Travail pratique 7

Le Chiffrement de César

Travail présenté dans le cadre du cours
Logique de programmation

Par Jérémie Normand
Groupe 00001

Le Chiffrement de César

Énoncé

Ce que l'on appelle le chiffrement de César est probablement l'un des plus anciens codages au monde (et plus certainement l'un des plus simples qui soient), dans la mesure où Jules César lui-même l'aurait utilisé.

Aussi appelé chiffrement par décalage, il consiste simplement en une permutation de chaque lettre par une autre, par translation d'un certain nombre de positions dans l'alphabet (toujours dans le même sens bien sûr). Si l'on fait un décalage à droite de trois positions du mot CESAR, cela donne FHVDU (car C + 3 = F dans l'alphabet).

Ce chiffrement par substitution est donc une simple permutation circulaire de l'alphabet qui peut s'exprimer à l'aide d'une congruence sur les entiers. Prenons l'entier nn comme clé de cryptage.

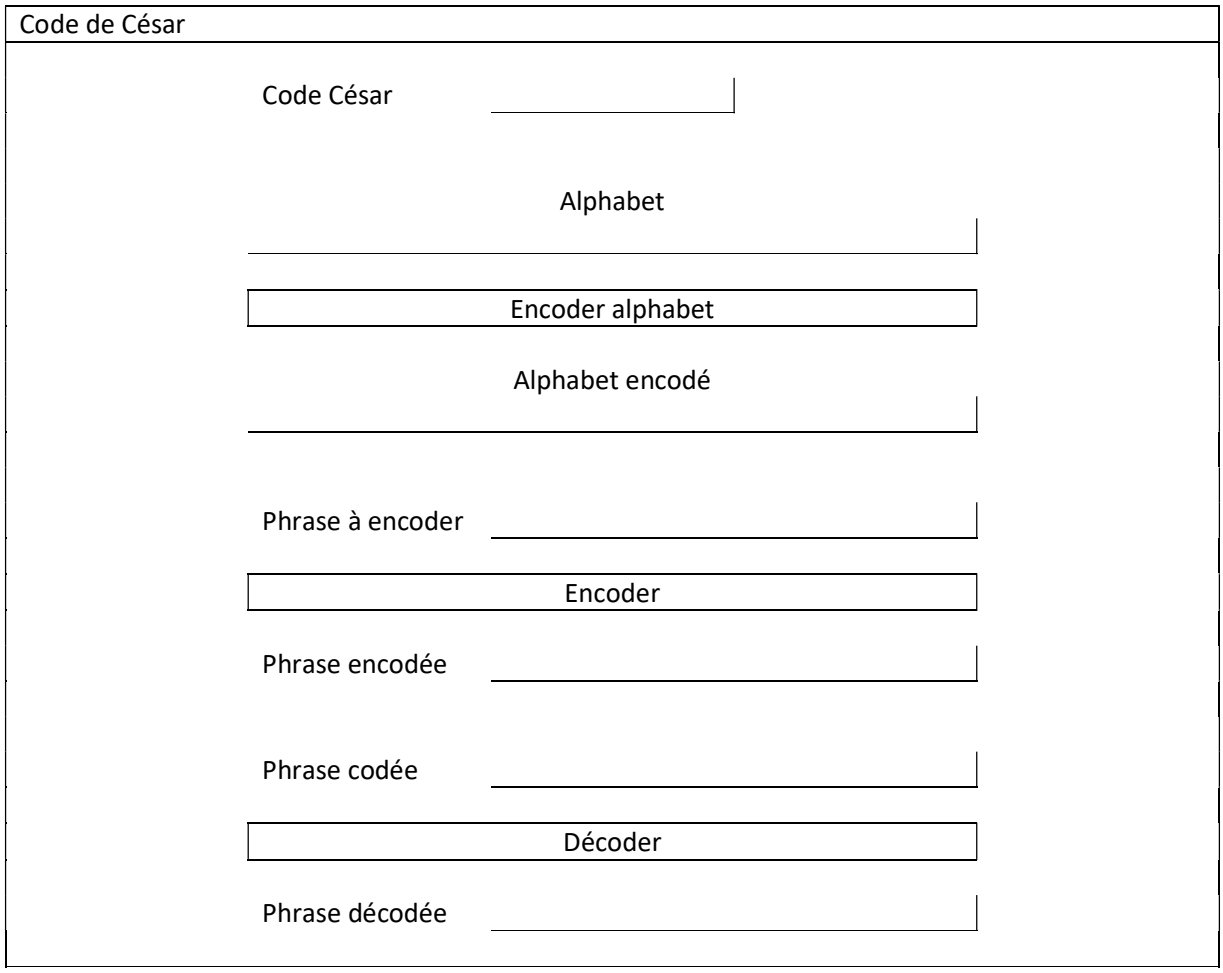
La clé de cryptage est le caractère correspondant à la valeur que l'on ajoute à chaque lettre pour effectuerle codage. Dans le premier exemple, la clé est C (étant la $3^{ème}$ lettre de l'alphabet).

Ce système de cryptage symétrique a pour inconvénient d'être particulièrement simple à casser, une soustraction permettant de remonter à la lettre substituée. Afin de connaitre la clé de cryptage, il suffit d'une petite étude statistique. En effet, certaines lettres sont plus fréquentes que d'autres : en français par exemple, c'est la lettre « e » qui revient le plus souvent. Ainsi, la lettre étant la plus fréquente dans le message à décoder. peut correspondre au « e ». Il ne reste plus ensuite qu'à décrypter le reste du message.

Données

Entrées	Constantes	Sorties
CodeCesar	Alphabet [ ]	AlphabetEncode [ ]
PhraseAEncoder [ ]		PhraseEncodee [ ]
PhraseCodee [ ]		PhraseDecodee [ ]

Conception visuelle



Algorithmes

CodeCesar
Alphabet [ ]
PhraseAEncoder [ ]
PhraseCodee [ ]

AlphabetEncode [ ]
PhraseEncodee [ ]
PhraseDecodee [ ]

Initialiser ( )
    Alphabet [ ] ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

AfficherAlphabetEncode (evenement)
    codeCesar ← **LireCodeCesar** (CodeCesar)
    alphabet [ ] ← **LireChaine** (Alphabet [ ])

    Si **ValiderEntrees** (codeCesar, alphabet [ ]) = Faux
        Renvoyer

    AlphabetEncode [ ] ← **EncoderChaine** (alphabet [ ], codeCesar)

AfficherPhraseEncodee (evenement)
    codeCesar ← **LireCodeCesar** (CodeCesar)
    phraseAEncoder [ ] ← **LireChaine** (PhraseAEncoder [ ])

    Si **ValiderEntrees** (codeCesar, phraseAEncoder [ ]) = Faux
        Renvoyer

    PhraseEncodee [ ] ← **EncoderChaine** (phraseAEncoder [ ], codeCesar,)

AfficherPhraseDecodee (evenement)
    codeCesar ← **LireCodeCesar** (CodeCesar)
    phraseCodee [ ] ← **LireChaine** (PhraseCodee [ ])

    Si **ValiderEntrees** (codeCesar, phraseCodee [ ]) = Faux
        Renvoyer

    PhraseDecodee [ ] ← **DecoderChaine** (phraseCodee [ ], codeCesar)

LireCodeCesar (code)
    codeLu, estValide ← *Lire* code

    Si estValide = Faux
        codeLu ← -1

    Renvoyer codeLu

LireChaine (chaine [ ])
    chaineLu [ ] ← *Lire* chaine [ ]

    Renvoyer chaineLu [ ]

```
ValiderEntrees ( code, chaine [ ] )
    Si ValiderCodeCesar ( code ) = Faux
        Afficher Exception
        Renvoyer Faux

    Si ValiderChaine ( chaine [ ] ) = Faux
        Afficher Exception
        Renvoyer Faux

    Renvoyer Vrai

ValiderCodeCesar ( code )
    Si code < 0 Ou code ≥ Alphabet [ ].Quantite
        Renvoyer Faux

    Renvoyer Vrai

ValiderChaine ( chaine [ ] )
    Si chaine [ ].Quantite  = 0
        Renvoyer Faux

    Renvoyer Vrai

EncoderChaine (chaine [ ], code )
    chaineEncodee [ ]
    PourChaque caractere Dans chaine [ ]
        chaineEncodee [ ].Ajouter ( EncoderCaractere ( caractere, code ) )

    Renvoyer chaineEncodee [ ]

EncoderCaractere ( caractere, code )
    indexDeCaractereDansAlphabet ← Alphabet [ ].IndexDe ( caractere.Majuscule )

    caractereEncode
    Si indexDeCaractereDansAlphabet < 0
        caractereEncode ← '_'
    SinonSi indexDeCaractereDansAlphabet + code < Alphabet [ ].Quantite
        caractereEncode ← Alphabet [ indexDeCaractereDansAlphabet + code ]
    Autrement
        caractereEncode ← Alphabet [ indexDeCaractereDansAlphabet + code - Alphabet [ ].Quantite ]

    Renvoyer caractereEncode
```

DecoderChaine (chaine [ ], code)
    chaineDecodee [ ]
    PourChaque caractere *Dans* chaine [ ]
        chaineDecodee [ ].**Ajouter** (**DecoderCaractere** (caractere, code))

    Renvoyer chaineDecodee

DecoderCaractere (caractere, code)
    indexDeCaractereDansAlphabet ← Alphabet [ ].**IndexDe** (caractere.**Majuscule**)

    caractereDecode
    Si indexDeCaractereDansAlphabet < 0
        caractereDecode ← '_'
    SinonSi indexDeCaractereDansAlphabet - code ≥ 0
        caractereDecode ← Alphabet [indexDeCaractereDansAlphabet - code]
    Autrement
        caractereDecode ← Alphabet [indexDeCaractereDansAlphabet - code + Alphabet [ ].**Quantite**]

    Renvoyer caractereDecode

Essais

Variables	**Entrées 1**	**Sorties 1**	**Entrées 2**	**Sorties 2**
CodeCesar	3		4	
Alphabet [ ]	"ABC…YZ "		"ABC...YZ	
AlphabetEncode [ ]		"DEF…ABC"		"EFG…BCD"
PhraseAEncoder [ ]	"Salle de Bain"		"SalLe # 42!"	
PhraseEncodee [ ]		"VDOOH_GH_EDLQ"		"WEPPI_____"
PhraseCoder [ ]	"VDOOH GH EDLQ"		"WEPPI # 42"	
PhraseDecoder [ ]		"SALLE_DE_BAIN"		"SALLE_____"