre. En la matriz no almacenaremos ni los números 1, 2, 3, 4, ... ni las vocales a, e, i, o, u.

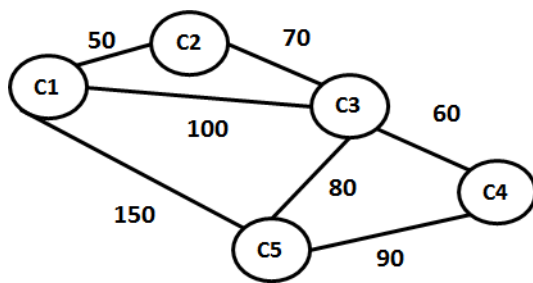
Escriba un programa que lea una serie indefinida de caracteres de la entrada estándar (terminados en #) y muestre la figura (las barras, los conteos y las vocales) que representa la frecuencia absoluta de las vocales introducidas.

Por ejemplo, la anterior figura sería el diagrama de barras de la siguiente entrada:

```
You should solve this typical exam problem
in no more than 30 minutes#
```

Finalidad: Construir índices y trabajar con las componentes de una matriz. Dificultad Media.

28. [Distancias entre ciudades] Se desea construir una matriz de reales para almacenar las distancias de los caminos directos que conectan un conjunto de ciudades. Si entre dos ciudades no existe un camino directo, se almacenará un cero. Se supone que la distancia de una ciudad consigo misma será cero y que las distancias son simétricas. Un ejemplo con 5 ciudades sería:



	C1	C2	C3	C4	C5
C1	0	50	100	0	150
C2	50	0	70	0	0
C3	100	70	0	60	80
C4	0	0	60	0	90
C5	150	0	80	90	0

Suponga que nunca se trabajará con más de 50 ciudades.

Construya un programa que lea el número de filas y columnas y a continuación lea los datos de la diagonal superior. Según el ejemplo anterior, los datos a introducir serían:

50, 100, 0, 150, 70, 0, 0, 60, 80, 90

Almacene dichos datos en una matriz cuadrada, poniendo un 0 en la diagonal principal y almacenando los simétricos correspondientes en la diagonal inferior. Asumimos que esta aproximación no es óptima ya que la matriz es simétrica y por tanto la mitad de los datos están duplicados.

Se quiere resolver las siguientes tareas, imprimiendo el resultado en pantalla:

- a) Obtener la ciudad (su índice) con mayor número de conexiones directas. En el ejemplo, sería la ciudad 3 con 4 conexiones.
- b) Dada una ciudad j , obtener las ciudades conectadas directamente con j . En el ejemplo, si consideramos la ciudad 4, vemos que está conectada directamente con las ciudades 3 y 5.
- c) Dadas dos ciudades i y j para las cuales no existe un camino directo, obtener aquella ciudad intermedia z que permita hacer el trayecto entre i y j de la forma más económica posible. Es decir, se trata de encontrar una ciudad z tal que $d(i, z) + d(z, j)$ sea mínima ($d(a, b)$ es la distancia entre las ciudades a y b).

Si no existe dicha ciudad se imprimirá el valor -1.

Por ejemplo, si se desea viajar desde la ciudad 2 a la 5, hacerlo a través de la ciudad 1 tiene un costo de $50 + 150 = 200$ mientras que si se hace a través de la ciudad 3, el costo sería $70 + 80 = 150$.

Nota: Para la inicialización del mínimo, puede usar la constante `INFINITY`, que garantiza que cualquier dato de tipo `double` es menor que ella.

- d) Lea ahora una serie de enteros con terminador -1 que serán índices de ciudades. El programa debe decir si todas las ciudades correspondientes a esos índices tienen conexiones directas entre sí.

29. **[Maximin]** Lea los datos de una matriz de reales tal y como se indica en el ejercicio 14 **[Traspuesta]** de esta relación de problemas.

Vea si existe un valor *MaxiMin*, es decir, que sea a la vez, máximo de su fila y mínimo de su columna.

Finalidad: Recorrido de las componentes de una matriz. Dificultad Media.

30. **[Matriz promedio]** Defina dos matrices de reales *original* y *suavizada* de tamaño 50×30 . Lea desde teclado los valores de la matriz *original*, obligando a que sea simétrica. Para ello, lea el número de filas n y a continuación introduzca los $n \times n$ datos de la matriz.

Construya la matriz *promedio* acorde a las siguientes indicaciones:

- a) La tabla resultante será **simétrica**.
- b) Los valores de la **diagonal principal** de la tabla resultante serán iguales a los de la tabla original.
- c) Los valores del **triángulo superior** de la tabla resultante se calculan de la siguiente manera: si (i, j) es una posición en el triángulo superior de la tabla resultante, su valor es la media aritmética de los valores que ocupan las posiciones de las columnas $j, j + 1, \dots, n - 1$ en la fila i de la tabla original.

Ejemplo de entrada:

```
4
9   7   4   5
2   18  2  12
7   9   1   5
0   1   2   3
```

— — Salida correcta:

```
9          5.33333 4.5      5
5.33333 18          7      12
4.5       7          1      5
5         12         5      3
```

Finalidad: Recorrido de las componentes de una matriz. Dificultad Baja.

31. Para ahorrar espacio en el almacenamiento de matrices cuadradas simétricas de tamaño $k \times k$ se puede usar un vector con los valores de la diagonal principal y los que están por debajo de ella. Por ejemplo, para una matriz $M = \{m_{ij}\}$ el vector correspondiente sería:

$$\{m_{11}, m_{21}, m_{22}, m_{31}, m_{32}, m_{33}, m_{41}, \dots, m_{kk}\}$$

Declarar una matriz clásica `double matriz[50][50]` en el `main`, asignarle valores de forma que sea cuadrada simétrica y construir el vector pedido. Haced lo mismo pero a la inversa, es decir, construir la matriz a partir del vector.

Finalidad: Recorrido de las componentes de una matriz. Dificultad Media.

32. Escribir un programa que permita a dos jugadores jugar al tres-en- raya. El programa preguntará por los movimientos alternativamente al jugador X y al jugador O. El programa mostrará las posiciones del juego como sigue:

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

Los jugadores introducen sus movimientos insertando los números de posición que desean marcar. Después de cada movimiento, el programa mostrará el tablero cambiado. Un tablero de ejemplo se muestra a continuación.

```
X  X  O
4  5  6
O  8  9
```

El programa detectará al final de la partida si hay o no empate y en caso contrario, qué jugador ha ganado. Además, pedirá empezar una nueva partida y reiniciar el proceso.

RELACIÓN DE PROBLEMAS III. Vectores y Matrices

Finalidad: Practicar con el uso de matrices sencillas en una aplicación. Dificultad Media.

33. Escribir un programa para asignar asientos de pasajeros en un avión. Asumimos un avión pequeño con la numeración de asientos como sigue:

1	A	B	C	D
2	A	B	C	D
3	A	B	C	D
4	A	B	C	D
5	A	B	C	D
6	A	B	C	D
7	A	B	C	D

El programa mostrará con una X el asiento que está ya asignado. Por ejemplo, después de asignar los asientos 1A, 2B, y 4C, lo que se mostrará en pantalla tendrá un aspecto como este:

1	X	B	C	D
2	A	X	C	D
3	A	B	C	D
4	A	B	X	D
5	A	B	C	D
6	A	B	C	D
7	A	B	C	D

Después de mostrar los asientos disponibles, el programa pregunta por el asiento deseado, el usuario teclea un asiento y el programa actualiza la asignación mostrando el esquema anterior. Primero se pide el número de fila y después la letra de asiento. Esto continua hasta que todos los asientos se asignen o hasta que el usuario indique que no quiere asignar más asientos (introduciendo el valor -1 en el número de la fila). Si el usuario introduce un asiento ya asignado, el programa mostrará un mensaje indicando que el asiento está ocupado y volverá a solicitarlo.

Finalidad: Practicar con el uso de matrices sencillas en una aplicación. Dificultad Baja.

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

Problemas sobre funciones

1. [Errores en funciones] Encuentre los errores de las siguientes funciones:

```
int ValorAbsoluto (int entero) {    void Imprime(double valor) {
    if (entero < 0)                  double valor;
        entero = -entero;
    else                            cout << valor;
        return entero;              }
}

void Cuadrado (int entero) {        bool EsPositivo(int valor) {
    return entero*entero;            if (valor > 0)
}                                    return true;
                                    }
}
```

Finalidad: Familiarizarnos con la definición de funciones, el paso de parámetros y el ámbito de las variables. Dificultad Baja.

2. [Lee entero en rango] Defina una función `LeeIntRango` para leer un número entero obligando a que esté en un rango `[min, max]`. Para ello, dicha función debe ir leyendo números enteros (de tipo `int`) desde la entrada por defecto, hasta que se lea un valor que pertenezca al rango `[min, max]` (no hay ningún límite en el número de intentos). La función devolverá dicho valor.

Escriba un pequeño programa de prueba que lea dos números `min` y `max`. El valor de `min` puede ser cualquiera y supondremos que el valor introducido de `max` es correcto, es decir, que será mayor o igual que `min`. A continuación llame a la función `LeeIntRango` tres veces para leer tres valores en el rango `[min,max]` y finalmente calcule la suma de dichos valores.

Ejemplo de entrada: 3 6 -1 2 3 7 2 1 4 4 — Salida correcta: 11

Ejemplo de entrada: 3 6 3 4 4 — Salida correcta: 11

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

3. [Lee entero mayor o igual que otro] Defina una función `LeeIntMayorIgualQue` para leer un entero mayor o igual que un número dado (éste será un parámetro a la función). Para ello, dicha función debe ir leyendo números enteros (de tipo `int`) desde la

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

entrada por defecto, hasta que se lea un valor correcto que sea mayor o igual que el número especificado. La función devolverá dicho valor.

Añada esta función a la solución del ejercicio 2 [Lee entero en rango] y utilícela para leer el valor de `max`, obligando a que sea mayor que `min`.

Ejemplo de entrada: 3 2 1 6 -1 2 3 7 2 1 4 4

— Salida correcta: 11

Ejemplo de entrada: 3 6 3 4 4

— Salida correcta: 11

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

4. [Máximo de 3 valores] En las transparencias de clase se ha visto la función `Max3` que calculaba el máximo de tres valores enteros.

Defina ahora la función `Max` que calcule el máximo de dos valores enteros y cambie la implementación de `Max3` para que llame a la función `Max`.

Construya un programa principal que llame a `Max3` con tres valores leídos desde teclado.

Finalidad: Familiarizarnos con las llamadas entre funciones. Dificultad Baja.

5. [Potencia entera] Defina una función `PotenciaEntera` para elevar un número real a otro entero. En la implementación, no puede usar la función `pow` de `cmath`. Escriba un programa simple de prueba.

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

6. [Diferencia entre instantes] Cree un programa que lea un número de horas (entre 0 y 23), minutos (entre 0 y 59) y segundos (entre 0 y 59) iniciales/finales y calcule e imprima en pantalla el número de minutos y segundos entre ambos instantes.

Debe definir todas las funciones que estime oportuno para no repetir código.

Puede re-utilizar la solución del ejercicio 8 [Segundos entre dos instantes] de la Relación de Problemas I.

Ejemplo de entrada: 10 20 5 11 22 6

— Salida correcta: 3721 62

Finalidad: Familiarizarnos con las llamadas entre funciones. Dificultad Baja.

7. [Decimal redondeado] Recupere la solución del ejercicio 9 [Decimal redondeado] de la Relación de Problemas I. Defina una función `Redondea` para calcular el real aproximado a un número de cifras decimales. En la implementación de esta función, necesita calcular una potencia entera. Para ello, llame a la función `PotenciaEntera` definida en el ejercicio 5 [Potencia entera]

Finalidad: Familiarizarnos con las llamadas entre funciones. Dificultad Baja.

8. **[Elimina últimos]** Defina una función `EliminaUltimos` que elimine todos los caracteres de un `string` que sean igual a un carácter determinado. Para eliminar el último carácter de un dato cadena de tipo `string`, basta ejecutar la sentencia `cadena.pop_back()` ;

Construya un programa que vaya leyendo caracteres con `cin.get()` hasta llegar al terminador `#` y los vaya introduciendo en una variable `cadena` de tipo `string`. A continuación, el programa leerá un carácter y llamará a la función `EliminaUltimos` para eliminar todos los caracteres al final de `cadena` que sean igual al carácter. Imprima la cadena resultante.

Ejemplo de entrada: `TabcTdTTTT#T`

— Salida correcta: `TabcTd`

Ejemplo de entrada: `TabcTd#T`

— Salida correcta: `TabcTd`

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

9. **[double to string]** La función `to_string` de C++11 es muy útil ya que permite convertir un real en una cadena de caracteres. Por ejemplo, `to_string(23.51)` devuelve la cadena `"23.510000"`. Queremos definir una función `DoubleToString` que redondee un número real a un número de cifras determinado y devuelva una cadena de caracteres con la representación correspondiente pero sin los ceros finales.

Para ello, debe llamar a las funciones `Redondea` del ejercicio 7 **[Decimal redondeado]** y `EliminaUltimos` del ejercicio 8 **[Elimina últimos]** .

Construya un programa que lea un real, un número de cifras decimales e imprima la cadena resultante de llamar a la función `DoubleToString`

Ejemplo de entrada: `3.49 1` — Salida correcta: `"3.5"`

Ejemplo de entrada: `3.49 2` — Salida correcta: `"3.49"`

Ejemplo de entrada: `3.496 2` — Salida correcta: `"3.5"`

Ejemplo de entrada: `0.12345678 7` — Salida correcta: `"0.123457"`

Finalidad: Familiarizarnos con las llamadas entre funciones. Dificultad Baja.

10. **[double to string mejorado]** En el ejercicio 9 **[double to string]** hemos usado la función `to_string` para convertir un `string` a un `double`. Un inconveniente de esta función es que siempre redondea a 6 cifras decimales. Por ejemplo, el resultado de `to_string(0.12345678)` es la cadena `"0.123457"`.

Modifique la función `DoubleToString` del ejercicio 9 **[double to string]** para que redondee un real a un número de cifras arbitrario y obtenga la cadena de caracteres correspondiente.

Escriba un programa de prueba como el indicado en el ejercicio 9 **[double to string]**

Ejemplo de entrada: `3.49 1` — Salida correcta: `"3.5"`

Ejemplo de entrada: `3.49 2` — Salida correcta: `"3.49"`

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

Ejemplo de entrada: 3.496 2 — Salida correcta: "3.5"

Ejemplo de entrada: 0.12345678 7 — Salida correcta: "0.1234568"

Finalidad: Familiarizarnos con las llamadas entre funciones. Dificultad Media.

Problemas sobre Clases

11. [Recta] En este ejercicio se plantean varias modificaciones. Debe entregar un fichero `cpp` por cada uno de los apartados.

Se desea implementar una clase `Recta` para representar una recta en el plano. Una recta viene determinada por tres coeficientes `A`, `B`, `C`, de forma que todos los puntos (x, y) que pertenecen a la recta verifican lo siguiente (*ecuación general de la recta*):

$$Ax + By + C = 0$$

a) *Definición de la clase y creación de objetos*

Defina la clase `Recta`. En este apartado utilice únicamente datos miembro públicos. Cree un programa principal que haga lo siguiente:

- Defina dos objetos de la clase `Recta`.
- Lea seis reales desde teclado.
- Le asigne los tres primeros a los coeficientes de una recta y los otros tres a la segunda recta.
- Calcule e imprima la pendiente de cada recta aplicando la fórmula:

$$\text{pendiente} = -A / B$$

b) *Métodos públicos*

En vez de calcular la pendiente en el programa principal, vamos a ponerlo como un método de la clase y así lo reutilizaremos todas las veces que necesitemos. Añada un método para el cálculo de la pendiente y modifique el `main` para tener en cuenta este cambio.

¿Añadimos `pendiente` como dato miembro de la recta? La respuesta es que no ¿Por qué?

Añada también los siguiente métodos:

- Obtener la ordenada (`y`) dado un valor de abscisa `x`, aplicando la fórmula:
$$(-C - xA) / B$$
- Obtener la abscisa (`x`) dado un valor de ordenada `y`, aplicando la fórmula:
$$(-C - yB) / A$$

En la función `main` lea un valor de abscisa e imprima la ordenada según la recta. A continuación lea un valor de ordenada e imprima la abscisa que le corresponde. Hágalo sólo con la primera recta.

c) *Datos miembro privados*

Cambie ahora los datos miembro públicos y póngalos privados. Tendrá que añadir métodos para asignar y ver los valores de los datos miembro. Añada métodos

para asignar un valor a cada uno de los tres datos miembro. Modifique el `main` para tener en cuenta estos cambios.

A partir de ahora, todos los ejercicios deben resolverse utilizando únicamente datos miembro privados.

IMPORTANT

d) *Política de acceso a los datos miembros*

En vez de usar un método para asignar un valor a cada dato miembro, defina un único método `SetCoeficientes` para asignar los tres a la misma vez.

Observe que los métodos permiten definir la política de acceso a los datos miembro. Si tengo previsto cambiar por separado los coeficientes de la recta, usaré métodos de asignación individuales. En caso contrario, usaré un único método que modifique a la misma vez todos los datos miembro. Incluso pueden dejarse en la clase ambos tipos de métodos para que así el cliente de la clase pueda usar los que estime oportunos en cada momento. Por ahora, mantenga únicamente el método de asignación *en bloque* `SetCoeficientes`.

e) *Constructor*

Modifique el programa principal del último apartado e imprima los valores de los datos miembros de una recta, **antes** de asignarles los coeficientes. Mostrará, obviamente, un valor indeterminado. Para evitar este problema, añada un constructor a la recta para que el objeto esté en un estado válido en el mismo momento de su definición. El constructor deberá tener como parámetros, obligatoriamente, los tres coeficientes de la recta. Tendrá que modificar convenientemente el `main` para tener en cuenta este cambio.

f) *Política de acceso a los datos miembro*

Suprima ahora el método `SetCoeficientes`. De esta forma, una vez creado el objeto (pasándole los datos apropiados en el constructor) ya no podremos modificar los datos miembro. Esto es útil en aquellas situaciones en las que no queremos permitir que el estado del objeto cambie, una vez se ha creado.

g) *Métodos privados*

Vuelva a recuperar el método `SetCoeficientes`. Añada un método privado que nos indique si los coeficientes son correctos, es decir, A y B no pueden ser simultáneamente nulos. Llame a este método donde sea necesario.

Finalidad: Familiarizarnos con la definición de clases. Dificultad Baja.

12. [Cronómetro] Considere la siguiente definición de la clase `Cronometro`:

```
#include <chrono>

class Cronometro{
private:
```

```
typedef std::chrono::time_point<std::chrono::steady_clock>
    Punto_en_el_Tiempo;
typedef chrono::duration <long long, nano>    IntervaloTiempo;

Punto_en_el_Tiempo inicio;
Punto_en_el_Tiempo final;

public:

    void Reset(){
        inicio = chrono::steady_clock::now();
    }
    long long NanoSegundosTranscurridos(){
        final = chrono::steady_clock::now();
        IntervaloTiempo diferencia = final - inicio;

        return diferencia.count();
    }
};
```

Esta clase sirve para medir el tiempo de ejecución de un conjunto de instrucciones. Para usarla, haga copy-paste en su código. No olvide insertar también los `#include` que aparecen al inicio.

No hace falta que entienda el código de la clase sino únicamente cómo utilizar sus métodos públicos. Para ello, basta crear un objeto de esta clase y justo antes del conjunto de instrucciones que queramos cronometrar, debemos ejecutar el método `Reset`. Justo después de las instrucciones, llamaremos al método `NanoSegundosTranscurridos` para saber el número de nanosegundos transcurridos. El cronómetro seguirá en marcha (por lo que podremos llamar al método `NanoSegundosTranscurridos` tantas veces como queramos) hasta que se reseedee de nuevo con el método `Reset`.

Se pide añadir un nuevo método `MiliSegundosTranscurridos` para saber cuántos milisegundos han transcurrido. Este método debe llamar al anterior.

Recupere alguna de las soluciones de los ejercicios de la Relación de Problemas III y mida los tiempos de ejecución.

Finalidad: Enfatizar la importancia de los constructores, la ocultación de información y de la interfaz pública de una clase. Dificultad Baja.

13. [Generador aleatorio] Considere la siguiente definición de la clase `GeneradorAleatorioEnteros`:

```
#include <random>    // -> generación de números pseudoaleatorios
#include <chrono>    // -> para la semilla

class GeneradorAleatorioEnteros{
private:
    mt19937 generador_mersenne;    // Mersenne twister
    uniform_int_distribution<int> distribucion_uniforme;

    long long Nanosec(){
        return chrono::high_resolution_clock::now().
            time_since_epoch().count();
    }
public:
    GeneradorAleatorioEnteros()
        :GeneradorAleatorioEnteros(0, 1){
    }

    GeneradorAleatorioEnteros(int min, int max){
        const int A_DESCARTAR = 70000;
        // ACM TOMS Volume 32 Issue 1, March 2006

        auto semilla = Nanosec();
        generador_mersenne.seed(semilla);
        generador_mersenne.discard(A_DESCARTAR);
        distribucion_uniforme =
            uniform_int_distribution<int> (min, max);
    }

    int Siguiente(){
        return distribucion_uniforme(generador_mersenne);
    }
};
```

Esta clase sirve para generar números aleatorios enteros en un rango de valores. Para usarla, haga copy-paste en su código. No olvide insertar también los `#include` que aparecen al inicio.

El objetivo de este ejercicio es que aprenda a utilizar sus métodos sin que necesite entender cómo están implementados.

Esta clase tiene dos constructores. Uno con dos parámetros, `min` y `max` que delimitan el rango correspondiente. El otro constructor no tiene parámetros y establece que

únicamente se van a generar ceros y unos (este segundo constructor llama al anterior) Cada vez que se llame al método `Siguiente`, éste devolverá un valor aleatorio en el rango especificado.

Utilice dicha clase para crear varios números aleatorios de la siguiente forma. En primer lugar debe generar un número aleatorio entre 1 y 5 (llámelo `num_0_1_a_generar`) y a continuación debe generar tantos números aleatorios entre 0 y 1 como indique `num_0_1_a_generar`. Haga lo anterior 100 veces. Por ejemplo, una posible salida sería:

```
01
00101
1
100
.....
```

Nota: Realmente, los números generados son *pseudoaleatorios* (puede consultar Internet para tener una idea de este concepto) En este ejemplo se han generado según una distribución uniforme pero se pueden generar números con las probabilidades dadas por muchas otras distribuciones -disponibles en la biblioteca `random` de C++11-.

Finalidad: Enfatizar la importancia de los constructores, la ocultación de información y de la interfaz pública de una clase. Dificultad Baja.

14. **[Coordenadas geográficas]** Se quiere construir una clase `CoordenadasGPS` para representar las coordenadas geográficas de una posición terrestre, dada por tres datos reales, a saber, su longitud, latitud y altura, tal y como se describe en el ejercicio 5 **[Coordenadas geográficas (distancia)]** de la Relación de Problemas II. La clase debe proporcionar métodos para asignar los datos de latitud y longitud en grados, aunque dentro de la clase se trabajará en radianes. La altura vendrá en metros.

Debe comprobar que los grados sean correctos, es decir, los grados de latitud deben estar en el intervalo $[-90, 90]$ y los de longitud en $[-180, 180]$. La altura debe estar entre -423 (Valle del Jordán) y 8848 (monte Everest).

Por ahora, la clase sólo debe proporcionar los métodos para asignar y recuperar los valores de latitud, longitud y altura. Posteriormente se verán ejercicios para calcular la distancia entre dos coordenadas.

Finalidad: Diseño de una clase básica. Dificultad Baja.

15. **[Dinero]** Defina una clase `Dinero` para poder trabajar de forma precisa con datos monetarios. La clase tendrá dos datos miembro, euros y centimos y debe permitir que se introduzca un número de céntimos mayor de 100. Por ejemplo, si asignamos 20 euros y 115 céntimos, el objeto debe almacenar 21 en euros y 15 en centimos. En el último tema veremos cómo sumar o restar dos objetos de la clase `Dinero`.

Construya un programa que lea dos cantidades de dinero (euros y céntimos de una y euros y céntimos de la otra) y construya dos objetos de la clase `Dinero`. A continuación, el programa debe sumar las cantidades correspondientes y construir un tercer objeto con dicha suma. Finalmente, el programa debe imprimir en pantalla el número de euros y céntimos resultantes.

Ejemplo de entrada: 20 90 30 115

— Salida correcta: 52 5

Finalidad: Trabajar con una clase como una abstracción de un concepto. Dificultad Baja.

16. **[Instante]** Defina la clase `Instante` para representar un instante de tiempo dentro de un día. Por lo tanto, la clase debe representar un número de horas, minutos y segundos. Añádale métodos para:

- Obtener el número de segundos y minutos transcurridos desde las 0h 0min 0seg. Por ejemplo, si el instante es 1h 2min 1seg, el número total de segundos transcurridos es de 3721 y el de minutos 62.
- Establecer el instante a partir del número de segundos transcurridos. Por ejemplo, si se establece a 3721, el instante debe contener 1h 2min 1seg.
- Puede definir todos los métodos adicionales que estime oportuno.

Cree un programa que lea un número de horas (entre 0 y 23), minutos (entre 0 y 59) y segundos (entre 0 y 59) iniciales/finales. El programa creará los objetos `instante_inial` e `instante_final` y calculará el número de segundos que hay de diferencia entre ambos instantes. A continuación creará un objeto `instante_diferencia` correspondiente a la cantidad de segundos de diferencia e imprimirá en pantalla el valor de hora, minuto y segundo que le corresponde. También imprimirá el total de segundos y minutos enteros de `instante_diferencia`.

Puede re-utilizar la solución del ejercicio 8 **[Segundos entre dos instantes]** de la Relación de Problemas I.

Ejemplo de entrada: 2 10 4 3 12 5

— Salida correcta:

Diferencia: 1h 2' 1''

Total segundos diferencia: 3721

Total minutos diferencia: 62

Finalidad: Diseño de una clase. Dificultad Baja.

17. **[Formateador de doubles]** Se quiere construir una clase para poder convertir una dato de tipo `double` a `string` tal y como se hizo con funciones en el ejercicio 9 **[double to string]** de esta Relación de Problemas.

Dado un dato de tipo `double`, queremos *formatearlo* adecuadamente, especificando lo siguiente:

- Un delimitador a la izquierda y otro a la derecha.
- Si el separador es el punto o la coma.
- Número de decimales a obtener.

Por ejemplo, si el delimitador izquierda se fija a la cadena "<<", el derecha a ">>", el separador de decimales a la coma y el número de decimales a 7, el resultado de formatear el número 0.12345678 sería "<<0,1234568>>"

En el diseño de esta clase debe decidir qué datos miembro va a definir, si alguno de éstos tiene un valor por defecto, cómo especificar los delimitadores (izquierda y derecha) y el separador (punto o coma), cuántos constructores va a necesitar, etc.

Puede trabajar sobre la solución del ejercicio 9 [double to string] disponible en http://decsai.ugr.es/jccubero/FP/IV_double_to_string.cpp

Finalidad: Diseño de una clase. Dificultad Media.

18. [Calificación final] Defina la clase `CalificacionFinal` para calcular la nota en la convocatoria ordinaria de Febrero de un alumno en la asignatura de Fundamentos de Programación. Para ello, debe considerar lo siguiente:

- Cada alumno es calificado con 4 notas (especificadas de 0 a 10): evaluación continua, dos exámenes prácticos y un examen escrito. Por defecto, la ponderación de cada parte en el cómputo de la nota final es 10 %, 10 %, 20 % y 60 % respectivamente.

Estos porcentajes son los considerados por defecto. En cualquier caso, se quiere contemplar la posibilidad de manejar otros distintos.

- El profesor de cada grupo, tiene la posibilidad, si así lo desea, de subir la nota del examen escrito. Dicha subida no puede ser mayor de 0.5 puntos y puede variar entre grupos distintos, aunque, obviamente, será la misma para todos los alumnos de un mismo grupo. En cualquier caso, la subida sobre el examen escrito sólo se aplica a aquellos alumnos que, después de realizar la subida, saquen un 5 o más.

- Para poder aprobar la asignatura, es preciso haber sacado al menos un 4 en la nota del examen escrito (esta restricción se aplica antes de la subida de nota especificada en el apartado anterior).

Si el alumno no supera la nota mínima de 4 en el examen escrito, la nota final será la nota del examen escrito.

El 4 es el límite por defecto. Al igual que los porcentajes del primer apartado, se quiere contemplar la posibilidad de manejar otros distintos.

Construya un programa principal que lea los datos en el siguiente orden (no hay límite en el número de grupos que se van a introducir):

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

```
0.5          -> Subida lineal del grupo 1
2.5 3.5 7.5 4.4 -> Notas del alumno 1 del grupo 1
6.4 9.5 8.5 7.2 -> Notas del alumno 2 del grupo 1
.....
-1           -> Fin de datos del grupo 1
0.3          -> Subida lineal del grupo 2
3.5 6.4 5.5 6.4 -> Notas del alumno 1 del grupo 2
1.4 2.5 3.4 1.3 -> Notas del alumno 2 del grupo 2
.....
-1           -> Fin de datos del grupo 2
-1           -> Fin de datos
```

El programa debe imprimir la nota final de cada alumno utilizando dos posibilidades:

- a) Suponiendo que las ponderaciones son 10 %, 10 %, 20 % y 60 % y la nota mínima para aprobar un 4
- b) Suponiendo que las ponderaciones son 5 %, 5 %, 20 % y 70 % y la nota mínima para aprobar un 4.4

En la dirección

http://decsai.ugr.es/jccubero/FP/notas_parciales.txt

se encuentra un fichero de prueba para este ejercicio. Las notas finales que el programa debería obtener se encuentran en:

http://decsai.ugr.es/jccubero/FP/notas_finales.txt

Ejemplo de entrada:

```
0.5
5.7  4.9  5.2  5.5
6.3  7.1  5.1  8.1
9.5  9.8  8    9.3
.....
-1
0.4
4.2  4.7  4.2  4.9
4    4.5  4.8  4.3
6.8  7.3  5.5  5.7
.....
-1
-1
```

— Salida correcta:

```
9.41/9.425
1.4/1.4
```

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

6.28/6.27

.....

4.91/4.995

4.39/4.3

6.17/6.075

.....

Finalidad: Diseño de una clase. Dificultad Media.

Problemas sobre vectores dentro de clases

En los ejercicios que pida trabajar sobre la clase `SecuenciaCaracteres`, use la siguiente definición:

SecuenciaCaracteres	
- const int	TAMANIO
- char	vector_privado[TAMANIO]
- int	total_utilizados
- void	IntercambiaComponentes(int pos_izda, int pos_dcha)
+	SecuenciaCaracteres()
+ int	TotalUtilizados()
+ int	Capacidad()
+ void	Añade(char nuevo)
+ void	Modifica(int pos_a_modificar, char valor_nuevo)
+ char	Elemento(int indice)
+ void	EliminaTodos()
+ int	PrimeraOcurrenciaEntre(int pos_izda, int pos_dcha, char buscado)
+ int	PrimeraOcurrencia(char buscado)
+ int	BusquedaBinaria(char buscado)
+ int	PosicionMinimoEntre(int izda, int dcha)
+ int	PosicionMinimo()
+ void	Inserta(int pos_insercion, char valor_nuevo)
+ void	Elimina(int pos_a_eliminar)
+ void	Ordena_por_Seleccion()
+ void	Ordena_por_Insercion()
+ void	Ordena_por_Burbuja()
+ void	Ordena_por_BurbujaMejorado()
+ string	ToString()

Puede encontrar el código en la dirección siguiente:

<http://decsai.ugr.es/jccubero/FP/SecuenciaCaracteres.cpp>

19. [Palíndromo e invierte dentro de la clase `SecuenciaCaracteres`] Trabaje sobre la clase `SecuenciaCaracteres` tal y como viene descrita arriba. Añádale los siguientes métodos (descritos en el ejercicio 1 [Palíndromo e Invierte] de la Relación de Problemas III):

- `EsPalindromo` para comprobar si la secuencia es o no un palíndromo.
- Se desea poner público el método `IntercambiaComponentes`. ¿Qué habría que cambiar?

- Invierte para invertir la secuencia. Este método debe llamar a `IntercambiaComponentes`

Incluya un programa principal de prueba similar al del ejercicio 1 [Palíndromo e Invierte] de la Relación de Problemas III, de forma que los caracteres que se vayan leyendo se vayan añadiendo al objeto de la clase `SecuenciaCaracteres`.

Finalidad: Trabajar con un vector dentro de una clase. Dificultad Baja.

20. [Elimina ocurrencias ineficiente] Sobre la clase `SecuenciaCaracteres`, añada el método `EliminaOcurrencias` para eliminar todas las apariciones de un determinado carácter `a_borrar`. Por ejemplo, después de eliminar el carácter 'o' de la secuencia {'S','o','Y',' ','y','o'}, ésta debe quedarse con {'S','Y',' ','y'}.

Resuelva este problema con el algoritmo ineficiente descrito en el ejercicio 5 [Elimina ocurrencias de una componente -versión ineficiente-] de la Relación de Problemas III. Recordemos que dicho algoritmo era el siguiente:

Recorrer todas las componentes del vector

Si la componente es igual al carácter `a_borrar`, eliminarla
(desplazando hacia la izda las componentes que hay a su dcha)

Una posible implementación del anterior algoritmo en un programa que trabaja directamente con un vector declarado dentro del `main` se encuentra en el siguiente enlace:

http://decsai.ugr.es/jccubero/FP/III_elimina_ocurrencias_ineficiente.cpp

Dentro del método `EliminaOcurrencias` debe llamar dentro de un bucle al método `Elimina` (que borra un único carácter). La implementación del método `Elimina` se encuentra en

<http://decsai.ugr.es/jccubero/FP/SecuenciaCaracteres.cpp>

Incluya un programa principal de prueba similar al del ejercicio 5 [Elimina ocurrencias de una componente -versión ineficiente-] de la Relación de Problemas III, de forma que los caracteres que se vayan leyendo se vayan añadiendo al objeto de la clase `SecuenciaCaracteres`.

Finalidad: Trabajar con un vector dentro de una clase. Llamadas entre métodos. Dificultad Baja.

21. [Elimina ocurrencias eficiente] Se pide modificar la solución del ejercicio 20 [Elimina ocurrencias ineficiente] para que elimine las ocurrencias de un carácter de forma eficiente, tal y como se describe en el ejercicio 5 [Elimina ocurrencias de una componente -versión ineficiente-] de la Relación de Problemas III. El método debe modificar la misma secuencia sobre la que se aplica (es decir, debe modificar los caracteres del vector privado)

Aplique este método para eliminar todos los espacios en blanco en el texto del Quijote:

<http://decsai.ugr.es/jccubero/FP/Quijote.txt>

Finalidad: Trabajar con un vector dentro de una clase. Dificultad Media.

22. **[Elimina repetidos ineficiente]** Sobre la clase `SecuenciaCaracteres`, añada un método `EliminaRepetidos` que quite los elementos repetidos, de forma que cada componente sólo aparezca una única vez. Se mantendrá la primera aparición, de izquierda a derecha. Por ejemplo, si la secuencia contiene
{ 'b', 'a', 'a', 'h', 'a', 'a', 'a', 'a', 'c', 'a', 'a', 'a', 'g' }
después de quitar los repetidos, ésta quedaría como sigue:
{ 'b', 'a', 'h', 'c', 'g' }

Observe que se pide explícitamente que el método modifique los datos contenidos en la secuencia.

Implemente los siguientes algoritmos para resolver este problema:

- a) Usando un **vector local** `sin_repetidos` dentro del método `EliminaRepetidos`. En este vector local iremos almacenando la primera ocurrencia de cada carácter. Una vez terminemos de añadir convenientemente caracteres al vector `sin_repetidos`, lo volcamos en `vector_privado`. Nos quedaría:

```
Recorrer todas las componentes de "vector_privado"
  Si la componente NO está en "sin_repetidos",
    la añadimos a "sin_repetidos"
Volcar "sin_repetidos" en "vector_privado"
```

- b) El problema del algoritmo anterior es que usa un vector local, lo que podría suponer una carga importante de memoria si trabajásemos con vectores grandes. Por lo tanto, vamos a resolver el problema sin usar vectores locales. Defina otra versión del método `EliminaRepetidos` que implemente el siguiente algoritmo:

```
Recorrer todas las componentes de "vector_privado"
  Si NO es la primera aparición de la componente,
    Eliminarla
```

Para comprobar que no sea la primera aparición, basta buscarla a su izquierda (en la parte del vector privado que hay a su izquierda) Para eliminarla, basta llamar al método `Elimina` (de un único carácter). La implementación del método `Elimina` se encuentra en

<http://decsai.ugr.es/jccubero/FP/SecuenciaCaracteres.cpp>

Construya un programa que lea los caracteres de la cadena uno a uno con `cin.get()`, hasta que se introduzca el carácter `#` y muestre el resultado de quitarle los repetidos.

Ejemplo de entrada: ggabaabghc# — Salida correcta: gabhc

Finalidad: Uso de vectores locales dentro de los métodos. Reutilización de métodos. Dificultad Media.

23. [Elimina repetidos eficiente] Recupere la solución del ejercicio 22 [Elimina repetidos ineficiente]. El algoritmo visto en dicho ejercicio, nos obliga a desplazar muchas componentes cada vez que encontremos una repetida. Proponga una alternativa (sin usar vectores locales) para que el número de desplazamientos sea el menor posible e impléméntela.

Consejo: Use la misma técnica que se indicó en el ejercicio 6 [Elimina ocurrencias de una componente -versión eficiente-] de la Relación de Problemas III.

Puede probar la diferencia radical en el tiempo de ejecución de los algoritmos anteriores con el texto del Quijote, disponible en:

<http://decsai.ugr.es/jccubero/FP/Quijote.txt>

Mientras que la versión eficiente tarda algunos *milisegundos*, la versión ineficiente puede tardar *una hora* (en un i7)

Finalidad: Desarrollo de algoritmos eficientes. Dificultad Media.

24. [Moda] Se quiere calcular la moda de una secuencia de caracteres, es decir, el carácter que más veces se repite. Por ejemplo, si la secuencia fuese

`{ 'l', 'o', 's', ' ', 'd', 'o', 's', ' ', 'c', 'o', 'f', 'r', 'e', 's' }`

los caracteres que más se repiten son 'o' y 's' con un total de 3 apariciones. La moda sería cualquiera de ellos, por ejemplo, el primero encontrado 'o'.

Defina un método sobre la clase `SecuenciaCaracteres` para realizar esta tarea. Para almacenar de forma conjunta el carácter y la frecuencia usaremos el siguiente struct:

```
struct FrecuenciaCaracter{
    char caracter;
    int  frecuencia;
}
```

El campo `caracter` contendrá el carácter en cuestión ('o') y en el campo `frecuencia` el conteo de la moda (3). Este struct es lo que el método deberá devolver.

Construya un programa que lea los caracteres de la secuencia, tal y como se hizo en el ejercicio 19 [Palíndromo e invierte dentro de la clase `SecuenciaCaracteres`] de esta Relación de Problemas. A continuación, calcule la moda y muéstrela en pantalla junto con su frecuencia.

Para calcular la moda, se recomienda que use un vector auxiliar en el que almacene los caracteres que ya se han procesado en algún momento anterior.

Ejemplo de entrada: los dos cofres#

— Salida correcta:

Moda: o
Frecuencia: 3

Utilice el texto del Quijote sin espacios en blanco (obtenido al ejecutar el método del ejercicio 21 [Elimina ocurrencias eficiente]) disponible en:

http://decsai.ugr.es/jccubero/FP/Quijote_sin_espacios.txt

La moda es la letra 'e' con un total de 217900 apariciones.

Finalidad: Devolución de un struct. Trabajar con un vector local a un método. Dificultad Media.

25. [Tabla de temperaturas] Se desea construir una clase `TablaTemperaturas` para almacenar y gestionar las temperaturas de 10 ciudades tomadas cada hora durante un día. Por lo tanto, internamente trabajaremos con una matriz de 10 filas y 24 columnas de datos de tipo `double`. Defina, al menos, los siguientes métodos:

- `Valor` para obtener la temperatura de una ciudad (dada por un índice entero entre 0 y 9) a una hora determinada (será otro entero entre 0 y 23)
- `Modifica` para modificar la temperatura de una ciudad a una hora determinada.
- `Minimo` para obtener la mínima temperatura de una ciudad dada. Como siempre ocurre cuando trabajamos con vectores y matrices, el método debe devolver la columna (hora) en la que se alcanza, en vez del valor de la temperatura.
- `MaxMinimos` para obtener la ciudad y la hora en la que se alcanzó la máxima temperatura de entre los mínimos alcanzados por cada ciudad. El método debe devolver un `struct` del tipo

```
struct ParFilaColumna{  
    int fila;  
    int columna;  
}
```

con la fila y columna en la que se alcanza el máximo de los mínimos. El método debe calcular el mínimo de cada fila (ciudad) y luego el máximo de todos ellos (de forma análoga a lo pedido en el ejercicio 17 [Máximo de los mínimos] de la Relación de Problemas III)

En el ejercicio 17 [Máximo de los mínimos] de la Relación de Problemas III se vio una solución eficiente, pero ahora queremos implementar otra solución no tan optimizada pero que nos permita trabajar con vectores locales y reutilizar otros métodos. En concreto, se pide que el método `MaxMinimos` implemente el siguiente algoritmo:

```
Recorrer todas las ciudades
    Llamar al método Minimo para calcular el mínimo
    de cada ciudad.
    Guardar los resultados en un vector de mínimos
Recorrer el vector de mínimos para calcular el máximo
```

Construya un programa que lea las 24 temperaturas de 10 ciudades e imprima la fila y columna en la que se alcanza el máximo de los mínimos. Puede comprobar que la solución es correcta usando como datos de entrada los del siguiente fichero:

http://decsai.ugr.es/jccubero/FP/datos_temp.txt

La salida debe ser:

```
Ciudad: 2
Hora:    7
Temperatura: 16.2
```

Finalidad: Devolución de un struct. Vectores locales. Llamadas entre métodos. Dificultad Baja.

26. [Túnel] Se quiere gestionar la información de los vehículos que pasan por un túnel. Para ello, se quiere construir la clase `Tune1` acorde a lo siguiente:

- Debe tener en cuenta la longitud en km del túnel. Debe elegir si se especifica en el constructor o en algún método.
- Las entradas y salidas de los vehículos al túnel, se indicarán ejecutando sendos métodos `Entra` y `Sale` respectivamente. Ambos métodos recibirán como parámetros la matrícula del vehículo y el número de segundos transcurridos desde las 0h,0m,0s.

Por ejemplo, si tenemos un objeto `tune1` y el vehículo con matrícula "0001ABC" entra a los 53 segundos, habrá que ejecutar la sentencia `tune1.Entra("0001ABC", 53);`. Si ese mismo vehículo sale a los 30 segundos después de haber entrado, habría que ejecutar la sentencia `tune1.Sale("0001ABC", 83);`.

- Declare dentro de la clase `Tune1` tres vectores `matriculas`, `entradas`, `salidas` para poder almacenar las matrículas, los segundos (desde las 0h,0m,0s) en los que se produjo la entrada y los segundos (desde las 0h,0m,0s) en los que se produjo la salida de cada vehículo. Al principio, todas las componentes de los vectores se pondrán con un valor nulo. Cada vez que entre un vehículo, se añadirá la matrícula al vector `matriculas` y los segundos correspondientes al vector `entradas`. Cuando salga un vehículo, en la correspondiente componente del vector `salidas` se pondrá el número de segundos.

Por ejemplo, si se producen los siguientes movimientos:

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

```
Entra 4467JHG 0 seg
Entra 5678KGR 4 seg
Sale 5678KGR 108 seg
```

en el vector de matrículas tendremos ["4467JHG", "5678KGR"], en el vector de entradas [0, 4] y en el vector de salidas [NULO, 108].

- Añada los métodos que estime oportuno para poder consultar los datos almacenados de los vehículos que hay en el túnel.

Construya un programa que lea los siguientes datos:

- En primer lugar, leerá la longitud en Km del túnel
- A continuación irá leyendo los movimientos de los coches de la siguiente forma:
 - Se leerá un `char` indicando el tipo de movimiento: 'E' si es una entrada y 'S' si es una salida. El carácter '#' indicará la finalización de los datos de entrada.
 - A continuación se leerá un string con la matrícula del vehículo
 - Finalmente, se leerá el instante en el que se produce el movimiento. Vendrá dado en el formato hora, minuto, segundo.

Por ejemplo:

```
3.4          -> Longitud en Km del túnel
E 4467JHG 0 1 2 -> El vehículo 4467JHG entra en el inst. 0h1m2s
E 5678KGR 0 1 6 -> El vehículo 5678KGR entra en el inst. 0h1m6s
S 5678KGR 0 2 50 -> El vehículo 5678KGR sale en el inst. 0h2m50s
E 1234ABC 0 3 14 -> El vehículo 1234ABC entra en el inst. 0h3m14s
#              -> Fin entrada datos
```

El programa tendrá que calcular, a partir de los datos de cada instante, el número de segundos transcurridos desde las 0h0m0s. Para ello debe usar la clase `Instante` vista en el ejercicio 16 [Instante]. Puede usar la implementación de la clase `Instante` que se encuentra en el siguiente enlace:

http://decsai.ugr.es/jccubero/FP/IV_InstanteClase.cpp

Una vez calculados los segundos, podrá llamar al correspondiente método `Entra` o `Sale` del túnel, dependiendo de si era una entrada o una salida.

Una vez leídos todos los datos, el programa mostrará un informe mostrando las matrículas de cada vehículo y su velocidad media en el túnel (en el caso de que hayan salido). Por ejemplo, con los anteriores datos sería:

```
Matrícula:      4467JHG
Velocidad:      No ha salido

Matrícula:      5678KGR
```

Velocidad: 117.7 km/h

Matrícula: 1234ABC

Velocidad: No ha salido

Para mostrar los datos de la velocidad (que es un double) use la clase `FormateadorDoubles` vista en el ejercicio 17 [[Formateador de doubles](#)] y redondee a 1 cifra decimal.

Si lo desea, puede recuperar sus propias versiones de las clases `Instante` y `FormateadorDoubles`, o bien utilizar las implementadas en el siguiente fichero:

http://decsai.ugr.es/jccubero/FP/IV_TunelEsbozo.cpp

Puede comprobar que su programa es correcto comparando los datos de entrada del fichero

http://decsai.ugr.es/jccubero/FP/datos_tunel.txt

con los resultados correctos incluidos en este fichero:

http://decsai.ugr.es/jccubero/FP/resultados_tunel.txt

Finalidad: Trabajar con vectores dentro de una clase. Construir un programa con varias clases. Dificultad Media.

Ejercicios complementarios sobre funciones

27. [Sistema de Hond't] Recupere la solución del ejercicio 4 [Sistema de D'Hondt] . Reescribalo utilizando una función para calcular el cociente de Hond't.
Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.
28. [Es letra del alfabeto español] Defina una función que compruebe si un dato de tipo `char` es una letra del alfabeto español. Debe considerar la 'ñ' así como las vocales mayúsculas y minúsculas acentuadas.
Escriba un sencillo programa de prueba.
Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.
29. Reescriba la solución del ejercicio 21 (factorial y potencia) de la Relación de Problemas II, modularizándola con funciones.
Para el factorial, use la función `Factorial` vista en las transparencias de clase. Para el cómputo de la potencia, defina la función `Potencia`.
Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.
30. Reescriba la solución del ejercicio 64 que calcula la suma de los primeros T factoriales. Para ello, debe leer el valor T usando la función `LeeIntRango` del ejercicio 2 para obligar a que esté en el intervalo $[1, 20]$.
Debe definir la función `SumaFactoriales` que calcule la suma pedida. Implemente dos versiones de esta función:
- En una primera versión, la función `SumaFactoriales` debe llamar a la función `Factorial`, para realizar la suma tal y como se indica en el ejercicio 64
 - En una segunda versión, la función `SumaFactoriales` debe realizar la suma de forma directa tal y como se indica en el ejercicio 65. Ponga dentro de un comentario la primera versión.
- Finalidad: Familiarizarnos con la llamada entre funciones. Dificultad Baja.*
31. Retome la solución del ejercicio 31 (Gaussiana) y modifíquela introduciendo funciones dónde crea conveniente. Al menos debe definir la función `gaussiana` para que calcule el valor de la ordenada, para unos valores concretos de abscisa, esperanza y desviación.
Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

32. Retome la solución del ejercicio 31 (Gaussiana)

Ahora estamos interesados en obtener el área que cubre la función gaussiana en el intervalo $[-\infty, x]$. Dicho valor se conoce como la *distribución acumulada (cumulative distribution function)* en el punto x , abreviado $CDF(x)$. Matemáticamente se calcula realizando la integral:

$$CDF(x) = \int_{-\infty}^x \text{gaussiana}(t) dt$$

Puede probar algunos valores ejecutando la siguiente calculadora online:

<https://www.easycalculation.com/statistics/normal-distribution.php>

El valor de x hay que introducirlo en el apartado *Below*.

Para no tener que implementar el concepto de integral, vamos a recurrir a una aproximación numérica para obtener $CDF(x)$. Puede consultarse en la Wikipedia (buscar *Normal distribution*) que la siguiente fórmula proporciona una aproximación al valor de $CDF(x)$:

$$CDF(x) = \text{Área hasta } (x) \approx 1 - \text{gaussiana}_{0,1}(x)(b_1t + b_2t^2 + b_3t^3 + b_4t^4 + b_5t^5)$$

dónde:

$$t = \frac{1}{1 + b_0x} \quad b_0 = 0.2316419 \quad b_1 = 0.319381530 \quad b_2 = -0.356563782$$
$$b_3 = 1.781477937 \quad b_4 = -1.821255978 \quad b_5 = 1.330274429$$

Cree otra función para calcular el área hasta un punto cualquiera x , es decir, $CDF(x)$, usando la anterior aproximación. Para implementar esta función, use la función *Potencia* del ejercicio 29 cuando tenga que calcular los términos t^i .

Modifique el programa principal del ejercicio 31 para que llame a la función $CDF(x)$ e imprima las ordenadas correspondientes a las abscisas `minimo`, `minimo + incremento`, `minimo + 2*incremento`, etc.

Ejemplo de entrada: P 12 5 R 11 13 0.5 V S

-- Salida correcta:

f(11)=0.0782085

CDF(11)=0.998414

f(11.5)=0.0793905

CDF(11.5)=0.998573

f(12)=0.0797885

CDF(12)=0.998727

f(12.5)=0.0793905

CDF(12.5)=0.998875

f(13)=0.0782085

CDF(13)=0.999016

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

Finalidad: Entender las llamadas entre funciones y la importancia de la ocultación de información. Dificultad Baja.

33. Retome la solución del ejercicio 43 (parking) de la Relación de Problemas II. Se quiere extender para poder trabajar con varios parkings o con varias tarifas distintas. Supondremos que en todos los casos, se tiene el mismo número de tramos (4) aunque puede variar la cuantía a tarifar por minutos y los límites de cada uno de los tramos.

Para ello, se pide definir la función `Tarifa` que obtenga la tarifa final aplicable a cualquier caso. En concreto, en este ejercicio, se van a leer sólo dos casos, correspondientes a dos parkings o tarificaciones distintas. Se pide por tanto construir un programa que lea los siguientes datos:

- En primer lugar el programa lee los datos de cada uno de los dos casos, es decir, los límites de los tramos y las tarifas que se aplican en cada tramo (ver tabla debajo)
- A continuación, se leen varios pares de instantes de entrada y salida. Se leen en el orden instante de entrada (hora, minuto y segundo) e instante de salida.

La entrada de datos finaliza cuando se introduce un -1 como hora de entrada.

El programa imprimirá la tarifa resultante de cada uno de los parkings para cada par de instantes de entrada y salida, así como la suma total recaudada en cada caso.

Por ejemplo:

```
30      -> Limite 1 del parking 1
90      -> Limite 2 del parking 1
120     -> Limite 3 del parking 1
660     -> Limite 4 del parking 1
0.0412  -> Tarifa Tramo 1 del parking 1
0.0370  -> Tarifa Tramo 2 del parking 1
0.0311  -> Tarifa Tramo 3 del parking 1
0.0305  -> Tarifa Tramo 4 del parking 1
31.55   -> Tarifa día completo del parking 1
35      -> Limite 1 del parking 2
85      -> Limite 2 del parking 2
110     -> Limite 3 del parking 2
660     -> Limite 4 del parking 2
0.0402  -> Tarifa Tramo 1 del parking 2
0.0375  -> Tarifa Tramo 2 del parking 2
0.0319  -> Tarifa Tramo 3 del parking 2
0.0315  -> Tarifa Tramo 4 del parking 2
32      -> Tarifa día completo del parking 2
2 1 30  -> Entra a las 2 de la madrugada, 1 minuto, 30 segundos
4 2 50  -> Sale a las 4 de la madrugada, 2 minutos y 50 segundos
```

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

2 2 5 -> Entra a las 2 de la madrugada, 2 minutos, 5 segundos
4 3 7 -> Sale a las 4 de la madrugada, 3 minutos, 7 segundos
-1 -> Fin de la entrada de datos.

Ejemplo de entrada:

```
30 90 120 660 0.0412 0.0370 0.0311 0.0305 31.55
35 85 110 660 0.0402 0.0375 0.0319 0.0315 32
2 1 30 4 2 50
2 1 30 3 41 31
2 1 30 5 41 31
2 1 30 23 1 1 -1
```

— Salida correcta:

```
4.4195 4.4262
3.767 3.7605
7.439 7.5445
31.55 32
```

```
47.1755
47.731
```

Finalidad: Diseño de una función. Dificultad Media.

34. Retome la solución del ejercicio 55 (población) de la Relación de Problemas II. Re-escribalo usando las funciones `LeeIntRango` del ejercicio 2 para leer los valores de las tasas y `LeeIntMayorIgualQue` del ejercicio 3 para leer el número de años que sea positivo. Defina también sendas funciones para calcular los dos valores que se piden en el ejercicio, a saber, el número de habitantes después de tres años y el número de años que pasarán hasta doblar la población inicial. Intente diseñar las funciones para que sean lo más generales posible.

Ejemplo de entrada:

```
1375570814 2000 32 2000 2000 12 7 -4 -4 3
```

— Salida correcta: 1490027497 27 2824131580

Finalidad: Diseño de una función. Dificultad Baja.

35. Retome la solución de los ejercicios 44 y 57 (servicio atención telefónica) de la Relación de Problemas II. Recordemos que el criterio de subida salarial era el siguiente:

Entre 20 y 30 casos resueltos:	+3%
Más de 30 casos resueltos:	+4%
Grado de satisfacción ≥ 4 :	+2%

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

Defina una función `SalarioFinal` que calcule el salario final del trabajador, en función de los datos anteriores.

Al igual que se pedía en el ejercicio 57 debe ir leyendo los datos de tres empleados en el siguiente orden:

```
7.5      <- Salario de 7.5 euros por hora (el mismo para todos)
2 124 1 3 <- Empleado 2, 124'', resuelto,      grado sat: 3
1 32 0 0  <- Empleado 1, 32'', no resuelto, grado sat: 0
2 26 0 2  <- Empleado 2, 26'', no resuelto, grado sat: 2
-1        <- Fin de entrada de datos
```

El número de horas trabajadas de cada empleado será un número real y se calculará en función de la suma total de segundos dedicados a cada llamada telefónica (la compañía no paga por el tiempo de estancia en la empresa sino por el tiempo dedicado a resolver casos)

El programa debe llamar a la función `SalarioFinal` para calcular el salario final de cada uno de los tres empleados y los debe mostrar en pantalla.

Puede utilizar el fichero de datos `datos_atencion_telefonica.txt` disponible en [PRADO](#) . La salida correcta para este fichero es
1016.196 118.287 128.893

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Media.

36. Retome la solución del ejercicio 35 de esta Relación de Problemas. Modifíquela para tener en cuenta que los límites correspondientes a los casos resueltos (20 y 30) y el grado de satisfacción media (4), así como los porcentajes de incrementos correspondientes (3%, 4% y 2%) ya no son constantes sino que pueden variar.

Por lo tanto, debe leer desde teclado dichos valores límites (justo después del salario por hora y en el orden indicado anteriormente) y cambiar la función definida en el ejercicio 35 para que tenga en cuenta este cambio.

Puede utilizar el fichero de datos

`datos_atencion_telefonica_limites_variables.txt`

disponible en [PRADO](#) . La salida correcta para este fichero es
1016.196 118.287 128.893

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

37. Implemente la solución del ejercicio 27 (Narcisista) de la relación de problemas II, usando funciones.

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

38. Escriba una función en C++ `LeeOpcion2Alternativas` que imprima en pantalla un mensaje, lea una opción como un carácter y sólo permita aceptar los caracteres 'S' o 'N' (mayúscula o minúscula). ¿Qué debería devolver la función? ¿El carácter leído o un `bool`? Aplique esta función en la solución del ejercicio 48 (Renta bruta y neta) de la relación de problemas II, para leer si una persona es pensionista o si es autónomo.

Finalidad: Familiarizarnos con la definición de funciones y el paso de parámetros. Dificultad Baja.

39. A un trabajador le pagan según sus horas trabajadas y la tarifa está a un valor por hora. Si la cantidad de horas trabajadas es mayor de 40 horas, la tarifa por hora se incrementa en un 50 % para las horas extras (las que haya por encima de 40). Construir una función que dado el número total de horas trabajadas y el precio por hora, devuelva el salario del trabajador.

Finalidad: Familiarizarnos con la definición de funciones y paso de parámetros. Dificultad Baja.

40. Cree las siguientes funciones relacionadas con la progresión geométrica que se vio en el ejercicio 68 de la relación de problemas II. Analice cuáles deben ser los parámetros a estas funciones.

- a) Una función `SumaHasta` que calcule la suma de los primeros k valores de una progresión geométrica.

Para implementarla, use el mismo algoritmo (con un bucle `for`) que se vio como solución del ejercicio 68 de la relación de problemas II.

- b) Una función `ProductoHasta` para que multiplique los k primeros elementos de la progresión, aplicando la siguiente fórmula:

$$\prod_{i=1}^{i=k} a_i = \sqrt{(a_1 a_k)^k}$$

Observe que no se pide calcular los productos acumulados en un bucle sino que simplemente evalúe la expresión $\sqrt{(a_1 a_k)^k}$ que le da directamente el producto de los k primeros términos.

- c) Una función `SumaHastaInfinito` para calcular la suma hasta infinito, según la siguiente fórmula:

$$\sum_{i=1}^{i=\infty} a_i = \frac{a_1}{1 - r}$$

De nuevo, observe que sólo hay que aplicar la expresión $\frac{a_1}{1 - r}$ para obtener la suma pedida. Esta fórmula sólo se puede aplicar cuando el valor absoluto de la razón es menor o igual que 1, ya que, en caso contrario, la suma saldría infinito.

Cree un programa principal que llame a estas funciones.

Finalidad: Enfatizar la importancia de la ocultación de información. Dificultad Baja.

41. Amplie el ejercicio 40 cambiando la implementación de la función `SumaHasta`. Para ello, en vez de usar un bucle aplicamos la siguiente fórmula que nos da la sumatoria aplicando únicamente cinco operaciones:

$$\sum_{i=1}^{i=k} a_i = a_1 \frac{r^k - 1}{r - 1}$$

Es muy importante remarcar que el programa `main` no cambia nada. Hemos cambiado la implementación de la función y lo hemos podido hacer sin cambiar el `main`, ya que éste no tenía acceso al código que hay dentro de la función. Esto es *ocultación de información* tal y como se describió en las clases de teoría.

Nota. Calculad la potencia (r^k) con la función `pow` y hacerlo también usando la función `Potencia` definida en el ejercicio 29 de esta Relación de Problemas.

Hay que destacar que el cómputo de la potencia es una operación costosa, por lo que hasta podría ser más lenta la versión nueva que la antigua usando un bucle `for`. Probad distintos valores para ver si hay diferencias significativas. En cualquier caso, lo importante es que mientras no cambiemos la cabecera de la función `SumaHasta`, podemos cambiar su implementación sin tener que cambiar ni una línea de código del `main`.

Finalidad: Enfatizar la importancia de la ocultación de información. Dificultad Baja.

42. Se pide construir las siguientes funciones:

- Una función que compruebe si un carácter es una mayúscula:

```
bool EsMayuscula(char caracter)
```

- Una función que realice un filtro de entrada para mayúsculas, es decir, dentro de la función se van leyendo caracteres (con `cin`) en un bucle hasta que se introduzca una mayúscula cualquiera o hasta que se introduzca un carácter terminador (asuma que dicho carácter es `#`)

La cabecera de la función será la siguiente:

```
char LeeMayuscula()
```

Esta función debe llamar a la anterior `EsMayuscula`. En el caso de que el carácter leído sea el terminador, la función devolverá ese mismo valor (`#`)

Construya ahora un programa principal que vaya leyendo caracteres, para lo cual debe llamar a la función `LeeMayuscula`. La entrada de datos terminará cuando se introduzca el terminador `#` y el programa debe mostrar en pantalla el número total de mayúsculas que se han introducido.

Puede suponer que no se introducen espacios en blanco.

Por ejemplo, si la entrada de datos es `abcDeFGHi j#`, la salida será 4 (se han introducido cuatro mayúsculas: D, F, G, H)

Finalidad: Mostrar cómo encapsular tareas dentro de funciones y cómo se realiza la llamada entre ellas. Dificultad Baja.

43. Recupere la solución del ejercicio 15 de la Relación de Problemas II (pasar de mayúscula a minúscula y viceversa usando un enumerado) Para que el tipo de dato enumerado sea accesible desde dentro de las funciones, debemos ponerlo antes de definir éstas, es decir, en un ámbito global a todo el fichero. Se pide definir las siguientes funciones y cread un programa principal de ejemplo que las llame:

- a) `Capitalizacion` nos dice si un carácter pasado como parámetro es una minúscula, mayúscula u otro carácter. A dicho parámetro, llamadlo `una_letra`. La función devuelve un dato de tipo enumerado.
- b) `Convierte_a_Mayuscula` comprueba si un carácter pasado como parámetro es minúscula (para ello, debe llamar a la función `Capitalizacion`), en cuyo caso lo transforma a mayúscula. Si el carácter no es minúscula debe dejar la letra igual. A dicho parámetro, llamadlo `caracter`.

Esta función hace lo mismo que la función `tolower` de la biblioteca `cctype`

Observad que el parámetro `una_letra` de la función `Capitalizacion` podría llamarse igual que el parámetro `caracter` de la función `Convierte_a_Mayuscula`. Esto es porque están en ámbitos distintos y para el compilador son dos variables distintas. Haced el cambio y comprobarlo.

- c) `Convierte_a_Minuscula` análoga a la anterior pero convirtiendo a minúscula. Observad que la constante de amplitud

```
const int AMPLITUD = 'a' - 'A';
```

es necesaria declararla como constante local en ambas funciones. Para no repetir este código, ¿qué podemos hacer? Implemente la solución adoptada.

- d) `CambiaMayusculaMinuscula`, a la que se le pase como parámetro un `char` y haga lo siguiente:
 - si el argumento es una letra en mayúscula, devuelve su correspondiente letra en minúscula,
 - si el argumento es una letra en minúscula, devuelve su correspondiente letra en mayúscula,
 - si el argumento no es ni una letra mayúscula, ni una letra minúscula, devuelve el carácter pasado como argumento.

Finalidad: Entender cómo se llaman las funciones entre sí. Dificultad Media.

44. *Examen Septiembre 2014*. Dos números amigos son dos números naturales a y b , tales que la suma de los divisores propios de a más uno es igual a b , y viceversa. Un ejemplo de números amigos es el par de naturales (220; 284), ya que:

- Los divisores propios de 220 son 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110, que suman 283, y $283 + 1 = 284$.
- Los divisores propios de 284 son 2, 4, 71 y 142, que suman 219, y $219 + 1 = 220$.

Realice un programa que implemente estas dos tareas:

- a) En primer lugar debe leer dos números naturales e indicar si son o no amigos.
- b) A continuación leerá otro número natural, n , e informará si existe algún número amigo de n en el intervalo centrado en n y de radio 3.

Utilice las funciones que estime oportuno.

Finalidad: Descomponer la solución de un problema en varias funciones. Dificultad Media.

45. Defina una función para implementar la solución del ejercicio 70 de la relación de problemas II (Serie)

Dificultad Media.

46. Defina una función para implementar la solución del ejercicio 35 de la relación de problemas II (número feliz)

Dificultad Media.

Ejercicios complementarios sobre clases

47. [Cuadrado] En este ejercicio se plantean varias modificaciones. Debe entregar un fichero `cpp` por cada uno de los apartados.

Se desea implementar una clase `Cuadrado` para representar figuras geométricas de tipo cuadrado. Un cuadrado viene definido por la coordenada de la esquina inferior izquierda (x, y) y la longitud del lado. Por lo tanto, el tipo de cuadrados que se consideren son los que su base es paralela al eje de las abscisas.

- a) Defina una clase en C++ para representar un cuadrado. Por ahora, utilice únicamente datos miembro públicos.

Cree un programa principal que lea las coordenadas y la longitud, se las asigne a una variable `parcela` de tipo `Cuadrado`, calcule el área y la imprima en pantalla. Introduzca ahora las coordenadas y longitud de otro cuadrado y haga lo siguiente:

- Modifique los datos del cuadrado anterior con los nuevos datos.
- Defina un segundo cuadrado (cree otro objeto) con los nuevos datos.

- b) Cambie ahora los datos miembro públicos y póngalos privados. Tendrá que añadir métodos para asignar y ver los valores de los datos miembro. Añada un método para asignar un valor a cada uno de los tres datos miembro. Modifique el `main` para tener en cuenta estos cambios.

- c) En vez de usar un método para asignar la abscisa de la esquina y otro método para asignar la ordenada, defina un único método para asignar las coordenadas de la esquina a la misma vez.

Observe que los métodos permiten definir la política de acceso a los datos miembro. Si tengo previsto cambiar por separado las coordenadas de la esquina, usaré métodos de asignación por separado - opción b) - En caso contrario, usaré un único método - opción c) -.

- d) En vez de calcular el área en el programa principal, vamos a ponerlo como un método de la clase y así lo reutilizaremos todas las veces que necesitemos.

Añada un método para el cálculo del área y modifícalo el `main` para tener en cuenta este cambio.

¿Añadimos `area` como dato miembro del cuadrado? La respuesta es que no ¿Por qué?

Añada también un método para el cálculo del perímetro.

- e) Añada un constructor al cuadrado para que el objeto esté en un estado válido en el mismo momento de su definición. El constructor deberá tener como parámetros, obligatoriamente, las coordenadas de la esquina y la longitud del lado. Tendrá que modificar el `main` para tener en cuenta este cambio.

f) Suprima ahora los métodos que modificaban los datos miembro. De esta forma, una vez creado el objeto (pasándole los datos apropiados en el constructor) ya no podremos modificar los datos miembro. Esto es útil en aquellas situaciones en las que no queremos permitir que el estado del objeto cambie, una vez que se ha creado.

48. [Elimina ocurrencias entre dos posiciones] Se pide modificar la solución del ejercicio 21 [Elimina ocurrencias eficiente] para que elimine todas las ocurrencias de un carácter que hay entre dos índices de la secuencia.

Por ejemplo, el resultado de eliminar todas las ocurrencias del carácter a entre las posiciones 2 y 6 de la secuencia abacdeaaafa sería la secuencia abcdeafa.

¿Mantendría los dos métodos, el visto en este ejercicio y el visto en el ejercicio 21 [Elimina ocurrencias eficiente] ?

Finalidad: Trabajar con un vector dentro de una clase. Reutilizar un método. Dificultad Baja.

49. [Elimina ultimos] Sobre la clase `SecuenciaCaracteres`, construya un método `EliminaUltimos` para eliminar todos los caracteres que haya al final de la secuencia y que sean iguales a un carácter determinado. En definitiva, se desea realizar lo mismo que se implementó con funciones en el ejercicio 8 [Elimina últimos] , salvo que:

- a) En vez de trabajar con funciones a las que se le pasa como parámetro un `string`, trabajamos con métodos que actúan sobre un objeto de la clase `SecuenciaCaracteres`.
- b) Se desea modificar la secuencia de caracteres sobre la que se aplica el método `EliminaUltimos`.

Finalidad: Trabajar con un vector dentro de una clase. Dificultad Baja.

50. [Elimina exceso de blancos] Sobre la clase `SecuenciaCaracteres`, añada un método `EliminaExcesoBlancos` para eliminar el exceso de caracteres en blanco, es decir, que sustituya todas las secuencias de espacios en blanco por un sólo espacio. Por ejemplo, si la secuencia original es (' ', 'a', 'h', ' ', ' ', ' ', 'c'), que contiene una secuencia de tres espacios consecutivos, la secuencia resultante debe ser (' ', 'a', 'h', ' ', 'c').

Nota: Debe hacerse lo más eficiente posible.

Construya un programa que lea los caracteres de la cadena uno a uno con `cin.get()`, hasta que se introduzca el carácter # y muestre el resultado de quitarle el exceso de blancos. Puede probar el programa con el siguiente fichero, que contiene el Quijote con más de un espacio en blanco entre palabras:

http://decsai.ugr.es/jccubero/FP/Quijote_con_exceso_de_blanco.txt

Finalidad: Recorrido de las componentes de un vector, en el que hay que recordar lo que ha pasado en la iteración anterior. Dificultad Media.

51. [Circunferencia con structs] Recupere la definición del registro `CoordenadasPunto2D` del ejercicio 73 de esta relación de problemas y la solución al ejercicio 12 (Circunferencia) de la relación de problemas I.

Cree ahora una clase llamada `Circunferencia`. Para establecer el centro, se usará un dato miembro que ha de ser de tipo `CoordenadasPunto2D`.

Añada métodos para obtener la longitud de la circunferencia y el área del círculo interior.

Añada también un método para saber si la circunferencia contiene a un punto cualquiera. Recordemos que un punto (x_1, y_1) está dentro de una circunferencia con centro (x_0, y_0) y radio r si se verifica que:

$$(x_0 - x_1)^2 + (y_0 - y_1)^2 \leq r^2$$

Observe que el valor de π debe ser constante, y el mismo para todos los objetos de la clase `Circunferencia`.

Cree un programa principal que lea el centro y el radio de una circunferencia, las coordenadas de un punto y muestre en pantalla la longitud de la circunferencia, el área del círculo y nos diga si el punto está dentro o no de la circunferencia.

Ejemplo de entrada: 2.1 3.2 5.8 2.2 4.6

— Salida correcta: 36.4425 105.683 El punto está dentro

Ejemplo de entrada: 2.1 3.2 5.8 2.2 10.36

— Salida correcta: 36.4425 105.683 El punto no está dentro

Finalidad: Trabajar con clases y el tipo struct. Dificultad Baja.

52. [Login] (*Examen Febrero 2013*) Recupere la solución del ejercicio 10 [Login] de la Relación de Problemas III. Implemente la clase `Login` y defina el método `Codifica` que recibirá una cadena de caracteres (tipo `string`) formada por el nombre y apellidos (separados por uno o más espacios en blanco) y devuelva otra cadena con la sugerencia de login.

```
class Login{
private:
    int num_caracteres_a_coger;
public:
    Login (int numero_caracteres_a_coger)
        :num_caracteres_a_coger(numero_caracteres_a_coger)
    { }
    string Codifica(string nombre_completo){
```

```
        .....  
    }  
};
```

Los únicos métodos que necesita usar de la clase `string` son `size` y `push_back`. Construya un programa que lea los caracteres de la cadena uno a uno con `cin.get()`, hasta que el usuario introduzca el carácter `#` y muestre el resultado de la codificación en pantalla.

Finalidad: Recorrido de las componentes de un vector, controlando qué ha ocurrido anteriormente. Dificultad Media.

53. [Cuenta mayúsculas en una clase] Sobre el ejercicio 26 de la Relación de Problemas III, construya una clase específica `ContadorMayusculas` que implemente los métodos necesarios para llevar el contador de las mayúsculas. Lo que se pretende es que la clase proporcione los métodos siguientes:

```
void IncrementaConteo (char mayuscula)  
int  CuantasHay (char mayuscula)
```

El primer método aumentará en uno el contador de la correspondiente mayúscula y el segundo indicará cuántas hay. Modifique el programa principal para que cree un objeto de esta clase y llame a sus métodos para realizar los conteos de las mayúsculas. Finalmente, hay que imprimir en pantalla cuántas veces aparece cada mayúscula.

Finalidad: Diseño de una clase contadora de frecuencias. Dificultad Media.

54. Recupere la solución del ejercicio 32 de esta relación de problemas sobre la función gaussiana. En vez de trabajar con funciones, plantee la solución con una clase. Debe diseñar la clase teniendo en cuenta que la función matemática gaussiana viene determinada unívocamente por el valor de la esperanza y la desviación, es decir, son estos dos parámetros lo que distinguen a una función gaussiana de otra.

Finalidad: Diseño de una clase. Dificultad Baja.

55. Recupere la solución del ejercicio 34 de esta relación de problemas (población con funciones). Re-escribalo para que los cálculos relacionados con la población estén encapsulados en una clase. La lectura de los valores en los rangos adecuados se hará con las mismas funciones que ya se definieron en ese ejercicio. Modifique apropiadamente el programa principal.

Finalidad: Diseño de una clase. Dificultad Baja.

56. Recupere la solución del ejercicio 33 de esta relación de problemas (parking con funciones). Re-escribalo para que los cálculos relacionados con el cálculo de la tarifa, estén encapsulados en una clase. Mantenga la definición de la función

`MinutosEntreInstantes` tal y como está. Modifique apropiadamente el programa principal.

Finalidad: Diseño de una clase. Dificultad Media.

57. En el ejercicio 40 de esta relación de problemas se definieron varias funciones para operar sobre una progresión geométrica. Definid ahora una clase para representar una progresión geométrica.
- Diseñad la clase pensando cuáles serían los datos miembro *esenciales* que definen una progresión geométrica, así como el constructor de la clase.
 - Definir un método `Termino` que devuelva el término k -ésimo.
 - Definid los métodos `SumaHastaInfinito`, `SumaHasta`, `MultiplicaHasta`.
 - Cread un programa principal que lea los datos miembro de una progresión, cree el objeto correspondiente y a continuación lea un entero `tope` e imprima los `tope` primeros términos de la progresión, así como la suma hasta `tope` de dichos términos.

Finalidad: Comparar la ventaja de un diseño con clases a uno con funciones. Dificultad Baja.

58. Se quiere construir una clase `DepositoSimulacion` para simular préstamos, ofreciendo la funcionalidad descrita en los ejercicios 52 (reinvierte capital e interés un número de años) y 53 (reinvierte capital e interés hasta obtener el doble de la cantidad inicial) de la relación de problemas II (página RP-II.25). Por tanto, la clase debe proporcionar, para un capital y unos intereses dados, métodos para:
- Calcular el capital que se obtendrá al cabo de un número de años,
 - Calcular el número de años que deben pasar hasta obtener el doble de la cantidad inicial.

A la hora de diseñar la clase, tendremos que analizar cuestiones como:

- ¿Cuáles son sus datos miembro? Parece claro que el capital y el interés sí lo serán ya que cualquier operación que se nos ocurra hacer con un objeto de la clase `DepositoSimulacion` involucra a ambas cantidades. ¿Pero y el número de años?
- ¿Qué constructor definimos?
- ¿Queremos modificar el capital y el interés una vez creado el objeto?
- ¿Queremos poder modificarlos de forma independiente?
- ¿Hay alguna restricción a la hora de asignar un valor al capital e interés?

- ¿Es mejor un método para calcular el número de años hasta obtener el doble de la cantidad inicial, o por el contrario es mejor un método para calcular el número de años hasta obtener una cantidad específica?

Finalidad: Diseñar la interfaz de una clase. Dificultad Baja.

59. Recupere la solución del ejercicio 39 de esta relación de problemas (cómputo del salario en función de las horas trabajadas) Defina una clase *Nomina* para gestionar el cómputo del salario final. Suponga que el porcentaje de incremento en la cuantía de las horas extras (50 %) y el número de horas que no se tarifican como extra (40) son valores que podrían cambiar, aunque no de forma continua. El número de horas trabajadas y la cuantía a la que se paga cada hora extraordinaria, sí son cantidades que varían de un trabajador a otro.

Finalidad: Diseñar la interfaz de una clase. Dificultad Baja.

60. Recupere la solución del ejercicio 48 (actualización de la retención fiscal) de la relación de problemas II. En este problema se leían caracteres de teclado ('s'/'n') para saber si una persona era autónomo, pensionista, etc.

```
cout << "\n¿La persona es un trabajador autónomo? (s/n) ";  
  
do{  
    cin >> opcion;  
    opcion = toupper(opcion);  
}while (opcion != 'S' && opcion != 'N');
```

Este código era casi idéntico para la lectura del resto de los datos. Para evitarlo, definí una clase *MenuSiNO* que encapsule esta funcionalidad y cambiar el programa principal para que use esta clase.

61. Recupere la solución del ejercicio 11 (recta) de esta relación de problemas. Se pide crear un programa principal que haga lo siguiente:
- Se presentará al usuario un menú principal para salir del programa o para introducir los valores de los coeficientes A, B, C de la recta.
 - Una vez introducidos los coeficientes se presentará al usuario un segundo menú, para que elija alguna de las siguientes opciones:
 - Mostrar el valor de la pendiente de la recta.
 - Mostrar la ordenada dada una abscisa (el programa tendrá que pedir la abscisa)
 - Mostrar la abscisa dada una ordenada (el programa tendrá que pedir la ordenada)
 - Volver al menú principal.

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

Para resolver este problema, debe crear dos clases `MenuPrincipal` y `MenuOperaciones`.

Finalidad: Trabajar con varias clases en un programa. Dificultad Media.

62. Se quiere construir una clase `Nomina` para realizar la funcionalidad descrita en el ejercicio 35 de la relación de problemas I sobre la nómina del fabricante y diseñador (página RP-I.15). Cread los siguientes programas (entregad un fichero por cada uno de los apartados):

- a) Suponed que sólo gestionamos la nómina de una empresa en la que hay un fabricante y tres diseñadores. Los salarios brutos se obtienen al repartir los ingresos de la empresa, de forma que el diseñador cobra el doble de cada fabricante.

El programa leerá el valor de los ingresos totales y calculará los salarios brutos de los fabricantes y diseñador, llamando a los métodos oportunos de la clase `Nomina`.

- b) Supongamos que se aplica una retención fiscal y que ésta es la misma para los fabricantes y el diseñador. En el constructor se establecerá el porcentaje de retención fiscal (de tipo `double`) y posteriormente no se permitirá que cambie, de forma que todas las operaciones que se hagan serán siempre usando la misma retención fiscal. Los salarios netos se obtienen al aplicar la retención fiscal a los salarios brutos (después de repartir los ingresos totales de la empresa):

```
salario_netto = salario_bruto -  
                salario_bruto * retencion_fiscal / 100.0
```

El programa leerá el valor de los ingresos totales y la retención fiscal a aplicar y calculará los salarios brutos y netos de los fabricantes y diseñador, llamando a los métodos oportunos de la clase `Nomina`.

- c) Supongamos que gestionamos las nóminas de varias sucursales de una empresa. Queremos crear objetos de la clase `Nomina` que se adapten a las características de cada sucursal:

- En cada sucursal hay un único diseñador pero el número de fabricantes es distinto en cada sucursal. Por tanto, el número de fabricantes habrá que especificarlo en el constructor y posteriormente no podrá cambiarse.
- La forma de repartir el dinero es la siguiente: el diseñador se lleva una parte del total y el resto se reparte a partes iguales entre los fabricantes. En los apartados anteriores, por ejemplo, la parte que se llevaba el diseñador era $\frac{2}{5}$ y el resto ($\frac{3}{5}$) se repartía entre los tres fabricantes. La parte que el diseñador se lleva puede ser distinta entre las distintas sucursales ($\frac{2}{5}$, $\frac{1}{6}$, etc), pero no cambia nunca dentro de una misma sucursal. Por tanto, el porcentaje de ganancia ($\frac{2}{5}$, $\frac{1}{6}$, etc) habrá que especificarlo en el constructor y posteriormente no podrá cambiarse.

- Las retenciones fiscales de los fabricantes y diseñador son distintas. Además, se prevé que éstas puedan ir cambiando durante la ejecución del programa. Por lo tanto, no se incluirán como parámetros en el constructor.

El programa leerá los siguientes datos desde un fichero externo:

- El número de sucursales.
- Los siguientes valores por cada una de las sucursales:
 - Ingresos totales a repartir
 - Número de fabricantes
 - Parte que se lleva el diseñador
 - Retención fiscal del diseñador
 - Retención fiscal de los fabricantes

Por ejemplo, el siguiente fichero indica que hay dos sucursales. La primera tiene unos ingresos de 300 euros, 3 fabricantes, el diseñador se lleva 1/6, la retención del diseñador es del 20 % y la de cada fabricante un 18 %. Los datos para la segunda son 400 euros, 5 fabricantes, 1/4, 22 % y 19 %.

```
2
300 3 6 20 18
400 5 4 22 19
```

El programa tendrá que imprimir los salarios brutos y netos del diseñador y de los fabricantes por cada una de las sucursales, llamando a los métodos oportunos de la clase *Nomina*.

Finalidad: Diseño de una clase y trabajar con datos miembro constantes. Dificultad Media.

63. Se quiere construir una clase para representar la tracción de una bicicleta, es decir, el conjunto de estrella (engranaje delantero), cadena y piñón (engranaje trasero). Supondremos que la estrella tiene tres posiciones (numeradas de 1 a 3, siendo 1 la estrella más pequeña) y el piñón siete (numeradas de 1 a 7, siendo 1 el piñón más grande). La posición inicial de marcha es estrella = 1 y piñón = 1.

La clase debe proporcionar métodos para cambiar la estrella y el piñón, sabiendo que la estrella avanza o retrocede de 1 en 1 y los piñones cambian a saltos de uno o de dos. Si ha llegado al límite superior (inferior) y se llama al método para subir (bajar) la estrella, la posición de ésta no variará. Lo mismo se aplica al piñón.

Cree un programa principal que lea desde un fichero externo los movimientos realizados e imprima la situación final de la estrella y piñón. Los datos se leerán en el siguiente formato: tipo de plato (piñón o estrella) seguido del tipo de movimiento. Para codificar esta información se usarán las siguientes letras: E indica una estrella, P un piñón, S para subir una posición, B para bajar una posición, T para subir dos posiciones y C para bajar dos posiciones. T y C sólo se aplicarán sobre los piñones.

E S P S P S P S P C E S E B #

En este ejemplo los movimientos serían: la estrella sube, el piñón sube en tres ocasiones sucesivas, el piñón baja dos posiciones de golpe, la estrella sube y vuelve a bajar. Supondremos siempre que la posición inicial de la estrella es 1 y la del piñón 1. Así pues, la posición final será Estrella=1 y Piñón=2.

Mejorad la clase para que no permita cambiar la marcha (con la estrella o el piñón) cuando haya riesgo de que se rompa la cadena. Este riesgo se produce cuando la marcha a la que queremos cambiar es de la siguiente forma:

- Estrella igual a 1 y piñón mayor o igual que 5
- Estrella igual a 2 y piñón o bien igual a 1 o bien igual a 7
- Estrella igual a 3 y piñón menor o igual que 3

Finalidad: Diseñar la interfaz de una clase. Dificultad Media.

64. Recuperad la solución del ejercicio 56 de la Relación de Problemas II (Empresa). Re-escribid el programa principal usando una clase `Ventas` para gestionar los cálculos de las ventas realizadas. Únicamente se pide que se indiquen las cabeceras de los métodos públicos de la clase y las llamadas a éstos en el programa principal. No hay que implementar ninguno de los métodos.

Debe suponer que la clase gestionará las ventas de exactamente tres sucursales. Los códigos de dichas sucursales son enteros cualesquiera (no necesariamente 1, 2, 3, como ocurría en el ejercicio 56 de la Relación de Problemas II)

El programa principal sería de la siguiente forma:

```
Ventas ventas_empresa;
.....
while (identif_sucursal != TERMINADOR){
    cin >> cod_producto;
    cin >> unidades_vendidas;

    --> Actualiza el número de unidades
        vendidas de la sucursal leída
        llamando a un método de ventas_empresa

    cin >> identif_sucursal;
}

--> Obtener el identificador y el número de ventas
    de la sucursal ganadora llamando a un método
    de ventas_empresa
```

Finalidad: Diseño de una clase. Dificultad Media.

RELACIÓN DE PROBLEMAS IV. Funciones y Clases

65. Implementar los métodos de la clase *Ventas* del ejercicio anterior.

Finalidad: Diseño de una clase. Dificultad Media.

66. Implemente una clase para representar un número complejo. Un complejo se define como un par ordenado de números reales (a, b) , donde a representa la parte real y b la parte imaginaria. Construya un programa principal que lea la parte real e imaginaria, cree el objeto e imprima el complejo en la forma $a + bi$.

Por ahora no podemos implementar métodos para sumar, por ejemplo, dos complejos. Lo veremos en el último tema.

67. Una empresa quiere gestionar las nóminas de sus empleados. El cómputo de la nómina se realiza en base a los siguientes criterios:

- a) Hay cuatro tipos de categorías laborales: Operario, Base, Administrativo y Directivo.
- b) Se parte de un salario base que depende de la antigüedad del trabajador y de su categoría laboral. Para la categoría Operario, el salario base es de 900 euros, 1100 el puesto Base, 1200 los Administrativos y 2000 los Directivos. Dicho salario base se incrementa con un tanto por ciento igual al número de años trabajados.
- c) Los trabajadores tienen complementos en su nómina por el número de horas extraordinarias trabajadas. La hora se paga distinta según la categoría: 16 euros por hora para los operarios, 23 para el puesto Base, 25 los Administrativos y 30 los Directivos. Además, al complemento que sale al computar el número de horas extraordinarias, se le aplica una subida con un tanto por ciento igual al número de años trabajados.

Se pide diseñar la interfaz de una clase (también hay que incluir los datos miembro privados) para poder trabajar con esta información. No se pide implementar la clase, únicamente determinar la interfaz.

Finalidad: Diseñar la interfaz de una clase. Dificultad Media.

68. Implementad la clase del ejercicio 67 de esta relación de problemas. *Dificultad Media.*

69. Recuperad la solución del ejercicio 65 (Empresa) y modificadlo convenientemente para que los datos miembros que referencia los identificadores de las sucursales sean constantes.

Finalidad: Trabajar con datos miembros constantes. Dificultad Baja.

70. La sonda Mars Climate Orbiter fue lanzada por la NASA en 1998 y llegó a Marte el 23 de septiembre de 1999. Lamentablemente se estrelló contra el planeta ya que se acercó demasiado. El error principal fue que los equipos que desarrollaron los distintos módulos de la sonda usaron sistemas de medida distintos (el anglosajón y el métrico). Cuando un componente software mandaba unos datos en millas (o libras),

otro componente software los interpretaba como si fuesen kilómetros (o Newtons). El problema se habría arreglado si todos hubiesen acordado usar el mismo sistema. En cualquier caso, cada equipo se encuentra más a gusto trabajando en su propio sistema de medida. Por tanto, la solución podría haber pasado por que todos utilizasen una misma clase para representar distancias (idem para fuerzas, presión, etc), utilizando los métodos que les resultasen más cómodos.

Para ello, se pide construir la clase `Distancia` que contendrá métodos como `SetKilometros`, `SetMillas`, etc. Internamente se usará un único dato miembro privado llamado `kilometros` al que se le asignará un valor a través de los métodos anteriores, realizando la conversión oportuna (una milla es 1.609344 kilómetros). La clase también proporcionará métodos como `GetKilometros` y `GetMillas` para lo que tendrá que realizar la conversión oportuna (un kilómetro es 0.621371192 millas).

Observad que la implementación de la clase podría haber utilizado como dato miembro privado, una variable `millas`, en vez de `kilómetros`. Esto se oculta a los usuarios de la clase, que sólo ven los métodos `SetKilometros`, `SetMillas`, `GetKilometros` y `GetMillas`.

Cread un programa principal que pida algunos datos y muestre los valores convertidos.

Nota. Otro de los fallos del proyecto fue que no se hicieron suficientes pruebas del software antes de su puesta en marcha, lo que podría haber detectado el error. Esto pone de manifiesto la importancia de realizar una batería de pruebas para comprobar el correcto funcionamiento del software en todas las situaciones posibles. Esta parte en el desarrollo de un proyecto se le conoce como *pruebas de unidad (unit testing)*

Finalidad: Trabajar con una clase como una abstracción de un concepto. Dificultad Baja.

71. Construid una clase llamada `MedidaAngulo` que represente una medida de un ángulo. Al igual que se hizo en el ejercicio 70, la clase aceptará datos que vengan de alguna de las siguientes formas: número de grados con decimales (real); número de radianes (entero); número de segundos (entero); número de grados, minutos y segundos (en un struct que represente estos tres valores)

Dificultad Baja.

72. Recupere la solución del ejercicio 56 (Parking con una clase) Defina un struct llamado `InstanteTiempo` para almacenar la hora, minutos y segundos que constituyen un instante de tiempo. Cambie la definición de la función `MinutosEntreInstantes` y el programa principal para que trabaje con este tipo struct.

Finalidad: Trabajar con funciones y el tipo struct. Dificultad Baja.

73. Defina un struct llamado `CoordenadasPunto2D` para representar un par de valores reales correspondientes a un punto en \mathbb{R}^2 .

Defina una función `DistanciaEuclidea` para que calcule la distancia entre dos puntos cualesquiera. Cree un programa principal que vaya leyendo 4 valores reales

desde teclado representando las coordenadas de dos puntos y calcule la distancia euclídea entre ellos. Cada vez que se lean los cuatro valores se le preguntará al usuario si quiere seguir introduciendo datos o no (con las opciones 's'/'n').

Ejemplo de entrada:

```
s 3.1 4.2 5.3 6.4 j k s 2.1 4.9 -3.2 0 s 1 5 1 5 n
```

— Salida correcta: 3.11127 7.21803 0

Finalidad: Trabajar con funciones y el tipo struct. Dificultad Baja.

74. **[Parking]** Recupere la solución del ejercicio 33 de la relación de problemas III (párking). Re-escribalo definiendo la clase `TarifadorParking` para calcular la tarifa.

La clase debe permitir cualquier número de tramos. Para ello, haga lo siguiente:

- Defina dos vectores como datos miembro de la clase. En uno almacenaremos los límites de los tramos y en el otro la correspondiente tarifa.
- Defina el siguiente método:

```
void AniadeTramo(double limite_superior_tramo,  
                double tarifa_tramo)
```

Este método se llamará tantas veces como tramos tengamos.

- Defina el método `GetTarifa` para calcular la tarifa según el número de minutos de un estacionamiento.
- Cree dos objetos de la clase `TarifadorParking` (uno para cada parking) y modifique adecuadamente el programa principal para calcular las tarifas a partir de los métodos de los objetos.

Mantenga la definición de la función `MinutosEntreInstantes` para calcular los minutos que hay entre dos instantes.

Finalidad: Diseñar las cabeceras de los métodos que acceden a las componentes del vector. Dificultad Baja.

75. **[Fibonacci]** La sucesión de Fibonacci de orden n es una secuencia de números en la que los dos primeros son el 0 y el 1. A partir del tercero, los elementos se calculan como la suma de los n anteriores, si ya hay n elementos disponibles, o la suma de todos los anteriores si hay menos de n elementos disponibles.

Por ejemplo, la sucesión de Fibonacci de orden 4 sería la siguiente:

0, 1, 1, 2, 4, 8, 15, 29, ...

Defina una clase llamada `Fibonacci`. Para almacenar los enteros, se usará un vector de enteros. Al constructor se le pasará como parámetro el valor de n . Defina los siguientes métodos:

- `int GetBase()` para obtener el valor de n .
- `void CalculaPrimeros(int tope)` para que calcule los $tope$ primeros elementos de la sucesión.
- `int TotalCalculados()` que devuelva cuántos elementos hay actualmente almacenados (el valor $tope$ del método anterior)
- `int k_esimo(int k)` para que devuelva el elemento k -ésimo de la sucesión.

Escriba un programa que lea los valores de dos enteros, n y k y calcule, almacene y muestre por pantalla los k primeros términos de la sucesión de Fibonacci de orden n :

```
.....
Fibonacci fibonacci(n);

fibonacci.CalculaPrimeros(k);
tope = fibonacci.TotalCalculados();    // tope = k

for (int i=0; i<tope; i++)
    cout << fibonacci.k_esimo(i) << " ";
```

Dificultad Media.

76. [Eratóstenes] ([Examen Septiembre 2012](#)) La **criba de Eratóstenes** (Cirene, 276 a. C. Alejandría, 194 a. C.) es un algoritmo que permite hallar todos los números primos menores que un número natural dado n .

El procedimiento consiste en escribir todos los números naturales comprendidos entre 2 y n y *tachar* los números que *no* son primos de la siguiente manera: el primero (el 2) se declara primo y se tachan todos sus múltiplos; se busca el siguiente número entero que no ha sido tachado, se declara primo y se procede a tachar todos sus múltiplos, y así sucesivamente. El proceso para cuando el cuadrado del número entero es mayor o igual que el valor de n .

El programa debe definir una clase llamada `Eratostenes` que contendrá:

- Como dato miembro debe declarar un vector privado `primos` tal que en la componente k se almacenará el primo k -ésimo ($[2, 3, 5, 7, \dots]$). El cómputo de los primos se hará en el siguiente método.
- El método `void CalculaHasta(int n)` calcula los primos menores que n . Cuando se ejecute el método, se calcularán todos los primos menores que n , según el método de Eratóstenes descrito anteriormente.
Para realizar esta tarea, tendrá que definir un vector local al método con todos los números menores que n y procederá a *tachar* los no primos según el algoritmo de Eratostenes. Los números no tachados serán los primos y serán los que almacene en el dato miembro `primos`.

- El método `int TotalCalculados()` devuelva cuántos primos hay actualmente almacenados.
- `int k_esimo(int k)` para que devuelva el k -ésimo primo.

El programa principal quedaría de la forma:

```
Eratostenes primos;
int n = 100; int num_primos;

primos.CalculaHasta(n);
num_primos = primos.TotalCalculados();

for (int i=0; i<num_primos; i++)
    cout << primos.k_esimo(i) << " ";
```

Dificultad Media.

77. [Palabras en una frase] (*Examen Septiembre Doble Grado 2013*) Defina una clase `Frase` para almacenar un conjunto de caracteres (similar a la clase `SecuenciaCaracteres`). Defina un método para localizar la k -ésima palabra.

Una palabra es toda secuencia de caracteres delimitada por espacios en blanco a izquierda y derecha. La primera palabra no tiene por qué tener espacios a su izquierda y la última no tiene por qué tener espacios a su derecha. Puede haber varios caracteres en blanco consecutivos.

Si k es mayor que el número de palabras, se considera que no existe tal palabra.

Por ejemplo, si la frase es `{' ', ' ', 'h', 'i', ' ', ' ', 'b', 'i', ' ', ' '}`. Si $k = 1$, la posición es 2. Si $k = 2$ la posición es 6. Si $k = 3$ la posición es -1.

Si la frase fuese `{'h', 'i', ' ', 'b', 'i', ' ', ' '}`, entonces si $k = 1$, la posición es 0. Si $k = 2$ la posición es 3. Si $k = 3$ la posición es -1.

Dificultad Media.

78. [Palabras en una frase -continuación-] Sobre el ejercicio 77, añada los siguientes métodos:

- `void EliminaBlancosIniciales()` para borrar todos los blancos iniciales.
- `void EliminaBlancosFinales()` para borrar todos los blancos finales.
- `int NumeroPalabras()` que indique cuántas palabras hay en la frase.
- `void BorraPalabra(int k_esima)` para que borre la palabra k -ésima.
- `void MoverPalabraFinal(int k_esima)` para desplazar la palabra k -ésima al final de la frase.

Dificultad Media.

79. [Búsqueda por interpolación] (*Examen Prácticas Septiembre 2016*) Implemente la **Búsqueda por Interpolación** en la clase `SecuenciaCaracteres`. El método busca un valor buscado entre las posiciones `izda` y `dcha` y recuerda a la *búsqueda binaria* porque requiere que el vector en el que se va a realizar la búsqueda esté ordenado y en cada consulta sin éxito se descarta una parte del vector para la siguiente búsqueda.

La diferencia fundamental con la búsqueda binaria es la manera en que se calcula el elemento del vector que sirve de referencia en cada consulta (que ocupa la posición `pos`). Ya no es el que ocupa la posición central del subvector en el que se efectúa la búsqueda (el delimitado únicamente por `izda` y `dcha`), sino que depende también del contenido de esas casillas, de manera que `pos` será más cercana a `dcha` si buscado es más cercano a `v[dcha]` y más cercana a `izda` si buscado es más cercano a `v[izda]`. En definitiva, se cumple la relación:

$$\frac{\text{pos} - \text{izda}}{\text{dcha} - \text{izda}} = \frac{\text{buscado} - v[\text{izda}]}{v[\text{dcha}] - v[\text{izda}]}$$