

¿Cuántos objetos *Fecha* se crean aquí (C++)?


```
Fecha f1, f2(12,9,2016);
```

```
Fecha *f3 = new Fecha(f2);
```

```
Fecha *f4 = f3;
```

```
Fecha& f5 = f2;
```

Seleccione una:

☒ 3  Correcto, se trata de los objetos f1, f2 y f3.

☐ 2

☐ 4

☐ 5

☐ 1

Respuesta correcta


Debes contar objetos 'automáticos', que se crean en la pila (stack) y objetos creados con 'new', que se crean en el 'heap'. También debes saber cuando una asignación entre objetos, punteros o referencias realiza una copia de un objeto (y por tanto crea uno nuevo) y cuando simplemente estamos copiando la dirección de un objeto, sin duplicarlo.

¿En. C++, porqué debemos usar 'delete' con objetos creados mediante 'new'?

Seleccione una:

- ☐ Porque si no lo hacemos provocaremos tarde o temprano un error de 'violación de segmento' ('segmentation fault')
- ☐ No es necesario, solo es conveniente hacerlo.
- ☐ Porque nos obliga C++.
- ☒ Porque es responsabilidad del programador administrar la memoria dinámica del programa. ✓

```
Fecha f1(12,9,2016);
Fecha *f5;
if ( ... ) {
    Fecha f2(f1);
    ...
} else if ( ... ) {
    Fecha f3(f1);
    ...
} else if ( ... ) {
    Fecha *f4 = new Fecha(f1);
    f5=f4;
    delete f5; f5=NULL;
}
// ¿Cuántos objetos 'Fecha' hay en este punto del código?
```

Respuesta: 

Sólo f1 ha sobrevivido. f2 y f3 han sido destruidos al terminar el bloque donde se habían definido. f4 y f5 son dos punteros apuntando al mismo objeto, que es destruido al hacer 'delete'.

¿Dónde se guarda el valor de un atributo de instancia de un objeto que está en memoria?

Seleccione una:


- ☐ En la memoria estática.
- ☐ En el disco duro.
- ☒ En la memoria que se reservó para el objeto al crearlo. ✓
- ☐ En una zona especial de la memoria destinada al almacenamiento de atributos, distinta de donde se guardan los objetos y sus punteros
- ☐ En la memoria que se reservó para la clase del objeto al crearlo.

¿Por qué ocultamos los datos (atributos) en una clase?

Seleccione una o más de una:


- ☐ Para poder usar los nombres de atributos que queramos sin tener que publicarlos.
- ☒ Para evitar que el código cliente pueda manipular el estado de nuestros objetos de forma que queden en un estado inconsistente o corrupto (p. ej., que el día de una fecha sea negativo o mayor que 31).
✓
- ☐ Para poder tener atributos que sólo usamos internamente en los métodos (funciones) de la clase pero que el código cliente no necesita.
- ☒ Para ocultar los detalles de implementación de nuestra clase al código cliente. ✓

Seleccione una:

- ☐ Un constructor de oficio
- ☒ Un constructor por defecto  El constructor por defecto nos permite inicializar los atributos de Fecha al crear el objeto. Creación del objeto e inicialización de sus propiedades son dos acciones que los lenguajes como C++ o Java realizan de forma conjunta mediante los constructores.
- ☐ Sobrecargar el operador de entrada
- ☐ Un método de instancia que lo haga, como por ejemplo Fecha::inicializar()

¿Cómo podemos crear objetos Fecha con valores iniciales diferentes?

Seleccione una:

- ☐ Implementando un método de instancia, p. ej. `Fecha::set(int dia, int mes, int anyo)`.
- ☐ Sobrecargando el operador de entrada.
- ☐ Implementando un constructor de copia.
- ☒ Implementando un constructor sobrecargado.  Aunque `Fecha::set(int dia, int mes, int anyo)` hace prácticamente el mismo trabajo que el constructor sobrecargado (de hecho, podríamos llamar al método desde el constructor), éste lo hace justo en el momento en el que se crea el objeto, garantizando que el objeto 'nace' con esos valores iniciales, antes de que podamos hacer uso de él de alguna forma, cosa que no está garantizada con el método `set(...)`.

¿Cómo podemos crear fechas que sean copia de otras?

Seleccione una:

- ☐ Implementando un constructor sobrecargado.
- ☒ Implementando un constructor de copia. ✓
- ☐ Implementando un método que haga la copia, como `Fecha::copiar(Fecha f)`
- ☐ Implementando un constructor sobrecargado que luego usamos para copiar la fecha : `Fecha f2(f1.getDía(), f1.getMes(), f1.getAnyo())`.

Cuando decimos que un objeto que hemos creado 'ha sido eliminado', 'ya no existe', ¿qué estamos diciendo exactamente?

Seleccione una:

- ☒ Que el objeto ha sido eliminado de la memoria de nuestro programa y ya no es accesible. ✓
- ☐ Que la variable mediante el cual nos referimos a él ya no existe porque hemos salido del ámbito donde fue definida.
- ☐ Que hemos hecho 'delete'.

¿Qué pasa cuando un objeto 'desaparece'?

Seleccione una o más de una:

- ☒ Es eliminado de la memoria de nuestro programa. ✓
- ☐ Todas las variables que hacen referencia a él (punteros, referencias) ya no se pueden usar.
- ☒ Se ejecuta el destructor de la clase sobre ese objeto (C++) ✓