

Module :	PRB - Programming Basics		
Opdracht :	LUSSEN : coderen en decoderen	Type :	Individuele Opdracht

Lussen – Coderen & decoderen

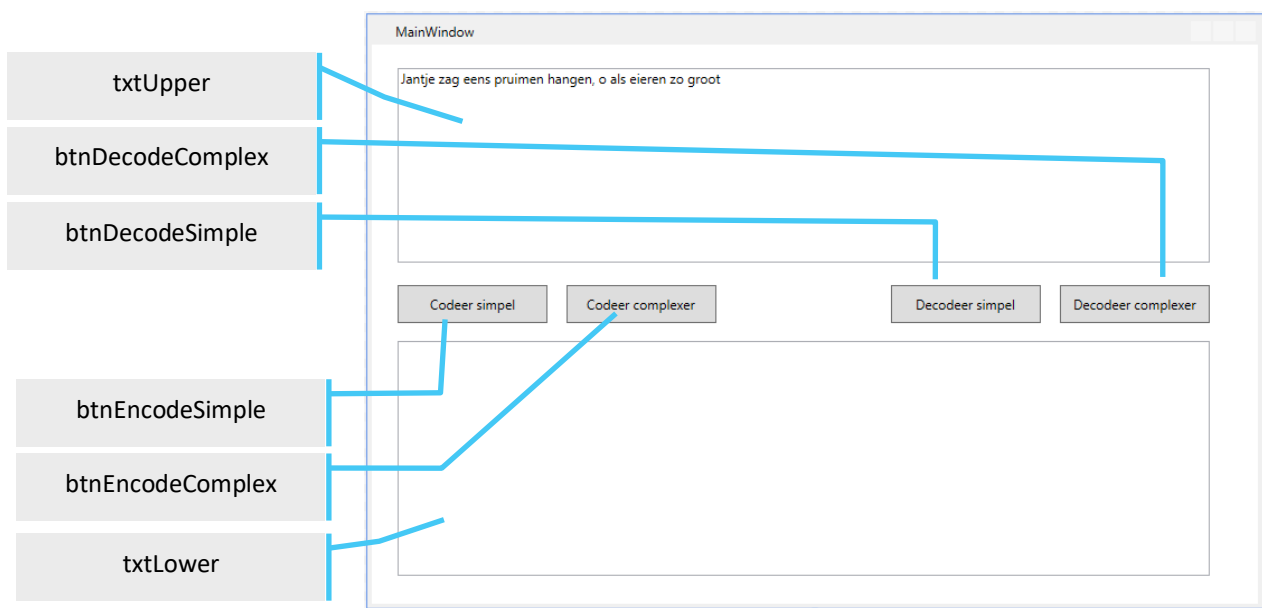
OPGAVE



Code repository

Haal de startsituatie van GitHub

`git clone https://github.com/howest-gp-prb/oe-lussen-coderen-en-decoderen-start.git`



DEEL 1

Bovenaan type je gewoon een tekst in. Wanneer op `btnEncodeSimple` wordt geklikt dient de tekst van `txtUpper` gelezen te worden en versleuteld. De versleutelde tekst dient in `txtLower` te worden getoond.

Voor het versleutel ga je letter per letter uit `txtUpper` omzetten naar een ander teken.

Je weet dat elk karakter (letter, cijfer, leesteken) een zgn ASCII waarden heeft (zie <http://www.asciitabel.be/>)

Wanneer je een letter uit een string omzet naar een char, volstaat het om die dan op zijn beurt om te zetten naar een byte (zie hoofdstuk variabelen).

Je telt vervolgens bij die byte-waarde een bepaald getal bij (bv 128) en je doet dan het omgekeerde (van byte naar char naar string).

Terzelfder tijd bouw je een nieuwe string met de aldus verkregen tekens en je toont het resultaat in txtLower.

Dus

- Haal 1 letter uit de string
- Zet die letter op naar een char (gebruik hiervoor de char.parse() methode)
- Typecast dit karakter naar een byte (gebruik hiervoor ...(byte)karakter ...)
- Verhoog de waarde in die byte met 128
- Zet die byte om naar een char en voeg die tenslotte toe aan de nieuw op te bouwen (gecodeerde) tekst.

Wat je ook nog moet weten

Wanneer er zich bij een byte overflow voordoet (het getal wordt te groot om in de byte te bewaren) dan geeft dit GEEN fout maar gaat er een gedeelte van de waarde verloren (zie voorbeeld hieronder). In deze oefening komt ons dit uitstekend van pas.

Stel :


- Karakter 'ä' heeft ASCII waarde 228.
- Als 228 in een byte variabele bewaard wordt en we verhogen die met 128, dan komen we uit op 356, wat te groot is voor een byte (max waarde = 255). Er treedt overflow op (de meest betekenisvolle bit – de linkse – gaat verloren) waardoor in die byte variabele de waarde 100 komt.
De ASCII waarde 100 is de letter 'd'.
- Dus 'ä' wordt gecodeerd naar 'd'.

btnDecodeSimple doet net hetzelfde, maar dan in de omgekeerde richting



Code repository

De oplossing van deel 1 kan je hier terugvinden :

 git clone <https://github.com/howest-gp-prb/oe-lussen-coderen-en-decoderen-einde-deel1.git>

DEEL 2

Opgave 2 (btnEncodeComplex) doet min of meer hetzelfde maar nu wordt het coderingsmechanisme voorzien van een sleutel.

Stel dat je een methode hiervoor maakt (wat ik ten sterkste kan aanbevelen) dan dient die methode 2 parameters te ontvangen : de te coderen tekst, en een sleutel.

De sleutel is gewoon een stuk tekst (een woord) dat je zelf mag kiezen (of je kan die bijvoorbeeld laten genereren door een globale variabele te maken en daarin bij opstart een GUID waarde te plaatsen).


- Je maakt een byte variabele aan die je instelt op 0, je overloopt alle letters van de sleutel en je telt alle ascii waarden op in die byte variabele : er zal waarschijnlijk meerdere keren overflow gebeuren, maar dat geeft niet.
- Vervolgens overloop je weer de brontekst, maar dit keer lees je eerst de laatste letter, vervolgens de voorlaatste, enz tot je de eerste gelezen hebt.
- We verhogen nu elke bytewaarde van de letter niet meer met een vaste waarde (zoals 128) hierboven maar met de waarde die onze sleutel ons opleverde.

Uiteraard dient bij btnDecodeComplex net het omgekeerde te gebeuren.



Code repository

De oplossing van deel 1 kan je hier terugvinden :

 `git clone` <https://github.com/howest-gp-prb/oe-lussen-coderen-en-decoderen-einde-deel2.git>