**Programmation Orientée Objet – Prosit 2**

*Prosit Aller : 5/11/2021*

*Animateur : Théophile*

*Scribe : Paul*

*Secrétaire : Mattéo*

*Gestionnaire : Tom*

**Prendre connaissance de la situation et clarifier**

*Mots-clés/ à définir*

* **Cryptage :** Opération par laquelle un message est rendu inintelligible à quiconque ne possède pas la clé permettant de retrouver la forme initiale.
* **Décryptage :** procédé inverse du chiffrement.
* **Chiffrement :**procédé suivant lequel un document compréhensible de tous est transformé en un autre, incompréhensible.
* **Niveau de chiffrement :** C’est la complexité d’un chiffrement.
* **XOR :** La fonction OU exclusif, souvent appelée XOR ou disjonction exclusive, ou somme binaire en cryptographie où il est noté + est un opérateur logique de l'algèbre de Boole.
* **Type César :** En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César ou le code de César, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes.
* **Accroissement :**Fait de croître, d'augmenter.
* **Architecture :** En informatique, architecture désigne la structure générale inhérente à un système informatique, l'organisation des différents éléments du système et des relations entre les éléments.
* **Modéliser**
* **Compartimenter**
* **Accesseur :** Un accesseur est une fonction membre permettant de récupérer le contenu d'une donnée membre protégée. Un accesseur, pour accomplir sa fonction : doit avoir comme type de retour le type de la variable à renvoyer. Ne doit pas nécessairement posséder d'arguments.
* **Encapsulation :**L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données par un autre moyen que les services proposés.
* **Méthode**
* **Fichier source :**Les fichiers sources, ou documents de travail, sont ceux qui vous ont permis de construire le produit commandé par votre client.

*Établir le contexte*

On cherche à réaliser un programme qui a pour but de crypter/décrypter une chaîne de caractère inscrite dans un fichier texte.

**Analyser le besoin**

*Analyser et décomposer le problème*

* Comment crypter/décrypter une chaîne de caractère ?
* Comment utiliser un opérateur logique de type XOR ?
* Comment éviter la répétition d’un même code pour un même mode de chiffrement ?
* Quels moyens mettre en place afin de trouver la méthode/architecture à rajouter ?
* Comment lire et écrire le contenu d’un fichier texte ?

*Livrables*

* Diagramme UML
* Code de cryptage/décryptage
* Note explicative sur les différents types de chiffrement

*Contraintes*

* 3 modes de chiffrement (XOR, césar)

**Généraliser**

*Saisir la nature du problème et déterminer la famille de problèmes à laquelle il appartient*

Problème de sécurité (cryptage/décryptage/chiffrement)

Modification d’un fichier

**Trouver des pistes de solutions**

*Identifier des approches de solutions possibles*

* Le 3ème chiffrement sera-t-il un mélange de 2 classes (césar et XOR) ? Oui
* La classe du premier mode de cryptage aura-t-elle une hérédité avec la classe du second mode ? Oui
* Le code dans les ressources est mal structuré/faux.
* Le 3ème mode de sécurité aura un meilleur cryptage que les 2 premiers.

**Elaborer le plan d’actions**

*Identifier les éléments d’étude nécessaires à la résolution du problème*

* Manipulation des fichiers (lecture/écriture)
* Cryptage/décryptage (César et XOR)
* Architecture

*Identifier les éléments de réalisation à produire*

* Cf livrables

*Liste, organiser, planifier et formaliser les actions à mener pour résoudre le problème*

* Etudier les éléments d’étude (Le week-end)
* Comparer les différents modes de cryptage (7h)
* Produire un code permettant de lire/écrire dans un fichier (10h)
* Note explicative sur les différents types de chiffrement (1h)

**Phase AER :**

Nous allons dans un premier temps, étudier les types de cryptage que nous allons avons utiliser tout le long de ce prosit.

# Cryptage en César

Tout d’abord, nous allons commencer par définir ce qu’est un algorithme de cryptologie. C’est un procédé mathématique utilisé pour le chiffrement et le déchiffrement. Il décrit pas à pas toutes les étapes nécessaires à la transformation d’un document. Nous allons maintenant décrire le cryptage en César. Il faut savoir que, lors de ses batailles, l’empereur romain JULES CÉSAR cryptait les messages qu’il envoyait à ses généraux. Sa méthode de codage consistait à **décaler les lettres de 3 rangs**, vers la droite, dans l’alphabet. Cette méthode de chiffrage est appelée **chiffrement en César** ou **Code César**. Le nombre du rang de décalage des lettres est appelé **la clé**. La clé employée par ce type de chiffrement est donc 3 pour le Code César.

Par exemple, en prenant le mot « ENQUETE », nous avons :

|  |  |
| --- | --- |
| **Lettres cryptées** | **Lettres décryptées** |
| B | E |
| K | N |
| N | Q |
| R | U |
| B | E |
| Q | T |
| B | E |

Nous avons donc vu le code César et son fonctionnement, nous allons maintenant voir le deuxième type de cryptage que nous avons utilisé pour résoudre ce prosit.

# Cryptage en XOR

Le cryptage XOR est un système de cryptage basique mais pas très limité. Ainsi, il a beaucoup été utilisé dans les débuts de l'informatique et continue à l'être encore aujourd'hui car il est facile à implémenter, dans toutes sortes de programmes. Pour rappel, le XOR est un opérateur logique qui signifie « Exclusive OR » soit « OU Exclusif ». Lorsque nous faisons un test logique avec un XOR, le résultat est vrai lorsque nous avons une solution ou l’autre, mais jamais deux solutions en simultanée. Pour imager nos propos, nous allons voir la table de vérité du XOR :

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Résultat** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

En informatique, chaque caractère du message à coder est représenté par un entier, le code ASCII. Ce nombre est lui-même représenté en mémoire comme un nombre binaire à 8 chiffres (les bits). Nous choisissons une clé que nous plaçons en dessous du message à coder, en la répétant autant de fois que nécessaire. Le message et la clé étant converti en binaire, nous effectuons un XOR, bit par bit, le 1 représentant VRAI et le 0 FAUX. Le résultat en binaire peut être reconverti en caractères ASCII et donne alors le message codé.

Nous avons donc un algorithme symétrique car la même opération est appliquée pour retrouver le message initial. Cependant, ce type de cryptage possède un désavantage lorsque nous utilisons de long messages : nous pouvons aisément décrypter un message en étudiant les couples de lettres qui apparaissent le plus souvent dans le message chiffré, et en supposant qu’ils représentent les diagrammes les plus courants dans la langue (par exemple pour le français : 'es', 'en', 'on', 'ou', 'te', 'nt', et 'de').

Une image contenant table

Description générée automatiquement

# Corbeille

Pour résoudre ce prosit, nous avons eu à disposition une corbeille d’exercice. Nous allons donc expliquer notre travail. Pour le premier exercice, nous avions cet énoncé :

Une image contenant texte

Description générée automatiquement

Dans un premier temps, nous avons créer notre classe. Nous avons fait nos deux header pour définir nos classes. Le premier header que nous avons appelé corbeilleHeader.h qui va nous permettre d’importer les bons modules. Le deuxième qui va nous permettre de créer la classe que nous avons appelé Corbeille.h.

corbeilleHeader.h :

Une image contenant texte

Description générée automatiquement

Corbeille.h :

Une image contenant texte

Description générée automatiquement

Nous pouvons donc voir que dans la classe que nous avons créée, nous avons trois attributs privés.

* fichierEcriture
* fichierLecture
* message

Nous avons aussi ajouté deux méthodes.

* Lire qui prend en compte deux paramètres : le nom du fichier dans lequel nous allons écrire un message, et le message à écrire.
* Afficher qui prend en compte un seul paramètre : le nom du fichier dans lequel nous allons récupérer tout le texte qui s’y trouve et l’afficher sur la console pour que l’utilisateur puisse le lire.

Une image contenant texte

Description générée automatiquement

Nous faisons appelle à l’entête **fstream** qui va nous permettre de manipuler des fichiers. Il faut savoir que nous utilisons par défaut l’entête **iostream** qui signifie « **input/output stream** » c’est-à-dire les flux d’entrée et de sortie. Alors que **fstream** signifie « **file stream** » c’est-à-dire les flux de fichiers. Les flux que nous utilisons sont en réalité **des objets**. Pour cet exercice de la corbeille nous allons utiliser deux types de flux :

* Ofstream qui correspond à une classe d’écriture de fichiers.
* Ifstream qui correspond à une classe de lecture de fichiers.

Dans la méthode lire, nous cherchons à écrire dans un fichier. Nous avons donc mis en paramètre de cette méthode le nom du fichier. Le nom du fichier est rentré dans le main. Nous créons donc ensuite une variable de type ofstream que nous appelons fichier. Nous affectons à cette variable la méthode open qui va nous permettre d’ouvrir le fichier. Il faut savoir que les constructeurs ofstream et ifstreams ne prennent en paramètre que des const char\*, c\_str() renvoit ce type de valeur, c’est pour cela que nous l’appliquons à la variable fileName. Nous initialisons ensuite une variable de type string que nous appelons message, cette variable nous servira pour entrer le message que nous voulons inscrire dans le fichier texte. Nous mettons une condition if(fichier) pour vérifier que le fichier est bien ouvert. Nous mettons un else pour afficher message d’erreur dans la console dans le cas où le fichier ne s’ouvre pas. Une fois que nous sommes dans le if, nous demandons à l’utilisateur de rentrer le message qu’il veut inscrire dans le fichier texte. Pour cela, nous utilisons la fonction getline dans laquelle nous mettons en paramètre cin >> ws, message. Nous utilisons cette fonction et ces paramètres, car, si nous utilisons un cin présent dans le namespace std, lorsque l’utilisateur entre un message contenant des espaces, le cin ne prend en compte que le premier mot. Pour résoudre ce problème, comme nous l’avons dit plus tôt, nous utilisons la fonction getline avec pour paramètre cin >> ws. Le cin est permet d’annoncer que nous voulons lire des données entrer par l’utilisateur, et le ws signifie whitespace. En utilisant ce premier paramètre, nous prenons donc en compte les espaces dans la string et nous pouvons donc prendre en compte toute une phrase ou plus. Le deuxième paramètre message est juste la variable à laquelle nous allons affecter la string lue. Ensuite, nous inscrivons dans notre fichier, le message entrer. Bien-sûr à la fin de la méthode, nous fermons le fichier pour pas qu’il y ait de conflit dans le cas où nous le réutilisons.

Une image contenant texte

Description générée automatiquement

Nous avons ensuite fait la méthode afficher, nous y avons mis de nouveau le nom du fichier en paramètre. Cette fois-ci, nous déclarons une autre variable fichier mais en ifstream, car nous souhaitons lire le contenue du fichier en paramètre. Ensuite, nous ouvrons notre fichier. Dans le cas où nous avons réussi à ouvrir le fichier, nous créons une variable string ligne, à laquelle nous allons affecter le contenue du fichier à lire. Avec une boucle while, nous allons regarder toutes les lignes présentes dans le fichier et les afficher. Enfin dans le cas où nous n’avons pas réussi à ouvrir notre fichier, nous mettons un message d’erreur pour informer l’utilisateur que le fichier n’a pas été ouvert correctement. Puis nous fermons le fichier lorsque nous avons fini de lire tout le contenue.

Une image contenant texte

Description générée automatiquement

Pour finir avec ce premier exercice, nous avons fait une fonction Exo1() que nous appelons dans le main. Dans cette fonction, dans un premier temps, nous créons notre objet. Puis, nous créons les variables suivantes :

* originFile qui est le nom du fichier que nous allons manipuler.
* messageToAdd qui est le message que nous allons rentrer dans le fichier texte.

Nous avons aussi fait une variable choix, car nous avons créer un menu pour l’utilisateur. Il peut choisir ce qu’il veut faire.

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Voici donc le résultat que nous avons lorsque nous exécutons notre programme. Comme nous pouvons le voir, dans un premier temps, nous avons cherché à écrire dans notre fichier. Nous avons donc rentré le nom de notre fichier texte et nous avons ensuite rentrer le message que nous voulions mettre dans le fichier texte. Puis nous avons fait le choix d’afficher le texte du fichier texte.

Pour la suite de la corbeille nous n’allons pas expliquer car le code sera commenté.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

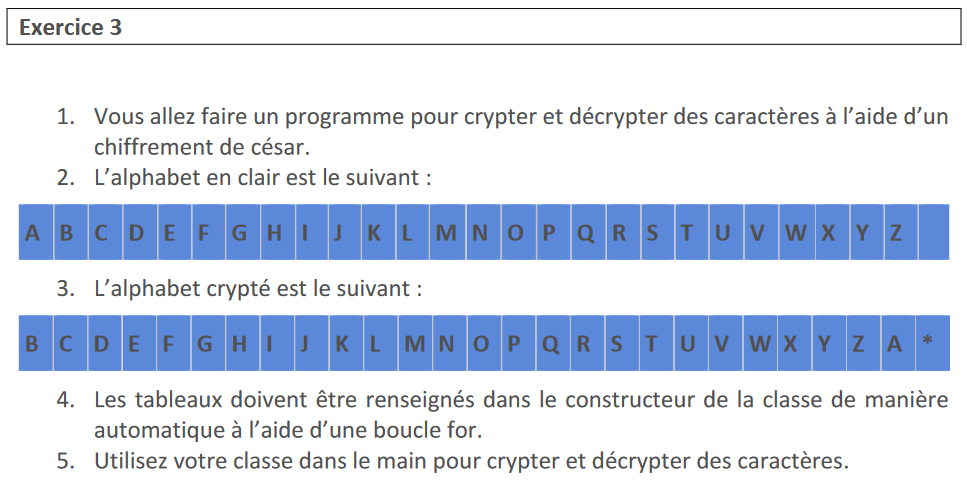
Description générée automatiquement

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Exercice 3 :



Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Exercice 4 :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Concernant l’exercice 5, comme nous avons respecter à la lettre les diagrammes qui nous étaient imposer, nous ne mettrons pas de commentaire.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Maintenant que nous avons vu la corbeille, nous allons passer à la résolution du prosit. Dans un premier temps, nous avons fait deux classes. Une première pour le cryptage en César et une deuxième pour le cryptage en XOR. Ces deux classes, nous avons mis dans le namespace Classes. **Il faut savoir qu’un namespace, ou espace de nom (parfois aussi espace de nommage, voire référentiel lexical) est une zone de déclaration d'identificateurs permettant au compilateur de résoudre les conflits de noms.**

Nous avons ensuite créé un deuxième namespace que nous avons appeler programme. Dans ce namespace, nous y avons mis la classe prosit, qui à pour méthode principale la méthode run et pour attribue des objet découlant des classes Cesar et XOR.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Dans le fichier Classe.cpp, nous avons déclarer toutes nos méthodes. Comme nous le voyons, nous avons utilisé des namespace. Il faut donc faire appelle aux namespace pour ensuite faire appelle aux classes puis au méthodes. Nous avons commencé par créer le constructeur de la classe Cesar. Ce constructeur permet de d’intialiser les attribues de la classe Cesar comme nous pouvons le voir.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Pour les classes Cesar et XOR, nous avons repris le code de la corbeille, nous n’allons donc pas les décrire.

Une image contenant texte

Description générée automatiquement

Nous allons maintenant passer à la fonction run qui est une méthode de la classe prosit qui se trouve dans le namespace programme. Dans cette fonction, nous avons dans un premier temps, initialiser nos deux objets, à savoir notre writingFile qui est le fichier dans lequel nous allons écrire le message crypter et notre readingFile qui est le fichier dans lequel nous allons lire le message de base. Nous avons aussi initialisé une variable int qui est choixMode, cette variable va nous servir pour le menu. Comme nous le savons, nous devons offrir la possibilité à l’utilisateur de choisir entre plusieurs modes de cryptage. Nous avons aussi initialisé deux autres variables de types string, elles correspondent aux noms des fichiers que nous allons utiliser. Nous avons ensuite affiché le menu que nous proposons à l’utilisateur. Puis nous lui demandons de rentrer le nom des fichiers qu’il va utiliser.

Une image contenant texte

Description générée automatiquement

Ensuite, si l’utilisateur choisi d’utiliser le cryptage en Cesar. Dans un premier temps, nous initialisons une variable string message qui sera le message à crypter. Puis nous ouvrons les deux fichiers à utiliser. Si les fichiers sont bien ouverts, alors nous lisons le message qui se trouve dans le fichier à lire. Puis, une fois que nous avons récupérer l’intégralité de notre message, nous l’encryptons. Puis nous refermons l’accès aux deux fichiers.

Une image contenant texte

Description générée automatiquement

Pour le code en XOR, nous faisons le même code, à la seule différence que nous récupérons chaque lettres du message et nous les transformons en binaire.

Une image contenant texte

Description générée automatiquement

Pour le troisième mode, nous combinons les deux codes.

Une image contenant texte

Description générée automatiquement

Nous avons aussi rajouté une condition qui fait que si le mode entré de base n’existe pas, nous affichons un message d’erreur et nous demandons à ce que l’utilisateur rentre une nouvelle fois le numéro du mode qu’il veut utiliser.

Une image contenant texte

Description générée automatiquement