Correction du partiel de Rattrapage TYLA

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

Septembre 2008 (1h00)

Correction: Le sujet et sa correction ont été écrits par 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. Le sucre syntaxique

Réponses possibles :

- → permet d'ajouter la surcharge de fonctions dans un langage.
- × est indispensable dans certains cas.
- × est une notion qui n'existe que dans les langages à objet.
- × est une notion qui relève purement de la syntaxe : on peut le toujours traiter au stade du parser dans un compilateur.

2. Dans un (des) compilateur(s), la présence d'une représentation intermédiaire du programme

Réponses possibles :

- × est indispensable.
- × est un langage inutile, car non exécutable.
- → permet la factorisation de code dans certains cas.
- × est dépendante du langage cible.

3. Dans un langage de programmation, les types

Réponses possibles :

- × sont indispensables.
- × sont forcément évalués à la compilation.
- × sont forcément évalués à l'exécution.
- → peuvent participer à l'optimisation des performances.

4. Citez une caractéristique essentielle de la programmation orientée objet.

Réponses possibles :

× Le masquage de données (en C++: `public`, `private`, `protected`).

→ L'encapsulation (regroupement des données et des traitements).

× La surcharge d'opérateurs.

× L'absence de fonctions indépendantes (n'appartenant pas à une classe).

5. Qu'est-ce qui distingue fondamentalement une méthode (fonction membre) d'une fonction en C++?

Réponses possibles :

× Le fait que sa déclaration soit située dans celle de sa classe.

× La convention d'appel utilisant l'opérateur "`.`".

→ Son "premier" argument.

× On peut surcharger des méthodes, pas des fonctions.

6. Le type dynamique d'un objet

Réponses possibles :

× est un sur-type de son type statique.

→ intervient lors de la résolution d'une fonction virtuelle.

× intervient lors de la résolution d'une fonction surchargée.

× est connu à la compilation.

7. Les multiméthodes sont disponibles dans

Réponses possibles :

× C++

× C#

→ Common Lisp (CLOS)

× Java

8. Quel *design pattern* permet de garantir une unique instance pour un type donné ?

Réponses possibles :

× Fabrique Abstraite

× Poids Plume

× Prototype

→ Singleton

9. Lequel de ces design patterns n'est pas mis en œuvre dans l'implémentation des Abstract Syntax Trees (ASTs) de TC et de leur traitement ?

Réponses possibles :

× Composite

× Patron de Méthode

→ Singleton

× Visiteur

10. En C

Réponses possibles :

→ on peut avoir des pointeurs non initialisés.

× on peut avoir des références non initialisées.

× on dispose de `new` et `delete`.

× on dispose d'un garbage collector.

11. En C++, `std::list<int>` est

Réponses possibles :

× un type.

× un paramètre.

→ une classe.

× un concept.

12. En C, quel est le type d'un pointeur constant vers un entier mutable ?

Réponses possibles :

× `const int *`

× `int const *`

→ `int * const`

× `const int * const`

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

Réponses possibles :

→ Ses éléments sont triés.

× C'est un conteneur à accès aléatoire.

× Ce conteneur ne dispose pas de `const_iterator`.

× La recherche d'un élément a une complexité temporelle en $O(n^2)$, où n est le nombre d'éléments.

14. La classe C++ `std::list<T>`

Réponses possibles :

→ modèle (incarne) le concept `Container`.

× modèle le concept `Random Access Container`.

× modèle le concept `Associative Container`.

× ne modèle aucun de ces concepts.

15. La classe C++ `std::vector<T>`

Réponses possibles :

× modèle le concept `Container`.

→ modèle le concept `Random Access Container`.

× modèle le concept `Associative Container`.

× ne modèle aucun de ces concepts.

16. La classe C++ `std::set<T>`

Réponses possibles :

× modèle le concept `Container`.

× modèle le concept `Random Access Container`.

→ modèle le concept `Associative Container`.

× ne modèle aucun de ces concepts.

17. La classe C++ `std::map<T>`

Réponses possibles :

× modèle le concept `Container`.

× modèle le concept `Random Access Container`.

→ modèle le concept `Associative Container`.

× ne modèle aucun de ces concepts.

18. Qu'est ce qui rend les `union` s difficiles à utiliser en C et en C++?

Réponses possibles :

× On ne peut pas les placer sur la pile.

× On ne peut y mettre aucune `struct`.

× On ne sait jamais exactement quelle place elles occuperont en mémoire.

→ L'identification du type de données effectivement stocké est à la charge de l'utilisateur.

19. Pour laquelle de ces tâches le préprocesseur du C++ est-il réellement utile ?

Réponses possibles :

→ La compilation séparée.

× La définition de constantes.

× La mise en ligne de définitions de fonctions.

× La création de code générique à l'aide de macros.

20. Le programme Tiger `42 + 51`

Réponses possibles :

× provoque une erreur à la compilation.

× provoque une erreur à l'exécution.

→ renvoie un code de sortie (exit status) valant `0` à l'exécution.

× renvoie un code de sortie (exit status) valant `93` à l'exécution.

21. En Tiger, les tableaux sont construits

Réponses possibles :

× sans être initialisés.

× sur la pile.

→ sur le tas.

× dans le segment de données.

22. Dans TC, un environnement ou table de symboles

Réponses possibles :

× est présent dans la pile Tiger.

→ est relatif à une portée donnée.

× est une structure de données nécessaire à l'exécution d'un programme.

× contient une copie des registres.

23. En Tiger, comment la mémoire alloué dynamiquement est-elle libérée ?

Réponses possibles :

× À l'aide du mot clef `delete`.

× À l'aide de la routine de la bibliothèque standard `dispose`.

→ Cette tâche est dévolue à l'environnement d'exécution du programme.

× Il n'y a jamais d'allocation dynamique en Tiger.

24. Quel type à l'expression `foo := 42` en Tiger ?

Réponses possibles :

× `Bool`

× `Int`

× `Nil`

→ `Void`

25. Dans tous les langages fonctionnels

Réponses possibles :

× l'affectation est proscrite.

× il n'y a pas de méthode (fonctions membre).

× les expressions sont évaluées paresseusement.

→ on peut manipuler une fonction comme n'importe quel autre variable.

26. Quelle fonctionnalité de Java est assez dépendante du mode d'exécution à base de machine virtuelle ?

Réponses possibles :

→ L'introspection.

× Les types génériques.

× L'invocation de méthodes à distance ou Remote Method Invocation (RMI).

× L'absence de fonctions indépendantes (n'appartenant pas à une classe).

27. Qu'est-ce qui distingue les références Java de celles de C++?

Réponses possibles :

× Les références Java sont toujours valides.

→ Les références Java peuvent êtres invalides.

× Les références C++ peuvent être réutilisées pour référencer plusieurs objets au cours de leur durée de vie.

× Les références C++ ne permettent pas de référencer des objets sur la pile.

28. L'inventeur de la technique de parsing LR est

Réponses possibles :

× John Backus.

× Noam Chomsky.

→ Donald Knuth.

× Lars Rasmussen.

29. Quel est le premier "vrai" langage de programmation haut niveau ayant eu du succès ?

Réponses possibles :

× COBOL

→ FORTRAN

× Lisp

× PL/I

30. Quel fameux langage de programmation nécessitait un clavier particulier pour en écrire les symboles ?

Réponses possibles :

× ALGOL

→ APL

× FP

× Simula