

TYLA - Sujet A

Statut	Terminée
Commencé	lundi 23 juin 2025, 10:15
Terminé	lundi 23 juin 2025, 10:44
Durée	29 min 23 s

Question 1
Terminé
Noté sur 0,50
Marquer la question

Les cartes perforées ont été utilisées en informatique jusque dans les années...

- a. 30-40
- b. 50-60
- c. 70-80
- d. 90-2000

Question 2
Terminé
Noté sur 0,50
Marquer la question

Laquelle de ces inventions n'a pas été présentée lors de la *Mother of All Demos* de 1968 ?

- a. La souris
- b. La visioconférence
- c. Le drag-and-drop
- d. L'affiche sur écran en couleur

Question 3
Terminé
Noté sur 0,50
Marquer la question

Quel a été le premier langage avec un compilateur optimisant ?

- a. Algol
- b. Fortran
- c. Forth
- d. Groovy
- e. OCaml

Question 4
Terminé
Noté sur 0,50
Marquer la question

Quel a été le premier langage avec un IDE ?

- a. Algol 68
- b. Java
- c. Javascript
- d. Smalltalk
- e. OCaml

Question 5
Terminé
Noté sur 0,50
Marquer la question

Quel a été le premier langage fonctionnel ?

- a. Haskell
- b. Lisp
- c. Scala
- d. Fortran
- e. OCaml

Question 6
Terminé
Noté sur 0,50
Marquer la question

Quel a été le premier langage avec de la programmation par contrat ?

- a. Eiffel
- b. C++
- c. Ada
- d. D
- e. OCaml

Question 7
Terminé
Noté sur 2,00
Marquer la question

Associez les langages à leur date de publication

1956 Fortran ↕

1958 Algol ↕

1959 Cobol ↕

1972 C ↕

1990 Python ↕

2015 Rust ↕

Question 8
Terminé
Noté sur 0,50
Marquer la question

Les invariants de types...

- a. Sont vérifiés à la compilation uniquement
- b. Sont une forme de contrat
- c. Facilitent la monomorphisation
- d. Empêchent le typage statique

Question 9
Terminé
Noté sur 0,50
Marquer la question

Les templates C++ sont compilés par...

- a. Monomorphisation
- b. Typage dynamique
- c. Boxing
- d. Passage de dictionnaire

Question 10
Terminé
Noté sur 0,50
Marquer la question

En OCaml, les foncteurs permettent...

- a. D'effectuer du dispatch dynamique.
- b. D'effectuer du typage dynamique.
- c. D'améliorer les performances du programme.
- d. D'effectuer des opérations sur des paramètres de type.

Question 11
Terminé
Noté sur 0,50
Marquer la question

En Rust, le borrow-checking ne permet pas de garantir...

- a. Qu'un accès dans un tableau est valide.
- b. Qu'une ressource n'a qu'un seul propriétaire.
- c. Que les accès mémoire sont toujours sûrs.
- d. Que la mémoire n'est jamais désallouée prématurément.

Question 12
Terminé
Noté sur 2,00
Marquer la question

Les points suivants sont-ils des avantages ou désavantages de la généricité par monomorphisation ?

Optimisation des instances Non impacté par la monomorphisation ↕

Performances à la compilation Désavantage ↕

Taille du binaire Désavantage ↕

Qualité des messages d'erreur Avantage ↕

Performances à l'exécution Désavantage ↕

Question 13
Terminé
Noté sur 2,00
Marquer la question

Les types options...

- a. Permettent d'éviter certains bugs
- b. Sont adaptés à la gestion de tous les cas d'erreur
- c. Peuvent alourdir le flot de contrôle du programme
- d. Sont un substitut viable au concept de "pointeur null"
- e. Rend la gestion des cas d'erreurs implicite
- f. Garantissent à la compilation la gestion des valeurs nulles

Question 14
Terminé
Noté sur 4,00
Marquer la question

Pour cet exercice, on se place dans le contexte du pseudo-code suivant :

```
var t : integer
    foo : array [0..1] of integer;

procedure shoot_my(x : Mode integer);
begin
    t := 1;
    foo[t] := 2;
    x := x * 2;
end;

begin
    t := 0;
    foo[0] := 3;
    foo[1] := 0;
    shoot_my(foo[t]);
end;
```

Remplir les valeurs des variables **à la fin de l'exécution du programme** en fonction du mode de passage utilisé.

Avec un mode de passage par **valeur** :

- foo[0] = , foo[1] = , t =

Avec un mode de passage par **valeur-résultat**, à la Algol W (la l-value dans laquelle est copiée la valeur résultat est évaluée *au retour de la fonction*) :

- foo[0] = , foo[1] = , t =

Avec un mode de passage par **valeur-résultat**, à la Ada (la l-value dans laquelle est copiée la valeur résultat est évaluée *à l'appel de la fonction*) :

- foo[0] = , foo[1] = , t =

Avec un mode de passage par **référence** :

- foo[0] = , foo[1] = , t =

Avec un mode de passage par **nom** :

- foo[0] = , foo[1] = , t =

Question 15
Terminé
Noté sur 5,00
Marquer la question

On cherche à implémenter un parser de requêtes HTTP en Python ou en C. Présentez les deux langages, et comparez leur utilisation dans le cadre de l'implémentation de ce logiciel.

Python est un langage de script multi-tâches et haut niveau. C est des langages les plus bas-niveau parmi les langages haut-niveau. Il donne notamment au développeur la tâche de gérer lui-même la mémoire. Dans le cadre de l'implémentation de ce logiciel il serait probablement plus facile de le faire avec Python. En effet un parser de requêtes HTTP va doublement demander de la gestion mémoire en C alors que cela ne sera invisible pour le développeur qui le développera en Python. D'autres tâches comme la gestion des chaînes de caractères sont plus faciles en Python. Cependant cela peut se faire au détriment de la performance du programme.