Correction du partiel de TYLA

EPITA – Apprentis promo 2010
Tous documents (notes de cours, polycopiés, livres) autorisés

Février 2008 (1h00)

Correction: Le sujet et sa correction ont été écrits par Akim Demaille et Roland Levillain.

Attention, dans ces questions il y a toujours une et une seule réponse valable. En particulier, lorsque plusieurs réponses sont possibles, prendre la plus restrictive. Par exemple s'il est demandé si 0 est *nul*, *non nul*, *positif*, ou *négatif*, sélectionner *nul* qui est plus restrictif que *positif* et *négatif*, tous deux vrais.

1. Les multiméthodes permettent

Réponses possibles :

- × aux méthodes de retourner plusieurs résultats.
- → le polymorphisme dynamique sur plusieurs arguments de fonctions.
- × aux classes d'avoir plusieurs méthodes de même nom.
- × différents paradigmes de programmation.

2. Le patron de conception « Visitor » permet l'utilisation

Réponses possibles :

- × d'itérateurs en profondeur d'abord.
- × d'itérateurs en largeur d'abord.
- → des multiméthodes dans un langage objet qui en est démuni.
- × d'accesseurs sur des membres pourtant privés.

3. Les templates de classe du C++

Réponses possibles :

- × sont des collections de templates de fonctions libres.
- → sont des générateurs de classes.
- × sont des classes dont toutes les méthodes sont virtuelles.
- × sont des classes dont toutes les méthodes sont virtuelles pures.

4. Un design pattern est

Réponses possibles :

× un langage de conception universel.

→ une bonne solution à un problème connu.

× une méthode de conception d'application orientée objet.

× un générateur de classes de conception.

5. La liaison dynamique en C++

Réponses possibles :

× fait référence aux bibliothèques dynamiques.

× a rapport avec la surcharge d'opérateurs.

→ a rapport avec « virtual ».

× repose sur « template ».

6. La résolution des appels « virtual » nécessite

Réponses possibles :

× la connaissance du type des contenants.

× la connaissance du type des classes.

× la connaissance du type des opérateurs.

→ la connaissance du type des contenus.

7. La résolution de la surcharge nécessite

Réponses possibles :

→ la connaissance du type des contenants.

× la connaissance du type des classes.

× la connaissance du type des opérateurs.

× la connaissance du type des contenus.

8. Les visiteurs

Réponses possibles :

× permettent de parcourir de façon générique les conteneurs.

× sont des fonctions objets.

→ permettent d'implémenter le « dispatching » une fois pour toute.

× remplacent les accesseurs.

9. Les multiméthodes sont disponibles dans

Réponses possibles :

× C++

→ CLOS

× Tiger

× Haskell

10. Surcharge vs méthodes virtuelles: quelle est la bonne réponse ?

Réponses possibles :

× La surcharge et les méthodes virtuelles sont des mécanismes statiques.

× La surcharge et les méthodes virtuelles sont des mécanismes dynamiques.

→ La surcharge est un mécanisme statique, les méthodes virtuelles un mécanisme dynamique.

× La surcharge est un mécanisme dynamique, les méthodes virtuelles un mécanisme statique.

11. Le support des fonctions récursives nécessite

Réponses possibles :

→ une pile (*stack*).

× un tas (*heap*).

× que le langage propose des pré-déclarations (*forward declarations*).

× la liaison des fonctions dynamiques.

12. Généralement, lorsque l'on définit un constructeur non trivial en C++ dans une classe qui possède des attributs de type pointeurs, on écrit systématiquement aussi

Réponses possibles :

× un accesseur.

× un propulseur.

→ un `operator=`.

× un pretty-printer.

13. En C++, l'encapsulation est réalisée grâce

Réponses possibles :

× aux mots-clef `public`, `private` et `protected`.

× à la séparation du code en fichiers d'interface (`foo.hh`) et d'implémentation (`foo.cc`).

× à `#include`.

→ aux classes.

14. Dans l'absolu, les relations entres les modules d'un logiciel devraient former

Réponses possibles :

- × un graphe quelconque.
- × un graphe planaire.
- × un graphe non orienté.
- → un graphe orienté acyclique.

15. Lequel de ces éléments n'entre pas en compte lors de la résolution d'une méthode surchargée en C++?

Réponses possibles :

- × le nom de la fonction.
- × les arguments de la fonction.
- → le type de retour.
- × le qualificateur `const` de la méthode.

16. En C

Réponses possibles :

- × on ne dispose pas de `malloc` et `free`.
- × on dispose d'un garbage collector.
- → on peut avoir des pointeurs non initialisés.
- × on peut avoir des réferences non initialisées.

17. En C++

Réponses possibles :

- × on ne dispose pas de `malloc` et `free`.
- × on dispose d'un garbage collector.
- → on peut avoir des pointeurs non initialisés.
- × on peut avoir des réferences non initialisées.

18. En Java

Réponses possibles :

- × on dispose de `malloc` et `free`.
- × on ne dispose pas d'un garbage collector.
- × on peut avoir des pointeurs non initialisés.
- → on peut avoir des réferences non initialisées.

19. En C++, on appelle objet-fonction

Réponses possibles :

- × un objet construit à l'intérieur d'une fonction.
- → un objet disposant d'un `operator()`.
- × une méthode.
- × un fichier de code compilé (`foo.o`) ne contenant qu'une seule fonction (ex: `foo ()`).

20. On dit d'un langage qu'il est fonctionnel

Réponses possibles :

- × s'il supporte le concept de fonction, éventuellement récursive.
- → lorsqu'il permet de manipuler des fonctions comme n'importe quel autre entité/objet.
- × lorsqu'il dispose d'un compilateur implémenté et en état de marche.
- × s'il n'effectue aucun effet de bord.

21. Qui est l'auteur du langage C?

Réponses possibles :

- × Brian Kernighan
- → Dennis Ritchie
- × Bjarne Stroustrup
- × Ken Thompson

22. De nos jours, les programmes C++

Réponses possibles :

- × sont interprétés.
- × sont compilés vers un équivalent en C, puis vers le langage d'assemblage de la machine.
- → sont compilés nativement vers le langage d'assemblage de la machine.
- × sont compilés vers un code-octet (*byte-code*).

23. Le type dynamique d'un objet

Réponses possibles :

- → est un sous-type de son type statique.
- × est un sur-type de son type statique.
- × est connu à la compilation .
- × est utilisé pour résoudre les appels de fonctions/méthodes. surchargées

24. Lequel de ces opérateur ne peut être surchargé en C++?

Réponses possibles :

× `,`

× `%`

→ `@`

× `|`

25. BNF

Réponses possibles :

× est un « langage de langages ».

× est une « grammaire de grammaires ».

→ est un « langage de grammaires ».

× est une « grammaire de langages ».

26. Lequel de ces langages n'est pas normalisé ?

Réponses possibles :

× C

× C++

× C#

→ D

27. On dit qu'un langage de programmation dispose d'un « typage statique fort » lorsque

Réponses possibles :

× la liaison des noms est effectuée à la compilation.

→ tous les types sont vérifiés à la compilation.

× tous les types sont vérifiés à l'exécution.

× ce langage est dépourvu du polymorphisme d'inclusion.

28. Parmi les assertions suivantes, laquelle est fausse ?

Réponses possibles :

→ Il est possible d'écrire des macros récursives en C.

× Il est possible d'écrire quelques conteneurs génériques en C grâce aux macros.

× Les constantes numériques définies à l'aide de macros C sont portables.

× Les macros utilisées comme fonctions peuvent produire des effets de bord.

29. Une fonction C++ ne peut être mise en ligne (*inlined*) si

Réponses possibles :

- × elle utilise des variables globales.
- → elle est récursive.
- × elle fait usage de `new`.
- × elle renvoit une valeur (son type de retour est différent de `void`).

30. Laquelle de ces déclarations Tiger est invalide ?

Réponses possibles :

- → `var a : int`
- × `var a := 42`
- × `var a : int := 42`
- × aucune

31. Le programme Tiger `(print_int ((4;2) + (5;1)); print ("\n"))`

Réponses possibles :

- × ne compile pas.
- → affiche `3` à l'exécution.
- × affiche `9` à l'exécution.
- × affiche `9;3` à l'exécution.

32. Quelle structure de données ne peut-on construire en Tiger ?

Réponses possibles :

- × un tableau dynamique de chaînes de caractères.
- → un tableau statique de pointeurs.
- × une liste chaînée.
- × un dictionnaire (*map*).

33. Parmi les propositions suivantes, laquelle est une propriété essentielle de `std::set` (en C++) ?

Réponses possibles :

- × C'est un conteneur à accès aléatoire.
- × La recherche d'un élément a une complexité temporelle en O(1).
- → Un élement y est présent au plus une fois.
- × C'est le seul conteneur de la bibliothèque standard du C++ capable de contenir d'autres conteneurs.

34. Quel polymorphisme est supporté par le langage C?

Réponses possibles :

- × le polymorphisme d'inclusion
- → le polymorphisme de coercition
- × le polymorphisme paramétrique
- × aucun

35. En C++, `std::list` est

Réponses possibles :

- × un type.
- × un paramètre.
- → un identifiant.
- × un concept.

36. Quel est la signature du constructeur par copie d'une classe C++ `Foo` ?

Réponses possibles :

- × `Foo::Foo (Foo&) const`
- × `Foo::Foo (const Foo)`
- → `Foo::Foo (const Foo&)`
- × `Foo::Foo (const Foo&) const`

37. En C++, un paramètre effectif d'une classe (paramétrée, donc) ne peut pas valoir

Réponses possibles :

- → `const`.
- × `unsigned`.
- × une constante entière.
- × un type de classe défini par l'utilisateur.

38. Tiger supporte

Réponses possibles :

- → les fonctions imbriquées.
- × les fonctions d'ordre supérieur (manipulables comme n'importe quelle autre valeur).
- × les fonctions paramétrées.
- × les classes paramétrés.

39. Les concepts du C++ ISO 2003

Réponses possibles :

- × se définissent grâce au mot clef `concept`.
- → sont vérifiés implicitement par le compilateur.
- × sont compilés automatiquement aux sites d'utilisations.
- × sont des types qui supportent la redéfinition.

40. En C++, `int main () {}`

Réponses possibles :

- × ne compile pas.
- × produit une erreur à l'exécution.
- → produit un programme dont le code de retour est 0.
- × produit un programme dont le code de retour est dépendant de l'implémentation du compilateur utilisé.