Hauber Henrik ICLI3D

Szövegtitkosítás feladat dokumentáció

A program célja

A cél egy szövegtitkosító program megvalósítása, ami a konzolról tud szövegeket olvas be, azokat a felhasználó által kiválasztott módon titkosítja, majd a titkosított szöveget kiírja a konzolra. A felhasználó 4 titkosítási mód közül választhat:

- Caesar
- Vigenére
- ADFGVX
- Enigma

A program menüvezérelt. A futás menete:

1. A program bekéri a konzolról a titkosítani kívánt szöveget
2. Bekéri a titkosító algoritmus módját
3. Bekéri, hogy kódolni vagy dekódolni szeretnénk-e
4. Bekéri a titkosító algoritmushoz szükséges kódot/kódokat.
5. Kiírja a kódolt szöveget

Útmutató az egyes titkosító algoritmusok kódjának megadásához:

- Caesar: egy egész számot vár
- Vigenére: egy kulcsszót vár
- ADFGVX: egy kódábécét és egy kulcsszót vár. A kódábécében szerepelnek az angol ábécé betűi és a 10 számjegy is. A kódábécében a sorrend adja meg, hogy az egyes karakterekhez milyen betűket rendelünk hozzá

  Példa a kódábécére:

	A	D	F	G	V	X
A	a	b	c	d	e	f
D	g	h	i	j	k	l
F	m	n	o	p	q	r
G	s	t	u	v	w	x
V	y	z	0	1	2	3
X	4	5	6	7	8	9

  Forrás: https://dkalemis.wordpress.com/wp-content/uploads/2021/03/square.jpg

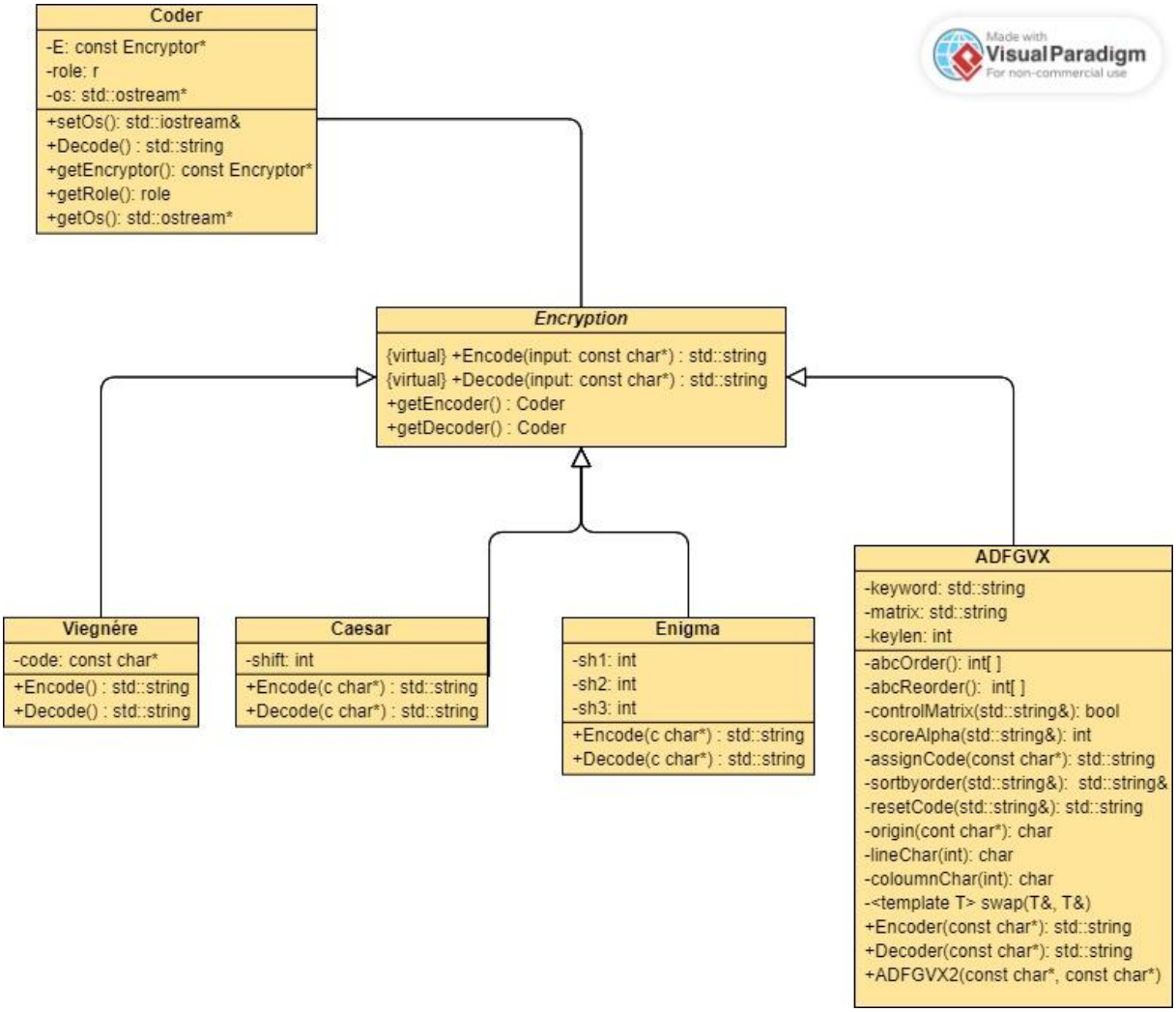
  A mátrixot balról jobbra, fentről lefele kell megadni.

- Enigma: 3 egész számot vár

Mintabemenet a .txt-ben

A program megvalósítása

A titkosító algoritmusokat megvalósító osztályok közös ősből származnak. A Coder osztály a feladatkiírásban megadott interfészhez kell. Az osztálystruktúra UML-diagramja:



A diagramból véletlen kimaradt 3 konstuktor:

- +Vigenére(const char*)
- +Caesar(int)
- +Enigma(int, int, int)

Codernél véletlenül szerepel Decoder, aminek nem kellene.

Az Encryptor getEncoder() és getDecoder() függvényeivel hozható létre a Coder osztály. Az Encoder() és Decoder() függvények hajtják végre a kódolást a leszármazottakban más privát segédfüggvények segítségével.

Mindegyik algoritmus kezeli a kis- és nagybetűket, más karaktereket nem kódol, azokat változtatás nélkül írják be az új sztringbe. Így kódolás majd dekódolás után mindig az eredeti szöveget kapjuk vissza.

Hauber Henrik ICLI3D

Caesar

A kapott szöveg betűit a megadott számmal tolja el, így hozza létre a kódot. Negatív és nagy eltolásra is működik, az abc túlindexelésekor túlcsordul, visszaugrik az abc elejére, amit maradék képzésével oldottam meg.

Vigenére

Minden betűt a kulcsszó rá eső betűjének A-tól való eltolásával tolunk el. Az egyes betűk eltolására Caesart használ. Ahogy bejárjuk a sztringet, mindig a kulcsszóban lévő indexet is növeljük, ha elérünk a kulcsszó végére, visszaugrunk az elejére (maradékképzés).

Enigma

Az eredeti titkosító gépben 3 tárcsa volt amin körben az ábécé betűi voltak. Minden leütéskor az egyes tárcsa elfordult. A második tárcsa akkor fordult el, ha az első megtett egy kört, a harmadik, ha a második megtett egy kört. A tárcsák forgását Caesar eltolásokkal imitáltam. A konstruktorban kell megadni a tárcsák kezdeti eltolását (sh1, sh2, sh3). Az egyes tárcsa eltolása az input minden karakterénél (nem betűjénél) növekszik, minden 26- leütéskor a második tárcsa eltolása is nő, a harmadik eltolást 676 karakterenként növeljük. Dekódoláskor a kezdeti eltolás a megadott értékek -1-szerese, az egyes eltolások pedig megfelelő lépésenként eggyel csökkennek. A kódolt szöveg egymásba ágyazott Caesar eltolásokkal alakul ki

ADFGVX

A kód egy megfeleltetési mátrix és egy kulcsszó. Az eredeti algoritmusban először a mátrixból kikeressük a két kódoló karaktert, majd az új karaktereket a kulcsszó alá írjuk sorokba. Soronként a sorrendet a kulcs betűrendjébe rendezzük, majd a oszloponként leírva a betűket kapjuk meg a kódolt szöveget. Én az oszloponként leírást kihagytam, Csak megfeleltetek, majd betűrend szerint keverek. Dekódoláskor figyelni kell, hogy érvényes-e a bejövő sztring. Csak páros számú betű jöhet, azok is csak „adfgvx” elemei lehetnek. Az attribútumok: a mátrix, a kód és a kódhossz. A megvalósításkor nehéz volt a számokon és betűkön kívüli karakterekre figyelni, hogy dekódolás után is ugyanazt kapjuk, mint kódolás előtt. Sok segédfüggvényt használok, amik privátak:

- abcOrder: egy keylen hosszú tömböt növekvő sorban feltölti számokkal, majd a számokat úgy rendezi a swap template segítségével, hogy mutassák, a lépés után mely helyre kell kerülniük az egy sorban lévő karaktereknek
- abcReorder: meghívja abcOrdert, majd úgy rendezi a számokat, hogy arra a helyre mutassanak a sorban, hogy visszaálljon a keverés előtti sorrend.
- scorealpha: megszámolja a sztringben lévő betűket (nem alfanumerikus karaktereket)
- sortbyOrder: ez hajtja végre a keverést order szerint, ami kódoláskor abcOrder, dekódoláskor abcReorder, így mindkét esetben használható Az utolsó, kulcshossznál rövidebb szakaszon nem keverünk. Először felírjuk a keverendő betűk indexeit, majd beszúrjuk őket keverve
- coloumnChar és lineChar kiszámolja a sor- és oszlopszámot az indexből, majd ennek megfelelő betűt adnak vissza.
- controlMatrix: egy teljes ábécét tartalmazó sztringből törölve ellenőrizzük, hogy teljes-e a mátrix
- origin: egy két karakterből álló tömböt kapva (amik a sor- és oszlopszámot határozzák meg), megkeresi az eredeti karakter indexét a mátrixban, visszaadja az eredeti karaktert

- assignCode: lineChar és coloumnChar segítségével végzi a mátrixnak megfelelő hozzárendelést
- resetCode: a sztringet bejárva origin segítségével végzi a mátrix szerinti hozzárendelést
- Encoder: először assignCode-ot, majd sortbyOrdert hívja
- Decoder: ellenőrzi a bemenetet. Először sortbyOrdert, majd resetCode-ot hívja

Coder

Csak a kiírásban megadott interfész megvalósításához kell, << operátor overloadjával valósítja meg az interfészt, A túlterhelt operátor hívja a tagfüggvényeit.

- getRole és setRole a feladatát állítja be, adja vissza
- setOs és setOs a tárolt ostreamet (amire << operátorra írunk) állítja be, adja vissza
- getEncryptor adja << operátornak az objektumot, amin Encodert vagy Decodert hív

A menü

A menühöz a dekompozíció érdekében létrehoztam Mmenu osztályt, ami a választási lehetőségek kiírásához, az algoritmus- és módválasztáshoz szükséges értékeket kéri be a felhasználótól.

A menüben a bekért adatok alapján a switch hívja meg azokat a függvényeket (Test.ccp teteje), amik létrehozzák az algoritmushoz tartozó objektumot, a konstruktor paramétereit bekérik a felhasználótól, majd elvégzik a kódolást. A mainben kezelem a hibákat, amiket az érvénytelen bemenet vált ki, ilyenkor a program visszaugrik az elejére. Számok, egy-egy szó bekéréséhez a getIn() függvénytemplatet használom több helyen is. Egy dekódolás elvégzése után végtelen ciklusban újra a bejövő adatot kéri a főprogram.

Továbbfejlesztési lehetőségek

Korábban terveztem még, hogy fájlból is tudjon olvasni a program, de erre nem jutott idő. A menüt főleg, ha több funkciót tudna, célszerű lenne State Design Patternnel megvalósítani.

Források, példák, amikel teszteltem a programot:

- https://hu.wikipedia.org/wiki/Vigen%C3%A8re-rejtjel
- https://en.wikipedia.org/wiki/ADFGVX_cipher
- https://hu.wikipedia.org/wiki/Caesar-rejtjel
- https://hu.wikipedia.org/wiki/Enigma_(g%C3%A9p)