

Module Algorithmie et programmation

Alorithmes simples, chaînes de caractères, tableaux 2D, réusinage

Pierre Künzli

Adapté des cours de Paul Albuquerque, Guido Bologna et Orestis Malaspinas

Avant de commencer

Télécharger et installer la messagerie Matrix Element :

<https://element.io/>

Rejoindre le salon “Algorithmique et programmation” dans l’espace

[https://matrix.to/#/!xoHlwwGcPDQjZBQiOZ:
matrix.org?via=matrix.org](https://matrix.to/#/!xoHlwwGcPDQjZBQiOZ:matrix.org?via=matrix.org).

Quelques algorithmes simples

Quelques algorithmes simples

- Plus petit commun multiple (PPCM) de deux nombres
- Plus grand commun diviseur (PGCD) de deux nombres
- Générer des nombres premiers : crible d'Ératosthène

Le calcul du PPCM

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

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

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 assignments, et 6 comparaisons.

Le calcul du PPCM

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 (5 min)

Postez votre proposition sur le salon Element

Le calcul du PPCM

Proposition d'implémentation

```
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);  
}
```

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

Le calcul du PPCM

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

Réusinage : Proposition d'implémentation

```
int ppcm(int a, int b){
    int mult_a = a, mult_b = b;
    while (mult_a != mult_b) {
        if (mult_a > mult_b) {
            mult_b += b;
        } else {
            mult_a += a;
        }
    }
    return mult_a;
}

int main() {
    int m = 15, n = 12;
    printf("Le ppcm de %d et %d est %d\n", n, m, ppcm(m, n));
}
```

Le calcul du PGCD

Définition

Le plus grand commun diviseur (PGCD) de deux nombres entiers non nuls est le plus grand entier qui les divise en même temps.

Exemples :

$$\text{PGCD}(3, 4) = 1,$$

$$\text{PGCD}(4, 6) = 2,$$

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

Le calcul du PGCD

Définition

Le plus grand commun diviseur (PGCD) de deux nombres entiers non nuls est le plus grand entier qui les divise en même temps.

Exemples :

$$\text{PGCD}(3, 4) = 1,$$

$$\text{PGCD}(4, 6) = 2,$$

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

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 basse

$$\text{PGCD}(36, 90) = 2^{\min 1, 2} \cdot 3^{\min 2, 2} \cdot 5^{\min 0, 1} = 18.$$

Le calcul du PGCD

Exemple d'algorithme

PGCD(36, 90):

```
90 % 36 != 0 && 36 % 36 == 0
```

```
90 % 35 != 0 && 36 % 35 != 0
```

```
90 % 34 != 0 && 36 % 35 != 0
```

```
...
```

```
90 % 19 != 0 && 36 % 19 != 0
```

```
90 % 18 == 0 && 36 % 18 == 0 // The end!
```

- 18 modulus, 18 assignments, et 18 comparaisons.

Le calcul du PGCD

Exemple d'algorithme, proposition d'implémentation

```
void main() {  
    int n = 90, m = 78;  
    int gcd = 1;  
    for (int div = n; div >= 2; div--) {  
        if (n % div == 0 && m % div == 0) {  
            gcd = div;  
            break;  
        }  
    }  
    printf("Le pgcd de %d et %d est %d\n", n, m, gcd);  
}
```


Le calcul du PGCD

Exemple d'algorithme, proposition d'implémentation

```
void main() {  
    int n = 90, m = 78;  
    int gcd = 1;  
    for (int div = n; div >= 2; div--) {  
        if (n % div == 0 && m % div == 0) {  
            gcd = div;  
            break;  
        }  
    }  
    printf("Le pgcd de %d et %d est %d\n", n, m, gcd);  
}
```

- Combien d'additions / comparaisons au pire ?
- Un moyen de le rendre plus efficace ?

Le calcul du PGCD

Réusinage : l'algorithme d'Euclide

Si a et b sont deux entiers naturels, si r est le reste de a divisé par b , alors le pgcd de a et b vaut le pgcd de b et r .

PGCD(35, 60):

35 / 60 = 0, reste 35

60 / 35 = 1, reste 25

35 / 25 = 1, reste 10

25 / 10 = 2, reste 5

10 / 5 = 2, reste 0 -> PGCD = 5

Le calcul du PGCD

Réusinage : l'algorithme d'Euclide

Si a et b sont deux entiers naturels, si r est le reste de a divisé par b , alors le pgcd de a et b vaut le pgcd de b et r .

PGCD(35, 60):

35 / 60 = 0, reste 35

60 / 35 = 1, reste 25

35 / 25 = 1, reste 10

25 / 10 = 2, reste 5

10 / 5 = 2, reste 0 \rightarrow PGCD = 5

- Combien d'opérations / comparaisons ?
- Combien d'opérations / comparaisons pour la version précédente ?

Le calcul du PGCD

Réusinage : l'algorithme d'Euclide

Si a et b sont deux entiers naturels, si r est le reste de a divisé par b , alors le pgcd de a et b vaut le pgcd de b et r .

PGCD(35, 60):

35 / 60 = 0, reste 35

60 / 35 = 1, reste 25

35 / 25 = 1, reste 10

25 / 10 = 2, reste 5

10 / 5 = 2, reste 0 \rightarrow PGCD = 5

- Combien d'opérations / comparaisons ?
- Combien d'opérations / comparaisons pour la version précédente ?

Proposez un pseudocode et une implémentation de l'algorithme d'Euclide sur le chat du cours (8 min)

Un corrigé possible

```
#include <stdio.h>

void main() {
    int n = 90;
    int m = 78;
    printf("n = %d et m = %d\n", n, m);
    int tmp_n = n;
    int tmp_m = m;
    while (tmp_n % tmp_m > 0) {
        int tmp = tmp_n;
        tmp_n = tmp_m;
        tmp_m = tmp % tmp_m;
    }
    printf("Le pgcd de %d et %d est %d\n", n, m, tmp_m);
}
```

Le calcul du PGCD

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

Le calcul du PGCD

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

En C on pourrait la décrire comme

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

Réusinez le code précédent avec cette fonction, à faire en exercice

Crible d'Ératosthène

Algorithme de génération de nombres premiers.

Exercice

- À l'aide d'un tableau de booléens,
- Générer les nombres premiers plus petits qu'un nombre N

Pseudo-code

- Par groupe de deux ou trois, réfléchir à un algorithme.

Programme en C

- Implémenter l'algorithme et le poster sur le salon `Element`.

Crible d'Ératosthène : solution

```
#include <stdio.h>
#include <stdbool.h>
#define SIZE 51
int main() {
    bool tab[SIZE];
    for (int i=0;i<SIZE;i++) {
        tab[i] = true;
    }
    for (int i = 2; i < SIZE; i++) {
        if (tab[i]) {
            printf("%d ", i);
            int j = i;
            while (j < SIZE) {
                j += i;
                tab[j] = false;
            }
        }
    }
    printf("\n");
}
```

Chaînes de caractères, suite

Qu'est-ce qu'un tableau statique ?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**
- alloués de façon **contigüe** (en bloc) en mémoire
- sur la **pile**
- dont la taille **ne peut pas** être changée.

Chaînes de caractères (strings), rappel

- Chaîne de caractère == tableau de caractères **terminé par la valeur** `'\0'` ou `0`.

Exemple

```
char *str  = "HELLO !";  
char str[] = "HELLO !";
```

Est représenté par

char	H	E	L	L	O		!	\0
ASCII	72	69	76	76	79	32	33	0

Syntaxe

```
char name[5];  
name[0] = 'P'; // = 70;  
name[1] = 'a'; // = 97;  
name[2] = 'u'; // = 117;  
name[3] = 'l'; // = 108;  
name[4] = '\\0'; // = 0;  
char name[] = {'P', 'a', 'u', 'l', '\\0'};  
char name[5] = "Paul";  
char name[] = "Paul";  
char name[100] = "Paul is not 100 characters long.";
```

Remarque

```
char str[] = "abc" != char str[] = {'a', 'b', 'c'}  
char str[] = "abc";  
char str2[] = {'a', 'b', 'c'};  
// Que vaut sizeof(str) ? et sizeof(str2) ?
```

Fonctions

- Il existe une grande quantités de fonction pour la manipulation de chaînes de caractères dans `string.h`.
- Fonctions principales :

```
// longueur de la chaîne (sans le \0)  
size_t strlen(char *str);  
// copie jusqu'à un \0  
char *strcpy(char *dest, const char *src);  
    // copie len char  
char *strncpy(char *dest, const char *src, size_t len);  
// compare len chars  
int strcmp(char *str1, char *str2, size_t len);  
// compare jusqu'à un \0  
int strcmp(char *str1, char *str2);
```

- Pour avoir la liste complète : man `string`.

Fonctions

- Il existe une grande quantités de fonction pour la manipulation de chaînes de caractères dans `string.h`.
- Fonctions principales :

```
// longueur de la chaîne (sans le \0)
size_t strlen(char *str);
// copie jusqu'à un \0
char *strcpy(char *dest, const char *src);
// copie len char
char *strncpy(char *dest, const char *src, size_t len);
// compare len chars
int strncmp(char *str1, char *str2, size_t len);
// compare jusqu'à un \0
int strcmp(char *str1, char *str2);
```

- Pour avoir la liste complète : `man string`.

Quels problèmes peuvent se produire avec `strlen`, `strcpy`, `strcmp` ?

Les anagrammes

Définition

Deux mots sont des anagrammes l'un de l'autre quand ils contiennent les mêmes lettres mais dans un ordre différent.

Exemple

t	u	t	u	t	\0
t	u	t	t	u	\0

Problème : Trouvez un algorithme pour déterminer si deux mots sont des anagrammes.

Les anagrammes

Il suffit de :

1. Trier les deux mots.
2. Vérifier s'ils contiennent les mêmes lettres.

Une implémentations

On suppose qu'on sait trier un tableau

```
int main() { // pseudo C
    tri(mot1);
    tri(mot2);
    if egalite(mot1, mot2) {
        // anagrammes
    } else {
        // pas anagrammes
    }
}
```

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite :

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite :

- rotor, kayak, ressasser, ...

Problème : proposer un algorithme pour détecter un palindrome

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite :

- rotor, kayak, ressasser, ...

Problème : proposer un algorithme pour détecter un palindrome

Solution 1

```
while (first_index < last_index {  
    if (mot[first_index] != mot [last_index]) {  
        return false;  
    }  
    first_index += 1;  
    last_index -= 1;  
}  
return true;
```

Autre idée ?

Les palindromes

Solution 2

```
mot_tmp = revert(mot)
return egalite(mot, mot_tmp)
```

Nécessite une fonction `revert` qui inverse une chaîne de caractères.

Tableau à deux dimensions

Tableau à deux dimensions

Mais qu'est-ce donc ?

Tableau à deux dimensions

Mais qu'est-ce donc ?

- Un tableau où chaque cellule est un tableau.

Quels cas d'utilisation ?

Tableau à deux dimensions

Mais qu'est-ce donc ?

- Un tableau où chaque cellule est un tableau.

Quels cas d'utilisation ?

- Tableau à double entrée ;
- Image ;
- Écran (pixels) ;
- Matrice (mathématique) ;

Tableau à deux dimensions

Exemple : tableau à 3 lignes et 4 colonnes d'entiers

indices	0	1	2	3
0	7	4	7	3
1	2	2	9	2
2	4	8	8	9

Syntaxe

```
int tab[3][4]; // déclaration d'un tableau 4x3  
tab[2][1]; // accès à la case 2, 1  
tab[2][1] = 14; // assignation de 14 à la position 2, 1
```

Tableau à deux dimensions

Exercice : déclarer et initialiser aléatoirement un tableau 50x100

Tableau à deux dimensions

Exercice : déclarer et initialiser aléatoirement un tableau 50x100

```
#define NX 50
#define NY 100
int tab[NY][NX];
for (int i = 0; i < NY; ++i) {
    for (int j = 0; j < NX; ++j) {
        tab[i][j] = rand() % 256; // 256 niveaux de gris
    }
}
```

Exercice : afficher le tableau

Tableau à deux dimensions

Exercice : déclarer et initialiser aléatoirement un tableau 50x100

```
#define NX 50
#define NY 100
int tab[NY][NX];
for (int i = 0; i < NY; ++i) {
    for (int j = 0; j < NX; ++j) {
        tab[i][j] = rand() % 256; // 256 niveaux de gris
    }
}
```

Exercice : afficher le tableau

```
for (int i = 0; i < NY; ++i) {
    for (int j = 0; j < NX; ++j) {
        printf("%d ", tab[i][j]);
    }
    printf("\n");
}
```

Tableau à deux dimensions

Attention

- Les éléments ne sont **jamais** initialisés.
- Les bornes ne sont **jamais** vérifiées.

```
int tab[3][2] = { {1, 2}, {3, 4}, {5, 6} };  
printf("%d\n", tab[2][1]); // affiche?  
printf("%d\n", tab[10][9]); // affiche?  
printf("%d\n", tab[3][1]); // affiche?
```

La couverture de la reine

- Aux échecs la reine peut se déplacer horizontalement et verticalement
- Pour un échiquier 5x6, elle *couvre* les cases comme ci-dessous

	0	1	2	3	4	5
0	*		*		*	
1		*	*	*		
2	*	*	R	*	*	*
3		*	*	*		
4	*		*		*	

Exercice

- En utilisant les conditions, les tableaux à deux dimensions, et des `char` uniquement.
- Implémenter un programme qui demande à l'utilisateur d'entrer les coordonnées de la reine et affiche un tableau comme ci-dessus pour un échiquier 8x8.

Réusinage de code

Réusinage de code (refactoring)

Le réusinage est ?

Réusinage de code (refactoring)

Le réusinage est ?

- le processus de restructuration d'un programme :
 - en modifiant son design,
 - en modifiant sa structure,
 - en modifiant ses algorithmes
 - mais en **conservant ses fonctionnalités**.

Réusinage de code (refactoring)

Le réusinage est ?

- le processus de restructuration d'un programme :
 - en modifiant son design,
 - en modifiant sa structure,
 - en modifiant ses algorithmes
 - mais en **conservant ses fonctionnalités**.

Avantages ?

Réusinage de code (refactoring)

Le réusinage est ?

- le processus de restructuration d'un programme :
 - en modifiant son design,
 - en modifiant sa structure,
 - en modifiant ses algorithmes
 - mais en **conservant ses fonctionnalités**.

Avantages ?

- Amélioration de la lisibilité,
- Amélioration de la maintenabilité,
- Réduction de la complexité (performances).

“Make it work, make it nice, make it fast”, Kent Beck.

Réusinage de code (refactoring)

Le réusinage est ?

- le processus de restructuration d'un programme :
 - en modifiant son design,
 - en modifiant sa structure,
 - en modifiant ses algorithmes
 - mais en **conservant ses fonctionnalités**.

Avantages ?

- Amélioration de la lisibilité,
- Amélioration de la maintenabilité,
- Réduction de la complexité (performances).

“Make it work, make it nice, make it fast”, Kent Beck.

Exercice :

- Réusiner ce code
<https://githopia.hesge.ch/algoprogramsoir/algoprogram-etu-22-23/-/blob/main/Programmation/Code/Refactor/comprendre.c>