**Rule of 3 - Q&A**

* **Content**
* **Questions**

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

* **RuleOf3Basics**
* Crash details

|  |
| --- |
| In which function of the Container class happens the crash? |
| In the Constructor |
| Where in the Game class was this function called? (which function and where in that function, use the Call Stack window) |
| In the Constructor (initializer) |
| What happens there regarding the Container objects |
| the c1 members were copied to c2 |

* 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? |
| They have the same values |
| What can you conclude about the dynamic array in both Container objects? |
| The dynamic array in both containers is the same, because m\_pElement is the same. Therefore, they point to the same object. The copy copied the member values from c1 into c2 |
| Conclude: What happens by default when you create an object initializing it with another object of the same type? |
| It copies all data members |
| Why does a crash happen when the containers c1 and c2 go out of scope? |
| Because there is no copy constructor in the Container class |

* Changing an element in one of the containers.

|  |
| --- |
| What happens when you change an element in one of the 2 containers |
| that element will also be changed in the second container |
| Why? |
| because m\_pElement points to the same memory address in both containers |

* 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? |
| Everydata memeber expect for m\_pElement[0] |
| Which data member(s) have a different value? |
| m\_pElement[0] |
| Does changing the content of a container element, still influence the content of the elements in the other container? |
| Before the copy constructor = yes  after = yes |

* Crash details

|  |
| --- |
| In which function of the Container class happens the crash? |
| In the deafult assignment operator |
| Where in the Game class was this function called? (which function and where in that function, use the Call Stack window) |
| RulleOf3Basics |

* 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? |
| No. Capacity is different as well as m\_pElement |
| What can you conclude about the dynamic array in both Container objects? |
| They do not point at the same array |
| Conclude: What happens by default when you assign a Container object to another one ? |
| The elemnts of the array were copied |
| Why does a crash happen when the containers go out of scope ? |
| it crashed because the capacities are different and therefore cannot copy all the elements |

* 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, copy constructor, destructor, |
| 0 |
|  |

* 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, destructor |
|  |
|  |
|  |

* 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. |
| assignemnt operator, destructor |
|  |
|  |
|  |

* 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 |
| copy constructor |
| destructor |
| assignment operator |
| destructor |
|  |
|  |

* 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. |
| destructor |
|  |
|  |
|  |

* 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? |
| The texture is initialized before the Game Constructor is called. because it is a static texture |

* 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? |
| Assignemnt Operator Function |

* When passing a Texture object by value to a function

|  |
| --- |
| What deleted function is attempted to call |
| Texture &Texture::operator =(const Texture &)': attempting to reference a deleted function  assignemnt operator |
| How can you solve this error without changing the Texture class? |
|  |