Aix-Marseille Université
Master 2 ISIC

Introduction au C++ - Critères sur les codes

A) Préambule
Ce document liste les mauvaises pratiques qui serviront à évaluer les programmes écrits lors des TP. En priorité, le code doit compiler et s'éxecuter pour fournir les resultats attendus. Tous ces critères correspondent aux normes et bonnes pratiques de développement C++ pour la production de codes fonctionnels et réduisants le risque d'erreurs.
Ces critères vont être classés en 4 niveaux, avec un niveau de validé uniquement si toutes les conditions liées à ce niveau le sont.

Niveau 1
- Utiliser les bons types d'accès pour les fonctions/données membres d'une classe : public, private ou protected.
- Utiliser de variables locales plutot que des variables globales.
- Initialiser toutes les variables lors des déclarations.
- Utiliser les équivalent C++ de fonctions C *"classiques"*.

Niveau 2
- Utiliser le pointeur *this* à chaque fois que cela est possible
- Faire une allocation dynamique plutôt que déclarer des tableaux statiques de taille disproportionnée.
- Fermer un fichier en fin de fonction.
- Commenter le code.
- Libérer la mémoire (via **delete** ou **delete[]**) pour un bloc mémoire alloué via **new**, **new []**.

Niveau 3
- utiliser la STL (en particulier les vector) pour les tableaux.
- Tester les valeurs de retour des fonctions.
- S'assurer de la "const correctness".
- Ouverture d'un fichier en vérifiant avec **is_open** que la fonction s'est déroulée avec succès.

Niveau 4
- Faire de tests sur les fonctions globales/les fonctions membres pour s'assurer des résultats fournis.
- Faire une allocation mémoire avec gestion des exceptions.

Rappel : commentaire d'une fonction

```
/**
 * definition :  type_retour nom_fonction(type_1 const param1, type_2 param2);
 * description :  But de  fonction
 *
 * param  : param1: variable de type type_1 qui va servir à...,
                    non modifié par la fonction
 * param  : param2: variable de type type_2 qui va servir à...,
                   modifié par la fonction pour stocker tel résultat
```

```
* valeur renvoyée : variable de type, type_retour, qui  contiendra telle valeur.
                    Il peut s'agir d'un code erreur, du résultat d'un calcul...
*/
```