procedure DecrypteCesar ( Str : in string , str2 : out string);

fonction MySucc (C : in caractere ) renvoie caractere

debut

si (C vaut 'Z') renvoie 'A';

sinon\_si (C vaut 'z') renvoie 'a';

sinon renvoie succ(C);

fsi

fin

fonction IsAlpha (C : in caractere ) renvoie booleen

debut

renvoie (isUpper (C) OU\_SINON isLower(C));

fin

fonction IsAlpha (C : in caractere ) renvoie booleen

debut

declarer alphabet : string <- "abc...xyzABC...XYZ";

renvoie (trouveCarDansString (C, alphabet) ne\_vaut\_pas taille (alphabet));

fin

fonction IsAlpha (C : in caractere ) renvoie booleen

debut

declarer alphabet : string <- "abc...xyzABC...XYZ";

declarer i : entier\_naturel <- 0;

tant\_que (i < taille (alphabet))

faire

si (C vaut alphabet[i]) sortie;

i <- i + 1;

ffaire

renvoie (i ne\_vaut\_pas taille (alphabet));

fin

fonction DecrypteCesar ( Str : in string ) renvoie string

debut

declarer chaineDecryptee : string;

pour (i variant\_de 0 a taille (Str) - 1)

faire

si (IsAlpha (str[i]))

chaineDecryptee <- chaineDecryptee + MyPred (Str[i]);

fsi

ffaire

renvoie chaineDecryptee;

fin

fonction DecrypteCesar ( Str : in string ) renvoie string

debut

declarer chaineDecryptee : string;

redimensionner (chaineDecryptee, taille (Str));

pour (i variant\_de 0 a taille (Str) - 1)

faire

si (IsAlpha (str[i]))

chaineDecryptee [i]<- MyPred (Str[i]);

fsi

ffaire

renvoie chaineDecryptee;

fin

Un geek ne crie pas il URL

algorithme cesar1

debut

boucle

Afficher ("Saisir une string : ");

declarer strACrypter : string;

saisir (strACrypter);

si (taille (strACrypter) vaut 0) sortie;

declarer chaineDecryptee : string <- CrypteCesar (strACrypter);

si (DecrypteCesar (chaineDecryptee) ne\_vaut\_pas strACrypter) sortie;

fboucle

fin

fonction CrypteCesar2 ( Str : in string , Cle : in entier\_naturel ) renvoie string

debut

declarer chaineCryptee : string;

pour (i variant\_de 0 a taille (Str) - 1)

faire

si (IsAlpha (str[i]))

declarer c : caractere <- str[i];

pour (j variant\_de 0 a cle)

faire

c <- MySucc (c);

ffaire

chaineDecryptee <- chaineDecryptee + c;

fsi

ffaire

renvoie chaineCryptee;

fin

fonction CrypteCesar3 ( Str : in string ) renvoie string

declarer chaineCryptee : string

debut

pour (i variant\_de 0 a taille (Str) - 1)

faire

declarer c : caractere <- str[i];

pour (j variant\_de 0 a i)

faire

c <- MySucc (c);

ffaire

chaineDecryptee <- chaineDecryptee + c;

ffaire

renvoie chaineCryptee;

fin

fonction CrypteCesar4 ( Str : in string ) renvoie string

declarer chaineCryptee : string

debut

pour (i variant\_de 0 a taille (Str) - 1)

faire

declarer c : caractere <- str[i];

si (modulo (i, 2) vaut 0)

c <- MySucc (c);

sinon

c <- MyPred (c);

fsi

chaineDecryptee <- chaineDecryptee + c;

ffaire

renvoie chaineCryptee;

fin

fonction CrypteCesar5 ( Str : in string ) renvoie string

declarer chaineCryptee : string

debut

pour (i variant\_de 0 a taille (Str) - 1)

faire

declarer c : caractere <- str[i];

si (modulo (i, 2) vaut 0)

pour (j variant\_de 0 a i)

faire

c <- MySucc (c);

ffaire

sinon

pour (j variant\_de 0 a i)

faire

c <- MyPred (c);

ffaire

fsi

chaineDecryptee <- chaineDecryptee + c;

ffaire

renvoie chaineCryptee;

fin