Examen écrit ZZ3 – Exemple – 1h

Première Partie : C++

Si l'on définit le type suivant `std::map<int,string,less>`
Le dernier paramètre peut être un foncteur.

1. Qu'est-ce qu'un foncteur ?

2. Le foncteur peut-il avoir un état ? Si oui, donner un exemple

3. Quel concept issu du CPP2011 est plus utilisé que les foncteurs même si, sous le « capot », cela génère le plus souvent un foncteur ?

4. Donner un exemple

5. Comment cet objet interagit avec son environnement ?

6. Voici un extrait d'un des TPs de l'année, précisez le type d'instruction :

```
int main(void) {

   C c1(1,2);              ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   C c2(3,4);              ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   C c3(c1);               ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   C c4(std::move(c2));    ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   c3=c1;                  ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   c4=C(5,6);              ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement
   C c5 = c1 ;             ☐ Construction ☐ Affectation ☐ Copie ☐ Mouvement

   return 0;
}
```

7. Dans quelles conditions est-il intéressant d'écrire les opérateurs de mouvement d'une classe ?

8. Quelle est l'erreur mise en avant dans ce code ? Comment la résout-on ?

```
Personne.cpp: In member function 'virtual void
Personne::afficher(std::ostream&) const':
Personnne.cpp:23:16: error: passing 'const Personne' as 'this'
argument discards qualifiers [-fpermissive]
   23 |     o << getNom() << " "  << getId();
      |                ^
In file included from Personne.cpp:3:
Personne.hpp:46:16: note:   in call to 'std::string
Personne::getNom()'
   46 |     std::string getNom() { return nom; }
      |                 ^~~~~~
9. make: *** [build/personne.o] Error
```

10. Vous manipulez une instance de classe Fille en passant par un pointeur de type Mere *. Comment convertissez-vous de manière sécurisée le pointeur en Fille * ?

11. Pourquoi un template CPP ne doit-il pas être défini dans un fichier de code .cpp ?

12. Comment définit-on une classe abstraite en CPP ?

13. Qu'est-ce que l'inlining ?

14. Est-ce que la virtualité et l'inlining vont bien ensemble ? Expliquer ?

15. Un constructeur peut-il être virtuel ?

Partie 2 : Modélisation / Concepts / Patrons de conception

1. Qu'est-ce qu'une interface ? En quoi est-ce différent d'une classe abstraite ?

2. La pile, la file et la pile à priorité dans la bibliothèque standard sont des conteneurs adaptés ? Qu'est-ce que cela veut dire ?

3. Qu'est-ce que le polymorphisme ? Donner les deux types de polymorphismes (nom en français et en anglais, principe et illustration) .

4. Qu’est-ce qu’un patron de conception ?

5. Un des grands principes des patrons de conception est d’ « encapsuler ce qui varie » . Donner la définition de l’encapsulation

6. Donner un exemple d’encapsulation de ce qui varie :

7. Qu’est-ce que le principe de « responsabilité unique » ?

8. Lors de la modélisation, on peut utiliser de l'héritage multiple. Donner deux manières de s'en passer avec un avantage et et un inconvénient pour chacune des manières

9. Dans la patron « observateur », est-ce l'observateur qui possède la liste des observés ou bien l'observé qui connaît ses observateurs ? Pourquoi ?

10. Quel est l'intérêt des exceptions dans la gestion d'erreur ?.

11. En ce qui concerne l' « exception safety » pouvez-vous citer et expliquer deux niveaux sur 5 ?