



Alphorm.com

Présentation de la formation

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Présentation du formateur
- Qu'est-ce que C++ ?
- Le plan de la formation
- Les références bibliographiques
- Autres liens utiles

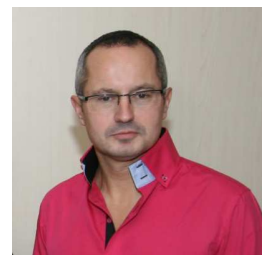


Programmer en C++

alphorm.com™©

C++ Présentation du formateur

- Fabien Brissonneau
- Email : fabien.brissonneau@gmail.com
- Consultant Concepteur et Formateur
- Missions d'architecture, de conception , de réalisation logicielles
- Fondateur de **eiXa6**
- Actuellement en mission sur un projet de gestion
- Mes références :
 - Mon profil Viadeo : <http://fr.viadeo.com/fr/profile/fabien.brissonneau>
 - Mon profil LinkedIn : <http://fr.linkedin.com/pub/fabien-brissonneau/65/902/92a/>



C++ Qu'est-ce que C++ ?

- Un langage de programmation informatique
- Orienté objets
- Dérivé du C et compatible avec le langage C
- Standardisé ISO depuis 1998
- Dans sa version C++11, de 2011
- Une version nouvelle prévue en 2014

C++ Le plan de la formation

- Chapitre 1 : Du procédural à l'objet
- Chapitre 2 : L'organisation du projet C++
- Chapitre 3 : La syntaxe de base
- Chapitre 4 : Le C++, langage orienté objets
- Chapitre 5 : Les templates
- Chapitre 6 : Les exceptions
- Chapitre 7 : Les fonctions et classes amies
- Chapitre 8 : Les opérateurs
- Chapitre 9 : La librairie standard

C++ Les références bibliographiques

- The C++ Programming Language, Stroustrup, Addison-Wesley
- Programmer en langage C++, Delannoy, Eyrolles
- Apprendre le C++, Delannoy, Eyrolles

C++ Autres liens utiles

- www.cppreference.com
- www.cplusplus.com
- www.learncpp.com
- www.stroustrup.com
- www.open-std.org

C++ Il n'y a plus qu'à...

GO



Alphorm.com

Du procédural à l'objet

Historique des langages

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les ancêtres du C++, C et Simula
- Les premiers temps du C++
- Les langages proche Java et C#
- Les dernières évolutions du C++



Programmer en C++

alphorm.com™©

C++ Les ancêtres du C++

- Simula depuis 1962
 - Langage à classes
 - Ancêtre de C++ et aussi de Smalltalk
-
- Le langage C, depuis 1972
 - Orienté système

C++ Les premiers temps du C++

- Le « C avec des classes », en 1979
- Permettre d'éviter l'usage de l'assembleur ou du C
- Tout en étant performant, et simple d'utilisation
- Version standard ISO en 1998
- Templates, namespaces, et RTTI
- Révision en 2003

C++ Les langages Java et C#

- Java créé par Sun en 1995
- Langage portable, grande bibliothèque de classes
- C# créé par Microsoft en 2001
- Adapté à l'environnement .Net

C++ Les évolutions de C++

- La version standardisée C++11
- Plus simple à apprendre
- Librairie standard suivant TR1

C++ Ce qu'on a couvert

- Le C++ est un langage héritier des C et Simula
- Langage orienté objet, mais compatible avec le C
- Une première version standard en 1998, révisée en 2003
- La dernière version date de 2011



Programmer en C++

alphorm.com™©



Alphorm.com

Du procédural à l'objet

Les critères de qualité

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les critères de qualité du code logiciel
- Fiabilité
- Efficacité
- Maintenabilité
- Portabilité



C++ Les critères de qualité

- Qualité fonctionnelle
- Ergonomie
- Fiabilité
- Efficacité
- Maintenabilité
- Portabilité

C++ La fiabilité

- Le développeur doit être sûr de ce qu'il écrit
- Le code doit être lisible
- Le code doit rester simple
- Pas d'algorithmes compliqués
- Il faut maîtriser le cycle de vie des données

C++ L'efficacité

- Un langage proche de la machine
- Un algorithme facile à améliorer, lisible
- Disposer d'algorithmes tout faits, de façon à éviter de réinventer la roue

C++ La maintenabilité

- Le code doit être lisible et sans surprise
- L'intention du design doit transparaître dans le code
- Pas d'effets de bord malheureux
- Utilisant un maximum de standards, de façon à pouvoir reprendre le code des autres

C++ La portabilité

- Le langage doit être utilisable sur plusieurs plateformes
- Le C++ est exploitable très largement
- Les standards sont bien définis

C++ Ce qu'on a couvert

- Parmi les critères de qualité du logiciel
- Un bon nombre sont des problèmes liés au code
- Le C++ améliore la fiabilité, la maintenabilité, l'efficacité et la portabilité



Programmer en C++

alphorm.com™©



Alphorm.com

Du procédural à l'objet
L'orienté objets

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les concepts statiques
 - Classes, objets, méthodes, attributs, héritages
- Les concepts dynamiques
 - Messages, séquences, polymorphisme



C++ Les classes

- Avec un langage orienté objets, il faut créer ses propres types
- Une classe définit un type
- La définition d'une classe contient la définition des fonctions et des données
- Une classe déclare des visibilité qui empêche le code extérieur de venir interférer avec ses états

C++ Les objets

- L'allocation mémoire passe par un « new »
- La zone obtenue est un objet
- Contient les données membres
- Sur lequel s'applique les fonctions membres

C++ Les méthodes

- Une méthode est une fonction dans une classe
- En C++, on parle de fonction membre
- Il est possible de prendre des paramètres, d'avoir des variables locales
- Une fonction membre accède aux données membres de l'objet

C++ Les attributs

- Les attributs sont des données attachées aux objets
- En C++, on parle de données membres
- En général, ces données sont propres à un objet (à une allocation mémoire)

C++ La généralisation

- Une relation entre deux classes
- Permet de réutiliser le code de l'ancêtre sans le modifier
- Appelée aussi héritage

C++ Les messages

- Les objets communiquent entre eux par message
- Correspond en C++ à un appel de fonction membre
- Peuvent être accompagnés de passage d'arguments
- En général, le message et la fonction membre coïncident

C++ Les séquences

- Dans une fonction membre, le code est séquentiel
- Donc le C++ peut être utilisé comme un langage impératif
- Les séquences de code devraient être courtes

C++ Le polymorphisme

- Lorsqu'une classe B hérite d'une classe A
- Que la classe B redéfinit (au lieu d'hériter simplement) une fonction membre de A
- Pour un appelant de cette fonction, il est possible de ne pas connaître la classe B alors qu'il appelle la fonction de B

C++ Ce qu'on a couvert

- Les concepts liés à l'orienté objet ne sont pas nombreux
- Leur utilisation pertinente améliore la qualité du code
- Le C++ supporte les concepts de classes, objets, héritage, polymorphisme
- Le C++ peut néanmoins être utilisé comme un langage impératif





Alphorm.com

L'organisation du projet en C++

Le compilateur en ligne

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le compilateur gnu
- Ajouter un fichier source et créer un programme
- Ajouter plusieurs fichiers, choisir une sortie
- Compiler seulement, puis lier



Programmer en C++

alphorm.com™©

C++ Le compilateur en ligne de commande

- Installer et positionner le PATH
- Lancer g++ <fichier>
- Création d'un exécutable a.exe
- Exemple d'option : -std=c++11

C++ Ajouter des fichiers sources

- Editer un code pour un programme « hello world » dans main.cpp
- Lancer g++ main.cpp
- Sortie dans a.exe, exécuter et observer

C++ Ajouter plusieurs fichiers

- Créer un second fichier source contenant la fonction display()
- Appeler la fonction display depuis le main
- Noter l'usage de « extern »
- Choisir le nom de la sortie

C++ Compiler seulement et lier

- Compiler seulement chacun des fichiers : -c
- Lier les .o obtenus et créer un exe

C++ Ce qu'on a couvert

- Nous avons vu les possibilités d'un compilateur en ligne de commande
- Nous invoquons la compilation
- Puis l'édition de lien



Programmer en C++

alphorm.com™©



Alphorm.com

L'organisation du projet en C++

Le préprocesseur

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le rôle du préprocesseur
- Intérêt de #include
- Usage de #define
- Intérêt de #define



C++ Le rôle du préprocesseur

- Préparer les fichiers sources
- Construire les unités de compilation
- Inclure des lignes de code de façon conditionnelle
- Définir des macros, plus ou moins complexes

C++ Intérêt de #include

- Avec la directive #include [fichier], il y a inclusion du fichier dans l'unité de compilation
- Le chemin de recherche est différent selon que l'on utilise <fichier> ou bien "fichier"
- Pas de « ; » en fin de ligne

C++ Usage de #define

- On peut définir un symbole remplacé :
 - #define MACRO SUITE_DE_CARACTERES
- Ou bien seulement définir un symbole :
 - #define MON_SYMBOLE
- Il est possible de tester si un symbole est défini :
 - #ifdef MON_SYMBOLE ou bien #ifndef MON_SYMBOLE
 - Ne pas oublier le #endif

C++ Eviter d'inclure plusieurs fois

- Pour protéger le contenu d'un entête, que l'on inclurait plusieurs fois :
 - `#ifndef __SYMBOLE__`
 - `#define __SYMBOLE__`
 - `//blah blah`
 - `//déclaration du vrai symbole, fonction, classe, etc`
 - `#endif`

C++ Ce qu'on a couvert

- Le préprocesseur est capable de préparer les unités de compilation
- Les fichiers sont inclus les uns dans les autres
- Il est possible de gérer des inclusions conditionnelles





Alphorm.com

L'organisation du projet en C++ Sources et entêtes

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le fichier source, base de l'unité de compilation
- Le contenu du fichier source
- Le contenu des entêtes
- Règles concernant les entêtes



Programmer en C++

alphorm.com™©

Ⓒ++ Le fichier source, base de l'unité de compil

- Une unité de compilation est :
 - Le fichier source
 - L'ensemble des fichiers inclus
- Pour compiler, tous les symboles doivent être déclarés
- L'unité donne naissance à un fichier « objet », extension .o ou .obj
- L'édition de lien sera réalisée entre les unités de compilation pour produire l'exécutable

Ⓒ++ Le contenu du fichier source

- Un fichier source contient les corps des fonctions
- Ou bien le corps des fonctions membres
- Un seul dans l'application contient le point d'entrée, « main »
- Ce type de fichier n'est pas inclus ailleurs
- Il inclut tous les fichiers déclarant les symboles qu'ils utilisent

C++ Le contenu des entêtes

- Un fichier entête est inclus dans d'autres fichiers
- Il peut inclure d'autres entêtes, mais attention
- Il contient la déclaration des fonctions, classes, etc
- Sauf cas des fonctions(-membres) « inline », ne contient pas de corps

C++ Quelques conseils ...

- Eviter d'inclure les entêtes plusieurs fois n'est pas possible, donc pratiquer `#ifndef/#define/#endif` ou bien `#pragma once` (MS)
- Un entête ne devrait pas inclure de fichiers, sauf cas particuliers, car les « nœuds » d'inclure sont possibles
- Un entête ne devrait pas faire de `using namespace ...` car il propage l'info à tous les fichiers incluant
- Plutôt qu'inclure une déclaration, un entête peut souvent faire une déclaration anticipée

C++ Ce qu'on a couvert

- Les fichiers du projet sont soit des fichiers sources, contenant les corps des fonctions ou fonctions membres, soit des entêtes, contenant des déclarations
- Un fichier source est à l'origine d'une unité de compilation
- Un fichier d'entête évite d'inclure d'autres fichiers d'entête du même projet



Programmer en C++

alphorm.com™©



Alphorm.com

L'organisation du projet en C++

Les makefiles

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- L'outil make
- Le fichier makefile



C++ L'outil make

- L'utilitaire « make » en ligne de commande
- Permet de lancer une cible décrite dans le makefile
- Compilation / édition de lien / ...
- Usage :
 - Seulement make, par défaut
 - Ou bien make -f <fichier>

C++ Le makefile

- Format classique **Cible : dépendance commande**
- Choix de la cible lors du lancement de make

```
main : main.o display.o  
g++ -o main display.o main.o
```

```
main.o : main.cpp  
g++ -o main.o -c main.cpp -Wall
```

```
display.o : display.cpp  
g++ -o display.o -c display.cpp -Wall
```

C++ Ce qu'on a couvert

- L'outil make permet d'enchaîner les compilation/édition de lien
- Cet outil utilise un fichier Makefile





Alphorm.com

L'organisation du projet en C++ IDE CodeBlocks

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Création d'un projet
- Ajout de fichiers au projet
- Compilation et création de l'exécutable
- Exécution et debug



Programmer en C++

alphorm.com™©

C++ Création du projet

- Choisir un projet vide
- Choisir les répertoires
- Paramètres du projet, exemple de C++11

C++ Ajouter des fichiers

- Ajout d'un nouveau fichier
- Choisir un fichier d'extension .cpp
- Il est possible de rajouter un fichier d'entête

Ⓒ++ Compiler et construire le programme

- Compiler un seul fichier : sélection et « build »
- Editer l'exécutable : sélection du projet et « build »

Ⓒ++ Exécuter en release ou debug

- Exécution en mode release
- Exécution en mode debug
- Affichage des fenêtres de debug, avec callstack, espion, ...

C++ Ce qu'on a couvert

- Un IDE est très utile pour maîtriser le développement
- Non seulement nous éditons le code, mais nous commandons les compilations et link
- Il est possible d'exécuter ou de déboguer à partir de l'IDE



Programmer en C++

alphorm.com™©



Alphorm.com

L'organisation du projet en C++ IDE MS Visual Studio

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Création d'un projet
- Ajout de fichiers au projet
- Compilation et création de l'exécutable
- Exécution et debug



C++ Création du projet

- Choisir un projet vide
- Choisir les répertoires

C++ Ajouter des fichiers

- Ajout d'un nouveau fichier
- Choisir un fichier d'extension .cpp
- Il est possible de rajouter un fichier d'entête

C++ Compiler et construire le programme

- Sélection du fichier cpp, puis « compile »
- Sur le projet, « build », « clean », « rebuild »

C++ Exécuter en release ou debug

- Debug -> Start debugging / Start without debugging
- Point d'arrêt, pas à pas

C++ Ce qu'on a couvert

- Un IDE est très utile pour maîtriser le développement
- Non seulement nous éditons le code, mais nous commandons les compilations et link
- Il est possible d'exécuter ou de déboguer à partir de l'IDE





Alphorm.com

La syntaxe de base Le main

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le démarrage d'un nouveau projet sous Visual Studio
- Ajouter des fichiers sources
- La fonction principale du programme
- L'exécution du programme
- Les codes de retour



Programmer en C++

alphorm.com™©

C++ Démarrer un nouveau projet C++

- Visual Studio, les types de projets C++
- L'organisation du projet sous Visual Studio

C++ Ajouter des fichiers

- Fichiers sources et fichiers d'en-tête
- Rôle des fichiers sources

C++ La fonction principale d'un programme C++

- Créer la fonction main, qui sera exécutée au démarrage du programme
- Différentes syntaxes possibles
 - -> void main()
 - -> int main(int , char**)
 - N'importe quels paramètres, même sans signification

C++ L'exécution du programme avec Visual

- Compilation
- Edition de lien
- Exécution

C++ Les codes de retour du main

- Définis dans cstdlib
- EXIT_SUCCESS
 - Équivalent valeur 0
 - Équivalent à ne rien retourner
- EXIT_FAILURE
- Interprétation dépendante de l'implémentation

C++ Ce qu'on a couvert

- La présentation du démarrage d'un projet Visual Studio
- La fonction main, sa syntaxe et son rôle





Alphorm.com

La syntaxe de base

Les entrées-sorties

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les sorties traces sur la console avec cout/cerr
- Les bibliothèques à utiliser
- Les modificateurs
- Les saisies clavier
- Généralités sur les flux



Programmer en C++

alphorm.com™©

C++ Les sorties traces sur la console

- Sortie standard : cout
- Sortie d'erreur standard : cerr
- Entrée standard : cin

C++ Les bibliothèques à utiliser

- Bibliothèques :
 - - pour cout, cin, cerr ... : iostream
 - - pour ofstream, ifstream... : fstream
 - - pour ostream, istream : sstream
 - - pour les manipulateurs : ios, iomanip

C++ Les modificateurs

- Modifient le comportement du flux :
 - Base :dec/hex/oct
 - Booléens : boolalpha, noboolalpha
 - Format flottants : fixed/scientific/hexfloat/defaultfloat
 - ...
 - Affichage des décimales : setprecision
 - Largeur de l'affichage : setw

C++ Les saisies clavier

- Utiliser cin comme entrée standard
- L'opérateur extraction >>
- Surchargeable
- Type istream

C++ Généralités sur les flux

- Les flux :
 - Entrées-sorties : cout/cerr/cin
 - Fichiers : ofstream, ifstream
 - String : ostream/istream
- Héritiers de ios
- Mappés sur streambuf / wstreambuf
- Existent en version char et wchar_t

C++ Ce qu'on a couvert

- Réaliser des entrées-sorties avec la console
- Utiliser des modificateurs
- La notion de flux





Alphorm.com

La syntaxe de base

Les types de base

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le type booléen
- Les types caractères
- Les types entier
- Les modèles mémoire
- Les types flottants
- Premières remarques sur std::string



Programmer en C++

alphorm.com™©

C++ Le type booléen

- Le type bool
- Prend les valeurs true ou false
- La valeur 0 est souvent assimilée à false

C++ Les types caractères

- Le type char
- Le type wchar_t
 - Dépendant du compilateur
 - Voir char16_t et char32_t

C++ Les types entier

- Les types int :
 - Le short int
 - Le int
 - Le long int
 - Le long long int
- Existent en version signed ou unsigned
- Le typedef size_t est typiquement un unsigned int
- Utilisé comme valeur de retour de sizeof

C++ Les modèles mémoire

- Win32 et Unix32
 - 4 octets pour int , long et pointeur
- Win64
 - 4 octets pour int, long et 8 pour pointeur
- Unix64
 - 4 octets pour int, 8 pour long et pointeur
- Pour connaître les valeurs limites, utiliser numeric_limits<T>

C++ Les types flottants

- Le type float
 - IEEE 754 32 bits
- Le type double
 - IEEE 754 64 bits
- Le type long double

C++ Remarque sur std::string

- Conteneur confortable pour manipuler une chaîne de caractères
- Faire `#include<string>`
- Classe `std::string`

C++ Ce qu'on a couvert

- La présentation des types fondamentaux :
 - Le char
 - Les int
 - Les flottants
 - Utiliser sizeof et size_t
 - Utiliser numeric_limits



Programmer en C++

alphorm.com™©



Alphorm.com

La syntaxe de base

La classe string

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La classe `std::string`
- Les fonctions membres de `string`
- Les fonctions non-membres



C++ La classe `std::string`

- Inclusion de `<string>`
- Dans l'espace de nommage `std`
- Utiliser `std::string` ou bien `using namespace std;`

C++ Les fonctions membres

- Les constructeurs avec ou sans paramètres
- Les opérateurs d'affectation
- Les fonctions assign

C++ Les fonctions membres

- L'opérateur d'affectation
- La fonction at, avec contrôle des bornes
- La fonction data()
- La fonction c_str()

C++ Les itérateurs

- Les fonctions begin() et rbegin()
- Les fonctions end() et rend()

C++ Les fonctions membres

- La fonction empty()
- La fonction capacity()
- La fonction reserve()
- Les fonctions size() et length()

C++ Les fonctions membres

- La fonction append()
- La fonction clear()
- La fonction insert()
- La fonction erase()
- La fonction operator+()
- La fonction compare()
- La fonction replace()

C++ Les fonctions membres

- Les fonctions find(), rfind()
- Les fonctions find_first_of()

C++ Les fonctions non-membres

- Opérateurs de comparaison
- Opérateur +
- Possibilité d'injecter ou d'extraire une string dans un flux << et >>

C++ Les fonctions non-membres

- Depuis C++11, fonctions de conversions vers ou à partir de numériques
- Les fonctions stoi, stol, ...
- Les fonctions stof, stod, ...

C++ Ce qu'on a couvert

- Nous avons présenté la classe `std::string`, une représentation avantageuse de la chaîne de caractères.
- Elle possède des fonctions membres
- Elle peut être considérée comme un conteneur, donc manipulable par itérateurs
- Des fonctions non-membres peuvent y être appliqués



Programmer en C++

alphorm.com™©



Alphorm.com

La syntaxe de base

Les struct et class

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les struct et les class
- Les déclarations anticipées
- La déclaration des membres
- Les visibilités
- Séparer la déclaration de la définition
- Les fonctions membres générées automatiquement



C++ Les struct et class

- Les structures ou les classes ne se distinguent que par la visibilité par défaut de leurs membres
- Avec struct ou class, nous définissons un type d'objet, dont le déclaration va contenir à la fois les données et les fonctions associées
- Déclaration : `class nom_de_la_classe attributs_éventuels base_éventuelle déclaration_des_membres`
- Les attributs sont en général dépendants de l'implémentation

C++ Les déclarations anticipées

- Une déclaration anticipée déclare un type au compilateur
- Mais le type reste incomplet
- Utile pour éviter d'inclure un entête, si on n'a besoin que de pointeurs ou références

C++ Les déclarations des membres

- Déclarer les données membres
- Déclarer les fonctions membres
- Déclarer des types internes

C++ Les visibilité

- Il peut y avoir des spécifications de visibilité par zones : public, privée, protégée
- Par défaut la visibilité est privée

C++ Séparer déclaration et définition

- Dans un fichier d'entête, il faudra mettre la déclaration
- Dans le fichier de corps, il faudra placer les définitions des fonctions membres

C++ Les fonctions membres générées

- Le constructeur par défaut
- Le destructeur
- Le constructeur par copie
- L'opérateur d'affectation
- Le constructeur de move
- L'opérateur d'affectation move

C++ Ce qu'on a couvert

- Les struct et class sont des types utilisateurs
- Contiennent les données et traitements associés
- Certaines fonctions membres sont générées automatiquement





Alphorm.com

La syntaxe de base

Le contrôle du flux

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les boucles
 - Avec while
 - Avec do-while
 - Avec for
 - Avec le nouveau for
- Les conditions
 - Avec if-else
 - Avec switch-case
 - L'opérateur ternaire
- Les labels et goto



Programmer en C++

alphorm.com™©

C++ Les boucles

- Avec while (condition) ...
- Avec do ... while(condition)
- Avec for(init ; tant que ; incrémentation) ...
- Avec for(variable : ensemble) ...

C++ Les conditions

- Avec if (condition) ...
- Avec ... else ...
- Avec switch (variable) { case valeur : ...}
- Le default

C++ L'opérateur ternaire

- Remplace un if (condition) traitement1 else traitement2
- S'exprime comme : (condition) ? Traitement1 : traitement2
- Très utile lorsque cette expression est un retour

C++ Les labels et goto

- Il faut poser des labels dans le code label:
- Puis avec goto label, on est branché à l'endroit
- Pratique, mais rend le code difficile à suivre, ne pas abuser

C++ Ce qu'on a couvert

- Les boucles avec while, do-while, for et la variante for « range »
- Les conditions avec if, else, et switch-case
- L'opérateur ternaire
- Les gotos et les labels (étiquettes)



Programmer en C++

alphorm.com™©



Alphorm.com

La syntaxe de base

Les cycles de vie

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les objets automatiques
- Les objets dynamiques
- Les objets statiques



C++ Les objets automatiques

- La mémoire dite automatique est gérée sur la pile
- Rapide
- Destruction garantie des objets

C++ La manifestation du destructeur

- Un objet automatique est détruit, et voit son destructeur appelé
- Faire un destructeur et tracer son exécution

C++ Les objets dynamiques

- Un objet dynamique est créé avec un new
- Il faut le détruire avec un delete

C++ Détruire un objet ou un tableau

- Il existe deux formes principales de delete
 - Pour un objet, delete
 - Pour un tableau delete[]

C++ Les objets statiques

- Les objets statiques ont la durée de vie du programme
- Ils peuvent exister avant le main

C++ Ce qu'on a couvert

- Les objets automatiques
- Les objets dynamiques
- Les objets statiques



Programmer en C++

alphorm.com™©



Alphorm.com

La syntaxe de base

Les transtypages

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le contrôle de types
- La promotion entière
- Le cast « à l'ancienne »
- L'opérateur static_cast
- L'opérateur dynamic_cast
- L'opérateur const_cast
- L'opérateur reinterpret_cast



C++ Le contrôle de type

- Une variable est déclarée et typée
- Le compilateur interdit la perte de valeur, et de convertir une donnée dans un type non compatible
- Dans conversions standards existent

C++ La promotion entière

- D'une façon générale, lorsqu'on applique un opérateur sur un entier, il est promu en int, afin d'éviter les erreurs de débordements
- Par exemple, l'opérateur + sur deux short fournira un résultat bon, alors que += peut donner un mauvais résultat

C++ Le cast « à l'ancienne »

- Mettre le type entre parenthèses :
 - Faire `int v = 56;`
 - Puis `short s = (short)v;`
 - Ou bien `short s = short(v);`
- Aucun contrôle, écriture peu lisible

C++ Le static_cast

- L'opérateur static_cast valide le transtypage entre types reliés
- Utilisation : Chat* p = static_cast<Chat*>(f), si f est Felin*
- Détectera l'erreur à la compilation
- Eclaire l'intention du développeur
- Peut s'appliquer sur une référence aussi

C++ Le dynamic_cast

- Si Felin* p ...
- Chat* c = dynamic_cast<Chat*>(p) retourne NULL si la conversion n'est pas correcte runtime
- La conversion est possible si les types sont reliés par héritage

C++ Le const_cast

- Permet d'enlever le caractère const à un pointeur
- Le résultat d'une modification via un pointeur qui était auparavant const n'est pas garanti
- Si `const Type* ct ...`
- Et `Type* t = const_cast<Type*>(ct) ;`

C++ Le reinterpret_cast

- Instruit le compilateur de considérer les données comme étant du nouveau type
- Permet de faire quasi tout et surtout n'importe quoi

C++ Ce qu'on a couvert

- Les variables sont déclarées et typées
- Mais il est possible de convertir les variables
- Conversions implicites standard ou utilisateurs
- Conversions explicites, par des transtypages
- Des opérateurs existent, qui évitent de recourir aux casts traditionnels



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets

Définir ses propres classes

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La création du modèle de domaine
- La création du modèle de conception
- La traduction des classes en C++
- La mise en œuvre de ces types utilisateurs



C++ La création du modèle de domaine

- La modèle du domaine est une représentation du métier
- Il n'y pas de représentation de l'informatique

C++ Le modèle de conception

- Issu du modèle du domaine
- Ajoute les caractéristiques liées à l'informatique

C++ La traduction des classes en C++

- Les types d'objets sont des classes ou struct
- Les attributs sont des données membres
- Les opérations sont des fonctions membres
- Les associations sont représentées par des données membres de type pointeur, référence ou objet

C++ La mise en oeuvre de ces types utilisateur

- Il faut créer des objets à partir de ces classes
- S'arranger pour que les variables membres soient affectées
- Déclencher les traitements en appelant les fonctions membres

C++ Ce qu'on a couvert

- Les classes C++ sont issues d'un travail en amont visant à :
 - Définir le modèle métier
 - Définir le modèle de conception
- Les classes C++ définissent des nouveaux types, liés à l'utilisateur





Alphorm.com

Le C++, langage orienté objets

Les visibilités de membres

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les zones de visibilité
- Les membres privés
- Les membres protégés
- Les membres publiques



Programmer en C++

alphorm.com™©

C++ Les zones de visibilité

- Les étiquettes possibles sont private, public, protected
- Par défaut, les membres sont private
- Une zone se termine là où une autre commence

C++ Les membres privés

- Ne sont accessibles que de la classe
- Cache les données en général
- Ou bien les fonctions membres à usage interne

C++ Les membres protégés

- Accessibles de la classe ou bien des classes dérivées
- Représente une information non critique
- Utile aux classes filles

C++ Les membres publiques

- En général la visibilité des fonctions membres
- Accessibles de partout
- Représentent les services offerts par la classe

C++ Ce qu'on a couvert

- Les zones de visibilité définissent l'accessibilité aux membres
- Un membre privé ne peut pas être utilisé par l'extérieur de la classe
- Cacher les données en privé
- Rendre les fonctions membres, qui sont des services, publiques
- Attention à la visibilité protégée



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets Manipuler les objets

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les objets automatiques ou dynamiques
- Les accès via « . »
- Les accès via « -> »
- Les messages d'erreur lorsqu'on se trompe d'opérateur de sélection



C++ Les différents types d'objets

- Un objet peut être manipulé via :
 - Son nom
 - Une référence
 - Un pointeur

C++ L'opérateur de sélection « . »

- Utilisable sur la variable objet elle-même, ou bien une référence
- Non surchargeable
- Utilisation :
 - Voiture v;
 - Puis v.demarrer();
 - Ou bien Voiture& r = v;
 - Puis r.demarrer();

C++ L'opérateur de sélection « -> »

- Utilisé sur un pointeur
- Suppose que l'objet est bien alloué
- Surchargeable
- Utilisation :
 - Voiture* p = new Voiture;
 - Puis p->demarrer();

C++ Les messages d'erreur

- Utiliser le mauvais opérateur
- Le message d'erreur signale que la partie gauche n'est pas une classe

C++ Ce qu'on a couvert

- Les différents types d'objets
- Les moyens d'accès aux objets
- L'opérateur « . »
- L'opérateur « -> »





Alphorm.com

Le C++, langage orienté objets

Constructeurs et destructeur

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Les états d'un objet
- Le constructeur par défaut
- La surcharge des constructeurs
- Le destructeur



Programmer en C++

alphorm.com™©

C++ Les états d'un objet

- Un état d'un objet est donné par les valeurs de ses attributs

C++ Le constructeur par défaut

- Si aucun constructeur n'est déclaré, un constructeur est généré
- Sans paramètre

C++ La surcharge des constructeurs

- Il est possible, voire souhaitable, de surcharger les constructeurs

C++ Le destructeur

- Généré par défaut si non déclaré
- Gère la fin de vie de l'objet

C++ Ce qu'on a couvert

- Un objet a des états correspondant aux valeurs prises par ses attributs
- Un constructeur par défaut est généré automatiquement si aucun autre n'est déclaré
- La surcharge est très utile pour les constructeurs
- Un destructeur peut exister pour gérer la fin de vie de l'objet



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets

Copie et move

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- L'opérateur de copie
- Le constructeur par copie
- L'opérateur de move
- Le constructeur de move
- Application de copie et move



C++ L'opérateur de copie

- Lorsque $a=b$
- La variable a est l'objet sur lequel s'applique $operator=$
- Il faut recopier le contenu de b dans a

C++ Le constructeur par copie

- Lorsque $A\ a = b$ ou bien $A\ a(b)$ ou bien $\text{void } f(A\ a)$
- Il faut construire a à partir de b

C++ L'opérateur de move

- Depuis C++11, il existe les opérateurs de move
- Utilisé lorsque le paramètre est une rvalue
- Il faut recopier les données internes et « vider » le contenu de l'objet d'origine

C++ Le constructeur de move

- Lorsque A a(b) et b est une rvalue
- Il faut construire a à partir de b et « vider » b

C++ Les usages et priorités

- Par défaut, le constructeur de copie est utilisé
- Si un constructeur de move existe, il sera utilisé lorsque nécessaire

C++ Ce qu'on a couvert

- Pour gérer la copie, nous avons :
 - Le constructeur de copie
 - L'opérateur de copie
- Depuis C++11, il existe le move
 - Le constructeur de move
 - L'opérateur de move
 - Utilisés lorsque les paramètres sont des rvalue



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets

La dérivation

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La relation de généralisation-spécialisation
- La traduction de l'héritage publique
- La manipulation des objets
- Les visibilités
- Les héritages protégés et privés



C++ La relation de généralisation-spécialisation

- Une relation entre classes
- Permet de traiter des types différents sous une forme commune
- Permet d'hériter des fonctions et données membres
- La base de la notion de polymorphisme

C++ La traduction en C++

- L'héritage publique est la généralisation « classique »
- Lorsque : class B : public A ...
- Tout ce qui est publique sur A reste publique sur B

C++ La manipulation des objets

- Si B hérite de A, il est possible de :
 - `A* p = new B;`
 - `A& r = b;` si B b;
 - Il y a compatibilité entre A et B
- Il est possible d'appeler une fonction ou donnée membre publique de A sur un objet B

C++ Les visibilité

- Suivants les visibilité des membres de la classe mère :
 - Si le membre est private, la classe fille ne voit pas ce membre
 - Si le membre est protected, la classe fille peut y accéder, mais pas l'extérieur
 - Si le membre est public, tout le monde peut y accéder

C++ Les héritages protégés et privés

- Par défaut, l'héritage est privé
- Héritage privé :
 - La classe fille ne définit pas un sous-type de la base
 - Tout ce qui est publique sur la mère est privé sur la fille
- Héritage protégé :
 - La classe fille ne définit pas un sous-type de la base
 - Tout ce qui est publique sur la mère est protégé sur la fille

C++ Ce qu'on a couvert

- La généralisation-spécialisation est une relation entre classes
- Le C++ connaît 3 types d'héritages
- L'héritage publique est l'héritage habituel
- Par défaut, le C++ applique l'héritage privé



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets

La dérivation multiple

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La syntaxe de l'héritage multiple
- La manipulation des objets
- Les collisions et le démasquage
- Un ancêtre commun et l'héritage virtuel



C++ La syntaxe de l'héritage multiple

- Lorsque class C : A,B ...
- Héritage publique ou non : class C : public A, public B ...
- Suit les mêmes règles que l'héritage simple

C++ La manipulation des objets

- Si C hérite de A et de B
- $A^* p = \text{new } C$, mais aussi $B^* p = \text{new } C$
- Les accès aux membres sont bien entendu limités par le type du pointeur

C++ Les collisions et le démasquage

- Lorsque le même membre existe sur plusieurs branches dérivées
- Exemple si m existe sur A et sur B, sur C il existe $A::m$ et $B::m$
- Régulé au niveau de l'appel : $p \rightarrow A::m$ ou bien $p \rightarrow B::m$
- Ou bien réécriture sur C
- Ou bien sur C : `using A::m`, par exemple
- Il existe aussi des problèmes liés à la résolution de la surcharge

C++ Un ancêtre commun et l'héritage virtuel

- Si A et B ont un ancêtre commun O, alors C qui hérite de A et de B possède plusieurs versions de O
- Il faut régler les problèmes de collisions
- L'héritage virtuel permet d'éviter d'avoir plusieurs versions de O

C++ Ce qu'on a couvert

- L'héritage multiple donne la possibilité à une classe d'avoir plusieurs ancêtres
- Il y a collision lorsque ces ancêtres ont les mêmes membres
- Le démasquage consiste à favoriser un ancêtre ou l'autre
- Il peut même y avoir un ancêtre commun, dans ce cas l'héritage virtuel est fortement intéressant
- Noter aussi le problème de résolution de la surcharge à travers les niveaux d'héritage





Alphorm.com

Le C++, langage orienté objets Le polymorphisme

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- L'héritage et la redéfinition
- Le mot-clé « virtual » et le polymorphisme
- L'intérêt du polymorphisme
- L'opérateur `dynamic_cast`



Programmer en C++

alphorm.com™©

C++ L'héritage et la redéfinition

- Lorsque la classe B hérite de A
- Si une fonction m existe sur A, on peut réécrire m sur B
- Alors il existe deux versions de la fonction, selon qu'on fait un appel sur A ou B
- Ne pas confondre redéfinition et surcharge

C++ Le mot-clé virtual et le polymorphisme

- Avec un pointeur sur A, il possible de déclencher la fonction m de B
- Si m est une fonction polymorphe sur A
- Le mot-clé à utiliser est « virtual »

C++ L'intérêt du polymorphisme

- Lorsque le code est écrit sur A^* , il est compatible avec n'importe quel sous-type de A
- Le code ne changera pas malgré le changement de l'objet manipulé

C++ La détection de type avec dynamic_cast

- Le polymorphisme s'oppose à la détection de type des objets
- Parfois nécessaire, peut s'obtenir avec `dynamic_cast`
- Sur pointeur ou référence

C++ Ce qu'on a couvert

- Le polymorphisme ne fonctionne pas par défaut en C++
- Le mot-clé « virtual » permet de mettre en œuvre ce polymorphisme
- Il y a un surcoût à la présence runtime des infos de type
- L'opérateur « dynamic_cast » peut détecter le type de l'objet



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, langage orienté objets Les fonctions générées

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Le constructeur par défaut
- Le destructeur
- Le constructeur de copie
- L'opérateur =
- Le constructeur de move
- L'opérateur move



C++ Le constructeur par défaut

- Est généré lorsque aucun constructeur n'est généré
- Ou bien marqué default
- Peut être supprimé explicitement par delete
- Les 6 fonctions automatiquement générées peuvent être marquées default ou delete

C++ Le destructeur

- Avant sa disparition, un objet voit son destructeur appelé
- Fonction de nettoyage pour toute ressource que l'objet s'est allouée
- Non surchargeable
- Généré automatiquement si aucun destructeur n'existe

C++ Le constructeur de copie

- Utilisé lorsqu'un objet doit être construit à partir d'un autre
- Généré automatiquement si ni constructeur de move, ni opérateur de move (en C++03, ceux-ci n'existent pas)

C++ L'opérateur =

- Pour écraser un objet par un autre
- Généré automatiquement si ni constructeur de move, ni opérateur de move (en C++03, ceux-ci n'existent pas)

C++ Le constructeur de move

- Appelé automatiquement pour éviter une recopie
- Généré s'il n'existe aucun constructeur de copie, opérateur de copie, ni destructeur
- Il faut aussi qu'il n'y ait pas de membres constants

C++ L'opérateur move

- Remplace l'usage de la copie
- Généré s'il n'existe aucun constructeur de copie, opérateur de copie, ni destructeur
- Il faut aussi qu'il n'y ait pas de membres constants

C++ Ce qu'on a couvert

- En C++, il y a 6 fonctions générées automatiquement
- En plus des 4 du C++03, il faut ajouter les move
- Depuis C++11, il est possible de marquer « delete » ou « default » ces fonctions automatiquement générées





Alphorm.com

Le C++, les templates

Utiliser des classes templates

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La syntaxe d'utilisation de `vector<T>`
- La génération de types différents
- La syntaxe avec `map<K,V>`
- Les paramètres optionnels



Programmer en C++

alphorm.com™©

C++ La syntaxe d'utilisation de vector<T>

- Pour utiliser vector, il faut inclure <vector>
- Puis choisir un type de contenu pour vector
- Par exemple, vector<int>
- Créer un objet de ce type : vector<int> v1;
- Ici nous avons un vecteur vide

C++ La génération de types différents

- A priori, vector<int> et vector<double> sont des types différents
- Si vector<int> v1 et vector<double> v2
- Alors v1 = v2 n'existe pas ...

C++ La syntaxe de map

- Avec map, il faut fournir 2 paramètres
- Inclure <map>
- Un tableau associatif attend clé et valeur :
- Par exemple map<char,string>

C++ Les paramètres optionnels

- La syntaxe de map permet l'utilisation d'un troisième paramètre
- On peut l'omettre, c'est un paramètre optionnel
- Pour map, map<char,string> et map<char,string,less<char>> sont équivalents

C++ Ce qu'on a couvert

- L'utilisation d'une classe template suppose en général l'inclusion du corps de la classe
- Il est possible d'avoir plus d'un paramètre template
- Certains paramètres sont optionnels



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les templates

La création d'une classe template

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Une exemple de classe qui existerait en plusieurs versions
- La classe template est un modèle pour les classes réelles
- La classe peut nécessiter plus d'un paramètre
- Les paramètres peuvent aussi être des valeurs



C++ Un exemple de classe en plusieurs versions

- Pour avoir besoin des templates, il faut une classe dont les versions différent selon le type
- Par exemple pile_de_int et pile_de_double
- Implémentation quasi-identique, sauf le type int ou double

C++ La classe template est un modèle

- Pour éviter les doublons, nous créons la classe template
- Il faudra créer pile_de_T
- Syntaxe : `template <typename T> class pile { ... };`

C++ Il peut y avoir plus d'un paramètre

- Pour un tableau associatif :
- Syntaxe : `template<typename K, typename V> class tableau { ... };`
- Et proposer des valeurs par défaut
- Comme : `template <typename K, typename V, typename S = less<K> >
class tableau { ...};`

C++ Certains paramètres sont des valeurs

- Avec un exemple dans lequel la taille maxi est une valeur
- Exemple avec la pile : `template <typename T, int max_size> pile {...};`
- A l'utilisation : `pile<int,1024> p1;`

C++ Ce qu'on a couvert

- Une classe template ressemble à une classe, mais certains paramètres sont laissés à l'utilisateur
- Il est possible d'attendre plusieurs paramètres
- On peut proposer des valeurs par défaut
- Certains paramètres sont des valeurs





Alphorm.com

Le C++, les templates

Une fonction template

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Une fonction qui existe en plusieurs versions selon les types
- La fonction template réduit les redondances
- La fonction peut avoir de nombreux paramètres, valeurs par défaut ...
- Il existe un grand nombre de fonctions déjà faites



Programmer en C++

alphorm.com™©

C++ Une fonction en plusieurs versions

- Nous prenons le cas de la fonction max
- Appliquée à int ou à double
- Implémentation idendique
- Utilisation de l'opérateur <
- A priori, on utilise un entête et un corps

C++ La fonction template réduit les dépendances

- Pour éviter les doublons, nous créons la fonction template
- L'implémentation est la même
- Le type est générique
- Déclaration et définition dans le même fichier
- Lors de l'utilisation, il n'est pas nécessaire de spécifier le type

C++ Paramètres multiples, valeurs par défaut

- Il est possible d'avoir plusieurs paramètres
- Une valeur par défaut permet d'éviter de fournir les paramètres
- Il est possible de fournir des valeurs

C++ Des fonctions template existent

- Exemples avec max, for_each, find_if ...

C++ Ce qu'on a couvert

- Une fonction template est paramétrée par le type sur lequel est elle appliquée
- Il n'est pas nécessaire de spécifier le type à l'appel
- Les paramètres suivent les mêmes règles que pour les classes
- Il est possible de spécialiser les fonctions template



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les templates

Template avancés ...

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Une classe template peut contenir un objet paramétré
- Une classe template peut hériter d'une classe template
- Une classe non-template peut hériter d'une version d'un template
- Un paramètre template peut être lui-même un type template
- Avec « typedef », on simplifie l'écriture du template



C++ Une classe template contient un template

- Une classe template contient ...
- Un objet d'un type ...
- Qui est une classe template
- Aussi possible de référencer une fonction template (amie)

C++ Une classe template hérite ...

- D'une classe template
- Il y a concordance des types
- On peut mélanger héritage et objet inclus

C++ Une classe non-template hérite ...

- D'une classe template
- Il faut donc choisir le paramètre

C++ Un paramètre template...

- Peut être un type template
- Exemple avec `vector< complex<int> >`
- Dans la version C++03, ne pas coller les `> >`

C++ Avec « typedef », l'écriture simplifiée

- Plutôt que d'utiliser le type `vector<complex<int> >`
- En faisant : `typedef vector<complex<int> > vector_comp ;`
- Puis `vector_comp ma_variable ;`
- Cela évite de répéter une syntaxe compliquée

C++ Ce qu'on a couvert

- Une classe template peut contenir un objet paramétré
- Une classe template peut servir de classe de base
 - À une classe template
 - À une classe non-template
- Un paramètre de template peut être lui-même un type template
- Avec « typedef », on simplifie la syntaxe



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les templates

Limites et pièges

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- L'implémentation d'un template suppose la disponibilité de fonctionnalités sur les types paramètres
- Une erreur peut apparaître si le type ne possède pas ces fonctionnalités
- Ou bien la sémantique peut ne pas être respectée
- Il est possible de spécialiser des classes ou fonctions template



C++ L'implémentation suppose la disponibilité ...

- De certaines fonctions ou opérateurs sur les types template
- Exemple de max, avec l'opérateur <
- Tant que l'opérateur existe sur les types utilisés en paramètres, tout va bien

C++ Si le type ne possède pas l'opérateur ...

- Une erreur apparaît à la compilation
- Le message d'erreur n'est pas forcément explicite
- Il apparaît au gré du type utilisé en paramètre

C++ La sémantique peut ne pas être respectée

- Exemple du max
- Avec des char*
- Le maximum qui ressort n'est pas l'ordre lexicographique
- L'erreur est d'autant plus grave qu'elle n'apparaît pas ... !

C++ Les spécialisations de template

- Il faudrait donc spécialiser le template
- Syntaxe `template<> char* max(char*, char*)`
- Evite la génération de max pour char*

C++ Les spécialisations de classes templates

- La spécialisation existe aussi pour les classes templates
- Exemple avec

```
template<class _Ty>
    struct _Is_floating_point
        : false_type
    { // determine whether _Ty is floating point
    };
```

```
template<>
    struct _Is_floating_point<float>
        : true_type
    { // float is floating point
    };
```

C++ Ce qu'on a couvert

- L'implémentation d'un template suppose la disponibilité de certaines fonctions sur les paramètres
- Soit les fonctions n'existent pas
- Soit la sémantique n'est pas respectée
- Il faut spécialiser le template, fonction ou classe



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les exceptions

Gérer les erreurs

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Intérêt général des exceptions
- Récupérer une erreur levée bas niveau
- Les possibilités de récupération des exceptions
- Les exceptions standards



C++ Intérêt général des exceptions

- Le mécanisme des exceptions est un mécanisme de gestion des erreurs
- Pas facile à bloquer
- Sépare le code nominal du code d'erreur
- Explicite : les exceptions sont typées
- Riches : les exceptions sont des objets

C++ Récupérer des erreurs levées bas niveau

- Si une exception est levée ...
- Le mot-clé try introduit le bloc de code nominal
- Le mot-clé catch introduit chaque bloc de code de gestion d'erreur
- Plusieurs blocs catch peuvent se suivre
- Le bloc catch est typé, sauf catch(...)
- Possibilité d'imbriquer les try

C++ Les possibilités de récupération

- Avec un catch
- Possibilité de manipuler des super-types d'exceptions
- Si personne ne récupère l'exception ... terminate
- Spécifier les exceptions levées ... unexpected (déprécié en C++11)

C++ Les exceptions standards

- La classe mère des exceptions standards est exception (<exception>)
- La fonction what() retourne une chaîne explicative
- Des sous-classes d'exceptions : bad_alloc (alternative new avec std::nothrow) , bad_cast, ...
- Ou aussi : logic_exception, runtime_error (avec d'autres sous-classes)

C++ Ce qu'on a couvert

- Le mécanisme est de gestion des erreurs par exceptions est efficace
- Pour récupérer les exceptions, il faut utiliser try/catch
- Si personne ne gère les exceptions, on peut intervenir en dernier recours
- La librairie standard utilise des exceptions pour signaler des erreurs





Alphorm.com

Le C++, les exceptions

Lever ses exceptions

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Lever des exceptions avec throw
- Relancer une exception avec throw
- Spécifier les levées d'exception



Programmer en C++

alphorm.com™©

C++ Lever des exceptions avec throw

- Pour lever une exception, il faut utiliser throw
- Comme throw 42; => dans ce cas, le type de l'exception est un entier
- En général, il est préférable de lever une exception riche en données
- Donc un objet, d'une classe bien choisie
- Il est préférable de lever un objet (valeur), il sera récupéré par référence

C++ Relancer une exception

- Dans un bloc catch ...
- Il est possible d'utiliser throw;
- L'exception repart vers le try englobant
- Ne pas lever une nouvelle exception (voir throw_with_nested)

C++ Spécifier les levées d'exception

- Dans la déclaration d'une fonction
- Mettre `throw(ex1,ex2)` signifie qu'elle ne lève que `ex1, ex2`
- Si `throw()`, aucune exception ne sera levée
- Obsolète, remplacée en C++11 par `noexcept(expr)`
- Avec C++03, appel éventuel de `unexpected`
- Obsolète en C++11

C++ Ce qu'on a couvert

- Lever une exception est simplement utiliser le mot-clé `throw` pour retourner un objet exception à l'appelant
- Il est possible de relancer une exception
- Il est possible de spécifier ce que lève une fonction





Alphorm.com

Le C++, les exceptions

Créer ses classes d'exceptions

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Présentation de deux fonctions qui peuvent lever des erreurs
- Lever une exception sans créer sa propre classe : limitée
- Créer une classe d'exception pour gérer un premier cas
- Créer une seconde classe, et l'intégrer dans une famille



Programmer en C++

alphorm.com™©

C++ Des fonctions qui nécessitent les exceptions

- Imaginons deux fonctions qui gèrent un tableau de taille fixe
- La première retourne un élément / index
- La seconde retourne l'index /élément
- Il est possible que :
 - L'index ne soit pas bon
 - L'élément n'existe pas

C++ Lever une exception banale

- Une exception de type banalisé n'est pas pratique
- Pas d'infos sur l'erreur
- Pas de localisation
- ...

C++ Créer sa propre classe d'exception

- Pour la première fonction :
 - Récupération de l'indice tenté
 - Récupération de la taille max du tableau
 - Un constructeur
 - Une fonction de d'explication

C++ Créer une seconde classe

- Pour la seconde fonction, nous pouvons faire pareil
- Mais nous pouvons ensuite généraliser
- Le polymorphisme va fonctionner

C++ Ce qu'on a couvert

- Créer une classe d'exception n'est pas bien difficile
- Elle permet de capter le contexte d'erreur
- Préparer au minimum un constructeur et des fonctions utiles
- Il est possible d'intégrer ses classes dans une hiérarchie



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les amis

Une fonction amie

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Une classe pour les entiers de Gauss
- Une fonction membre pour ajouter 2 entiers de Gauss
- L'addition avec une fonction non-membre
- Syntaxe avancée pour les fonctions amies



C++ Une classe pour les entiers de Gauss

- Une classe avec deux champs entiers
- Choix de constructeurs ou initialisation directe
- Une fonction membre d'affichage

C++ Utiliser une fonction membre

- Pour ajouter deux entiers de Gauss
- Nous choisissons une fonction membre
- Facilité de réalisation
- Accès direct aux informations

C++ Problèmes avec une fonction membre

- Implémentation différente de $a+b$ et $b+a$
- Il est très simple de se tromper dans l'implémentation
- Certaines écritures ne fonctionneront tout simplement pas

C++ Utiliser une fonction non-membre

- Si nous utilisons une fonction non-membre
- Difficile d'accéder aux données -> getter ?
- Mais rôles symétriques des paramètres simples à implémenter
- Déclarer la fonction non-membre comme amie

C++ Syntaxe avancée pour les fonctions amies

- Lorsqu'il est légitime de déclarer la fonction avec la classe
- Enchaîner : friend ... et définir la fonction dans la classe...
- Mais ce n'est pas une fonction membre !!

C++ Ce qu'on a couvert

- Pour accéder aux données privées d'une classe
- Soit implémenter une fonction membre
- Soit une fonction non-membre déclarée comme amie
- Une syntaxe spécifique permet de simplifier la définition des fonctions amies



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les amis

Une classe amie

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Une classe de conteneur
- Une classe d'itérateur
- Déclarer la classe itérateur comme amie du conteneur



C++ Une classe de conteneur

- Une classe `array_list`, qui :
 - Contient une implémentation tableau
 - Offre des service d'accès
- Son implémentation est connue par l'utilisateur

C++ Une classe itérateur

- Un itérateur propose de parcourir le conteneur
- Afin d'isoler l'utilisateur de l'implémentation du conteneur
- L'itérateur doit accéder au ième élément du conteneur

C++ Déclarer l'itérateur en tant que amie

- Une classe amie a accès aux données privées
- Plus besoin d'accesseurs
- C'est le conteneur qui fait la déclaration

C++ Ce qu'on a couvert

- Les membres privés d'une classe ne sont pas accessibles de l'extérieur
- Une classe déclarée amie a accès aux membres privés
- C'est un accès privilégié, qui n'est pas transmissible, ni héritable...



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les opérateurs

Les opérateurs unaires et binaires

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Réaliser un opérateur binaire
 - Fonction membre
 - Fonction non-membre
- Réaliser un opérateur unaire
 - Fonction membre
 - Fonction non-membre
- Opérateur non-membre comme fonction amie



C++ Réaliser un opérateur binaire

- En fonction membre
- Un seul paramètre
- Qui correspond à l'opérande de droite
- L'opérande gauche est this

C++ Opérateur binaire non-membre

- Fonction non-membre
- Prend 2 paramètres, respectivement opérande de gauche et de droite
- Implémentation symétrique sur les 2 paramètres

C++ Réaliser un opérateur unaire

- Fonction membre
- Pas de paramètre
- L'opérande est this
- Facilité d'implémentation

C++ Opérateur unaire fonction non-membre

- Fonction avec un paramètre
- Ce paramètre est l'opérande
- Obligation d'accès aux données ou bien amie

C++ Les opérateurs amis

- Si l'opérateur est une fonction non-membre
- Il doit avoir accès aux données internes ...
- Le mieux est de le déclarer ami de la classe

C++ Ce qu'on a couvert

- Un opérateur peut être unaire, binaire
- Il peut être réalisé sous la forme d'une fonction membre ou non
- Pour l'implémentation non-membre, il peut être déclaré ami



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les opérateurs

Réaliser ses opérateurs

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Faire correspondre la sémantique : opérateur +
- Un opérateur + qui modifie l'objet courant (this)
- Un opérateur + implémenté en fonction membre
- Implémenter + avec +=
- Remarque sur les conversions implicites



C++ Faire un opérateur +

- Qui n'additionne pas !
- Aucune détection sémantique par le compilateur
- Très difficile à détecter
- Sur des types où la sémantique n'est pas naturelle

C++ Un opérateur + modifiant this

- Un opérateur + réalisé en fonction membre
- Va vite modifier l'objet courant
- Erreur !

C++ Fonction membre, rôles différents

- Opérateur + fonction membre
- Appliqué sur un type convertible en le type courant
- Le compilateur ne trouvera pas la conversion

C++ Implémenter + avec +=

- Pour obtenir la cohérence +, +=, ++ ...
- Réaliser + avec +=
- Simplifie l'écriture de la fonction non-membre

C++ Remarques sur les conversions implicites

- Lors de l'application de l'opérateur
- Si des conversions existent sur les paramètres
- On ne sait plus quel opérateur est utilisé
- Ou bien on génère des ambiguïtés

C++ Ce qu'on a couvert

- Nous avons donné quelques conseils :
 - Les opérateurs binaires sont plutôt des fonctions non-membres
 - Les opérateurs unaires modifiant this sont membres
 - Garder la sémantique des opérateurs
 - Préférer la surcharge aux conversions implicites



Programmer en C++

alphorm.com™©



Alphorm.com

Le C++, les opérateurs

Les opérateurs [] et ()

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- L'opérateur []
- Application à un conteneur
- L'opérateur ()
- Application aux foncteurs



C++ Un conteneur

- Nous réalisons un conteneur
- Données sous forme de tableau
- Besoin d'exporter le ième élément

C++ L'opérateur []

- Avec un opérateur [int]
- Ecriture simple et intuitive
- Deux syntaxes possibles
 - Sans modification possible des données
 - Avec passage par référence

C++ Implémenter l'opérateur ()

- Syntaxe simple operator () (a,b,...)
- Exemple d'implémentation à la place de []

C++ Opérateur () pour mimer une fonction

- Pour une fonction template qui fait un appel fcnt(int a)
- Créer une classe :
 - L'opérateur () prend un seul paramètre
 - Le constructeur prend un paramètre et garde la valeur
 - L'opérateur () applique un appel avec les 2 paramètres

C++ Ce qu'on a couvert

- Nous avons présenté les opérateurs [] et ()
- L'opérateur [] permet une syntaxe d'indexeur
- L'opérateur () sera utile pour les foncteurs





Alphorm.com

Le C++, les opérateurs L'opérateur ->

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Imaginons un wrappeur de pointeur
- Son destructeur détruit la ressource pointée
- Remplaçons la fonction d'accès à l'objet pointé par un opérateur->



Programmer en C++

alphorm.com™©

C++ Un wrapper

- Une classe « wrapper » encapsule un pointeur
- Le pointeur donne accès à un objet
- Lorsque le wrapper est détruit, l'objet est détruit

C++ Accès aux données pointées

- Une fonction d'accès à l'objet est nécessaire
- Il faut utiliser -> pour accéder aux fonction de l'objet encapsulé

C++ Implémenter l'opérateur ->

- Remplaçons la fonction d'accès
- Avec P* operator->()
- L'interface de l'objet encapsulé est promu

C++ Ce qu'on a couvert

- Nous avons proposé l'opérateur ->
- La syntaxe est originale, il y a promotion de l'interface encapsulée
- Très utile pour la réalisation des smart-pointers





Alphorm.com

Le C++, les opérateurs

Les opérateurs de conversion

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- La conversion par le constructeur à 1 paramètre
- Exemple avec une classe complexe
- Conversion de int vers complexe
- Conversion de complexe en double
- Problème des conversions implicites / opérateur



Programmer en C++

alphorm.com™©

C++ La conversion due au constructeur

- Un constructeur à 1 paramètre
- Réalise un opérateur de conversion
- Implicite, mais peut être empêché par « explicit »

C++ Une classe complexe

- Contenant deux données int
- Un constructeur avec liste d'initialisation
- Un constructeur à 1 paramètre int
- On peut créer un complexe à partir d'un int

C++ Implémenter l'opérateur int()

- Pour convertir de façon implicite complexe en int
- Sémantique pas naturelle
- Mot-clé « implicit » depuis C++11

C++ Autre opérateur de conversion

- Pour convertir complexe en double
- Sémantique à choisir, ici module

C++ La surcharge et la conversion implicite

- Lorsqu'on applique une fonction surchargée
- Les paramètres peuvent être convertis
- Si implicitement, difficile détection de la fonction appelée
- Exemple avec l'opérateur + sur les complexe

C++ Ce qu'on a couvert

- Nous avons montré la syntaxe des opérateurs de conversion
- La sémantique n'est pas forcément naturelle
- Les conversions implicites peuvent poser des problèmes





Alphorm.com

Le C++, la librairie standard

Les conteneurs séquentiels

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Présentation générale des conteneurs de la librairie standard
- Les conteneurs séquentiels
- Le vector



Programmer en C++

alphorm.com™©

C++ Présentation générale des conteneurs

- Les conteneurs séquentiels : vector, list, forward_list, deque
- Les adaptateurs : stack, queue, priority_queue
- Les conteneurs associatifs ordonnés : map, multimap, set, multiset
- Les conteneurs associatifs non-ordonnés : unordered_map, unordered_multimap, unordered_set, unordered_multiset
- Les pseudo-conteneurs : string, array, valarray, bitset

C++ Les conteneurs séquentiels

- Les conteneurs : vector, list, forward_list, deque
 - Constructeurs surchargés : par défaut (vide), avec n éléments, ...
 - Fonctions size(), empty(), capacity(), reserve(), resize(), clear()
- Le conteneur vector est un tableau associatif
 - Accès avec [] en temps constant
 - Accès aléatoire
 - Possède shrink_to_fit() (comme deque et string)

C++ Manipuler vector

- Construit éventuellement sans taille
- Redimensionnement automatique
- Accès via back(), front()
- Ou bien at(i), [i]
- Alimentation avec push_back(e), insert(...)

C++ Traiter vector avec les iterator

- Accès aux itérateurs par : begin(), cbegin(), rbegin(), crbegin()
- Marqueur de fin : end(), cend(), rend(), crend()
- Abstraction vis-à-vis de l'implémentation

```
for(auto p = v.begin(); p!=v.end(); ++p)
    cout <<*p <<'\n'

for( auto& x : v)
    cout <<x <<'\n'
```

C++ Le vector est efficace

- Utiliser vector comme conteneur par défaut
- Les fonctions insert et push_back sont plus efficaces sur vector
- Les listes souvent vides doivent utiliser forward_list
- Les éléments des conteneurs STL doivent supporter copie et move
- Utiliser conteneur de pointeurs ou de smart_pointers
- Pour traverser un conteneur, utiliser for(each) ou un itérateur

C++ Ce qu'on a couvert

- Nous avons présenté en largeur les conteneurs de la STL
- Les conteneurs séquentiels sont 4, il y a des adaptateurs
- Le vector est un conteneur de choix
- Il faut utiliser les itérateurs pour traverser un conteneur





Alphorm.com

Le C++, la librairie standard

Les conteneurs associatifs

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Présentation générale des conteneurs associatifs
- Les conteneurs associatifs ordonnés
- Les conteneurs associatifs non-ordonnés



Programmer en C++

alphorm.com™©

C++ Présentation des conteneurs associatifs

- Les conteneurs associatifs ordonnés : map, multimap, set, multiset
- Les conteneurs associatifs non-ordonnés : unordered_map, unordered_multimap, unordered_set, unordered_multiset
- Recherche basée sur des clés
- Soit simples, une entrée unique par clé
- Multi, plusieurs entrées par clé

C++ Les conteneurs associatifs ordonnés

- Les constructeurs surchargés : valeurs par défaut pour :
 - Le comparateur (<)
 - L'allocateur (new, delete)
 - La source
- Les accès via :
 - [k] ou at(k), pour map
 - Ou bien find(k), lower_bound(k), upper_bound(k)
- Insertion :
 - Avec insert(value_type) (remarque : value_type : pair<const K, T>)

C++ Les conteneurs associatifs non-ordonnés

- Basés sur des hash-table
- Partagent la plupart des opérations avec les ordonnés
- Pas besoin de relation d'ordre, mais utilise une fonction `hash<T>` et `equal_to<T>`
- La fonction `hash<T>` existe pour des types usuels
- La fonction `equal_to<T>` est par défaut `==`
- Ordre non garanti, dépend de `hash<T>`

C++ Ce qu'on a couvert

- Nous avons présenté les conteneurs associatifs
- Les conteneurs associatifs ordonnés tels qu'en C++03, basés sur des arbres binaires équilibrés
- Les conteneurs associatifs non-ordonnés (C++11), basés sur des tables de hachage.





Alphorm.com

Le C++, la librairie standard

Les algorithmes

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Plan

- Présentation générale des algorithmes
- Les algorithmes ne modifiant pas les conteneurs
- Les algorithmes modifiant les conteneurs
- Les algorithmes de recherche et de tri
- Les algorithmes comme max, min...



Programmer en C++

alphorm.com™©

C++ Présentation des algorithmes

- Environ 80 algorithmes existent dans <algorithm>
- Ils travaillent sur des séquences entre itérateurs
- Certains ont besoin d'accès aléatoire
- La plupart retourne la fin de la séquence pour signaler « non trouvé »
- Permet d'obtenir un code sûr, utilisant des fonctions bien nommées, connues

C++ Les algorithmes ne modifiant pas

- Les algos :
 - Applique : `for_each(b,e,f)`
 - Predicats : `all_of(b,e,f)`, `any_of(b,e,f)`, `none_of(b,e,f)`
 - Compte : `count(b,e,v)`, `count_if(b,e,f)`
 - Recherche : `find(b,e,v)`, `find_if(b,e,f)`...
 - Comparaison : `equal(b,e,b2)`, `mismatch(b,e,b2)`
 - Recherche de séquences : `search(b,e,b2,e2)`, `search_n(b,e,n,v)`...

C++ Les algorithmes modifiant les conteneurs

- Nombreux algos : transform, copy, unique, remove, replace, rotate, ...
- Permutations
- Remplissage
- Echanges

C++ Les algorithmes de recherche et tri

- Comme sort(b,e) ou sort(b,e,f), utilisant <
- Autres algos : stable_sort, partial_sort, ...
- Nécessite un accès aléatoire
- Algos binary_search(...)
- Fusion avec merge(b,e,b2,e2, out)...
- Traitements d'ensembles
- Comparaison lexicographique

C++ Min et max

- Base : $\min(a,b)$, $\min(a,b,f), \dots$ max idem
- Obtenir une paire : $\text{pair}(x,y) = \text{minmax}(a,b)$
- Min et max sur des séquences : $\text{min_element}(b,e) \dots$

C++ Ce qu'on a couvert

- Il existe de nombreux algorithmes déjà implémentés
- Le bénéfice est un code plus lisible, entre autres
- Les algos opèrent sur des séquences
-





Alphorm.com

Le langage C++ Conclusion

Site : <http://www.alphorm.com>
Blog : <http://www.alphorm.com/blog>
Forum : <http://www.alphorm.com/forum>

Fabien Brissonneau

Consultant, concepteur et formateur
Objets Logiciels

Contact : fabien.brissonneau@gmail.com

Programmer en C++

alphorm.com™©

C++ Conclusion

- Nous avons présenté le langage C++
- Un langage proche du C, proche de la machine
- Mettant en œuvre des concepts orientés objets



Programmer en C++

alphorm.com™©

C++ Conclusion

- Le langage est dans sa version C++11
- Supporté maintenant par la plupart des environnements modernes
- Une syntaxe plus simple
- Une librairie plus étoffée
- Un langage moderne et efficace
- Une version arrive nommée C++14, mise à jour mineure de C++11