FUNDAMENTOS DE PROGRAMACIÓN. Convocatoria: **Febrero**. Semana: **2ª**. 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 función:

```
int main()
```

   A. Se tiene que utilizar en todas las unidades de compilación
   B. Se tiene que utilizar en todos los ficheros con extensión `cpp`
   C. Sólo se puede utilizar en el módulo principal
   D. Sólo no se puede utilizar en el programa principal y en el módulo de interfaz

2. Si antes de la ejecución del siguiente fragmento de código en C±, `vector` contiene los valores `{7,4,5,3,6}`:

```
for (int i = 1; i < 5; i++){
  tmp = vector[i];
  j = i;
  while((j > 0) && (tmp > vector[j-1])){
    vector[j] = vector[j-1];
    j--;
  }
  vector[j] = tmp;
}
```

   A. Al terminar el bucle, `vector` tendrá `{3,4,5,6,7}`
   B. Al terminar el bucle, `vector` tendrá `{7,4,5,3,6}`
   C. Al terminar el bucle, `vector` tendrá `{7,5,4,3,6}`
   D. Al terminar el bucle, `vector` tendrá `{7,6,5,4,3}`

3. Los identificadores en C±:
   A. Pueden tener mayúsculas y minúsculas
   B. Deben empezar por mayúscula
   C. Pueden incluir cualquier signo de puntuación
   D. Deben incluir dígitos

4. ¿Qué sentencias hay que sustituir por el comentario para que la función `fun` realice la suma de los `n` (siendo `n` mayor que 0) primeros números que sean impares?

```
int fun(int n){
  /* SUSTITUIR */
}
```

   A. if(n<=1) {return 0;} else {return((2*n-1)+fun(n-1));}
   B. if(n<=1) {return 1;} else {return((2*n-1)+fun(n-1));}
   C. if(n<=1) {return 1;} else {return((2*n)+fun(n-1));}
   D. if(n<=1) {return 1;} else {return((2*(n-1))+fun(n-1));}

5. En C±, la sentencia `switch` equivale a:
   A. Una sentencia de selección
   B. Una sentencia de iteración
   C. Una sentencia de asignación
   D. Una sentencia de importación

6. En C± cuando se utilizan argumentos de tipo formación y no se quiere que se modifiquen los parámetros reales en la llamada al procedimiento, los argumentos formales deben ir precedidos de:
   A. `const`
   B. `private`
   C. `var`
   D. `&`

7. Dada la siguiente sentencia del lenguaje C±:

```
Algo(uno % dos);
```

   A. Es la cabecera de una función con un argumento por referencia
   B. Es la llamada a un procedimiento con un argumento por valor
   C. Es la cabecera de una función con dos argumentos por referencia
   D. Es la llamada a un procedimiento con un argumento por referencia

8. ¿Qué modelo abstracto de cómputo sigue C±?
   A. Modelo de programación funcional
   B. Modelo de flujo de datos
   C. Modelo de programación imperativa
   D. Modelo de programación lógica

9. En C±, una cadena de caracteres de un máximo de 20 caracteres se define como:
   A. `typedef char Cadena[18];`
   B. `typedef char Cadena[19];`
   C. `typedef char Cadena[20];`
   D. `typedef char Cadena[21];`

10. Supongamos el fragmento de código en C±:

```
int x;
void P(int y, int & z) {
  x = x-1;
  y = y+3;
  z = z+2;
}
...
x = 2;
P(x+1,x);
```

    Después de la ejecución de `P(x+1,x)`:
    A. La variable `x` vale `1`
    B. La variable `x` vale `3`
    C. La variable `x` vale `4`
    D. La variable `x` vale `8`

Ejercicio de Programación

Realizar en C± el TAD **DatosMultaTrafico** para guardar el estado de pago de una multa. Los datos son DNI (ristra de 10 caracteres), estado de la multa (inicial, notificada, recurrida, pagada), cuantía (real), puntos (entero). La operación **CambiarMulta** permite cambiar el estado y los puntos de la multa. La operación **AplicarDescuento** reduce la cuantía en el porcentaje introducido (real). La operación **MostrarMulta** escribe los datos de la multa.