Fundamentos de Programación.

Guión de Prácticas.

Curso 2015/2016

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. 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.

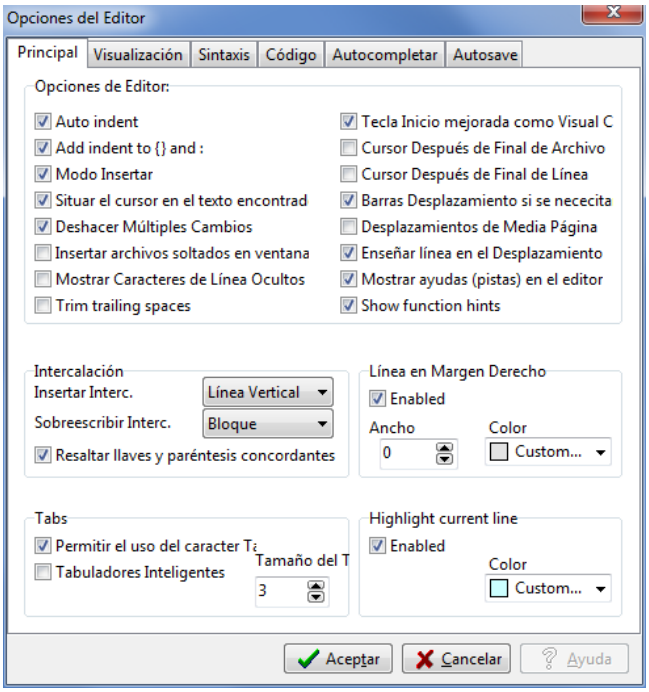
Instalación de Orwell Dev C++ en nuestra casa

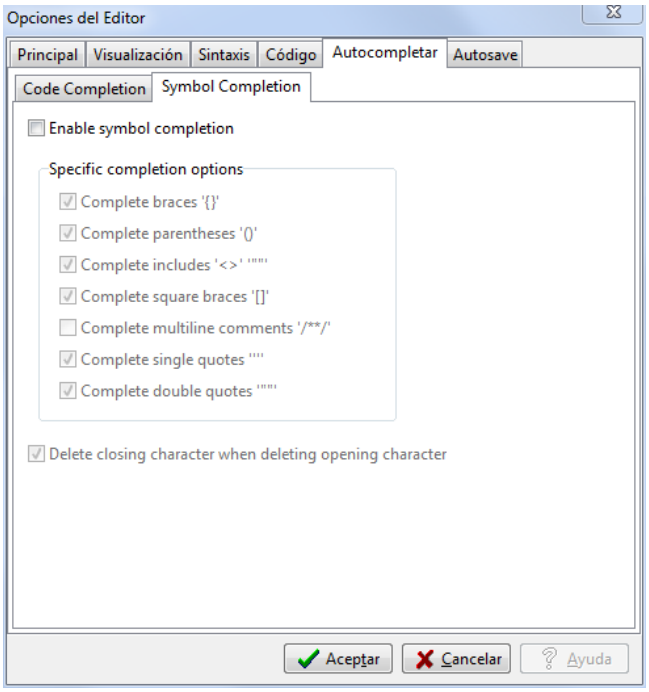
El entorno de desarrollo que usaremos será Orwell Dev C++. Puede descargarse desde la página:

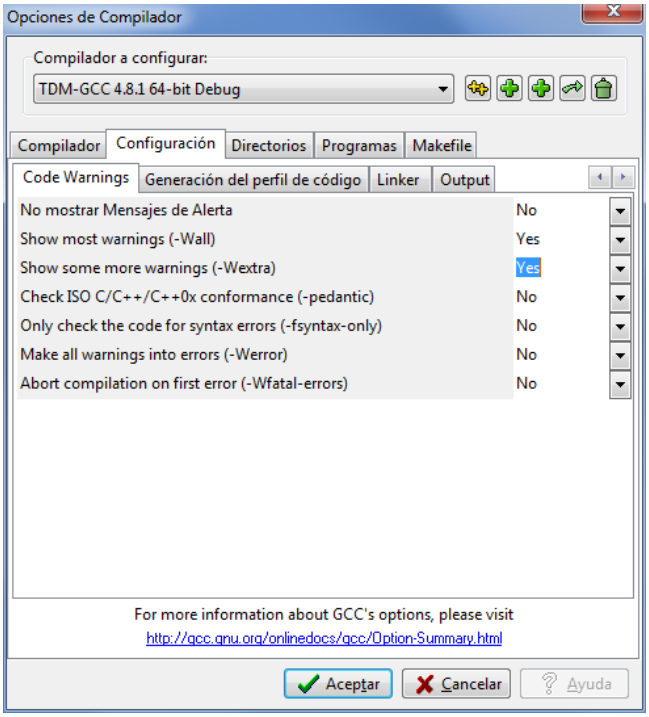
http://sourceforge.net/projects/orwelldevcpp/?source=typ_redirect

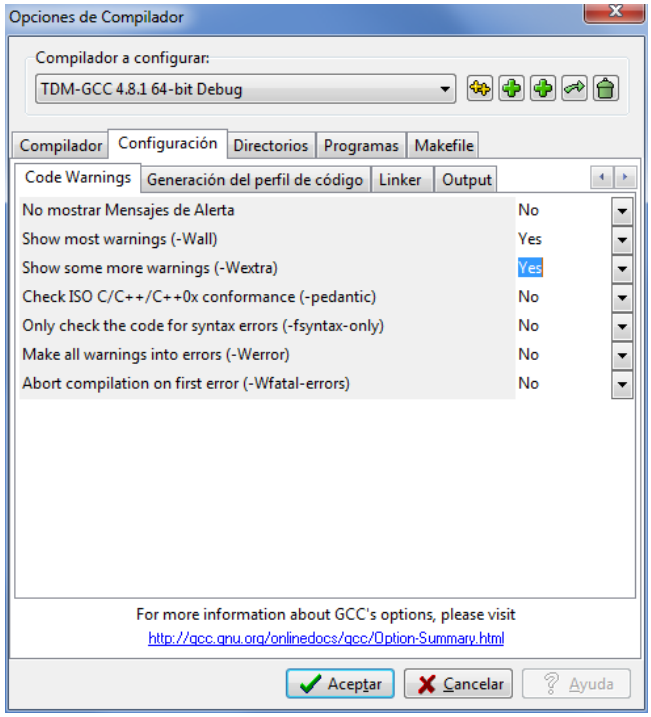
Cuando lo instalemos en nuestra casa, configurar las siguientes opciones:

```
Herramientas -> Opciones del Compilador
   Compilador a configurar: TDM-GCC ... Debug
   Configuración -> Code Warnings. Marcar los siguientes:
      Show most warnings
      Show some more warnings
   Configuración -> Linker.
      Generar información de Debug: Yes
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) 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 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.

   b) Si queremos hacerlo para una única consola, basta ejecutar el comando
   `chcp 1252`
   sobre la consola. El problema es que cada vez que se abre una nueva consola (por ejemplo, como resultado de ejecutar un programa desde Orwell Dev C++) hay que realizar este cambio. En nuestro caso, pondríamos (por ejemplo, al inicio del programa, justo después de las declaraciones de las variables) lo siguiente:

   ```
   system("chcp 1252");
   ```

   En cualquier caso, remarcamos que esta solución no es necesaria si se adopta la primera, es decir, el cambio del registro de Windows.

   c) Usar `setlocale` (buscad documentación en Internet)

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

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 `decsai`.

► *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 an 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++. Consultad la sección de Instalación (página 3) de este guión.

Actividad: Resolución de problemas.

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

6 (Interés bancario)
11 (Circunferencia)
12 (Gaussiana)

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

Realizad una lectura rápida de las actividades a realizar la semana próxima durante las horas de prácticas en las aulas de ordenadores (ver página siguiente)

Actividades de Ampliación

Leer 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` . Si durante la sesión se requiere abrir algún fichero, éste puede encontrarse en la plataforma web de la asignatura https://decsai.ugr.es (acceso identificado) o en la carpeta del Sistema Operativo instalado en las aulas

```
H:\CCIA\Grado_FP\
```

En el escritorio de Windows, se encuentra un acceso directo a dicha carpeta.

Muy Importante. Los ficheros que se encuentran en la unidad `H:` están protegidos y no pueden modificarse. Por tanto, habrá que copiarlos a la unidad local `U:`, dónde ya sí podrán ser modificados.

El primer programa

Copiando el código fuente

En el directorio `H:\ccia\Grado_FP\ProblemasI` se encuentra el directorio `I_Pitagoras`. Copiadlo entero a vuestra carpeta local (dentro de `U:\FP`).

Importante: Siempre hay que copiar localmente las carpetas que aparecen en la unidad `H:` del departamento ya que están protegidos contra escritura y no se puede trabajar directamente sobre ellos.

Desde el Explorador de Windows, entrad en la carpeta recién creada en vuestra cuenta:

```
U:\FP\I_Pitagoras
```

y haced doble click sobre el fichero `I_Pitagoras.cpp`. Debe aparecer una ventana como la de la figura 1

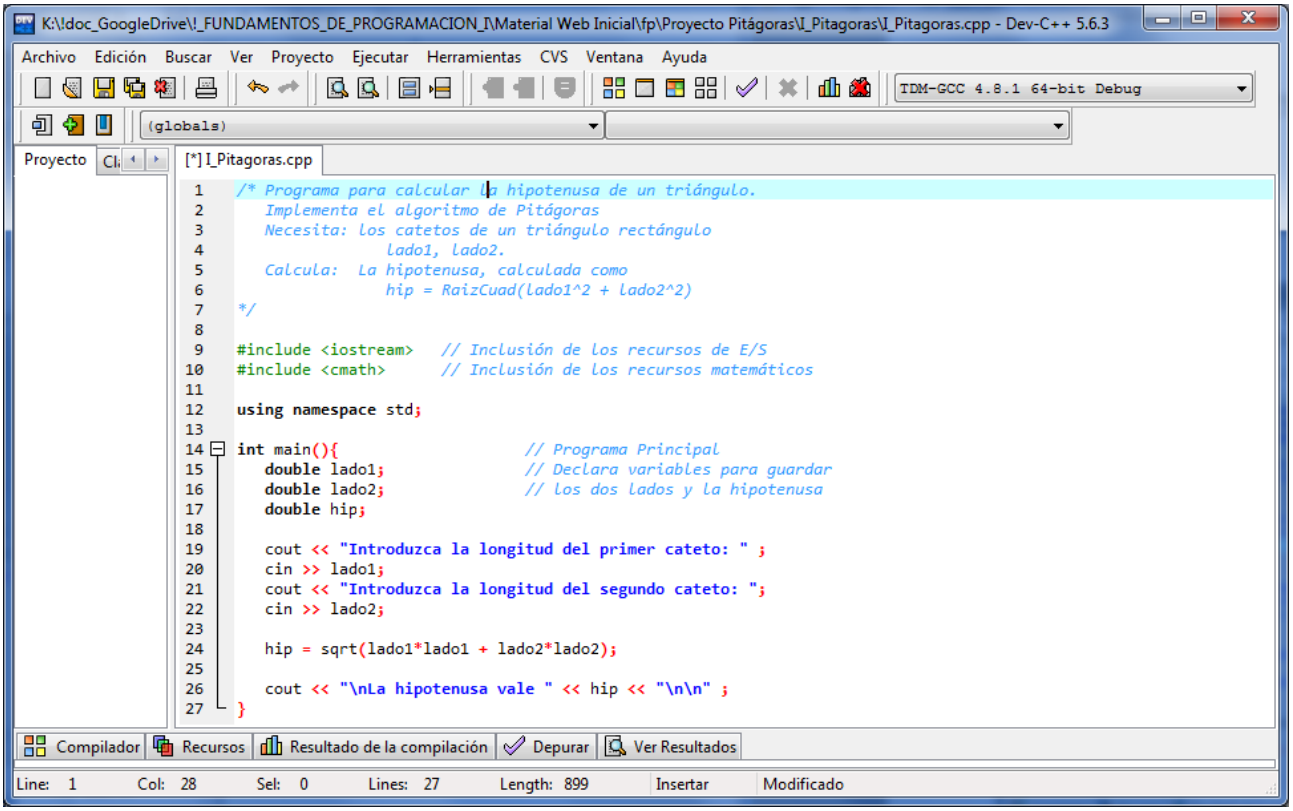


Figura 1: Programa que implementa el algoritmo de Pitágoras

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.

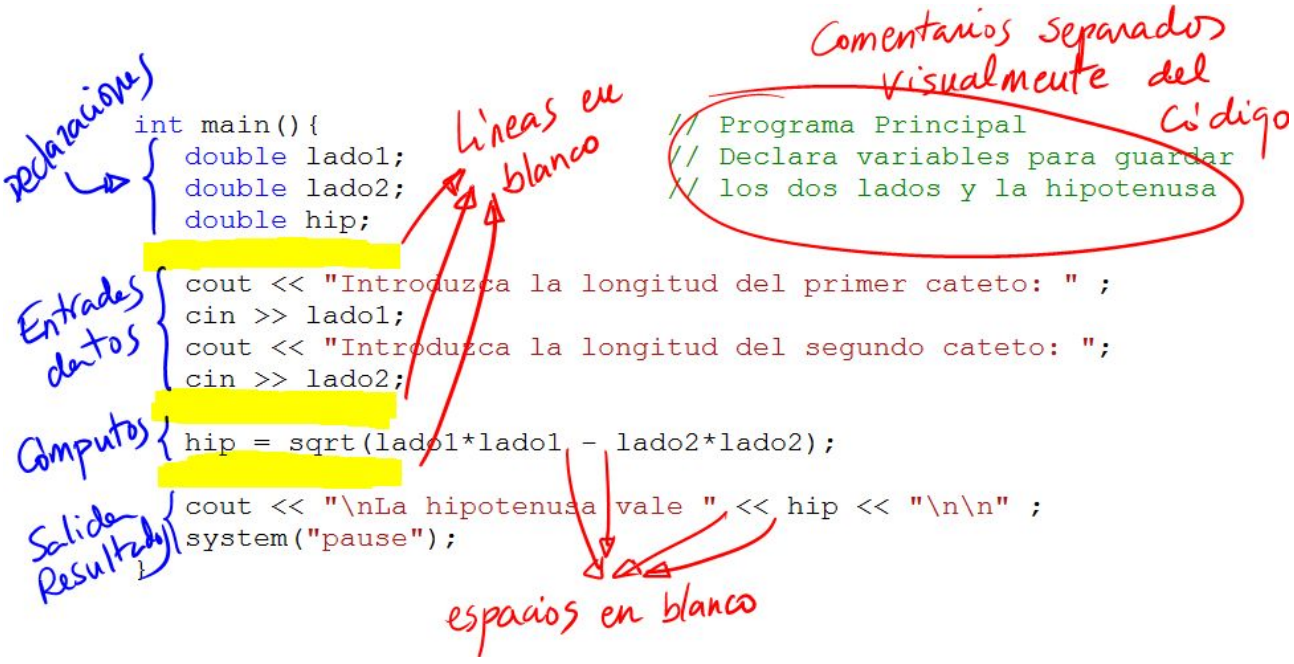


Figura 2: Escritura de código

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

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 la 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.

Introducid 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. Borrad 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. Borrad alguna de las dobles comillas en una constante de cadena de caracteres, tanto las comillas iniciales como las finales.
9. Borrad alguna de las llaves que delimitan el inicio y final del programa.
10. Borrad 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 (Ctr-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`

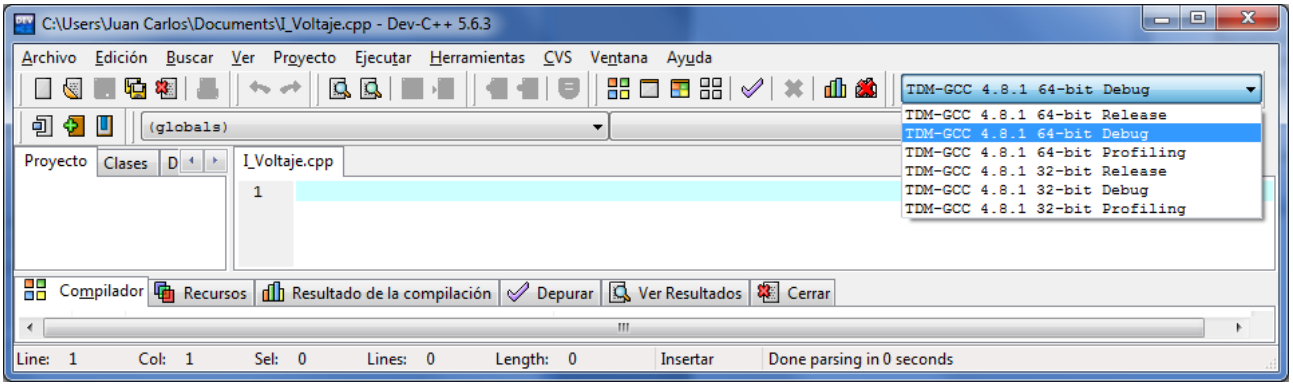


Figura 3: Creación de un programa nuevo

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.

Implementad los ejercicios que había que resolver en papel para esta sesión. Guardad los programas en el directorio en red. No hace falta entregar la solución.

Fundamentos de Programación.

Relaciones de Problemas.

RELACIÓN DE PROBLEMAS I. Introducción a C++

1. Indique cuáles serán los valores de las variables a y x después de ejecutar el código siguiente

```
a = 0;
i = 1;
x = 0;
a = a + i;
x = x + i / a;
a = a + i;
x = x + i / a;
a = a + i;
x = x + i / a;
a = a + i;
x = x + i / a;
```

   Obsérvese que normalmente no usaremos nombres de variables tan cortos como los anteriores. Este ejemplo es una excepción, al tratarse de un ejercicio básico.

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

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

   voltaje = intensidad * resistencia

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

3. Escriba un programa que lea por pantalla la cantidad en millas (como un real) y muestre la cantidad equivalente en kilómetros. Debe tener en cuenta que 1 milla equivale a 1'609 kilómetros.

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

4. Realizar un programa que nos pida una longitud cualquiera dada en yardas. El programa deberá calcular el equivalente de dicha longitud en pulgadas, pies, millas y millas marinas, y mostrarnos los resultados en pantalla. Para el cálculo, utilice la siguiente tabla de conversión del sistema métrico:

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

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

5. De `http://countrymeters.info` se obtienen los siguientes datos estimados sobre la población de China:

   - nace una persona cada 1.87 segundos
   - muere una persona cada 3.27 segundos
   - emigra una personada cada 71.9 segundos

   Escriba un programa que muestre la población dentro de 2 años, considerando que la población actual es de 1.375.570.814 personas.

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

6. Un banco presenta la siguiente oferta. Si se deposita una cantidad de euros `capital` durante un año a plazo fijo, se dará un interés dado por la variable `interes`. Realizad un programa que lea una cantidad `capital` y un interés `interes` desde teclado y calcule en una variable `total` el dinero que se tendrá al cabo de un año, aplicando la fórmula:

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

   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`.

   A continuación, el programa debe imprimir en pantalla el valor de la variable `total`. Tanto el capital como el interés serán valores reales. 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.

   Supongamos que queremos modificar la variable original `capital` con el nuevo valor de `total`. ¿Es posible hacerlo directamente en la expresión de arriba?

   Nota: El operador de división en C++ es /

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

7. 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 cuantos 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.

8. Las ganancias de un determinado producto se reparten entre el diseñador y los tres fabricantes del mismo. Diseñar 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. Utilizad el tipo `double` para todas las variables.
   Importante: No repetid cálculos ya realizados.

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

9. Queremos realizar un programa para intercambiar los contenidos de dos variables enteras. El programa leerá desde teclado dos variables `edad_Pedro` y `edad_Juan` e intercambiará sus valores. A continuación, mostrará en pantalla las variables ya modificadas. El siguiente código no funciona correctamente.

   ```
   edad_Pedro = edad_Juan;
   edad_Juan = edad_Pedro;
   ```

   ¿Por qué no funciona? Buscad una solución.

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

10. Escribid un algoritmo para calcular la media aritmética muestral y la desviación estándar (o típica) muestral de las alturas de tres personas (`n`=3). Éstos valores serán reales (de tipo `double`). La fórmula general para un valor arbitrario de `n` es:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \ , \ \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})^2}$$

   $\overline{X}$ representa la media aritmética y σ la desviación estándar. 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.disfrutalasmatematicas.com/datos/desviacion-estandar-calculadora.html

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

11. Cread un programa que nos pida la longitud del radio, calcule el área del círculo y la longitud de la circunferencia correspondientes, y nos muestre los resultados en pantalla. Recordad que:

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

Usad el literal `3.1416` a lo largo del código, cuando se necesite multiplicar por π.

Una vez hecho el programa, cambiad las apariciones de `3.1416` por `3.14159`, recompilad y ejecutad (La parte de compilación y ejecución se realizará cuando se vea en clase de prácticas el entorno de programación).

¿No hubiese sido mejor declarar un dato *constante* `PI` con un valor igual a `3.14159`, y usar dicho dato donde fuese necesario? Hacedlo tal y como se explica en las transparencias de los apuntes de clase.

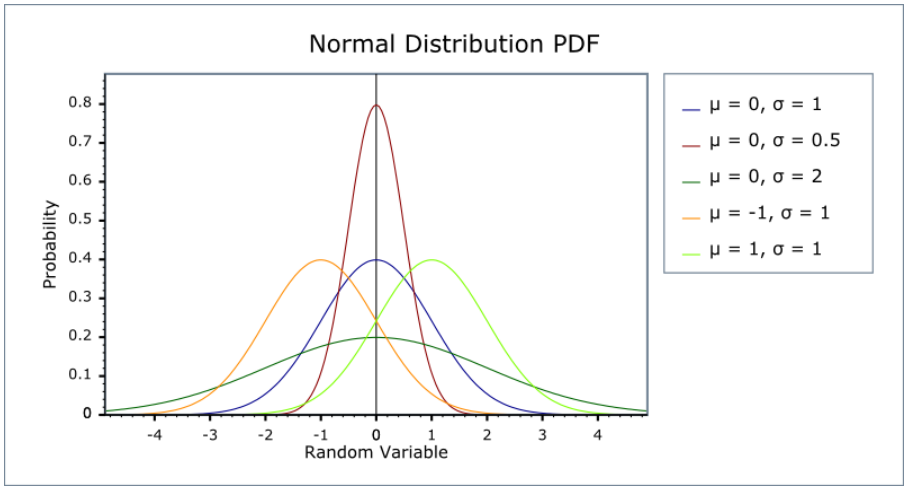
Cambiad ahora el valor de la constante `PI` por el de `3.1415927`, recompilad y ejecutad.

Finalidad: Entender la importancia de las constantes. Dificultad Baja.

12. Realizar un programa que lea los coeficientes reales μ y σ de una función gaussiana (ver definición abajo). A continuación el programa leerá un valor de abscisa x y se imprimirá el valor que toma la función en x

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

La función gaussiana es muy importante en Estadística. Es una función real de variable real en la que el parámetro μ se conoce como *esperanza* o *media* y σ como *desviación típica* (*mean* y *standard deviation* en inglés). En la gráfica de abajo pueden verse algunos ejemplos de esta función con distintos parámetros.



Para definir la función matemática e usad la función `exp` de la biblioteca `cmath`. En la misma biblioteca está la función `sqrt` para calcular la raíz cuadrada. Para elevar un número al cuadrado se puede usar la función `pow`, que se utiliza en la siguiente forma:

```
pow(base, exponente)
```

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

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

13. 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*).

    Escribid 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.

14. Escribir un programa que lea un valor entero. Supondremos que el usuario introduce siempre un entero de tres dígitos, como por ejemplo `351`. Escribid en pantalla los dígitos separados por tres espacios en blanco. Con el valor anterior la salida sería:

    ```
    3   5   1
    ```

Dificultad Baja.

15. Leed desde teclado tres variables correspondientes a un número de horas, minutos y segundos, respectivamente. Diseñar un algoritmo que calcule las horas, minutos y segundos dentro de su rango correspondiente. Por ejemplo, dadas 10 horas, 119 minutos y 280 segundos, debería dar como resultado 12 horas, 3 minutos y 40 segundos. El programa no calculará meses, años, etc sino que se quedará en los días.

    Como consejo, utilizad el operador `/` que cuando trabaja sobre datos enteros, representa la división entera. Para calcular el resto de la división entera, usad el operador `%`.

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

16. Calcular 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.

    Finalidad: Trabajar con expresiones numéricas y algoritmos. *Dificultad Media.*

17. Realizar un programa que declare las variables `x`, `y` y `z`, les asigne los valores 10, 20 y 30 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).

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

18. Realizad 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. Realizad 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. Diseñar un programa que lea un carácter (supondremos que el usuario introduce una mayúscula), lo pase a minúscula y lo imprima en pantalla. Hacedlo sin usar las funciones `toupper` ni `tolower` de la biblioteca `cctype`. Para ello, debe considerarse la equivalencia en C++ entre los tipos enteros y caracteres.

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

21. Supongamos el siguiente código:

```
int entero;
char caracter;

caracter = '7';
entero = caracter;
```

    La variable entero almacenará el valor 55 (el orden en la tabla ASCII del carácter `'7'`). Queremos construir una expresión que devuelva el entero 7, para asignarlo a la variable `entero`. Formalmente:

    Supongamos una variable `car` de tipo carácter que contiene un valor entre `'0'` y `'9'`. Construid un programa que obtenga el correspondiente valor entero, se lo asigne a una variable de tipo `int` llamada `entero` y lo imprima en pantalla. Por ejemplo, si la variable `car` contiene `'7'` queremos asignarle a `entero` el valor numérico 7.

    Nota. La comilla simple para representar un literal de carácter es la que hay en el teclado del ordenador debajo de la interrogación `?`.

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

22. 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 )
```

23. Razonar 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.

24. Indicar si se produce un problema de precisión o de desbordamiento en los siguientes ejemplos indicando 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. Hacedlo escribiendo la sentencia `cout.precision(numero_digitos);` en cualquier sitio del programa antes de la ejecución de `cout << real1 << "," << real2;`. Hay que destacar que al trabajar con reales siempre debemos asumir representaciones aproximadas por lo que no podemos pensar que el anterior valor `numero_digitos` esté indicando un número de decimales con representación exacta.

    a)
    ```
    int chico, chico1, chico2;
    chico1 = 123456789;
    chico2 = 123456780;
    chico = chico1 * chico2;
    ```

    b)
    ```
    long grande;
    int chico1, chico2;
    chico1 = 123456789;
    chico2 = 123456780;
    grande = chico1 * chico2;
    ```

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

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

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

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

g)
```
float chico;
double grande;

grande = 2e+150;
chico = grande;
```

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

25. Escribid una expresión lógica que sea verdadera si una variable de tipo carácter llamada `letra` es una letra minúscula y falso en otro caso.

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

    Escribid una expresión lógica que nos informe cuando 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.

    Escribid un programa que lea las variables `letra`, `edad` y `anio`, calcule el valor de las expresiones lógicas anteriores e imprima el resultado. Tened 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.

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

26. 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)
    - La cualidad de que un número entero sea primo o no.
    - Estado civil (casado, soltero, separado, viudo)
    - Sexo de una persona (hombre o mujer exclusivamente)

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

27. 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. Diseñar 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.

28. Cread 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:

```
Grados Fahrenheit = (Grados Celsius * 180 / 100) + 32
```

Buscad en Internet el por qué de dicha fórmula.

Dificultad Baja.

29. Cread 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`.

30. Declarar las variables necesarias y traducir las siguientes fórmulas a expresiones válidas del lenguaje C++.

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

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

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

Algunas funciones de `cmath`
$\mathrm{sen}(x) \longrightarrow$ `sin(x)`
$\cos(x) \longrightarrow$ `cos(x)`
$x^y \longrightarrow$ `pow(x, y)`
$\ln(x) \longrightarrow$ `log(x)`
$\mathrm{e}^x \longrightarrow$ `exp(x)`

Dificultad Baja.

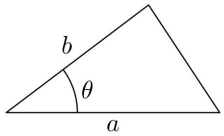
31. Dos locomotoras parten de puntos distintos avanzando en dirección contraria sobre la misma vía. Se pide redactar un programa para conocer las distancias que habrán recorrido ambas locomotoras antes de que choquen teniendo en cuenta que la primera locomotora viaja a una velocidad constante V_1, que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

32. 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 $A = \frac{1}{2}ab\,sin(\theta)$. Construid 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.

33. 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 cómputos, 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.