|  |
| --- |
| creeper1.jpg |
| Assignment 3 |
| CPTN230 |
|  |
| **Paul Fahey** |
| **10/6/2011** |



Contents

[Introduction 3](#_Toc305690454)

[Application Summary 3](#_Toc305690455)

[Object and Variable Functionality 3](#_Toc305690456)

[Pseudo Code 4](#_Toc305690457)

[Assignment 3 Header File 4](#_Toc305690458)

[Assignment 3 Class File 4](#_Toc305690459)

[Assignment 3 Application File 5](#_Toc305690460)

[Conclusions 8](#_Toc305690461)

[References 9](#_Toc305690462)

# Introduction

The goal of this assignment much like assignment 2 is to create a class, and demonstrate various techniques of accessing that class. An object will be created of our choosing that will demonstrate techniques such as multiple instances of objects dynamically, creating a pointer to the class object type, use a function to access objects with a pointer, use a pointer within an object, and become more familiar with the “this” pointer. Also, for this assignment the class will include the use of a static data member and a static member function. This assignment will also demonstrate the traditional way to make a program by making the design document first then code as the last step instead of reverse engineering already written code. (Bettle, Assignment 2 & 3 for CPTN230 Assignment Description)

# Application Summary

This program will demonstrate the functionality of a creeper object which came from the sandbox building game minecraft. The user will be greeted to the program, and be prompted to input a target, explosion, awareness and direction for the creeper object. Once each of those attributes have been accounted for they will be then printed to the screen with the proper instance of the creeper object. This program will also display multiple messages of creeper instances in existence or deleted to show the understanding of dynamic memory allocation. Finally, the user will receive a thank you message for using the program.

# Object and Variable Functionality

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Members of the Creeper Object** | | | | | |  |
| **Members:** | target | explosion | awareness | direction | targetptr | count of creepers |
| **Data Type:** | string | string | integer | integer | creeper | Static integer |
| **Initial Value:** | empty string | empty string | 0 | 0 | Address of target | 0 |
| **Purpose:** | To give a target for the creeper | To give an explosion for the creeper | To give an awareness for the creeper | To give a direction for the creeper | To point to the target of the target data member of the creeper | Tracks the number of creeper instances present |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Local Variables of the Main Function and instances of the Creeper Object** | | | | |
| **Local Variables:** | objective | blast | alertness | path |
| **Data Type:** | string | string | integer | integer |
| **Initial Value:** | empty string | empty string | 0 | 0 |
| **Creeper object Instances:** | Creeper 1 | Creeper 2 |  | |

# Pseudo Code

## Assignment 3 Header File

* A class called creeper is created

**Data Members (Private):**

* A *target* with type string
* An *explosion* with type string
* An *awareness* with type integer
* A *direction* with type integer
* A *pointer* to a target of type string
* A count of creepers with type static integer

**Member Functions (Public):**

* A **creeper** constructor that will initialize all the data members
* A **creeper** destructor that destroy the creeper class/object
* A **set** function to set the *target* that will not return any value but take in a string value
* A **get** function to get the *target* that will return an integer value but not take in any value
* A **set** function to set the *explosion* that will not return any value but take in a string value
* A **get** function to get the *explosion* that will return an integer value but not take in any value
* A **set** function to set the *awareness* that will not return any value but take in a string value
* A **get** function to get the *awareness* that will return an integer value but not take in any value
* A **set** function to set the *direction* that will not return any value but take in a string value
* A **get** function to get the *direction* that will return an integer value but not take in any value
* A **pointer** function that will get the address of the creeper object
* A **get** function that will get the *count of creepers* that will return a static integer value, but not take in any value

## Assignment 3 Class File

* Includes the creeper class header file
* *Count of creepers* is initialized to zero

**Creeper** constructor initializes the data members:

* A message saying a **creeper** object is being created
* *Target* is initialized with an empty string value
* *Explosion* is initialized with an empty string value
* *Awareness* is set with an integer value of zero
* *direction* initialized with an integer value of zero
* *Target pointer* is initialized with the address of *target*
* *Count of creepers* is constructed to increment up from zero when an instance is created

**Creeper** destructor that will remove the target(s) when called upon:

* A message that will say "Remove *target* from creeper" using the **get** target function
* *Count of creepers* is destructed to decrement down when an instance is removed
* The **set target** function is initialized that takes in a *input* target as a parameter
* The *input* target is given as the value for data member *target pointer*
* The **get** **target** function is initialized that takes in no parameters
* It returns the *target pointer* data member to the application file
* The **set explosion** function is initialized that takes in a *input* explosion as a parameter
* The *input* explosion is given as the value for data member *explosion*
* The **get** **explosion** function is initialized that takes in no parameters
* It returns the *explosion* data member to the application file
* The **set** **awareness** function is initialized that takes in a *input* awareness as a parameter
* The *input* awareness is given as the value for data member *awareness*
* The **get** **awareness** function is initialized that takes in no parameters
* It returns the *awareness* data member to the application file
* The **set** **direction** function is initialized that takes in a *input* direction as a parameter
* The *input* direction is given as the value for the *this pointer* of the **creeper** object to the data member *direction* to ensure is a data member from the creeper class
* The **get** **direction** function is initialized that takes in no parameters
* It returns the *this pointer* to the data member *direction* to the application file
* The **get** **address pointer** function is initialized that doesn't take in any parameters
* It returns the address of the *current instance* of the creeper class
* The **get count of creepers** function is initialized that takes in no parameters
* It returns the count of creepers at the current calling point

## Assignment 3 Application File

* Includes the creeper class header file
* A constant of five is defined for an instance limit
* A prototype function is created that will display the contents of the creeper object, and take in a pointer of type creeper as a parameter
* The main function begins
* A **current creeper pointer** to an object of type creeper is instantiated
* A **pointer to a pointer** for all the creepersto an object of type creeper is instantiated
* A message that welcomes the user to the program
* A message that shows the current number of creepers in existence using the get count of creeper function
* New memory is allocated to allow five values to be assigned to the **all creepers pointer**
* **all creepers** is put in a for loop to clear out any possible junk data by assigning the space allocated to be filled with zeros until it is used
* A message that shows the current number of creepers in existence using the get count of creeper function at this point
* A local variable of type string to the application file is created for *target* called *objective*
* A local variable of type string to the application file is created for *explosion* called *blast*
* A local variable of type integer to the application file is created for *awareness* called *alertness*
* A local variable of type integer to the application file is created for the *direction* called *path*
* A message that prompts for a target to be entered
* The input is assigned to the local variable of *objective*
* A message that prompts for an explosion to be entered
* The input is assigned to the local variable of *blast*
* A message that prompts for an awareness to be entered
* The input is assigned to the local variable of *alertness*
* A message that prompts for a direction to be entered
* The input is assigned to the local variable of *path*
* current creeper pointer is assigned a new creeper instance
* A message that shows the current number of creepers in existence using the get count of creeper function at this point
* The **current creeper pointer** points to the **set target** function that takes in the local variable of *objective*
* The **current creeper pointer** points to the **set explosion** function that takes in the local variable of *blast*
* The **current creeper pointer** points to the **set awareness** function that takes in the local variable of *alertness*
* The **current creeper pointer** points to the **set direction** function that takes in the local variable of *direction*
* The first index of the **all creepers pointer** is assigned to the **current creeper pointer**
* The **display the creeper** function is called that takes in the **current creeper pointer** as a parameter
* The **display the creeper** function is called again that takes in the **all creepers pointer** at its first index as a parameter
* current creeper pointer is assigned a second new creeper instance
* The **current creeper pointer** points to the **set target** function that takes in the local variable of *objective*
* The **current creeper pointer** points to the **set explosion** function that takes in the local variable of *blast*
* The **current creeper pointer** points to the **set awareness** function that takes in the local variable of *alertness*
* The **current creeper pointer** points to the **set direction** function that takes in the local variable of *direction*
* The second index of the **all creepers pointer** is assigned to the **current creeper pointer**
* The **display the creeper** function is called that takes in the **current creeper pointer** as a parameter
* The **display the creeper** function is called again that takes in the **all creepers pointer** at its first index as a parameter
* A message that shows the current number of creepers in existence using the **get count of creeper** function at this point
* Another message that shows the current number of creepers in existence, but with the **current creeper pointer** pointing to the **get count of creeper** function
* A third message that shows the current number of creepers in existence, but with the second index of the **all creepers pointer** pointing to the get count of creepers function
* The first index of space for **all creeper pointer** has been deleted
* A message that shows the current number of creepers in existence using the **get count of creeper** function at this point
* The second index of space for **all creeper pointer** has been deleted
* A message that shows the current number of creepers in existence using the **get count of creeper** function at this point
* The entire space allocated for all creepers pointer is deleted
* 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 creeper**pointer as a parameter of type **creeper**
* **the** **creeper** pointer points to the member function of **get target** which is called and displays the target on the screen
* **the creeper** pointer points to the member function of **get explosion** which is called and displays the explosion on the screen
* **the creeper** pointer points to the member function of **get awareness** which is called and displays the awareness on the screen
* **the creeper** pointer points to the member function of **get direction** which is called and displays the direction on the screen

# Conclusions

This assignment was about understanding dynamic memory allocation and the utilization of pointers, objects, static data members, etc. Some of the things that I felt went well was my original design plan. I was able to choose an object, develop a design for it, and copy and paste the design right into my program with minor changes. I believe it was the resources I had available to me such as the sample files, and assignment two that really made my design successful. However, there were a few things that gave me some difficulty completing this assignment. First thing I notice was that I had issues compiling when I implemented the destructor early on in the coding process. I found that making it the last thing to add in was the best approach to have the program compile correctly. Also the utilization of the "this" pointer was still unclear to me of what its main purpose is, but I was able to use my best judgment according to my resources to implement it right in my program. It also took me a little bit to understand memory allocation, the static data member added, and the static function added in relation to my object, constructor, and destructor to what it was doing in comparison to assignment 2 . There was nothing in the creating of my program that I never got to "work" the way I wanted it to. The things that didn't work I was able to resolve. Although it works fine, I feel there could have been a better way to organize my input code so it's less lengthy, and not using the method of just taking preset parameters in. If my program was to have an updated version, some things that could be added is more data members and members functions for the creeper object, and maybe a less lengthy way to organize the code in the application file with being able to keep its present features in some sort fashion.

# References

Bettle, H. (n.d.). Assignment 2 for CPTN230 Assignment Description. 8.

Bettle, H. (n.d.). Assignment 3 for CPTN230 Assignment Description. 6.

Bettle, H. (n.d.). Assignment 3 Sample Files. 3.

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

Fahey, P. (2011). Assignment 2 Design Document. 8.