Tips and tricks de C++

Noms de variables

- Règles :
  - Pas de caractères spéciaux : !, « , /, $, %, ?, &, *, (,), etc.
  - Pas d'accents : é, è, à, ò, etc.
  - Seul exception : _nom_variable
  - Exception à l'exception :
    - __nom_variable (réservé à la librairie standard)
    - _Nom_Classe (résevé à la librairie standard)
    - Exemple

Passage par référence vs par copie

- Voir exemple : test_function_ref_vs_copy.cpp

Utilisation des opérateurs

- Utiliser l’opérateur << au lieu d’une fonction .print()
  - Permet de passer votre objet à tout les flux (cout, cerr, ifstream, socket…)
- Utiliser == au lieu d’une fonction .compare()
  - Permet d’accéder aux membres privées de la classe sans appel à un .get()

Mauvaises pratiques

- Utiliser des variables globales
- Utiliser « using namsepace std » dans les fichiers d'entête
- Utiliser « friend » en dehors des opérateurs
- Utiliser des macros du préprocesseur
- Utiliser des fonctions qui dépendent de la plateforme. Ex: _getcwd
- Implémenter les fonctions dans le .h, toujours les implémenter dans le .cpp

Variable de retour booléenne

- Voir boolean_return_variable.cpp

Utiliser seulement le minimum requis

- Mauvais
  - using namespace std;

- Good
  - using std::string;
  - using std::cout;
  - using std::map;

Simplification booléenne

- Voir fichier boolean_simplification.cpp

Noms descriptifs et explicatifs

- Mauvais
  - bool cEstLui(std::string valeur);
- Bon
  - bool estLeBonElement(const std::string& valeur);

Construction des membres via la liste d’initialisation

- Mauvais
  - Object::Object(const std::string& arg)
  - {
  - member = arg;
  - }
- Bon
  - Object::Object(const std::string& arg) : member(arg) {}

Voir consturctor_init_list.cpp

Range-based loop

- Voir range_based_for_loop.cpp

Utilisation des parenthèses

- NON
  - return (valeur);
- OUI
  - return valeur;

Comprendre les switch

- NON
  - case 1:{
  - 	return (valeur);
  - 	break;
  - }
- OUI
  - case 1:
  - 	return valeur;

Assignation lors de la création

- Mauvais
  - std::string valeur
  - valeur = "quelque chose"
- Bien
  - std::string valeur = "quelque chose"

Déférencement

- Mauvais
  - (*this).fonction()
  - (*pointer).fonction()
- Bien
  - this->fonction()
  - pointer->fonction()

Parenthèses inutiles

- Mauvais
  - if((nombre < 1) || (nombre > 15))
- Bien
  - if(nombre < 1 || nombre > 15)

Déclaration de variable de retour

- Voir return_value_declaration.cpp

Utiliser le constructeur par défaut

- Mauvais
- object::object()
- {
- 	string1 = "";
- 	string2 = "";
- }
- Bien
- object::object() {}

http://en.cppreference.com/w/cpp/string/basic_string/basic_string

Utiliser les fonctions C++

- Mauvais
  - int number = atoi(string.c_str());
- Bien
  - int number = stoi(string);

Retourner la même valeur lors d’un if

- Voir return_same_value_if.cpp

Éviter les copies inutiles

- Mauvais
  - string array[] = {"truc1","truc2","truc3"};
  - vector<string> string_vector(array, array + 3);
- Bien
  - vector<string> string_vector = {"truc1", "truc2", "truc3"};

Bien faire les tableaux statiques

- Mauvais
  - char ligne[500];
- Bien
  - std::string ligne;
  - std::array<char, 500> ligne;

Nettoyer ses déchets

- Lorsque vous avez un std::vector<Foo*> container
- object::~object()
  - {
  - for(int i = 0; i < container.size(); ++i)
  - delete container[i];
  - }