

Autres algorithmes simples (partie 3)

Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 13 octobre 2025

Palindrome numérique

Enoncé du problème

Un entier positif n est un palindrome s'il peut se lire dans deux sens (ex : 35653).

Spécification du problème

- Donnée d'entrée : n , entier (le nombre à tester)
- Donnée de sortie : *palindrome*, booléen (résultat du test)
- Pré-condition : $n \geq 0$
- Post-condition : *palindrome* est VRAI si n peut se lire dans les 2 sens.

Signature de la fonction

- **est_palindrome** (n : entier) : booléen

Palindrome numérique

Corps de la fonction



Palindrome numérique

```
FONCTION est_palindrome (n : entier) : booléen
```

```
VARIABLE inverse : entier
```

```
VARIABLE copie : entier
```

```
inverse ← 0
```

```
copie ← n
```

```
TANT QUE copie > 0
```

```
    inverse ← inverse * 10 + copie mod 10
```

```
    copie ← copie div 10
```

```
FIN TANT QUE
```

```
RETOURNER inverse = n
```

```
FIN FONCTION
```

Palindrome numérique

- En réutilisant `inverse`

```
FONCTION  est_palindrome (n : entier) : booléen
```

```
    RETOURNER n = inverse(n)
```

```
FIN FONCTION
```

Primalité

Enoncé du problème

Un entier strictement positif n est **premier** s'il n'est divisible que par lui-même et par 1

Spécification du problème

- Donnée d'entrée : n , entier (le nombre à tester)
- Donnée de sortie : **premier**, booléen (résultat du test)
- Pré-condition : $n \geq 1$
- Post-condition : *premier* est VRAI si n est premier.

Signature de la fonction

- **est_premier** (n : entier) : booléen

Primalité

Corps de la fonction



Primalité

- Version 1 : compter les diviseurs

```
FONCTION est_premier (n : entier) : booléen
```

```
VARIABLE i : entier
```

```
VARIABLE diviseurs : entier
```

```
diviseurs ← 0
```

```
POUR i DE 1 A n PAR PAS DE 1
```

```
  SI n mod i = 0 ALORS
```

```
    diviseurs ← diviseurs + 1
```

```
  FIN SI
```

```
FIN POUR
```

```
RETOURNER diviseurs = 2
```

```
FIN FONCTION
```

Primalité

- Version 2 : s'arrêter si diviseur entre 1 et n exclus

```
FONCTION est_premier (n : entier) : booléen
```

```
VARIABLE i : entier
```

```
POUR i DE 2 A n-1 PAR PAS DE 1
```

```
SI n mod i = 0 ALORS  
  RETOURNER FAUX
```

```
FIN SI
```

```
FIN POUR
```

```
RETOURNER VRAI
```

```
FIN FONCTION
```

Primalité

- Version 3 : avec TANT QUE

FONCTION est_premier (n : entier) : booléen

VARIABLE i : entier

$i \leftarrow 2$

TANT QUE ($i < n$ ET $n \bmod i \neq 0$)

$i \leftarrow i + 1$

FIN SI

FIN POUR

RETOURNER $i = n + 1$

FIN FONCTION

Plus grand chiffre

Enoncé du problème

Trouver le plus grand chiffre d'un entier positif n

Spécification du problème

- Donnée d'entrée : n , entier (le nombre à tester)
- Donnée de sortie : chiffre, entier (le plus grand chiffre)
- Pré-condition : $n \geq 0$
- Post-condition : c est le plus grand chiffre présent dans n

Signature de la fonction

- **plus_grand_chiffre** (n : entier) : entier

Plus grand chiffre

Corps de la fonction



Plus grand chiffre

```
FONCTION plus_grand_chiffre (n : entier) : entier
```

```
VARIABLE copie : entier
```

```
VARIABLE c      : entier
```

```
copie ← n
```

```
c      ← 0
```

```
TANT QUE copie div 10  $\neq$  0
```

```
    c      ← copie mod 10
```

```
    copie ← copie div 10
```

```
FIN TANT QUE
```

```
RETOURNER c
```

```
FIN FONCTION
```

Somme des chiffres de rang impair

Enoncé du problème

Calculer la somme des chiffres de rang impair (le chiffre des unités, des centaines, des dizaines de milliers, ...) d'un entier n positif

Spécification du problème

- Donnée d'entrée : n , entier
- Donnée de sortie : $somme$, entier (somme des chiffres de rang impair)
- Pré-condition : $n \geq 0$
- Post-condition : $somme$ est la somme des chiffres de rang impair de n

Signature de la fonction

- $somme_impairs (n : \text{entier}) : \text{entier}$

Somme des chiffres de rang impair

Corps de la fonction



Somme des chiffres de rang impair

```
FONCTION somme_impairs (n : entier) : entier
```

```
VARIABLE copie : entier
```

```
VARIABLE somme : entier
```

```
VARIABLE rang : entier
```

```
copie ← n
```

```
some ← 0
```

```
rang ← 1
```

... (suite après)

```
FIN FONCTION
```

Somme des chiffres de rang impair

```
FONCTION somme_impairs (n : entier) : entier
```

```
...
```

```
REPETER
```

```
  SI rang mod 2 = 1 ALORS
```

```
    somme ← somme + copie mod 10
```

```
  FIN SI
```

```
  rang ← rang + 1
```

```
  copie ← copie div 10
```

```
JUSQU A copie = 0
```

```
RETOURNER somme
```

```
FIN FONCTION
```

Fin !

