Introduction to C++

|  |  |
| --- | --- |
| **Assessment Task Number:** Part 2 – Debugging Problems | |
| **Unit Code(s):** | **Unit Title(s):** |
| ICTPRG443 | Apply intermediate programming skills in different languages |
| ICTICT449 | Use version control systems in development environments |
| **Instructions to Learners:** | |

Download the student samples GitHub repository containing the project to debug from GitHub:

<https://github.com/AcademyOfInteractiveEntertainment/AIEYear1Samples>

You have received the following email from your supervisor:

Hi,

I have a problem with a bit of code one of the other interns has made.

The designers wanted a simple program to let them double-check the NPC data stored in the game’s data file.

In the *AIEYear1Samples* git repo you’ll find the project *IntroCPP\_BinaryFileIO\_NPCData*.

There are a few problems I have with this program:

* + - The designers tell me when they run this program on their real data files it takes way too long to load.   
      It seems the program reads the whole data file and stores the whole thing in memory – this is bad!  
      I need it to only load the current record. You’ll need to create a random-access algorithm to jump directly to the required record. DON’T sequentially read the file!
    - For some reason the NPC’s name isn’t being read in correctly. Can you fix this?
    - I get a crash when pressing the right arrow to go the next record while I’m on the last one (there are only 5 records in the sample file).
    - And finally, the most annoying thing is the whole program has no comments! Please add comments so the next intern knows what they’re doing.

When you’re done with your fixes, I want you to really *TEST* your work. Give me a brief report of how you tested your work – including what tests you ran and their results (I’m expecting at least 3 or 4 tests).

I also need to make sure you know what you’re doing in Visual Studio, so give me a few screenshots of you using the debugger. Show me:

* + - Some breakpoints you’ve placed in the code,
    - Some auto and custom watch variables displayed while debugging, and
    - The callstack while you’re debugging the program

Oh, and tell me the keyboard shortcuts for stepping through code in the debugger.

Do a good job and I’ll recommend you to the boss!

Cheers,  
Sam

Fix the problems described in the email above. Your final program must resolve all bugs, and use a random-access algorithm to read and load a single record at a time from the data file.

|  |  |  |
| --- | --- | --- |
| **Task** | | **Evidence Criteria** |
| 1. | Random Access File I/O | Update the given program to read a file using a random-access algorithm |
| 2. | Fix Name Display | Resolve the issue with the name displaying incorrectly |
| 3. | Program Crash | Resolve the issue with the program crashing when getting the next record |
| 4. | Comment Code | Add comments to the program.  Comments must be in line with industry standards or as defined by your trainer/assessor. |
| 5. | Testing Document | Document at least 3 test cases.  Documentation for each test case must list   * What is being tested * The input given * The expected output * The actual output * Any modifications made as a result of testing |
| 6. | Visual Studio Screenshots | A document containing screenshots of the following Visual Studio debugging features:   * Some breakpoints you’ve placed in the code, * Some auto and custom watch variables displayed while debugging, and * The callstack while you’re debugging the program |
| 7. | Debugging Shortcuts | A document containing a list of Visual Studio debugging shortcuts for navigating through code while using the debugger, including:   * Step Into * Step Over * Step Out |
| **Submission Requirements:** | | |
| You will need to submit the following:   * A Release build of each application that can execute as a stand-alone program * Your complete Visual Studio project   Be sure to remove any temporary build folders (i.e., the Debug and Release folders). Only project files, source code files, and any resource files used should be included in your submission.  Package all files in a single compressed archive file (.zip, .7z, or .rar) | | |

1. Random Access File I/O

Completed per the attached Visual Studio solution.

1. Fix Name Display

Completed per the attached Visual Studio solution.

1. Program Crash

Completed per the attached Visual Studio solution.

1. Comment Code

Completed per the attached Visual Studio solution.

1. Testing Document
   1. Test 1
      1. What is being tested

Retrieval of the previous record in the collection on left arrow press where the program is not already looking at the first record.

A computer screen shot of a program

Description automatically generated

* + 1. The input given

The left arrow button is pressed while the program is running, while the record being viewed is not the first NPC in the collection (any NPC other than “Tom”).

* + 1. The expected output

Where the current record being viewed is in position *r* within the collection, stop showing the particulars of the NPC in position *r* and instead show the particulars from the NPC record at position *r – 1*. This will be reviewed with breakpoints to identify which code sections are being triggered by their logical expressions.

* + 1. The actual output

The test starts on NPC “Grutt”, record 2 (index 1) of the collection of NPCs in the binary file. All of the code inside the first nested *if* statement is true - we will press the left arrow key and the current record is not the first one in the collection.

A computer screen shot of a cartoon character

Description automatically generated

The first breakpoint is triggered in the callstack as expected and we can see that the particulars of the previous record are now loaded instead (“Tom” has been loaded instead of “Grutt”).

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

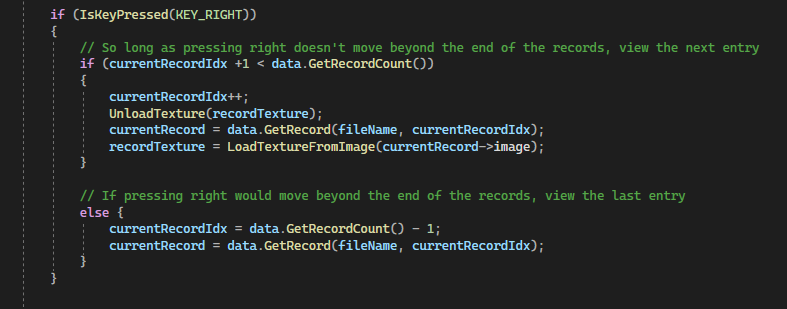
Description automatically generated

* + 1. Any modifications made as a result of testing

Not required.

* 1. Test 2
     1. What is being tested

Retrieval of the final record in the collection on right arrow press where the program is already viewing the final record.



* + 1. The input given

The right arrow button is pressed while the program is running, while the record already being viewed is the last NPC in the collection, “Judy” (record 5, index 4).

* + 1. The expected output

When the right arrow key is pressed, the program should evaluate the first nested *if* statement as false, and the *else* statement as true. The program should continue to view the “Judy” NPC record and make no changes.

This will be reviewed with watch variables on both the **current record number**, and the **current record** object so as to view its member variables (principally the name and age of the NPC).

* + 1. The actual output

After navigating to the final record, “Judy”, but before pressing the right arrow key again:

A screenshot of a computer

Description automatically generated

The NPC record indices are 0-4, which has “Judy”, the 5th record, in 4th. The right arrow key is pressed, the first breakpoint expression should evaluate to *false*, and we should move to the second breakpoint.A screenshot of a computer program

Description automatically generated

... which it does, with each of the member variables of the current record object remaining unchanged in the watch window.

A screenshot of a computer program

Description automatically generated

* + 1. Any modifications made as a result of testing

Row 92 and 93, in which the same record is re-loaded if the code is already viewing the last record, are not necessary, so were removed – it is more efficient to load nothing at all.

* 1. Test 3
     1. What is being tested

Main game loop logic (*if* & *else* logic statements for both left and right arrow keypresses) excluding moving beyond the last record (tested previously) and moving from record 2 to record 1 (tested previously).

A screenshot of a computer program

Description automatically generated

* + 1. The input given

The program will start at record 1 (index 0) and will be given three keystrokes in this order:

* 1 left arrow keystroke (move from record 1 to 0 [no change]).
* 1 right arrow keystroke (move from record 1 to 2).

The NPC being viewed will start with “Tom”.

* + 1. The expected output

The 2nd breakpoint should trigger on the first keystroke, then the 3rd breakpoint should trigger on the second keystroke, and neither the 1st nor 4th should trigger at all. The NPC being viewed will end with “Tom”.

* + 1. The actual output

A screenshot of a computer program

Description automatically generated

Breakpoint 2, line 75, is correctly executed as a result of the first keystroke because the record ID is 0 (we started on the first index of the collection).

A screenshot of a computer program

Description automatically generated

Breakpoint 3, line 85, is correctly executed as a result of the second keystroke because the record ID of the next NPC is not 5 (we were not at the last index of the collection).

After these two keystrokes, pressing ‘continue’ in debug mode did not trigger either of the other two breakpoints, and the NPC being viewed is correct.

A screenshot of a computer

Description automatically generated

* + 1. Any modifications made as a result of testing

Not required.

1. Visual Studio Screenshots

Screenshots showing each of the requirements have been placed throughout section 5.

1. Debugging Shortcuts
   1. Step Into (F11)
   2. Step Over (F10)
   3. Step Out (Shift + F11)