

faire plusieurs chapitres sur l'internationalisation. Séparer les chaînes, la localisation, le formatage. Peut être faire un chapitre sur ICU et sur Qt::tr ?

Les chaînes de caractères internationales

De nos jours, avec les progrès des moyens de communications, en particulier d'internet, il est très facile de partager ses programmes à l'international. Des outils de partage du code, comme par exemple GitHub que vous verrez par la suite, permettent de mettre en place des équipes de développement collaboratif, même sur des projets lancés par une seule personne. Il est donc conseillé, dès le début d'un projet, de penser en termes de travail en équipe et d'écrire ses programmes en anglais (code, commentaires, documentation).

Le corollaire à cela est qu'un programme pourra facilement être utilisé par des personnes qui ne parlent pas l'anglais (en premier lieu vous peut être) et il sera intéressant de pouvoir afficher des chaînes de caractères dans d'autres langages.

Et c'est à ce niveau que la gestion des chaînes de caractères peut devenir complexe. Par défaut, les chaînes en C++ sont basé sur le système anglais, qui ne contient aucun caractère accentué. En français par exemple, nous utilisons des accents aigu, grave ou circonflexe, des trémas. D'autres langues utilisent plus de types d'accents (Suédois). D'autres encore utilisent des alphabètes complètement différents (Russe, Chinois).

En plus des alphabets différents, il faut également gérer d'autres problématiques. Par exemple, en anglais, le point est utilisé comme séparateur décimal. En français, nous utilisons la virgule. En anglais américain, une date s'écrit sous la forme mois-jour-année. En français, nous écrivons jour-mois-année.

Certaines langages s'écrivent de gauche à droite, d'autres de droite à gauche, d'autres encore de haut en bas.

On voit bien, par ces quelques exemples, que la gestion des langages est quelque chose de complexe. Dans ce chapitre, nous allons commencer par étudier les principaux types d'encodage des caractères et les chaînes utilisables dans ce contexte en C++.

Pour une gestion complète des langues ayant des alphabets complexes, il sera préférable d'utiliser des bibliothèques dédiées, comme [ICU](#). Ce chapitre est une simple introduction aux problématiques posées lorsque l'on souhaite prendre en compte l'internationalisation des programmes.

Les norme ASCII

ASCII et ASCII étendu.

http://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Exchange

http://fr.wikipedia.org/wiki/ASCII_%C3%A9tendu

ASCII : 7bit codé dans 8bit, ne représente pas tous les caractères possible. Wide string = caractère non 8 bits. Plusieurs format

Attention : wide string peut être vu comme une suite de caractère 8 bits et l'afficher tel quel (et donc ne pas afficher correctement les caractères

Autres normes :

- Universal Character Set (UCS) par International Standard ISO 10646
- Unicode

wstring, u32string, u12string

caractère de 16 ou 32 bits (wchar_t), préfixe L'a' et L"bla bla"

également wcout, wcerr, wclog, wofstream, etc

- char : 8 bits, UTF-8
- wchar : implémentation spécifique
- char16_t : 16 bits, UTF-16
- char32_t : 32 bits, UTF-32

Différencier affichage et données en mémoire

Localisation

Phrase "Une main a 5 doigts". Mettre valeur en paramètre : "Une main a %1 doigts" (ou "\$1" ou autre). Problèmes :

- doigts au pluriel ou non selon n
- ordre différents selon la langage

Utilisation regex pour faire le remplacement

<http://en.cppreference.com/w/cpp/locale>

Chapitre précédent	Sommaire principal	Chapitre suivant
---------------------------	---------------------------	-------------------------