**WJEC Computer Science  
Controlled Assessment**

**Lucky Numbers**

**Produced by Adam Myczkowski**

Contents

[Initial Analysis 4](#_Toc477270761)

[Which scenario and why? 4](#_Toc477270762)

[What is Lucky Numbers all about? 4](#_Toc477270763)

[How I am going to do that? 4](#_Toc477270764)

[What language I am going to write it in and what IDE will I use to produce it? 4](#_Toc477270765)

[Design Specification 5](#_Toc477270766)

[Algorithm 5](#_Toc477270767)

[Flow Charts 5](#_Toc477270768)

[Documentation 5](#_Toc477270769)

[Main algorithms 6](#_Toc477270770)

[Change history 7](#_Toc477270771)

[Version 0.2 7](#_Toc477270772)

[Version 0.3 8](#_Toc477270773)

[Version 1.0 8](#_Toc477270774)

[Version 1.1 – Add support for capital letters 9](#_Toc477270775)

[Version 1.2 – Actually terminate program when error message is being called 9](#_Toc477270776)

[Version 1.3 – Fix case when letters add up to 3 digit number 10](#_Toc477270777)

[Version 1.4 – Better way of closing the program using kbhit() 10](#_Toc477270778)

[Version 1.5 – Log results of calculation onto a text file 11](#_Toc477270779)

[Version 1.6 – Fix wrong order logging 12](#_Toc477270780)

[Version 1.7 – Introduce switch() 13](#_Toc477270781)

[Version 1.8 – Drop switch() for find() function 13](#_Toc477270782)

[Version 1.9 – Add option to use middle name and optimise memory allocation 14](#_Toc477270783)

[Version 2.0 – Rewrite the comments and make the way of how output is displayed more user-friendly 15](#_Toc477270784)

[Version 2.1 – A little bit more optimisations 15](#_Toc477270785)

[Version 2.2 – Change the way int is separated into single digts and declare whole array at once 16](#_Toc477270786)

[Version 2.3 – Optimise the output 16](#_Toc477270787)

[Problems: 16](#_Toc477270788)

[Known issues and how to solve them 16](#_Toc477270789)

[Postproduction Testing 17](#_Toc477270790)

[Evaluation 17](#_Toc477270791)

[Success criteria 17](#_Toc477270792)

[How could I improve my program: 18](#_Toc477270793)

[Maintainability: 18](#_Toc477270794)

[Reliability: 18](#_Toc477270795)

[Performance: 18](#_Toc477270796)

[Further improvements: 18](#_Toc477270797)

# Initial Analysis

## Which scenario and why?

I have chosen to produce a working solution for Lucky Numbers scenario. I have chosen Lucky Numbers because this scenario seems to be easier to implement than the Money Bags, at least for me. I feel quite confident choosing this scenario because I knew what I have to do just after I have read it.

## What is Lucky Numbers all about?

My program has to use Pythagoras’ method to calculate a lucky number based on persons’ name. You have to add numbers associated with letters together, and then add number’s digits together. If it’s still more than one digit you have to add them again and again until it’s a single digit (1-9). Each digit has its own meaning. My program has to input a person’s name, calculate their lucky name number using the Pythagoras’ method, display user’s name and also display their lucky number with its meaning.

## How I am going to do that?

Input name and store as a string, then convert it into an array that then I will loop through to check if it equals to certain letters that I associated with certain number e.g. if the first character of user’s name is ‘a’, ‘j’ or ‘s’ add 1 to integer that stores total value of the name. If it doesn’t equal to any character from English alphabet program will throw and error and terminate the process. When it’s all right my program will add these numbers and using loop I will split it into digits that then add together, check it it’s still a single digit, if yes display name, lucky number and its meaning, if not split the number again.

## What language I am going to write it in and what IDE will I use to produce it?

I have chosen C++ as a language in which I will produce my solution mainly because it is the only programming language I fully understand what makes it easier for me. Also C++ is one of the most efficient programming languages because it compiles and runs directly on the machine you are working on, what means nothing slows it down, what is the case with Java as it runs in a virtual machine. There are obviously few concerns of C++ running directly on your computer’s hardware: The main concern is that it’s hard to write multiplatform programs as it’s being compiled for certain operating system and won’t run on different one. For example at home I work with Linux and if I compile C++ program on Linux I cannot run it on Windows at school, I would have to copy the source code and recompile it with C++ compiler for Windows which is not very efficient way of creating multiplatform application. As I have only Windows to work with I do not need to care about multiplatform applications and I can still use C++. The school was very kind and installed Microsoft Visual Studio which is a C++ IDE for Windows that comes with its own compiler. I asked for it because it’s one of the most popular C++ IDEs for Windows, I use Eclipse CDT with G++ compiler at home (Linux) and I would use Eclipse CDT but there would be many issues setting up the Eclipse for Windows with MinGW (minimal GNU compiler for Windows) which I also had at home, it doesn’t automatically find the compiler and I have to add linker flags to each project manually, plus I need administrator rights to set up MinGW itself. Microsoft Visual Studio can be used to write programs in range of different languages e.g. Python, Java, C, C++ and C#. It can be used to write visual apps for Windows as well as usual command line programs which I will produce as I do not know Microsoft’s library for the visual programs. It has syntax highlighters, debugger, error messages and other standard features each IDE should have.

# Design Specification

The stakeholders of this project are the user, the customer, the developer. The user is the person that uses the program at the end of the day, they expect it to be easy to use, tell them if they made any mistakes, etc. The customer is the person that pays for the project to be produced in the real world; they expect a fully working solution that will complete all the tasks set in the success criteria; program is expected to be efficient and user friendly, validation is needed. The developer (me) is the person that develops the solution for given task; we expect good facilities like suitable and functional development environment such as Visual Studio.

## Algorithm

Look “Algorithm” in my Appendix.

## Flow Charts

Look “Flow charts” in my Appendix.

## Documentation

* **Author's name :** Adam Myczkowski
* **Last Modified by**: Adam Myczkowski
* **Date last Modified**: 14/03/2017
* **Program description:**

The app uses Pythagoras’ method to calculate your lucky number. All you have to do is enter you first and last name as it tells you in the app, it will do all the hard work for you. Don't worry if you have a middle name - I thought about it. The app asks if you have a middle name, if you want to enter it simply press 'y' for yes, if you don't - well, no worries, press 'n' for no and it won't ask you for it. If you really curious what values your names sum up to the app creates a text file LOG.txt in the same location - This files displays some of the calculations.

* **Instructions:**
* Double click on the program.
* Enter your first name when asked.
* Press ‘y’ to enter you middle name, ‘n’ to skip it.
* Enter last name when asked.
* View results
* **System Requirements:** Windows 7, 8.1 (\*), Visual C++ Redistributable
* **Files size:**
* LuckyNumbers-v2.3-EWTC-x86\_release.exe (36KB)
* LuckyNumbers.cpp (13KB)
* msvcp140.dll (422KB)
* vcruntime140.dll (83KB)
* **Program language used:** C++
* **Current stable version:** v2.3

\* - As we have computers with Windows 7 at school the only supported platform versions in Visual Studio are Windows 7 and Windows 8/8.1. It might work on Windows 10 but it is very unlikely and might cause performance issues. To safely run on Windows 10 you would have to recompile on this platform.

Main algorithms**:**

* Get name, convert it into an array, get total number value of the input:

cout << "Enter FIRST name here >> " << flush;

cin >> FrNm;

LOG << "First name: " << FrNm << endl;

char \*FrNmAr = const\_cast<char\*>(FrNm.c\_str()); // converts a string into char\* (pointer to a char, basically a char array)

for (unsigned int i = 0; i<FrNm.length(); i++) {

tmp = alphabet.find(FrNmAr[i]); // This function takes a char from char\*, tries to find it in the alphabet string, returns its positiion as a integer and writes results to tmp integer

if (tmp == string::npos) { // npos (-1) is returned when char we looking for is not in a given string (no position)

tmp = CAPalphabet.find(FrNmAr[i]); // This function takes a char from char\*, tries to find it in the CAPalphabet string, returns its positiion as a integer and writes results to tmp integer

if (tmp == string::npos) { // npos (-1) is returned when char we looking for is not in a given string (no position)

cout << "Invalid character " << FrNmAr[i] << " ,please try again! Press any key to exit . . ." << endl;

while (true) {

if (\_kbhit()) { // Terminate the program if any key is pressed

LOG << "Session Failed" << endl;

LOG.close();

return EXIT\_FAILURE;

}

}

}

}

FrNmVal += (tmp % 9) + 1; // Use simple mathematical equation to change the tmp int into a right number as in the cheatsheet, for example 'c' is in pos 2, 2 % 9 = 2 because 2 / 9 = 0.(2), 2 + 1 = 3. ^Lookup cheatsheet: c = 3.

LOG << i + 1 << ". " << FrNmVal << endl;

}

LOG << "Total is " << FrNmVal << endl; //My name (Adam) should be 1 + 4 + 1 + 4 = 10, lets see...

* Ask for middle name:

cout << "Do you have a middle name?? (y)es/(n)o" << endl;

MID = \_getch(); // getch is a function from conio.h that is able to detect what key have you pressed on keyboard, you can use it with keys' ASCII and Hex values

while (MID != KEY\_Y && MID != KEY\_N) {

if (MAX\_LIMIT <= 0) {

cout << "You have pressed wrong button 10 times, application will be closed now. Press any key to exit . . ." << endl;

while (true) {

if (\_kbhit()) { // Terminate the program if any key is pressed

LOG << "Session Failed" << endl;

LOG.close();

return EXIT\_FAILURE;

}

}

}

cout << "Wrong key pressed, press (y) for yes and (n) for no!" << endl;

MID = \_getch();

MAX\_LIMIT--;

}

if (MID == KEY\_Y) {

HasMidName = true;

}

else if (MID == KEY\_N) {

HasMidName = false;

}

if (HasMidName == true) { // Execute only if user choose option to use middle name

* Split numbers into single-digit integers

while (FrNmVal > 9) { // if number is lower than 9 it's a single digit and doesn't need to be split

FrNmValS = to\_string(FrNmVal);

FrNmValAr = const\_cast<char\*>(FrNmValS.c\_str());

if (FrNmVal > 99) {

FrNmVal = (FrNmValAr[0] - '0') + (FrNmValAr[1] - '0') + (FrNmValAr[2] + '0');

LOG << "Dig1 is: " << FrNmValAr[2] << " , " << flush;

LOG << "Dig2 is: " << FrNmValAr[0] << " , " << flush;

LOG << "Dig3 is: " << FrNmValAr[1] << " , " << flush;

// dig3 is actually the farest number to to left when 3-digit number is called,

// changing int names would introduce too much mess to the program hence

// just putting dig3 as dig1 and moving dig1 and dig2 by 1 to the right

}

else {

FrNmVal = (FrNmValAr[0] - '0') + (FrNmValAr[1] - '0');

LOG << "Dig1 is: " << FrNmValAr[0] << " , " << flush;

LOG << "Dig2 is: " << FrNmValAr[1] << " , " << flush;

}

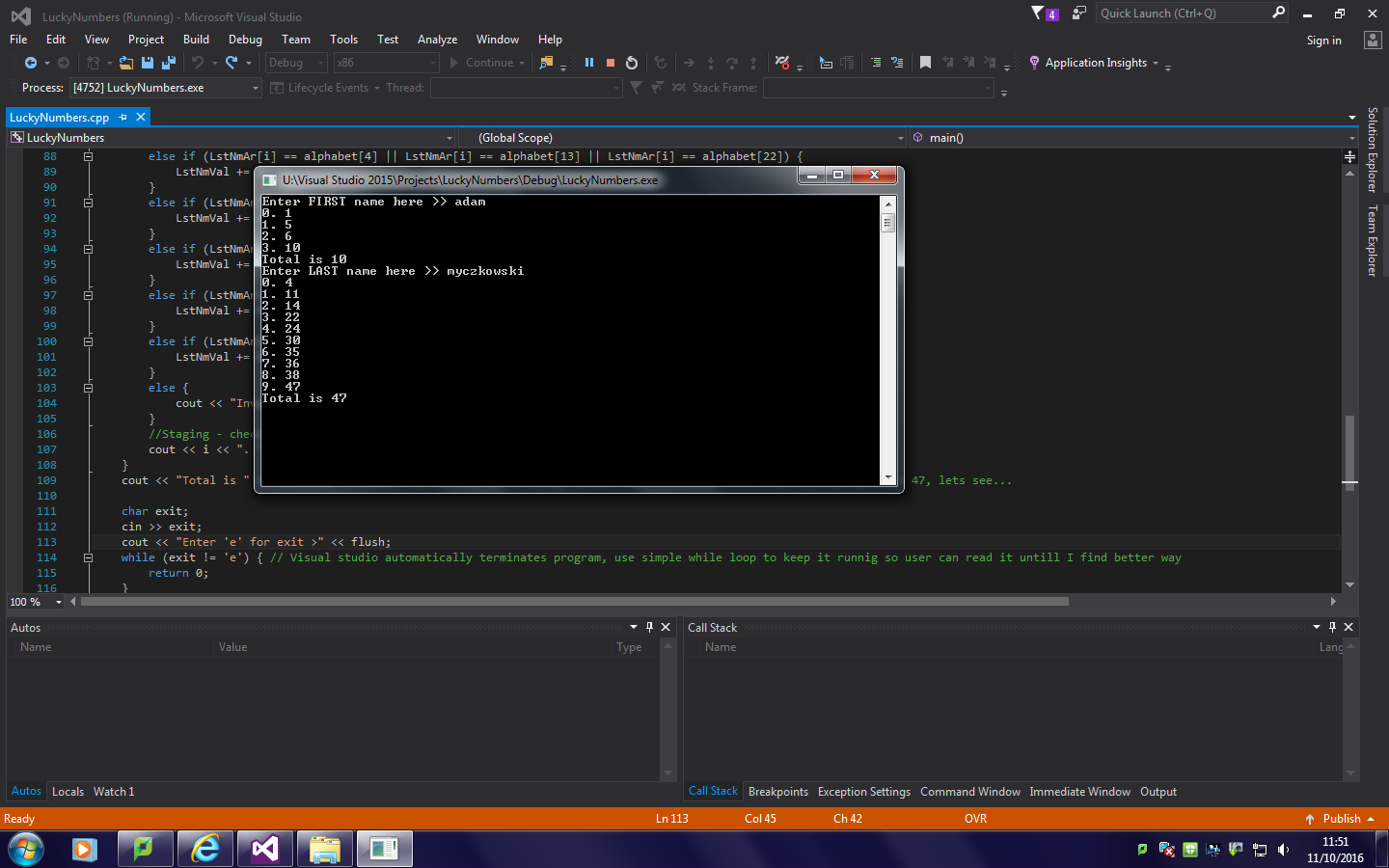
}

LOG << "First name digit: " << FrNmVal << endl;

Change history**:**

### Version 0.2

For code look “Source Code 0.2 ” pdf file.



After first time I sat to code my solution was half way finished. I managed to write a simple program that gathers input as a string and then converts it into char\* array. Then using simple “if” statement inside a “for” loop that loops through the characters in the char\* array I compare those characters to characters from array with alphabet letters. It checks if it equals to certain letters and then adds appropriate number to an integer that stores total value. It does the same thing for first and last name, only the variables’ names are different. Visual Studio closes the command window directly after it returns 0 so I had to implement very simple method of closing the program so that the user is able to read the information before program is closed down – input character and while loop that checks if the character is equal to “e” for exit.

### Version 0.3

For code look “Source Code 0.3 ” pdf file.

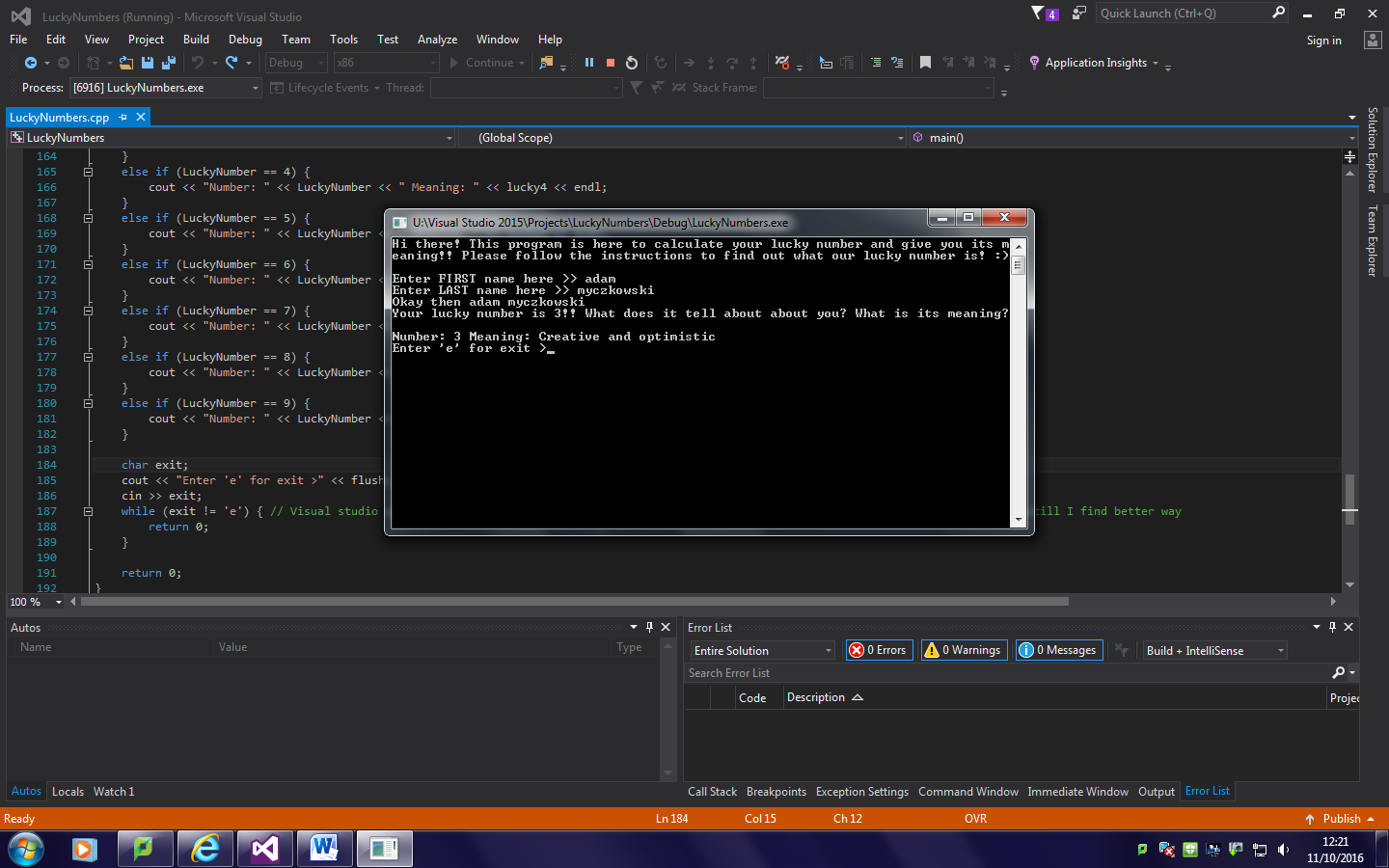
“For” loops were returning a warning:

"warning: comparison between signed and unsigned integer expressions [-Wsign-comapre]”

This was because in the for loops I compare int “i” with a function that check the length of a string and returns its value, but as I didn’t store it as a variable it was unsigned, there was 2 possible ways of fixing it – make int unsigned (what I did) or store the length of a string as an integer and then compare it. Making int unsigned was more efficient and saved few lines of code that’s why I did it.

### Version 1.0

For code look “Source Code 1.0” pdf file.

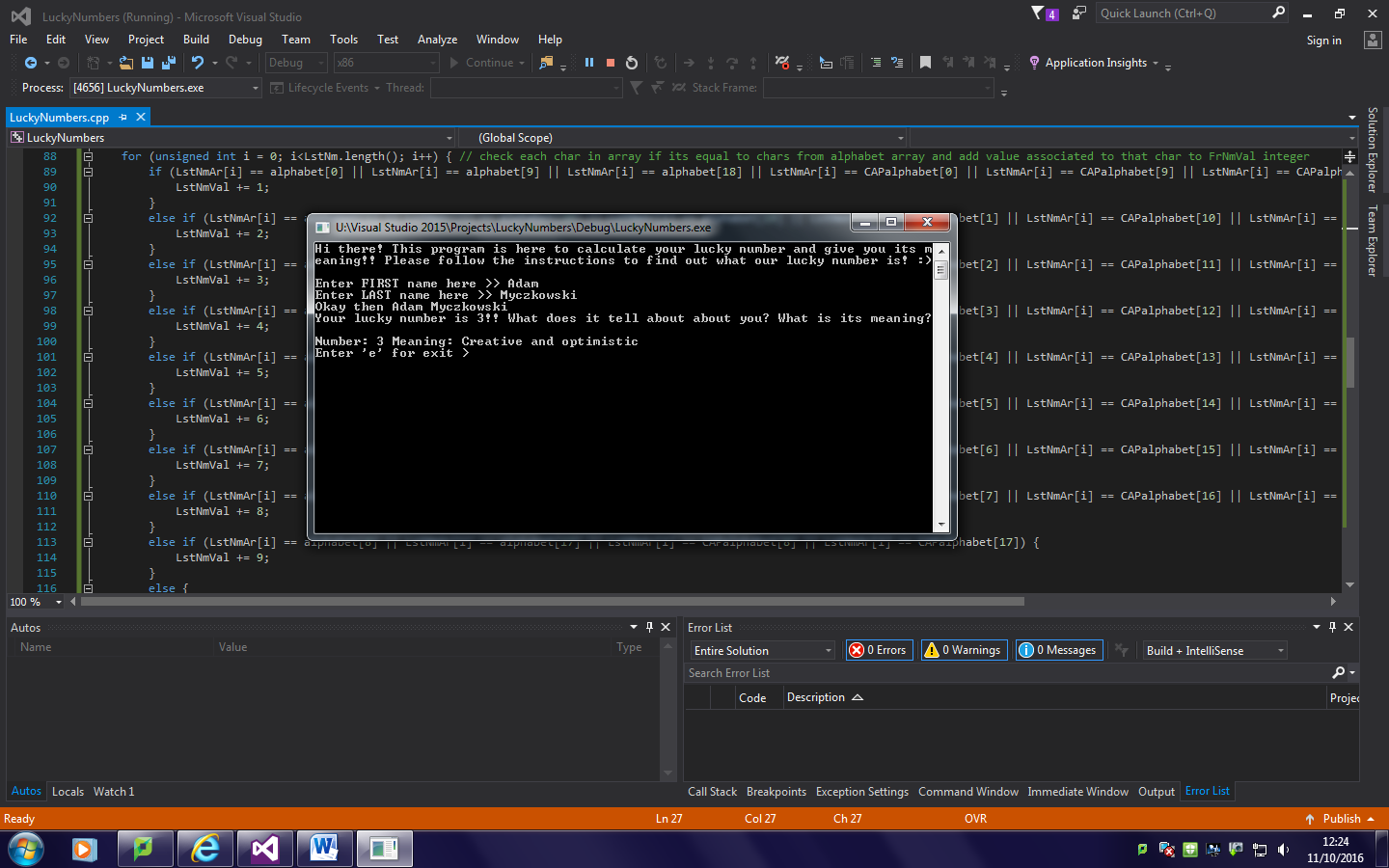


The code isn’t well optimised yet but it does what it’s supposed to – calculates and displays your lucky number and its meaning. I have added set if string with meanings of the numbers. I also added greetings that tell you what the program does and what you’re supposed to do. I’ve commented out all the debug code that displays what particular functions I used return. I found a way to split the number into single digits:

First of all I’ve made a copy of first and last name total integers in case I had to use these numbers again. Then I used these copies in a while loop. While loop check if the number is more than 9, if it’s not it must be a single digit so it moves to the next function. If it’s more than 9 it must have 2 or more digits so I used % (modulo) function that returns reminder from division. To find the front digit I divided the copy if total by 10 and then % 10. For example let’s say that number is 47. To find fir digit I would do 47 / 10 = 4 (because it’s integer). Then 4 % 10 = 4 because 4 / 10 = 0.4 (reminder 4) hence 4 % 10 = 4. To calculate second digit I’ve done exactly the same thing except I didn’t divide by 10. So first digit of 47 is (47 / 10) % 10 = 4 and second digit is 47 % 10 = 7. Once I’ve got both digits calculated I re-declare the copy of the original number by setting it equal to digit 1 + digit 2. Then while loop goes back and checks if it’s still more than 9. Again if not, it moves to the next function and if yes it does the same thing again. The same function is used for last name and actual lucky number. When lucky number is calculated, using simple ‘if’ statements I print the meaning of lucky number on the screen.

### Version 1.1 – Add support for capital letters

For code look “Source Code 1.1 ” pdf file.



I thought it would be nice to be able to use capitals as well as lowercase letters so I have added another char array with alphabet letters but this time capitals and in the function that looks up the alphabet array I added code so it check if it’s not a capital letter as well.

### Version 1.2 – Actually terminate program when error message is being called

For code look “Source Code 1.2 ” pdf file.

I knew there can be a case when some mistypes and enters a number or other special symbol so I made my program to print error message when the character cannot be found in any of the arrays but I didn’t realise that program carries on, it doesn’t stop because I’ve forgotten to call exit.. Returning 0 was too mainstream so I used EXIT\_SUCCESS and EXIT\_FAILURE functions from cstdlib and now program is being terminated when character cannot be found.

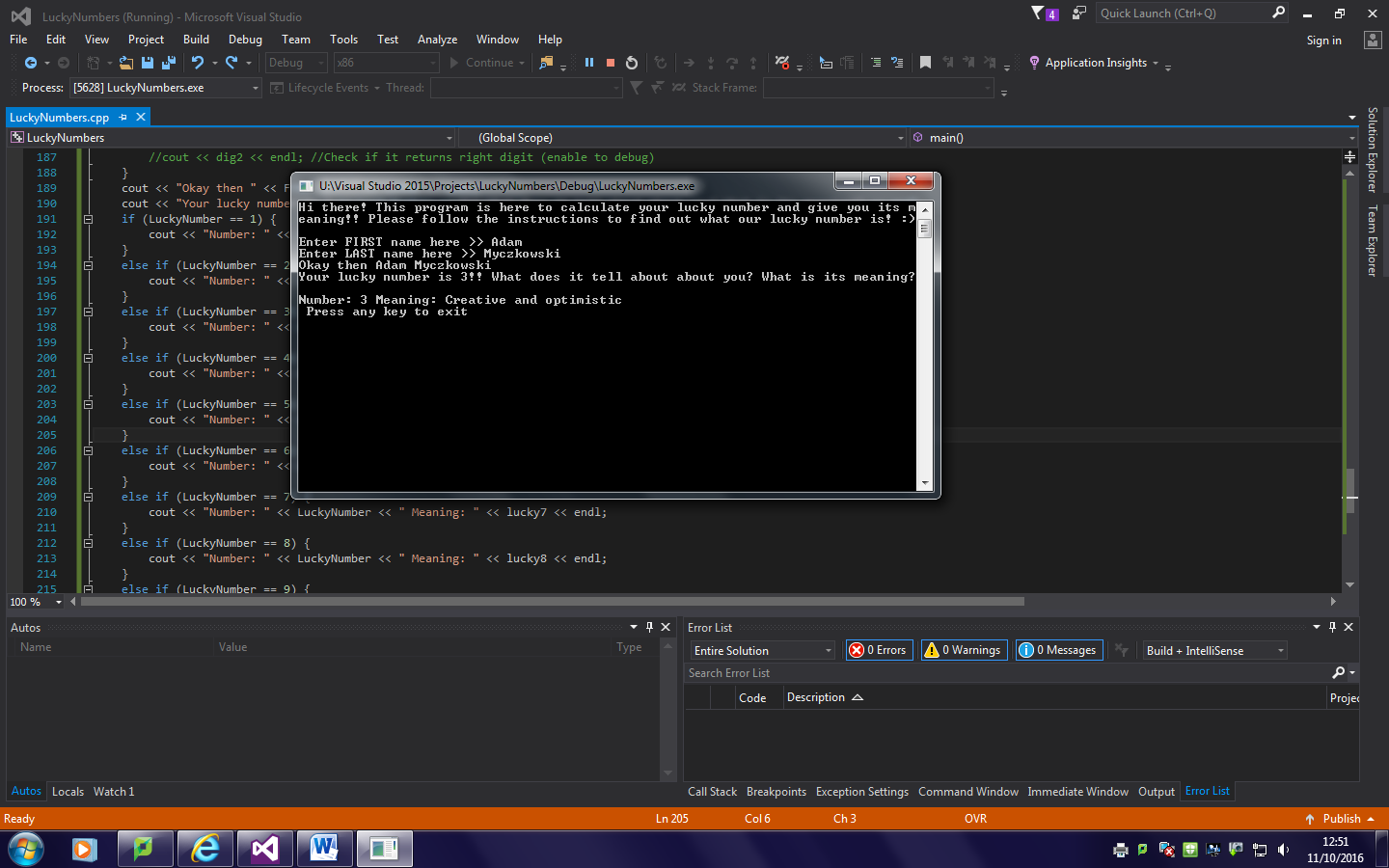
### Version 1.3 – Fix case when letters add up to 3 digit number

For code look “Source Code 1.3” pdf file.

Mr Newton pointed out to me that there is a small possibility of someone having 3-digit number from their name or surname so I decided to support 3 digit numbers – In the functions that split number into digits I have added if statement that checks if number is bigger than 99 and if yes find the 3rd digit and add 1st, 2nd and 3rd together and if not only add 1st and 2nd.

### Version 1.4 – Better way of closing the program using kbhit()

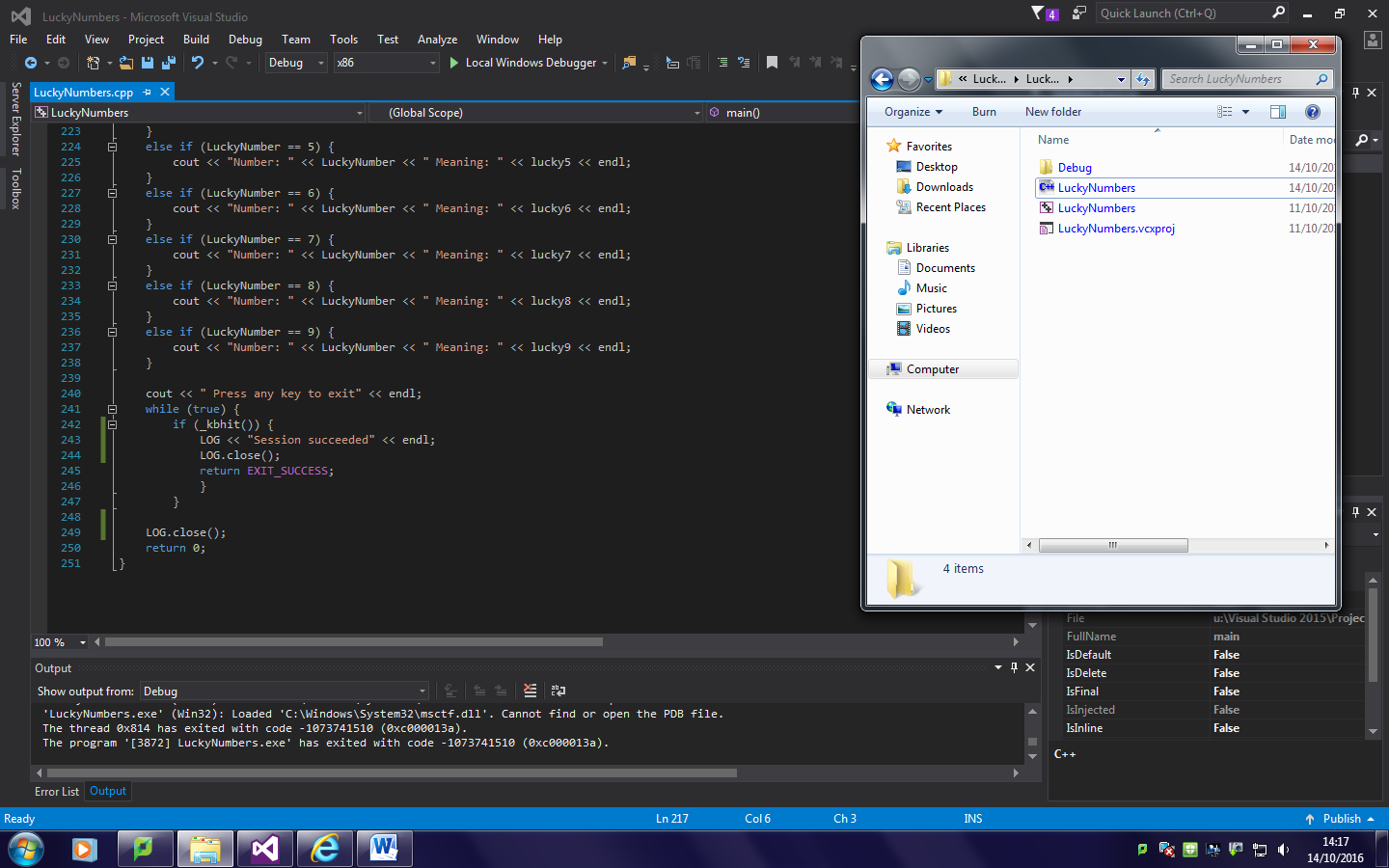
For code look “Source Code 1.4” pdf file.

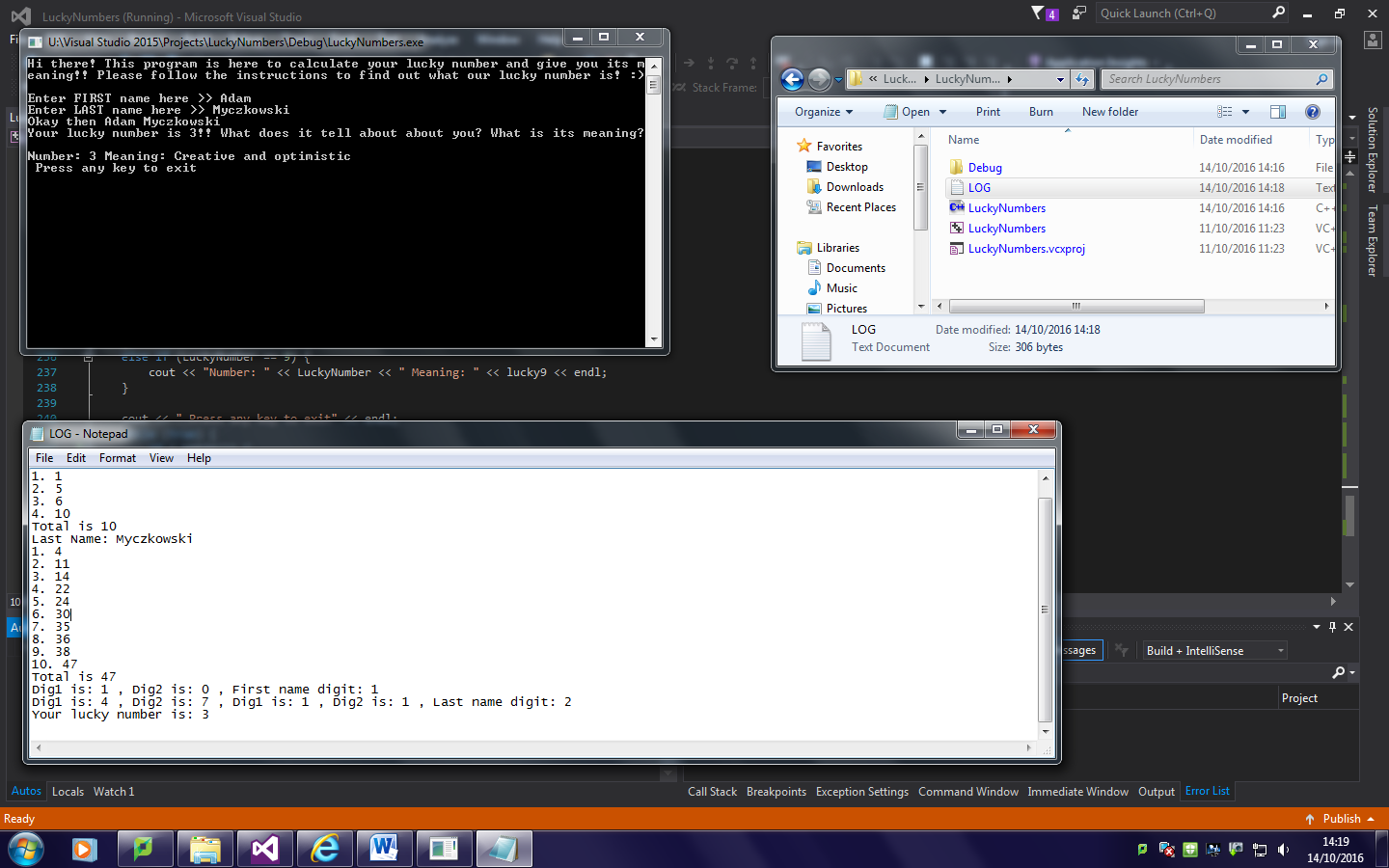


That’s the best way for user to read what’s on the screen and don’t have to type ‘yes’ or ‘ok’ or whatever to exit the program – function kbhit() hit from conio.h, the lib exclusive for windows. This function checks if keyboard is pressed and can be used without any comparison. When wrong character is pressed and when program gets to the end I have created a while loop that loops forever and checks if kbhit() (if the keyboard is pressed), when it is it simply calls return EXIT\_SUCESS or EXIT\_FAILURE.

### Version 1.5 – Log results of calculation onto a text file

For code look “Source Code 1.5” pdf file.

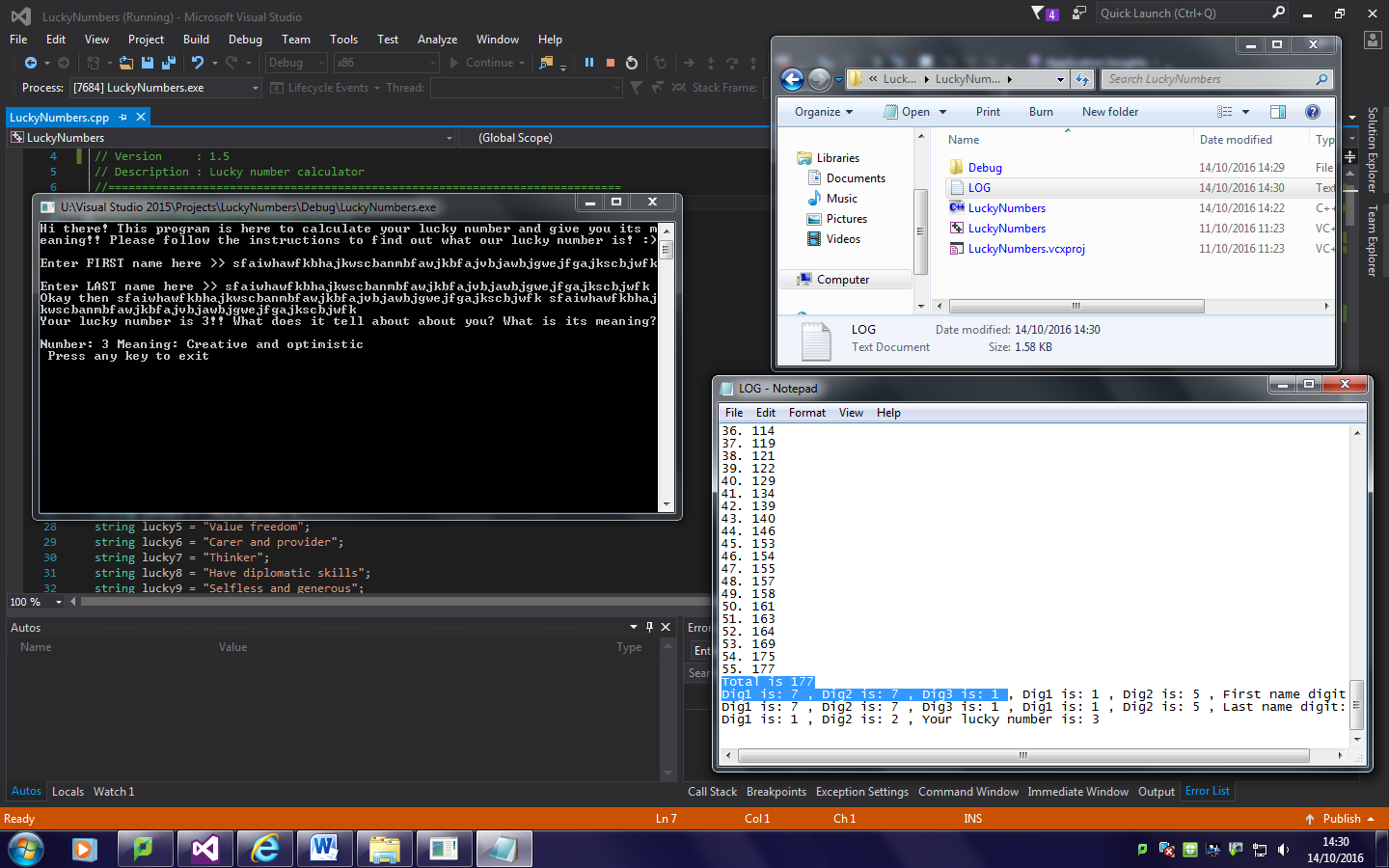


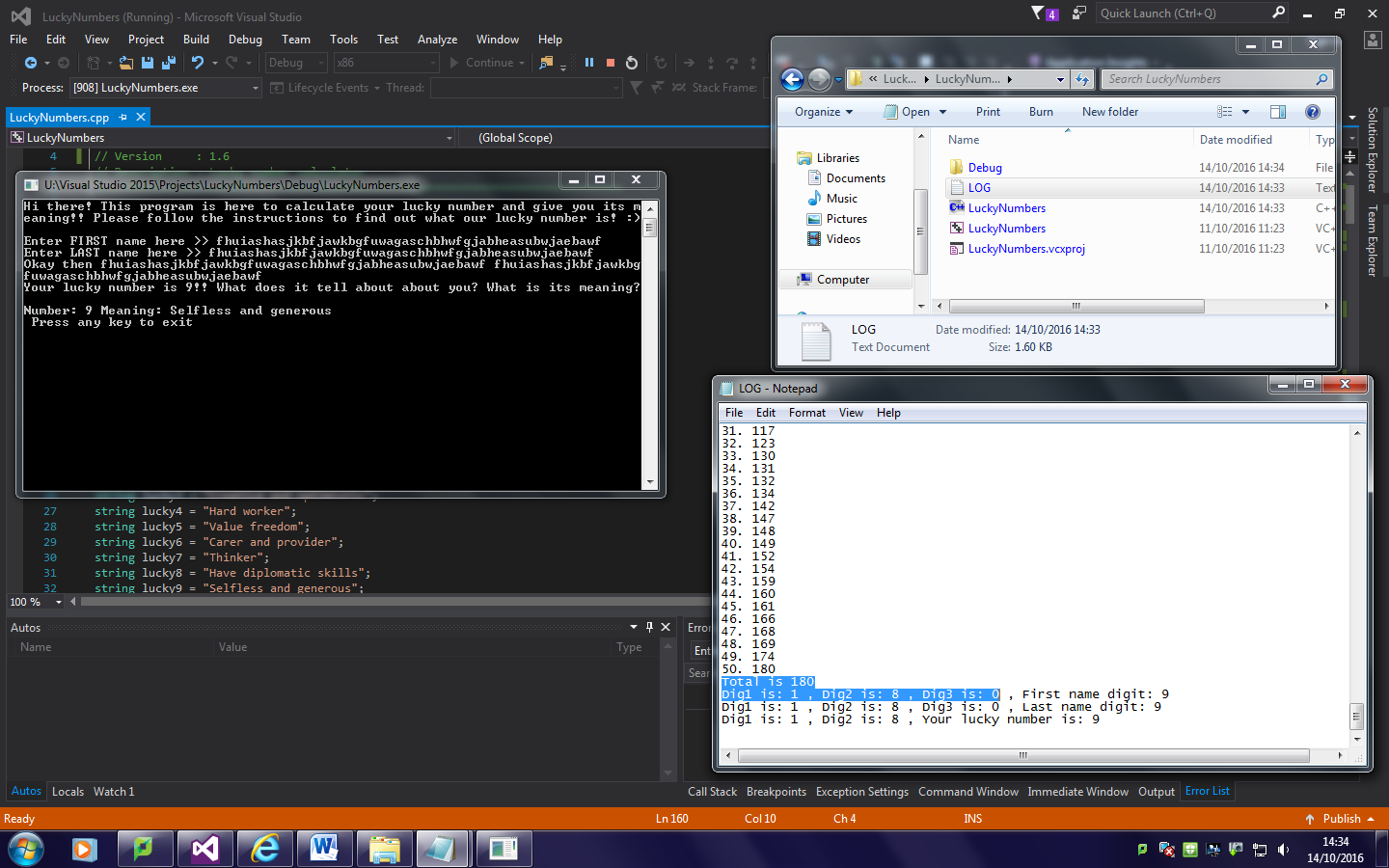


People might be curious what their names add up to etc. but some of them might not so there isn’t a point of printing info useless for some people. That’s why I decided to log onto a text file. I used ofstream function from fstream library. First of all it open or creates the text file called LOG.txt in the form that if the file exists and it’s being opened with some text written to it, it appends the text and writes new log on the bottom of the file. I logged first and second name and also their values as well as the single digits after splitting and the lucky number.

### Version 1.6 – Fix wrong order logging

For code look “Source Code 1.6” pdf file.





As I explained in the comment I noticed that when we have 3 digit number the order of logging is incorrect because when we have 2-digit number, dig1 would be the first digit and dig2 would be the second digit. However when we have 3-digit number dig1 is actually second digit not first, dig3 is the first digit. To have correct names and correct order I would need to add too much extra code so I just right shifted the order of the digits in log.

### Version 1.7 – Introduce switch()

For code look “Source Code 1.7 ” pdf file.

I wanted my code to be really basic at the beginning but fully working and I said to myself I will make it efficient once I sort all the bugs out. I did so it’s time to start implementing the improvements. I really didn’t like the big block of the ‘if’ statement so I had a look through stackoverflow and other C++ forums what can I use. I completely forgot about the switch – function similar to ‘if & else’ but apparently more efficient. You can’t tell the difference really because we talk about thousandths of a second but I decided to try it out. It actually may be more efficient because we don’t have to allocate memory for the alphabet char arrays.

### Version 1.8 – Drop switch() for find() function

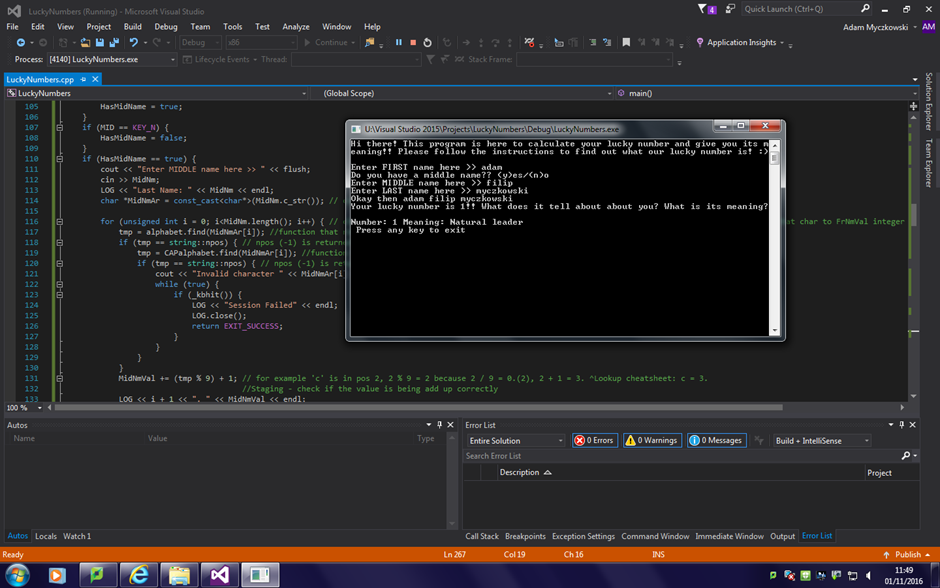
For code look “Source Code 1.8” pdf file.

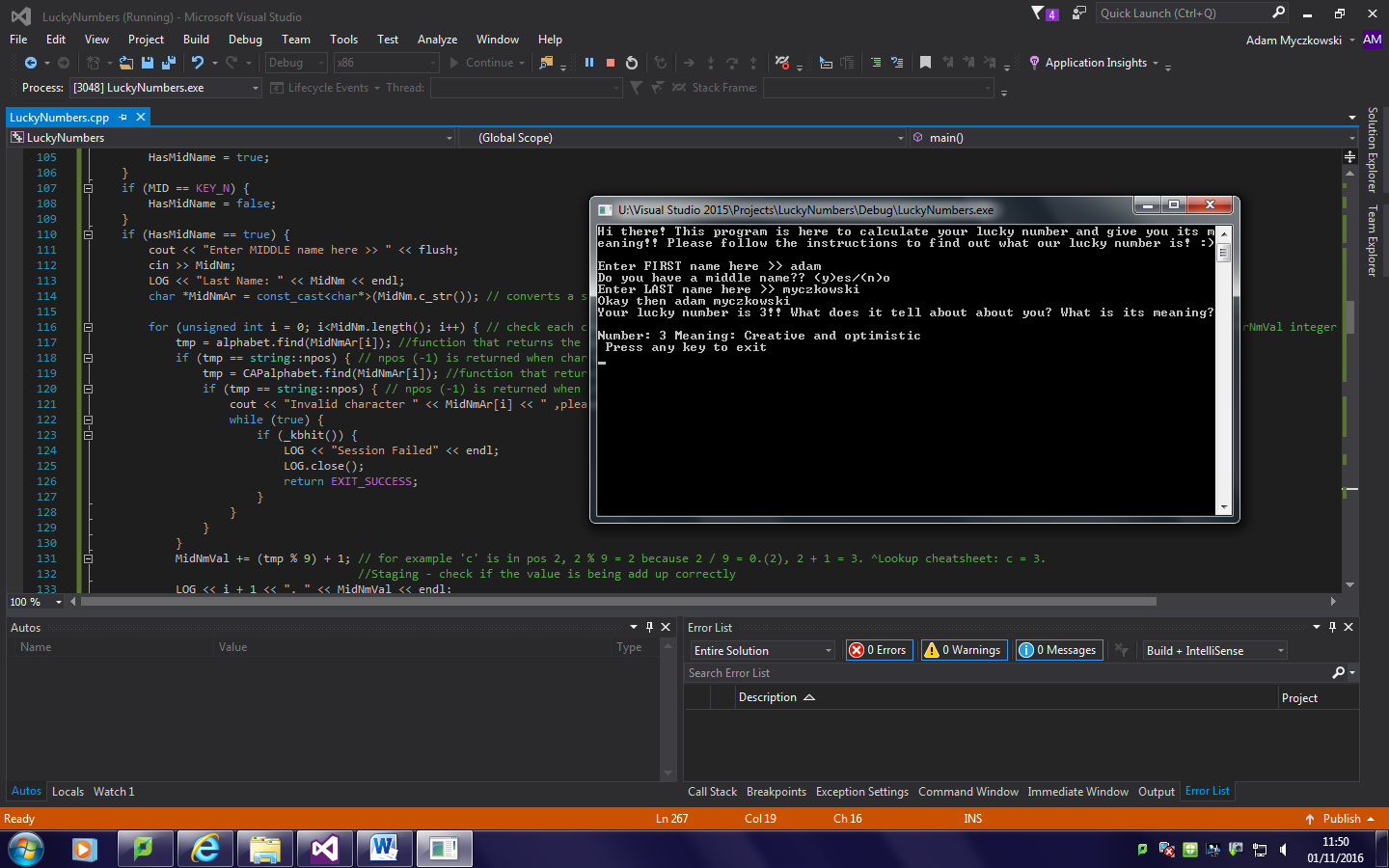
I felt like I could improve it even more so I asked a question on a C++ forum how I could improve it. Someone said that instead of the switch I could use indexOf() function so I was trying to implement it. That failed. It had something to do with the fact that I use names as char\*, I’m not quite sure, I do not fully understand the error. If I did I would use this function. However looking up indexOf I found another function called find() that can return position of w word or a character from a string. I have created two new alphabet strings instead of a char array I previously had. If character or a word is not in a string find() function returns -1 or a npos integer from string namespace.

I have created tmp integer variable in which results from find() function will be stored. First of all it checks if the character can be found in alphabet string, if yes it does the next function, if no it check CAPalphabet string. If it’s also not there program is being terminated. Once we’ve got the right position of the character in a string I do modulo 9 and then add 1 what always returns the value of the letter it equals to according to Pythagoras.

### Version 1.9 – Add option to use middle name and optimise memory allocation

For code look “Source Code 1.9 ” pdf file.



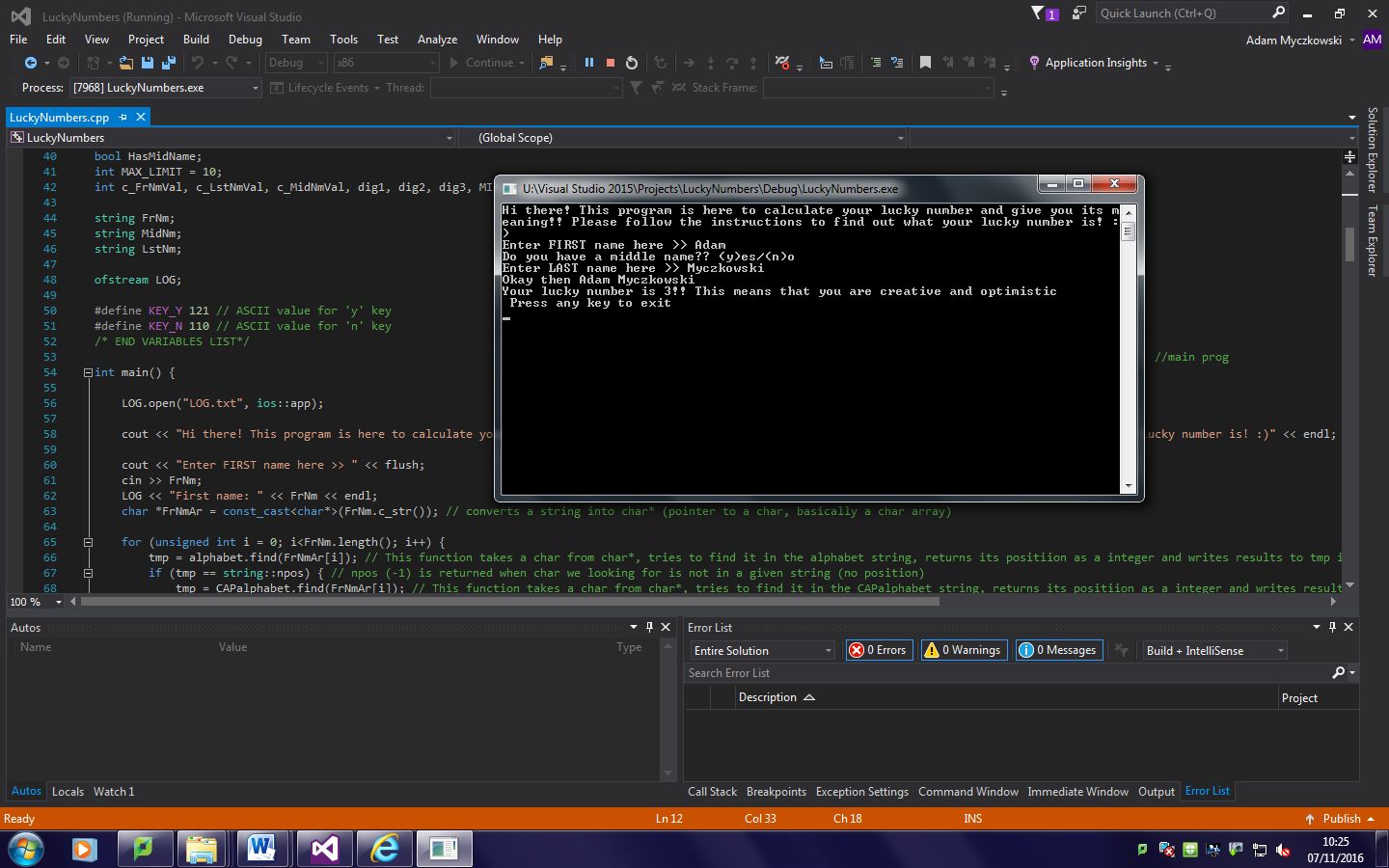


Using middle name in calculations is not required by the task but quite a few people I shown my program to have said it would be nice to have this feature so I decided to implement it. The function to calculate middle name’s total value is the same as for first and last name but it’s only being called when user choose to. I know nobody want to type and press enter so I used getch() function from conio.h library to check if ‘y’ for ‘yes’ or ‘n’ for ‘no’ button is pressed. The getch() function uses keys’ ASCII or Hex values for comparison. I defined ‘Y’ and ‘N’ keys in ASCII. I used loop to check what key is pressed, if neither ‘Y’ and ‘N’ are pressed it function prints on screen information that wrong key is pressed. When wrong button is pressed 10th time program just terminates. When ‘Y’ key is pressed function sets bool ‘hasMidName’ to true and so function to use middle name is called. If ‘N’ is pressed the bool is set to false and skips the function going straight to last name.

I also optimised memory allocation of the program – I realised that most of the variables could be declared on the top of the file and that few of them like ‘dig1’, ‘dig2’ and ‘dig3’ are used in few function in a while loops and it declares and allocates memory for them each time the loop iterates which is just a waste of memory and slows the program down. These variables are not static so each function has its own copy of it but at least they are not being re-declared each time the loop iterates.

### Version 2.0 – Rewrite the comments and make the way of how output is displayed more user-friendly

For code look “Source Code 2.0 ” pdf file.



I rewrote most of the comments because the old and new ones mixed up and it didn’t really make sense. I also changed the way lucky number and the meanings are output so it’s easier to read for an end user now.

### Version 2.1 – A little bit more optimisations

For code look “Source Code 2.1” pdf file.

‘If’ statement that check what key is pressed when waiting for input whether you want to use middle name or not were linked so I used ‘else if’ instead in the second statement. As these statements are linked now this reduces memory allocation.

I realised copies of FrNnVal, LstNmVal and MidNmVal are actually useless because original variables are not used and/or changed by any other function what means there is no point of creating copies of those. I deleted the copies and made functions that calculate lucky number use the original values. This speeds the program up because less memory needs to be allocated.

The last lines of code:

LOG.close();

return 0;

They are useless because the program never gets there – process is being terminated in function above those lines.

### Version 2.2 – Change the way int is separated into single digts and declare whole array at once

For code look “Source Code 2.2” pdf file.

Instead of performing calculation parse LuckyNumber to string that then can be converted to char\* so that we have access to all indexes in this array. Char can be easily converted to int with " - '0'". This should increase performance of the application.

Declare the whole array at once instead of declaring each string on its own, this should increase performance of the application.

### Version 2.3 – Optimise the output

For code look “Source Code 2.3” pdf file.

As I declared meaning as one array I can use indexes to navigate in display the meanings – Instead of the block of if statements that check for the value of LuckyNumber, I replaced it with one line that that displays the index of lucky[] array using LuckyNumber-1 as arrays always start with 0.

## Problems:

My program is working well but when I tried to show my friends what I have done on a different machine it wouldn’t run because of dynamic libraries missing the system (.dll). Visual Studio C++ dll’s are only installed on my computer on the disk – I was told that it’s the only way I can get access on to Visual Studio on a school computer. As they are installed on my computer’s hard drive and not on a school’s server like other apps they are only accessible by the person that logs onto the computer I use it won’t run anywhere else. I didn’t like that so I attempted to fix it. I read in the internet that if the required libraries are in the same directory as the .exe file it will return an error that the libraries cannot be loaded properly and I have to write a manifest file that tells program which libraries to load. I couldn’t do it. I couldn’t test it if the library load error actually occurs because it was a weekend. I tried the other way – the LoadLibrary() function. I implemented and tested it at home but then I realised it’s impossible that it will run from a simple reason the libraries would be loaded on runtime but I need them to start the app. If it can’t start it won’t load them so that was a really bad idea. The solution was way easier than I thought - I had to download msvcp140.dll and vcruntime140.dll and put them in the same directory as the program, modules were dynamically loaded and therefor it worked.

Known issues and how to solve them:

Depends on what users has installed on their computers they might get ‘missing library’ error, I attached two most common VC++ specific libraries with the program that should make it run but if you are still facing errors please make sure you have Visual C++ Redistributable installed on your computer. Needed libraries can be downloaded from [Microsoft’s website](https://www.microsoft.com/en-gb/download/details.aspx?id=48145).

# Postproduction Testing

Besides testing and fixing errors on the go while coding I have to fully test the final version as well.

Look “Testing” in my Appendix.

Analysis of testing:

Results of my testing show that my program exactly as I expect it to work.

# Evaluation

Success criteria (program must):

**Problems 1:** Input a person’s name - Yes

**Problems 2:** Calculate their lucky name number using Pythagoras’ method - Yes

**Problems 3:** Display the names - Yes

**Problems 4:** Display the lucky name number - Yes

**Problems 5:** Display the meaning of their lucky name number – Yes

**Solution 1:** My program is able to correctly get input, store and use it across the program. Original name variable is a simple *string*, and then it is being converted to different type variables as needed.

**Solution 2:** My program converts name *string* into *char\** (array), loops through the characters of the name, and calculates it number using ***FrNmVal += (tmp % 9) + 1;*** where tmp is the index number of currently loaded character in the alphabet. Total number for a name is then being split into single digit integers by converting the *int* into a *string* and then into *char\**. Split numbers can be easily added together as integers if we do *-‘0’* on the *char*. This is repeated until we have a single digit.

**Solution 3:** My program correctly displays the names just before the lucky number with:

*cout << "Okay then " << FrNm << " " << LstNm << endl;*

**Solution 4/5:** My program correctly displays the lucky number and its meaning by:

*cout << "Your lucky number is " << LuckyNumber << "!! This means that you " << lucky[LuckyNumber-1] << endl;*

To display meaning we need to subtract 1 from *LuckyNumber* because arrays start with 0 not 1.

# How could I improve my program:

To improve my program I could completely redesign the log layout because it’s quite messy and for a user might be quite hard to read the information given.

To provide better user experience I could make a GUI for the app. Some of the users might be quite confused with the layout of instruction within CommandPrompt on Windows. In order to achieve it I could create visual elements using SDL2 library that allows you to use images, sounds, set pixels, draw on screen etc. Though it’s a really good practice in takes long time and I wouldn’t be able to complete my solution within the given time. Alternatively I could create C# Forms app with graphical interface but I prefer C++ over it.

Thinking about compatibility, I could rewrite and/or optimise the program for more platforms such Linux, Mac or Android. My program does not support Windows 10 as we have Windows 7 on the school computers and Visual Studio SDK for Windows 7 only supports Windows 7 and 8/8.1. If we had Windows 10 I could run it on all Windows platform versions because I can tweak it to be backwards compatible to Windows 7. Unfortunately it does not work the other way around so I am not able to improve my program unless I decide to upload and maintain opensource version of this code after examination what would allow me to add more tweaks, make it more efficient etc.

## Maintainability:

My program is easy to maintain, support for more names and bigger numbers can be easily added.

## Reliability:

My program is very reliable in terms of the input interpretation. I implemented error handling that catches and displays all the errors and most importantly cannot be crashed or made to fall into infinite loop. Everything is carefully checked under my criteria. In terms on the works flow – not quite as it requires few libraries to be installed, I provided the needed dynamic libraries within the program what should allow it to run even without Visual Studio C++ Redistributable installed. If you have any issues on runtime regards to missing libs please install Visual Studio C++ Redistributable and if issue still occurs contact me via email at [mycaxd511@gmail.com](mailto:mycaxd511@gmail.com)

## Performance:

My program is very efficient; I tried my best to reduce the number of lines of code and to use the lowest number of calculations possible.

## Further improvements:

I changed a lot regards to my initial algorithm and flowcharts. I found more efficient way of adding a number for a letter and to get a single digit out of 2-or-more digit integer. I also added LOG, initially for debugging to see what happens without spamming the screen with messages useless for the end user but it seems interesting enough to keep it as a LOG. Few of my friends asked an important question – What happens when you have a middle name? So I decided to implement it as an option.