Introduction à la récursivité

Cours de spécialité NSI de Terminale

D Pihoué

Lycée Camille Jullian Bordeaux

4 novembre 2023

Capacités attendues

1. Analyser le fonctionnement d'un programme récursif.
2. Écrire un programme récursif.

Capacités attendues

1. Analyser le fonctionnement d'un programme récursif.
2. Écrire un programme récursif.

Définition

On dit qu'une fonction est **récursive** lorsque son exécution peut provoquer un ou plusieurs appels à elle-même.

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.
À la main, on peut écrire

$$S(n) = 0 + 1 + 2 + \cdots + n$$

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.
À la main, on peut écrire

$$S(n) = 0 + 1 + 2 + \cdots + n$$

1. Proposez un code Python qui renvoie la valeur de $S(n)$ pour une valeur de n donnée.

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.
À la main, on peut écrire

$$S(n) = 0 + 1 + 2 + \cdots + n$$

1. Proposez un code Python qui renvoie la valeur de $S(n)$ pour une valeur de n donnée.
2. Modifiez l'écriture littérale de $S(n)$ pour formuler un calcul récursif.

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.
À la main, on peut écrire

$$S(n) = 0 + 1 + 2 + \cdots + n$$

1. Proposez un code Python qui renvoie la valeur de $S(n)$ pour une valeur de n donnée.
2. Modifiez l'écriture littérale de $S(n)$ pour formuler un calcul récursif.
3. Écrivez alors un algorithme en langage naturel pour calculer récursivement la valeur de $S(n)$ pour une valeur de n donnée.

Exemple

Considérons le problème simple du calcul de la somme $S(n)$ des n premiers nombres entiers naturels où n est un nombre entier naturel.
À la main, on peut écrire

$$S(n) = 0 + 1 + 2 + \cdots + n$$

1. Proposez un code Python qui renvoie la valeur de $S(n)$ pour une valeur de n donnée.
2. Modifiez l'écriture littérale de $S(n)$ pour formuler un calcul récursif.
3. Écrivez alors un algorithme en langage naturel pour calculer récursivement la valeur de $S(n)$ pour une valeur de n donnée.
4. Écrivez enfin le code de cet algorithme en langage Python.

Types de récursivité

Les exemples présentés ne comportent que des récursions simples, avec un seul appel. Il est cependant possible de rencontrer :

Types de récursivité

Les exemples présentés ne comportent que des récursions simples, avec un seul appel. Il est cependant possible de rencontrer :

- des cas récursifs multiples plutôt qu'un seul cas ;

Types de récursivité

Les exemples présentés ne comportent que des récursions simples, avec un seul appel. Il est cependant possible de rencontrer :

- des cas récursifs multiples plutôt qu'un seul cas ;
- une double récursion, la fonction s'appelle deux fois ;

Types de récursivité

Les exemples présentés ne comportent que des récursions simples, avec un seul appel. Il est cependant possible de rencontrer :

- des cas récursifs multiples plutôt qu'un seul cas ;
- une double récursion, la fonction s'appelle deux fois ;
- une récursion imbriquée, l'appel récursif comporte un appel à la fonction.

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

- un cas de base, qui ne nécessite pas d'appel récursif ;

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

- un cas de base, qui ne nécessite pas d'appel récursif ;
- des appels récursifs dont les arguments sont plus simples que ceux du contexte.

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

- un cas de base, qui ne nécessite pas d'appel récursif ;
- des appels récursifs dont les arguments sont plus simples que ceux du contexte.

Pour la **correction** d'un algorithme récursif, on veillera à

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

- un cas de base, qui ne nécessite pas d'appel récursif ;
- des appels récursifs dont les arguments sont plus simples que ceux du contexte.

Pour la **correction** d'un algorithme récursif, on veillera à

- ce que le problème soit bien adapté à un traitement récursif,

Terminaison et correction

Comme pour les boucles, on doit s'assurer de la **terminaison** en s'assurant que l'algorithme comporte bien

- un cas de base, qui ne nécessite pas d'appel récursif ;
- des appels récursifs dont les arguments sont plus simples que ceux du contexte.

Pour la **correction** d'un algorithme récursif, on veillera à

- ce que le problème soit bien adapté à un traitement récursif,
- reconstituer correctement la valeur de renvoi à partir des résultats des appels récursifs.

Preuve de la correction

Elle se démontre le plus souvent par **induction** en établissant les deux vérités suivantes :

Preuve de la correction

Elle se démontre le plus souvent par **induction** en établissant les deux vérités suivantes :

cas de base La fonction renvoie le bon résultat.

Preuve de la correction

Elle se démontre le plus souvent par **induction** en établissant les deux vérités suivantes :

cas de base La fonction renvoie le bon résultat.

appel récursif Si la fonction renvoie le bon résultat à l'issue de l'appel précédent, alors elle renvoie le bon résultat à l'issue de cet appel.

Preuve de la correction

Elle se démontre le plus souvent par **induction** en établissant les deux vérités suivantes :

cas de base La fonction renvoie le bon résultat.

appel récursif Si la fonction renvoie le bon résultat à l'issue de l'appel précédent, alors elle renvoie le bon résultat à l'issue de cet appel.

Conception

Pour concevoir un algorithme récursif, on commencera par réfléchir à l'expression d'une solution au problème posé à partir d'une solution pour une instance plus simple puis on déterminera le ou les cas de base.