Fundamentos de la programación y la informática
Test final. 18 de junio de 2026
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

- Contesta brevemente las siguientes preguntas.

Ejercicio 1 (2.5 puntos)

Sean a, b, c, d variables booleanas. Sean x e y variables reales. Sean las expresiones

```
e1 := not ( a or  not b) or not (not c or not d);

e2 :=  not (x <= -2) and  (y >0 );
```

1. A partir de *e1*, escribe una expresión lógica en Pascal, equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica en Pascal, equivalente, más legible, de nombre *s2*.
3. Escribe una expresión en Pascal *n1* que sea la negación de *s1*.
4. Escribe una expresión en Pascal *n2* que sea la negación de *s2*.

Respuesta

```
s1 := (not a and b) or (c and d);
s2 := (x > -2) and (y > 0);
n1 := (a or not b) and (not c or not d);
n2 := (x <= -2) or (y <= 0);
```

Teniendo en cuenta la precedencia de operadores en Pascal (`not > and  > or`): ¿Qué sucede si omitimos los paréntesis?

1. La expresión `s1 := not a and b or c and d;` es equivalente. Pero no es más legible que la original, todo lo contrario. Por tanto esta respuesta sería incorrecta.
2. Omitir los paréntesis en n1 tiene consecuencias distintas. Si el único problema fuera que resulta menos claro, como en este apartado el enunciado no dice nada sobre la legibilidad, la respuesta sería válida.

   Pero no es el caso. Si escribiésemos

   `n1 := a or not b and not c or not d;`

   esto se intepretaría como

   `a or ((not b) and (not c)) or (not d);`,

   que es completamente diferente. Por tanto, también es incorrecta.

Ejercicio 2 (2.5 puntos)

¿Qué es un *error en tiempo de ejecución*? Comparado con otros tipos de errores ¿es relativamente complicado de detectar o relativamente sencillo?

Respuesta

Un error en tiempo de ejecución (*runtime error*) es un error que no se detecta al compilar el programa, sino cuando éste se está ejecutando. Por ejemplo dividir entre cero, acceder a una posición inexistente de un array, intentar abrir un fichero que no existe, quedarse sin memoria, entre otros.

Los errores de compilación son mucho más fáciles de detectar: los ve el compilador. Los errores lógicos normalmente son más difíciles de detectar: hay que conocer la respuesta correcta y darse cuenta de que la que está dando el programa, es otra. Los *errores en la claridad del código* [1] también pueden resultar bastante complicados de detectar: hay que tener el código fuente y ser capaz de analizarlo.

Ejercicio 3 (2.5 puntos)

Explica brevemente la diferencia entre *paso por valor* y *paso por referencia*.

Respuesta

El paso por valor consiste en que un subprograma recibe como parámetro una copia del valor de una variable. Por tanto, las modificaciones realizadas sobre el parámetro dentro del subprograma no afectan a la variable original.

En cambio, en el paso por referencia el subprograma recibe una referencia a la variable original. O, informalmente, *recibe la variable original*. Cualquier modificación realizada sobre el parámetro en el subprograma afecta directamente a dicha variable, incluso cuando el subprograma ha finalizado

Ejercicio 4 (2.5 puntos)

1. ¿Qué significa *reservar memoria*? ¿Cuándo se hace? ¿Quién lo hace?
2. ¿Qué significa *liberar memoria*? ¿Cuándo se hace? ¿Quién lo hace?

Respuesta

1. Reservar memoria significa asignar una zona de la memoria principal para almacenar datos o variables que va a utilizar un programa.
   - A veces esto es automático [2] . Por ejemplo para los enteros, reales, booleanos, etc.
   - Cuando se trata de memoria dinámica, por ejemplo una lista encadenada, entonces depende del lenguaje:
     - En lenguajes *tradicionales* como C o Pascal, lo hace el programador.
     - En lenguajes *modernos* como Python, Java o JavaScript, suele ser automático, no es responsabilidad del programador.

[1] Esta categoría no es estándar.

[2] lo hace el entorno de ejecución del programa y el sistema operativo.

2. Liberar memoria significa devolver al sistema una zona de memoria que ya no va a utilizarse, de forma que pueda ser reutilizada posteriormente.

   Para enteros, reales, booleanos, etc es automático. En el caso de la memoria dinámica, si reservó el programador, también libera el programador. Si la reserva es automática, también lo es la liberación (lo hace el *recolector de basuras*).