Piscine C++ - d17

Algorithm

Alexandre "Tipiak" BOSSARD bossar_a@epitech.eu

Abstract: *Ce document est le sujet du d17 : Les algorithmes de la STL*

Table des matières

Chapitre I

REMARQUES GÉNÉRALES

- LISEZ LES REMARQUES GÉNÉRALES ATTENTIVEMENT !!!!!
  - Vous n'aurez aucune excuse si vous avez 0 parce que vous avez oublié une consigne générale ...

- REMARQUES GÉNÉRALES :
  - La STL est un ensemble de choses à voir. Donc faites tous les exos de la journée. Ca N'EST PAS progressif. Donc ne soyez pas stupides et faites toute la journée, la STL est un des concepts indispensables du C++
  - Les classes doivent être canoniques (Comprendre : on teste que les classes sont bien canoniques. Si ce n'est pas le cas, 0 à l'exercice).
  - Le C n'est pas d'actualité. Toutes les fonctions `[asvn]printf, *alloc` sont interdites.

- COMPILATION DES EXERCICES :
  - La moulinette compile votre code avec les flags : `-W -Wall -Werror`
  - Pour éviter les problèmes de compilation de la moulinette, incluez les fichiers nécéssaires dans vos fichiers include (*.hh).
  - Notez bien qu'aucun de vos fichiers ne doit contenir de fonction `main` . Nous utiliserons notre propre fonction `main` pour compiler et tester votre code.
  - Le repértoire de rendu est : `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/exN` (N étant bien sur le numéro de l'exercice).

Chapitre II

Exercice 0

	Exercice : 00	points : 4
Trouvez-moi ca		
Répertoire de rendu: `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/ex00`		
Compilateur : `g++`	Flags de compilation: `-W -Wall -Werror`	
Makefile : `Non`	Règles : `n/a`	
Fichiers a rendre : `find.hpp`		
Remarques : `n/a`		
Fonctions Interdites : `*alloc - free - *printf`		

Pour cet exercice, vous devez créer une fonction template `do_find` prenant 2 paramétres :

- Une référence sur le type template
- Un entier

Cette fonction devra chercher l'entier passé en paramètre dans le conteneur également passé en paramètre. S'il existe au moins une occurrence de cet entier dans le conteneur, un `iterator` sur la première occurrence sera retourné. Sinon, un `iterator` sur la fin sera retourné (cf. `end()`).

Vous devez utiliser l'algorithme `find` de la STL pour résoudre cet exercice.

Postulat : le type template sera toujours un conteneur STL contenant des entiers, et sera compatible avec l'algorithme find de la STL.

`Résoudre des symboles définis dans un type template n'est pas toujours possible pour votre compilateur. 'typename' peut se reveler utile.`

`Le code de la fonction est trivial, une ligne devrait suffir.`

Chapitre III

Exercice 1

	Exercice : 01	points : 6
STL Algorithmsssssssss		
Répertoire de rendu: `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/ex01`		
Compilateur : `g++`	Flags de compilation: `-W -Wall -Werror`	
Makefile : `Non`	Règles : `n/a`	
Fichiers a rendre : `MyAlgorithms.hpp`		
Remarques : `n/a`		
Fonctions Interdites : `*alloc - free`		

Le but de cet exercice est de vous faire fouiller les algorithmes de la STL afin que vous puissez vous familiariser avec.

L'exercice est simple, mais nécessite que vous vous documentiez un peu.

Nous vous fournissons un fichier `MyAlgorithms.hpp` qui contient des fonctions à trou. Vous devez complèter ces trous par l'algorithme de la STL le plus approprié. Le but étant que le tout compile et produise un résultat similaire et cohérent avec le main d'exemple ci dessous.

`Vous ne devez utiliser QUE les algorithmes de la STL pour résoudre cet exercice, vous NE DEVEZ PAS recoder vous même ces comportements, simplement faire appel au bon algorithme !`

Vous NE DEVEZ PAS modifier le prototype des fonctions, simplement remplacer les XXXXXXXXXX par le bon appel !

***** Exemple *****

Vous trouverez le main exemple en fichier fourni. Attention, ce main n'est pas celui qui sera utilisé pour vérifier votre rendu (logique).

***** Sortie *****

```
Thefly$> g++ -W -Werror -Wall main.cpp -o test
Thefly$> ./test | cat -e
============ Step 01 ==========$
Dump (11)    4,    9,    1,    1,   99,    8,   42,   42,   42,   -1,    3, $
Dump (10)   99,    1,  -42,   21,   12,   21,   99,   -7,   42,   42, $
============ Step 02 ==========$
3$
0$
============ Step 03 ==========$
true$
false$
false$
true$
============ Step 04 ==========$
Dump (11)    4,    9,    1,    1,   99,    8,   42,   42,   42,   -1,    3, $
Dump (11)    4,    9,    1, -421, -421,    8,   42,   42,   42,   -1,    3, $
============ Step 05 ==========$
Dump (11)    4,    9,    1, -421, -421,    8,   42,   42,   42,   -1,    3, $
Dump (11)    4,   18,    2, -842, -842,   16,   84,   84,   84,   -2,    3, $
============ Step 06 ==========$
Dump (11)    4,   18,    2, -842, -842,   16,   84,   84,   84,   -2,    3, $
Dump ( 8)    4,   18,    2, -842, -842,   16,   -2,    3, $
============ Step 07 ==========$
Dump (10)   99,    1,  -42,   21,   12,   21,   99,   -7,   42,   42, $
Dump (10)   42,   42,   -7,   99,   21,   12,   21,  -42,    1,   99, $
============ Step 08 ==========$
Dump (10)   42,   42,   -7,   99,   21,   12,   21,  -42,    1,   99, $
Dump ( 9)   42,   -7,   99,   21,   12,   21,  -42,    1,   99, $
============ Step 09 ==========$
Dump ( 8)    4,   18,    2, -842, -842,   16,   -2,    3, $
Dump ( 8) -842, -842,   -2,    2,    3,    4,   16,   18, $
============ Step 10 ==========$
Dump ( 9)   42,   -7,   99,   21,   12,   21,  -42,    1,   99, $
Dump ( 9)   99,   99,   42,   21,   21,   12,    1,   -7,  -42, $
============ Step 11 ==========$
Dump ( 9)   99,   99,   42,   21,   21,   12,    1,   -7,  -42, $
Dump ( 9)   21,   21,   12,    1,   -7,  -42,   99,   99,   42, $
Dump ( 9)   99,   42,   21,   21,   12,    1,   -7,  -42,   99, $
============ Step 12 ==========$
Dump ( 9)   99,   42,   21,   21,   12,    1,   -7,  -42,   99, $
Dump ( 9)   99,   42,  777,  777,   12,    1,   -7,  -42,   99, $
============ Step 13 ==========$
Dump ( 9)   99,   42,  777,  777,   12,    1,   -7,  -42,   99, $
```

```
Dump ( 7)   99,   42,   12,    1,   -7,  -42,   99, $
============ Step 14 ==========$
Dump ( 8) -842, -842,   -2,    2,    3,    4,   16,   18, $
Dump ( 7)  -42,   -7,    1,   12,   42,   99,   99, $
Dump (15)    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    \
0,    0,    0,    0, $
Dump (15) -842, -842,  -42,   -7,   -2,    1,    2,    3,    4,   12,   16,   1\
8,   42,   99,   99, $
Thefly$>
```

Aidez-vous du nom des fonctions ainsi que de la sortie produite pour devinez de quel algorithme il s'agit.

Chapitre IV

Exercice 2

	Exercice : 02	points : 5
Ave Cesar & One time pad		
Répertoire de rendu: `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/ex02`		
Compilateur : `g++`	Flags de compilation: `-W -Wall -Werror`	
Makefile : `Non`	Règles : `n/a`	
Fichiers a rendre : `Cesar.cpp, Cesar.h, OneTime.cpp, OneTime.h`		
Remarques : `n/a`		
Fonctions Interdites : `*alloc - free - *printf`		

Fichiers fournis :

- IEncryptionMethod.h

IV.1 Partie 1/2 : Chiffre de César

Le but de cette partie est de chiffrer et de déchiffrer des messages envoyés par César.

Pour faire cela, vous allez créer une classe `Cesar` implémentant l'interface `IEncryptionMethod` qui vous est fourni dans le fichier `IEncryptionMethod.h`.
Nous compilerons vos sources en copiant ce fichier dans le même dossier, vous pouvez donc en faire un include, mais pas le modifier !

Description des méthodes :

```
1  // Chiffre le caractere passe en parametre et l'affiche.
2  void encryptChar(char plainchar);
3
4  // Dechiffre le caractere passe en parametre et l'affiche.
5  void decryptChar(char cipherchar);
6
7  // Reunitialise les champs internes a leur etat initial.
8  void reset();
```

Nous utiliserons une version légèrement modifiée de l'algorithme de César.
L'algorithme de César consiste à décaler chaque lettre de 3 lettres vers la droite pour chiffrer, et de 3 lettres vers la gauche pour déchiffrer.

Par exemple :

- a devient d
- B devient E
- z devient c // Et oui, après 'z' on revient à 'a' !

Comme vous le voyez, on conserve la casse.

Les caractères non alphabétiques seront affichés sans etre modifiés.
Nous modifierons cet algorithme pour décaler d'une lettre de plus à chaque caractère passé à la fonction `encryptChar` ou `decryptChar`.

Le premier caractère sera décalé de 3, le deuxième de 4, et ainsi de suite. Le 27ème caractère sera donc logiquement décalé de 29. N'est ce pas ? Mais l'alphabet ne contient que 26 caractères, donc un décalage de 29 lettres est strictement équivalent à un décalage de 3 lettres ! (ca fait une boucle !)

La mèthode `reset` permettra de réinitialiser le décalage comme si aucun caractère n'avait été chiffré ou déchiffré (donc un décalage de 3 lettres !).
La méthode `decryptChar` fera l'inverse de la méthode `encryptChar` , elle transformera un caractère chiffré en caractère déchiffré.

Un char ne pouvant pas dépasser la valeur 127, faites attention lors de vos calculs ! Faites également attention lors des soustractions si sa valeur descend dans les négatifs !

***** Exemple *****

```
#include ‘‘Cesar.h’’
#include <string>
#include <iostream>

static void encryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toEncrypt)
{
  size_t len = toEncrypt.size();

  encryptionMethod.reset();
  for (size_t i = 0; i < len; ++i)
  {
    encryptionMethod.encryptChar(toEncrypt[i]);
  }
  std::cout << std::endl;
}

static void decryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toDecrypt)
{
  size_t len = toDecrypt.size();

  encryptionMethod.reset();
  for (size_t i = 0; i < len; ++i)
  {
    encryptionMethod.decryptChar(toDecrypt[i]);
  }
  std::cout << std::endl;
}

int main()
{
  Cesar c;

  encryptString(c, ‘‘Je clair Luc, ne pas ? Ze woudrai un kekos !’’);
  decryptString(c, ‘‘Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !’’);
  encryptString(c, ‘‘KIKOO’’);
  encryptString(c, ‘‘LULZ XD’’);
  decryptString(c, ‘‘Ziqivun ea Ndcsg.Wji !’’);
  return 0;
}
```

***** Sortie *****

```
Thefly$> g++ -W -Werror -Wall Cesar.cpp main.cpp -o test
Thefly$> ./test | cat -e
Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !$
Je clair Luc, ne pas ? Ze woudrai un kekos !$
```

```
NMPUV$
OYQF FM$
Welcome to Zombo.Com !$
Thefly$>
```

IV.2 Partie 2/2 : One Time Pad

Le but de cette partie est de chiffrer et de déchiffrer des messages avec l'algorithme du `One time pad.`

Pour plus d'information sur cet algorithme :
http://en.wikipedia.org/wiki/One-time_pad

Il s'agit donc d'additionner (pour chiffrer) ou de soustraire (pour déchiffrer) de facon répétitive une clef à votre message. Vous ne chiffrerez que les caractères alphabètiques, mais passerez au caractere suivant de votre clef quelque soit le caractère rencontré. La première lettre de l'alphabet est à la position 0.

La clef sera toujours composée exclusivement de caractères alphabètiques.

Par exemple, chiffrer "A bah zour !" avec "bD" donnera "B cdi arvu !" (on conserve la casse du message, la casse de la clef est sans importance)

Pour faire cela, vous allez créer une classe `OneTime` implémentant l'interface `IEncryptionMethod` qui vous est fourni dans le fichier `IEncryptionMethod.h`. Nous compilerons vos sources en copiant ce fichier dans le même dossier, vous pouvez donc en faire un `include` , mais pas le modifier !

Description des méthodes :

```
// Chiffre le caractere passe en parametre et l'affiche.
void encryptChar(char plainchar);

// Dechiffre le caractere passe en parametre et l'affiche.
void decryptChar(char cipherchar);

// Reunitialise les champs internes a leur etat initial.
void reset();
```

OneTime ne disposera pas de constructeur par défaut, mais d'un constructeur prenant la clef en paramètre :

```
OneTime(std::string const& key);
```

La méthode reset permettra de réinitialiser la clef à sa position initiale.

un char ne pouvant pas dépasser la valeur 127, faites attention lors de vos calculs ! Faites également attention lors des soustractions si sa valeur descend dans les négatifs !

***** Exemple *****

```
#include ‘‘Cesar.h’’
#include ‘‘OneTime.h’’
#include <string>
#include <iostream>
static void encryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toEncrypt)
{
  size_t len = toEncrypt.size();

  encryptionMethod.reset();
  for (size_t i = 0; i < len; ++i)
  {
    encryptionMethod.encryptChar(toEncrypt[i]);
  }
  std::cout << std::endl;
}

static void decryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toDecrypt)
{
  size_t len = toDecrypt.size();

  encryptionMethod.reset();
  for (size_t i = 0; i < len; ++i)
  {
    encryptionMethod.decryptChar(toDecrypt[i]);
  }
  std::cout << std::endl;
}

int main()
{
  Cesar c;
  OneTime o(‘‘DedeATraversLesBrumes’’);
  OneTime t(‘‘TheCakeIsALie’’);

  encryptString(c, ‘‘Je clair Luc, ne pas ? Ze woudrai un kekos !’’);
  decryptString(c, ‘‘Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !’’);
  encryptString(c, ‘‘KIKOO’’);
  encryptString(c, ‘‘LULZ XD’’);
  decryptString(c, ‘‘Ziqivun ea Ndcsg.Wji !’’);

  encryptString(t, ‘‘Prend garde Lion, ne te trompes pas de voie !’’);
  encryptString(o, ‘‘De la musique et du bruit !’’);
  encryptString(t, ‘‘Kion li faras? Li studas kaj programas!’’);

  decryptString(t, ‘‘Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg !’’);
  decryptString(o, ‘‘Gi pa dunmhmp wu xg tuylx !’’);
  decryptString(t, ‘‘Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!’’);
```

```
50
51   return 0;
52 }
```

***** Sortie *****

```
Thefly$> g++ -W -Werror -Wall Cesar.cpp OneTime.cpp main.cpp -o test
Thefly$> ./test | cat -e
Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !$
Je clair Luc, ne pas ? Ze woudrai un kekos !$
NMPUV$
OYQF FM$
Welcome to Zombo.Com !$
Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg !$
Gi pa dunmhmp wu xg tuylx !$
Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!$
Prend garde Lion, ne te trompes pas de voie !$
De la musique et du bruit !$
Kion li faras? Li studas kaj programas!$
Thefly$>
```

Chapitre V

Exercice 3

	Exercice : 03	points : 6
STL algorithms are magical		
Répertoire de rendu: `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/ex03`		
Compilateur : `g++`	Flags de compilation: `-W -Wall -Werror`	
Makefile : `Non`	Règles : `n/a`	
Fichiers a rendre : `Cesar.cpp, Cesar.h, OneTime.cpp, OneTime.h, Encryption.cpp, Encryption.h`		
Remarques : `n/a`		
Fonctions Interdites : `*alloc - free - *printf`		

V.1 Partie 1/2 : Encryption Wrapper

Reprenez tous vos fichiers de l'exercice précédent, et écrivez la classe `Encryption` de facon à faire compiler et fonctionner correctement l'exemple suivant.

***** Exemple *****

```
#include ‘‘Encryption.h’’
#include ‘‘Cesar.h’’
#include ‘‘OneTime.h’’
#include <string>
#include <iostream>

static void encryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toEncrypt)
{
  Encryption e(encryptionMethod, &IEncryptionMethod::encryptChar);

  encryptionMethod.reset();
  size_t len = toEncrypt.size();
  for (size_t i = 0; i < len; ++i)
  {
    e(toEncrypt[i]);
  }
  std::cout << std::endl;
}

static void decryptString(IEncryptionMethod& encryptionMethod,
                          std::string const& toDecrypt)
{
  Encryption e(encryptionMethod, &IEncryptionMethod::decryptChar);

  encryptionMethod.reset();
  size_t len = toDecrypt.size();
  for (size_t i = 0; i < len; ++i)
  {
    e(toDecrypt[i]);
  }
  std::cout << std::endl;
}
int main()
{
  Cesar c;
  OneTime o(‘‘DedeATraversLesBrumes’’);
  OneTime t(‘‘TheCakeIsALie’’);

  encryptString(c, ‘‘Je clair Luc, ne pas ? Ze woudrai un kekos !’’);
  decryptString(c, ‘‘Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !’’);
  encryptString(c, ‘‘KIKOO’’);
  encryptString(c, ‘‘LULZ XD’’);
  decryptString(c, ‘‘Ziqivun ea Ndcsg.Wji !’’);

  encryptString(t, ‘‘Prend garde Lion, ne te trompes pas de voie !’’);
  encryptString(o, ‘‘De la musique et du bruit !’’);
  encryptString(t, ‘‘Kion li faras? Li studas kaj programas!’’);

```

```
50  decryptString(t, ‘‘Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg !’’);
51  decryptString(o, ‘‘Gi pa dunmhmp wu xg tuylx !’’);
52  decryptString(t, ‘‘Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!’’);
53
54  return 0;
55 }
```

***** Sortie *****

```
1  Thefly$> g++ -W -Werror -Wall Cesar.cpp OneTime.cpp Encryption.cpp main.cpp \
2          -o test
3  Thefly$> ./test | cat -e
4  Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !$
5  Je clair Luc, ne pas ? Ze woudrai un kekos !$
6  NMPUV$
7  OYQF FM$
8  Welcome to Zombo.Com !$
9  Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg !$
10 Gi pa dunmhmp wu xg tuylx !$
11 Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!$
12 Prend garde Lion, ne te trompes pas de voie !$
13 De la musique et du bruit !$
14 Kion li faras? Li studas kaj programas!$
15 Thefly$>
```

V.2 Partie 2/2 : STL algorithms are magical

Rajoutez dans la classe `Encryption` 2 fonctions membres static:

```
1 static void encryptString(IEncryptionMethod& encryptionMethod,
2                           std::string const& toEncrypt);
3 static void decryptString(IEncryptionMethod& encryptionMethod,
4                           std::string const& toDecrypt);
```

Celles-ci devront avoir le même comportement que les fonctions du même nom définies dans l'exemple ci-dessus, mais ne devront contenir que trois lignes de codes :

- Une pour reset la classe de chiffrement ;
- Une pour appeler l'algorithme de la STL correspondant ;
- Une pour afficher le retour à la ligne.

Vous devez IMPÉRATIVEMENT utiliser un algorithme de la STL pour valider cet exercice.

***** Exemple *****

```
1 #include ‘‘Encryption.h’’
2 #include ‘‘Cesar.h’’
3 #include ‘‘OneTime.h’’
4 #include <string>
5 #include <iostream>
6
7 int main()
8 {
9   Cesar c;
10  OneTime o(‘‘DedeATraversLesBrumes’’);
11  OneTime t(‘‘TheCakeIsALie’’);
12
13  Encryption::encryptString(c, ‘‘Je clair Luc, ne pas ? Ze woudrai un kekos
        !’’);
14  Encryption::decryptString(c, ‘‘Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk
        !’’);
15  Encryption::encryptString(c, ‘‘KIKOO’’);
16  Encryption::encryptString(c, ‘‘LULZ XD’’);
17  Encryption::decryptString(c, ‘‘Ziqivun ea Ndcsg.Wji !’’);
18
19  Encryption::encryptString(t, ‘‘Prend garde Lion, ne te trompes pas de voie
        !’’)\
20 ;
21  Encryption::encryptString(o, ‘‘De la musique et du bruit !’’);
22  Encryption::encryptString(t, ‘‘Kion li faras? Li studas kaj programas!’’);
23
24  Encryption::decryptString(t, ‘‘Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg
        !’’)\
25 ;
26  Encryption::decryptString(o, ‘‘Gi pa dunmhmp wu xg tuylx !’’);
27  Encryption::decryptString(t, ‘‘Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!’’);
28
29  return 0;
30 }
```

***** Sortie *****

```
1 Thefly$> g++ -W -Werror -Wall Cesar.cpp OneTime.cpp Encryption.cpp main.cpp \
2         -o test
3 Thefly$> ./test | cat -e
4 Mi isirb Xhq, ew jvo ? Zf zszjyir fz ytafk !$
5 Je clair Luc, ne pas ? Ze woudrai un kekos !$
6 NMPUV$
7 OYQF FM$
8 Welcome to Zombo.Com !$
9 Iyipd kijdp Pbvr, xi le bvhttgs tik om ovmg !$
10 Gi pa dunmhmp wu xg tuylx !$
11 Dpsp vm xaciw? Pk cxcvad otq rrykzsmla!$
```

```
Prend garde Lion, ne te trompes pas de voie !$
De la musique et du bruit !$
Kion li faras? Li studas kaj programas!$
Thefly$>
```

Chapitre VI

Exercice 4

	Exercice : 04	points : 4
Le MetaConteneur		
Répertoire de rendu: `(DÉPOT SVN - piscine_cpp_d17-promo-login_x)/ex04`		
Compilateur : `g++`	Flags de compilation: `-W -Wall -Werror`	
Makefile : `Non`	Règles : `n/a`	
Fichiers a rendre : `Container.hpp`		
Remarques : `n/a`		
Fonctions Interdites : `*alloc - free`		

Pour cet exercice vous pouvez utiliser les algorithmes, mais c'est plus pour le fun.

Vous allez devoir faire un MetaConteneur.

Faites une classe `contain` templatée pour recevoir en paramètre template le type qui va être stocké dans le conteneur stl, et en deuxieme paramètre le conteneur STL.

`Vous ne stockerez que des types scalaires.`

Les attributs de votre classe `contain` seront :

- le conteneur stl passé en paramètre template avec comme contenu le type passé en premier paramètre de votre template de classe.

Vous allez ensuite faire 2 fonctions membres :

Ces deux fonctions membres seront :

- `void aff()` celle ci doit parcourir le conteneur grace à `for_each` et appeler la fonction `aff` décrite en dessous

- `void add()` celle ci doit parcourir le conteneur grace à `for_each` et appeler la fonction `add` décrite en dessous

Vous allez ensuite coder 2 fonctions template, qui prennent en paramètre template le type que le conteneur contient.

- `void aff(?? b)` celle ci affiche "Valeur : 7" 7 est la valeur passée en paramètre soit b. Un saut de ligne doit etre present a la fin.
- `void add(?? a)` celle ci ajoutera +1 à la valeur passée en paramètre, elle devra modifier la valeur contenue dans le conteneur

Exemple :

```
int main()
{
 contain<char, std::list> test;
 test.push('t');
 test.aff();
 test.add();
 test.aff();
 contain <int, std::vector> test2;
 test2.push(1);
 test2.aff();
 test2.add();
 test2.aff();
 return 0;
}
```

Sortie :

```
thefly_d$>./a.out
Valeur : t
Valeur : u
Valeur : 1
Valeur : 2
thefly_d$>
```

Chapitre VII

Exercice 5

Si vous avez réussi à finir tous les exercices de la journée et qu'il vous reste un peu de temps avant l'exam de ce soir, voici quelques suggestions :

- Retestez tout au moins 3 fois ;
- Révisez vos notions pour l'examen de ce soir ;
- Revoyez tous les exercices des jours précédents et refaites ceux que vous n'avez pas réussi à faire ;
- Etes-vous certains d'avoir acquis parfaitement les notions vues en piscine, entrainez-vous à expliquer chacune de ces notions. Si vous n'arrivez pas à en expliquer une, c'est que vous l'avez mal comprise ! (essayez, vous allez avoir des surprises !)
- Si vous êtes arrivé à ce point, c'est que vous n'êtes pas honnête avec vous même, reprenez les points précédents !
- Vous pouvez toujours vous amuser à pousser le C++ au bout de ses limites, jeter un coup d'oeil aux futures évolutions apportées par le C++0x, et pourquoi pas vous amuser à faire un peu de meta-programmation ?