AST, WS 2010/11 Übung 03 Enigma

Name	Matrikelnummer	Gruppe	Punkte

Abgabetermin: 13.12.2010

Enigma Verschlüsselungsmaschine

Die *Enigma* ist eine Rotor-Chiffriermaschine, die im Zweiten Weltkrieg im Nachrichtenverkehr des deutschen Militärs verwendet wurde. Mit ihr wurde der größte Teil der Funksprüche der deutschen Wehrmacht und Marine verschlüsselt. Als Erfinder der *Enigma* gilt der deutsche Elektroingenieur *Arthur Scherbius* (1878–1929), dessen erstes Patent hierzu vom 23. Februar 1918 stammt. Das Wort *Enigma* kommt aus dem Griechischen und bedeutet „Rätsel“.

Abbildung 1: die Enigma Chiffriermaschine.

Aufbau

Die Enigma war eine elektro-mechanische Chiffriermaschine. Sie bestand im Wesentlichen aus drei Bauteilen: über eine *Tastatur* wurde der Klartext bzw. der chiffrierte Text eingeben. Diese war mit der *Verschlüsselungseinheit* verkabelt, welche wiederum mit einem *Lampenfeld* verbunden war. Nach Eingabe eines Buchstaben auf der Tastatur,

leuchtet dieser auf dem Lampenfeld chiffriert auf. Die Verschlüsselungseinheit setzte sich aus mehreren *Rotoren*, einem *Reflektor* und einem *Steckbrett*[1] zusammen.

Funktionsweise

Rotoren waren, grob ausgedrückt, dicke Isolator-Scheiben, kreuz und quer durchzogen von jeweils 26 Drähten mit Schleifkontakten auf beiden Seiten. Kombiniert man eine Tastatur, einen Rotor und das Lampenfeld erhält man bereits eine einfache monoalphabetische Verschlüsselung (ein Buchstabe wird auf einen bestimmten anderen Buchstaben abgebildet). Um eine polyalphabetische Codierung zu erreichen, wurde der Rotor mit jeder Eingabe um eine Position gedreht. So wurde derselbe Buchstabe bei wiederholter Eingabe jeweils unterschiedlich verschlüsselt. Durch Kombination von n Rotor-Scheiben erhöht sich die Zahl der Schlüssel auf 26^n. Die Enigma wurde so konstruiert, dass sich die verschiedenen Rotoren in beliebiger Reihenfolge einbauen ließen, wodurch die Anzahl der Schlüssel wiederum um den Faktor $n!$ erhöht wurde. Durch einen Haken im inneren des Rotors zog dieser nach Vollendung einer ganzen Umdrehung seinen Nachfolger um eine Position weiter (ähnlich einem Kilometerzähler).

Um ein synchrones Verschlüsselungsverfahren zu realisieren, wurde neben den Rotor-Scheiben eine weiteres scheibenförmiges Bauteil, der *Reflektor*, eingesetzt, welcher starr installiert war und dessen Kontakte an der selben Seite raus- wie reinkamen. Strom floss also durch die n Rotoren zum Reflektor und durch die n Rotoren wieder zurück zu den Lampen (siehe Abbildung 2).

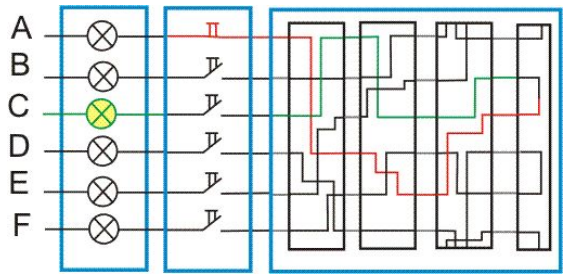

Abbildung 2: Aufbau mit drei Rotoren und einem Reflektor.

[1] Nachdem diese Aufgabe lediglich die Implementierung einer Enigma mit n Rotoren und einem Reflektor umfasst, wird hier nicht näher auf die Funktionsweise des Steckbretts eingegangen, näher Information hierzu finden sie unter http://www-ivs.cs.uni-magdeburg.de/bs/lehre/wise0102/progb/vortraege/steffensauer/enigma_aufbau_04.html oder unter http://de.wikipedia.org/wiki/Enigma_(Maschine).

Bedienung

Vor der Bedienung der Enigma musste ein täglich wechselnder Schlüssel, der sog. *Tagschlüssel*, eingestellt werden. Er bestand aus einer Kombination von n Buchstaben und bestimmte die Ausgangstellungen der Rotoren. Darüber hinaus konnte von jedem Funker vor Versenden einer Nachricht ein sog. *Spruchschlüssel* (bestehend aus n Buchstaben) eingestellt werden, der eine *zusätzliche* Verschiebung der Rotoren bewirkte. So konnte trotz einheitlichem Tagesschlüssel jeder Spruch mit einer individuellen Einstellung codiert werden. Die Vorgehensweise zum Versenden einer Nachricht gestaltete sich also wie folgt:

- Die Rotoren wurden in der dem Tagschlüssel entsprechenden Reihenfolge angeordnet und in die vorgegebene Ausgangstellung gebracht.
- Der Funker wählte einen individuellen Spruchschlüssel ein, welcher mit den Tagschlüsseleinstellungen kodiert wurde und in kodierter Form der zu versendenden Nachricht voran gestellt wurde.
- Die Rotoren würden um die vom Spruchschlüssel definierten Werte zusätzlich gedreht.
- Die Nachricht wurde mit den Tag- und Spruchschlüsseleinstellungen codiert.

Zum Entschlüsseln einer Nachricht wurde wie folgt vorgegangen:

- Die Rotoren wurden in der dem Tagschlüssel entsprechenden Reihenfolge angeordnet und in die vorgegebene Ausgangstellung gebracht.
- Die ersten n Buchstaben der verschlüsselten Nachricht enthielten den Spruchschlüssel und wurden mit dem Tagschlüsseleinstellungen dekodiert.
- Die Rotoren wurde in die vom dekodierten Spruchschlüssel vorgegebene Positionen gebracht.
- Der Rest der verschlüsselten Nachricht wurde mit den Tag- und Spruchschlüsseleinstellungen decodiert.

Implementierung

Realisieren Sie eine Enigma Chiffriermaschine indem Sie eine geeignete Klassenarchitektur entwerfen. Achten Sie dabei auf einen angebrachten Einsatz von Vererbung und Komposition. Beachten Sie außerdem unten angeführte Klassendefinitionen. Trennen Sie Definition und Implementierung der Klassen mit Hilfe von Header-Files und eines cpp-Files. Kommentieren Sie Ihren Code soweit es für das Verständnis erforderlich ist. Dies gilt inbesondere für aufwändigere Methoden wie `encrypt`, `decrypt`, `processMsg` und `init`. Testen Sie Ihre Klassen ausführlich, insbesondere für ungültige Benutzereingaben. Achten Sie darauf, reservierten Speicher wieder freizugeben!

Die Enigma-Klasse

Die Enigma Klasse stellt die zentrale Einheit Ihrer Implementierung dar. Sie verbindet die Rotor-Elemente und den Reflektor und bietet dem Benutzer eine Schnittstelle zur Konfiguration und Bedienung der Chiffriermaschine. Anzahl und Reihenfolge der Rotoren soll dabei beliebig und veränderbar sein. Tag- und Spruchschlüssel sollen individuell setzbar sein.

Listing 1: Klassendefinition der Enigma-Klasse:

```
1  class Enigma {
2
3  public:
4
5      Enigma(int numRotors);
6      ~Enigma();
7
8      // add a rotor of a specific type
9      void addRotor(RotorType type);
10     // exchange the rotor at position 'posIdx'
11     void changeRotor(Rotor& rotor, int posIdx);
12
13     // set the daily key settings
14     void setDailyKey(char key[]);
15     // set the message key settings
16     void setMessageKey(char key[]);
17      // clear daily and message key settings
18     void reset();
19
20     // encrypt the message key with the daily key settings
21     // encrypt 'msg' with the current daily and message key settings
22     std::string encrypt(const std::string& msg);
23     // decrypt the message with current daily and message key settings
24     // (the message key is read at the beginning of the message)
25     std::string decrypt(const std::string& msg);
```

```
26
27 private:
28     // apply the message key setting (set offset for each rotor)
29     void applyMsgKey();
30     // read each word from 'msg' and encrypt/decrypt it
31     void processMsg(const std::string& msg, std::string& out);
32     // get the index of the char 'c' (range: [A..Z,a..z])
33     int toInt(char c);
34    // get the corresponding char for the int 'i' (range: [0..NUM_LETTERS[)
35     char toChar(int i);
36
37 private:
38     int          m_numRotors; // the number of rotors
39     Rotor**      m_rotors;    // array of pointers to rotors
40     Reflector    m_reflector; // the reflector module
41     char*        m_msgKey;     // the current msgKey
42 };
```

Die EncoderModule-Klasse, Rotor und Reflektor

Die Klasse EncoderModule stellt die zentrale Basisklasse für die Rotoren und den Reflektor dar. Es soll fünf unterschiedliche Typen von Rotoren geben, deren Belegung (`m_configuration[]`) eine zufällige, aber für jeden Programmstart gleiche, Permutation des 26-stelligen Alphabets ist. Der Reflektor verbindet jeweils zwei zufällige (aber bei jedem Programmstart gleiche) Buchstaben. Verwenden Sie hierzu die Funktionen `srand()` und `rand()` aus der Bibliothek `stdlib`.

Listing 2: Basisklasse der Enigma-Bauteile:

```
1  class EncoderModule {
2
3  public:
4      static const int NUM_LETTERS = 26;
5
6      EncoderModule(int randSeed);
7      virtual ~EncoderModule();
8
9      // initialize the configuration array
10     virtual void init() = 0;
11     // print the configuration array
12     virtual void print() const;
13     // get element at index 'idx'
14     virtual int getCode(int idx) const;
15     // get the index of element 'code'
16     virtual int getIndex(int element) const;
17
```

```
18 protected:
19     int       m_configuration[NUM_LETTERS];
20 };
```

Um vergleichbare Implementierunge zu realisieren, verwenden Sie folgende Enumeration von Rotortypen als `seed` zur Initialisierung der Zufallszahlenfolge:

Listing 3: Rotortypen-Enumeration:

```
1 enum RotorType {
2     ROTOR_TYPE_I = 56789,
3     ROTOR_TYPE_II = 33170,
4     ROTOR_TYPE_III = 7013023,
5     ROTOR_TYPE_IV = 3017,
6     ROTOR_TYPE_V = 42,
7 };
```

Benutzereingabe

Der Benutzer soll über die Kommandozeile die Möglichkeit haben, Anzahl, Typ und Reihenfolge der Rotoren zu wählen, sowie den jeweils aus n Buchstaben bestehenden Tag- und Spruchschlüssel zu setzen (n = Anzahl der Rotoren). Außerdem soll die Eingabe einer (verschlüsselten) Nachricht und deren Encodierung bzw.Dekodierung wiederholt möglich sein, bis der Benutzer das Programm durch Eingabe von 'q' beendet. Schreiben Sie für die Eingabeaufforderung und das Lesen der verschiedenen Benutzereingaben jeweils passende Funktionen, deren Deklaration und Implementierung Sie in einem Header- und cpp-File namens *Input.h* bzw. *Input.cpp* trennen. Verwenden Sie dazu `getline()` und die aus der ersten Übung bekannte Vorgehensweise zum typsicheren Einlesen.