Los constructores...

Respuesta seleccionada: ✓ sirven para inicializar los objetos.

Respuestas:
- ✓ sirven para inicializar los objetos.
- Ninguna de las otras opciones.
- sirven para definir los setters y getters.
- sirven para abstraer la interfaz.
- son necesarios para que se pueda definir un destructor.
- sirven para asegurarse que no existan dos objetos repetidos.

Pregunta 2

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

Respuesta seleccionada: ✗ Ninguna de las otras opciones.

Respuestas:
- Ninguna de las otras opciones.
- ✓ Para distinguir los comentarios del código.
- Para distinguir los comentarios de los datos exteriores.
- Para distinguir los comentarios de los datos interiores.
- Para distinguir los datos miembro entre sí.

Pregunta 3

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

Respuestas seleccionadas:
- ✓ Para impedir que alguien modifique datos o acceda a métodos por error.
- ✓ Para separar la interfaz de la implementación.

Respuestas:
- ✓ Para evitar errores en el código.
- ✓ Para impedir que alguien modifique datos o acceda a métodos por error.
- ✓ Para separar la interfaz de la implementación.

Pregunta 4

En C++, ¿cuál es la diferencia entre `delete` y `delete[]`?

Respuesta seleccionada: ✓ `delete` se usa para liberar un solo objeto, mientras que `delete[]` se usa para liberar arreglos de objetos.

Respuestas:
- `delete[]` comprueba los errores, y por tanto no puede producir los fallos que puede producir `delete`.
- `delete` se usa para eliminar objetos en el stack, mientras que `delete[]` se usa para borrar objetos apuntados por punteros.
- `delete` es una palabra clave mientras que `delete[]` es un identificador.
- No hay diferencia. Se distinguen sólo porque ayudan a escribir código más prolijo.
- Ninguna de las otras opciones.
- ✓ `delete` se usa para liberar un solo objeto, mientras que `delete[]` se usa para liberar arreglos de objetos.

Pregunta 5

Para acceder rápidamente a un dato por una clave, ¿qué contenedor STL es preferible?

Respuesta seleccionada: ✗ `map`

Respuestas:
- `set`
- `list`
- `priority_queue`
- `map`
- `vector`
- ✓ `unsorted_map`

Pregunta 6

Cuando un programa posee una interfaz de usuario, conviene separarlo en modelo, vista y controlador porque esto...

Respuestas seleccionadas:
- ✓ ayuda a reducir las dependencias entre las partes del código.
- ✓ fomenta la reutilización de código.

Respuestas:
- ✓ ayuda a reducir las dependencias entre las partes del código.
- ✓ fomenta la reutilización de código.
- ayuda a crear cohesión de código.
- reduce el uso de recursos computacionales.

Pregunta 7

Los destructores...

Respuesta seleccionada: ✓ tienen por fin liberar los recursos reservados por un objeto.

Respuestas:
- ✓ tienen por fin liberar los recursos reservados por un objeto.
- suelen llamarse explícitamente cuando ya no se requiere el objeto.
- requieren que se definan constructores.
- se escriben en función del número de datos miembro que existen en la clase.
- no deberían usarse en exceso porque tienen un costo computacional muy alto.
- Ninguna de las otras opciones.

Pregunta 8

¿Por qué debemos factorizar el código?

Respuestas seleccionadas:
- ✓ Para evitar errores, ya que simplificará la revisión de código.
- ✓ Para escribir código más claro, fácil de entender.
- ✓ Para evitar, a largo plazo, trabajo innecesario.

Respuestas:
- ✓ Para evitar redundancia que, en caso de error, debamos revisar repetidamente.
- ✓ Para evitar errores, ya que simplificará la revisión de código.
- ✓ Para escribir código más claro, fácil de entender.
- ✓ Para evitar, a largo plazo, trabajo innecesario.

Pregunta 9

¿Por qué es importante escribir código independiente de la plataforma?

Respuestas seleccionadas:
- ✓ No siempre es importante. Existen casos en los que no tiene sentido hacerlo.

Respuestas:
- ✓ No siempre es importante. Existen casos en los que no tiene sentido hacerlo.
- ✓ Porque permite abarcar a un público más grande con relativamente poco esfuerzo.
- Porque está bien visto por toda la comunidad de programadores.
- Porque demuestra que somos competentes en C++.

Pregunta 10

Los templates de clase...

Respuesta seleccionada: ✓ Permiten reutilizar el mismo código con diferentes tipos de datos.

Respuestas:
- Permiten reutilizar el mismo tipo de datos con diferentes funciones.
- Permiten reutilizar el mismo objeto con diferentes clases.
- Permiten reutilizar el mismo código con diferentes constructores.
- ✓ Permiten reutilizar el mismo código con diferentes tipos de datos.
- Permiten reutilizar el mismo tipo de datos con diferentes clases.
- Ninguna de las otras opciones.