



Fundamentos de Programación.

Guión de Prácticas.

Curso 2016/2017

Para cualquier sugerencia o comentario sobre este guión de prácticas, por favor, enviad un e-mail a Juan Carlos Cubero (JC.Cubero@decsai.ugr.es)

"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 de decsai, en el plazo que el profesor determine. Para ello, el alumno debe entrar en el acceso identificado de decsai, seleccionar **Entrega Prácticas** y a continuación la práctica correspondiente a la semana en curso. 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. Además, al final de la semana, las soluciones estarán disponibles en decsai. Es muy importante que el alumno revise estas soluciones y las compare con las que él había diseñado.

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.

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.

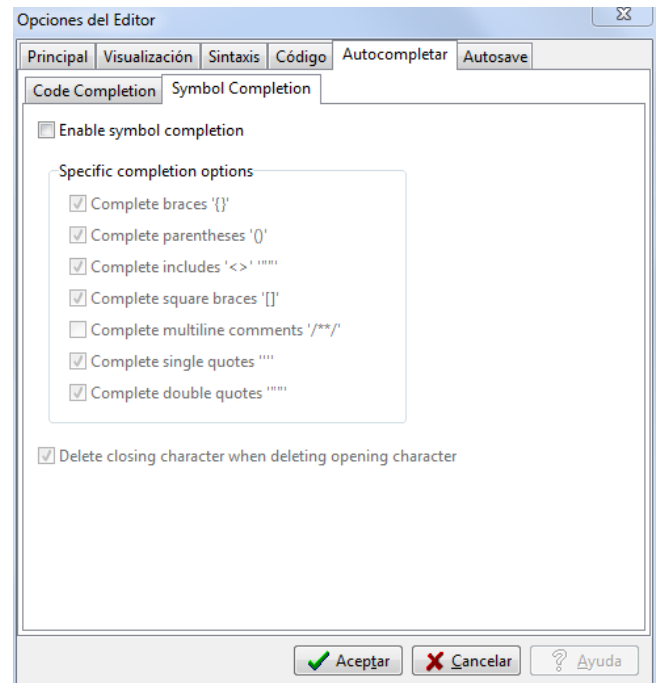
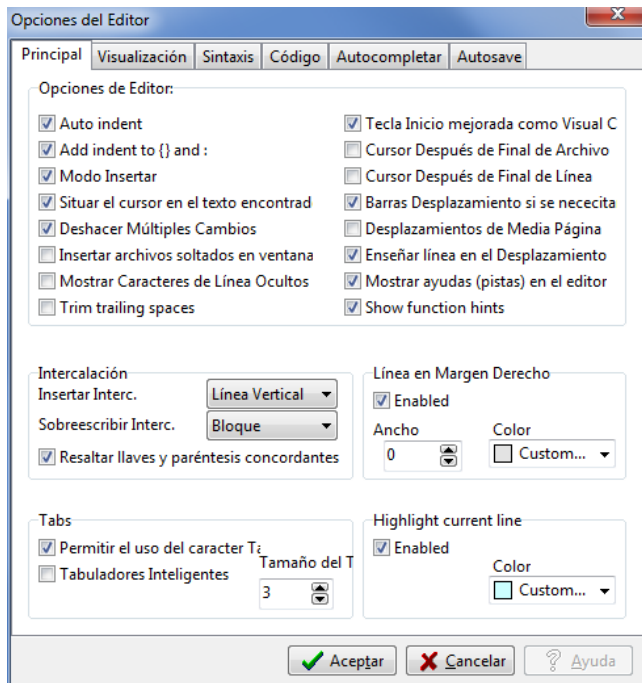
Instalación de Orwell Dev C++ en nuestra casa

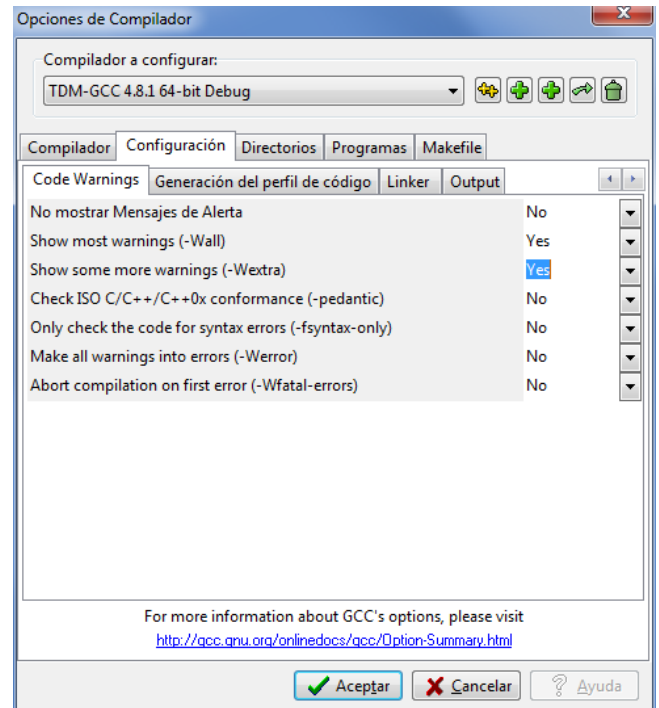
El entorno de desarrollo que usaremos será Orwell Dev C++. Puede descargarse desde la página:

http://sourceforge.net/projects/orwelldvcpp/?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
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
```

Para que podamos ver correctamente dichos caracteres, debemos seguir los siguientes pasos:

1. 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.

2. Debemos cargar la página de códigos correspondiente al alfabeto latino. Para ello, tenemos varias alternativas:

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

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

- b) Lo siguiente es sólo para el SO Microsoft Windows. 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 1

Tal y como se ha indicado al inicio de este documento, en la primera semana de clase se publica la sesión 1. En esta sesión se detalla las tarea y ejercicios que el alumno debe resolver en su casa durante la primera semana y que defenderá en la siguiente. Esta es la única sesión en la que el alumno no tendrá que entregar las soluciones a través de de csa.i.

► **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 primera semana de clase, el alumno debería instalar en su casa el compilador Orwell Dev C++. Consulte la sección de Instalación (página 3) de este guión.

Actividad: Resolución de problemas.

Resuelva en papel los ejercicios siguientes de la relación de problemas I:

- *Obligatorios:*
 - 1 (Asignaciones secuenciales)
 - 3 (Circunferencia)
 - 4 (Subir sueldo)
 - 5 (Interés bancario)
- *Opcionales:*
 - 7 (Triles: intercambiar variables)

Actividad: Preparar la clase de prácticas de la semana próxima.

Realice una lectura rápida de las actividades a realizar la semana próxima durante las horas de prácticas en las aulas de ordenadores (ver páginas siguientes)

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 durante las clases de prácticas en la segunda semana de clase.

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.

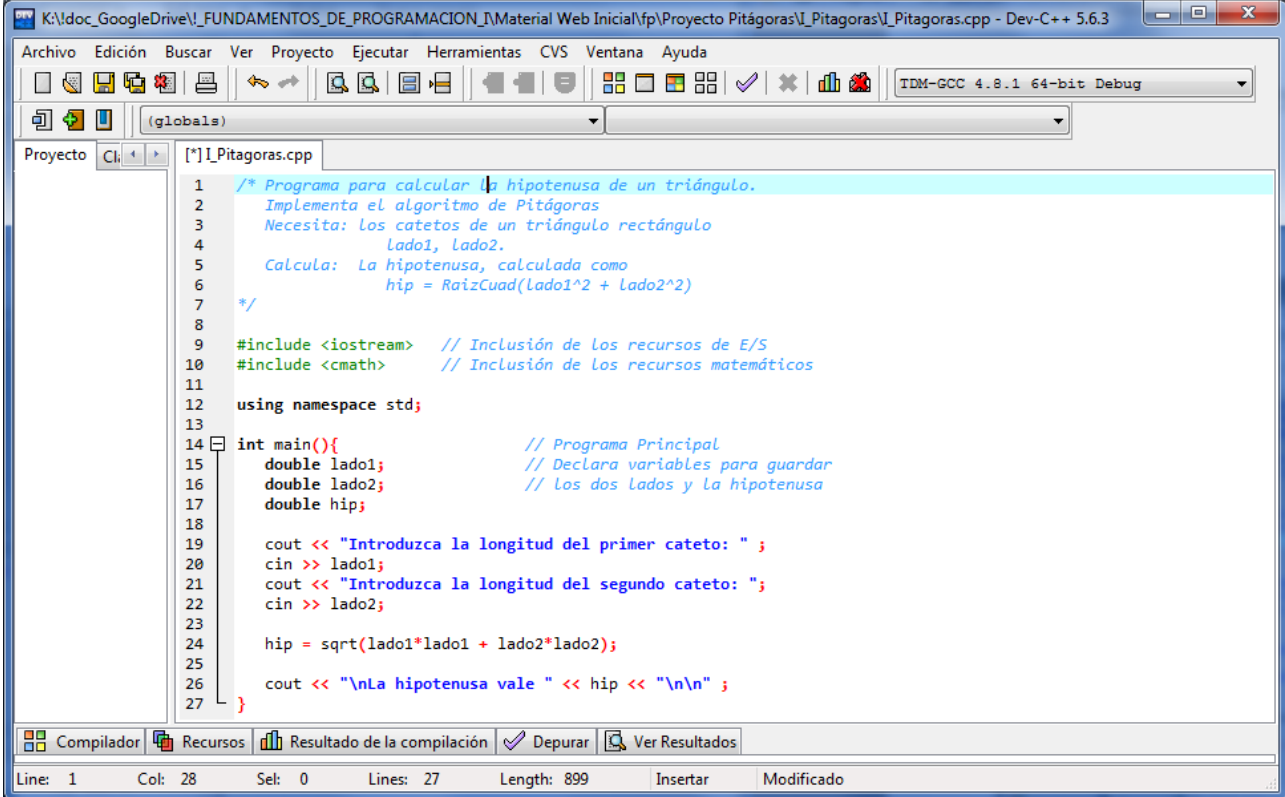
El primer programa

Copiando el código fuente

Descargue de decsai el fichero I_Pitagoras y cópielo en su carpeta local (dentro de U:\FP). También puede bajarse el fichero desde la siguiente dirección:

http://decsai.ugr.es/~carlos/FP/I_Pitagoras.cpp

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.

The image shows a C++ code snippet with several handwritten annotations in blue and red ink. On the left, blue annotations group lines of code: 'Declaraciones' for the variable declarations, 'Entradas datos' for the input statements, 'Computos' for the calculation, and 'Salida Resultados' for the output and pause statements. Red annotations highlight specific formatting rules: 'líneas en blanco' points to empty lines between code blocks; 'espacios en blanco' points to spaces around operators like '<<', '>>', and '='; and a red circle around the comments is labeled 'Comentarios separados visualmente del código'.

```
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

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.

Introducíd ahora los valores pedidos en el ejemplo de Pitágoras y comprobad 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, hacemos lo siguiente:

1. Cerramos Orwell Dev C++.
2. Abrid una ventana de Mi PC.
3. Situarse en la carpeta que contiene el ejecutable.
4. Haced 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.

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.

Cargad el fichero `I_Pitagoras.cpp`. Quitadle una 'u' a alguna aparición de `cout`. Intentad 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:

- Comentad la línea de cabecera `#include <iostream>` desde el principio. El compilador no reconocerá las apariciones de `cin` o `cout`.
- Quitad 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. Cambiad algún punto y coma por cualquier otro símbolo
2. Cambiad `double` por `dpuble`
3. Cambiad la línea `using namespace std;` por `using namespace STD;`
4. Poned en lugar de `iostream`, el nombre `iotream`.
5. Borrard alguno de los paréntesis de la declaración de la función `main`
6. Introducid algún identificador incorrecto, como por ejemplo `cour`
7. Usad una variable no declarada. Por ejemplo, en la definición de variables cambiad el nombre a la variable `lado1` por el identificador `lado11`.
8. Borrard alguna de las dobles comillas en una constante de cadena de caracteres, tanto las comillas iniciales como las finales.
9. Borrard alguna de las llaves que delimitan el inicio y final del programa.
10. Borrard la línea `using namespace std;` (basta con comentarla con `//`)
11. Cambiad un comentario iniciado con `//`, cambiando las barras anteriores por las siguientes `\\`
12. Cambiad la aparición de `<<` en `cout` por las flechas cambiadas, es decir, `>>`. Haced lo mismo con `cin`.
13. Suprimid 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`, haced lo siguiente:

- Cambiad la sentencia
`sqrt(lado1*lado1 + lado2*lado2)` por:
`sqrt(lado1*lado2 + lado2*lado2)`
Ejecutad 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, declarad tres variables **ENTERAS** (tipo `int`) `resultado`, `numerador` y `denominador`. Asignadle cero a `denominador` y 7 a `numerador`. Asignadle a `resultado` la división de `numerador` entre `denominador`. Imprimid 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`.

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

TDM-GCC ... Debug

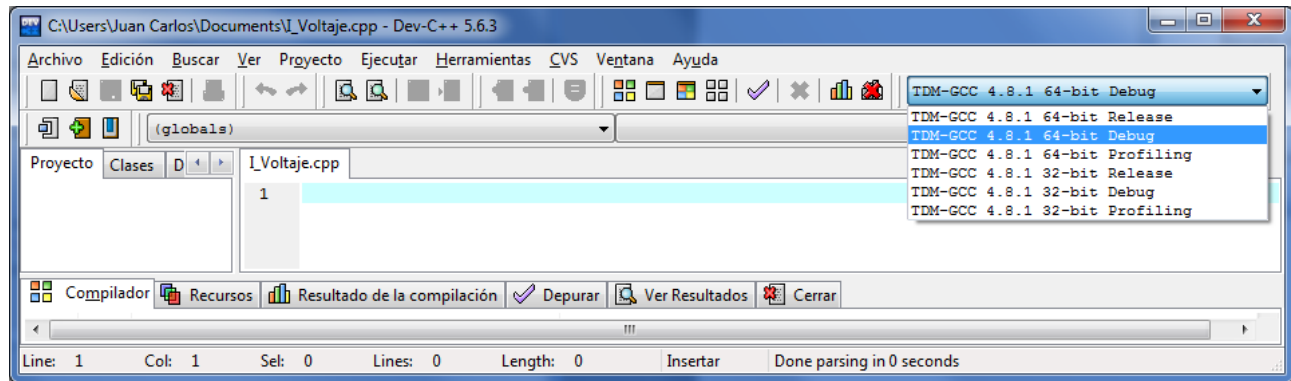


Figura 3: Creación de un programa nuevo

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.

Resuelva el ejercicio 6 (Diferencia entre instantes de tiempo)

Este ejercicio no tiene que entregarlo, pero lo puede guardar en su unidad en decsa.i. La solución también estará disponible en decsa.i al final de la semana.

Sesión 2

Tipos básicos y operadores

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios (disponible en decsai)

8 (Conversión sistema métrico)

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la relación I. Recuerde que antes del domingo por la noche hay que subir las soluciones a decsai, tal y como se explica en la página 2. Debe subir un fichero llamado `sesion2.zip` que incluya todos los ficheros `cpp` (pero **no** los `.exe`).

- **Obligatorios:**

- 9 (Dos subidas de sueldo)

- 10 (Gaussiana)

- 11 (Uso de constantes)

- 12 (Población)

- **Opcionales:**

- 13 (Pinta dígitos)

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.disfrutalasmatematicas.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, el resto de alumnos 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 (disponible en [decsai](#))

20 (Índice de mayúscula)

21 (Pasar de carácter a entero)

22 (Expresiones diversas)

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la relación de Problemas I:

- *Obligatorios:*

- 23 (Horas, minutos, segundos)

- 24 (Intercambiar tres variables)

- 25 (Pasar de mayúscula a minúscula)

- 27 (Expresiones lógicas)

- *Opcionales:*

- 28 (Precisión y desbordamiento)

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 `decsai`.

Para comprobarlo, copiad localmente el fichero `II_cin` disponible en `decsai`. Observad el código del programa y ejecutadlo. Para introducir los datos pedidos (un entero y dos caracteres) 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, cread un fichero de texto con un entero y dos caracteres. Separad estos tres datos con varios espacios en blanco. Seleccionad con el ratón los tres y copiadlos al portapapeles (Click derecho-Copiar). Ejecutad el programa y cuando aparezca la consola del sistema haced click derecho sobre la ventana y seleccionad `Editar-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, descargad desde `decsai` el fichero `II_cin_datos_entrada.txt` y copiadlo dentro de la misma carpeta en la que se ha descargado el programa `II_cin`. 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*.

Resuelva los ejercicios siguientes de la Relación de Problemas I, página **RP-I.1**:

26 (Expresiones lógicas)

29 (Elección tipo de dato)

Estos ejercicios no han de entregarse en `decsai`.

Sesión 4

Estructura condicional

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios (disponible en decsai)

- 1 (Media aritmética)
- 2 (Pasar de mayúscula a minúscula)
- 3 (Se dividen)
- 4 (Pasar de mayúscula a minúscula y viceversa)

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:**
 - 5 (Subir sueldo -una única subida salarial-)
 - 6 (Subir sueldo -dos subidas salariales compatibles-)
 - 7 (Tres valores ordenados)
 - 8 (Año bisiesto)
- **Opcionales:**
 - 10 (Párking)

► 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.

Vamos a emplear como base para esta práctica el ejercicio 1 (media aritmética con enteros) de la Relación de Problemas II (página [RP-II.1](#)).

En primer lugar, crearemos la carpeta `II_Media_int` en `U:\FP` y copiaremos en ella el fichero fuente `II_Media_int.cpp` (disponible en [decsai](#))

Forzad los siguientes errores en tiempo de compilación, para habituarnos a los mensajes de error ofrecidos por el compilador:

- Suprima los paréntesis de alguna de las expresiones lógicas de la sentencia `if`
- Quite la llave abierta de la sentencia condicional (como únicamente hay una sentencia dentro del `if` no es necesario poner las llaves, pero añadimos las llaves a cualquier condicional para comprobar el error que se produce al eliminar una de ellas)
- Quite la llave cerrada de la sentencia condicional

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 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 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.

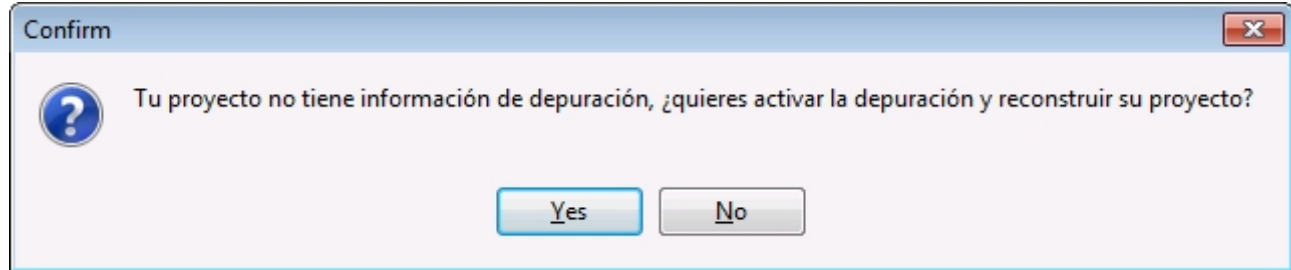


Figura 4: Ventana emergente que aparece cuando la configuración actual del compilador no permite tareas de depuración

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.

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

2. Comenzar la depuración:

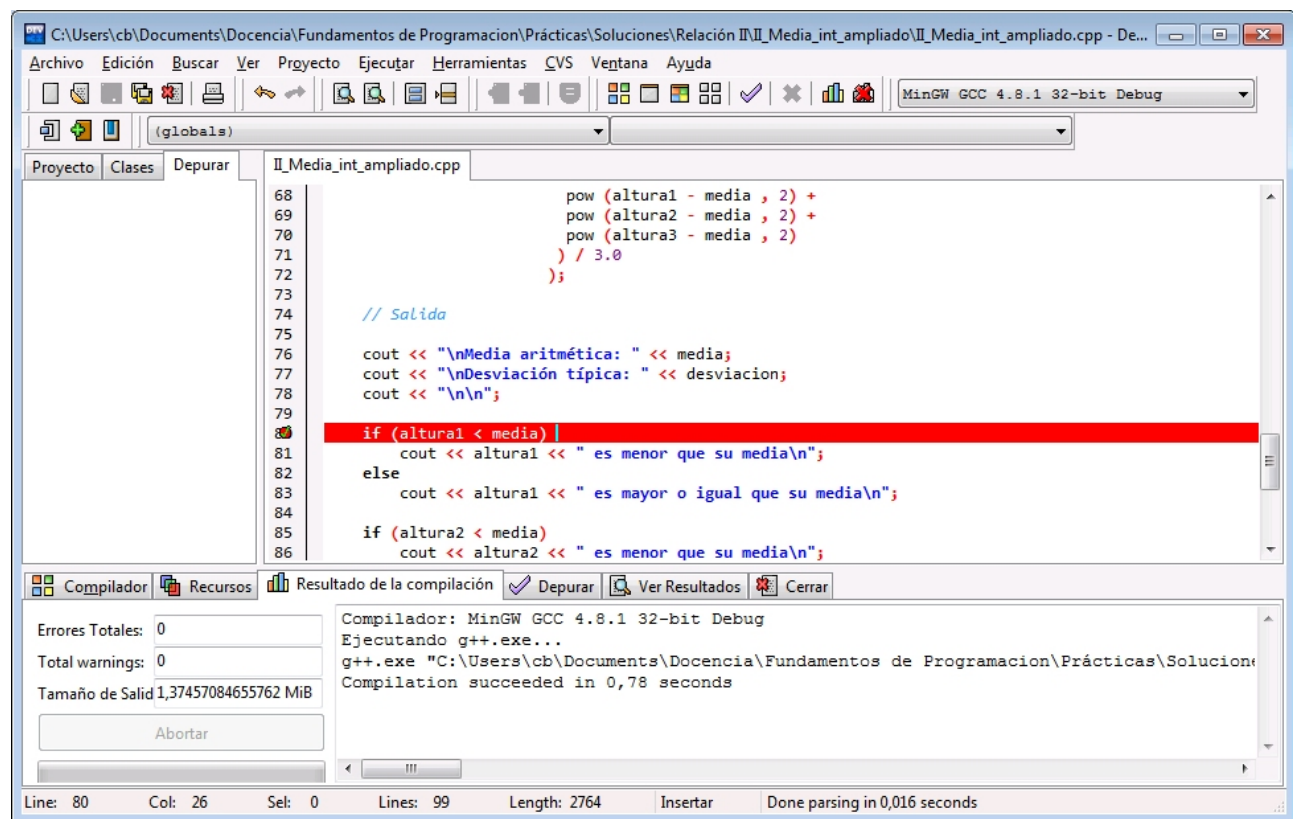



Figura 5: Se ha activado un punto de ruptura

- 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)

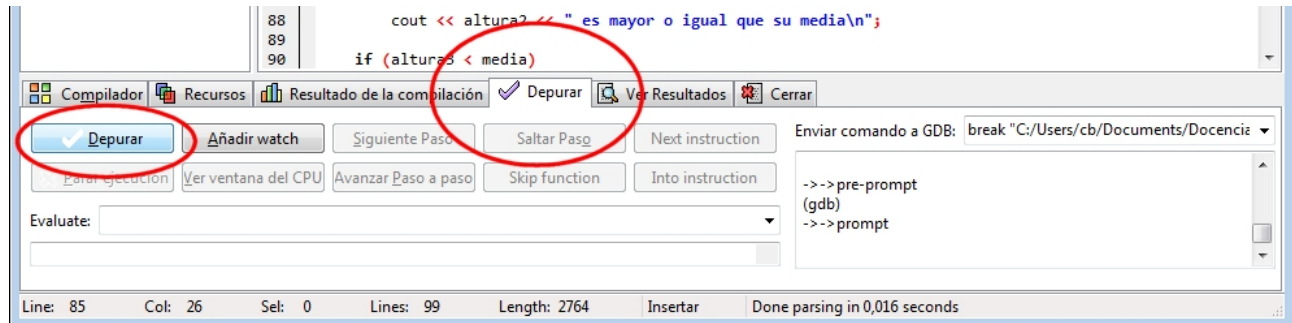


Figura 6: Inicio del proceso de depuración

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

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.

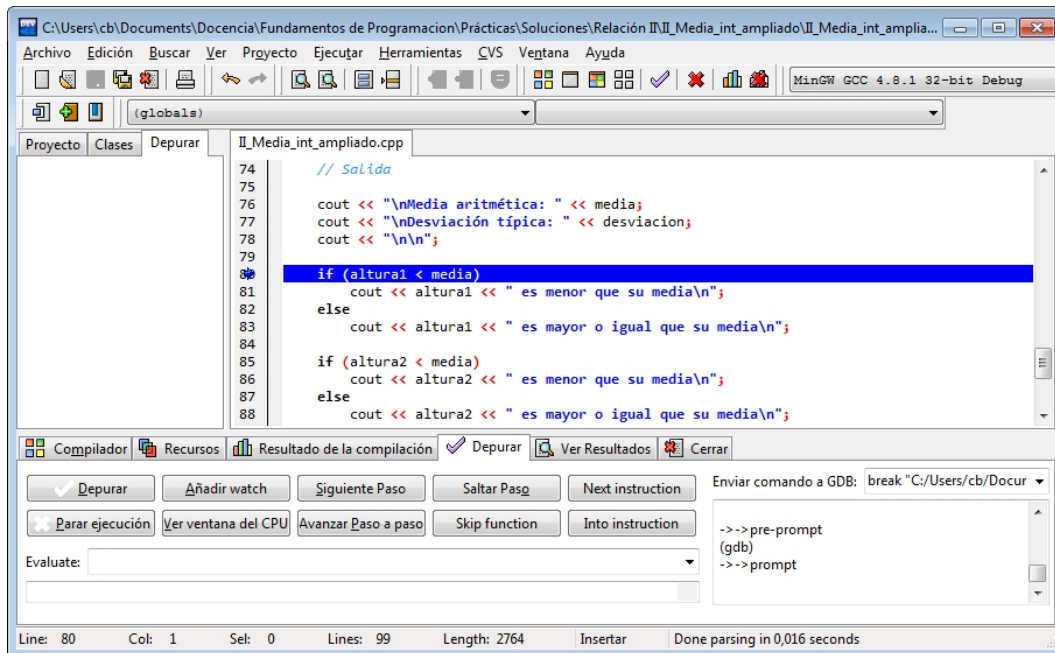


Figura 7: Inicio del proceso de depuración

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 (figura 8). 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.

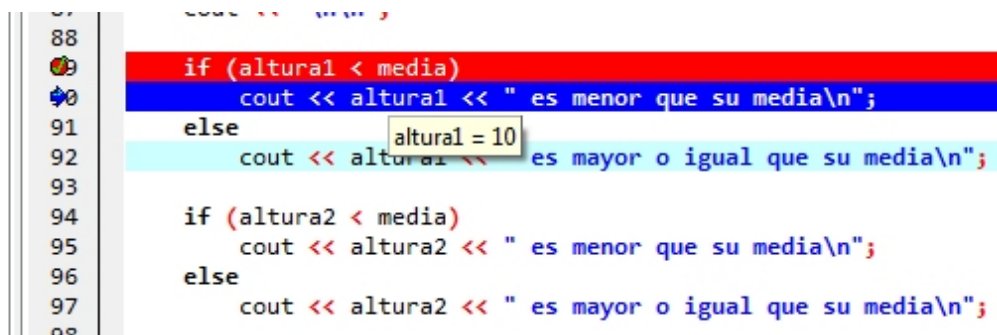


Figura 8: Inspeccionando el valor de una variable

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 (figura 9.A). Al seleccionar OK aparece la información de esa variable en el Explorador de Proyectos/Clases (figura 9.B).

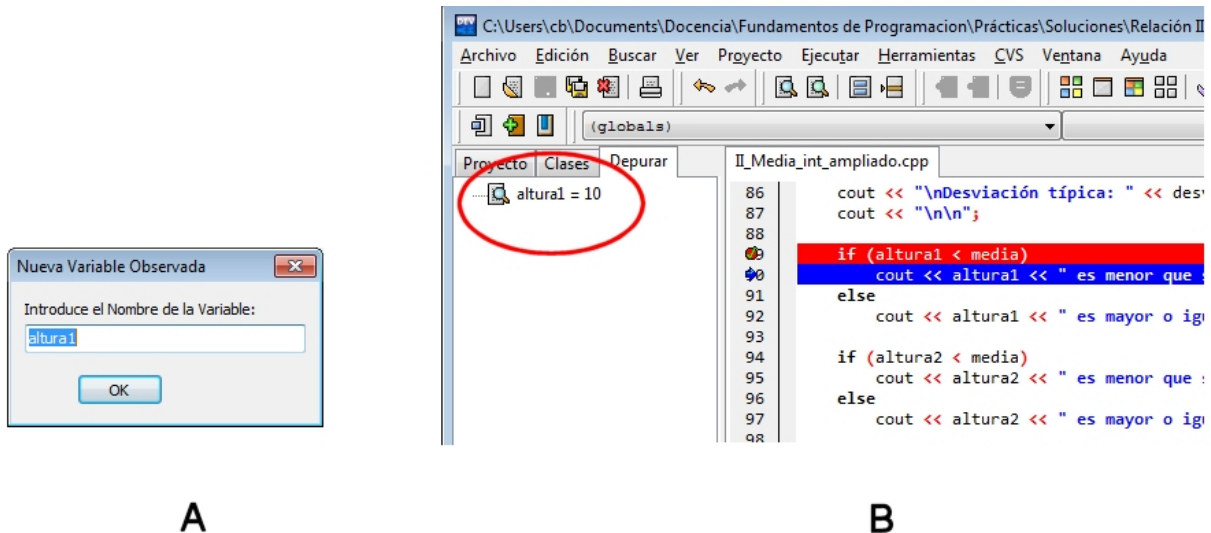


Figura 9: Añadiendo una variable para su inspección permanente

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**.

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".



Resuelva el ejercicio 9 (subida salarial excluyente)

La solución de este ejercicio no hay que entregarla.

Sesión 5

Estructura Repetitiva. Bucles `while`

► **Actividades a realizar en casa**

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios (disponible en `decsai`)

13 (Tres valores ordenados separando E/S y C con un `bool`)

15 (Mayúscula a minúscula y viceversa separando E/S y C con un enumerado)

19 (Divisores de un número)

Actividad: Resolución de problemas.

- **Obligatorios:**

Ejercicios sobre condicionales:

11 (Subida salarial, con condicional anidado)

14 (Año bisiesto separando E/S y C)

16 (Tres valores ordenados separando E/S y C con un enumerado)

Ejercicios sobre bucles:

22 (Gaussiana)

23 (Población)

- **Opcionales:**

24 (Leer valores dentro de un rango)

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

Resuelva el ejercicio 29 (Mayor secuencia ascendente de temperaturas) Este ejercicio estará incluido en la próxima sesión.

Sesión 6

Actividad: Lectura de programas resueltos.

Lea la solución de los siguientes ejercicios (disponible en decsai)

33 (RLE)

38 (Narcisista)

► **Actividades a realizar en casa**

Actividad: Resolución de problemas.

Resuelva los siguientes problemas de la Relación de Problemas II.

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

- *Obligatorios:*

- 28 (Lectura de los datos de la subida salarial)

- 29 (Mayor secuencia ascendente de temperaturas)

- 30 (Pinta dígitos generalizado)

- *Opcionales:*

- 31 (Multiplicación rusa)

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

Resuelva los ejercicios 40 y 46 (Factorial y sumatoria de factoriales) Estos ejercicios estarán incluidos en la próxima sesión.



Fundamentos de Programación.

Relaciones de Problemas.

RELACIÓN DE PROBLEMAS I. Introducción a C++

1. Indique cuál sería el resultado de las siguientes operaciones:

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

salario_base = 1000;
salario_final = salario_base;

incremento = 200;
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. 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. 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:

$$\text{área circ} = \pi r^2 \quad \text{long. circunf} = 2\pi r$$

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

RELACIÓN DE PROBLEMAS I. Introducción a C++

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

4. 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.

5. 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`.

RELACIÓN DE PROBLEMAS I. Introducción a C++

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.

6. 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.

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.

7. 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.

8. 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

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.

9. Recupere la solución del ejercicio 4 (Subir sueldo usando la variable `salario_final`) 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

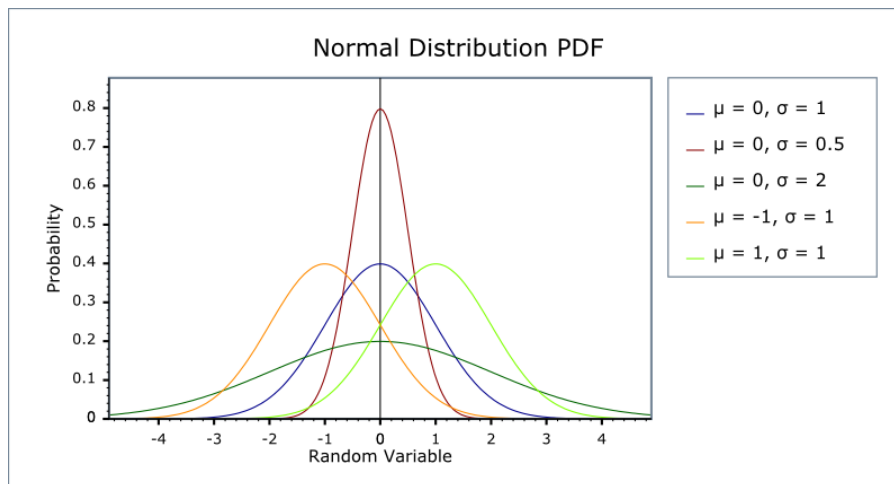
Ejemplo de entrada: 0 — Salida correcta: 0 0

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

10. 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 representar el número π defina una constante con el valor 3.1416 -no use una coma para separar la parte entera de la decimal, sino un punto-

Para realizar las operaciones indicadas, debe utilizar las siguientes funciones de la biblioteca `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.

RELACIÓN DE PROBLEMAS I. Introducción a C++

11. Re-escriba las soluciones de los ejercicios 3, 9, 6, 10 (circunferencia, subir sueldo, diferencia de tiempo, gaussiana) utilizando datos de tipo constante en aquellos sitios donde considere oportuno.
12. 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.

13. 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 tres dígitos, como por ejemplo 351. En este caso, la salida sería:

3 5 1

Finalidad: Ejemplo de asignación acumulada.

Dificultad Media.

14. 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.

15. 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 la biblioteca `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>

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

16. 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*).

Escribía 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.

17. 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.

18. 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.

19. Realice el ejercicio del cálculo de la desviación típica, pero cambiando el tipo de dato de las variables x_i a `int`.

Nota: Para no tener problemas en la llamada a la función `pow` (en el caso de que se haya utilizado para implementar el cuadrado de las diferencias de los datos con la media), obligamos a que la base de la potencia sea un real multiplicando por 1.0, por lo que la llamada quedaría en la forma `pow(base*1.0, exponente)`

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

20. 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.

21. 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.

22. 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.

23. 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.

24. Se quiere generalizar el ejercicio 7 que intercambiaba el valor de dos variables al caso de tres variables. Construya un programa que declare las variables x, y 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.

25. 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` de la biblioteca `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.

26. 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 && z > y )
```

27. 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.

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

RELACIÓN DE PROBLEMAS I. Introducción a C++

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

30. 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.

31. 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.

32. 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:

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

Para calcular el cuadrado no puede usar ninguna función de la biblioteca `cmath`.

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

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

Algunas funciones de `cmath`

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

$\cos(x) \rightarrow \cos(x)$

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

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

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

Dificultad Baja.

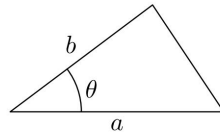
34. Dos locomotoras parten de puntos distintos avanzando en dirección contraria sobre la misma vía. Se pide redactar un programa para conocer las distancias que habrán recorrido ambas locomotoras antes de que choquen teniendo en cuenta que la primera locomotora viaja a una velocidad constante V_1 , que 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.

35. 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.

36. 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. Amplie el ejercicio 15 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.*

2. 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 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.*

3. Realice un programa en C++ 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.*

4. Queremos modificar el ejercicio 2 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.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

- 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.

5. 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.

6. Recupere la solución del ejercicio 5 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%

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

7. 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.

8. 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.

9. Modifique la solución del ejercicio 6 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%

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.

10. 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

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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 6 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.

11. Modifique la solución del ejercicio 6 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.

12. 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.

13. Modifique la solución del ejercicio 7 (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.

14. Modifique la solución del ejercicio 8 (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.

15. Modifique la solución al ejercicio 4 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.

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

16. Modifique el ejercicio 7 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: Separar E/S y C. Usar el tipo enumerado para detectar cuándo se produce una situación determinada. Dificultad Baja.

17. 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:

RELACIÓN DE PROBLEMAS II. Estructuras de Control

- 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:
 - Se sube otros dos puntos la retención fiscal si la renta bruta es menor de 20.000 euros
 - Se sube otros 2.5 puntos la retención fiscal a los casados con renta bruta superior a 20.000 euros
 - 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.

18. 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. Si el destino está a menos de 200 kilómetros, el precio final es la tarifa inicial. Para destinos a más de 200 Km, se suman 10 céntimos por cada kilómetro de distancia al destino (a partir del Km 200). En una campaña de promoción se va a realizar una rebaja lineal de 15 euros a todos los viajes. Además, se pretenden añadir otras rebajas y se barajan las siguientes alternativas de políticas de descuento:
- a) Una rebaja del 3 % en el precio final, para destinos a más de 600Km.
 - b) Una rebaja del 4 % en el precio final, para destinos a más de 1100Km. En este caso, no se aplica el anterior descuento.
 - c) Una rebaja del 5 % si el comprador es cliente previo de la empresa.

Cree un programa para que lea el número de kilómetros al destino y si el billete corresponde a un cliente previo de la empresa. Calcular el precio final del billete con las siguientes políticas de descuento:

- Aplicando c) de forma adicional a los descuentos a) y b)

RELACIÓN DE PROBLEMAS II. Estructuras de Control

- Aplicando c) de forma exclusiva con los anteriores, es decir, que si se aplica c), no se aplicaría ni a) ni b)

Finalidad: Plantear una estructura condicional anidada. Dificultad Media.

Ejercicios sobre bucles

19. Realice un programa que lea desde teclado un entero *tope* e imprima en pantalla todos sus divisores propios. 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.

20. Modifiquemos el ejercicio 5 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:

```
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.

21. 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.

22. Recupere la solución del ejercicio 10 (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 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,

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

23. Amplie el ejercicio 12 (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:

```
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.

24. 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 (el usuario introduce un valor en el rango pedido), se mostrará en pantalla el resultado de calcular `nuevo - min` y `max - nuevo`.

Debe resolver este problema separando los bloques de Entrada/Salida de los de Cómputos.

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.

25. Se pide leer un carácter desde teclado, obligando al usuario a que sea una letra mayúscula. Para ello, habrá que usar una estructura repetitiva `do 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.

26. 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).

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

27. Amplíe el ejercicio 6 de manera que se permita que los dos instantes puedan pertenecer a dos días distintos, pero eso sí, consecutivos.

Filtrar adecuadamente los datos leídos.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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

28. Se quiere construir un programa para leer los datos necesarios del ejercicio 11 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 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, 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.

29. 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:

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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

— 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.

30. En el ejercicio 13 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.

31. 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.

32. Amplíe el ejercicio 8 (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.

33. 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 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.

34. 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.

35. 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.

36. 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.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

37. Un número entero n se dice que es *desgarrable* (torn) 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. Dificultad Baja.

38. 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.

39. 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:

- Modifique la solución del ejercicio 19 de forma que el filtro de entrada usado para leer la variable `tope`, se haga con un bucle pre-test `while`.
- Modifique la solución del ejercicio 20 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.

40. 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 de la biblioteca `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:

$$0! = 1$$

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

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

41. 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`.

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

42. Escriba un programa que lea cuatro valores de tipo char (min_izda, max_dcha, 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.

43. 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
```

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

44. 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
```

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

45. Modifique los dos ejercicios anteriores para que se lea desde teclado el valor inicial y el número de filas a imprimir. En los ejemplos anteriores, el valor inicial era 1 y se imprimían un total de 6 filas.

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

46. 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.

47. Resuelva el ejercicio 46 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.

48. Recupere la solución del ejercicio 10 (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.

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

49. 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.

50. ([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.

51. 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.

52. Reescribid la solución a los ejercicios 19 (divisores) y 20 (interés) usando un bucle `for`

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

53. 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.

54. Sobre la solución del ejercicio 20 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.

55. 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.

56. 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.

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

RELACIÓN DE PROBLEMAS II. Estructuras de Control

Escribir un programa que diga si un número natural n es feliz para un grado k dado de antemano. Tanto n como k son valores introducidos por el usuario.

Finalidad: Ejercitar los bucles anidados. Dificultad Media.

58. 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.

59. 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 $x = -50, -48, \dots, 48, 50$ y $y = -40, -39, \dots, 39, 40$, es decir queremos mostrar en pantalla los valores de la función en los puntos

$$(-50, 40), (-50, -39), \dots, (-50, 40), (-48, 40), (-48, -39), \dots, (50, 40)$$

Dificultad Baja.

60. 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.

61. 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.

62. 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.

63. 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

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

64. 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.

65. 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 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.

66. 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 tope introducido desde teclado.

Dificultad Baja.

67. Escriba un programa que lea por teclado un número entero positivo tope y muestre por pantalla el factorial de los tope 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.

68. 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

RELACIÓN DE PROBLEMAS II. Estructuras de Control

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.

69. 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.

70. 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.

71. 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, ...

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.

72. 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 71).

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.

73. 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:

$$\sigma_1(10) - 10 = 5 + 2 + 1 = 8$$

$$\sigma_1(8) - 8 = 4 + 2 + 1 = 7$$

$$\sigma_1(7) - 7 = 1$$

$$\sigma_1(1) - 1 = 0$$

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.