STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 1

Décrire les opérations avant leur stockage

Comparez type abstrait, structure concrète et contrat. Construisez un invariant pour conserver un minimum pendant un parcours.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Distinguer contrat abstrait et structure concrète.
- Le contrat précise le résultat, les erreurs et les mutations ; une représentation réalise les opérations de ce contrat.
- Un invariant doit être établi avant le premier tour, puis préservé à chaque étape.

Prérequis. Avoir suivi les fondations du cours d'algorithmique, y compris les indices et la récursivité.

Partir d'un besoin

Un service doit retrouver le plus petit score reçu. Faut-il conserver tous les scores ? Cela dépend des opérations futures : demander seulement le minimum n'exige pas la même mémoire que restituer chaque score.

Définir la structure et les conventions

Un **type abstrait de données** décrit les valeurs et les opérations disponibles, sans imposer leur stockage. Une pile propose par exemple de déposer puis de retirer le dernier élément. Une **structure de données** est une organisation concrète qui réalise ces opérations. Plusieurs représentations peuvent satisfaire le même contrat.

Un **contrat** précise les entrées, les résultats, les erreurs et les modifications autorisées. Une **précondition** doit être vraie avant l'opération ; une **postcondition** doit être vraie après. Ici, l'entrée est une suite non vide de nombres finis ; la sortie est sa plus petite valeur, sans modification de la suite.

Un **invariant** est une propriété conservée entre les étapes. Après chaque valeur traitée, `smallest` est le minimum du préfixe déjà parcouru. Un **préfixe** est ici la partie allant du début à la position courante.

Suivre les états

Minimum du préfixe déjà traité

Valeur lue	Minimum provisoire
8	8
3	3
5	3

Comparer les réalisations

Le résultat est 3. L'initialisation choisit une valeur réellement présente. Le premier tour compare cette valeur à elle-même : cela ne change pas le résultat. En MATLAB, l'exemple utilise explicitement un vecteur ligne : cette boucle en parcourt les valeurs.

Pseudo-code

```
// example.pseudo
// prefix_minimum
// Input: non-empty sequence of finite numbers.
smallest <- values[0]
FOR EACH value IN values DO
    IF value < smallest THEN
        smallest <- value
    END IF
END FOR
PRINT smallest
```

Python

```python
# prefix_minimum.py
values = [8, 3, 5]
smallest = values[0]
for value in values:
    if value < smallest:
        smallest = value
print(smallest)
```

MATLAB

```
% prefix_minimum.m
values = [8, 3, 5];
smallest = values(1);
for value = values
    if value < smallest
        smallest = value;
    end
end
disp(smallest);
```

Justifier et compter

Le parcours effectue n comparaisons pour n valeurs. Une seule variable de travail conserve le minimum. L'entrée vide est exclue par le contrat : un programme destiné à l'accepter doit définir un résultat ou une erreur avant de lire le premier élément.

Prévoir puis expérimenter

Refaites la trace avec [5, 3, 8], [-8, -3] et [4]. Le minimum ne dépend pas de l'ordre, mais les états intermédiaires peuvent changer. Essayez ensuite le vide : identifiez l'accès qui ne respecte plus la précondition.

Exercice corrigé

▼ Pourquoi initialiser smallest à 0 serait-il incorrect ?

Pour [8, 3, 5], le minimum serait alors annoncé comme 0, une valeur absente. L'initialisation doit établir l'invariant avant le parcours.

À retenir

- Le contrat précise le résultat, les erreurs et les mutations ; une représentation réalise les opérations de ce contrat.
- Un invariant doit être établi avant le premier tour, puis préservé à chaque étape.
- Initialiser un minimum avec le premier élément exige une entrée non vide.

Version web : https://www.clearcode.fr/courses/programming/dsa/foundations/contracts