Fundamentos de Programación
Relación de ejercicios 1

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 $interes$. Realizar 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:

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

   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. Crear 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:

   $$longitud = 2\pi * r \quad area = \pi * r^2.$$

   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.

4. 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

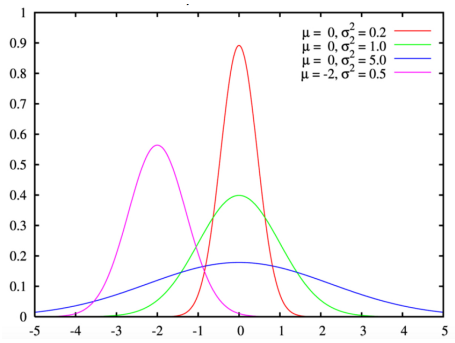
   $$Gaussiana(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

   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, de forma aproximada, con 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.

5. 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.

6. 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.

7. 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:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

   donde $\overline{x}$ representa la media aritmética y σ_x 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. 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.

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. 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.

10. Realizar un programa en C++ que simule el procedimiento a seguir para dar la vuelta en una máquina de refrescos. El programa deberá tener como entrada -por teclado- el precio del producto y la cantidad ingresada por el cliente (en céntimos). Como salida, el programa deberá mostrar por pantalla el número de monedas de 1 euro, 50, 20, 10 y 5 ctmos que la máquina deberá devolver como cambio. NOTA: Suponer que la cantidad introducida por el cliente es igual o superior al precio del producto.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. Indique qué valor devuelven las siguientes expresiones y de qué tipo de dato es dicho valor. Indique, en su caso, si la expresión está mal formulada -o es errónea- y el motivo:

    a. $2 + 3 < 5 + 8$

    b. $2 < 3 < 4$

    c. $7/3 + 4.0$

    d. $x == 4 \&\& y == 3 || z! = 2 == 1! = 2$, si `x=4`, `y=4`, `z=4` son variables de tipo entero.

    e. $x <= 4 || x >= 4$, si `x` es variable de tipo entero.

    f. $x < 4 \&\& x > 4$, si `x` es variable de tipo entero.

    g. $z > 6 || a <' z' \&\& a >' a' || z < 6$, si `z` es de tipo entero y `a` es de tipo caracter.

19. En los siguientes programas, indicar qué errores existen y corregirlos. Arregle también los errores de indentación.

```
ina main{} (

   cout >> 'hola'
)
```

```
includeo <istream>
using namespace sdt

main[ ] (
        const int X==9;
        Double Y==3.2;

        cin >> "Dime un valor para Y">>X;
        cout<<Y;
)
```

```
Int Main()  {

        Boolean a= 0; b= false;

   a= true; b= true;
   return 0
]
```

```
#include <oistream>
using namespace std

int main() {
```

```
5
6           char x;
7
8           cout << X vale<< x;
9       cint << "Dime un valir para X";
10          cont << "Ahora X vale x"
11  }
```

```
1   #include <oistream>
2   using namespace std
3
4   main() {
5
6           char x,
7           int a;
8
9           cin << "Dime un numero entre 32 y 255";
10      cout >> a;
11
12      x==a;
13      cout < "El codigo ASCII para el valor introducido a es x";
14
15  }
```

20. 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.

21. Indique qué tipo de dato usaría para representar:

    a) Edad de una persona

    b) Producto interior bruto de un país.

    c) La cualidad de que un número entero sea primo o no.

    d) Estado civil (casado, soltero, separado, viudo)

    e) 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.

22. 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 IVA. 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 IVA aplicable es del 16 %.

    Dificultad Baja.

23. Realizar un programa que pida una temperatura en grados Celsius y la convierta a grados Fahrenheit. El programa pedirá por teclado la temperatura en Celsius y mostrará el mensaje *X grados Celsius son Y grados Fahrenheit*. Recuerde que la fórmula de conversión entre grados Celsius y Fahrenheit es:

$$\frac{F-32}{9} = \frac{C}{5}$$

Dificultad Baja.

24. 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`.

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

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

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

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

Nota: Algunas funciones de `cmath`: $\sin(x)$ →`sin(x)`, $\cos(x)$ →`cos(x)`, x^y →`pow(x,y)`, $ln(x)$ →`log(x)`, e^x →`exp(x)`.

26. 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

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

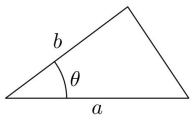
donde D es la distancia que separa los puntos iniciales de partida. Los datos de entrada al programa serán D, V_1 y V_2.

Dificultad Baja.

27. 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.

28. 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:

$$6 = 1 * 22 + 1 * 21 + 0 * 20 \equiv 110$$

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.

29. Haga un programa que pida por teclado el nombre y los dos apellidos, y que los guarde en variables separadas. Posteriormente, todos ellos deberán copiarse en el orden *apellido1 apellido2, nombre* en otra nueva variable. Mostrar el contenido de esta variable por pantalla, junto con la longitud total de la cadena resultante.

30. Defina una estructura para representar un punto en 3 dimensiones (x,y,z). Elabore un programa que declare 4 variables de este nuevo tipo de dato y asígnele valores a cada una de ellas. Por último, muéstrelas por pantalla.

31. Se define `TipoCiudad` como una estructura para almacenar la posición de una ciudad en el plano. Se define en la forma siguiente:

```
struct TipoPunto{
   double abscisa;
   double ordenada;
}

struct TipoCiudad{
   TipoPunto situacion;
   string nombre;
};
```

donde `nombre` es el nombre de la ciudad, y `situacion` son las coordenadas de la ciudad en el plano.

Hacer un programa que pida dos ciudades y que cree una nueva ciudad cuyo nombre sea la concatenación de ambas y cuya situación sea la multiplicación de cada una de sus coordenadas. Mostrar las características de la nueva ciudad por pantalla.

32. Dada la declaración de la siguiente estructura y de la variable:

```
struct CuentaCD {
  double saldo;
  double tasa_de_interes;
  int plazo;
  char inicial1;
  char inicial2;
};

CuentaCD cuenta;
```

¿Qué tipo de dato tiene cada una de las siguientes variables? Si hay alguna errónea, indicar dónde está el error.

a. `cuenta.saldo`
b. `cuenta.tasa_de_interes`
c. `CuentaCD.plazo`
d. `cuenta_ahorros.inicial1`
e. `cuenta.inicial2`
f. `cuenta`

33. Crear una estructura apropiada para representar un número complejo. Un número complejo es aquel que matemáticamente se representa como `<parte_real>+<parte_imaginaria>*i`, donde i es la constante imaginaria. Construir, además, un programa para:

    a. Sumar dos complejos.
    b. Multiplicar dos complejos.
    c. Hallar el conjugado de un complejo.
    d. Imprimir un número complejo en formato `(r,i)`, donde *r* es la parte real e **i** la parte imaginaria.

34. Implementar una estructura que represente a un rectángulo. Partiendo de esta estructura, realizar un programa para:

    a. Calcular el área del rectángulo.
    b. Calcular el perímetro del rectángulo.
    c. Comprobar si dos rectángulos son idénticos. Dos rectángulos son idénticos si tienen el mismo área y perímetro.

35. Implementar una estructura `fecha` que represente a una fecha de tipo dia/mes/año (ejemplo: 12/3/2008). Hacer un programa para:

    a. Dado un dia, mes y año, asignar estos valores a una variable de tipo fecha.
    b. Actualizar una fecha de entrada a la fecha del dia siguiente.
    c. Actualizar una fecha de entrada a la fecha del dia anterior.
    d. Copiar una variable de tipo fecha en otra.
    e. Decir si el año de la fecha es bisiesto o no.

36. Implementar una estructura `racional` que represente a un número racional. Recordemos que estos números son las fracciones de tipo `x/y`, donde `x` e `y` son números enteros. Implementar un programa para:

    a. Inicializar un número racional a un valor dado por teclado.
    b. Simplificar un número racional a su fracción irreducible.
    c. Sumar dos números racionales.
    d. Multiplicar dos números racionales.
    e. "Evaluar" un número racional. Como resultado, este programa devolverá el valor real resultante de dividir el numerador entre el denominador.

37. Implementar una estructura que permita representar polinomios de grado máximo 3. Recordemos que un polinomio de grado máximo 3 se representa como $a_3x^3 + a_2x^2 + a_1x + a_0$, donde a_i son los coeficientes de grado i del polinomio. Realizar un programa para:

    a. Inicializar un polinomio a partir de los coeficientes dados por valores desde teclado.

    b. Sumar dos polinomios.

    c. Evaluar un polinomio en un punto x.