Introducción a la programación I

Ejercicios Tema 2

Los siguientes ejercicios están englobados dentro de la evaluación continua del alumno. Si bien no son de obligada realización para aprobar, se aconseja su realización para afianzar los conocimientos mostrados en el tema 2 de la asignatura.

Se aconseja realizar primero el pseudocódigo de los programas más complejos, para luego pasar a la implementación de los mismos. Además, se recuerda que los ejercicios deberían ser realizados de forma individual por los alumnos.

Ejercicio 1:

Se pide implementar un código que realice la siguiente función:

- Usar dos variables globales que almacenen los datos 450000 y 3500 respectivamente.
- Crear una función llamada “suma”, que sume esas dos variables y la almacene en una tercera variable global.
- Dentro de la función main, invocar a la función “suma”, y mostrar el resultado, usando la función printf vista en clase.

Ejercicio 2:

Se pide implementar un código que realice la siguiente función:

- Usar macros para definir dos valores constantes llamados CONST1 y CONST2, que almacenen los valores ‘A’ y ‘B’ respectivamente (son caracteres en mayúscula).
- Crear dos variables globales que puedan almacenar los valores constantes anteriormente indicados. Inicializar esas dos variables con los valores CONST1 y CONST2 respectivamente.
- Crear una función que transforme esas dos variables de mayúsculas a minúsculas.
- Crear una función main que use esa función, y muestre por consola el valor de las dos variables globales indicadas anteriormente.

Ejercicio 3:

Se pide implementar un código que realice la siguiente función:

- Crear una función que muestre por consola, en dos líneas distintas, el resultado de calcular el perímetro y área de un círculo de radio 5.2 cm.

Ejercicio 4:

Se pide implementar un programa que calcule el Movimiento Rectilíneo Uniformemente Acelerado (MRUA) de un móvil. Para t0=0 (tiempo inicial), el móvil se encuentra en la posición inicial "0", y su velocidad inicial es de 2m/s. Se pide averiguar su posición pasados 5s, teniendo en cuenta que se ve sometido a una aceleración de 6,8m/s^2 . Se piden los resultados en unidades internacionales.

Una vez calculado, el programa mostrará el resultado a través de terminal. Realizar la implementación usando las variables, constantes y tipos de datos que el alumno crea conveniente.

$$x = x_0 + v_0 \cdot t + \frac{a \cdot t^2}{2} \Rightarrow$$

Ejercicio 5:

Dado el siguiente código, se esperaba que la salida del mismo fuera 1222.5 . Sin embargo, el resultado dista de la solución ideal. Se pide al alumno buscar los errores, y corregirlos. Se debe mantener el número de variables (ni más, ni menos), las asignaciones, y el código del main seguirá usando la variable "c" para mostrar el valor por terminal.

```
#include <stdio.h>

int a=1222.5;
char c = 0;

void main()
{
	c=a;
	printf("El valor correcto es: %d\n",c);

}
```

Anexo: Uso de printf

La función printf se utiliza para mostrar por consola valores de variables con tipos de datos sencillos. Dado que es una función que puede recibir un número de parámetros variable, se usa una cadena de caracteres para poder describir los parámetros que se imprimirán. Un ejemplo de uso sería el siguiente:

- Si se quiere imprimir el valor de una variable entera llamada "var1", se podrá realizar de la siguiente manera:
  - printf("la variable entera vale: %d\n", var1);

En el caso anterior, el primer parámetro es una cadena de caracteres, en la cual hay una palabra que comienza por "%", seguida de un carácter. El carácter % indica que se rellenará con una variable pasada a printf por parámetros, y el siguiente carácter indicará el tipo de esa variable ("d", entero decimal). Dado que se pide imprimir una variable decimal, la variable "var1" debería haber sido declarada como un "integer". A continuación, se lista algunos de los tipos de datos más usados, soportados por printf:

specifier	**Output**	**Example**
`%d` *or* `%i`	Signed decimal integer	`392`
`%u`	Unsigned decimal integer	`7235`
`%o`	Unsigned octal	`610`
`%x`	Unsigned hexadecimal integer	`7fa`
`%X`	Unsigned hexadecimal integer (uppercase)	`7FA`
`%f`	Decimal floating point, lowercase	`392.65`
`%F`	Decimal floating point, uppercase	`392.65`
`%e`	Scientific notation (mantissa/exponent), lowercase	`3.9265e+2`
`%E`	Scientific notation (mantissa/exponent), uppercase	`3.9265E+2`
`%g`	Use the shortest representation: `%e` or `%f`	`392.65`
`%G`	Use the shortest representation: `%E` or `%F`	`392.65`
`%a`	Hexadecimal floating point, lowercase	`-0xc.90fep-2`
`%A`	Hexadecimal floating point, uppercase	`-0XC.90FEP-2`
`%c`	Character	`a`
`%s`	String of characters	`sample`
`%p`	Pointer address	`b8000000`
`%n`	Nothing printed. The corresponding argument must be a pointer to a `signed int`. The number of characters	

	written so far is stored in the pointed location.	
%%	A % followed by another % character will write a single % to the stream.	%

Ejemplos de uso:

```
float a=10.2;
int b=11;
char c='y';
printf("las variables valen: %f,%d,%c\n",a,b,c,d);
```