POO: Memo C++ *

1-Types de données et variables

Types de base: `int`, `float`, `double`, `char`, `bool`, `void`.
Déclaration et initialisation de variables:

```
int entier = 10;
float flottant = 3.14f;
char caractere = 'A';
bool booleen = true;
```

Type énuméré (classe):

```
enum class Couleur { ROUGE, VERT, BLEU };
....
Couleur c = Couleur::Bleu;
```

2-Opérateurs

arithmétiques (`+`, `-`, `*`, `/`, `%`), relationnels (`==`, `!=`, `>`, `<`, `>=`, `<=`), logiques: `&&`, `||`, `!`, incrémentation/décrémentation: (`++`, `--`), allocation mémoire: (`new`, `delete`).

3-Structures de contrôle

Conditionnelle (si alors): `if`, `else if`, `else`

```
if (x > 0) {
    std::cout << "Positif" << '\n'; }
} else if (x < 0) {
  std::cout << "Negatif" <<  '\n';
} else { std::cout << "Zero" <<  '\n'; }
```

Conditionnelle (cas):

```
switch(couleur)
{
 case Couleur::ROUGE:
 { std::cout << "rouge" << '\n'; break; }
 case Couleur::VERT:
 { std::cout << "vert" << '\n'; break; }
 ...
 default:
 { std::cout << "autre" << '\n'; break; }
}
```

Itérative (boucle pour): `for`

```
for (int i = 0; i < 10; i++) {
  std::cout << i << '\n';
}
```

Itérative (boucle tant que): `while`

```
int i = 0;
while(i<10) {
   std::cout << i << '\n'; i++; }
```

Itérative (boucle tant que): `do while`

```
int i = 0;
do { std::cout << i << '\n'; i++;
} while(i<11);
```

*Auteur: Yannick Pencolé. Ce mémo de C++ ne contient que les éléments syntaxiques de C++ qui seront vus et utilisés dans le cours POO, il n'est absolument pas exhautif.

4-Chaînes de caractères

Classe C++ string (STL):

```
#include<string>

int main() {
 std::string chat = "chat";
 std::string chou = chat;
 chou[2] = 'o';
 chou[3] = 'u';
 std::string chouchou = chou + chou;
 std::string chouchoute(chouchou);
 chouchoute += "te";  }
```

5-Flux entrée sortie

```
#include<iostream>
#include<string>

int main() {
  std::string s1, s2, s3;
  char p;
  int n1, n2;
  std::cin >> s1 >> s2 >> s3 >> n1 >> n2 >> p;
  std::cout << s1 << s2 << s3 << n1 << n2 << p;
}
```

6-Tableaux et Vecteurs

Tableau C, déclaration et accès:

```
int tableau[5] = {1, 2, 3, 4, 5};
tableau[2] = 2*tableau[0] + 3*tableau[1];
// -> { 1, 2, 8, 4, 5 }
```

Tableau C++ (vector), déclaration et accès:

```
#include <vector>

int main() {
 std::vector<int> entiers = { 1 , 2 , 3 , 4 };
 std::vector<int> entiers2;
 entier2 = entiers;
 for(size_t i = 0; i < entiers.size(); ++i) {
    cout << entiers[i] << " "; }
 entiers[0] = 0;
 entiers.push_back(5);
 entiers.pop_front();
 entiers.clear();
 if(entier.empty()) { cout << "vide\n"; } }
```

7-Pointeurs et références

Pointeurs:

```
int x = 10;
int y = 20;
int *ptr = &x;
std::cout << *ptr << '\n'; // Affiche 10
ptr = &y;
std::cout << *ptr << '\n'; // Affiche 20
*ptr = 15;
std::cout << y << '\n'; // Affiche 15
```

Références:

```
int x = 10;
int y = 20;
int &ref = x;
std::cout << ref << '\n'; // Affiche 10
ref = 15;
std::cout << ref << '\n'; // Affiche 15
std::cout << x << '\n'; // Affiche 15
ref = y; // ERREUR
```

Références/Pointeurs constants:

```
int x = 10;
const int &ref = x;
const int *ptr = &x;
*ptr = 15; // ERREUR
ref = 15; // ERREUR
```

8-Fonctions

Déclaration:

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

Définition:

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

Appel:

```
int main() {
  int resultat = addition(2,addition(3,6));
  std::cout << resultat << '\n'; //affiche 11
}
```

Fonction principale:

```
int main() { ... }
```

```
int main(int argc, char ** argv) { ... }
```

9-Classes et objets

Déclaration d'une classe:

```
class Voiture {
 private:
   std::string _marque;
   ...
 public:
 std::string marque();
 void changeMarque(const std::string & m);
   ...
 protected:
   ...
 };
```

Définition de méthodes:

```
std::string Voiture::marque(){return _marque;}
void Voiture::changeMarque(const std::string & m)
     { _marque = m; }
```

Constructeurs/Destructeur:

```
class Voiture {
...
// constructeurs
Voiture(); // defaut
Voiture(const Voiture & voiture); // copie
Voiture(const std::string & marque); // parametre
// destructeur
~Voiture();
};
```

Creation/Accès objet:

```
#include "Voiture.h"
#include <iostream>

int main()
{
  Voiture v1("Renault");
  std::cout << v1.marque() << '\n';
  Voiture v2 = v1;
  v2.changeMarque("Peugeot");
  std::cout << v2.marque() << '\n';
}
```

Pointeur d'autoréférencement: `this`

```
void Voiture::changeMarque(const std::string & m)
     { this->_marque = m; }
```

10-Héritage simple

Déclaration:

```
class R5 : public Voiture { .... };
```

Déclaration méthode virtuelle:

```
virtual void changeMarque(const std::string & m);
```

Déclaration surcharge de méthode virtuelle:

```
virtual void changeMarque(const std::string & m)
    override;
```

11-Allocation dynamique et pointeur

Allocation dynamique d'objets:

```
Voiture * r5 = new Voiture("Renault");
```

Manipulation d'objet par pointeur:

```
std::cout << r5->marque() << '\n';
r5->changeMarque("Alpine");
std::cout << *r5.marque() << '\n';
```

Destruction dynamique d'objets:

```
delete r5; r5 = nullptr;
```

12-Espaces de nommage

Bibliothèque standard STL: `std`

`std::cout`, `std::cin`, `std::vector<T>`,...

Utilisation tacite d'un espace de nommage:

```
using namespace std;
```

Création d'un espace de nommage:

```
namespace MonEspace {
 // ajout de classes, fonctions, variables ....
...
};
```

13-Macros d'unicité d'inclusion (entête)

Fichier entête: `mon_fichier.h`

```
#ifndef MON_FICHIER_H
#define MON_FICHIER_H
// debut de fichier

...

//fin de fichier
#endif
```