KEEPIT

NOM : AZZOUZ

PRENOM : NAZIM

1. ***Introduction :***

De nos jours il est devenu nécessaire de sécuriser la sauvegarde des mots de passe, pour cela on a opté pour une solution de sécurité il s'agit bien de cryptage des mots de passe. C'est une procédure où on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de chiffrement. Ce principe est généralement lié au principe d'accès conditionnel.

L'objectif de ce projet est sauvegardé les mots de passé après les avoir cryptés, et de généré des mots de passes robustes et les sauvegardées aussi.

2. ***Bibliothèques utilisées :***

- *Bibliothèque string.h :* *c'est une bibliothèque qui contient* constantes et les déclarations de fonctions et de types utilisées pour la manipulation de chaînes de caractères comme :
  - ✓ *Strleen()* ➔ renvoie la longueur d'une chaine de caractère
  - ✓ *Strcpy()* ➔ permet de copier une chaine de caractère vers une autre .
  - ✓ *Strcmp()* ➔ permet de comparer entre deux chaine de caractères.
  - ✓ ***….***
- *Bibliothèque studio.h :* pour « Standard Input/Output Header » ou « En-tête Standard d'Entrée/Sortie », est l'en-tête de la bibliothèque standard du C déclarant les macros, les constantes et les définitions de fonctions utilisées dans les opérations d'entrée/sortie.
- *Bibliothèque stdlib.h :* Il contient les déclarations de fonctions traitant d'allocation-mémoire, de conversion de chaînes de caractères en types numériques (int, long, double), de tirages aléatoires, ...
- *Bibliothèque time.h :* Cette librairie fournit un ensemble de fonctions permettant la manipulation de dates, de temps et d'intervalles.

3. ***Conception du programme :***

- *Procédure qui permet d'afficher un menu de gestion de passwords*

```c
// Procédure qui permet d'afficher le mennu pour la gestion des passwords
void Affichier_Mennu(){
    printf( format: "\n\n-------mennu--------");
    printf( format: "\n1. Listing passwords\n"
    "2.Getting a password \n"
    "3.Editing passwords \n"
    "4.Deleting passwords \n"
    "5.Generate a password"
    "\n6. Exit");
}
```

- *Fonction qui permet de chiffrer un mot a partir d'une clé passé en paramètres :*

```c
// fonction qui permet de chiffrer le mot de passe a partir d'une clé passé en paramatére.
char *encrypt(char *plaintext_password,char *KEY)
{
    int len = strlen( Str: plaintext_password);

    // allocate memory for the result string
    char *result = malloc( Size: len + 1);

    // initialize the result string
    result[0] = '\0';

    // compute the XOR of the input strings
    for (int i = 0; i < len; i++) {
        result[i] = plaintext_password[i] + KEY[ i % strlen( Str: KEY) ];
    }
    result[len] = '\0'; // null-terminate the result string
```

- *Fonction qui permet de déchiffrer un mot chiffré a partir d'une clé passé en paramètres*

```c
// fonction qui permet de déchiffrer le un mot de passe chiffré partir d'une clé
char *decrypt(char *encrypted_password,char *KEY)
{
    int len = strlen( Str: encrypted_password);

    // allocate memory for the result string
    char *result = malloc( Size: len + 1);

    // initialize the result string
    result[0] = '\0';

    // compute the XOR of the input strings
    for (int i = 0; i < len; i++) {
        result[i] = encrypted_password[i] - KEY[ i % strlen( Str: KEY)];
    }
    result[len] = '\0'; // null-terminate the result string

    return result;
```

- *Fonction qui permet de vérifier si un fichier existe déjà ou non*

```c
// fonction permet de vérifier si un fichier existe déja ou non.
int file_exists (char *filename)
{
    // try to open the file
    FILE *file = fopen( Filename: filename,  Mode: "r");
    if (file == NULL) {
        // the file does not exist
        return 1 ;
    }

    // the file exists
    fclose( File: file);
    return 0;
}
```

- *Fonction qui permet établir une session après la création de fichier vault.txt de telle sorte donner au utilisateur le droit de saisis un username et un mot de passe puis comparer le username saisie avec le username crypté(ayant le ID : -1) dans la 1iere ligne de fichier vault.txt.*

```c
// fonction qui permet d'etablir une session  aprés avoir créé le fichier vault.txt
int open_session()
{
    printf( format: "\nusername :");
    char *username=malloc( Size: max_size);
    scanf( format: "%s",username);
    printf( format: "\npassword:");
    char *password=malloc( Size: max_size);
    scanf( format: "%s",password);
    FILE *file = fopen( Filename: "vault.txt",  Mode: "r");
    if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n");
    }
    char *contenu=malloc( Size: max_size);
    fgets( Buf: contenu, MaxCount: 256, File: file);
    char *username_crypt= malloc( Size: max_size);
```

```c
scanf( format: "%s",password);
FILE *file = fopen( Filename: "vault.txt",  Mode: "r");
if (file == NULL) {
    // unable to open the file
    fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n");
}
char *contenu=malloc( Size: max_size);
fgets( Buf: contenu, MaxCount: 256, File: file);
char *username_crypt= malloc( Size: max_size);
    int j=3; int k=0;
    while (contenu[j]!=':') {username_crypt[k]=contenu[j];k++; j++;} // recupéré l'username crypté du ID -1
    if (strcmp(encrypt( plaintext_password: username, KEY: THE_KEY),username_crypt)==0) return 0;
    else return 1;
```

- *procédure qui permet créé le fichier vault.txt et l'initialisé avec le username crypté 1 Ière password crypté ayant comme ID -1 .*

```c
// procédure qui permet créé le fichier vault.txt et l'inisialisé avec le 1 iere password ID -1
void create_vault(char* filename)
{
    char *username=malloc( Size: max_size),
    *password=malloc( Size: max_size),
    *encrypted_username=malloc( Size: max_size);
    char *encrypted_password= malloc( Size: max_size);
    printf( format: "\nWelcome on LockIT\n"
           "It's the first time you open LockIT, we created `vault.txt`."
           "\nYou must choose a username and a password to decode your vault !\n\n");
    printf( format: "\nUsername: ");
    gets( Buffer: username);
    printf( format: "\nPassword: ");
    gets( Buffer: password);
    encrypted_username = encrypt( plaintext_password: username, KEY: THE_KEY);
    encrypted_password = encrypt( plaintext_password: password, KEY: THE_KEY);
```

```c
    FILE *file = fopen( Filename: filename,  Mode: "w"); // open file to write
    if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file '%s' for writing.\n", filename);
    }

    // write the username and password to the file
    fprintf( stream: file,  format: "-1:%s:-1\n", encrypted_username);
    printf( format: "\nYour vault file has been initialized");
    // close the file
    fclose( File: file);
}
```

- *Procédure permet de lister les passwords qui existe dans le fichier vault.txt a l'exception de password ID -1 pour raison de sécurité*

```c
// procédure permet de lister les password qui existe dans le fichier vault.txt
void list_password(){
    printf( format: "\nlist of password:\n");
    FILE *file = fopen( Filename: "vault.txt",  Mode: "r");
if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n" );
          }
char *contenu=malloc( Size: max_size);
int i=0;
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
    if (i==0){i++;} // exepction de password avec ID -1
    else {
        int j;
        if (i>10 )  { j=3;} // si le ID>10 la longere de ID est 2 + longeure (:)
        else {j=2;}
```

```
    int j;
    if (i>10 )  { j=3;} // si le ID>10 la longere de ID est 2 + longeure (:)
    else {j=2;}
    while (contenu[j]!=':') j++; // pour retourner la postion du debut de password
    int k=0;j++;
    char *password =malloc( Size: strlen( Str: contenu)-j);
    // recupérer le password
    while(j<strlen( Str: contenu)-1)
    {
        password[k]=contenu[j];
        j++;k++;
    }
   printf( format: "%d:{%s}\n",i,decrypt( encrypted_password: password, KEY: THE_KEY)); // afichier le password recupérer déchifré
   i++;
 }
```

- procédure permet de modifier un password dans fichier vault.txt depuis le ID ➔ copier tout les lignes du fichier vault.txt dans un fichier temp.txt sauf la ligne ayant le ID demandé , modifier cette ligne avec le nouveau mot du passe puis l'ajouter dans le fichier temp.txt , après supprimer le fichier vault.txt et renommer temp.txt vers vault.txt (nouveau fichier avec la nouvelle modification).

```
// procédure permet de modifier un password dans fichier vault.txt depuis le ID
void edit_password(){
      printf( format: "\ndonner le ID du password :");
    int ID;
    scanf( format: "%d",&ID);
    FILE *file = fopen( Filename: "vault.txt",  Mode: "r");

if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n" );
          }
char *contenu=malloc( Size: max_size);
int i=0;
if (ID>0) {
FILE *temp=fopen( Filename: "temp.txt", Mode: "a"); // créé le fichier temp  en mode ajout
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
```

```
if (i==ID){
       char *s=malloc( Size: max_size);
        int j;
        if (i>10 )  { j=3;} // si le ID>10 la longere de ID est 2 + longeure (:)
        else {j=2;}
        while (contenu[j]!=':') j++; // recupérer la postion du mdp


        strncpy( Dest: s, Source: contenu, Count: j); // copier contenu (ID+nomdumdp)
        printf( format: "\n donner le nouveau password : ");
        char *new=malloc( Size: max_size);
        scanf( format: "%s",new);
        char *new_encrypt=malloc( Size: strlen( Str: new));
        new_encrypt=encrypt( plaintext_password: new, KEY: THE_KEY);
        fprintf( stream: temp, format: "%s:%s\n",s,new_encrypt);  // ajouter le ID du password + nomdupassword+ mouveau mdp chiffré
}
else fprintf( stream: temp, format: "%s",contenu);// ajouter le restes de lignes (les autres passwords sans les modifier)
```

- procédure permet d'affichier un password a partir du ID

```c
// procédure permet d'affichier un password a partir du ID
void get_password(){

    printf( format: "\ndonner le ID du password :");
    int ID;
    scanf( format: "%d",&ID);
    FILE *file = fopen( Filename: "vault.txt",  Mode: "r");

if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n" );
          }
char *contenu=malloc( Size: max_size);
int i=0;
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
    if (i==0){i++;}
    else {
```

```c
    if (i==0){i++;}
    else {
        if(i==ID){
        int j;
        if (i>10 )  { j=3;}
        else {j=2;}
        while (contenu[j]!=':') j++;
        int k=0; j++;
        char *password =malloc( Size: strlen( Str: contenu)-j);

        while(j<strlen( Str: contenu)-1)
        {
            password[k]=contenu[j];
            j++;k++;
        }
        printf( format: "\nle password equivalent au ID %d est : %s",ID,decrypt( encrypted_password: password, KEY: THE_KEY));
        break;
```

- <u>procédure qui permet de supprimer un mot de passe a partir du son ID , copier tout les lignes du fichier vault.txt dans un fichier temp.txt sauf la ligne ayant le ID demandé , , après supprimer le fichier vault.txt et renommer temp.txt vers vault.txt (nouveau fichier avec la nouvelle modification).</u>

```c
// procédure qui permet de supprimer un mot de passe a partir du son ID
void delete_password(){
    printf( format: "\ndonner le ID du password :");
    int ID;
    scanf( format: "%d",&ID);
    FILE *file = fopen( Filename: "vault.txt",  Mode: "r");

if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n" );
          }
char *contenu=malloc( Size: max_size);
int i=0;
if (ID>0) {
FILE *temp=fopen( Filename: "temp.txt", Mode: "a");
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
```

```
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
if (i!=ID)  fprintf( stream: temp, format: "%s",contenu);// copier tout les lignes dans le fichier temp sauf la ligne ayant le ID asu
i++;
}

fclose( File: file);
fclose( File: temp);
remove( Filename: "vault.txt");
rename( OldFilename: "temp.txt", NewFilename: "vault.txt");
}
}
```

- procédure permet de générer des mots de passes randoms, Pour générer le mot de passe, nous avons besoin d'un endroit d'où nous pouvons saisir des nombres, des alphabets et des symboles aléatoires. On a utilisé srand() sert à initialiser le générateur. Cela ne doit être fait qu'une seule fois durant l'exécution du programme. Après avoir généré le mot de passe on doit le sauvegarder dans le fichier vault.txt

```
// procédure permet de générer des mots de passes randoms                    ⚠ 17  ⚠ 1
void generate_password(){
    int l ;
    printf( format: "\n\n **let's generate a password** ");
    printf( format: "\nDonner la longeure du mot de passse : ");
    scanf( format: "%d",&l);
    if (l<=0) { printf( format: "password length mus be >= 1 ");}
    char *password=malloc( Size: l+1);
    char *digits="0123456789"; // tableau des nombres
    char *lowers="abcdefghijklmnopqrstuvwxyz"; // tableau des caractéres miniscules
    char *uppers="ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // tableau des caractéres majucules
    char *symbols="!@#$%^&*()";// tableau des caractéres spéciaux
    srand( Seed: time( Time: NULL));
    for(int i=0;i<l;i++){
        int char_type=rand() % 4 ; // selection random soit un nombre soit majscule soit miniscule soit symboles
        if (char_type==0) password[i]=digits[rand() % strlen( Str: digits)];
        if (char_type==1) password[i]=lowers[rand() % strlen( Str: lowers)];
        if (char_type==2) password[i]=uppers[rand() % strlen( Str: uppers)];
```

```
printf( format: "\npassword generated is : %s ",password);
// stocker password dans fichiers
FILE *file = fopen( Filename: "vault.txt",  Mode: "r");
if (file == NULL) {
        // unable to open the file
        fprintf( stream: stderr,  format: "Error: unable to open file 'vault.txt' for writing.\n" );
          }
char *contenu=malloc( Size: max_size);
int k=0;
while(fgets( Buf: contenu, MaxCount: 256, File: file))
{
k++;
}
fclose( File: file);
FILE *f = fopen( Filename: "vault.txt",  Mode: "a"); // ouvrir le fichier pour ajouter le mot de passe généré
printf( format: "\ndonner un nom au password  :");
```

```c
scanf( format: "%s",name);
fprintf( stream: f, format: "%d:%s:%s\n",k,name,encrypt( plaintext_password: password, KEY: THE_KEY));
        fclose( File: f);

}
 // programme principale
int main() {
create_vault( filename: "vault.txt");
int choix=0;
int bol ;
char YN;
printf( format: "\nvoulez vous conecctez ? Y/N : ");
     scanf( format: "%c",&YN);

     if (YN=='Y' || YN=='y')
     {
        do {
```

4. captures de l'exécution :