Assignment Exercise 2

UFCFK4-30-2 C++ Development

Introduction

This assignment focuses on simulating a cipher machine called the Lorenz SZ40 (SZ is derived from the word *Schlüsselzusatz* meaning cipher attachment) used by Adolf Hitler in the Second World War for encrypting important messages such as battle plans. The machine was thought to be unbreakable by Hitler and in fact the Russians used the machine for several years following the war for the same reason. However, some very clever men at Bletchley Park managed to crack the cipher without ever seeing the machine and in the process created one of the first "computers" called Colossus.

Task

This task involves developing a program to simulate the Lorenz SZ40 machine. In particular, you will need to read the pin settings from the provided *pinsettings.dat* file into memory. Appendix A gives details of how to do this. The data file provides the 501 pin settings for all 12 wheels in the Lorenz Machine. You should use these settings throughout this task. Below is a table showing how many pins there are on each wheel. You can assume that the provided pin settings match the order of the wheels in this table, e.g. the first 43 characters in the data file represent the pin settings for the Ψ_1 wheel and so on.

Wheel Number	1	2	3	4	5	6	7	8	9	10	11	12
Name	Ψ_1	Ψ_2	Ψ_3	Ψ_4	Ψ_5	M_{37}	M_{61}	X_1	X_2	X_3	X_4	X_5
Pins	43	47	51	53	59	37	61	41	31	29	26	23

For the purposes of this assignment you can assume all wheels initial settings are position pin 1.

Instructions

1. Research the Lorenz SZ40 machine
    a. Write a short introduction to the machine which should include some historical context (250-300 words).
    b. Provide a description of how the machine works which should include a short overview of Baudot-Murray ITA2 code and how this is used in the machine (250-300 words).

[20 marks]

2. Write a pseudo-code description of how the machine works.

[10 marks]

3. Implement the Lorenz SZ40 machine in C++. Suggested classes to develop include:
    a. `Baudot` class that provides a facility for converting between ASCII and Baudot-Murray ITA2 code. *NOTE: there are several non-printable codes – these should be replaced with the following printable characters: carriage return (,), line feed (-), letter shift(.), figure shift(!) and null (*). And, you do not need to implement the figure shift or letter shift for this assignment.*
    b. `LorenzWheel` class. This class should store the pin-settings for the wheel and its current position as well as providing a capability for rotating the wheel.
    c. `LorenzWheelCollection` class to hold groups of wheels such as Ψ and X wheels.
    d. `LorenzMachine` class to provide the main simulation. You should be able to both encrypt and decrypt a message using this class.

[50 marks]

4. Encrypt the following messages - resetting the machine after each message:
    a. **THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG**
    b. **A MAN PROVIDED WITH PAPER PENCIL AND RUBBER AND SUBJECT TO STRICT DISCIPLINE IS IN EFFECT A UNIVERSAL MACHINE**

[10 marks]

5. Reset the machine so that each wheel as at position 1 before each decryption. Decrypt the following messages:
    a. **BVJM,*V*TMAK*ZMK-XHGDVEMGOZUER .XWNIASN NCOVJ,ABELCLOUG,AD*OBRFLJXH! APZ*HJFHG.CQJRB**
    b. **KIK!F*AOYTUUU,VGBDQKDP*GPVAZMROTX,VCF!C AFS NUARKLOGISGU*FBG!EBLQ-H* LX*K**

[10 marks]

Components to be submitted

1. A report detailing the Lorenz machine, how it works and what Baudot-Murray ITA2 code is.
2. Pseudo-code description of the machine.
3. A C++ implementation of the Lorenz SZ40 Machine.
4. The encrypted and decrypted messages detailed in part 4 and 5.

Submission should be made through the Assignment section of Blackboard.

Please make sure that any code submitted compiles and runs inside of Visual Studio 2010, and that any documents are in .doc, .docx or .pdf format. You should add both the report and Visual Studio 2010 project to a single **.zip** file before submitting to Blackboard.

Assessment

You will be assessed on the following criteria:

1. Your code is properly commented and works.
2. Your adherence to the specification.
3. Correctly encrypting and decrypting the provided messages.
4. A coherent and well-presented report.

Appendix A

The following code should be used to read the contents of the *pinsettings.dat* file into memory. The *pinsettings* string will contain the 501 pin settings.

```
string pinSettings;
ifstream infile;
infile.open("pinsettings.dat");
getline(infile,pinSettings);
infile.close();
```

NOTE: you must include the string and fstream libraries in your .cpp/.h file:

```
#include <fstream>
#include <string>
```