|  |
| --- |
| creeper1.jpg |
| Assignment 5 |
| CPTN230 |
|  |
| **Paul Fahey** |
| **11/11/2011** |



Contents

[Introduction 3](#_Toc308637793)

[Nothing Class Summary 3](#_Toc308637794)

[Nothing Class Conclusion 3](#_Toc308637795)

[Nothing Class Output 3](#_Toc308637796)

[Application Summary 4](#_Toc308637797)

[Object and Variable Functionality 5](#_Toc308637798)

[Pseudo Code 5](#_Toc308637799)

[Assignment 5 Creeper Header File 5](#_Toc308637800)

[Assignment 5 Zombie Header File 6](#_Toc308637801)

[Assignment 5 Creeper Class File 6](#_Toc308637802)

[Assignment 5 Zombie Class File 7](#_Toc308637803)

[Assignment 5 Application File 8](#_Toc308637804)

[Conclusions 9](#_Toc308637805)

[Captured Screen Output 9](#_Toc308637806)

[References 12](#_Toc308637807)

# Introduction

This assignment will take the existing "nothing" class and application and fill it in using cout statements to explicitly show their existence and purpose. Also this assignment is to create a class and demonstrate operator overloading with that class. The application will create multiple instances of the class chosen and demonstrate the use of overloaded C++ operators with built in data types of the language. There will also be demonstration of an overloaded constructor as well. Another class will also be created to make "friends" with the first class. A variety of tests will be performed to test the overloaded operators to make sure they are working as intended and compare both classes via these overloaded operators. (Bettle, Assignment 5 for CPTN230 Assignment Description)

# Nothing Class Summary

This program will explicitly display the default member functions of a C++ class. These member functions include the default constructor, default constructor, copy constructor, and copy assignment. These functions would have been automatically included if not defined and a couple of object instances will be created to show these functions in action using cout statements in each one. Once these functions are called upon by the creation of the object the proper cout statement will be displayed on the screen.

# Nothing Class Conclusion

This application was a good way to see and learn what was already provided behind the scenes. The cout statements really helped show me what each default function was doing in a simple and easy way to grasp the concept. I did have some trouble with explicitly defining the copy assignment function but eventually I figured out its structure. I learned that there definitely is a whole lot of something in the nothing class.

# Nothing Class Output

Welcome to the Nothing class tester for assignment 5

Hi I am the default constructor

Hi I am the default constructor

Hi I am the copy constructor

Hi I am the copy assignment

Thank you for using the Nothing class tester for assignment 5

Hi I am the Default destructor

Hi I am the Default destructor

Hi I am the Default destructor

Press any key to continue . . .

# Application Summary

This program will demonstrate the functionality of a creeper and zombie object which came from the sandbox building game minecraft. The user will be greeted to the program, and there will be a target, explosion, awareness, and direction set for a bunch of creeper and zombie instances displayed on the screen. Following this a message will be displayed saying if a test passed or not for the overloaded operators used comparing sets of creeper and zombie instances with each other and with their own instances.

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Creeper** | **Object** | **Zombie** |
| Creeper1 | 2 | Zombie1 | 2 |
| Creeper2 | 4 | Zombie2 | 4 |
| Creeper3 | 6 | Zombie3 | 6 |
| Creeper4 | 8 | Zombie4 | 8 |
| Creeper5 | 2 | Zombie5 | 2 |
| Creeper6 | 4 | Zombie6 | 4 |
| Creeper7 | 6 | Zombie7 | 6 |
| Creeper8 | 8 | Zombie8 | 8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Status** | **Result** | **Comment** |
| Creeper1==Creeper5 | Pass | True | == |
| Creeper2==Creeper3 | Fail | False | != |
| Creeper4!=Creeper7 | Pass | True | != |
| Creeper8!=Creeper4 | Fail | False | == |
| Creeper6>Creeper8 | Fail | False | < |
| Creeper2>Creeper6 | Fail | False | == |
| Creeper3>Creeper1 | Pass | True | > |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Status** | **Result** | **Comment** |
| Zombie1==Zombie5 | Pass | True | == |
| Zombie2==Zombie3 | Fail | False | != |
| Zombie4!=Zombie7 | Pass | True | != |
| Zombie8!=Zombie4 | Fail | False | == |
| Zombie6>Zombie8 | Fail | False | < |
| Zombie2>Zombie6 | Fail | False | == |
| Zombie3>Zombie1 | Pass | True | > |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Status** | **Result** | **Comment** |
| Creeper1==Zombie5 | Pass | True | == |
| Zombie2==Creeper3 | Fail | False | != |
| Zombie4!=Creeper7 | Pass | True | != |
| Creeper8!=Zombie4 | Fail | False | All fields == |
| Zombie6>Creeper8 | Fail | False | < |
| Creeper2>Zombie6 | Fail | False | == |
| Creeper3>Zombie1 | Pass | True | > |

# Object and Variable Functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Creeper Object | | Zombie Object | |
| **Members:** | creeper | **Members:** | zombie |
| **Data Type:** | integer | **Data**  **Type:** | integer |
| **Initial Value:** | 2 | **Initial**  **Value:** | 2 |
| **Purpose:** | To give a creeper for the creeper | **Purpose:** | To give a zombie for the zombie |

# Pseudo Code

## Assignment 5 Creeper Header File

* Will include the zombie class
* A class called creeper is created
* The creeper class will make a friend with the zombie class

**Data Members (Private):**

* A *creeper* with type integer

**Member Functions (Public):**

* A default **creeper** constructor that will initialize all the data members
* A overloaded **creeper** constructor that will initialize all the data members and take in one integer value
* A creeper destructor that will destroy the creeper objects
* An **operator** function that will overload the *==* operator using the address of a creeper object
* An **operator** function that will overload the *!=* operator using the address of a creeper object
* An **operator** function that will overload the *>* operator using the address of a creeper object
* An **operator** function that will overload the *==* operator using the address of a zombie object
* An **operator** function that will overload the *!=* operator using the address of a zombie object
* An **operator** function that will overload the *>* operator using the address of a zombie object
* A **set** function to set the *creeper* that will not return any value but take in a integer value
* A **get** function to get the *creeper* that will return an integer value but not take in any value

## Assignment 5 Zombie Header File

* Will include the creeper class
* A class called zombie is created
* The zombie class will make a friend with the creeper class

**Data Members (Private):**

* A *zombie* with type integer

**Member Functions (Public):**

* A default **zombie** constructor that will initialize all the data members
* A overloaded **zombie** constructor that will initialize all the data members and take in one integer value
* A zombie destructor that will destroy the creeper objects
* An **operator** function that will overload the *==* operator using the address of a zombie object
* An **operator** function that will overload the *!=* operator using the address of a zombie object
* An **operator** function that will overload the *>* operator using the address of a zombie object
* An **operator** function that will overload the *==* operator using the address of a creeper object
* An **operator** function that will overload the *!=* operator using the address of a creeper object
* An **operator** function that will overload the *>* operator using the address of a creeper object
* A **set** function to set the *zombie* that will not return any value but take in a integer value
* A **get** function to get the *zombie* that will return an integer value but not take in any value

## Assignment 5 Creeper Class File

* Includes the creeper class header file
* Includes the zombie class header file

Default **Creeper** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *creeper* is set with an integer value of 2

Overloaded **Creeper** constructor initializes the data members with a creeper parameter:

* A message will be displayed showing the overloaded constructor’s existence
* *creeper* is set with an integer value that came from the application file

**Creeper** destructor that will destroy the creeper objects:

* A message to show the destructors existence when called
* An **operator** function with a return type of boolean that will overload the *==* operator using the address of a creeper object
* Conditional statements that will check to see if the creeper is the same for both objects
* An **operator** function with a return type of boolean that will overload the *!=* operator using the address of a creeper object
* Conditional statements that will check to see if the creeper is not the same for the objects being compared
* An **operator** function with a return type of boolean that will overload the *>* operator using the address of a creeper object
* Conditional statements that will check to see if the creeper is greater than the creeper of another creeper object
* An **operator** function with a return type of boolean that will overload the *==* operator using the address of a zombie object
* Conditional statements that will check to see if the creeper is the same as the zombie object
* An **operator** function with a return type of boolean that will overload the *!=* operator using the address of a zombie object
* Conditional statements that will check to see if the creeper is not the same as the zombie object
* An **operator** function with a return type of boolean that will overload the *>* operator using the address of a zombie object
* Conditional statements that will check to see if the creeper object is greater than the zombie object
* The **set creeper** function is initialized that takes in a *input* creeper as a parameter
* The *input* creeper is given as the value for data member *creeper*
* The **get** **creeper** function is initialized that takes in no parameters
* It returns the *creeper* data member to the application file

## Assignment 5 Zombie Class File

* Includes the zombie class header file
* includes the creeper class header file

Default **Zombie** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *zombie* is set with an integer value of 2

Overloaded **Zombie** constructor initializes the data members with a zombie parameter:

* A message will be displayed showing the overloaded constructor’s existence
* *zombie* is set with an integer value that came from the application file

**Zombie** destructor that will destroy the creeper objects:

* A message to show the destructors existence when called
* An **operator** function with a return type of boolean that will overload the *==* operator using the address of a zombie object
* Conditional statements that will check to see if the zombie is the same for both objects
* An **operator** function with a return type of boolean that will overload the *!=* operator using the address of a zombie object
* Conditional statements that will check to see if the zombie is not the same for the objects being compared
* An **operator** function with a return type of boolean that will overload the *>* operator using the address of a zombie object
* Conditional statements that will check to see if the zombie is greater than the zombie of another zombie object
* An **operator** function with a return type of boolean that will overload the *==* operator using the address of a creeper object
* Conditional statements that will check to see if the zombie is the same as the creeper object
* An **operator** function with a return type of boolean that will overload the *!=* operator using the address of a creeper object
* Conditional statements that will check to see if the zombie is not the same as the creeper object
* An **operator** function with a return type of boolean that will overload the *>* operator using the address of a creeper object
* Conditional statements that will check to see if the zombie object is greater than the creeper object
* The **set zombie** function is initialized that takes in a *input* zombie as a parameter
* The *input* zombie is given as the value for data member *zombie*
* The **get** **creeper** function is initialized that takes in no parameters
* It returns the *zombie* data member to the application file

## Assignment 5 Application File

* Includes the creeper class header file
* Includes the zombie class header file
* A prototype function is created that will display the contents of the creeper object using the address of the creeper instance
* A prototype function is created that will display the contents of the zombie object using the address of the zombie instance
* The main function begins
* A message that welcomes the user to the program
* Eight creeper objects are made to test the constructors and the results are displayed to the screen
* Eight zombie objects are made to test the constructors and the results are displayed to the screen
* Tests described in the application summary in a table are executed for the eight objects for the creeper and zombie classes
* Mixed class tests for both classes created with an output of pass or fail to the screen
* A message is displayed that thanks the user for using the program
* The main function has ended
* The **display creeper** function is defined that takes in a **the address of a creeper object**as a parameter of type **creeper**
* **get creeper** is called and displays the creeper on the screen
* The **display zombie** function is defined that takes in a **the address of a zombie object**as a parameter of type **zombie**
* **get zombie** is called and displays the zombie on the screen

# Conclusions

This assignment was about demonstrating overloading of the C++ operators with single and mixed classes. What I thought went well was my original design plan and chosen operators for this assignment. I was able to overload the operators for the most part that I chose and was able to reuse my design in the previous assignments to save time. The issues I encountered were setting up the statements in the overloaded operator functions for mix class comparison. It wasn’t giving me the results I wanted for some things and I needed to delete some stuff to make it work right that took some time to fix. As far as things that didn’t work I was able to fix those mistakes so I didn’t have anything that was unresolved. I took me a little while to wrap my head around the concept of mix class operator overloading which stalled the design somewhat, but I was able to figure out the concepts that gave me confusion. There were no features that I had that could have been implemented better then they currently are to make the program better than it was. If there was a second version of this program the only thing I can think of to add to it would just be more data members to implement the object better and other overloaded operators to accompany them.

# Captured Screen Output

Welcome to assignment 5

Creeper default constructor

Creeper1: 2

Overloaded Creeper Constructor

Creeper2: 4

Overloaded Creeper Constructor

Creeper3: 6

Overloaded Creeper Constructor

Creeper4: 8

Overloaded Creeper Constructor

Creeper5: 2

Overloaded Creeper Constructor

Creeper6: 4

Overloaded Creeper Constructor

Creeper7: 6

Overloaded Creeper Constructor

Creeper8: 8

Default Zombie Constructor

Zombie1: 2

Overloaded zombie constructor

Zombie2: 4

Overloaded zombie constructor

Zombie3: 6

Overloaded zombie constructor

Zombie4: 8

Overloaded zombie constructor

Zombie5: 2

Overloaded zombie constructor

Zombie6: 4

Overloaded zombie constructor

Zombie7: 6

Overloaded zombie constructor

Zombie8: 8

Test 1 Passed

Test 2 Failed

Test 3 Passed

Test 4 Failed

Test 5 Failed

Test 6 Failed

Test 7 Passed

Test 8 Passed

Test 9 Failed

Test 10 Passed

Test 11 Failed

Test 12 Failed

Test 13 Failed

Test 14 Passed

Test 15 Passed

Test 16 Failed

Test 17 Passed

Test 18 Failed

Test 19 Failed

Test 20 Failed

Test 21 Passed

Thank you for using assignment 5

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default zombie destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Default creeper destructor

Press any key to continue . . .

# References

Bettle, H. (n.d.). Assignment 5 for CPTN230 Assignment Description. 7.

Bettle, H. (n.d.). Assignment 5 Sample Files. 6.

Deitel, P., & Deitel, H. (2010). *C++ How To Program.* Upper Saddle River: Pearson Education Inc.

Fahey, P. (2011). Assignment 4 Design Document. 10.