Mainc.c

```c
/**
* @file main.c
* @author Durand Thomas
* @brief fonction main avec preuve de concept des differents tri
* @version 0.1
* @date 2019-11-18
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


#include "cesar.h"
#include "vigenere.h"
#include "scytale.h"

/**
* @brief preuve de concept des differents tris
*
* @return int
*/
int main(){

char chaine[20] = "coucou a tous";

cryptageCesar(8,chaine);
printf("%s\n",chaine);
decryptageCesar(8,chaine);
printf("%s\n",chaine);

char cle[20] = "teSt";
char chaine2[20] = "couCou A toUs";
cryptageVigenere(cle,chaine2);
printf("%s\n",chaine2);
decryptageVigenere(cle,chaine2);
printf("%s\n",chaine2);

char* chaine3 = malloc(37*sizeof(char));
strcpy(chaine3,"RENDEZ VOUS DEMAIN SOIR A LA TIREUSE");
```

```
chaine3 = cryptageScytale(chaine3);
printf("%s\n",chaine3);
chaine3 = cryptageScytale(chaine3);
printf("%s\n",chaine3);


return 0;
}
```

cesar.c

```
/**
* @file cesar.c
* @author Durand Thomas
* @brief ensemble des fonctions de cesar
* @version 0.1
* @date 2019-11-18
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/**
* @brief cryptage de cesar
*
* @param decalage decalage de la chaine
* @param chaine chaine a crypter
*/
void cryptageCesar(int decalage,char* chaine){
for (int i = 0; i < strlen(chaine);i++){
if (chaine[i] =/= 32){
//minuscule
if (chaine[i] >= 97 && chaine[i] <= 122){
chaine[i] = chaine[i] + decalage;
if (chaine[i] > 122){
chaine[i] = (chaine[i] % 122) + 97;
}
}else if (chaine[i] >= 65 && chaine[i] <= 90){
chaine[i] = chaine[i] + decalage;
if (chaine[i] > 90){
```

```
chaine[i] = (chaine[i] % 90) + 65;
}
}
}
}
}

/**
* @brief decryptage de cesar
*
* @param decalage decalage de la chaine
* @param chaine chaine a decypter
*/
void decryptageCesar(int decalage,char* chaine){
for (int i = 0; i < strlen(chaine);i++){
if (chaine[i] =≠ 32){
//minuscule
if (chaine[i] ≥ 97 && chaine[i] ≤ 122){
chaine[i] = chaine[i] - decalage;
if (chaine[i] < 97){
chaine[i] = 122 - (97 - chaine[i]);
}
}else if (chaine[i] ≥ 65 && chaine[i] ≤ 90){
chaine[i] = chaine[i] + decalage;
if (chaine[i] < 65 ){
chaine[i] = 90 -(65 - chaine[i]);
}
}
}
}
}
```

vigenere.c

```
/**
* @file vigenere.c
* @author Durand Thomas
* @brief ensemble des fonctions lie a Vigenere
* @version 0.1
* @date 2019-11-18
*
* @copyright Copyright (c) 2019
*
```

```
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
* @brief cryptage de Vigenere
*
* @param cle cle de cryptage
* @param chaine chaine a crypter
*/
void cryptageVigenere(char* cle, char* chaine){
int j = 0;
int lencle = strlen(cle);
//passe la cle en minuscule
for (int i = 0; i < lencle; i++){
if (cle[i] ⩾ 65 && cle[i] ⩽ 90){
cle[i] += 32;
}
}
for (int i = 0; i < strlen(chaine);i++){
if (chaine[i] ≠ 32){
//minuscule
if (chaine[i] ⩾ 97 && chaine[i] ⩽ 122){
chaine[i] = (chaine[i] + (cle[j] - 97)) % 122;
if (chaine[i] < 97){
chaine[i] += 97;
}
}else if (chaine[i] ⩾ 65 && chaine[i] ⩽ 90){
chaine[i] = (chaine[i] + (cle[j] - 97)) % 90;
if (chaine[i] < 65){
chaine[i] += 65;
}
}
j++;
j = j % lencle;
}
}
}
/**
* @brief decryptage de Vigenere
*
* @param cle cle de cryptage
* @param chaine chaine a decrypter
```

```
*/
void decryptageVigenere(char* cle, char* chaine){
int j = 0;
int lencle = strlen(cle);
//passe la cle en minuscule
for (int i = 0; i < lencle; i++){
if (cle[i] ⩾ 65 && cle[i] ⩽ 90){
cle[i] += 32;
}
}
for (int i = 0; i < strlen(chaine);i++){
if (chaine[i] ≠ 32){
//minuscule
if (chaine[i] ⩾ 97 && chaine[i] ⩽ 122){
chaine[i] = chaine[i] - (cle[j] - 97);
if (chaine[i] < 97){
chaine[i] = 122 - (97 - chaine[i]);
}
}else if (chaine[i] ⩾ 65 && chaine[i] ⩽ 90){
chaine[i] = chaine[i] - (cle[j] - 97);
if (chaine[i] < 65 ){
chaine[i] = 90 -(65 - chaine[i]);
}
}
j++;
j = j % lencle;
}
}
}
```

scytale.c

```
/**
* @file scytale.c
* @author Durand Thomas
* @brief ensemble des fonctions lie a Scytale
* @version 0.1
* @date 2019-11-18
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>

/**
* @brief cryptage et decryptage de Scytale
*
* @param chaine chaine a crypter ou decrypter
* @return char*
*/
char* cryptageScytale(char* chaine){

int lenchaine = strlen(chaine);
int lenCarre = 1;
int i,pose = 0;
while((lenCarre*lenCarre) < lenchaine){
lenCarre++;
}
char* nvlChaine = malloc((lenCarre*lenCarre + 1)*sizeof(char) );

while((lenCarre*lenCarre) =≠ i){
if (pose ≥ lenchaine || chaine[pose] == '\0'){
nvlChaine[i] = 32;
}else{
nvlChaine[i] = chaine[pose];
}
pose += lenCarre;
if ((pose ≥ (lenCarre*lenCarre))){
pose = (pose % (lenCarre*lenCarre)) + 1;
}
i++;
}
nvlChaine[i] = '\0';
return nvlChaine;
}
```