Fundamentos de Programación.

Guión de Prácticas.

Curso 2012/2013

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

Profesores de prácticas: M.Gómez y S. Acid
Para cualquier cuestión metodológica ponerse en contacto con los profesores o bien enviando un e-mail a mgomez@decsai.ugr.es y acid@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

Este guión de prácticas contiene las actividades a realizar por el alumno, tanto de forma presencial (en las aulas de la Escuela de Informática) como no presencial a realizar en casa. Todas las actividades son obligatorias y están incluidas en la evaluación de la participacion de clase, excepto las marcadas como *Actividades de Ampliación*.

El guión está dividido en sesiones, que corresponden a cada semana lectiva. Como ya se indicara en clase, para la asignatura de FP a cada semana le corresponden 6 horas de trabajo personal del alumno, aparte de las 4 horas de clases presenciales. Por ello, las actividades no presenciales de una sesión deben desarrollarse **antes**, *durante toda una semana*, y culminan con una entrega de los ejercicios en la propia sesión de prácticas. Por lo que el alumno deberá descargarse al terminar una sesión, las actividades encargadas para la sesión siguiente.

Entre las actividades del guión, una actividad no presencial recurrente es, la resolución de problemas propuestos en las *Relaciones de Problemas*. Éstas constan de dos tipos de problemas:

- *Básicos*:

  Deben resolverse en la casa y el alumno deberá defenderlos individualmente. La defensa será individual y forma parte de la nota final de la asignatura (10 % tal y como se explica en el apartado de evaluación de la asignatura).

  Las soluciones a los problemas deberán ser subidas a la plataforma de decsai através del GAP. Para ello, el alumno debe entrar en el acceso identificado de decsai, entrar por el GAP, y seleccionar la entrega de prácticas correspondiente a la semana en curso. El alumno subirá cada uno de los ficheros con extensión `cpp` correspondientes a las soluciones de los ejercicios porpuestos (tanto obligatorios, como opcionales) fijados en esa sesión.
  Nota: Para una correcta gestión de los ficheros en el GAP, los nombres de los ficheros deberán ser de la forma: `solucion1.cpp`, `solucion2.cpp`, `solucion3.cpp`, etc. correspondientes al número de ejercicio con el que figuran en el guión de prácticas.

  El profesor entregará al final de cada semana las soluciones a los ejercicios básicos. Es muy importante que el alumno revise estas soluciones y las compare con las que él había diseñado.

  Del conjunto de problemas básicos, en el guión de prácticas se distinguirá entre:

  1. *Obligatorios*:

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

2. *Opcionales*:
   Si se realizan correctamente estos ejercicios, el alumno podrá sacar hasta un 10 (sobre 10) de nota.

- *Ampliación*: problemas cuya solución no se verá, pero que sirven para afianzar conocimientos. El alumno debería intentar resolver por su cuenta un alto porcentaje de éstos.

Las actividades marcadas como *Seminario* han de hacerse obligatoriamente y siempre serán expuestas por algún alumno elegido aleatoriamente.

Para la realización de estas prácticas, se utilizará el entorno de programación `Code::Blocks`. Se recomienda la instalación del programa lo antes posible (en la primera semana de prácticas). En cualquier caso, el alumno puede instalar en su casa cualquier otro compilador, como por ejemplo `Visual Studio` → `www.microsoft.com/visualstudio/`

Muy importante:

- La resolución de los problemas y actividades puede hacerse en grupo, pero la defensa durante las sesiones presenciales es individual.
- LLevar la asignatura al día para poder realizar los ejercicios propuestos para cada sesión.

RELACIÓN DE PROBLEMAS I. Introducción a C++

Problemas Básicos

1. 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.*

2. 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.*

3. 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.*

4. 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.

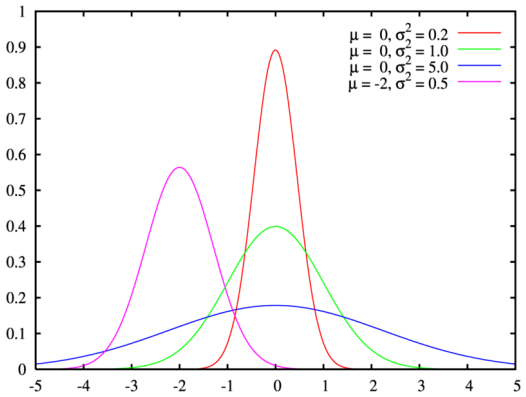
5. Realizar un programa que lea los coeficientes reales μ y σ de una función gaussiana (ver definición abajo). A continuación el programa leerá un valor de abscisa x y se imprimirá el valor que toma la función en x

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

   El parámetro μ se conoce como *esperanza* o *media* y σ como *desviación típica* (*mean* y *standard deviation* en inglés). 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 el applet disponible en
   http://www.danielsoper.com/statcalc/calc54.aspx
   o bien algunos de los ejemplos de la figura siguiente (observad que el valor de la desviación está elevado al cuadrado):



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

6. 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 cuanto 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.

7. Redactar un algoritmo para calcular la media aritmética muestral y la desviación estándar (o típica) muestral de las alturas de tres personas. Éstos valores serán reales (de tipo `double`)

$$\overline{X} = \frac{1}{3}\sum_{i=1}^{3} x_i \ , \ \sigma = \sqrt{\frac{1}{3}\sum_{i=1}^{3}(x_i - \overline{X})^2}$$

   $\overline{X}$ 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`.

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

8. 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.

9. 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. En el caso de que nos salgan más de 24 horas, daremos también los días correspondientes (pero ya no pasamos a ver los meses, años, etc)

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.

10. 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.

11. 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.

12. Indicar si se produce un problema de precisión o de desbordamiento en los siguientes ejemplos indicando cuál sería el resultado final de las operaciones.

    a)
    ```
    int resultado, entero1, entero2;
    entero1 = 123456789;
    entero2 = 123456780;
    resultado = entero1 * entero2;
    ```

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

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

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

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.

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

13. 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.*

14. 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.*

15. 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.

16. 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.

    Nota: 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.

17. 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.

Problemas de Ampliación

18. 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.

19. 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)`
$e^x \longrightarrow$ `exp(x)`

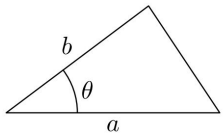
Dificultad Baja.

20. 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

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

Dificultad Baja.

21. 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.

Dificultad Baja.

22. 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 Media.

23. 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.

Sesión 2

► *Actividades a realizar en casa*

Actividad: Instalación de CodeBlocks.

Durante las primera semanas de clase, el alumno deberá instalar en su casa un compilador de C++. En decsai, Prácticas, está disponible una manual de instalación de CodeBlocks, IDE, utilizado en las sesiones presenciales de clase, documento **ManualInstalacionCODEBLOCKS.pdf**.

Actividad: Resolución de problemas.

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

Resolved los ejercicios siguientes de la relación de problemas I, página 4.

- *Obligatorios:*
  1 (Voltaje) `solucion1.cpp`
  2 (Interés bancario) `solucion2.cpp`
  4 (Circunferencia) `solucion4.cpp`
  6 (Fabricante) `solucion6.cpp`
  9 (Horas, minutos, segundos) `solucion9.cpp`
- *Opcionales:*
  11 (Media aritmética) `solucion11.cpp`
  5 (Gaussiana) `solucion5.cpp`

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*

Durante esta sesión de prácticas, el profesor irá corrigiendo individualmente (a algunos alumnos elegidos aleatoriamente) los ejercicios indicados anteriormente. Mientras tanto, el resto de alumnos deben seguir las instrucciones indicadas por el profesor.

Entrega de trabajos

Al finalizar la sesión de prácticas en el aula, se cierra la entrega de prácticas, por ello, con antelación es necesario realizar la entrega de las soluciones a los ejercicios de esa sesión. Esta se hace a través de decsai, mediante el acceso por el GAP, seleccionando la entrega activa de la semana en grupo que le corresponde a cada alumno.

Como ya se explicara, a través del acceso identificado de decsai, entrar por el GAP, y seleccionar la entrega de prácticas correspondiente a la semana en curso. El alumno subirá cada uno de los ficheros con extensión `cpp` correspondientes a las soluciones de los ejercicios porpuestos (tanto obligatorios, como opcionales) fijados en esa sesión. Para una correcta gestión de los ficheros en el GAP, los nombres de los archivos para esta sesión deben ser (en el caso en que se entregasen todos):

`solucion1.cpp`, `solucion2.cpp`, `solucion4.cpp`, `solucion6.cpp`, `solucion9.cpp`, `solucion11.cpp`, `solucion5.cpp`

correspondientes al número de ejercicios que figuran en el guión de prácticas.

Algunas consideraciones con respecto a la escritura de código en C++ (ver figura 1)

- 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.

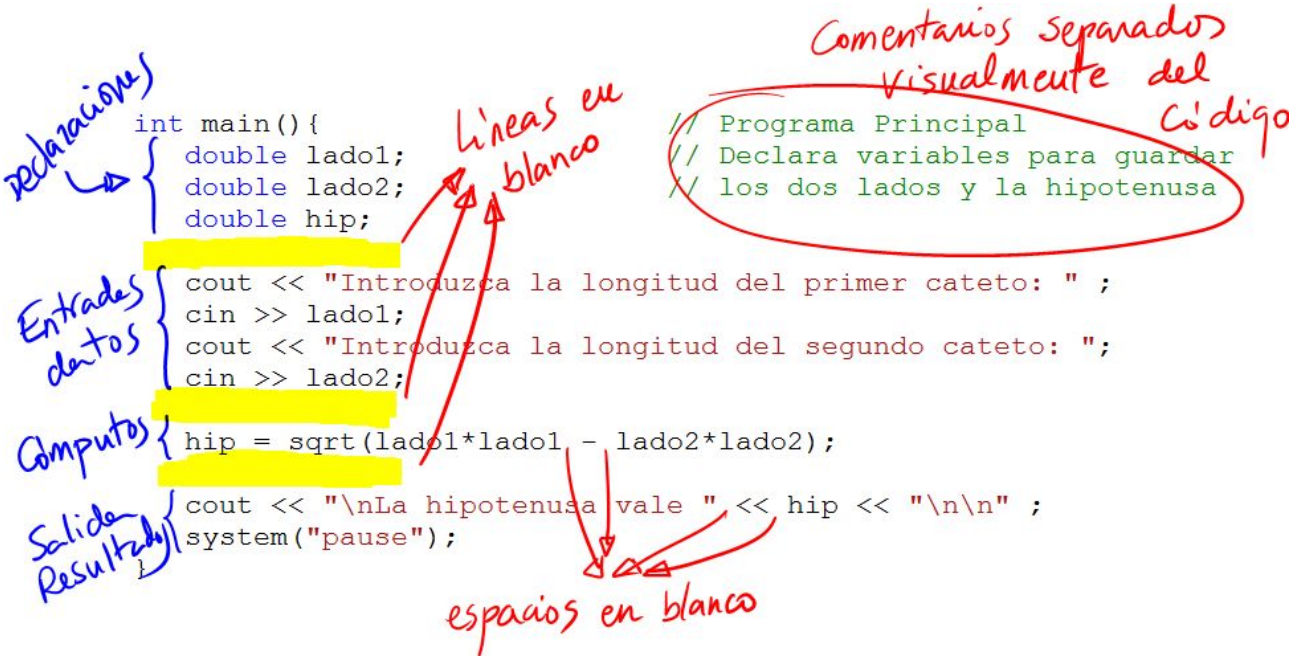


Figura 1: Escritura de código

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