71901020		Fundamentos de Programación		901
		MULTIPLES GRADOS		71
	Septiembre - 2012 Reserva	Duración: 120 min.	EXAMEN: Tipo B Mixto	Nacional - U.E. 1º Cuatrimestre
[-Ç□ýÍ] Material: Ninguno				Hoja 1 de 1

FUNDAMENTOS DE PROGRAMACIÓN. Convocatoria: **Septiembre**. Semana: **Reserva Nacional / U.E.** Tipo de examen: **B**

El **test** debe ser contestado en la **hoja de lectura óptica**. Sólo una de las cuatro respuestas posibles de cada pregunta es correcta.
El test es eliminatorio y aporta un 30% de la nota final. Son necesarias 8 preguntas correctas (6 con las prácticas aprobadas) para que se corrija el ejercicio.
Cada respuesta correcta: 1 punto. Respuesta incorrecta o en blanco: 0 puntos.

1. En el lenguaje C±, la sentencia:

   ```
   algo++;
   ```

   Se puede utilizar:
   A. En la parte declarativa pero no en la ejecutiva de un programa
   B. En la parte ejecutiva pero no en la declarativa de un programa
   C. Tanto en la parte declarativa como en la ejecutiva de un programa
   D. Sólo formando parte de un bucle `for`

2. El siguiente fragmento de código en C±:

   ```
   typedef int TipoVector[5];
   ...
   TipoVector vector;
   for(int i = 0; i <= i; i++){
     printf("%d", vector[i]);
   }
   ```

   A. Ejecuta la sentencia de impresión 0 veces
   B. Entra en un bucle infinito
   C. Ejecuta la sentencia de impresión 1 vez
   D. Ejecuta la sentencia de impresión 5 veces

3. En el lenguaje C±, las unidades de compilación son:
   A. Sólo el programa principal y el fichero de implementación de los módulos
   B. Sólo el fichero de implementación de los módulos
   C. El programa principal, el fichero de interfaz y el fichero de implementación de los módulos
   D. Sólo el programa principal y el fichero de implementación de los módulos

4. Dado el siguiente fragmento de programa en C±:

   ```
   const int v[],
   ```

   Es una declaración de:
   A. Un argumento de tipo vector abierto
   B. Una constante de tipo cadena
   C. Un argumento de tipo cadena
   D. Una constante de tipo vector abierto

5. El uso de variables globales:
   A. No tiene por qué evitarse
   B. Sólo puede evitarse mediante el paso de argumentos por valor
   C. Sólo puede evitarse mediante el paso de argumentos por referencia
   D. Puede evitarse mediante el paso de argumentos por valor y por referencia

6. ¿Cuál de los siguientes órdenes de complejidad de crecimiento asintótico es más eficiente?
   A. $O(\log n)$
   B. $O(n \log n)$
   C. $O(n^2)$
   D. $O(2^n)$

7. Dado el siguiente fragmento de programa en C±, con un valor de `j` positivo:

   ```
   while(j > 0){ v[j] = v[j-1]; j--;}
   ```

   Se produce:
   A. La inserción de un nuevo elemento en el vector `v`
   B. La búsqueda de un cierto elemento en el vector `v`
   C. Un desplazamiento de los elementos del vector `v`
   D. Un bucle que no finaliza nunca

8. Respecto al número de elementos podemos afirmar que:
   A. Las secuencias enlazadas son ilimitadas y las formaciones limitadas
   B. Las secuencias enlazadas y las formaciones son limitadas
   C. Las secuencias enlazadas y las formaciones son ilimitadas
   D. Las secuencias enlazadas son limitadas y las formaciones ilimitadas

9. Las redes de operadores se utilizan en:
   A. El modelo abstracto de cómputo funcional
   B. El modelo abstracto de cómputo lógico
   C. El modelo abstracto de cómputo imperativo
   D. El modelo abstracto de cómputo flujo de datos

10. Dada la función `posicion` en C±:

    ```
    typedef char Vchar[40];
    int posicion(Vchar cad, char c){
      int pos = -1; int i = 0;
      while (cad[i] != '\0'){
        if(cad[i] == c){ pos = i;}
        i++;
      }
      return(pos);
    }
    ```

    Si la llamamos con `posicion("EJEMPLO DE", 'E')`:
    A. La función `posicion` devuelve `0`
    B. La función `posicion` devuelve `9`
    C. La función `posicion` devuelve `2`
    D. La función `posicion` devuelve `-1`

Ejercicio de Programación

Realizar en C± un TAD, con fichero de interfaz y de implementación, para manejar una flota de hasta 50 vehículos. De cada vehículo se almacena su matrícula (4 números y 3 letras) y los kilómetros que lleva recorridos. Las operaciones a realizar son: **Insertar**, que inserta un nuevo vehículo en la flota, iniciando sus kilómetros a 0; **Aumentar**, que recibe una matrícula y el número de kilómetros a incrementar y aumenta los kilómetros del vehículo cuya matrícula se pasa como argumento, devolviendo `true` en caso de que la matrícula exista y `false` en otro caso; **Total**, que devuelve el número de kilómetros totales que han recorrido todos los vehículos de la flota.