

Introduction aux algorithmes II

Algorithmes et structures de données, 2025-2026

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA

2025-09-23

En partie inspiré des supports de cours de P. Albuquerque

- Quel est l'algorithme pour le calcul des nombres 1ers ?

Rappel

- Quel est l'algorithme pour le calcul des nombres premiers?

```
bool est_premier(int nombre) {  
    int i = 2; // bonne pratique!  
    while (i < nombre) { // penser à bien indenter!  
        if (0 == nombre % i) { // ça rend le code LISIBLE!  
            return false;  
        }  
        i += 1;  
    }  
    return true;  
}
```

Expressions et opérateurs (1/6)

Une expression est tout bout de code qui est **évalué**.

Expressions simples

- Pas d'opérateurs impliqués.
- Les littéraux, les variables, et les constantes.

```
const int L = -1; // 'L' est une constante, -1 un littéral
int x = 0;        // '0' est un littéral
int y = x;        // 'x' est une variable
int z = L;        // 'L' est une constante
```

Expressions complexes

- Obtenues en combinant des *opérandes* avec des *opérateurs*

```
int x;           // pas une expression (une instruction)
x = 4 + 5;       // 4 + 5 est une expression
                 // dont le résultat est affecté à 'x'
```

Expressions et opérateurs (2/6)

Opérateurs relationnels

Opérateurs testant la relation entre deux *expressions* :

- (a opérateur b) retourne **1** si l'expression s'évalue à **true**, **0** si l'expression s'évalue à **false**.

Opérateur	Syntaxe	Résultat
<	a < b	1 si a < b ; 0 sinon
>	a > b	1 si a > b ; 0 sinon
<=	a <= b	1 si a <= b ; 0 sinon
>=	a >= b	1 si a >= b ; 0 sinon
==	a == b	1 si a == b ; 0 sinon
!=	a != b	1 si a != b ; 0 sinon

Opérateurs logiques

Opérateur	Syntaxe	Signification
<code>&&</code>	<code>a && b</code>	ET logique
<code> </code>	<code>a b</code>	OU logique
<code>!</code>	<code>!a</code>	NON logique

Quiz : opérateurs logiques

Quiz : opérateurs logiques

Opérateurs arithmétiques

Opérateur	Syntaxe	Signification
+	a + b	Addition
-	a - b	Soustraction
*	a * b	Multiplication
/	a / b	Division
%	a % b	Modulo

Expressions et opérateurs (5/6)

Opérateurs d'assignation

Opérateur	Syntaxe	Signification
=	a = b	Affecte la valeur b à la variable a et retourne la valeur de b
+=	a += b	Additionne la valeur de b à a et assigne le résultat à a.
-=	a -= b	Soustrait la valeur de b à a et assigne le résultat à a.
*=	a *= b	Multiplie la valeur de b à a et assigne le résultat à a.
/=	a /= b	Divise la valeur de b à a et assigne le résultat à a.
%=	a %= b	Calcule le modulo la valeur de b à a et assigne le résultat à a.

Opérateurs d'assignation (suite)

Opérateur	Syntaxe	Signification
++	++a	Incrémente la valeur de a de 1 et retourne le résultat (a += 1).
--	--a	Décrémente la valeur de a de 1 et retourne le résultat (a -= 1).
++	a++	Retourne a et incrémente a de 1.
--	a--	Retourne a et décrémente a de 1.

Structures de contrôle : `if .. else if .. else` (1/2)

Syntaxe

```
if (expression) {  
    instructions;  
} else if (expression) { // optionnel  
    // il peut y en avoir plusieurs  
    instructions;  
} else {  
    instructions; // optionnel  
}
```

```
if (x) { // si x s'évalue à `vrai`  
    printf("x s'évalue à vrai.\n");  
} else if (y == 8) { // si y vaut 8  
    printf("y vaut 8.\n");  
} else {  
    printf("Ni l'un ni l'autre.\n");  
}
```

Structures de contrôle : `if .. else if .. else` (2/2)

Pièges

```
int x, y;  
x = y = 3;  
if (x = 2)  
    printf("x = 2 est vrai.\n");  
else if (y < 8)  
    printf("y < 8.\n");  
else if (y == 3)  
    printf("y vaut 3 mais cela ne sera jamais affiché.\n");  
else  
    printf("Ni l'un ni l'autre.\n");  
x = -1; // toujours évalué
```

Quiz : `if ... else`

Quiz : `if ... else`

Structures de contrôle : `while`

La boucle `while`

```
while (condition) {  
    instructions;  
}  
do {  
    instructions;  
} while (condition);
```

La boucle `while`, un exemple

```
int sum = 0; // syntaxe C99  
while (sum < 10) {  
    sum += 1;  
}  
do {  
    sum += 10;  
} while (sum < 100);
```

Structures de contrôle : `for`

La boucle `for`

```
for (expression1; expression2; expression3) {  
    instructions;  
}
```

La boucle `for`

```
int sum = 0; // syntaxe C99  
for (int i = 0; i < 10; i++) {  
    sum += i;  
}  
  
for (int i = 0; i != 1; i = rand() % 4) { // ésotérique  
    printf("C'est plus ésotérique.\n");  
}
```

Exercice : la factorielle

Écrire un programme qui calcule la factorielle d'un nombre

$$N! = 1 \cdot 2 \cdot \dots \cdot (N - 1) \cdot N.$$

Par groupe de 3 : écrire un pseudo-code

Exercice : la factorielle

Écrire un programme qui calcule la factorielle d'un nombre

$$N! = 1 \cdot 2 \cdot \dots \cdot (N - 1) \cdot N.$$

Par groupe de 3 : écrire un pseudo-code

```
entier factorielle(entier n)
    i = 1
    fact = 1
    tant que i <= n
        fact *= i
        i += 1
    retourne fact
```

Exercice : la factorielle

Écrire un programme qui calcule la factorielle d'un nombre

$$N! = 1 \cdot 2 \cdot \dots \cdot (N - 1) \cdot N.$$

Par groupe de 3 : écrire un code en C

Quand vous avez fini postez le code sur le salon matrix.

Exercice : la factorielle

Écrire un programme qui calcule la factorielle d'un nombre

$$N! = 1 \cdot 2 \cdot \dots \cdot (N - 1) \cdot N.$$

Par groupe de 3 : écrire un code en C

Quand vous avez fini postez le code sur le salon matrix.

```
#include <stdio.h>
int main() {
    int nb = 10;
    int fact = 1;
    int iter = 2;
    while (iter <= nb) {
        fact *= iter;
        iter++;
    }
    printf("La factorielle de %d est %d\n", nb, fact);
}
```

Exercice : la factorielle

Écrire un programme qui calcule la factorielle d'un nombre

$$N! = 1 \cdot 2 \cdot \dots \cdot (N - 1) \cdot N.$$

Par groupe de 3 : écrire un code en C

Quand vous avez fini postez le code sur le salon matrix.

```
#include <stdio.h>
int main() {
    int nb = 10;
    int fact = 1;
    int iter = 2;
    while (iter <= nb) {
        fact *= iter;
        iter++;
    }
    printf("La factorielle de %d est %d\n", nb, fact);
}
```

Comment améliorer ce code ? (notez ça sur une feuille)

Exercice : la factorielle en mieux

Individuellement

1. Écrivez l'algorithme de calcul de deux façon différentes.
2. Que se passe-t-il si $n \geq 15$?
3. Pour celles et ceux qui ont fini pendant que les autres essaient : faites-le en récursif (sans aide).

Exercice : la factorielle en mieux

Individuellement

1. Écrivez l'algorithme de calcul de deux façon différentes.
2. Que se passe-t-il si $n \geq 15$?
3. Pour celles et ceux qui ont fini pendant que les autres essaient : faites-le en récursif (sans aide).

Postez vos solutions sur matrix !

Exercice : test si un nombre est premier

Avec tout ce que vous avez appris jusqu'ici :

- Écrivez le code testant si un nombre est premier.
- Quels sont les ajouts possibles par rapport au code de la semaine passée ?
- Rencontrez-vous des problèmes éventuels de compilation ?
- Voyez-vous une façon de générer des nombres premiers avec votre programme ?

Exercice : test si un nombre est premier

Avec tout ce que vous avez appris jusqu'ici :

- Écrivez le code testant si un nombre est premier.
- Quels sont les ajouts possibles par rapport au code de la semaine passée ?
- Rencontrez-vous des problèmes éventuels de compilation ?
- Voyez-vous une façon de générer des nombres premiers avec votre programme ?

Implémentez-la et postez votre code sur le salon matrix (10 min).

Corrigé : enfin pas vraiment mais juste un possible

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
int main() {
    int nb = 1;
    printf("Entrez un nombre: ");
    scanf("%d", &nb);
    bool premier = true;
    for (int div = 2; div <= sqrt(nb); div++) {
        if (nb % div == 0) {
            premier = false;
            break;
        }
    }
    if (premier) {
        printf("Le nombre %d est premier\n", nb);
    } else {
        printf("Le nombre %d n'est pas premier\n", nb);
    }
    return 0;
}
```

Quelques algorithmes simples

Voyons quelques algorithmes supplémentaires

- Plus petit commun multiple (PPCM) de deux nombres
- Autre algorithme de calcul du PPCM de deux nombres
- Plus grand commun diviseur (PGCD) de deux nombres

Le calcul du PPCM (1/5)

Définition

Le plus petit commun multiple (PPCM), p , de deux entiers non nuls, a et b , est le plus petit entier strictement positif qui soit multiple de ces deux nombres.

Exemples :

$$\text{PPCM}(3, 4) = 12,$$

$$\text{PPCM}(4, 6) = 12,$$

$$\text{PPCM}(5, 15) = 15.$$

Le calcul du PPCM (1/5)

Définition

Le plus petit commun multiple (PPCM), p , de deux entiers non nuls, a et b , est le plus petit entier strictement positif qui soit multiple de ces deux nombres.

Exemples :

$$\text{PPCM}(3, 4) = 12,$$

$$\text{PPCM}(4, 6) = 12,$$

$$\text{PPCM}(5, 15) = 15.$$

Mathématiquement

Décomposition en nombres premiers :

$$36 = 2^2 \cdot 3^2, \quad 90 = 2 \cdot 5 \cdot 3^2,$$

On garde tous les premiers à la puissance la plus élevée

$$\text{PPCM}(36, 90) = 2^2 \cdot 3^2 \cdot 5 = 180$$

Le calcul du PPCM (2/5)

Exemple d'algorithme

PPCM(36, 90):

36 < 90 // 36 + 36

72 < 90 // 72 + 36

108 > 90 // 90 + 90

108 < 180 // 108 + 36

144 < 180 // 144 + 36

180 = 180 // The End!

- 5 additions, 5 assignations, et 6 comparaisons.

Le calcul du PPCM (2/5)

Exemple d'algorithme

PPCM(36, 90):

36 < 90 // 36 + 36

72 < 90 // 72 + 36

108 > 90 // 90 + 90

108 < 180 // 108 + 36

144 < 180 // 144 + 36

180 = 180 // The End!

- 5 additions, 5 assignations, et 6 comparaisons.

Transcrivez cet exemple en algorithme (groupe de 3), 5min

Le calcul du PPCM (2/5)

Exemple d'algorithme

PPCM(36, 90):

36 < 90 // 36 + 36

72 < 90 // 72 + 36

108 > 90 // 90 + 90

108 < 180 // 108 + 36

144 < 180 // 144 + 36

180 = 180 // The End!

- 5 additions, 5 assignations, et 6 comparaisons.

Transcrivez cet exemple en algorithme (groupe de 3), 5min

et codez-le !

Le calcul du PPCM (3/5)

Tentative de correction

```
int main() {  
    int m = 15, n = 12;  
    int mult_m = m, mult_n = n;  
    while (mult_m != mult_n) {  
        if (mult_m > mult_n) {  
            mult_n += n;  
        } else {  
            mult_m += m;  
        }  
    }  
    printf("Le ppcm de %d et %d est %d\n", n, m, mult_m);  
}
```


Le calcul du PPCM (3/5)

Tentative de correction

```
int main() {  
    int m = 15, n = 12;  
    int mult_m = m, mult_n = n;  
    while (mult_m != mult_n) {  
        if (mult_m > mult_n) {  
            mult_n += n;  
        } else {  
            mult_m += m;  
        }  
    }  
    printf("Le ppcm de %d et %d est %d\n", n, m, mult_m);  
}
```

- Combien d'additions / comparaisons au pire ?

Le calcul du PPCM (4/5)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

Le calcul du PPCM (4/5)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

En C on pourrait la décrire comme

```
int ppcm(int a, int b); // La **signature** de cette fonction
```

Le calcul du PPCM (4/5)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

En C on pourrait la décrire comme

```
int ppcm(int a, int b); // La **signature** de cette fonction
```

Algorithme

Par groupe de 3 (5-10min) :

- réfléchissez à un algorithme alternatif donnant le PPCM de deux nombres ;
- écrivez l'algorithme en pseudo-code.

Indication

Si un nombre, p , est multiple de a et de b alors il peut s'écrire $p = a * i = b * j$ ou encore $p / a = i$ et $p / b = j$.

Le calcul du PPCM (5/5)

Indication

Si un nombre, p , est multiple de a et de b alors il peut s'écrire $p = a * i = b * j$ ou encore $p / a = i$ et $p / b = j$.

Pseudo-code

```
int ppcm(int a, int b) {  
    for (i in [1, b]) {  
        if a * i est divisible par b {  
            return a * i  
        }  
    }  
}
```

Le code du PPCM de 2 nombres (1/2)

Implémentez le pseudo-code et postez le code sur matrix (5min).

Le code du PPCM de 2 nombres (1/2)

Implémentez le pseudo-code et postez le code sur matrix (5min).

Un corrigé possible

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int n = 15, m = 12;
    int i = 1;
    while (n * i % m != 0) {
        i++;
    }
    printf("Le ppcm de %d et %d est %d\n", n, m, n*i);
}
```


Le code du PPCM de 2 nombres (2/2)

Corrigé alternatif

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int res = n*m;
    for (int i = 2; i <= m; i++) {
        if (n * i % m == 0) {
            res = n * i;
            break;
        }
    }
    printf("Le ppcm de %d et %d est %d\n", n, m, res);
}
```