TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

Mihkel-Eduard Luude 192924MVEB
Yehor Karpichev 195062MVEB

CIPHER

Software Project

Supervisor: Vladimir Viies

Tallinn 2020

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. All works and major viewpoints of the other authors, data from other sources of literature and elsewhere used for writing this paper have been referenced.

Authors: Mihkel-Eduard Luude & Yehor Karpichev

TABLE OF CONTENT

SHORT DESCRIPTION

Quick facts:

- 5 ciphers to encrypt or decrypt the message
- Written on C++
- Developed basic functional user interface

The **CIPHER** is a software project developed by two young engineering students with passion for information technology. After passing the basic programming courses in C, we felt interested in C++ so this project was a perfect opportunity to challenge ourselves and learn.

The program contains five different ciphers: caesar, binary, numbers, vigenere and morse. Each of the ciphers is interesting in its own way. Some of those are customizable. Some are made in a way that even the developer doesn't know exactly how the data is encrypted. Eventually, the simple user interface was created to make the program more user friendly.

MOTIVATION AND DIVISION OF WORK

Why CIPHER?

The initial brainstorming - Ideas and their description:

- Universal Area and Volume Calculator
    - Abel to calculate area or volume of a surface or body
    - Generate the shape of the surface or body
    - Give user a simple way to calculate and see the shape of the area or body
    - Include integration and path length
- Ancient calendar
    - Convert any day into an ancient callender
    - Display the result of the date
    - Include the weekday in the naitive language
- Cipher
    - Use different ciphers to encrypt and decrypt the data
    - User is able to cipher and decipher text
    - User simply writes the text which then is ciphered by the program
    - Implement the random factor, for example the cipher could be generated randomly.
- Matrix Solver
- City searcher
    - User enter a city's name/any name
    - Program searches through a list of cities -> If there exist that city then program display basic information about it (Location, population, contact information, global coordinates, etc)
    - Include more than one countries
    - Include distance calculator between cities

Eventually, we stopped on the cipher idea as it was a good suit for both of us (particularly our interest) and also because it is an important topic (the aspects of privacy in the modern digital world and the methods on how to keep data safe are extremely valuable).

Our project doesn’t go deep in the methods of data encryption, however, the produced code could be considered as the prototype for the further development into the real program. Meanwhile, we based most of the decisions about this project on our interest and opportunity to advance the knowledge in programming.

Division of work

The project was fully developed by two people: Mihkel-Eduard Luude and Yehor Karpichev

Mihkel

- Vigenere Cipher
- Numbers Cipher
- User Interface (UI)

Yehor

- Caesar Cipher
- Binary Cipher
- Morse Code

After the decision was made to go with ciphers, each of us brought two ciphering ideas: Mihkel - Vigenere and Numbers, Yehor - Caesar and Binary. Later on Yehor also decided to add the Morse code.

As for the UI, it’s been a bit more complicated. It’s developed by Mihkel, however, more about why and how in the parts “MAIN DIFFICULTIES and DETAILED DESCRIPTION”

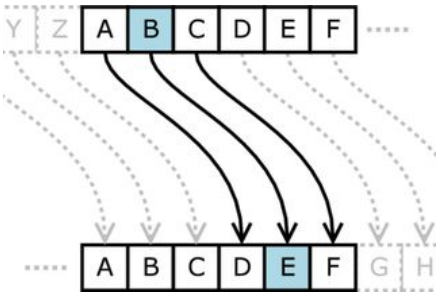
DETAILED DESCRIPTION

In this section, you will find a detailed description of each code based part of the project, in particular it talks about the ciphers (what are they and how are they handled by the program) and the developed user interface.

NB! The detailed description goes through all ciphers considered as the separate code parts. The ciphers codes that are incorporated into the “final project” (which is the integral part of GUI) is different.

Caesar Cipher

(by Yehor)
On the picture on the right , you may find the general principle of what a Caesar cipher is and how it works. In fact, it is quite simple: it creates a shift in the letters. For example, in english language, there are 26 letters. If you enter ‘A’ and you want the maximum shift (the maximum shift is 25 because A is already 1), the letter will turn to be ‘Z’.



In the code, there is an opportunity to both encrypt or decrypt the data. Due to the use of a <vector> container it allows the user to enter anything: text, numbers, signs, spaces. It’s able to recognize that if a user entered the letter, then this letter must be changed according to the key.

For example, if a user enters “my share is 25%!” as input and lets the key be 5, then the program output would be: “rd xmfwj nx 25%!”. Every letter is moved 5 letters ahad, meanwhile the spaces and other signs stay the same.

The user may enter any positive integer for the key value. Eventually, if the entered key is bigger than 26 - in such case the 26 is subtracted from the key until its value is equal or less than 26.

The only limitation for this cipher is that it cannot produce an absolutely identical message in terms of letters’ case. The program output will always be the lower-case, even if the user enters capital letters. It is so due to the fact that in the code each alphabet letter is given a unique pin ranging from 1 to 26. Checking if the letter is capital or not would make a program much more complicated. However, on the other hand, it is an opportunity for further development.

Vigenere Cipher - Mihkel

(by Mihkel)
Vigenere Cipher is basically two Alphabets put together in a matrix, where one is the top row and first column, like shown in the picture on the right. It was first described by Giovan Battista Bellaso in 1553 and was only first successfully and repeatedly DeCiphered in 1863, over 300 years, by Friedrich Kasiski, but was misattributed to Blaise de Vigenere, thus the name.
The Cipher consists of two parts, the plain text and a keyword. The person writes the text as normal, then chooses a keyword that gets “projected” over the whole text. Then you find the square where your letter from the plain text crosses from your letter from the keyword and that square is your new letter (Shown down below).

Plaintext:	`THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG`
Key:	`LIONLIONLIONLIONLIONLIONLIONLIONLIO`
Ciphertext:	`EPSDFQQXMZCJYNCKUCACDWJRCBVRWINLOWU`

The way I looked at it from the beginning was that if I were to assign every letter numerical value and add it into a matrix, then every combination square would have a unique value, of course for every combination would have two values, a lower one and higher one, I don't need the higher one so in the code if the value exceeds 27 equals or exceeds then I take 27 off.

The way I wrote the code is that I ask the user to write his/her text then chooses a keyword. Then I make sure that the key is smaller than the text otherwise the cipher will not work. Then I convert the letters into number ids, because adding is a lot easier with numbers than letters. Then I make the keyword array as long as the entered text one, both are numerical arrays, then add them together and convert the new array into letters and the text is printed out into a .txt file.

DeCiphering works quite similarly. The difference is that instead of taking the text from user input it is taken from a text file, technically meaning if you have a Vigenere cipher from somewhere else then you can use this program to DeCipher that text, then the program asks for the user to enter the keyword. Then the same operations are performed that were done in the Cipher part, the difference being that instead of adding subtracting is done. Finally the DeCiphered text is printed out in a text file.

Keep in mind the way I described how this program works is different from how the program operates in the full project since in the full project the input is gotten from GUI and then transferred into either the Ciphering or DeCiphering part, while the output stays the same.

Numbers Cipher - Mihkel

(by Mihkel)

Numbers Cipher is a cipher which I came up with on my own. I was trying to think of a way to create a randomly generated cipher. I had many ideas, like creating an id library where the user would choose a predefined id, that would give letters certain values, but that seemed too complicated and rigid. I considered using rand(), but I had no good idea how to create a secure or good DeCipher for it. Then I had an idea.

I thought about generating an array with 27 values, but I felt that would be too static, if every letter had only one value, but what about if I generate a long array and instead of using array values I would use that array places, meaning the values would be random and harder to DeCipher.

The code works soo. First user is asked to enter a number and a coefficient and of course the text that the user wishes to Cipher. After that the array is generated by the following function.

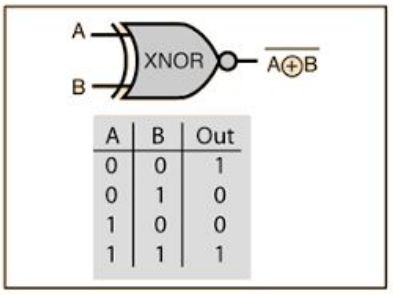
```
void UserArray(numbers* numbers)
{
    numbers->UserArray[0]= numbers->Unumber + numbers->UCoefficient + 1;
    for(int i=1; i<100; i++)
    {
        numbers->UserArray[i]= numbers->UserArray[i-1] + numbers->UCoefficient + 1;
    }
```

This function is very simple and generated array values are simple as well, but if I was to continue developing this cipher then this generation can be made a lot more complex. Then to add both security and customization then next the user could choose what places from the Cipher he/she wants to use: Even - program chooses every even placed array member, Odd - program chooses every odd place in the generated array or Random - program chooses random places I selected. In an ideal world I would allow the user to select exactly what places they want, but since I had no idea how to automate it then this makes this part of the cipher static.

Again in the whole project the program works differently then when I design it, instead of all asking for the input it gets it from the GUI and also the choice of the places is made from GUI.

Binary Cipher

(by Yehor)



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

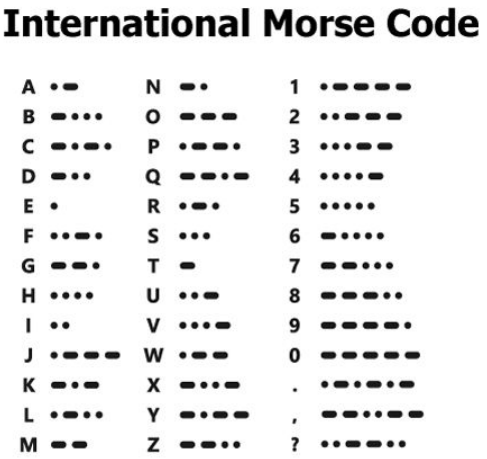
I decided to call this cipher the ‘binary’ as it operates with zeros and ones. However, it is based on the XNOR type of the digital logic date. On the right, there is a picture with the truth table for XNOR.
However, first of all, how does it work as the cipher? In the beginning the user enters text. Then, I use a table where I transform letters into the numbers (basically each letter is assigned a unique binary code). If we speak of the picture on the right, then this unique binary code would be what is in the ‘out’ column.

Then, the random array for each letter is generated (it is let’s say an ‘A’ on the picture). After that, the ‘Key’ array is calculated by the initial value and randomly generated array (the key is column ‘B’). As you perhaps already understood, each time the code runs the same message - the ciphered text would be different (due to the fact that there is every tie randomly generated array). The cipher is made so that the Key file or the Random file by itself won’t make any sense -> it is nothing but a list of zeros and ones. But if you combine the Key and the Random (remember you need to have both files) and compare them according to the truth table of XNOR, then we get the binary code that is associated with letters -> therefore, the message can be finally decrypted.

In terms of advantages, it is that it’s really hard to crack it without knowing that it is using XNOR or that for example, each letter consists of 5 binary digits. Furthermore, it becomes almost impossible to hack the cipher without having both (key and random) files and their content.As mentioned, there is a random factor, in other words, even the creator doesn’t know what exact digits are generated every time for every character. As for the weaknesses, the cipher can take all english alphabet letters, also it takes a dot and a coma. All other symbols, signs or spaces will be treated the same way - it all will be seen by program as the space. Another more important weakness (which I unfortunately thought about very late when there was no time to fix it) is that the program stops working when the user wants to decrypt, however, the key.txt and random.txt do not exist (BUT this problem exists only in the project where it’s a part with other ciphers and the GUI, there is no such problem in the file of this cipher taken separately).

Morse Code

(by Yehor)



The Morse “cipher” often referred to as code is simple and straightforward. Technically, It doesn’t have any variations. Each letter is assigned a mix of dots and dashes, which do not change.

On the image on the right you can see the script that was taken as the code for the morse cipher in our project. So, immediately it’s clear that this cipher supports limited signs. In other words, it works only if the user enters English alphabet letters, digits, dot, comma and the question mark. I also made sure that the program doesn’t break if the user would like to enter more than just a single word (so the program takes the space into account and in the output also produces the space in the identical place).

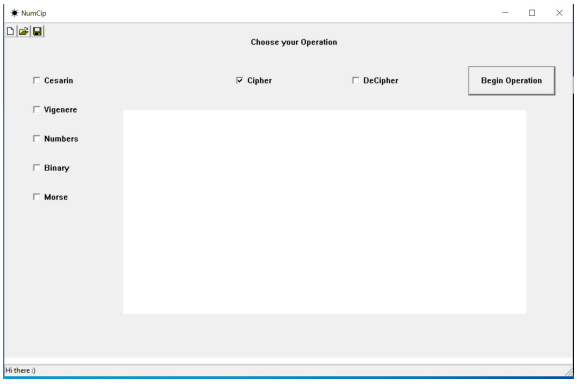
The most interesting “feature” of the morse code implemented in this project is that there is an option also to listen to the morse code (not just to see the encrypted data).

Nonetheless, the written program has a disadvantage. In particular, the morse code works only for the capitalized letters. So, if the user wishes to enter the text to transform it into the morse code, the user must enter the capitalized letters. It is so because in this program I tried to implement the solution by using the <map> attribute of C++ and it allows to hold two values associated with each other only, the repetition can not exist. However, similarly to Caesar cipher, it might be something to work on in future.

Another weakness (however, it’s an arguable weakness) is that the program can only encrypt the data. There is no opportunity to transform the morse code back to the letters. The reason for it is very simple. Let’s take the letter “P” (refer to the image) and the combination of “A” “T” “E”. As you may see in the morse code both options look exactly the same!

Graphical User Interface (GUI)

(by Mihkel)



In general design and implementing a GUI in c++ is both difficult and complicated. From the start of the project we knew we wanted to design a UI, with the inexperience that we have in the field we thought that we could design it Visual Studios, but when we were 2 months into development when we tried we discovered that designing anything in Visual studios is difficult and beyond our skill. For a time we considered QT or other programs, but we had linking problems so in the end we decided that I would make it in Windows.h, which turned out to be a mistake, because it is hard to find tutorials for it and even harder to implement those with our code, but with a weeks worth of coding I was finally create the GUI in the picture, of course I was not able to implement everything that I wanted or even how i wanted, but it seems do to the job well enough.

In general it is a pretty simple program. There is a big main edit box, where the user can write the text. At the top the user can choose whether they want to perform Cipher or DeCipher operations, keep in mind while you can perform DeCipher operations at the start, but unless you have some text the operations will do nothing.
On the side the user can select the type of Cipher they want, if the cipher requires more input like a key then above the main edit box a smaller edit box will appear and at the bottom extra options will appear at the bottom.

To begging any operation the user will have to press the “Beginning Operations” button, before the button is pressed the user can edit any input entered also the user does not have to rebuild or reload the program he can change any factor and just press the button again to generate more Ciphered text.

MAIN DIFFICULTIES

1. Ciphering

Ciphering is in fact a generalization of the problem In fact, we had quite a difficulty understanding and then implementing in the code how to quickly go between numbers and letters. All our ciphers (except Morse code) deal at the same time with both letters and numbers in one or another way. In every code we had to transfer letters to numbers and vise versa. Finding how it can be done in a more efficient way was the main issue. Eventually, we ended up in most of the codes using the simple if / else if / else statements. However, in the morse code an interesting solution was found. Yehor applied the <map> container of C++. It certainly has benefits because it allows you for example a particular letter (or word or character) to be associated with a particular number (or something else). As the codes were not written simultaneously (the Morse code was developed just a week before the submission) this solution wasn't implemented in other ciphers.

2. User Interface (both, but mostly Mihkel)

The user interface was a true issue as no one from the team had any previous experience or understanding how to do the user interface. Furthermore, as the project was based on C++, the only option we saw was to use the built-in features of C++ to make a simple but usable UI. We have conducted the consultations with more experienced people a few times. For example, one of the discovered applications that could help in UI development was "Qt". Both of us spent around 2 weeks trying to understand how this program functions and if it's possible to apply it to our project. Eventually, we came to the conclusion that this program is great, however, we at that movement didn't understand how to connect the UI code to the Cipher's (back-end) code + it was already November. So, due to uncertainty and time limit the decision was made to proceed with simple built-in features of C++. In the team, we agreed that Mihkel can fully take the responsibility to develop UI, meanwhile Yehor can focus on Morse code and other aspects of the project.

3. Sticking to the timeline (both)

According to the timeline (which was established within the first two weeks of the project's start) we planned to finish the project in the last week of November 2020. However, at the moment, it turned out that we finalize and submit the project in the mid of December 2020. Sticking to the timeline and the time management in general is one of the biggest problems nowadays. Even though we didn't manage to finish everything absolutely by the timeline, it may be stated that overall, we did a good job. Certain delays are probable, especially in the projects like CIPHER, since a lot of things had to be studied on our own. It is a learning project and throughout the development a few times we faced issues that we didn't even think about in the beginning. Therefore, it takes time to find the right solution. Nonetheless, every week there was at least a bit of communication between Mihkel and Yehor to discuss who is doing what within each week.

4. Binary Cipher (Yehor)

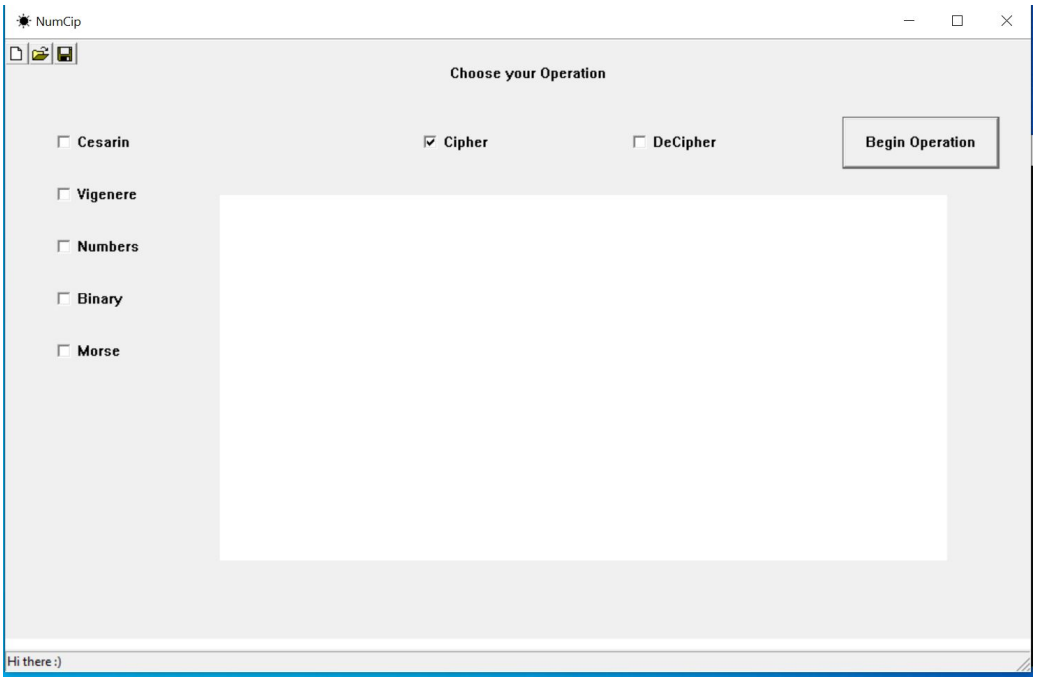
Writing the code and solving how to deal with the binary cipher was a true challenge to me (Yehor). The main difficulties were transforming digits to string and vise versa, and generating a random array of digits. So, the first problem was that user input letters had to be transformed into the digits. But if we speak of integer type -> the first digit of the number can be any except zero. So, for example A gives a binary code of 00001, but if 00001 is considered as an integer then the program reads it just as 1. The program always removes all zeros in front. This was a big issue. The solution? I made “A” equal to the string “00001” and then found a way how to transform the string “00001” to the array of five digits (e.g. arr[5] ={0, 0, 0, 0, 1}). Eventually, I needed numbers in order to generate the random array -> that’s the 2nd difficulty. Actually, it’s not so difficult but my problem was that I needed to generate a new random array for each letter. IN the beginning when I ran the program it was generating always a new array for the first letter but then it was repeating itself all over again. So, the problem was that I needed to nullify the ‘time’. It might sound simple, however, it’s not easy when you don’t know what exactly is wrong and you didn’t have such previous experience.

5. Linking different C++ files together (both)

Overall, linking wasn’t a great issue, however, it is really something that made both of us think a lot. We realized that it is pointless to have all function for all our ciphers in one single file. So, we divided it on many cpp files. However, how do you connect them afterwards? Well, we solved this problem and actually found two different solutions. Mihkel applied “extern” as the type to his function in order to transfer the specific function to separate files. Meanwhile, Yehor looked at it from a somewhat different perspective. Yehor applied a similar logic when we include the “.h” files. To the main executable file the other .cpp file must be included in the beginning. Also, within the prototypes the desired function must be declared. Then, in your other cpp file, just write the name of the function and it will be executed. You may find more about how exactly it works in the submitted code files. Nonetheless, in the end when we tried to put all the codes together with GUI -> the decision was made that it’s easier and ‘cleaner’ to use the extern.

FINAL OUTCOME

As the final outcome we can speak of developing 5 different ciphers: each with interesting features. For example, Binary code is exciting as it always generates a random array for each letter and then if you don't have one of the files (either Key or Random) - you can't decrypt the data. The Numbers is amazing in its own way - it has so many different variations that it makes the cipher very unpredictable. Or let's take the Morse code. It is simple, however, we found the way to produce the sound! Yes, preciously, it outputs the encryption in the form of the morse code but it also then produces the short-long beeps corresponding to the dot or the dash.



On the other hand, we must remember that this project wasn't just all about writing the code. The project was also a lot about teamwork, trust to each other, sharing the workload and helping each other. Eventually, we had to face many issues that had to be solved, hundreds of errors - we found the way to fix it all. Developing the CIPHER project was an amazing learning experience where we went from the stage of idea brainstorming to the stage of making it real.

IDEAS FOR FURTHER DEVELOPMENT

1. **Educational purposes**
    a. (ethical) hacking application
        i. The application in future could be used in the education purpose in order to introduce the beginners into the world of hacking. We fully understand that hacking in the modern world is a double sided coin. However, we also must admit that in the digitalised era it is crucial to have people that must be able to test the defence IT systems and find bugs that need to be fixed. So, we believe that we could use the current prototype to make an application that creates a cipher and gives a trainee a task to hack it in the given time. As mentioned before this path could be chosen only for the education purpose.
    b. Thesis opportunity
        i. A lot of work has been done throughout the project. Eventually, it could be a basement for further research. For example, the research could go into the theoretical background of the data encryption or data transfer.

2. **Data Encryption**
    a. Encrypted communication application
        i. Data protection always was a sensitive subject. Nowadays, its importance has risen much higher than ever before. We believe the current prototype could be developed into the desktop or mobile application that uses combined ciphers in order to transfer the content of messages between the users.
    b. Notes application
        i. Similarly, to the previous sub-bullet point - it’s the opportunity to encrypt the content of your notes. Imagine having the notes application. You want to keep your personal notes personal. So, there could be an app that uses simple encryption methods to make your notes readable only to the owner.

LIST OF REFERENCES

The link to our criteria document: LINK
The link to initially established timeline: LINK

Yehor Karpichev and Mihkel-Eduard Luude are the authors for the whole code submitted in this project, however, below you may find the list of sources that were most often used to solve one or another problem throughout the development stages.

- https://stackoverflow.com/
- https://www.geeksforgeeks.org/
- https://www.cplusplus.com/
- https://www.youtube.com/