

Compte rendu C++ TP4

Introduction :

Nous avons choisi ce TP car ayant commencé par un projet de niveau débutant qui avait pour but de se familiariser avec le langage C++, il était logique de monter d'un cran au-dessus afin d'approfondir nos connaissances.

Ce TP porte sur le cryptage de données et le but étant d'encoder/décoder un message avec n'importe quel type de chiffrement. Ce TP a pour objectif principal de nous familiariser avec le principe d'héritage.

Concernant la répartition du travail, une personne a fait la classe Encrypt et du code concernant la méthode Vigenere et l'autre a fait les deux codes Cesar.

Techniques et problème :

Concernant les classes, Cesar, Caesar2 et Vigenere sont des classes héritant de la classe Encrypt.

On nous a demandé de créer des fonctions virtuelles dans la classe mère afin de pouvoir les réécrire dans les classes filles :

```
virtual void encode() =0;  
virtual void decode() =0;
```

Code Cesar :

Nous avons décidé d'utiliser un type string pour contenir l'alphabet car il est possible d'en sortir une lettre en particulier.

La partie encode est constitué de 2 tests pour chaque lettre :

```

while (test != 1)
{
    if (decompte > 25)
    {
        _plain = _plain + _cypher[i];
        test = 1;
    }
    else if (_cypher[i] == _Alphabet[decompte])
    {
        decompte = (decompte - _nombre + 26) % 26;
        _plain = _plain + _Alphabet[decompte];
        test = 1;
    }

    decompte += 1;
}

```

Ici la boucle while s'arrête si une des 2 conditions est remplie. La première est que decompte est supérieur à 25 donc que le caractère n'est pas dans l'alphabet. Alors on le réécrit juste dans la partie décodée. La seconde est si l'on a une correspondance de caractère. Alors on applique le décalage que l'on remet au modulo 26 soit le nombre de lettres dans l'alphabet.

Pour encode on change juste le signe de decompte afin d'effectuer un décalage dans l'autre sens.

Code Caesar2 :

On fait juste comme la classe Cesar mais avec un modulo 127 correspondant au nombre de caractère dans le code ASCII.

Code Vigenere :

```

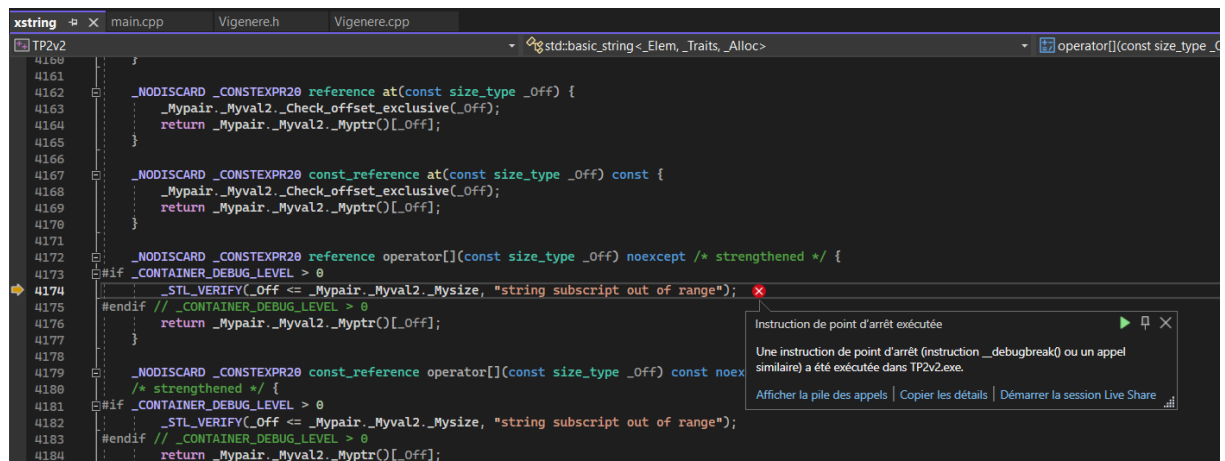
for (i = 0; i < l; i++)
{
    while (test != 1)
    {
        if (decompte > 25)
        {
            _cypher = _cypher + _plain[i];
            test = 1;
        }
        else if (_plain[i] == _Alphabet[decompte])
        {
            decompte = (decompte + _cle[i % _cle.size()]) % 26;
            _cypher = _cypher + _Alphabet[decompte];
            test = 1;
        }

        decompte += 1;
    }
}

```

Le seul changement par rapport au code Cesar est le décalage qui n'est pas constant. Donc j'ai stocké cette clé dans un vector. Ensuite il suffisait juste de parcourir le vecteur pour appliquer le décalage à chaque lettre comme dans les autres méthodes de cryptage.

Cependant nous avons rencontré un problème pour la méthode Vigenere. En effet, lorsqu'il fallait changer le contenu de la clé afin qu'elle soit une chaîne de caractère, en lançant le programme nous tombions sur une erreur incompréhensible et nous n'avions pas trouvé de solution.



```
4160 }
4161
4162 _NODISCARD _CONSTEXPR20 reference at(const size_type _Off) {
4163     _Mypair._Myval2._Check_offset_exclusive(_Off);
4164     return _Mypair._Myval2._Myptr()[_Off];
4165 }
4166
4167 _NODISCARD _CONSTEXPR20 const_reference at(const size_type _Off) const {
4168     _Mypair._Myval2._Check_offset_exclusive(_Off);
4169     return _Mypair._Myval2._Myptr()[_Off];
4170 }
4171
4172 _NODISCARD _CONSTEXPR20 reference operator[](const size_type _Off) noexcept /* strengthened */ {
4173     #if _CONTAINER_DEBUG_LEVEL > 0
4174         _STL_VERIFY(_Off <= _Mypair._Myval2._Mysize, "string subscript out of range");
4175     #endif // _CONTAINER_DEBUG_LEVEL > 0
4176     return _Mypair._Myval2._Myptr()[_Off];
4177 }
4178
4179 _NODISCARD _CONSTEXPR20 const_reference operator[](const size_type _Off) const noexcept /* strengthened */ {
4180     #if _CONTAINER_DEBUG_LEVEL > 0
4181         _STL_VERIFY(_Off <= _Mypair._Myval2._Mysize, "string subscript out of range");
4182     #endif // _CONTAINER_DEBUG_LEVEL > 0
4183     return _Mypair._Myval2._Myptr()[_Off];
4184 }
```

Erreur après exécution du programme

Conclusion :

Même si le projet n'a pas été complètement fini comme il a été demandé, cela nous a été très enrichissant surtout pour un membre du groupe qui a commencé le langage C++ cette année. De plus, ce projet nous a permis de comprendre le principe d'héritage et ainsi d'approfondir nos connaissances sur ce langage.