Pregunta 1

¿Cuál de las siguientes funciones tiene la **mayor** complejidad computacional?

Ocultar opciones de respuesta

A Incorrecta:
Todas las funciones tienen la misma complejidad computacional.

B

```
int function3(n)
{
  for (int i = 0; i < n; i++)
    function3(n / 2);
}
```

C

```
int function1(vector<char> data)
{
 int count = 0;
 for (char a : data)
   for (char b : data)
     if (a == b)
       count++;

 return count;
}
```

D

```
void function4(int n)
{
 int result = 0;
 for (int i = 0; i < n; i += 2)
  for (int j = 0; j < n; i += 3)
 result += i * j;
return result;
}
```

E

```
int function2(n)
{
  for (int i = 0; i < n; i++)
    function2(n - 1);
}
```

Pregunta 2

¿Cuál es el mejor tipo de datos para implementar **streaming de vídeo por Internet**?

Ocultar opciones de respuesta

A. Una lista simplemente enlazada.

B. Una pila con deque.

C. Una lista doblemente enlazada.

D. Una lista con arreglo.

E. Una cola de prioridad con lista con arreglo.

F. Una cola con lista doblemente enlazada.

Pregunta 3

¿A qué tipo de **eficiencia** debemos prestarle más atención?

Ocultar opciones de respuesta

A. A la complejidad computacional.

B. A la complejidad de memoria.

C. A reducir las dependencias con otros objetos y bibliotecas.

D. A la claridad del código.

E. Debemos atender el cuello de botella particular que tiene nuestro problema.

Pregunta 4

¿Cuál es una de las razones por las que debemos cuidar la **indentación** en C++?

Ocultar opciones de respuesta

A. Para distinguir los comentarios de los datos interiores.

B. Para distinguir los datos miembro entre sí.

C. Para distinguir los comentarios de los datos exteriores.

D. Para distinguir los comentarios del código.

E. Para distinguir los métodos de los datos miembro.

Pregunta 5

¿Qué **ocurre** si ejecutamos el siguiente código?

```
#include <stack>

int main(int argc, char *argv[])
{
 std::stack<int> s;
 int n1, n2, n3;

 s.push(17);
 s.push(143);
 s.push(42);
 n1 = s.top();
 s.pop();
 n2 = s.top();
 s.push(n1);
 n3 = s.top();
 s.pop();
 n1 = s.top();
}
```

Ocultar opciones de respuesta

A. s está vacío, n1 vale 17, n2 vale 143, n3 vale 42.

B. s contiene {17, 42 [último elemento en entrar en la pila]}, n1 vale 42, n2 vale 143, n3 vale 143.

C. s contiene {17, 143 [último elemento en entrar en la pila]}, n1 vale 143, n2 vale 143, n3 vale 42.

D. s contiene {17, 143, 42, 42 [último elemento en entrar en la pila]}, n1 vale 42, n2 vale 42, n3 vale 42.

E. s contiene {17, 143 [último elemento en entrar en la pila]}, n1 vale 42, n2 vale 42, n3 vale 42.

Pregunta 6

¿Por qué es preferible usar el mecanismo de **salida a consola** de C++ en lugar de usar `printf`?

Ocultar opciones de respuesta

(A) Porque es más eficiente en cálculo.

(B) **Porque asegura la seguridad de los tipos de datos.**

(C) Incorrecta:
Porque es más eficiente en memoria.

(D) Porque asegura que los datos se impriman en el orden correcto.

(E) La afirmación es falsa. Es preferible usar printf.

Pregunta 7

En una clase de C++, siempre debemos...

Ocultar opciones de respuesta

(A) colocar los métodos en la sección public: y los datos miembros en la sección private:.

(B) **colocar la interfaz en la sección public:.**

(C) colocar los contenedores en la sección public:.

(D) colocar el modelo en la sección private: y la vista en la sección public:.

(E) colocar los datos miembro en la sección public:.

Pregunta 8

¿Por qué debemos limitar el **alcance** de las variables y funciones lo más que podamos?

Crédito parcial
(A) Para evitar que el ..., (B) Para separar la interfaz ..., (C) Para que el código sea má... y (D) Para evitar errores en el... son correctas

Ocultar opciones de respuesta

- **A** Para evitar que el usuario de las clases pueda acceder por accidente a datos miembro y métodos privados.
- **B** Para separar la interfaz de la implementación.
- C Para que el código sea más claro.
- **D** Para evitar errores en el código.
- E Para simplificar el acceso a los datos.

Pregunta 9

La **sobrecarga de métodos**...

Ocultar opciones de respuesta

- A impide que se llame al destructor erróneo.
- **B** suele aplicarse cuando los métodos realizan tareas similares.
- C evita que el usuario de una clase pueda confundirse.
- D impide que se produzcan accesos a memoria inválidos.
- E no puede usarse con contenedores.

Pregunta 10

¿Por qué debemos **factorizar** el código?

Crédito parcial
(A) Para descomponer el có..., (B) Para evitar redundancia ..., (C) Para evitar, a largo ..., (D) Para evitar errores, ya ... y (E) Para escribir código más ... son correctas

Ocultar opciones de respuesta

- A Para descomponer el código en partes más fáciles de entender.
- **B** Para evitar redundancia que, en caso de error, debamos revisar múltiples veces.
- **C** Para evitar, a largo plazo, trabajo innecesario.
- **D** Para evitar errores, ya que simplificará la revisión de código.
- **E** Para escribir código más claro, fácil de entender.