Initiation à C++

Cahier d'Exercices

Août 2016

Nicolas BENOIT

Les Bases de C++

Variables, Conditions et Boucles

- Créez deux variables string contenant respectivement "Hello" et "World", créez une variable int contenant 2016. A l'aide de ces trois variables, remplissez une autre variable string avec la phrase "Hello World 2016" et affichez la.

- Créez une nouvelle variable type String et associez lui la chaine de caractères "42". Trouvez une solution pour afficher le résultat numérique de l'addition de l'int contenant 2016 et de cette variable string.

- Entrez manuellement à l'aide de cin les valeurs de deux int x et y. Réalisez une boucle comptant de x en x entre les nombres de 0 à y

- Créez une variable string vide. A l'aide d'une boucle, remplissez la de dix fois le caractère étoile (*) et affichez la

Point Cours : Pour représenter un saut de ligne dans une string utiliser la chaîne de caractères "\n".

- A partir de l'exercice précédent, à l'aide de deux boucles imbriquées, entrez maintenant dans la variable une chaine de caractères représentant un rectangle de 10 X 10 étoiles

Options

- Donnez à int1 la valeur 13 et à int2 la valeur 5. Affichez le résultat de la division de int1 par int2. Cherchez une solution pour avoir la réponse précise à la division

- Créez deux variables char1 et char2. Faites en sorte que char2 contienne la valeur de char1 et inversement.

- Reprenez votre boucle comptant de x en x de 0 à y. Reproduisez son comportement à l'aide d'un while(1) (mauvaise pratique à ne jamais implémenter dans du vrai code ;)), une condition et un break.

- A l'aide d'étoiles dessinez maintenant un triangle rectangle. Puis un losange.

Tableaux et Méthodes

Point Cours :

Pour générer une lettre aléatoire il faut

- importer cstdlib et ctime

- "initialiser" le random avec la ligne : srand(time(0));

- et notre lettre aléatoire est créée par la ligne : char C = rand() % 26 + 'A';

Explications :

La vérité sur les chars : un char stocke en fait un nombre entre 0 et 127 et affiche son caractère correspondants dans la table ASCII (la table ASCII liste 128 caractères dont les lettres).

rand() % x retourne un nombre entre 0 et x-1, du coup ici rand() % 26 retourne un nombre entre 0 et 25. Le + 'A' permet de se placer dans la zone de la table ASCII avec les lettres de l'alphabet en majuscule.

Ainsi, si le rand() tire 0, on obtient A, s'il tire 1, on obtient B jusqu'au 25 qui retourne Z.

Pour ce qui est de l'initialisation, rand() ne génère pas une valeur aléatoire mais pseudo aléatoire. Ici notre paramètre pour "créer" de l'aléatoire est l'heure qu'il est, déterminé par time(), srand() appliquant ce paramètre à rand().

Autre Point Cours :

Pour ajouter un char dans un string. Créez un string vide et concaténez lui des chars avec l'opérateur + ou +=.

- Réaliser une méthode motAleatoire() retournant des strings de 7 lettres aléatoires. Par exemple "ADGRTP", "HYETRPO" ou encore "CHIANTI"

- Créez une méthode tableauAleatoire() retournant un vecteur contenant 25 mots de 7 lettres aléatoires

Options

- Créez dictionnaire, un vecteur de string contenant quelques mots existants de 7 lettres

- Créez la fonction checkDico qui prend en argument dictionnaire et une string générée aléatoirement et renvoi un booléen. True si le mot est dans le dictionnaire, false sinon.

Créer ses Classes

Tous ces exercices doivent se réaliser en écrivant un fichier.cpp et un fichier .h par classe. Je vous conseille de créer un nouveau projet (au moins un nouveau dossier pour l'exercice suivant)

Classes en général : L'appartement

- Créez une classe Appartement avec comme variables de classe deux int : m_nombrePieces et m_superficie et deux String : m_couleurMur et m_typeSol. Codez le constructeur de manière à ce qu'il initialise les attributs. Développez la méthode description() qui retourne un string avec un petit texte décrivant ces éléments. Faîtes en sorte que tous les éléments, attributs et méthodes soient publics.

- Dans le main(), créez une instance de Appartement, appelez description(), modifiez un attribut de cette instance (synthaxe : instance.attribut = nouvelleValeur) et appelez de nouveau description().

- Passez les attributs en privés et constatez la différence de comportement.

- Créez la méthode repeindre(String nouvelleCouleur) qui change la couleur du mur par nouvelleCouleur. Repeindre() est un setter pour l'attribut m_couleurMur.

- Créez maintenant une nouvelle classe Canape avec comme attributs un int : m_nombrePlaces et deux String : m_motif et m_couleur. Ajoutez lui un constructeur et une méthode descriptionCanape(). Créez une instance de Canape et testez.

- Ajoutez maintenant à la classe Appartement un nouvel attribut : m_canapeSalon qui contiendra un objet de type Canape. Modifiez le constructeur d'Appartement pour qu'il puisse recevoir ce Canape. Modifiez maintenant la méthode description() pour qu'elle fasse également appel à la méthode descriptionCanape() de notre objet Canape.

Point cours : Lorsque l'on fait appel à la méthode d'un objet par un pointeur, on écrit plus objet.methode() mais objet->methode()

- Mettez en place un setter sur le motif du canape. Modifiez le motif après avoir instancié l'objet Appartement. Testez que si vous modifiez l'objet Canape cela n'influe pas sur le Canape de l'Appartement. Changez l'attribut m_canapeSalon pour ce soit maintenant un shared_pointer sur Canape. Modifiez maintenant le Canape à l'extérieur de Appartement, cela a t-il un impact sur le Canape dans l'Appartement ? Pourquoi ?

Je vous conseille de créer également un nouveau projet pour cet exercice.

Héritage : les Félins

- Créez une classe Felin avec comme attributs deux String : m_nom et m_espece et un int m_age. Créez un constructeur pour initialiser les attributs. Ajoutez les méthodes seNourrir() et sePresenter(). seNourrir() retourne un string contenant "Je mange" et sePresenter() retourne un string contenant "Je suis *nom* j'ai *age* ans, je suis un *espèce*".

- Créez la classe ChatSauvage héritant de Felin. Elle a comme attribut supplémentaire un string : proiePreferee. Modifiez les méthodes seNourrir() et sePresenter() pour intégrer ce nouvel attribut.

- Créez la classe ChatDomestique héritant également de Felin. ChatDomestique a comme nouvelle variable un string couleurCollier. Modifiez la méthode sePresenter() pour l'intégrer. La méthode seNourrir indiquera toujours "des croquettes".

UML

Dessinez les diagrammes de classe des exercices d'Appartement et de Felin. Faites le sur papier si apprendre l'utilisation de Dia ne vous intéresse pas ;)

Surcharge et Polymorphisme

Reprenez l'exercice sur les Félins

- Pour un programme de suivi des chats d'un quartier, surchargez le constructeur de ChatDomestique de manière à pouvoir créer un chat domestique avec uniquement un nom et une couleur de collier (avec initialisation de l'âge à 0 et l'espèce "croisement") ou avec un nom, une espèce et une couleur de collier (avec l'âge initialisé à 0).

- Parce que les ChatSauvages peuvent se nourrir de plusieurs proies différentes, surchargez la méthode seNourrir de manière à ce qu'elle puisse prendre un argument string proieDuJour et l'affiche en lieu et place de sa proie préférée.

- Créez plusieurs ChatDomestiques et ChatSauvages et créez un tableau pour les contenir. Ecrivez une boucle appelant la méthode sePresenter de tous les Felins du tableau. Assurez vous que les ChatDomestiques indiquent leur couleur de collier et les ChatSauvages leur proie préférée.

- Faîtes en sorte que Felin deviennent une classe abstraite, de manière à ce que l'on ne puisse pas instancier de Felin mais uniquement des ChatDomestiques et des ChatSauvages.

Concepts Avancés

Les Templates

Ecrire une fonction prenant en argument tous les types de tableaux numériques et qui retourne la plus grande valeur de ce tableau.

Sans toucher à aucune de vos classes actuelles, développez la classe FelinTemplate, réécriture de la classe Felin sans ses classes filles, de manière à ce son attribut âge puisse prendre des entiers ou des décimaux.

Les Constructeurs de Copie

Si vous êtes allé jusqu'au bout de l'exercice de l'Appartement, vous devriez avoir mit un shared_ptr sur l'attribut Canape de votre Appartement pour que modifier le Canape dans le main modifie également le Canape de l'Appartement. Gardez ce fonctionnement.

Faites maintenant une copie de votre Appartement. Modifiez le Canape dans le main. Constatez que les Canapes des deux Appartements ont été modifiés également.

Modifiez le constructeur de copie d'Appartement pour qu'il génère un nouveau Canape propre au nouvel Appartement.

Pour que le Canape créé par le constructeur de copie fonctionne il faut lui donner une place en mémoire. Pour cela, utilisez un make_shared.

Les Exceptions

Ajoutez un setter sur le nombre de pièces de votre Appartement.

Nous voulons nous assurer que lorsque l'on modifie l'agencement de son Appartement on ne puisse pas indiquer qu'il ait moins d'une pièce.

Intégrez ce concept en ajoutant une exception dans votre nouveau setter. D'abord avec un throw simple puis avec une classe ExceptionAppartementNombrePieces.

La Surcharge D'opérateur

Reprenez la classe Durée (fournie) de l'exemple du cours, implémentez lui les opérateurs -, += et <.

Les éléments de la STD

Conteneurs et Itérateurs

Reprenez la classe Durée.

Créez un vecteur contenant des durées.

Créez une map ayant comme clef des strings et comme valeur des durées. Remplissez la avec quelques trajets.

Dans les deux cas listez l'intégralité de votre conteneur à l'aide d'un itérateur.

Ajouter de nouveaux éléments dans votre vecteur, précisément après un élément existant précis.

Foncteurs et Algorithmes

Si vous ne l'avez pas fait, implémentez l'opérateur < à la classe Durée.

Triez votre vecteur de durées avec l'algorithme sort() de base (sans foncteur ni lambda).

A l'aide d'un foncteur, faites en sorte que l'on puisse trier votre vecteur de Durees avec l'algorithme sort() en fonction du nombre de minutes.

Concepts C++11

Mot clef auto et decltype

Reprendre la fonction qui retourne la valeur maximum d'un tableau de n'importe quel type (exercice Templates).

Vérifiez que quel que soit le type de données, vous pouvez récupérer le résultat dans une variable à l'aide du mot clef auto.

Créez une nouvelle variable dont vous déterminerez la type à l'aide de deltype. Cette nouvelle variable aura le type de la valeur retournée par la fonction maximum. Vérifiez que vous pouvez bien substituer cette nouvelle variable à une autre case de votre tableau.

Boucles basées sur un intervalle

Reprenez votre vecteur et votre map de Durées.

Remplacez les parcours par itérateur par des boucles basées sur intervalle.

Lambdas

Reprenez votre classe Durée.

Ré implémentez le tri en remplaçant votre foncteur par une lambda.

Move Semantics

Dans les expressions suivantes, déterminez les rvalues (tout ce qui a un nom qui ressemble à une variable est bien une variable initialisée) :

x = a%b;

y = 28*a;

vector<int> v = getComponent();

appart1 = appart2;

int* p = &a;

*(p+1) = 42;

Dans Appartement, remplacez le shared_pointer sur Canape par un unique_pointer.

Google Mock

Le but est de tester une classe MessageValidator qui a en attribut une instance de InterfaceQueueMessages.

Vous devrez développer la classe MessageValidator, uniquement un stub de la classe InterfaceQueueMessages, la mock class de InterfaceQueueMessages et un test unitaire vérifiant le bon fonctionnement de MessageValidator en utilisant la mock class de InterfaceQueueMessages.

InterfaceQueueMessages a deux méthodes : isQueueEmpty et giveMessage.

isQueueEmpty renvoie un simple booléen : 0 s'il n'y a pas de message en attente, 1 s'il y a un message en attente.

giveMessage retourne une chaîne de caractères : le message à vérifier.

MessageValidator a une méthode qui teste la validité des messages reçus, elle retourne une chaîne de caractères.

Elle retourne une chaîne vide s'il n'y a pas de message à vérifier.

Elle retourne ERROR suivi du message si le message est invalide.

Elle retourne le message si celui ci est valide.

Un message valide est un message dont les trois premiers caractères sont FRA, BEL, LUX ou SUI et qui contient au total entre 4 et 163 caractères.

Option : vérifiez que les caractères à partir du 4e sont alphanumériques.