Lab 7 – Debugging

Fork and clone the projects for Git Hub <https://github.com/Abertay-University-SDI/CMP105_W7> . This repository contains a collection of small programs that suffer from some compilation or runtime errors and need fixing.

1. Starting with application Week 7\_1, the application contains a player object that teleports to a random location when the *space bar* is pressed, and an enemy object that teleports to a random location after a set amount of time. However, the application does not compile successfully. You must find and fix the error, in the box below write down what the error was and how you found it (what debugging steps did you take, what information was useful).

|  |
| --- |
| Error: The error was a missing semi-colon after the last curly brace in the Enemy.h file.  Steps taken to solve: Simply tried to run it and looked at what the top error was in the Output window which told me “Enemy.cpp(6,1): error C2533: 'Enemy::{ctor}': constructors not allowed a return type”. This tells me that the first problem is in the Enemy.h file and to go looking there, upon further inspection of the IntelliSense error list window, this then told me exactly where to look in Enemy.h, and what the problem was. |

1. The application Week 7\_2, this application contains a player object that teleports to a random location when the *spacebar* is pressed, and the player contains a bullet object that spawns at the player’s location and fires when the *Enter* key is pressed. However, the application does not compile successfully. You must find and fix the error, in the box below write down what the error was and how you found it (what debugging steps did you take, what information was useful).

|  |
| --- |
| Error: The error was a missing colon after the private access modifier in the Player.h file.  Steps taken to solve: Simply tried to run it and looked at what the top error was in the Output window which told me “Player.h(18,2): error C2146: syntax error: missing ':' before identifier 'Bullet'”. This tells me that the first problem is in the Player.h file and to go looking there and also what the problem is, upon further inspection of the IntelliSense error list window, this then told me exactly where to look in Player.h, and what the problem was. |

1. Application Week 7\_3 is similar to 7\_2, however this application compiles, but suffers from a runtime error. You must find and fix the error; the fixed application should have a player that teleports and shoots several bullets. In the box below write down what the error was, how you found it (what debugging steps did you take, what information was useful) and how you would fix it.

|  |
| --- |
| Error: The error was a null pointer caused by trying to update a bullet object from the player class when no such bullet object even exists to update.  Steps taken to solve: I simply created a bullet object in the Player.h file and then assigned the Bullet pointer the address of the newly created bullet object. Job done, the update and setPosition functions now work as intended due to having a bullet object to act upon when trying to access it through pointers. |

1. Application Week 7\_4 contains a player and bullet objects. This application compiles but suffers from a runtime error. You must find and fix the error, in the box below write down what the error was, how you found it (what debugging steps did you take, what information was useful) and how you would fix it.

|  |
| --- |
| Error: The error was an access violation, namely we were trying to access a bullet object through a bullet pointer, which at first glance may appear to have been correctly assigned an address, however the bullet object itself was created in a spawn function and so when the scope of the function ended the bullet object was destroyed and so the bullet pointer was then left pointing to an object that no longer existed which was what we were then trying to access.  Steps taken to solve: Moved the creation of the newBullet object into the .h file and things work as intended, however only one bullet is able to be spawned, so I set hasFired to false whenever the Enter key is pressed in Player.h. This now works as 7\_3 did with unlimited bullets able to be spawned in. |

1. Application Week 7\_5 contains a player object and a companion object. The player object teleports around the level when *spacebar* is pressed. The companion will teleport to the player object when the *Enter* key is pressed. This application has a compilation error. You must find and fix the error (and make sure the application runs correctly), in the box below write down what the error was, how you found it (what debugging steps did you take, what information was useful) and how you would fix it.

|  |  |  |
| --- | --- | --- |
| Error: Circular dependencies, I would of thought pragma once would of helped with this issue but this full article helped me understand better, specifically this part:  The compiler will do the following:   |  |  | | --- | --- | | 1 2 3 4 5 6 7 8 9 10 11 12 | #include "a.h"  // start compiling a.h  #include "b.h"  // start compiling b.h  #include "a.h"  // compilation of a.h skipped because it's guarded  // resume compiling b.h  class B { A\* a }; // <--- ERROR, A is undeclared |   Ref: <http://www.cplusplus.com/articles/Gw6AC542/>  So, Player.h included Companion.h which included Player.h and so on. Bad news!  Steps taken to solve: Instead of including Player.h in the Companion.h file, which was causing the circular inclusion, I simply forward declared Player in Companion.h as we are only using the Player data type as a parameterized pointer type in the moveToPlayer function declaration, and so this is allowed. However in the Companion.cpp file we actually use the pointer type ‘p’ when we need to get access to its position, this means then that we need access to the full Player.h file and so it is here in Companion.cpp that we #include the Player.h file. Doing it this way prevents the infinite inclusion problem from happening. |

1. Application Week 7\_6 contains several *Bits*; small sprites that are given a random position and move up and down from the position. This application compiles but suffers from a runtime error. You must find and fix the error, in the box below write down what the error was, how you found it (what debugging steps did you take, what information was useful) and how you would fix it.

|  |
| --- |
| Error: Trying to access an element off the end of the bits array due to the update functions loop having a magic number 6 which will result in a count of 7, i.e. 0 through 6 and also the condition being set to <= rather than just <. There was also the same <= error in the render function loop.  Steps taken to solve: Created an int variable called bitsArrSize and set that equal to:  (sizeof(bits) / sizeof(bits[0]))  This way there is no need for magic numbers, set both of the loops in update and render to this new variable and also changed the condition to < rather than <=. |

1. Application Week 7\_7 should render a simple yellow game object near the centre of the window, but it doesn’t. You must find and fix the error, in the box below write down what the error was, how you found it (what debugging steps did you take, what information was useful) and how you would fix it.

|  |
| --- |
|  |

Make sure you show/discuss your answers with a member of staff once completed.

1. The coursework brief is on My Learning Space, under the Assessment tile. Read over the brief and write down a rough outline/design for your coursework game idea. Think carefully and write down how you game meets each of the coursework requirements.