CIS 200 - Lab 0302

1. Problem Statement

Create a program that will open a given file and print the contained values. If there is not a file or the file is empty then display an error.

2. Requirements

2.1 Assumptions

- Written in C++
- Compiled exclusively in Visual Studio
- File “*in.dat*” will be provided for testing
- File will contain integers

2.2 Specifications

- Array Size is 20
- Two External Functions
    - Reading File to Array
    - Printing Array to Console
- Assertion when array is empty

3. Decomposition Diagram

- Program
    - Input
        - Read in file [*in.dat*]
    - Process
        - Fill array with integers from file
    - Output
        - Print filled array in order

4. Test Strategy

- Valid Data
- Invalid Data

5. Test Plan Version 1

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	File Exists				
Valid Data	2	File contains 20 good integers *readIntFile*() *printFileValues*()				
Valid Data	3	File contains < 20 good integers *readIntFile*() *printFileValues*()				
Invalid Data	1	File Does Not Exist				

6. Initial Algorithm

1. Include Compiler Directives
    - a. “assert.h”
    - b. <iostream>
    - c. <fstream>
    - d. <string>
2. Create External Functions
    - *a.* Create Function
        - i. Name: *readIntFile*()
        - ii. Parameters: *ifstream* by reference, integer array *intArray*[], int *size*, int *length* by reference
        - iii. Return: void
        - iv. Method:
            1. Declare counter variable *i* = 0
            2. Until end of file
                - a. Fill array at index *i*
                - b. Iterate *i*
            3. Pass back *length* as *i*
    - *b.* Create Function
        - i. Name: *printFileValues*()
        - ii. Parameters: integer array *intArray*[], integer *length* by reference
        - iii. Return: Void
        - iv. Method:
            1. Assert that array has any values
            2. Print array length
            3. Loop until length reached

```
            a. Print each array index formatted
3. Main Method main()
    a. Prompt user for file name [in.dat]
    b. Open file with given file name
    c. If file is open
        i. Call readIntFile()
        ii. Call printFileValues()
    d. Else
        i. Alert user that file does not exist
```

7. Test Plan Version 2

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	File Exists	"*in.dat*"	Array Filled		
Valid Data	2	File contains 20 good integers *readIntFile*() *printFileValues*()	"*in.dat*" 7 8 9 10 11 -1 -3 0 15 88 99 200 15 99 8888 -2 0 -99 -300	Array Filled *intArray*[0] = 7 *intArray*[1] = 8 *intArray*[2] = 9 *intArray*[3] = 10 *intArray*[4] = 11 *intArray*[5] = -1 *intArray*[6] = -3 *intArray*[7] = 0 *intArray*[8] = 15 *intArray*[9] = 88 *intArray*[10] = 99 *intArray*[11] = 200 *intArray*[12] = 15 *intArray*[13] = 99 *intArray*[14] = 8888 *intArray*[15] = -2 *intArray*[16] = 0 *intArray*[18] = -99 *intArray*[19] = -300		
Valid Data	3	File contains < 20 good integers *readIntFile*() *printFileValues*()	"*in2.dat*" 404 6502 666 8088 7 42	*intArray*[0] = 404 *intArray*[1] = 6502 *intArray*[2] = 666 *intArray*[3] = 8088 *intArray*[4] = 7 *intArray*[5] = 42		

Invalid Data	1	File Does Not Exist	*"in.txt"*	FILE DNE		

8. Code

```
//Program Name: Square Root Calculator
//Programmer Name: Arthur Aigeltinger IV
//Description: Use series approximation to calculate the square root.
//Date Created: 10/03/18

#include "assert.h"
#include <iostream>
#include <fstream>
#include <string>

//Function Prototypes
void readIntFile(std::ifstream&, int intArray[], int &);
void printFileValues(int intArray[], int& length);

int main()
{
	//Declare Variables
	const int SIZE = 20;
	int length = 0;
	int intArray[SIZE];
	std::ifstream input;
	std::string fileName = "";

	//Prompt User
	std::cout << "Please enter the name of the file containing integers" <<
std::endl << "Default is [in.dat]: ";
	std::cin >> fileName;

	//Convert filename string to actual file
	input.open(fileName);

	//Check if file exists
	if (input.is_open())
	{
		readIntFile(input, intArray, length);
		printFileValues(intArray, length);
	}
	else
	{
		std::cout << "Input file does not exist!" << std::endl;
	}

	system("pause");
	return 0;
}

//Description: Take in a user specified file and fill an array passed by reference.
//Pre-Condition: File being open, array existing.
//Post-Condition: Filled array!
void readIntFile(std::ifstream &input, int intArray[], int &length)
{
	int i = 0;

	while (!input.eof())
	{
		input >> intArray[i];
		i++;
	}
	length = i;
}

//Description: Prints the values in order of a given array from a file.
//Pre-Condition: Array being filled
//Post-Condition: User will be satisfied.
void printFileValues(int intArray[], int & length)
{
	assert(intArray[0] != -858993460);

	std::cout << "Final length of array is " << length << std::endl;

	for (int i = 0; i < length; i++)
	{
		//Formatting
		std::cout << "Integer #" << i + 1 << ": " << intArray[i] << std::endl;
	}
}
```

9. Updated Algorithm

1. Include Compiler Directives
    a. “assert.h”
    b. <iostream>
    c. <fstream>
    d. <string>
2. Create External Functions
    a. Create Function
        i. Name: *readIntFile*()

```
            ii.   Parameters: ifstream by reference, integer array intArray[], ~~int size~~, int
                  length by reference
            iii.  Return: void
            iv.   Method:
                  1. Declare counter variable i = 0
                  2. Until end of file
                     a. Fill array at index i
                     b. Iterate i
                  3. Pass back length as i
      b. Create Function
            i.    Name: printFileValues()
            ii.   Parameters: integer array intArray[], integer length by reference
            iii.  Return: Void
            iv.   Method:
                  1. Assert that array has any values
                  2. Print array length
                  3. Loop until length reached
                     a. Print each array index formatted
3. Main Method main()
      a. Declarations
            i.    Constant int SIZE as 20
            ii.   Integer length as 0
            iii.  Integer Array intArray[SIZE]
            iv.   Input file stream
            v.    String to handle input file stream name
      b. Prompt user for file name [in.dat]
      c. Open file with given file name
      d. If file is open
            i.    Call readIntFile()
            ii.   Call printFileValues()
      e. Else
            i.    Alert user that file does not exist
```

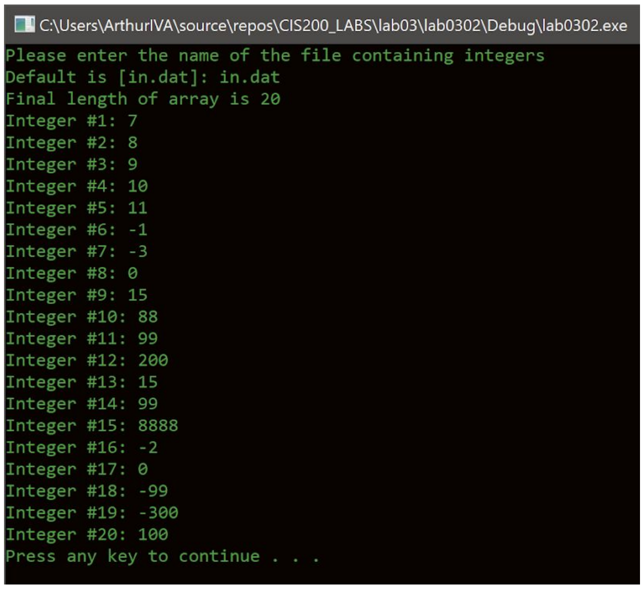
10. Test Plan Version 3

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail

Valid Data	1	File Exists	“*in.dat*”	Array Filled	Array Filled	Pass
Valid Data	2	File contains 20 good integers *readIntFile*() *printFileValues*()	“*in.dat*” 7 8 9 10 11 -1 -3 0 15 88 99 200 15 99 8888 -2 0 -99 -300	Array Filled *intArray*[0] = 7 *intArray*[1] = 8 *intArray*[2] = 9 *intArray*[3] = 10 *intArray*[4] = 11 *intArray*[5] = -1 *intArray*[6] = -3 *intArray*[7] = 0 *intArray*[8] = 15 *intArray*[9] = 88 *intArray*[10] = 99 *intArray*[11] = 200 *intArray*[12] = 15 *intArray*[13] = 99 *intArray*[14] = 8888 *intArray*[15] = -2 *intArray*[16] = 0 *intArray*[18] = -99 *intArray*[19] = -300	Array Filled & Print *Final length of array is 20* *Integer #1: 7* *Integer #2: 8* *Integer #3: 9* *Integer #4: 10* *Integer #5: 11* *Integer #6: -1* *Integer #7: -3* *Integer #8: 0* *Integer #9: 15* *Integer #10: 88* *Integer #11: 99* *Integer #12: 200* *Integer #13: 15* *Integer #14: 99* *Integer #15: 8888* *Integer #16: -2* *Integer #17: 0* *Integer #18: -99* *Integer #19: -300* *Integer #20: 100*	Pass
Valid Data	3	File contains < 20 good integers *readIntFile*() *printFileValues*()	“*in2.dat*” 404 6502 666 8088 7 42	*intArray*[0] = 404 *intArray*[1] = 6502 *intArray*[2] = 666 *intArray*[3] = 8088 *intArray*[4] = 7 *intArray*[5] = 42	Array Filled & Print *Final length of array is 6* *Integer #1: 404* *Integer #2: 6502* *Integer #3: 666* *Integer #4: 8088* *Integer #5: 7* *Integer #6: 42*	Pass
Valid Data	4	File contains 1 good integer *readIntFile*() *printFileValues*()	“*uno.dat*” 200	*intArray*[0] = 200	Array Filled & Print *Final length of array is 1* *Integer #1: 200*	Pass
Invalid Data	1	No File	“*in.txt*’	FILE DNE	FILE DNE	Pass
Invalid Data	2	File is Empty	“*emp.dat*”	*assert*()	*assert*()	Pass

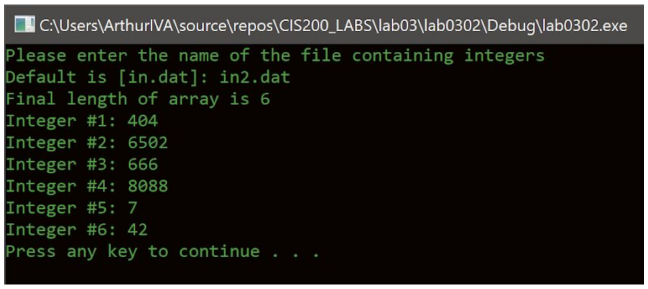
11. Screenshots

Valid Test Cases 1 and 2



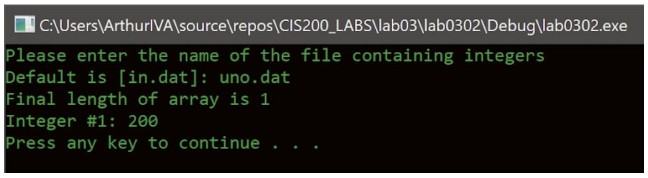
```
C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe
Please enter the name of the file containing integers
Default is [in.dat]: in.dat
Final length of array is 20
Integer #1: 7
Integer #2: 8
Integer #3: 9
Integer #4: 10
Integer #5: 11
Integer #6: -1
Integer #7: -3
Integer #8: 0
Integer #9: 15
Integer #10: 88
Integer #11: 99
Integer #12: 200
Integer #13: 15
Integer #14: 99
Integer #15: 8888
Integer #16: -2
Integer #17: 0
Integer #18: -99
Integer #19: -300
Integer #20: 100
Press any key to continue . . .
```

Valid Test Case 3



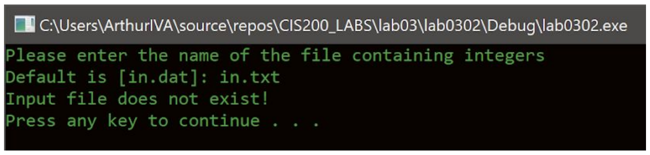
```
C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe
Please enter the name of the file containing integers
Default is [in.dat]: in2.dat
Final length of array is 6
Integer #1: 404
Integer #2: 6502
Integer #3: 666
Integer #4: 8088
Integer #5: 7
Integer #6: 42
Press any key to continue . . .
```

Valid Test Case 4



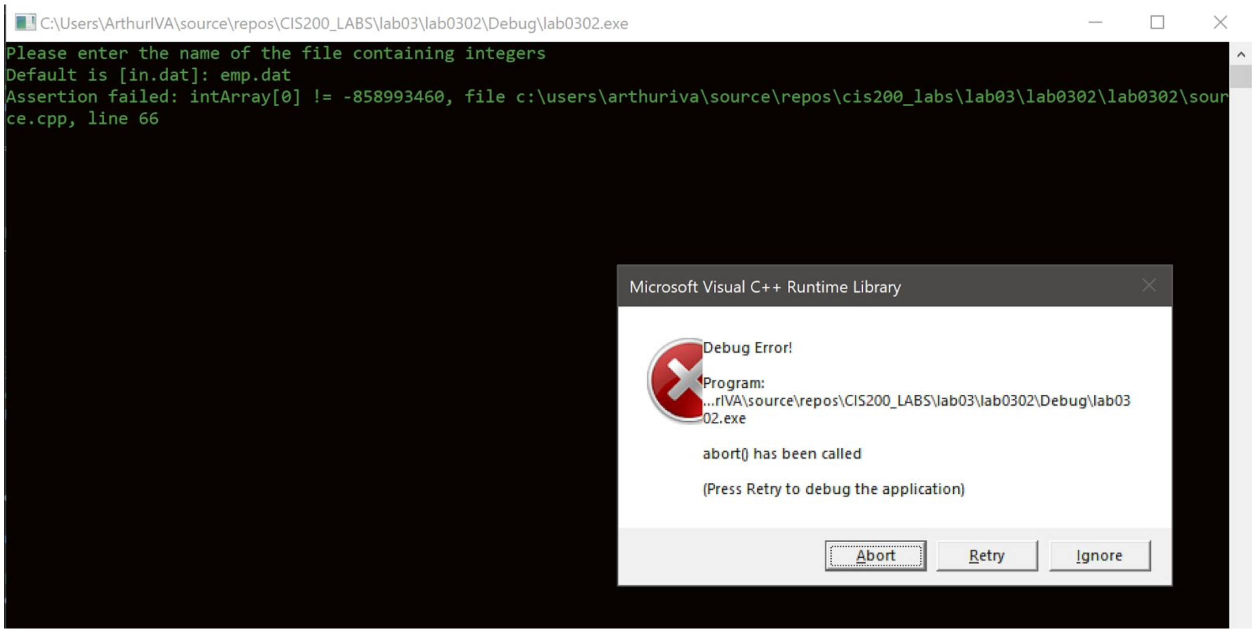
```
C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe
Please enter the name of the file containing integers
Default is [in.dat]: uno.dat
Final length of array is 1
Integer #1: 200
Press any key to continue . . .
```

Invalid Test Case 1



```
C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe
Please enter the name of the file containing integers
Default is [in.dat]: in.txt
Input file does not exist!
Press any key to continue . . .
```

Invalid Test Case 2


C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe

```
Please enter the name of the file containing integers
Default is [in.dat]: emp.dat
Assertion failed: intArray[0] != -858993460, file c:\users\arthuriva\source\repos\cis200_labs\lab03\lab0302\lab0302\source.cpp, line 66
```

Microsoft Visual C++ Runtime Library

Debug Error!

Program:
...rIVA\source\repos\CIS200_LABS\lab03\lab0302\Debug\lab0302.exe

abort() has been called

(Press Retry to debug the application)

12. Error Log

Error Type (Logic/Runtime)	Cause of Error	Solution to Error
Logic	Original *assert*() looked for *length* > 0, but given loop always gives at least *length* of 1.	Modify *assert*() to look for value of -858993460 in index 0 since that indicates nothing was loaded from file.

13. Status

The program works fully in its current form. With time allotted I would look for a more streamlined/consistent solution to check if the file was empty in the current *assert*() command.