M1 IM-MF/MPA **POO**
Université Côte d'Azur **2023-24**

Evaluation 1 C++

Consignes:

- Pas de document autorisé en dehors des notes fournies.
- Les réponses au QCM doivent être écrites directement sur le sujet du QCM qu'il ne faut pas oublier de rendre!

Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre. Le barême est indicatif et non définitif.

Durée: 2h15 (3h si tiers temps)

---

1 QCM [6 pts]

Voir la partie dédiée. Vous devez obligatoirement répondre sur le sujet du QCM!

2 Questions de cours [6 pts]

- En python, on exécute directement le code à l'aide d'un environnement de développement (*spyder*, *jupyter*, *ipython*, ...), expliquer en quoi C++ est différent sur cet aspect-là.
- Citer 3 structures de contrôle de C++, les décrire brièvement. Donner 1 exemple d'utilisation
- Comment crée-t-on une variable en C++? Comparer à python, donner deux types d'entiers, deux types de réels et un autre type courant de C++
- Expliquer la surcharge de fonction en C++. Donner un exemple.
- Qu'est-ce un pointeur? Citer une utilisation concrète d'un pointeur. Indiquer une commande, utilisée de nombreuses fois dans le cours, liée à la manipulation des pointeurs et expliquer son intérêt.
- Décrire, de manière assez détaillée, par quoi est composée une classe en C++. Indiquer les étapes nécessaires pour utiliser des objets d'une classe en C++. Indiquer la différence entre classe et structure.

3 Code à compléter/commenter [4 pts]

On considère la classe *Animal* modélisant, de façon simpliste, un animal par son nom et son poids.

On considère que les attributs de la classe sont privés, on aura une *string* pour le nom et un *double* pour le poids.

La classe possède deux constructeurs publics: l'un sans argument, l'autre à deux arguments de type *double* et *string* afin de définir les deux attributs.

La classe possède les méthodes publiques (getters) *get_poids* et *get_nom* renvoyant respectivement le poids et le nom d'un *Animal*. Ces méthodes doivent s'appliquer à des objets de la classe pouvant être constants.

Enfin, on considère la fonction membre publique *affiche*, sans argument, ne renvoyant rien, produisant un affichage à l'écran d'un objet de la classe et pouvant s'appliquer à des objets constants de la classe.

Les déclarations de la classe *Animal* sont faites dans le fichier *Animal.hh*.

3.1 Code à compléter

INDICATIONS: Dans ce qui suit, vous devez répondre aux questions sur votre copie et non pas sur l'énoncé. Veuillez indiquer explicitement le numéro de ligne dans vos réponses
Il y a 20 ajouts de code à faire en tout dans cet exercice.

- En utilisant les explications précédentes, compléter le code du fichier *Animal.hh*, donné dans le Listing 1.

On définit ensuite dans le fichier *Animal.cpp* les méthodes de la classe *Animal*.

- A partir des explications précédentes, compléter les définitions dans *Animal.cpp*, pour la classe *Animal*, données dans le Listing 2.

En plus de ces méthodes, on définit trois autres fonctions dans le fichier *Animal.cpp* ainsi que la fonction *main*.

- La fonction *poidsTotal* renvoie le poids total de n Animaux, défini dans le tableau t et n'est pas une fonction membre d'*Animal*.
- La fonction *affiche* (attention, celle-ci n'est pas la fonction membre de la classe *Animal*) produit un affichage de n objets de la classe *Animal* contenus dans le tableau t et fait appel à la fonction membre *affiche* de la classe *Animal*.
- La fonction *getMaxPoids* renvoie un entier correspondant à l'indice dans le tableau t de l'objet *Animal* ayant le poids le plus élevé.

- A partir des explications précédentes, compléter le code du fichier *Animal.cpp* pour ces trois fonctions, données dans la suite du Listing 2.

Listing 1: Animal.hh

```
1  using namespace std;
2
3  // declaration de la classe Animal
4  class Animal{
5          :
6    string _nom; // nom de l'Animal (prive)
7           _poids; // poids de l'Animal (prive)
8        :
9    Animal(); // constructeur sans argument
10                                  ; // constructeur a 2 arguments
11                                ; // fonction d'affichage
12                                  ; // getter du poids
13   string get_nom() const; // getter du nom
14 }
```

Listing 2: Animal.cpp

```cpp
#include<iostream>

using namespace std;

// fonction membre pour acceder au nom de l'Animal
string Animal::get_nom() const
{
                ;
}

// fonction membre pour acceder au poids de l'Animal

{
  return _poids;
}

// fonction membre affiche de la classe Animal

{
  cout << "Animal " << _nom << ", de "
       <<_poids << " kgs" << endl;
}

// constructeur a deux arguments
Animal::Animal(string c, double p) : _nom( ),
                                     _poids( ) {}

// constructeur sans argument
Animal::Animal() : _nom(""), _poids(0) {}

// fonction poidsTotal
double poidsTotal(Animal* t, int n)
{
        p=0;
  for(int i=0;i<n;i++)
      t[i].get_poids();
  return p;
}
```

```
44 // fonction affiche (pour plusieurs Animaux)
45 void affiche(Animal* t, int n)
46 {
47   for(int i=0;i<  ;i++)
48     .affiche();
49 }
50
51 // fonction getMaxPoids - indice de l'Animal le plus lourd
52 int getMaxPoids(Animal* t, int n)
53 {
54   p = t[0].get_poids();
55   int j = 0;
56   for(int i=1;i<n;i++)
57     {
58       if(t[i].get_poids()>  )
59         j = i;
60     }
61   return  ;
62 }
63
64 // main
65 int main()
66 {
67   Animal troupeau[3] = {Animal("vache", 1000),
68     Animal("mouton", 400), Animal("chevre", 60)};
69   affiche(troupeau, 3);
70   cout << "Poids Total: " <<
71     poidsTotal(troupeau, 3) << endl;
72   int j = getMaxPoids(troupeau, 3);
73   cout << "L'animal de poids le plus eleve est "
74        << troupeau[j].get_nom() << endl;
75   return 0;
76 }
```

3.2 Questions sur le code

Répondre aux questions suivantes sur le code des fichiers *Animal.hh* et *Animal.cpp*

- Pourquoi a-t-on défini les fonctions membres *get_poids* et *get_nom* dans la classe *Animal*?
- Dans la fonction *main*, qu'est-ce que la variable *troupeau*? Combien a-t-elle d'éléments et de quel type?
- Quel est le résultat attendu pour *PoidsTotal* (l.70 et 71 dans la fonction *main*)?
- Quel est l'affichage attendu dans le *main* pour les lignes 69 et 73-74?

4 Codes à écrire [4 pts]

INDICATION: une tolérance sur la syntaxe sera adoptée dans la correction, essayez cependant d'être le plus proche possible de la syntaxe habituelle de C++.

4.1 Sommes des éléments d'un tableau

Soit t un vecteur de $\mathbb{R}^n$. A l'aide d'une fonction C++ *theSum*, on voudrait calculer la quantité s suivante, selon un paramètre booléen *estPair*, prenant la valeur *true* or *false*.

Si *estPair* est *true*, alors la fonction retourne la somme des éléments d'indice pair du tableau t; sinon, la fonction retourne la somme des éléments d'indice impair du tableau. Autrement dit:

$$\text{Si} \quad estPair \text{ est } true, \quad s = \sum_{\substack{0 \le i < n, \\ i \text{ pair}}} t_i,$$

$$\text{Si} \quad estPair \text{ est } false, \quad s = \sum_{\substack{0 \le i < n, \\ i \text{ impair}}} t_i$$

- Ecrire le code de la fonction *theSum* prenant trois arguments: le 1er argument représentera le tableau t (on considèrera des valeurs réelles), le 2ème est la taille n du tableau et le 3ème argument est le paramètre booléen *estPair*
- Ecrire le test de la fonction *theSum* dans la fonction *main* sur un tableau de *double* contenant les éléments de votre choix (avec au moins 3 éléments). Faire afficher le résultat de l'appel à *theSum* en considérant d'abord *estPair=true* puis *estPair=false*.

4.2 Valeur maximale d'un tableau

- Ecrire le code de la fonction *myMax* prenant deux arguments: un pointeur sur entier (représentant un tableau d'entiers) et un entier (pour la taille du tableau). Cette fonction doit renvoyer un entier égal à la valeur maximale contenue dans le tableau.
- On souhaiterait écrire une autre version de la fonction *myMax* prenant deux arguments: un pointeur sur *double* (représentant un tableau de *double*) et un entier (pour la taille du tableau). Cette fonction doit renvoyer un *double* égal à la valeur maximale du tableau. Indiquer les différences entre cette nouvelle fonction et la précédente.
- Ecrire le code de la fonction *affiche* prenant deux arguments: un pointeur sur entier (représentant un tableau d'entiers) et un entier (pour la taille du tableau). Cette fonction doit produire un affichage à l'écran du tableau.
- On souhaiterait utiliser la fonction *affiche* sur un tableau de *double*. Est-ce possible? Sinon indiquer comment faire.
- Ecrire le code d'une fonction *main* qui teste ces fonctions de la manière suivante:
  - définir un tableau d'entiers avec les valeurs de votre choix, avec au moins 2 éléments
  - afficher ce tableau et afficher la valeur maximale de ce tableau calculée en appelant *myMax*
  - Mêmes questions avec un tableau de *double* avec les valeurs de votre choix, de taille au moins 2