# Calcul du PGCD

**Objectifs** : se familiariser avec la syntaxe C#, les opérateurs et structures de contrôle, la saisie et l’affichage sur la console

On souhaite écrire un programme de calcul du pgcd de deux entiers non nuls, en C# à partir de l’algorithme de la méthode dite « égyptienne ». Voici une spécification de l'algorithme de calcul du PGCD de deux nombres (entiers strictement positifs) p et q, selon cette méthode :

Lire (p, q);

Tantque p ≠ q faire

Si p > q alors

p ← p – q

sinon

q ← q – p

FinSi

FinTant;

Ecrire(" PGCD = ", p)

Ecrivez le programme C# complet qui produise le résultat suivant :

Entrez le premier nombre : 21

Entrez le deuxième nombre : 45

Le PGCD de 21 et 45 est : 3

# Calcul de nombres premiers

**Objectifs** : se familiariser avec la syntaxe C#, les opérateurs et structures de contrôle, la saisie et l’affichage sur la console.

On souhaite écrire un programme C# de calcul et d'affichage des N premiers nombres premiers. Un nombre entier est premier s’il n’est divisible que par 1 et par lui-même (ex : 1, 2, 3, 5, 7, 11, 13…)

Le programme ne doit utiliser que des boucles (do) while, mais des boucles for.

Spécifications de l’algorithme : (on étudie la primalité des nombres uniquement impairs)

Algorithme Premier

Entrée: n ∈ N

Sortie: nbr ∈ N

Local: estPremier ∈ {Vrai , Faux}

divis, compt ∈ N²;

début

lire(n);

compt ← 1;

ecrire(2);

nbr ← 3;

Tantque(compt < n) Faire

divis ← 3;

Est\_premier ← Vrai;

Répéter

Si reste(nbr par divis) = 0 Alors

estPremier ← Faux

Sinon

divis ← divis+2

Fsi

jusquà (divis > nbr / 2) ou (estPremier=Faux);

Si estPremier = Vrai Alors

ecrire(nbr);

compt ← compt+1

Fsi;

nbr ← nbr+2 // nbr impairs

Ftant

FinPremier

Ecrivez le programme C# complet qui produise le résultat suivant :

Combien de nombres premiers : 5

2

3

5

7

11

# Tri d’un tableau

**Etape 1** : Créer une fonction permettant d’afficher le contenu d’un tableau (sur une seule ligne).

**Etape 2** : Créer une fonction permettant de trier par ordre alphabétique les éléments d’un tableau de mots passé en paramètre.

Indications :

* Pour trier un tableau, on peut comparer ses éléments deux à deux, et les permuter jusqu’à ce qu’on arrive à parcourir le tableau sans faire aucune permutation.
* Pour comparer une chaîne avec une autre, on peut utiliser la méthode CompareTo(). Exemple : mot1.CompareTo(mot2) ;

**Etape 3** : tester le tri grâce à la première fonction

**Etape 4** : modifier le type du paramètre d’entrée de la fonction en IComparable[]. Que constatez-vous ?

**Etape 5** : faire en sorte que la fonction ne modifie pas le tableau passé en paramètre, mais renvoie plutôt le tableau trié en résultat. Tester avec la fonction d’affichage.

# Comptage des voyelles et consonnes

**Objectifs** : se familiariser avec les bases du langage C# : tableaux, fonctions, passage de paramètres en ref et out

**Etape 1** : Dans la fonction Main() :

* Faire saisir un mot à l’utilisateur. On ne fera pas de vérification du mot saisi ; on s’attend à ce qu’il ne comporte que des lettres.
* Créer une fonction vide qui prend en entrée un mot, calcule ses nombres de voyelles et consonnes, et les renvoie en paramètres out.
* Afficher le résultat de l’appel de cette fonction sous la forme : « ”livre” comprend 3 consonnes et 2 voyelles »

**Etape 2** : implémenter le corps de la fonction vide créée précédemment et tester.