Guide d'Introduction au C++

Comparaison C et C++:

Le langage C et le langage C++ partagent de nombreuses similitudes, mais ils présentent également plusieurs différences importantes. Voici les principales différences entre C et C++ :

C	C++
C a été développé par Dennis Ritchie entre 1969 et 1973 chez AT & T Bell Labs.	C++ a été développé par Bjarne Stroustrup en 1979 avec le prédécesseur de C++.
Comparé à C++, C est un sous-ensemble de C++.	C++ est un sur-ensemble de C. C++ peut exécuter la plupart du code C, alors que C ne peut pas exécuter de code C ++.
C ne supporte pas la programmation orientée objet; par conséquent, il ne prend pas en charge le polymorphisme, l'encapsulation et l'héritage.	En tant que langage de programmation orienté objet, C++ prend en charge le polymorphisme, l'encapsulation et l'héritage.
En C (parce que c'est un langage de programmation procédural), les données et les fonctions sont des entités séparées et libres.	En C++ (lorsqu'il est utilisé comme langage de programmation orienté objet), les données et les fonctions sont encapsulées ensemble sous la forme d'un objet. Pour la création d'objets, la classe fournit un plan de structure de l'objet.
En C, les données sont des entités libres et peuvent être manipulées par un code extérieur. En effet, C ne prend pas en charge l'encapsulation d'informations.	En C++, Encapsulation masque les données pour garantir que les structures de données et les opérateurs sont utilisés comme prévu.
C étant un langage de programmation procédurale, c'est un langage basé sur les fonctions.	Alors que C++, étant un langage de programmation orientée objet, c'est un langage orienté objet.
C ne prend pas en charge la surcharge des fonctions et des opérateurs.	C++ supporte à la fois surcharge des fonctions et d'opérateurs.
C ne permet pas de définir des fonctions dans des structures.	En C++, les fonctions peuvent être utilisées dans une structure.
C n'a pas de fonctionnalité de NAMESPACE	C++ utilise NAMESPACE pour éviter les conflits de noms de variables/fonctions/classes…
C utilise des fonctions pour les entrées/sorties. Par exemple, scanf et printf.	C++ utilise des objets pour la sortie. Par exemple cin et cout.
C ne supporte pas les variables de référence.	C++ supporte les variables de référence.
C n'a pas de support pour les fonctions virtuelles et friend.	C++ prend en charge les fonctions virtuelles et friend.
C fournit les fonctions malloc() et calloc() pour l'allocation dynamique de la mémoire et free() pour la dés-allocation de mémoire.	C++ fournit un nouvel opérateur pour l'allocation de mémoire et l'opérateur de suppression pour la dés-allocation de mémoire.
C ne fournit pas de support direct pour le traitement des erreurs (également appelé traitement des exceptions)	C++ fournit un support pour la gestion des exceptions. Les exceptions sont utilisées pour des erreurs « difficiles » qui rendent le code incorrect.
Extension du fichier C est .C	Extension du fichier C++ est .CPP

Bibliothèques :

Les bibliothèques sont des ensembles de fonctions et de classes prédéfinies qui facilitent le développement en C++. Voici quelques bibliothèques couramment utilisées :

- **iostream** : Pour les entrées/sorties standard (cin et cout) `#include <iostream>`
- **vector** : Pour utiliser des tableaux dynamiques `#include <vector>`
- **array** : Pour utiliser la classe std::array qui offre des tableaux de taille fixe `#include <array>`
- **cmath** : Pour utiliser les fonctions mathématiques `#include <cmath>`
- **string** : pour créer, manipuler et gérer des chaînes de caractères de manière plus pratique `#include <string>`

Structure du Code :

La structure de base d'un programme en C++ ressemble à ceci :

```cpp
// Inclusion des bibliothèques
#include <iostream>

// Espace de noms (namespace)
using namespace std;

// Fonction principale (main)
int main() {
    // Code principal du programme

    return 0; // Terminateur de programme
}
```

Dans la structure de base :

- Les bibliothèques sont incluses en utilisant `#include`
- L'espace de noms `using namespace std;`permet d'utiliser les éléments de la bibliothèque standard (comme cin et cout) sans le préfixe std::
- Les fonctions sont déclarées avant la fonction `main()` si elles sont utilisées dans main, et elles sont définies en dehors de main()

Entrées/Sorties :

En C++, vous pouvez utiliser **cin** pour les entrées (clavier) et **cout** pour les sorties (écran). Voici comment les utiliser :

```cpp
int age;
cout << "Entrez votre âge : ";
cin >> age;
cout << "Vous avez " << age << " ans." << endl;
```

Variables et Types de Données :

En C++, vous pouvez déclarer des variables pour stocker des données. Voici quelques types de données couramment utilisés :

```cpp
#include <iostream> //Inclut la bibliothèque iostream (affichage de texte)
#include <string>
using namespace std; //Indique quel espace de noms on va utiliser

int main()
{
	//-------------------- declaration et affichage des variables ----------------

		string nomUtilisateur = "Albert Einstein";// ou string
nomUtilisateur("Albert Einstein");
		bool estUnPhysician = true; // ou bool estUnPhysician(true);
		int mort = 1955; // int mort(1955);
		cout << "Hello user!" << endl; //Affiche un message
		cout << "user name: " << nomUtilisateur << endl << "est Un Physician?" <<
		estUnPhysician << endl;//affiche une variable
		cout << "date de mort :" << mort << endl;
}
```

Structures Conditionnelles :

Les structures conditionnelles permettent d'exécuter différents blocs de code en fonction de certaines conditions.

- **if-else** : Utilisé pour exécuter un bloc de code si une condition est vraie et un autre bloc si la condition est fausse.

```cpp
if (condition) {
	// Code à exécuter si la condition est vraie
}
else {
	// Code à exécuter si la condition est fausse

}
```

- **switch** : Utilisé pour évaluer une expression et exécuter différents blocs de code en fonction de la valeur de l'expression.

```cpp
switch (valeur) {
case 1:
  // Code si la valeur est 1
  break;
case 2:
case 3:
  // Code si la valeur est 2 ou 3
  break;
default:
  // Code par défaut si aucune des valeurs précédentes n'est trouvée
}
```

Boucles :

- **for** : Utilisé pour exécuter un bloc de code un certain nombre de fois.

```cpp
for (int i = 0; i < 5; i++) {
	// Code à répéter
}
```

- **while** : Utilisé pour exécuter un bloc de code tant qu'une condition est vraie.

```cpp
while (condition) {
	// Code à répéter tant que la condition est vraie
}
```

- **do-while** : Utilisé pour exécuter un bloc de code au moins une fois, puis tant qu'une condition est vraie.

```
do {
	// Code à répéter au moins une fois, puis tant que la condition est vraie
} while (condition);
```

Tableaux :

Les tableaux sont des collections de données du même type. En C++, vous pouvez utiliser des tableaux de différentes manières, y compris avec la classe **std::array**.

Déclaration d'un tableau :

```
int tableau[5]; // Déclare un tableau d'entiers de taille 5
```

Accès aux éléments du tableau :

```
tableau[0] = 10; // Affecte la valeur 10 au premier élément
int valeur = tableau[2]; // Accède à la valeur du troisième élément
```

Utilisation de la classe `std::array` :

```
#include <array>
using namespace std;
array<int, 5> tableau; // Déclare un tableau d'entiers de taille 5
```

Fonctions :

Les fonctions permettent de regrouper des blocs de code en vue de les réutiliser et de rendre le code plus lisible.

Déclaration / Prototype d'une fonction :

```
int addition(int a, int b);
```

Définition d'une fonction :

```
int addition(int a, int b) {
	return a + b;
}
```

Appel d'une fonction :

```
int resultat = addition(5, 3); // Appelle la fonction addition avec les arguments 5 et 3
```

Fonction principale main() :

Tous les programmes C++ doivent avoir une fonction main(), qui est le point d'entrée du programme.

```
int main() {
	// Code principal du programme
	return 0; // Terminateurs de programme
}
```