





Piscine C++ - d14m
The Cast

Francois-Xavier "FX" BOURLET bourle_f@epitech.eu

Abstract: Ce document est le sujet du d14m





Table des matières

Ι	REMARQUES GÉNÉRALES	4
II	Exercice 0	4
III	Exercice 1	7
IV	Exercice 2	8
\mathbf{V}	Exercice 3	10
VI	Exercice 4	12



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 :
 - Si vous faites la moitié des exercices car vous avez du mal, c'est normal. Par contre, si vous faites la moitié des exercices par flemme et vous tirez à 14h, vous AUREZ des surprises. Ne tentez pas le diable.
 - Toute fonction implémentée dans un header ou header non protègé signifie 0 à l'exercice.
 - o Toutes les classes doivent posséder un constructeur et un destructeur.
 - o Toutes les sorties se font sur la sortie standard et sont terminées par un retour à la ligne sauf si le contraire est precisé explicitement.
 - Les noms de fichiers qui vous sont imposés doivent être respectés À LA LETTRE, de même que les noms de classes et de fonctions membres / méthodes.
 - Rappelez-vous que vous faites du C++ et non plus du C. Par conséquent, les fonctions suivantes sont INTERDITES, et leur utilisation sera sanctionnée par un -42 :
 - *alloc
 - *printf
 - free
 - open, fopen, etc ...
 - o De facon générale, les fichers associés à une classe seront toujours NOM_DE_LA_CLASSE.h





- et NOM DE LA CLASSE.cpp (s'il y a lieu).
- o Les repértoires de rendus sont ex00, ex01, ..., exN
- o Toute utilisation de friend se soldera par un -42, no questions asked.
- Lisez attentivement les exemples, ils peuvent requérir des choses que le sujet ne dit pas...
- o Ces exercices vous demandent de rendre beaucoup de classes, mais la plupart sont TRÈS courtes si vous faites ca intelligemment. Donc, halte à la flemme!
- o Lisez ENTIÈREMENT le sujet d'un exercice avant de le commencer!
- REFLÉCHISSEZ. Par pitié.
- REFLÉCHISSEZ. Par Odin!
- o R.E.F.L.É.C.H.I.S.S.E.Z. Nom d'une pipe.

• COMPILATION DES EXERCICES :

- La moulinette compile votre code avec les flags : -W -Wall -Werror
- o Pour éviter les problèmes de compilation de la moulinette, incluez les fichiers nécéssaires dans vos fichiers include (*.h).
- 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.
- Ce sujet peut être modifié jusqu'à 4h avant le rendu. Rafraichissez-le régulièrement!
- Le repértoire de rendu est : (DÉPOT SVN piscine_cpp_d14m-promo-login_x)/exN (N étant bien sur le numéro de l'exercice).





Chapitre II

Exercice 0

KOALA	Exercice: 00 points: 3	
Exercice 0		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d14m-promo-login_x)/ex00		
Compil	ateur : g++	Flags de compilation: -W -Wall -Werror
Makefile: Non		Règles : n/a
Fichiers a rendre: Fruit.h Fruit.cpp Citron.h Citron.cpp Banana.h,		
Banana.cpp, Cagette.h, Cagette.cpp, FruitNode.h		
Remarques: n/a		
Fonctions Interdites : Tout. excepté new et delete		

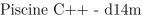
Les fruits c'est bon, mangez-en. C'est plein de bonnes petites vitamines qui font tout plein de bien à nos petits corps fatigués par cette dure piscine.

Mais avant de pouvoir déguster un bon jus pressé vitaminé, il va falloir travailler un peu.

Commencez par implémenter les classes suivantes : - Fruit - Citron - Banana Arrangez-vous pour avoir un arbre d'héritage cohérent et que ce code compile :

```
1 $> cat main.cpp
2 int main(void)
з {
           Citron c;
4
          Banana b;
5
           std::cout << c.getVitamins() << std::endl;</pre>
6
           std::cout << b.getVitamins() << std::endl;</pre>
           std::cout << c.getName() << std::endl;</pre>
8
           std::cout << b.getName() << std::endl;</pre>
9
           Fruit& f = c;
10
11
           std::cout << f.getVitamins() << std::endl;</pre>
           std::cout << f.getName() << std::endl;</pre>
          return 1337;
13
15 $>./a.out | cat -e
16 3$
```





The Cast

```
17 5$
18 citron$
19 banana$
20 3$
21 citron$
```

Toutes les spécialisations de la classe Fruit devront initialiser lors de leur construction, un attribut de type int déclaré dans Fruit contenant le nombres de vitamines.



Cet attribut devra s'appeller _vitamins

getName() renverra une instance de std::string contenant le nom du fruit. Chaque fruit aura sa propre version de getName().

On ne doit pas pouvoir réifier la classe Fruit.

Maintenant nous avons besoin de construire une Cagette, car il nous faut beaucoup, beaucoup de vitamines ce qui signifie beaucoup, beaucoup de fruits. Et deux mains, c'est pas très pratique pour transporter tout plein de fruits.

Notre Cagette sera un conteneur de Fruit, implémenté sous forme de liste chainée. Je veux une Cagette qui possède les fonctions membres suivantes (débrouillez-vous pour les const):

- Cagette(int size); // construit une cagette pouvant contenir size fruits.
- int nbFruits(); // renvoie le nombre de Fruit actuellement dans la Cagette
- bool putFruit(Fruit* f); // ajoute un Fruit en fin de Cagette.
- Fruit* pickFruit(); // retire un Fruit de la Cagette (le premier qui viens).
- FruitNode* head(); // renvoie la tête de liste chainée.
- putFruit(Fruit *) renvoie false si la Cagette est pleine ou si l'instance du Fruit est déjà présente dans la Cagette .
- pickFruit() renvoie 0 (pointeur null) si la Cagette est vide.
- head() renvoie 0 (pointeur null) si la Cagette est vide.

Afin de manipuler les Fruits sous forme de liste chainee, il va falloir les encapsuler dans une structure FruitNode. Débrouillez-vous pour l'implémentation, je veux au moins un





attribut next.

Je ne veux pas savoir comment la Cagette marche, je veux juste transporter des Fruit , pleins de fruits, pour toujours plus de vitamines. Attention, une Cagette ne peut pas être copiée.





Chapitre III

Exercice 1

KOALA	Exercice: 01 points:	
Fruits		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d14m-promo-login_x)/ex01		
Compilateur : g++		Flags de compilation: -Wextra -Werror -Wall
Makefile: Non		Règles : n/a
Fichiers a rendre: Fruit.h, Fruit.cpp, Citron.h, Citron.cpp, Banana.h,		
Banana.cpp, Cagette.h, Cagette.cpp, FruitNode.h, LittleHand.h,		
LittleHand.cpp, CitronVert.h, CitronVert.cpp		
Remarques: n/a		
Fonctions Interdites: tout		

Bonne nouvelle, on a tout un stock de Cagette , genre vraiment beaucoup, de quoi faire une putain de fruit'party!

Le problème c'est que rien n'est trié :'(, il va donc falloir placer tous les Fruit de même type ensemble.

Attention nous avons un nouveau type de Fruit , le CitronVert qui hérite du Citron bien sûr. Il est pauvre en vitamines (seulement 2). Et son petit nom, c'est "citron vert".

Nous allons y aller à la main pour trier toutes les cagettes. La classe LittleHand implémentera tout ce pénible travail. Elle possédera la fonction membre statique suivante:

Cette fonction va déplacer dans les Cagette correspondantes tous les Fruit de la Cagette en vrac. Tous les fruits retirés de la Cagette en vrac qui ne peuvent pas être placé dans une quelconque Cagette (pas le bon type de Fruit , ou Cagette pleine) seront simplement replacés dans la Cagette en vrac.





Chapitre IV

Exercice 2

KOALA	Exercice: 02 points: 4		
Eat my Coconut			
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d14m-promo-login_x)/ex02			
Compilateur : g++		Flags de compilation: -Wextra -Werror -Wall	
Makefile : Non		Règles : n/a	
Fichiers a rendre: Fruit.h, Fruit.cpp, Citron.h, Citron.cpp, Banana.h,			
Banana	Banana.cpp, Cagette.h, Cagette.cpp, FruitNode.h, LittleHand.h,		
LittleHand.cpp, CitronVert.h, CitronVert.cpp, Coconut.h, Coconut.cpp			
Remarques: n/a			
Fonctions Interdites: tout sauf new et delete			

Pendant votre Fruit'Party j'en ai profité pour vous négocier quelques fruits en vrac chez un grossiste vraiment pas cher.

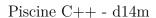
Il propose d'ailleurs un nouveau Fruit , le Coconut (qui est une sorte de Fruit). Avec un bon 15 de vitamines à cause de son lait, et son joli nom "coconut" ce fruit sympa devrait détoner dans nos jus de fruits. Je lui ai demandé de nous envoyer toute une quantité de Coconut . Par contre il ne livre qu'en paquet, il va donc falloir réceptionner les Coconut et les ranger dans des Cagettes .

C'est encore une fois vos petites mains qui vont se mettre au travail. LittleHand est maintenant étendue d'une nouvelle fonction membre de classe prenant en paramêtre un paquet de Coconut et renvoyant un tableau de Cagette pouvant contenir 6 Coconut chacune (c'est que c'est gros ces trucs là).

Voici le prototype de la fonction :

- Cagette * const * organizeCoconut(Coconut const * const * coconuts paquet);
 - o coconuts_paquet est un tableau de Coconut terminé par un pointeur null.
 - o organizeCoconut renvoie un tableau (alloué dynamiquement) de Cagette (al-





The Cast



louées dynamiquement), terminé par un pointeur null.

Histoire de me répéter : si les petites mains recoivent un tableau de 25 (pointeurs sur) Coconut , elles vont renvoyer un tableau de 5 (pointeurs sur) Cagette , 4 pleines et une ne contenant qu'une seule Coconut .

Dépêchez-vous de ranger tout ca, les carences en vitamine, ça arrive plus vite que l'on ne pense!

Ne changez le prototype de organizeCoconut sous aucun prétexte, nous ne pourrions plus être livrés!





Chapitre V

Exercice 3

KOALA	Exercice: 03 points: 5		
Pour plus de vitamines, mixons les fruits!			
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d14m-promo-login_x)/ex03			
Compilateur : g++		Flags de compilation: -Wextra -Werror -Wall	
Makefile : Non		Règles : n/a	
Fichiers a rendre: Fruit.h, Fruit.cpp, Citron.h, Citron.cpp, Banana.h,			
Banana.cpp, Cagette.h, Cagette.cpp, FruitNode.h, LittleHand.h,			
Little	LittleHand.cpp, CitronVert.h, CitronVert.cpp, Coconut.h, Coconut.cpp,		
Mixer.h, Mixer.cpp			
Remarques: n/a			
Fonctions Interdites: tout			

Nous avons enfin de quoi gérer nos fruits bourrés de vitamines. Attaquons le mixage afin d'obtenir un jus de fruit de qualité, très frais et surtout énergisant.

Toutefois il va falloir mixer notre jus. Et pour ca il nous faut un... mixer.

Nous n'avons rien de mieux sous la main qu'un vieux mixer fait de bric et de broc. Son interface n'est pas des plus aisée et il va falloir la bidouiller un petit peu pour le brancher électriquement.

Nous disposons donc d'un MixerBase , base de Mixer complètement broken, qui ne possède pas de cable de branchement et encore moins de lame de mixage... Il va falloir faire avec.



Cette classe est déclarée dans MixerBase.h, vous ne devriez donc pas la modifier. (La moulinette utilisera sa propre version de toute manière, nous utiliserons aussi notre propre définition des fonctions membres de cette classe.)

Voici à quoi ressemble la partie visible de notre pauvre MixerBase :





Nous savons que par défaut le mixer n'est pas branché et qu'il ne possède aucun moyen de mixer.

Il va falloir donc spécialiser tout ca afin, d'une part, d'initialiser le pointeur de fonction avec une fonction qui s'occupera de mixer la Cagette de fruit passée en paramètre, et d'autre part d'offrir le moyen de brancher électriquement le Mixer .

Le Mixer doit avoir une fonction membre du nom de votre choix qui se charge de brancher électriquement le mixer. La fonction de mixage elle même devra renvoyer la somme de toutes les vitamines des Fruit passés en paramêtre pour un jus d'une fraicheur et d'une qualité énergisante inégalée.

Même si le Mixer offre le moyen d'être branché électriquement, il va falloir le brancher avec vos petite minimes. Dans LittleHand , ajoutez une fonction membre statique pour le MixerBase passé en paramêtre :

```
void plugMixer(MixerBase& mixer);
```

Etant donné que vous n'avez bricolé qu'un seul type de MixerBase , vous pouvez être sur que plugMixer prendra comme type réel Mixer en paramêtre.

Je me régale d'avance de tous ces merveilleux et délicieux jus que nous allons pouvoir déguster! Je me sens déjà plein d'énergie!





Chapitre VI

Exercice 4

KOALA	Exercice: 04 points	
Petit coup de main		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d14m-promo-login_x)/ex04		
Compilateur : g++		Flags de compilation: -Wextra -Werror -Wall
Makefile : Non		Règles : n/a
Fichiers a rendre: Banana.cpp Banana.h Cagette.cpp Cagette.h Citron.cpp		
Citron.h CitronVert.cpp CitronVert.h Coconut.cpp Coconut.h Fruit.cpp		
Fruit.h FruitNode.h LittleHand.cpp LittleHand.h Mixer.cpp Mixer.h		
Remarques: n/a		
Fonctions Interdites : Aucune		

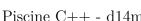
Ca vous fait pas un petit effet bizarre d'avaler autant de vitamines? Moi je sais pas, ca me donne envie de changer le monde, de créer quelque chose de nouveau... Mais à l'instar des modifications génétiques par plasmide dans l'excellent jeu Bioshock, je veux faire quelque chose de vrai, je promets de ne jamais les utiliser à mauvais escient et tel la devise de Google (don't be evil), je rendrai le monde meilleur, plus sain, les gens seront heureux. Et le petit lapin blanc vous a retrouvé. Je veux modifier les fruits, oui, augmenter leur potentiel vitaminique! Je veux plus de vitamines, toujours plus (afin de rendre le monde meilleur). Nous aurons tellement de bonnes et pures énergies, que nos chakras s'ouvriront, et alors, telle ta maman qui te borde sous la couette, le monde sera doucement recouvert d'une douce et paisible couche de crème chantilly (pour le rendre meilleur. D'ailleurs avec les Fraises, ca passe super bien).

Nous allons donc trouver un moyen de modifier directement les fruits, nous allons creuser au coeur de la matière, et nous conquerrons le monde! (pour le rendre meilleur).

C'est encore une fois vos petites mains qui se chargerons de cette délicate opération. Vous ajouterez donc à LittleHand la fonction membre statique suivante :

```
void injectVitamin(Fruit& fruit, int quantity);
```





Piscine C++ - d14mThe Cast

Cette fonction se chargera d'injecter (remplacer) dans un fruit la quantity de vitamines passée en paramètre. Essayer d'utiliser la structure à la Matrix suivante pour vous prendre pour Néo et modifier la matière :

```
struct InTheMatrixFruit
2 {
         virtual ~InTheMatrixFruit();
3
         int vitamins;
4
5 };
```

Maintenant, nous sommes presque égaux avec les grand dieux du C++, nous avons le pouvoir de modifier la matière! Le monde et l'avenir nous appartiennent!!!

