Rule of 3 - Q&A

# Content

[Rule of 3 - Q&A 1](#_Toc531258578)

[1. Content 1](#_Toc531258579)

[2. Questions 1](#_Toc531258580)

[2.1. RuleOf3Basics 1](#_Toc531258581)

# Questions

Add your answers in the white boxes (in Dutch or English).

## RuleOf3Basics

1. Crash details

|  |
| --- |
| In which function of the Container class happens the crash? |
| In de destructor |
| Where in the Game class was this function called? (which function and where in that function, use the Call Stack window) |
| Op het einde van de testcontainer functie |
| What happens there regarding the Container objects |
| De destructors worden gecalled |

1. Investigate the content of the variables just before the crash happens and draw your conclusions from this test.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Compare the content of both Container objects c1 and c2   |  |  | | --- | --- | | Which data? | Where can you find it? | | m\_Size | Locals window | | m\_Capacity | Locals window | | Pointer to the dyn array: m\_pElement | Locals window | | Array elements: m\_pElement,3 | Watch window |   Do they have the same values? If not, which one are different? |
| Alle data members zijn hetzelfde |
| What can you conclude about the dynamic array in both Container objects? |
| Ze wijzen allebij naar dezelfde array |
| Conclude: What happens by default when you create an object initializing it with another object of the same type? |
| Een default copy constructor wordt gegenereerd |
| Why does a crash happen when the containers c1 and c2 go out of scope? |
| Als de destructor van c2 wordt gecalled probeert deze de array te deleten maar die is al gedeletet door c1 |

1. Changing an element in one of the containers.

|  |
| --- |
| What happens when you change an element in one of the 2 containers |
| Het verandert ook in de andere |
| Why? |
| Omdat ze allebij naar dezelfde array wijzen |

1. Investigate the content of the variables related to the containers c1 and c2 just before they go out of scope.

|  |
| --- |
| Which data member(s) have the same value? |
| De capacity, de size, de elementen in de array |
| Which data member(s) have a different value? |
| De pointer naar de array |
| Does changing the content of a container element, still influence the content of the elements in the other container? |
| nee |

1. Crash details

|  |
| --- |
| In which function of the Container class happens the crash? |
| In de destructor |
| Where in the Game class was this function called? (which function and where in that function, use the Call Stack window) |
| Op het einde van de testContainer functie |

1. Investigate the content of the variables just before the crash happens and draw your conclusions from this test.

|  |
| --- |
| Compare the content of both Container objects c1 and c3  Do they have the same values? If not, which one are different? |
| Alle data members zijn hetzelfde |
| What can you conclude about the dynamic array in both Container objects? |
| Is dezelfde |
| Conclude: What happens by default when you assign a Container object to another one ? |
| Een default assignment operator wordt gegenereerd |
| Why does a crash happen when the containers go out of scope ? |
| Als de destructor van de tweede wordt gecalled probeert deze een array te deleten die al gedeletet is |

1. Investigation of what happens when an integer value is assigned to a Container object:   
   **c3 = 4;**

|  |
| --- |
| Using the “Step Into” Debugger button, give a list of Container functions (only mention constructor, copy constructor, assignment operator or destructor) when this statement is executed.  Write them down in order of execution.  When the constructor is called also write down the value of the capacity parameter. **Don’t mention the destructors** that are called when the 3 containers go out of scope at the closing curly brace of TestContainer. |
| Constructor: size = 0, capacity = 4 |
| Assignment operator |
| Destructor van de container met capacity 4 |

1. Investigation of what happens when this code is executed:   
   Container c4 = c1;

|  |
| --- |
| Only mention constructor, copy constructor, assignment operator or destructor when this statement is executed.  Write them down in order of execution.  **Don’t mention the destructors** that are called when the 4 containers go out of scope at the closing curly brace of TestContainer. |
| Copy constructor |
|  |
|  |
|  |

1. Investigation of what happens when this code is executed:   
   c4 = c2;

|  |
| --- |
| Only mention constructor, copy constructor, assignment operator or destructor when this statement is executed.  Write them down in order of execution.  **Don’t mention the destructors** that are called when the 4 containers go out of scope at the closing curly brace of TestContainer. |
| Assignment operator |
|  |
|  |
|  |

1. Investigation of what happens when this code is executed:  
   c4 = CreateMultiplied(c1, 2);

|  |
| --- |
| Only mention constructor, copy constructor, assignment operator or destructor when this statement is executed. But also indicate why one was called.  Write them down in order of execution.  **Don’t mention the destructors** that are called when the 4 containers go out of scope at the closing curly brace of TestContainer. |
| Copy constructor c1 wordt gekopieerd naar c |
| Copy constructor de returned constructor wordt gekopieerd naar c1 |
| Destructor c |
| Assignment operator de waarden in c4 worden de waarden van in c1 |
| Destructor |
|  |
|  |

1. Investigation of what happens when this code is executed:  
   AddValues(c4, 3, 20, 30);

|  |
| --- |
| Only mention constructor, copy constructor, assignment operator or destructor when this statement is executed. Write them down in order of execution.  **Don’t mention the destructors** that are called when the 4 containers go out of scope at the closing curly brace of TestContainer. |
| Er wordt geen constructor, copy constructor, assignment operator of destructor uitgevoerd |
|  |
|  |
|  |

1. Creating a **static** Texture object

|  |
| --- |
| During the creation of a static Texture object something goes wrong with as consequence that it can’t be drawn. When is the texture initialized? Why does the creation go wrong? |
| Helemaal in het begin word de texture initialized |

1. Assigning a Texture object to another one

|  |
| --- |
| When assigning a Texture object to another one, you get an error. Which deleted function are you trying to call? |
| Assignment operator |

1. When passing a Texture object by value to a function

|  |
| --- |
| What deleted function is attempted to call |
| Copy constructor |
| How can you solve this error without changing the Texture class? |
| Pass it by const reference |