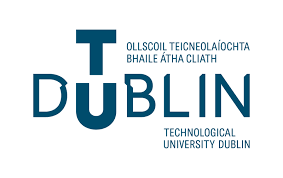
******

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**BEng Tech (Ord) Electronics and Communications Engineering Level 7**

**Programme Code: (DT008)**

**<YEAR 3>**

**Name of Module (Software Design 2)**

**Name of Lecturer (Paul J. Kiernan)**

***Assignment 1, 2019/2020***

**Student Name** Sajjad Ullah, C17344483**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Address Group** B\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Assignment title** Seismic Event Detection**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date returned \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table of Contents

[1.0 Introduction 4](#_Toc24808887)

[1.1 Procedure 4](#_Toc24808888)

[2.0 Solution design 4](#_Toc24808889)

[2.1.1 Attribute table 4](#_Toc24808890)

[3.0 Behaviour 5](#_Toc24808891)

[3.1 Behaviour Table 5](#_Toc24808892)

[4.0 class seismic diagram 8](#_Toc24808893)

[4.1.1 Attributes 9](#_Toc24808894)

[5.0 Test Plan Design 9](#_Toc24808895)

[5.1 Test plan for seismic 9](#_Toc24808896)

[6.0 RESULTS 12](#_Toc24808897)

[6.1 Test 1 Empty Class seismic 12](#_Toc24808898)

[6.2 Test 2 seismic //Constructor 13](#_Toc24808899)

[6.3 Test 3 seismic //Regular Constructor 14](#_Toc24808900)

[6.4 Test 4 setDataWindowST() 15](#_Toc24808901)

[6.5 Test 5 setDataWindowLT() 16](#_Toc24808902)

[6.6 Test 6 setTresholdVlaue() 17](#_Toc24808903)

[6.7 Test 7 getDataWindowST() 18](#_Toc24808904)

[6.8 Test 8 getDataWindowLT() 19](#_Toc24808905)

[6.9 Test 9 getTresholdVlaue() 20](#_Toc24808906)

[6.10 Test 10 askDataWindows() 21](#_Toc24808907)

[6.11 Test 11 askTresholdVlaue() 22](#_Toc24808908)

[6.12 Test 12 cal\_ST\_Power() 23](#_Toc24808909)

[6.13 Test 13 cal\_LT\_Power() 24](#_Toc24808910)

[6.14 Test 14 displayResults() 27](#_Toc24808911)

[7.0 Evaluation 29](#_Toc24808912)

[8.0 Conclusion 31](#_Toc24808913)

**Table of Figures**

[Figure 1 Result for empty class 13](#_Toc24808873)

[Figure 2 Result from Default Constructor 14](#_Toc24808874)

[Figure 3 Result from regular Constructor 15](#_Toc24808875)

[Figure 4 Result from setDataWindowST() 16](#_Toc24808876)

[Figure 5 Result from setDataWindowLT() 16](#_Toc24808877)

[Figure 6 Result from setTresholdVlaue() 18](#_Toc24808878)

[Figure 7 Result from getDataWindowST() 19](#_Toc24808879)

[Figure 8 Result from getDataWindowLT() 20](#_Toc24808880)

[Figure 9 Result from getTresholdVlaue() 20](#_Toc24808881)

[Figure 10 Result from askDataWindows() 21](#_Toc24808882)

[Figure 11 Result from askTresholdVlaue() 22](#_Toc24808883)

[Figure 12 Result from cal\_ST\_Power() 24](#_Toc24808884)

[Figure 13 Result from cal\_LT\_Power() 26](#_Toc24808885)

[Figure 14 Result from displayResults() 29](#_Toc24808886)

# Introduction

Assignment Description:

In this report we will be designing and creating code for a seismic event detection code in c++.

We will code the programming Microsoft visual studio 2010.

We will be creating attributes and making function identifiers for the solution. We will fill out a behaviour table to have a description of the functions used. We will look at what a function does, what its type can be and what are its attributes. We plan to individually test each new function before adding new code. In the program we will be entering float & int values from the monitor screen with the *cin* statement code.

**Description:**

We are to create a C++ Object Oriented program that reads seismometer data from a data (text) file named seismic.dat, determines whether seismic events have occurred, and reports the data and findings to the screen in text and graphics format and stores the findings to a data file.

Were told that the first line of the seismic data file contains two values: the number of data elements or readings and the time interval in seconds that occurred between consecutive measurements. .The time interval is a floating point value and it may be assumed that all the measurements were taken with the same time interval between them. After reading the data measurements the program should identify possible earthquakes or seismic events using a power ratio.

The **short-time power** is the average power, or average squared value, of the measurements using the specified point plus a small number of points that occurred just previous to the specified point.

The **long-time power** is the average power of the measurements using the specified point plus a larger number of points that occurred just previous to the specified point.

The data window length (number of measurements) for short-time power and long-time power are to be read from the keyboard as is the threshold value.

**Problem Statement:**

Determine the locations of possible seismic events using a set of seismometer measurements from a data file. Write the times of possible events to a data file with the corresponding power ratio and threshold. The times of possible events and the corresponding power ratio and threshold should be reported to the PC screen in text and graphic mode. Demonstrate an Iterative approach in your algorithm and program development.

## 1.1 Procedure

Open Microsoft Visual Studio 210.

Create a new project using the console application option.

Follow the code and steps that are in this report.

# 2.0 Solution design

We start by braking the problem into a set by step solution. Such as creating default, regular and deconstrustors while also incorporating ask…() ,set…() and get…() functions in our design. When we do this for any of the required attributes, we will have all the functions and type we will use laid out in this report filled in the behaviour table.

We plan to create a default constructor so that we can set values to our data.

We also plan to use an attribute table and behaviour table for use in the programming stage later. When we have these two tables filled out we will have a clear understanding of what our function will be and what will they do.

We will record each test of the functions so that we have a record of our progress and have a written copy of any obstacles faced during the programming stage.

## 2.1.1 Attribute table

This attribute table shows the attributes we will use and their type along with a short description of what they are.

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Type an Identifier |
| DataWindowST | This attribute represents the short-term data window. | int DataWindowST; |
| DataWindowLT | This attribute represents the long-term data window. | int DataWindowLT; |
| TresholdVlaue | This attribute represents the threshold value we will be using. | double TresholdVlaue; |

# 3.0 Behaviour

The behaviour table is a good way to show what attributes we will use, what their identifiers will be and their parameters along with a short description of what the function does.

The set….() is used as a function to set a value.

The get….() is used as a function to return the value stored to main.

The ask…() will be used as a function to ask input values from the user.

The calculate…() will be used as a function to do calculation.

The display…() will be used as a function to display any result with their values.

## 3.1 Behaviour Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes  Used | Function  Identifier | Description of Function | Type, Identifier,  Parameters |
| DataWindowST  DataWindowLT  TresholdVlaue | seismic();  //default | This function will be the default constructor.  Does this function take in any parameters?  No  Does this function return any value?  No. | seismic()  seismic(void) |
| DataWindowST  DataWindowLT  TresholdVlaue | seismic(…);  //regular | This function will be the regular constructor.  Does this function take in any parameters?  Yes, int & float.  Does this function return any value?  No, void | seismic(int aST, int aLT, float aTreshold); |
| DataWindowST | setDataWindowST(...); | This function will set the data value for short-term window.  Does this function take in any parameters?  Yes,int.  Does this function return any value?  No, void. | setDataWindowST();  setDataWindowST(int aSTvalue);  void setDataWindowST(int aSTvalue); |
| DataWindowLT | setDataWindowLT(…); | This function will set the data value for long-term window.  Does this function take in any parameters?  Yes,int.  Does this function return any value?  No, void. | setDataWindowLT();  setDataWindowLT(int aLTvalue);  void setDataWindowLT(int aLTvalue); |
| TresholdVlaue | setTresholdVlaue(…); | This function will set a threshold value.  Does this function take in any parameters?  Yes, double.  Does this function return any value?  No, void. | setTresholdVlaue();  setTresholdVlaue(double aTreshold);  void setTresholdVlaue(double aTreshold); |
| DataWindowST; | getDataWindowST() | This function will ask a value for the attribute.  Does this function take in any parameters?  No, void  Does this function return any value?  Yes, int. | getDataWindowST();  getDataWindowST(void);  int getDataWindowST(void); |
| DataWindowLT | getDataWindowLT() | This function will display  All the attributes  Does this function take in any parameters?  No, void  Does this function return any value?  Yes, int. | getDataWindowLT();  getDataWindowLT(void);  int getDataWindowLT(void); |
| TresholdVlaue | getTresholdVlaue() | This function will get the threshold value.  Does this function take in any parameters?  No, void  Does this function return any value?  Yes,float. | getTresholdVlaue();  getTresholdVlaue(void);  float getTresholdVlaue(void); |
| DataWindowST  DataWindowLT | askDataWindows() | This function will ask a value for  The short & long-term window.  Does this function take in any parameters?  No, void  Does this function return any value?  No, void | askDataWindows();  askDataWindows(void);  void askDataWindows (void); |
| TresholdVlaue | askTresholdVlaue() | This function will ask a value for  The threshold.  Does this function take in any parameters?  No, void  Does this function return any value?  No, void | askTresholdVlaue();  askTresholdVlaue(void);  void askTresholdVlaue(void); |
| DataWindowST | cal\_ST\_Power() | This function will calculate the short-term window.  Does this function take in any parameters?  Yes, float Array & int.  Does this function return any value?  Yes,float | cal\_ST\_Power( );  cal\_ST\_Power(float Array[], int num);  float cal\_ST\_Power(float Array[], int num); |
| DataWindowLT | cal\_LT\_Power() | This function will calculate the long-term window.  Does this function take in any parameters?  Yes, float Array & int.  Does this function return any value?  Yes,float | cal\_LT\_Power( );  cal\_ LT \_Power(float Array[], int num);  float cal\_ LT \_Power(float Array[], int num); |
| DataWindowLT  TresholdVlaue | displayResults(…) | This function will display the result to the user.  Does this function take in any parameters?  Yes, float & arrays  Does this function return any value?  No,void | displayResults();  displayResults(float NumElements, float TimeInterval,float tempArrayST[], float tempArrayLT[] );  void displayResults(float NumElements, float TimeInterval,float tempArrayST[], float tempArrayLT[] ); |

# 4.0 UML diagram

It contains a diagram with all the variables relating to the solution of the problem.

It has all the attributes and behaviours we will use in the solution.

The public code in it is available for the user and can be easily changed.

The private section stores data that we do not want to be changed easily.

|  |
| --- |
| seismic |
| **public:**  seismic();  seismic(int aST, int aLT, float aTreshold);  void askDataWindows();  void askTresholdVlaue();  void displayResults(float NumElements, float TimeInterval,float tempArrayST[], float tempArrayLT[] );  float cal\_ST\_Power(float Array[], int num);  float cal\_LT\_Power(float Array[], int num);  int getDataWindowST();  int getDataWindowLT();  float getTresholdVlaue();  void setDataWindowST(int aSTvalue);  void setDataWindowLT(int aLTvalue);  void setTresholdVlaue(float aTreshold); |
| **private:**  int DataWindowST;  int DataWindowLT;  double TresholdVlaue; |

|  |
| --- |
| graphics |
| **point** |
| **line** |

|  |
| --- |
| resource |

### Attributes

We have created a table above so that we have all the attributes listed, that will be used when we program the solution in Microsoft visual studio.

# 5.0 Test Plan Design

It is necessary to know what the outcome of the program code will give before starting coding. If the outcome is not known before than how does one know if the code works. This is why we will test each new part of code that is added to ensure it functions as intended before adding the next line of code. Thus, we will slowly build our program one line at a time instead of creating the whole program at once and likely getting debug/runtime errors, which would be laborious to fix.

## 5.1 Test plan for seismic

This is the test plan for the seismic class.

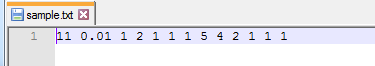
The first requirement is to make sure the files are set up in the project correctly. So a simple empty test will be performed first.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Function under  test | Description | Expected outcome |
| 1 | Empty class | This test checks that the files and project folder are set up correctly. | Main should run and display  “Main has started”  “Main is Ending” |
| 2 | seismic();  //default | This test will check that the default constructor initialises the attributes.    We add debug code to view what has been stored in the attributes.  To use the class an object needs to be created in main()  //The object can be :  seismic E1; | Debug from default constructor  The Short Term Data Window = 0  The Long Term Data Window = 0  The Threshold value is = 0 |
| 3 | seismic(…);  //regular | This test will check that the regular constructor works.  So, put in the say the values in main:  //To use the function  seismic Event(2,4,1.5); | Debug from Regular constructor  The Short Term Data Window = 2  The Long Term Data Window = 4  The Threshold value is = 1.5 |
| 4 | setDataWindowST(); | This test will set a short-term data window in main.  //To use the function  Event.setDataWindowST(3); | Debug from setDataWindowST(), DataWindowST = 3 |
| 5 | setDataWindowLT(); | This test will set a long-term data window in main.  //To use the function  Event.setDataWindowLT(6); | Debug from setDataWindowLT(), DataWindowLT = 6 |
| 6 | setTresholdVlaue(); | This test will set a threshold value in main.  //To use the function  Event.setTresholdVlaue(1.8); | Debug from setTresholdVlaue(), TresholdVlaue = 1.8 |
| 7 | getDataWindowST() | This test will check that the attribute value is returned correctly to the program main().  We add debug code to view what has been stored in the attributes.  //To use the function  cout<<"\nDebug from getDataWindowST(), ST data window = "<<Event.getDataWindowST()<<endl; | Debug from getDataWindowST(), ST data window = 3 |
| 8 | getDataWindowLT() | This test will check that the attribute value is returned correctly to the program main().  Add debug code to function to display attribute value.  //To use the function  cout<<"Debug from getDataWindowLT(), LT data window = "<<Event.getDataWindowLT()<<endl; | Debug from getDataWindowLT(), LT data window = 6 |
| 9 | getTresholdVlaue() | This test will check that the attribute value is returned correctly to the program main().  Add debug code to function to display attribute value.  //To use the function  cout<<"Debug from getTresholdVlaue(), Treshold = "<<Event.getTresholdVlaue()<<endl; | Debug from getTresholdVlaue(), Treshold = 1.8 |
| 10 | askDataWindows() | This test will check that the user can enter the attribute values.  //To use the function  void askDataWindows(); | Enter The Data window value you would like to use for Short-Term power =>  Enter the Data window value you would like to use for Long-Term power => |
| 11 | askTresholdVlaue() | This test will check that the user can enter the attribute values.  //To use the function  void askTresholdVlaue(); | Enter the Threshold value you would like to use => |
| 12 | cal\_ST\_Power() | This test will calculate the short-term values and save to an array called tempArrayST.  //To use the function  //this code is in a for loop.  tempArrayST[f] = Event.cal\_ST\_Power(array, s=offset+s ); | Element – value  Temp array ST is => 0 – 1  Temp array ST is => 1 – 13  …… |
| 13 | cal\_LT\_Power() | This test will calculate the long-term values and save to an array called tempArrayLT.  //To use the function  tempArrayLT[g] = Event.cal\_LT\_Power(array, s ); | Element – value  Temp array LT is => 0 - 1.6  Temp array LT is => 1 - 6.4  ……… |
| 14 | displayResults(…) | This test will display the result to the user.  //To use the function  Event.displayResults(TotalValues,  TimeInterval,tempArrayST,  tempArrayLT); | The point at 0.04  The Short-Term power = 1  The Long-Term power = 1.6  The Ratio = 1/1.6 = 0.6  The Threshold is = 1.5  Result - No Event at time 0.04  ………  ……… |

# 6.0 RESULTS

This section shows the results from the test plan.

We will be using the values from a sample file called “sample.txt” to test the program.



## 6.1 Test 1 Empty Class seismic

This test will check the empty class.

class seismic

{

public:

Code that will be added in public will go here.

private:

Code that will be added in private will go here.

};

Functions will go here.

We have put a cout in the .cpp file to show us an output which will let us know that the program has run without errors.

**Code in main:**

void main()

{

cout<<"-----------------------Main has Started----------------------------" <<endl;

cout<<"-----------------------Main is Ending------------------------------" << endl;

}

**Result:**

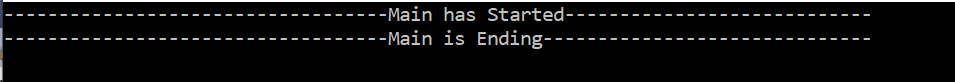


Figure 1 Result for empty class

## 6.2 Test 2 seismic //Constructor

class seismic

{

public:

seismic();

private:

int DataWindowST;

int DataWindowLT;

float TresholdVlaue;

};

inline seismic ::seismic()

{

DataWindowST = 0;

DataWindowLT = 0;

TresholdVlaue = 0.0;

cout << " Debug from default constructor" << endl;

cout << "The Short Term Data Window = " <<DataWindowST<< endl;

cout << "The Long Term Data Window = " <<DataWindowLT<< endl;

cout << "The Threshold value is = " <<TresholdVlaue<< endl;

}

**Code in main:**

void main()

{

seismic E1;

}

**Result:**

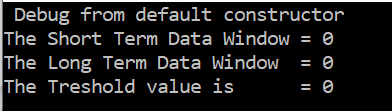


Figure 2 Result from Default Constructor

We have used the default constructor to initialise the attributes to zero.

## 6.3 Test 3 seismic //Regular Constructor

class seismic

{

public:

seismic(int aST, int aLT, float aTreshold);

private:

int DataWindowST;

int DataWindowLT;

float TresholdVlaue;

};

inline seismic::seismic(int aST, int aLT, float aTreshold)

{

DataWindowST = aST;

DataWindowLT = aLT;

TresholdVlaue = aTreshold;

cout << "\nDebug from Regular constructor" << endl;

cout << "The Short Term Data Window = " <<DataWindowST<< endl;

cout << "The Long Term Data Window = " <<DataWindowLT<< endl;

cout << "The Threshold value is = " <<TresholdVlaue<< endl;

}

**Code in main:**

void main()

{

seismic Event(2,4,1.5);

}

**Result:**

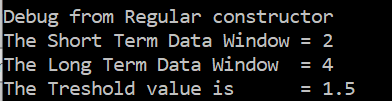


Figure 3 Result from regular Constructor

Regular constructor is allowing the user to enter values from main, where we had set them to 2,4 & 1.5.

## 6.4 Test 4 setDataWindowST()

class seismic

{

public:

void setDataWindowST(int aSTvalue);

private:

int DataWindowST;

};

inline void seismic::setDataWindowST(int aSTvalue)

{

DataWindowST = aSTvalue;

cout<<"Debug from setDataWindowST(), DataWindowST = "<<DataWindowST<<endl;

}

**Code in main:**

void main()

{

Event.setDataWindowST(3);

}

**Result:**



Figure 4 Result from setDataWindowST()

Figure 4 shows the set short term data window.

## 6.5 Test 5 setDataWindowLT()

class seismic

{

public:

void setDataWindowLT(int aLTvalue);

private:

int DataWindowLT;

};

inline void seismic::setDataWindowLT(int aLTvalue)

{

DataWindowLT = aLTvalue;

cout<<"Debug from setDataWindowLT(), DataWindowLT = "<<DataWindowLT<<endl;

}

**Code in main:**

void main()

{

Event.setDataWindowLT(6);

}

**Result:**

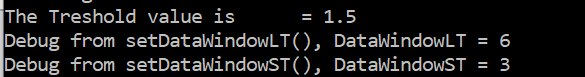


Figure 5 Result from setDataWindowLT()

Figure 5 shows the set long term data window.

## 6.6 Test 6 setTresholdVlaue()

class seismic

{

public:

void setTresholdVlaue(float aTreshold);

private:

double TresholdVlaue;

};

inline void seismic::setTresholdVlaue(float aTreshold)

{

TresholdVlaue =aTreshold;

cout<<"Debug from setThresholdVlaue(), TresholdVlaue = "<<TresholdVlaue<<endl;

}

**Code in main:**

void main()

{

Event.setTresholdVlaue(1.8);

}

**Result:**



Figure 6 Result from setTresholdVlaue()

Figure 6 shows the set threshold value.

## 6.7 Test 7 getDataWindowST()

class seismic

{

public:

int getDataWindowST();

private:

int DataWindowST;

};

inline int seismic::getDataWindowST()

{ return DataWindowST;

}

**Code in main:**

void main()

{

cout<<"\nDebug from getDataWindowST(), ST data window = "<<Event.getDataWindowST()<<endl;

}

**Result:**



Figure 7 Result from getDataWindowST()

Figure 7 shows that the attribute value from *DataWindowST* is returned correctly to main.

## 6.8 Test 8 getDataWindowLT()

class seismic

{

public:

int getDataWindowLT();

private:

int DataWindowLT;

};

inline int seismic::getDataWindowLT()

{ return DataWindowLT;

}

**Code in main:**

void main()

{

cout<<"Debug from getDataWindowLT(), LT data window = "<<Event.getDataWindowLT()<<endl;

}

**Result:**



Figure 8 Result from getDataWindowLT()

Figure 8 shows that the attribute value from *DataWindowLT* is returned correctly to main.

## 6.9 Test 9 getTresholdVlaue()

class seismic

{

public:

float getTresholdVlaue();

private:

float TresholdVlaue;

};

inline float seismic::getTresholdVlaue()

{ return TresholdVlaue;

}

**Code in main:**

void main()

{

cout<<"Debug from getTresholdVlaue(), Threshold = "<<Event.getTresholdVlaue()<<endl;

}

**Result:**



Figure 9 Result from getTresholdVlaue()

Figure 9 shows that the attribute value from *TresholdVlaue* is returned correctly to main.

## 6.10 Test 10 askDataWindows()

class seismic

{

public:

void askDataWindows();

private:

int DataWindowST;

int DataWindowLT;

};

inline void seismic::askDataWindows()

{

cout<<"\n Enter the Data window value you would like to use for Short-Term power => ";

cin>>DataWindowST;

cout<<"\n Enter the Data window value you would like to use for Long-Term power => ";

cin>>DataWindowLT;

}

**Code in main:**

void main()

{

Event.askDataWindows();

}

**Result:**

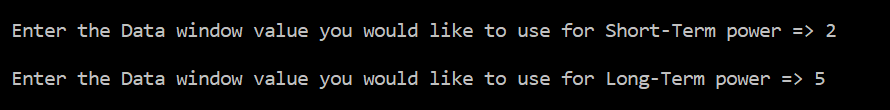


Figure 10 Result from askDataWindows()

The ask data windows has displayed text and waits for user to enter in a value before asking for the next attribute.

## 6.11 Test 11 askTresholdVlaue()

class seismic

{

public:

void askTresholdVlaue();

private:

float TresholdVlaue;

};

inline void seismic ::askTresholdVlaue()

{

cout<<"\n Enter The Treshold vlaue you would like to use => ";

cin>>TresholdVlaue;

}

**Code in main:**

void main()

{

Event.askTresholdVlaue();

}

**Result:**



Figure 11 Result from askTresholdVlaue()

The value entered by the user can be seen in figure 11 as 1.5 for the threshold.

## 6.12 Test 12 cal\_ST\_Power()

class seismic

{

public:

float cal\_ST\_Power(float Array[], int num);

private:

int DataWindowST;

int DataWindowLT;

float TresholdVlaue;

};

inline float seismic::cal\_ST\_Power(float Array[], int num)

{

float sum;

sum = 0;

for(int i = num; i > num - DataWindowST ; i--)

{ sum = (Array[i]\*Array[i]) + sum;

cout<<"The ST array value = "<<Array[i]<<"| squared = "<<sum<<endl;

}

return (sum /DataWindowST);

}

**Code in main:**

void main()

{

for (int s = ST+1 ; s<= TotalValues+1; s++ )

{

tempArrayST[f] = Event.cal\_ST\_Power(Array, s );

cout<<"the step value is = "<<s<<endl;

cout<<"Temp array ST is => "<<f<<" - "<<tempArrayST[f]<<endl;

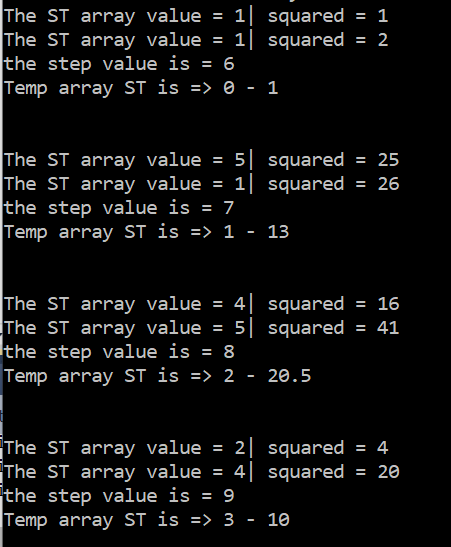
f++;

cout<<"\n"<<endl;

}

}

**Result:**



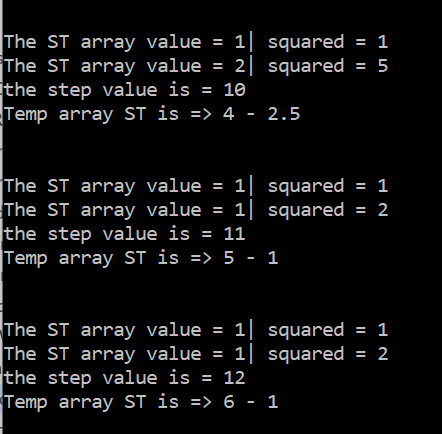


Figure 12 Result from cal\_ST\_Power()

Using the second result from figure 12: “The ST array value = 5| squared = 25

The ST array value = 1| squared = 26

The step value is = 7

Temp array ST is => 1 – 13 “

Line one tells us that for the short-term array it is first using the value five then squaring it, showing that it is twenty-five.

Line two tells us that for the short-term array it then goes to the next value of one, squares it which is still one and adds it to twenty-five thus both 25 & 1 squared equal 26.

Line three tells us that it’s going to the seventh data value and using the for loop to decrement by two, which is the short-term value used. In so doing it uses the data values of 5 and 1 for this calculation.

Line four tells us that it is saving the resulting answer from the calculation after it has been divided by the short-term value into the array *tempArrayST* that we created in main. So, in the array element of 1 we are saving the answer of 13.

## 6.13 Test 13 cal\_LT\_Power()

class seismic

{

public:

float cal\_LT\_Power(float Array[], int num);

private:

int DataWindowST;

int DataWindowLT;

float TresholdVlaue;

};

inline float seismic::cal\_LT\_Power(float Array[], int num)

{

float sum;

sum = 0;

for(int i = num; i> num - DataWindowLT ; i=i -1)

{ sum = (Array[i]\*Array[i]) + sum;

cout<<"The LT array value = "<<Array[i]<<"| squared = "<<sum<<endl;

}

return (sum /DataWindowLT);

}

**Code in main:**

void main()

{

for (int s = Event.getDataWindowLT()+1 ; s<= TotalValues+1; s++ )

{

tempArrayLT[g] = Event.cal\_LT\_Power(Array, s );

cout<<"-----the step value is => "<<s<<endl;

cout<<"-----Temp array LT is => "<<g<<" - "<<tempArrayLT[g]<<endl;

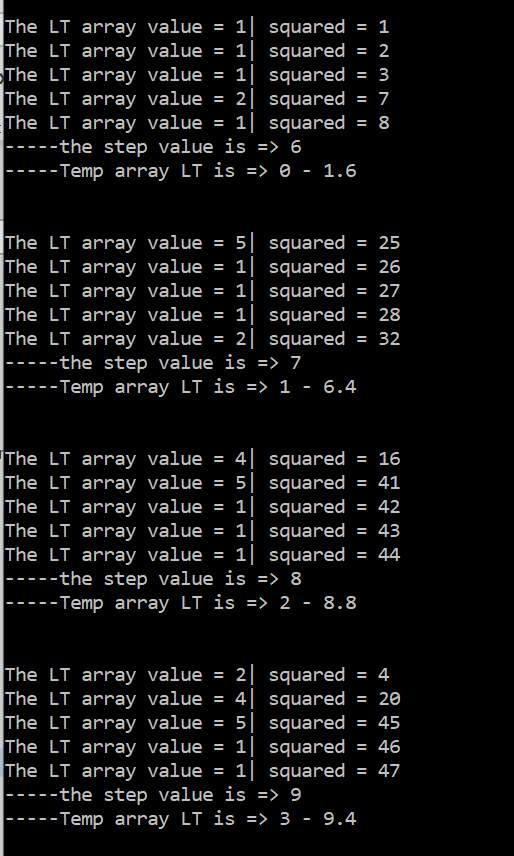
g++;

cout<<"\n"<<endl;

}

}

**Result:**



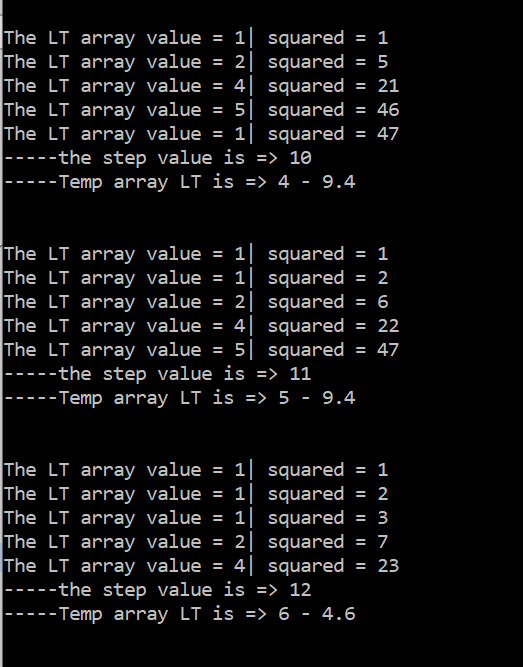


Figure 13 Result from cal\_LT\_Power()

Using the second result from figure 13: “The LT array value = 5| squared = 25

The LT array value = 1| squared = 26

The LT array value = 1| squared = 27

The LT array value = 1| squared = 28

The LT array value = 2| squared = 32

-----the step value is => 7

-----Temp array LT is => 1 - 6.4 “

Line one tells us that for the long-term array it is using the value five then squaring it, showing that it is twenty-five.

Line two tells us that for the long-term array it then goes to the next value of one, squares it which is still one and adds it to twenty-five thus both 25 + 1 equal 26.

By the time it gets to line three the function has the value 26 from the previous calculation so it adds to it be squaring the third data value of one which is still one and adds that to the previous calculation thus the calculation so far has the result of 25 + 1 + 1 equal 27.

Line six tells us that it’s going to the seventh data value and using the for loop to decrement by five, which is the long-term value used. In so doing it uses the data values of 5,1,1,1,2 for this calculation.

Last line tells us that it is saving the resulting answer from the calculation after it has been divided by the long-term value into the array *tempArrayLT* that we created in main. So, in the array element of 1 we are saving the answer of 6.4.

## 6.14 Test 14 displayResults()

class seismic

{

public:

void displayResults(float NumElements, float TimeInterval,float tempArrayST[], float tempArrayLT[] );

private:

int DataWindowST;

int DataWindowLT;

float TresholdVlaue;

};

inline void seismic::displayResults(float NumElements, float TimeInterval,float tempArrayST[], float tempArrayLT[] )

{

int h;

h= (DataWindowLT +1)-2 ;

for(int y=0 ; y<(NumElements+1)-DataWindowLT ; y++)

{

cout<<"\nThe point at "<<( h \* TimeInterval)<<endl;

cout<<"The Short Term power = "<<tempArrayST[y]<<endl;

cout<<"The Long Term power = "<<tempArrayLT[y]<<endl;

cout<<"The Ratio = "<<tempArrayST[y]<<"/"<<tempArrayLT[y]<<" = "<<(tempArrayST[y]/tempArrayLT[y])<<endl;

cout<<"The Threshold is = "<<TresholdVlaue<<endl;

if( tempArrayST[y]/tempArrayLT[y] >TresholdVlaue )

{

cout<<"Result - Event at time "<<( h \* TimeInterval)<<endl;

}

else cout<<"Result - No Event at time "<<( h \* TimeInterval)<<endl;

h++;

}

}

**Code in main:**

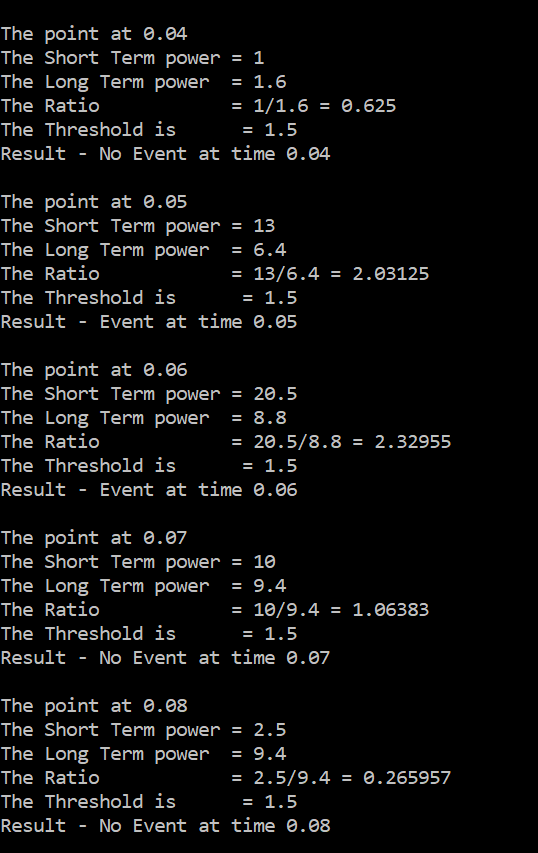
void main()

{

Event.displayResults(TotalValues,TimeInterval,tempArrayST,tempArrayLT);

}

**Result:**



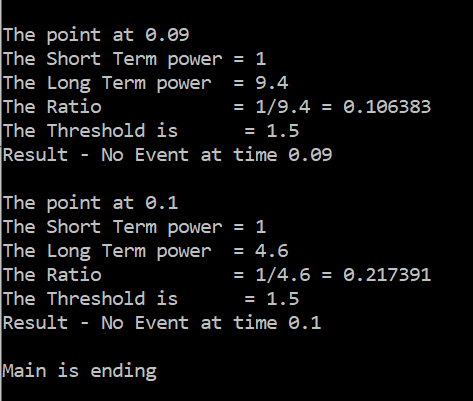


Figure 14 Result from displayResults()

In figure 14 we are displaying the results in an easy to read format and displaying if there has been an event or not, as per the requirements.

# 7.0 Evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Behaviour  Under Test | Expected outcome | Actual Outcome | Comments/  discussion |
| 1 | Empty class | Blank console | Blank console | Code functioned as expected |
| 2 | seismic();  //default | Debug from default constructor  The Short-Term Data Window = 0  The Long-Term Data Window = 0  The Threshold value is = 0 | Debug from default constructor  The Short Term Data Window = 0  The Long Term Data Window = 0  The Threshold value is = 0 | The code did what it we expected. |
| 3 | seismic(…);  //regular | Debug from Regular constructor  The Short Term Data Window = 2  The Long Term Data Window = 4  The Threshold value is = 1.5 | Debug from Regular constructor  The Short Term Data Window = 2  The Long Term Data Window = 4  The Threshold value is = 1.5 | The correct values are shown with their text. |
| 4 | setDataWindowST(); | Debug from setDataWindowST(), DataWindowST = 3 | Debug from setDataWindowST(), DataWindowST = 3 | The code works as expected. |
| 5 | setDataWindowLT(); | Debug from setDataWindowLT(), DataWindowLT = 6 | Debug from setDataWindowLT(), DataWindowLT = 6 | The code works as expected. |
| 6 | setTresholdVlaue(); | Debug from setTresholdVlaue(), TresholdVlaue = 1.8 | Debug from setTresholdVlaue(), TresholdVlaue = 1.8 | code shows the set threshold value correctly. |
| 7 | getDataWindowST() | Debug from getDataWindowST(), ST data window = 3 | Debug from getDataWindowST(), ST data window = 3 | The code returned the right value with debug text. |
| 8 | getDataWindowLT() | Debug from getDataWindowLT(), LT data window = 6 | Debug from getDataWindowLT(), LT data window = 6 | The code returned the right value with debug text. |
| 9 | getTresholdVlaue() | Debug from getTresholdVlaue(), Treshold = 1.8 | Debug from getTresholdVlaue(), Treshold = 1.8 | The code returned the right value with debug text. |
| 10 | askDataWindows() | Enter the Data window value you would like to use for Short-Term power =>  Enter the Data window value you would like to use for Long-Term power => | Enter the Data window value you would like to use for Short-Term power =>  Enter the Data window value you would like to use for Long-Term power => | asking for user input in the correct order. |
| 11 | askTresholdVlaue() | Enter the Threshold value you would like to use => | Enter the Threshold value you would like to use => | works as expected. |
| 12 | cal\_ST\_Power() | Element – value  Temp array ST is => 0 – 1  Temp array ST is => 1 – 13  …… | Element – value  Temp array ST is => 0 – 1  Temp array ST is => 1 – 13  …… | the debug code inside this function show the calculation as it runs. |
| 13 | cal\_LT\_Power() | Element – value  Temp array LT is => 0 - 1.6  Temp array LT is => 1 - 6.4  ……… | Element – value  Temp array LT is => 0 - 1.6  Temp array LT is => 1 - 6.4  ……… | the debug code inside this function show the calculation as it runs. |
| 14 | displayResults() | The point at 0.04  The Short-Term power = 1  The Long-Term power =1.6  The Ratio = 1/1.6 = 0.6  The Threshold is = 1.5  Result-No Event at time 0.04  ………  ……… | The point at 0.04  The Short-Term power = 1  The Long-Term power = 1.6  The Ratio = 1/1.6 = 0.6  The Threshold is = 1.5  Result - No Event at time 0.04  ………  ……… | The layout is just as we wanted and the correct values are showing in each line. |

# 8.0 Conclusion

We have solved the Class seismic problem, by recognising that we would require arrays, file read/write capability and graphics we were able to test each of these using simple classes, ensuring that we could correctly use all three for use in the solution of the seismic event. By putting all the attributes and behaviours that were required to solve the problem in a table we were able to make an expected outcome prediction. Knowing what our expected outcome should be we were able to follow out our solution design and code the program line by line, while making sure each of the added code preformed as expected. The final result on the monitor showed our program is working as expected, with all the behaviours and attributes working as one C++ program.