CIPHER KALK

Di ANDREA DEIDDA

Progetto di programmazione ad oggetti A.S. 2017/2018

INTRODUZIONE

Il progetto riguarda una calcolatrice che è in grado di cifrare e decifrare delle stringhe di caratteri alfabetici con tre diversi cifrari: Sostituzione, Vigenère e Vernam.

TEORIA SUI CIFRARI

Sostituzione

Il cifrario a sostituzione come suggerisce il nome prende il testo in chiaro e un numero N scelto e poi sostituisce ogni lettera del testo con quella che si trova N posizioni dopo rispetto all'ordine alfabetico.

Esempio:
Testo in chiaro: CIAO COME VA
N: 3
Testo cifrato: FLDR FRPH YD

Vigenère

Questo tipo di cifrario si può considerare come una generalizzazione del cifrario di Cesare ma invece di spostare sempre dello stesso numero di posti la lettera da cifrare, questa viene spostata di un numero di posti variabile ma ripetuto, determinato in base ad una parola chiave che va scritta ripetutamente sotto il messaggio, carattere per carattere.

La parola chiave deve essere concordata tra mittente e destinatario prima di iniziare la cifratura.

Esempio:
Testo in chiaro: CIAO COME VA
Chiave: BENE
Stringa di cifratura: BENE BENE BE
Testo cifrato: DMNS DSZI WE

Il testo cifrato si ottiene spostando la lettera chiara di un numero fisso di caratteri, pari al numero ordinale della lettera corrispondente della chiave ovvero si esegue una somma aritmetica tra l'ordinale del chiaro (A = 0, B = 1, C = 2...) e quello della parola chiave. Il numero ordinale risultante indica la lettera cercata.
Se si supera l'ultima lettera si ricomincia dalla A, secondo la logica delle aritmetiche finite.

Vernam

Anche per questo cifrario, come per Vigenère, ogni lettera del testo in chiaro viene spostata di un numero di posti variabile ma ripetuto, determinato in base ad una parola chiave che va scritta ripetutamente sotto il messaggio, carattere per carattere.
A differenza del cifrario di Vigenère questo cifrario utilizza chiavi che hanno la stessa lunghezza del testo in chiaro (spazi esclusi).
La parola chiave deve essere concordata tra mittente e destinatario prima di iniziare la cifratura.

Esempio:
Testo in chiaro: CIAO COME VA
Chiave: BENEGRAZIE
Stringa di cifratura: BENE GRAZ IE
Testo cifrato: DMNS IFMD DE

Il testo cifrato si ottiene spostando la lettera chiara di un numero fisso di caratteri, pari al numero ordinale della lettera corrispondente della chiave ovvero si esegue una somma aritmetica tra l'ordinale del chiaro (A = 0, B = 1, C = 2...) e quello della parola chiave. Il numero ordinale risultante indica la lettera cercata.
Se si supera l'ultima lettera si ricomincia dalla A, secondo la logica delle aritmetiche finite.

NOTA BENE

In questa calcolatrice l'input della frase e delle chiavi NON è case sensitive, inoltre una volta inserite frase e chiave, prima della codifica (o decodifica) entrambe le stringhe verranno convertite in nuove stringhe contenenti solo lettere maiuscole.

GERARCHIA DELLE CLASSI

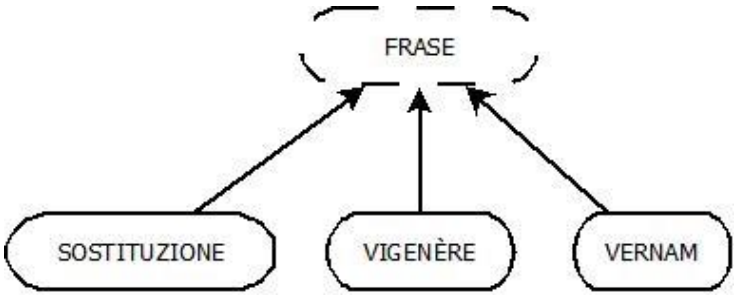
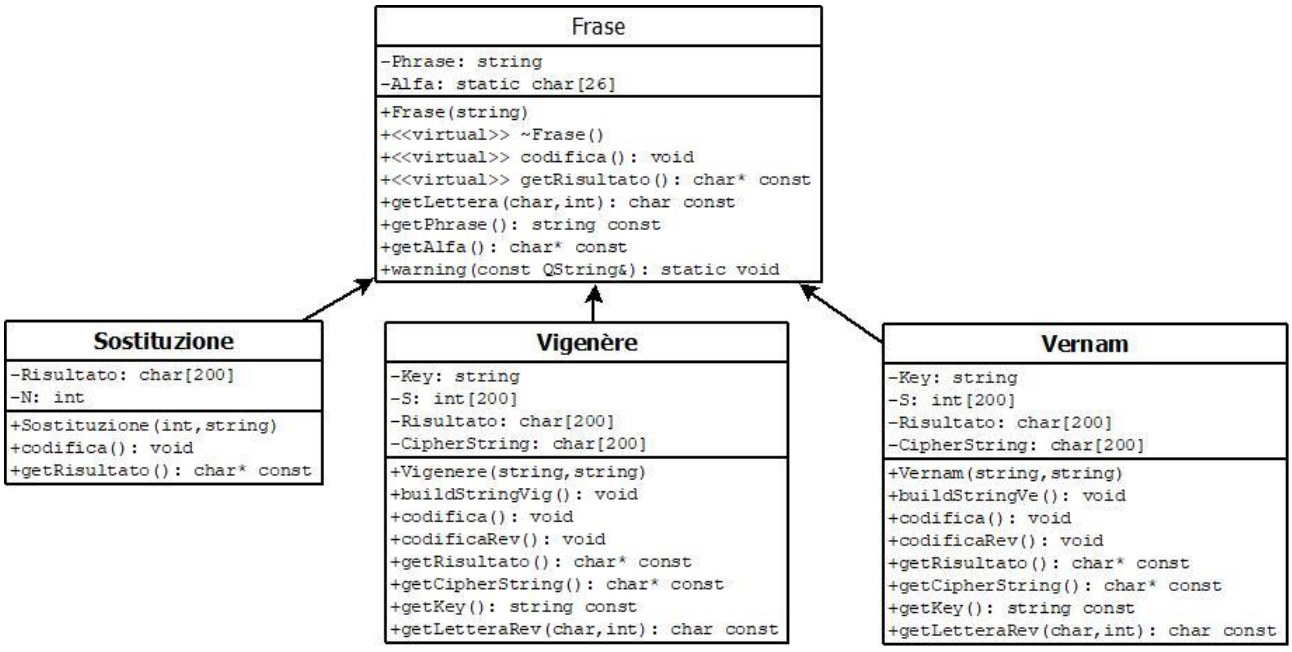


DIAGRAMMA UML



SPECIFICHE DELLE CLASSI

FRASE

Ereditarietà
Classe base astratta.

Attributi
`Phrase(int)` : testo inserito dall'utente
`Alfa(static char[26])` : elenco delle lettere maiuscole dell'alfabeto

Metodi

`Frase(string)`

Costruttore che effettua i controlli per vedere se la frase rispetta i vincoli imposti e in caso affermativo trasforma ogni lettera da minuscolo a maiuscolo, ignorando gli spazi.

`~Frase()`

Distruttore virtuale della classe Frase.

`void codifica()`

Metodo virtuale che cifra il testo in chiaro.

`char* getRisultato()`

Metodo virtuale che ritorna il risultato, ovvero la frase cifrata/decifrata.

`char getLettera(char,int)`

Metodo che trova e ritorna la lettera che si trova n posizioni dopo rispetto ad una determinata lettera.

`string getPhrase()`

Metodo che ritorna la frase inserita dall'utente.

`char* getAlfa()`

Metodo che ritorna l'elenco delle lettere dell'alfabeto.

`static void warning(const QString&)`

Metodo statico che manda in output un messaggio di errore.

SOSTITUZIONE

Ereditarietà

Classe ottenuta dalla derivazione pubblica di Frase.

Attributi

`N (int)` : numero di spostamenti

`Risultato (char[200])` : testo cifrato

Metodi

`Sosituzione(int,string)`

Costruttore che controlla il corretto inserimento di N.

`char* getRisultato()`

Metodo che ritorna la frase cifrata.

`void codifica()`

Metodo che cifra la frase.

VIGENÈRE

Ereditarietà

Classe ottenuta dalla derivazione pubblica di Frase.

Attributi

`Key (string)` : chiave

`S (int[200])` : array per numero di spostamenti

`Risultato (char[200])` : testo risultante

`CipherString (char[200])` : stringa di cifratura

Metodi

`Vigenere (string,string)`

Costruttore che controlla se la chiave inserita rispetta i vincoli imposti e in caso affermativo trasforma ogni lettera da minuscolo a maiuscolo.

`void buildStringVig()`

Metodo che costruisce la stringa di cifratura e l'array che conterrà il numero di spostamenti da effettuare per ciascuna lettera del testo inserito.

`void codifica()`

Metodo che cifra la frase.

`void codificaRev()`

Metodo che decifra la frase cifrata.

`char* getRisultato()`
Metodo che ritorna la frase cifrata/decifrata.
`char* getCipherString()`
Metodo che ritorna la stringa di cifratura.
`string getKey()`
Metodo che ritorna la chiave.
`char getLetteraRev(char,int)`
Metodo che trova e ritorna la lettera che si trova n posizioni prima rispetto ad una determinata lettera.

VERNAM

Ereditarietà
Classe ottenuta dalla derivazione pubblica di Frase.

Attributi
`Key (string)` : chiave
`S (int[200])` : array per numero di spostamenti
`Risultato (char[200])` : testo risultante
`CipherString (char[200])` : stringa di cifratura

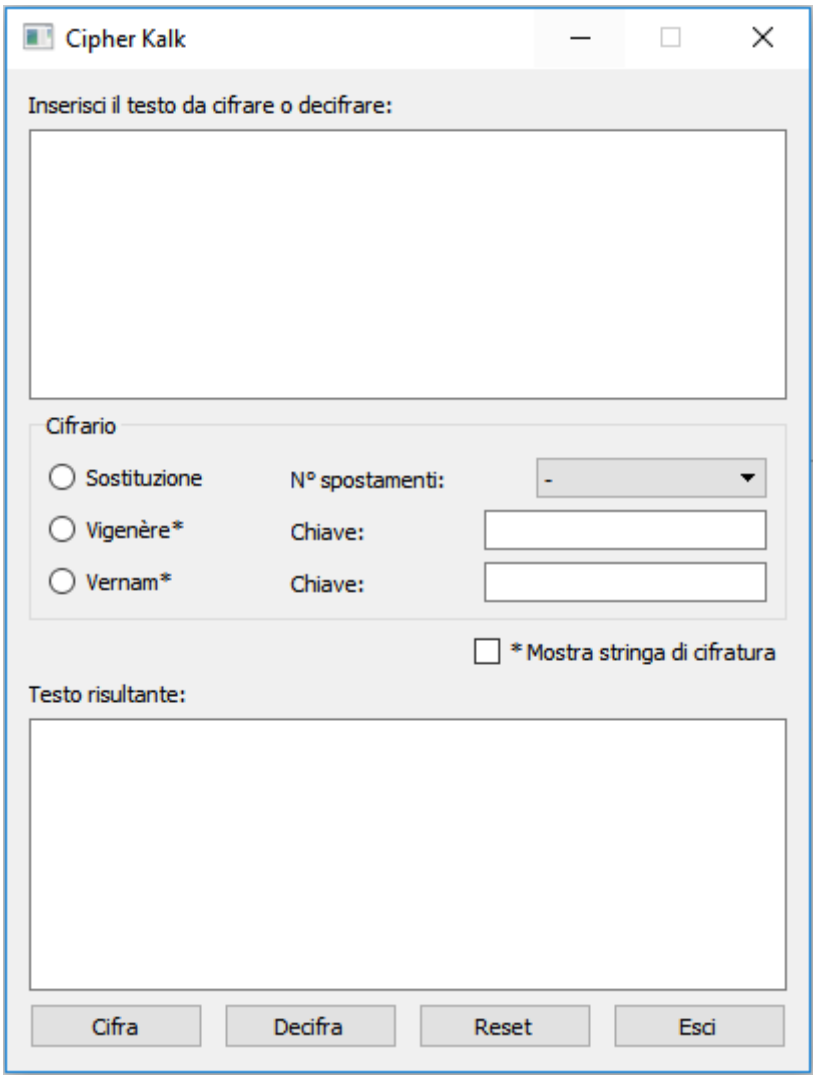
Metodi
`Vernam (string,string)`
Costruttore che controlla se la chiave inserita rispetta i vincoli imposti e in caso affermativo trasforma ogni lettera da minuscolo a maiuscolo.
`void buildStringVe()`
Metodo che costruisce la stringa di cifratura e l'array che conterrà il numero di spostamenti da effettuare per ciascuna lettera del testo inserito.
`void codifica()`
Metodo che cifra la frase.
`void codificaRev()`
Metodo che decifra la frase cifrata.
`char* getRisultato()`
Metodo che ritorna la frase cifrata/decifrata.
`char* getCipherString()`
Metodo che ritorna la stringa di cifratura.
`string getKey()`
Metodo che ritorna la chiave.
`char getLetteraRev(char,int)`
Metodo che trova e ritorna la lettera che si trova n posizioni prima rispetto ad una determinata lettera.

POLIMORFISMO
Nel codice è presente il puntatore polimorfo Frase* F che punta ad oggetti di tipo Sostituzione, Vigenère o Vernam, a seconda del cifrario scelto dall'utente, e viene usato per invocare i metodi virtuali: codifica() e getRisultato().

VOID CODIFICA()
In questo metodo avviene la codifica del testo cifrato: viene passata una lettera del testo in chiaro e il numero di spostamenti da effettuare alla funzione getLettera(char,int) della classe base Frase che ritornerà una lettera che andrà a comporre parte del nuovo testo cifrato.

CHAR* GETRISULTATO()
Questa funzione ritorna semplicemente un puntatore al nuovo testo cifrato.

GUI INIZIALE



MANUALE GUI

Come cifrare una stringa di testo:

- Inserire la frase scelta nell'apposito spazio di input nella parte superiore della finestra;
- Selezionare il cifrario con cui si desidera cifrare la frase e compilare il campo associato;
- (Opzionale) Se si desidera vedere anche la stringa di cifratura utilizzata spuntare il campo "Mostra stringa di cifratura";
- Cliccare sul pulsante "Cifra";

Come decifrare una stringa di testo:

- Inserire la frase scelta nell'apposito spazio di input nella parte superiore della finestra;
- Selezionare il cifrario con cui si desidera cifrare la frase e compilare il campo associato;
- (Opzionale) Se si desidera vedere anche la stringa di cifratura utilizzata spuntare il campo "Mostra stringa di cifratura";
- Cliccare sul pulsante "Decifra";

Tasti “Reset” e “Esci”
Come suggeriscono i nomi il tasto “Reset” una volta cliccato cancella il contenuto di tutti i campi di input e resetta i pulsanti di scelta del cifrario mentre il tasto “Esci” serve per chiudere la finestra ovvero la calcolatrice.

GUIDA AI CIFRARI

CIFRARIO	DESCRIZIONE
SOSTITUZIONE	Per questo cifrario è richiesto che l’utente inserisca anche il numero di spostamenti da effettuare per cifrare il testo in chiaro. Il numero da inserire deve essere dentro il range 0-26 (compresi). Con la sostituzione non è possibile decifrare un testo e non è possibile vedere la stringa di cifratura utilizzata perché questo tipo di cifrario non utilizza chiavi.
VIGENÈRE	Per questo cifrario è richiesto che l’utente inserisca una chiave* che verrà utilizzata per cifrare (o decifrare) il testo inserito. La chiave deve contenere solo caratteri alfabetici senza spazi. Se l’utente vuole può vedere la stringa di cifratura creata con la chiave inserita.
VERNAM	Per questo cifrario è richiesto che l’utente inserisca una chiave che verrà utilizzata per cifrare (o decifrare) il testo inserito. La chiave deve contenere solo caratteri alfabetici senza spazi e deve avere una lunghezza pari a quella del testo inserito (spazi esclusi). Se l’utente vuole può vedere la stringa di cifratura creata con la chiave inserita.

(*) Nel caso in cui l’utente inserisca, in questo tipo di cifrario, una chiave di lunghezza maggiore rispetto al testo inserito in precedenza durante il processo di cifratura (e anche decifratura) verrà presa in considerazione la parte della chiave entro i limiti della lunghezza del testo inserito.

ELENCO DEGLI ERRORI

- Mancato inserimento della frase da elaborare;
- Nessun cifrario selezionato;
- Mancato inserimento frase e nessuna selezione del cifrario;
- Frase inserita in forma errata (con caratteri diversi da quelli alfabetici e dal carattere spazio);
- Frase costituita da soli carattere spazio.

(Sostituzione)
- Il numero di spostamenti non è stato impostato;
- Operazione di decifratura non disponibile;
- Stringa di cifratura non disponibile;

(Vigenère)
- Chiave inserita in forma errata (con spazi o caratteri diversi da quelli alfabetici);
- Chiave non inserita;

(Vernam)
- Chiave inserita in forma errata (con spazi o caratteri diversi da quelli alfabetici);
- Lunghezza della chiave diversa da quella della frase inserita;
- Chiave non inserita;

ISTRUZIONI DI COMPILAZIONE

CIPHER KALK versione C++
-Aprire il terminale e recarsi dentro la cartella KALK;
-Digitare i seguenti comandi:
~$ qmake
~$ make
-Per eseguire il programma digitare:
~$ ./KALK
Nella cartella con i file .cpp e .h viene fornito anche il file .pro affinchè l'esecuzione del programma vada a buon termine.

CIPHER KALK versione Java
-Aprire il terminale e recarsi nella directory in cui sono contenuti i file .java
-Digitare il comando "javac *.java" (senza virgolette) per compilare i file;
-Per eseguire il programma digitare "java Use" (senza virgolette);

Precisazione: nelle stampe del programma in Java la parola "Vigenère" viene stampata senza l'accento per problemi legati alla codifica del carattere che impediscono la corretta compilazione.

ORE LAVORATIVE

Analisi preliminare del problema: ~5 ore;
La progettazione del modello e della GUI: ~4 ore;
Codifica di modello e GUI: ~20 ore;
Apprendimento della libreria Qt: ~15 ore;
Fase di debugging e di testing: ~11 ore;
Scrittura della relazione: ~5 ore;
Tempo di lavoro complessivo: ~60 ore;

Per la realizzazione di questo progetto il tetto delle 50 ore è stato superato perché durante la fase di codifica ci sono stati dei ripensamenti e delle correzioni sulle funzionalità della calcolatrice che hanno portato ad una nuova analisi iniziale del problema.
Inoltre la fase di apprendimento della libreria Qt non è stata semplice ed ha richiesto più ore del previsto.

SOFTWARE E SISTEMI OPERATIVI UTILIZZATI

C++
Sistema operativo di sviluppo: Windows 10
Versione Qt Creator: 4.7.0
Sistema operativo di test (laboratorio): Ubuntu 16.04.5
Versione Qt Laboratorio: 5.5.1

JAVA
Sistema operativo di sviluppo: Windows 10
Versione Eclipse: 4.8.0