Fundamentos de la programación y la informática
Test final
19 de Enero de 2022

Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

- Ejecuta en un terminal la orden
  `~mortuno/prepara`
- Comprueba que esto ha dejado en tu cuenta el fichero
  `~/fpi.enero.22/test.TULOGIN.txt`, donde deberás resolver el ejercicio [1].

Dados los párrafos siguientes, indica para cada uno de ellos si lo que dice es cierto o es falso. Justifica brevemente tu respuesta. Si hay más de un error, señálalos todos. Ambas cosas (justificar las respuestas y señalar todos los errores) son imprescindibles: si de cada párrafo lo único que dices es *cierto* o *falso*, tu respuesta no tendrá casi ningún valor.

Párrafo 1

Supongamos que:

1. Queremos escribir un programa que almacene y gestione los datos de los clientes de una empresa.

2. El programa debe funcionar para cualquier número de clientes, ya sean 2, 2000 o 20000 (siempre que el ordenador sea lo bastante potente)

Si usamos un lenguajes de programación *tradicional* como Pascal o C, esto no es demasiado complicado, porque siempre podemos usar *memoria estática*, esto es, punteros.

Lenguajes más modernos como Python o Java permiten usar *memoria dinámica*, esto es, arrays. También sirven aunque a priori no conozcamos el número de clientes, pero son más complicados de programar.

Párrafo 2

En Pascal, si un programador olvida cerrar un fichero el resultado es similar al de una persona que olvida cerrar un grifo: el disco se llenará de *datos basura* y posiblemente se perderá todo.

Soluciones

1. Falso, por los siguiente motivos:

   - En lenguajes tradicionales, resolver el problema indicado tiene cierta dificultad, porque hay que usar punteros.
   - Usar punteros es usar memoria dinámica, no memoria estática.
   - Los lenguajes más modernos se ocupan automática de la gestión de la memoria dinámica. No es necesario que lo gestione el programador, por lo que resulta más sencillo.

[1] TULOGIN será tu nombre de usuario en el laboratorio

2. Falso. El programador debería cerrar los ficheros, pero si se olvida, no suele resultar fatal porque al acabar la ejecución del programa, el sistema operativo se ocupa de cerrarlos [2].

[2]El inconveniente puede ser, por ejemplo, que si un programador abre un fichero en modo escritura, olvida cerrarlo y el programa tarda en acabar, durante ese tiempo otros programas no podrán acceder al fichero. Otro problema puede darse si hay muchas instancias del mismo programa abriendo un fichero en modo lectura, sin cerrarlo. Podrían superar el número máximo de veces que se puede abrir un fichero. No es demasiado frecuente porque este número suele ser alto, pero puede suceder.