

**Université libre de Bruxelles  
Faculté des Sciences Appliquées**

**INFOH200  
Algorithmique et programmation  
Fascicule d'exercices**

## Les structures conditionnelles et itératives.

---

### Exercice 1.

Ecrire un programme qui lit au clavier les coefficients  $a$ ,  $b$  et  $c$  de l'équation du second degré  $ax^2 + bx + c = 0$  et affiche à l'écran, si elle(s) existe(nt), la ou les solutions réelles ou complexes de cette équation. Si l'équation n'a pas de solution, le programme le signalera en affichant un message à l'écran. Veillez à envisager tous les cas possibles.

### Exercice 2.

Ecrire un programme qui calcule la moyenne d'une série de nombres entiers positifs ou nuls lus au clavier et l'affiche à l'écran. Le programme s'arrête dès qu'on introduit un nombre négatif. On pourra supposer qu'il y a au moins un nombre positif dans la série.

### Exercice 3.

Ecrire un programme qui lit des nombres entiers au clavier tant que ceux-ci sont en ordre croissant. Le programme affichera le nombre de valeurs en ordre croissant lues.

### Exercice 4.

Ecrire un programme qui lit des nombres au clavier et les affiche tant que le dernier nombre lu est la somme des deux précédents.

### Exercice 5.

Ecrire un programme qui affiche la factorielle d'un nombre entier positif lu au clavier.

### Exercice 6.

Le nombre  $e$  peut être défini comme la limite d'une série :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \Lambda + \frac{1}{n!} + \Lambda$$

Il est connu que l'approximation :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \Lambda + \frac{1}{n!}$$

diffère de  $e$  d'au plus deux fois le terme suivant dans la série, c'est-à-dire d'au plus  $2 \frac{1}{(n+1)!}$

Ecrire un programme calculant  $e$  avec une approximation dont l'erreur est inférieure à une constante  $\varepsilon$  donnée.

### Exercice 7.

Ecrire un programme qui permet de faire deviner un nombre choisi aléatoirement entre 0 et 100 (fonction `random` : `nbatrouver=(rand()%(max+1))+min;`, de la librairie `stdlib`). Le programme signalera à chaque tentative si le nombre proposé est plus petit ou plus grand que la solution.

## Les fonctions.

---

### Exercice 1.

Ecrire une fonction à valeur booléenne qui teste la primalité d'un nombre entier positif passé en paramètre.

### Exercice 2.

Ecrire la fonction  $\text{swap}(a,b)$  qui échange les valeurs de deux variables entières  $a$  et de  $b$  passée en paramètre.

### Exercice 3.

Ecrire une fonction qui retourne une approximation du sinus d'un nombre passé en paramètre en utilisant le développement de la fonction sinus suivant.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \Lambda$$

On arrêtera d'effectuer la somme si le dernier terme ajouté est plus petit qu'une constante  $\varepsilon$  fixée dans le programme ou si on a déjà sommé un nombre de termes égal à une constante  $MAX$  définie dans le programme.

### Exercice 4.

Ecrire une fonction qui trie en ordre croissant trois variables entières passée en paramètre.

### Exercice 5.

Ecrire une librairie contenant les fonctions maximum de deux entiers, minimum de deux entiers et valeur absolue d'un entier. Utiliser cette librairie pour écrire une fonction qui renvoie la valeur absolue du plus petit entier en valeur absolue parmi trois entiers passés en paramètres.

### Exercice 6.

On peut calculer la puissance  $n^{\text{ième}}$  ( $n \geq 0$ ) d'un réel  $x$  en effectuant le moins de multiplications possibles à partir des suites  $(y_i)$ ,  $(p_i)$  et  $(z_i)$  définies comme suit.

$(y_i)$  est la suite  $(x, x^2, x^4, x^8, \dots)$  c-à-d  $y_i = x^{2^i}$

$$\begin{aligned} p_0 &= n \\ p_{i+1} &= p_i / 2 \text{ (division entière)} \end{aligned}$$

$$\begin{aligned} z_0 &= 1 \\ z_{i+1} &= z_i y_i \quad \text{si } p_i \text{ est impair} \\ &= z_i \quad \text{sinon} \end{aligned}$$

A partir d'un certain indice, les termes de la suite  $(p_i)$  s'annulent. Le terme de la suite  $(z_i)$  correspondant à cet indice vaut alors  $x^n$ .

On demande d'écrire une fonction qui reçoit comme paramètres un réel  $x$  et un entier  $n$  et retourne  $x^n$  en utilisant la méthode décrite ci-dessus. On veillera à optimiser le nombre de multiplications dans le calcul des éléments de la suite  $(y_i)$ .

### Exercice 7.

On peut calculer une approximation de  $\varphi$  en utilisant le fait que

$$\varphi = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \Lambda}}}}$$

Si on définit la suite  $(x_i)$  comme ci-dessous, elle convergera vers  $\varphi$ .

$$x_1 = 1 = 1$$

$$x_2 = 1 + \frac{1}{1} = 2$$

$$x_3 = 1 + \frac{1}{1 + \frac{1}{1}} = 1,5$$

$$x_4 = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}} = 1.666\dots$$

Λ

Ecrire une fonction qui calcule une approximation de  $\varphi$  en retournant le  $n^{\text{ième}}$  élément de cette suite. La fonction recevra donc comme paramètre un entier  $n$ .

### Exercice 8.

Ecrire une fonction qui calcule la somme des chiffres d'un nombre entier strictement positif et recommence le calcul avec le résultat obtenu tant que celui-ci n'est pas compris entre 1 et 9. Après chaque calcul la fonction affiche à l'écran la somme obtenue. La fonction retournera le nombre entre 1 et 9 obtenu.

#### Exemple.

Si le nombre passé en paramètre à la fonction est 123456.

La fonction affichera

21 (car  $1+2+3+4+5+6=21$ )

3 (car  $2+1=3$ )

La fonction retournera 3.

*Indication* : le dernier chiffre décimal d'un nombre entier positif  $n$  est  $n \% 10$ . Le nombre obtenu en amputant un nombre entier positif  $n$  de son dernier chiffre décimal est  $n / 10$  (division entière).

### Exercice 9.

On souhaite disposer d'un module *tools* contenant les fonction max, min, abs, et swap pour des nombres entiers. On vous demande d'écrire ce module, le fichier d'entête correspondant et un exemple de module principal qui l'utilise.

## Les tableaux.

---

### Exercice 1.

Ecrire une fonction qui retourne le maximum d'un tableau à une dimension de  $n$  nombres entiers.

### Exercice 2.

Ecrire une fonction qui retourne la somme des éléments d'un tableau d'entiers passé en paramètre.

### Exercice 3.

Ecrire une fonction qui opère une symétrie sur les éléments d'un tableau de taille  $n$ , c'est-à-dire qui transforme le tableau  $(v_1, v_2, \dots, v_n)$  en  $(v_n, v_{n-1}, \dots, v_1)$ .

### Exercice 4.

Un polynôme de degré  $n$  à coefficients entiers,  $a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ , où  $a_n \neq 0$ , peut être représenté dans un tableau d'entiers dont seuls les  $n$  premiers éléments sont significatifs, l'élément d'indice  $i$  (où  $i \leq n$ ) du tableau contenant le coefficient  $a_i$  du polynôme. Le polynôme nul, de degré 0, sera représenté par un tableau d'entiers dont l'unique élément significatif, d'indice 0, est nul.

On supposera que la fonction demandée ne pourra manipuler que des polynômes de degré inférieur ou égal à une constante *DEGRE\_MAX*.

#### Exemple.

Le polynôme  $1 + 4x + 7x^3 + 2x^4$  sera représenté par le tableau

1	4	0	7	2			
---	---	---	---	---	--	--	--

dont seuls les 5 premiers éléments sont significatifs.

En utilisant cette représentation d'un polynôme, on demande d'écrire une fonction booléenne qui calcule et renvoie le résultat de la multiplication d'un polynôme par le binôme  $ax + b$  (la fonction recevra les coefficients  $a$  et  $b$  du binôme en paramètre). La fonction retournera la valeur booléenne *false* si le degré du polynôme résultat dépasse *DEGRE\_MAX*, *true* sinon. La fonction devra envisager tous les cas possibles : polynôme nul, coefficient  $a$  du binôme nul, ... La fonction ne pourra modifier le tableau représentant le polynôme à multiplier qui lui est passé en paramètre.

### Exercice 5.

Un ensemble de  $n$  intervalles peut être représenté par un tableau de  $n$  lignes et deux colonnes. Chaque ligne représente un intervalle. La première colonne de la ligne contient la borne inférieure de l'intervalle, la seconde colonne sa borne supérieure. On supposera ici que ces bornes sont du type double, que les intervalles sont fermés à gauche et à droite et qu'ils sont triés par ordre croissant de leur borne inférieure.

#### Exemple .

L'ensemble d'intervalles  $\{[-1.5, 3.4], [2.3, 6.9], [3.5, 4.7], [7.2, 8.5], [8.5, 9.1]\}$  peut être représenté par le tableau suivant :

-1.5	3.4
2.3	6.9
3.5	4.7
7.2	8.5
8.5	9.1

En utilisant cette représentation d'un ensemble d'intervalles, on demande d'écrire une fonction qui pour un ensemble d'intervalles passé en paramètre, renvoie l'ensemble d'intervalles obtenu en réunissant les intervalles qui se chevauchent.

Pour l'exemple précédent, la fonction devra renvoyer le tableau suivant :

-1.5	6.9
7.2	9.1

Correspondant à l'ensemble d'intervalles  $\{[-1.5, 6.9], [7.2, 9.1]\}$ .

### Exercice 6.

Ecrire une fonction qui reçoit en paramètre un tableau  $t$  d'entiers strictement positifs de taille  $n$  et vérifie que chaque élément sauf le premier est la somme d'un certain nombre d'éléments consécutifs de  $t$  qui le précèdent immédiatement, c'est-à-dire:

$$\forall k, 1 \leq k \leq n-1 \quad \exists i, 0 \leq i \leq k-1 \quad \text{tel que} \quad t[i] + t[i+1] + \dots + t[k-1] = t[k]$$

La fonction retournera une valeur booléenne qui indiquera si la condition est satisfaite ou pas.

#### Exemple.

Pour le tableau suivant.

2	2	4	6	10	24	34	34
---	---	---	---	----	----	----	----

La fonction retournera *true* car

$$2 = 2$$

$$4 = 2 + 2$$

$$6 = 4 + 2$$

$$10 = 6 + 4$$

$$24 = 10 + 6 + 4 + 2 + 2$$

$$34 = 24 + 10$$

$$34 = 34$$

Pour le tableau suivant.

2	2	4	6	10	18	34	34
---	---	---	---	----	----	----	----

La fonction retournera *false* car la condition n'est pas remplie pour 18, par exemple.

$$18 \neq 10$$

$$18 \neq 10 + 6$$

$$18 \neq 10 + 6 + 4 > 18 \text{ (la vérification peut donc s'arrêter)}$$

### Exercice 7.

Ecrire une fonction qui remplace plusieurs occurrences consécutives d'une même valeur dans un tableau d'entiers par une seule occurrence de cette valeur. La fonction recevra comme paramètre le tableau et son nombre d'éléments significatifs. Celui-ci devra être modifié par la fonction pour refléter les suppressions éventuelles d'éléments.

#### Exemple.

Si le tableau passé en paramètre à la fonction contient les 10 valeurs significatives suivantes.

2	2	1	2	3	3	3	2	1	1	?	?
---	---	---	---	---	---	---	---	---	---	---	---

Au retour de la fonction le tableau contiendra les 5 valeurs suivantes.

2	1	2	3	2	1	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---	---	---

Les "?" représentent des valeurs non significatives.

### Exercice 8.

Tout nombre entier strictement positif peut s'écrire comme une somme de puissances distinctes de 2.

#### Exemples.

$$9 = 1 + 8$$

$$15 = 1 + 2 + 4 + 8$$

$$156 = 4 + 8 + 16 + 128$$

Pour décomposer un nombre en une somme de puissances distinctes de 2, on initialise la puissance de 2 courante à 1 ( $= 2^0$ ). Si le nombre est impair, la puissance courante de 2 apparaît dans la décomposition sinon elle n'apparaît pas. On divise ensuite le

nombre par 2 (division entière), on multiplie par 2 la puissance de 2 courante et on recommence avec la nouvelle valeur du nombre. On s'arrête quand le nombre devient nul.

On demande d'écrire une fonction qui place dans un tableau d'entiers, supposé suffisamment grand, les puissances de 2 qui apparaissent dans la décomposition d'un nombre passé en paramètre.

### Exercice 9.

Une permutation des entiers de 0 à  $n-1$ , où  $n$  est une constante, peut être représentée par un tableau de  $n$  entiers, l'élément  $i$  du tableau contenant l'image de  $i$  par la permutation.

#### Exemple.

Le tableau

5	0	2	4	3	1
---	---	---	---	---	---

représente la permutation telle que l'image de 0 est 5, l'image de 1 est 0, l'image de 2 est 2, l'image de 3 est 4, l'image de 4 est 3, l'image de 5 est 1.

Une autre façon de représenter une permutation  $p$  consiste à donner la liste de ses cycles. Un cycle est constitué par la suite des images successives d'un nombre par  $p, p^2, p^3, \dots$  jusqu'à ce qu'on retombe sur le nombre de départ. Dans les notations mathématiques, on place les éléments d'un cycle entre parenthèses. La représentation sous forme de cycles de la permutation de l'exemple précédent peut donc s'écrire :

$$(5\ 1\ 0)\ (2)\ (3\ 4)$$

qui exprime que l'image de 5 est 1, l'image de 1 est 0, l'image de 0 est 5 car 0 est le dernier élément du cycle donc son image est le premier élément du cycle, l'image de 2 est 2, l'image de 3 est 4 et l'image de 4 est 3.

Remarquons que cette représentation n'est pas unique :  $(2)\ (4\ 3)\ (0\ 5\ 1)$  constitue une autre représentation de la permutation de l'exemple.

Une représentation par cycles peut être encodée dans un tableau d'entiers en séparant les cycles par une valeur sentinelle,  $-1$  par exemple, et en signalant la fin de la représentation par deux valeurs sentinelles successives. Pour l'exemple précédent, on obtient le tableau :

5	1	0	-1	2	-1	3	4	-1	-1
---	---	---	----	---	----	---	---	----	----

On demande d'écrire une fonction qui renvoie la représentation sous forme de cycles d'une permutation donnée sous forme d'un tableau des images.

### Exercice 10.

Ecrire une fonction qui reçoit en paramètre un tableau d'entiers et y écrit en ordre croissant les facteurs premiers d'un entier  $n$  strictement supérieur à 1 passé en paramètre. Un même facteur premier pourra être répété dans le tableau. Le fonction recevra également en paramètre la taille du tableau. La fonction retournera comme résultat le nombre de diviseurs premiers trouvés. Si le nombre de diviseurs premiers dépasse la taille du tableau, la fonction remplit le tableau avec des facteurs premiers jusqu'à l'avant dernier élément puis écrit dans le dernier élément la partie de  $n$  qui n'a pu être factorisée et retourne 0.

#### Exemple 1.

Si  $MAX = 4$  et  $n = 20$  alors la fonction doit placer dans le tableau les valeurs suivantes.

2	2	5	
---	---	---	--

Et retourne la valeur 3.

En effet  $20 = 2.2.5$

#### Exemple 2.

Si  $MAX = 4$  et  $n = 300$  alors la fonction doit placer dans le tableau les valeurs suivantes.

2	2	3	25
---	---	---	----

Et retourne la valeur 0.

En effet  $300 = 2.2.3.25$

On pourrait décomposer 25 en 5.5 mais il n'y a plus de place dans le tableau pour placer ces deux nombres.

### Exercice 11.

Ecrire une fonction qui reçoit comme paramètres une matrice de booléens et ses dimensions. Cette fonction ajoutera à la matrice une colonne contenant des valeurs booléennes de sorte que le nombre de valeurs *true* dans chaque ligne (en tenant compte de la colonne ajoutée) soit pair.

Le paramètre correspondant à la matrice sera un tableau de booléens à deux dimensions dont le nombre de colonnes est une constante *MAX* définie dans le programme. On suppose que le nombre de colonnes de la matrice passée en paramètre à la fonction est toujours inférieur à *MAX*-1.

La fonction devra modifier les dimensions de la matrice pour refléter l'ajout d'une colonne.

#### Exemple.

Si la fonction reçoit le tableau de booléens suivant.

<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>

Elle lui ajoutera une 5ème colonne comme représenté ci-dessous.

<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	<b><i>false</i></b>	2 <i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<b><i>true</i></b>	2 <i>true</i>
<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<b><i>true</i></b>	4 <i>true</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<b><i>false</i></b>	0 <i>true</i>

On a indiqué après chaque ligne le nombre de *true* dans la ligne.

### Exercice 12.

Ecrire une fonction qui reçoit comme paramètre une liste d'entiers dans un tableau d'entiers et parcourt cette liste en comptant les répétitions successives des valeurs. La fonction retournera un tableau à deux colonnes dont la première contient les valeurs de la liste dans leur ordre d'apparition et la seconde leur nombre de répétitions successives.

#### Exemple.

Si la liste est la suivante

5	5	5	2	3	3	4	4	4	2	2
---	---	---	---	---	---	---	---	---	---	---

la fonction devra retourner comme résultat

5	3
2	1
3	2
4	3
2	2

car la liste contient 3 fois la valeur 5 puis 1 fois la valeur 2 puis 2 fois la valeur 3 ...

### Exercice 13.

Ecrire une fonction qui reçoit comme paramètres deux tableaux d'entiers et leurs nombres d'éléments. Baptisons *tab* le premier et *indices* le second. La fonction devra supprimer du tableau *tab* les éléments dont les indices sont dans le tableau *indices*. On suppose que les valeurs dans *indices* sont triées en ordre croissant et qu'elles sont strictement inférieures au nombre d'éléments du tableau *tab*. On devra traiter le cas où il n'y a pas d'élément à supprimer (nombre d'éléments du tableau *indices* nul).



**Exemple.**

Si le tableau *tab* contient les valeurs

4	2	5	4	1	9	1	15	3	2
---	---	---	---	---	---	---	----	---	---

et le tableau *indices* les valeurs

3	5	6	9
---	---	---	---

au retour de la fonction le tableau *tab* contiendra

4	2	5	1	15	3
---	---	---	---	----	---

**Exercice 14.**

Ecrire une fonction qui reçoit comme paramètres une suite finie d'entiers dans un tableau ainsi que son nombre d'éléments et retourne la longueur du plus long plateau de cette suite. Un plateau d'une suite est une sous-suite d'éléments consécutifs qui ont la même valeur.

**Exemple.**

Le plus long plateau de la suite

4 4 8 5 5 7 7 8 8 8 8 3 3 3 9 9 9 4

a pour longueur 4 (il y a deux plateaux de cette longueur dans cette suite, il sont soulignés dans l'exemple).

**Exercice 15.**

Un ensemble de  $n$  intervalles peut être représenté par un tableau de  $n$  lignes et deux colonnes, chaque ligne représentant un intervalle, la première colonne de la ligne contenant la borne inférieure de l'intervalle, la seconde colonne sa borne supérieure. On supposera ici que ces bornes sont du type double, que les intervalles sont fermés à gauche et à droite et qu'ils sont triés par ordre croissant de leur borne inférieure.

On demande d'écrire une fonction qui reçoit comme paramètre un ensemble d'intervalles représenté de cette façon, leur nombre et deux doubles  $a$  et  $b$  tels que  $a \leq b$ . La fonction retournera une valeur booléenne indiquant si l'intervalle  $[a, b]$  est inclus dans la réunion des intervalles de l'ensemble.

**Exemple.**

Si l'ensemble d'intervalles est  $\{[-1.5, 3.4], [2.3, 6.9], [3.5, 4.7], [7.2, 8.5], [8.5, 9.1]\}$  qui peut être représenté par le tableau suivant.

-1.5	3.4
2.3	6.9
3.5	4.7
7.2	8.5
8.5	9.1

Pour l'intervalle  $[a, b] = [4, 6]$  qui est inclus dans la réunion des intervalles de l'ensemble la fonction devra retourner *true*.

Pour l'intervalle  $[a, b] = [4, 8]$  qui n'est pas inclus dans la réunion des intervalles de l'ensemble la fonction devra retourner *false*.

**Exercice 16.**

Ecrire une fonction qui reçoit comme paramètre un entier positif ou nul  $n$  et remplit les  $n + 1$  premières lignes d'un tableau également passé en paramètre avec les éléments du triangle de Pascal.

**Rappel.**

Élément (ligne, colonne)	Valeur
( $i, 0$ )	1
( $i, i$ )	1
( $i, p$ )	Valeur de ( $i-1, p-1$ ) + valeur de ( $i-1, p$ ) pour $0 < p < i$

**Exemple.**

Si la fonction reçoit pour  $n$  la valeur 5, elle devra remplir le tableau de la façon suivante.

1	-	-	-	-	-	...
1	1	-	-	-	-	...
1	2	1	-	-	-	...
1	3	3	1	-	-	...
1	4	6	4	1	-	...
1	5	10	10	5	1	...

**Remarques.**

- Les caractères '-' dans le tableau de l'exemple indiquent que la fonction ne doit rien écrire dans ces cases du tableau.
- Le tableau passé comme paramètre à la fonction aura un nombre de colonnes égal à une constante  $MAX$  du programme et on supposera que le paramètre  $n$  sera toujours strictement plus petit que  $MAX$  de même qu'au nombre de lignes du tableau.

**Exercice 18.**

Ecrire une fonction qui initialise tous les éléments d'une matrice de nombres entiers et de taille  $n \times m$  à une valeur passée en paramètre.

**Exercice 19.**

Ecrire une fonction qui calcule la trace d'une matrice carrée de taille  $n$  passée en paramètre.

# Les chaînes de caractères.

---

## Exercice 1.

Ecrire une fonction booléenne qui détermine si deux chaînes de caractères sont égales sans distinguer minuscules et majuscules pour les caractères non accentués.

## Exercice 2.

Ecrire une fonction qui supprime les espaces superflus d'une phrase. La fonction recevra comme paramètre une chaîne de caractères contenant la phrase et retournera une chaîne de caractères contenant la même phrase mais dans laquelle on a supprimé les espaces de début et de fin et dont les mots sont séparés par un seul espace. On considère qu'un mot est un ensemble de caractères quelconques bordés soit par un espace, soit par le début ou la fin de la chaîne.

### Exemple.

Chaîne fournie à la fonction :

```
"  Cet  exercice  est  très  facile  "
```

Chaîne retournée par la fonction :

```
"Cet exercice est très facile"
```

## Exercice 3.

Ecrire une fonction *tousPalindromes* à valeur booléenne qui teste si le vecteur de caractères qui lui est transmis contient une chaîne de caractères composée uniquement de mots palindromes. On considère qu'un mot est un ensemble de caractères quelconques bordés soit par un espace, soit par le début ou la fin de la chaîne.

Pour rappel : un palindrome est un groupe de lettres qui peut être lu de gauche à droite ou de droite à gauche.

### Exemples.

- *tousPalindromes* de "SOS ELLE A 1 RADAR !!!" est vrai car "SOS", "ELLE", "A", "1", "RADAR" et "!!!" sont tous des mots palindromes.
- *tousPalindromes* de "ESOPE RESTE ICI ET SE REPOSE" est faux car, bien que la phrase soit globalement un palindrome, les mots qui la composent ne sont pas des palindromes.

## Exercice 4.

Ecrire une fonction qui reçoit comme paramètre une chaîne contenant une expression représentant une somme d'entiers positifs placée dans une chaîne de caractères et retourne la valeur de cette somme. L'expression est uniquement constituée des caractères représentant les chiffres des nombres et de l'opérateur "+". Il n'y a pas d'espace dans l'expression. On pourra supposer que l'expression est syntaxiquement correcte.

### Exemple.

La valeur de l'expression

```
"12+345+6789"
```

est 7146.

*Indication:* un nombre entier  $x$  dont l'écriture décimale est  $c_1c_2c_3\dots c_n$  (les  $c_i$  sont les chiffres du nombre) a pour valeur  $(\dots(((c_1*10)+c_2)*10)+c_3)*10\dots)+c_n$ . Cette valeur peut être donc être obtenue en effectuant:

$$x_1 = c_1$$

$$x_2 = x_1 * 10 + c_2$$

$$x_3 = x_2 * 10 + c_3$$

...

$$x = x_{n-1} * 10 + c_n$$

### Exercice 5.

Ecrire une fonction qui traduit une chaîne de caractères ne contenant que des lettres et des espaces en code Morse.

La fonction recevra comme paramètre un tableau de caractères de 26 lignes et 5 colonnes. Chaque ligne du tableau contiendra une chaîne de caractères composée uniquement de caractères "." (points) ou "-" (tiret) (qui sont les deux symboles du code Morse). La première chaîne du tableau contiendra la traduction en morse de la lettre "A", la deuxième la traduction de "B", ...

La traduction ne différenciera pas les minuscules des majuscules. Un espace sera traduit par un espace.

La fonction retournera la traduction sous la forme d'une chaîne de caractères.

#### Exemple.

La traduction en Morse de la chaîne "Le code MORSE" est

" .- . . . - . - . . . . . - . . . . . - . . . . . - . . . . . "

Le tableau suivant donne la traduction des lettres en Morse.

A	. -	N	- .
B	- . . .	O	- - - -
C	- . - .	P	. - - .
D	- . .	Q	- - . -
E	.	R	. - .
F	. . - .	S	. . . .
G	- - .	T	-
H	. . . .	U	. . -
I	. .	V	. . . -
J	. - - -	W	. - -
K	- . -	X	- . . -
L	. - . .	Y	- . - -
M	- -	Z	- - . .

### Exercice 6.

Ecrire une fonction "frequence" ayant les caractéristiques suivantes :

- elle reçoit en paramètre une chaîne de caractères,
- elle reçoit également un vecteur de 27 entiers,
- elle retourne un entier égal à la longueur de la chaîne contenue dans le vecteur de caractères,
- au retour, les valeurs contenues dans le vecteur d'entiers correspondent aux fréquences des différentes lettres de l'alphabet dans la chaîne de caractères.

#### Exemple .

Si le vecteur de caractères contient la chaîne : "Cet exercice est vraiment trop FACILE !".

Le résultat de la fonction sera 39 et le vecteur d'entiers au moment du retour sera :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
7	2	0	4	0	7	1	0	0	3	0	0	1	1	1	1	1	0	3	1	4	0	1	0	1	0	0

La première case du vecteur sert à comptabiliser tous les caractères qui ne sont ni minuscule, ni majuscule (espaces, ponctuation, caractères accentués, ...)

NB : on ne distingue pas les minuscules des majuscules.

### Exercice 7.

Ecrire une fonction qui reçoit comme paramètre une chaîne de caractères et écrit dans un tableau la longueur de chaque mot de cette chaîne.

On considère qu'un mot est un ensemble de caractères quelconques bordés soit par un espace, soit par le début ou la fin de la chaîne.

### Exemple.

Si la chaîne est

" Une chaîne de caractères "

Le tableau des longueurs de mots contiendra les valeurs

3	6	2	10
---	---	---	----

### Exercice 8.

On peut déterminer si un nombre est divisible par 9 par la méthode suivante. On part du premier chiffre ou de zéro si ce chiffre est 9. On ajoute le deuxième chiffre (s'il y en a un), si le résultat est supérieur ou égal à 9, on lui soustrait 9 sinon on ne fait rien. On répète ensuite la même opération pour les chiffres suivants. Le nombre est divisible par 9 si et seulement si le résultat final est nul.

Ecrire une fonction booléenne *estDivisiblePar9* qui indique si le nombre entier positif est divisible par 9 en mettant en œuvre la méthode décrite ci-dessus. La fonction recevra le nombre entier sous la forme d'une chaîne de caractères contenant son écriture décimale.

### Exemple.

Pour le nombre 78192 (passé en paramètre à la fonction sous la forme de la chaîne "78192") la fonction effectuera donc les opérations suivantes:

7 est différent de 9  
7 + 8 = 15, 15 est supérieur ou égal à 9 on lui soustrait 9 pour obtenir 6  
6 + 1 = 7, 7 est strictement inférieur à 9  
7 + 9 = 16, 16 est supérieur ou égal à 9 on lui soustrait 9 pour obtenir 7  
7 + 2 = 9, 9 est supérieur ou égal à 9 on lui soustrait 9 pour obtenir 0

Le résultat est nul donc 78192 est divisible par 9 ( $78192 = 9 \times 8688$ ).

### Exercice 9.

Ecrire une fonction qui reçoit comme paramètre une chaîne de caractères. La fonction devra parcourir cette chaîne de la gauche vers la droite en comptant le nombre de répétitions successives de chaque caractère et écrire ce nombre et le caractère dans une autre chaîne. Le nombre écrit dans la chaîne résultat sera limité à un seul chiffre. S'il y a plus de 9 caractères identiques qui se suivent, la fonction devra les écrire dans la chaîne résultat par paquet de 9 jusqu'à ce qu'il en reste 9 ou moins.

Exemple.

Si la chaîne de caractères passée à la fonction est la suivante.

"aaa3333bB\*\*\*\*\*zz"

La fonction devra retourner la chaîne suivante.

"3a431b1B9\*9\*3\*2z"

### Exercice 10.

Ecrire une fonction qui supprime d'une chaîne de caractères les répétitions de caractères identiques consécutifs. La fonction remplacera une suite de plusieurs caractères identiques par un seul de ces caractères. La fonction ne doit pas créer une nouvelle chaîne de caractères mais modifier la chaîne qui lui est transmise comme paramètre.

Exemple:

La fonction devra transformer la chaîne "aaabccaaabbaaa" en la chaîne "abcaba".

## La récursivité.

---

### Exercice 1.

Ecrire une fonction récursive qui calcule  $x^n$  où  $x$  est un réel et  $n$  un entier positif ou nul en utilisant le fait que  $x^0 = 1$  et  $x^{n+1} = x \cdot x^n$ .

### Exercice 2.

Ecrire une fonction récursive qui calcule  $x^n$  en utilisant les propriétés suivantes.

Si  $n$  est non nul et pair ( $n = 2m$  avec  $m > 0$ ) alors  $x^n = x^m \cdot x^m$ .

Si  $n$  est impair ( $n = 2m + 1$ ) alors  $x^n = x \cdot x^m \cdot x^m$ .

Si  $n$  est nul alors  $x^n = 1$ .

Evaluer le nombre de multiplications effectuées en fonction de  $n$ .

### Exercice 3.

Ecrire une fonction récursive qui écrit à l'écran la représentation d'un nombre entier positif donné en base  $b$  avec  $b$  compris entre 2 et 16.

### Exercice 4.

Ecrire une fonction récursive qui affiche à l'écran des entiers positifs lus au clavier dans l'ordre inverse où il sont entrés. La suite sera terminée par une valeur sentinelle négative.

### Exercice 5.

Ecrire une fonction récursive qui affiche les éléments d'un tableau d'entiers du premier au dernier. La fonction ne peut pas utiliser de boucle.

### Exercice 6.

Ecrivez une fonction récursive permettant de calculer le pgcd de deux nombres.

Pour rappel:  $\text{pgcd}(a,b) = \text{pgcd}(b, a \text{ modulo } b)$

### Exercice 7.

Le schéma de Horner pour calculer la valeur en  $x$  d'un polynôme de degré  $N$

$$a[0] + a[1]x + a[2]x^2 + \dots + a[n]x^n$$

est de la forme :

$$(\dots(((a[N])*x + a[N-1])*x + a[N-2])\dots)*x + a[1])*x + a[0]$$

Ecrivez une fonction récursive renvoyant la valeur du polynôme reçu pour la valeur de  $x$  donnée. Le prototype de la fonction sera le suivant.

```
double Horner (Polynome a, double x, int degre);
```

avec le type Polynome défini comme suit :

```
typedef int Polynome[MAX];
```

### Exercice 8.

Ecrire une fonction récursive qui affiche à l'écran tous les sous-ensembles de l'ensemble des entiers de 0 à  $n-1$  en utilisant le fait qu'un sous-ensemble de  $\{0, 1, \dots, n-1\}$  est soit un sous-ensemble de  $\{1, 2, \dots, n-1\}$  soit l'union de  $\{0\}$  et d'un sous-ensemble de  $\{1, 2, \dots, n-1\}$ .

### Exercice 9.

Ecrire une fonction récursive qui calcule la valeur (booléenne) d'une expression logique en notation préfixe donnée sous la forme d'une chaîne de caractères. Les valeurs logiques vrai et faux seront représentées par les caractères 'T' et 'F' respectivement. Les opérateurs logiques admis dans l'expression seront le "et", le "ou", et la négation représentés respectivement par les caractères '&', '|', et '!'.

Dans la notation préfixe, l'opérateur précède les opérandes, voici quelques exemples d'expressions en notation préfixe avec leur équivalent dans la notation habituelle (infixe) et leur valeur.

Notation préfixe	Notation infixe	Valeur
&TF	T&F	faux
FT	T F	vrai
& FT TF	(F T)&(T F)	vrai
&FT&  FFT	(F&T) ((!(F F))&T)	vrai

### Exercice 10.

Le coefficient binomial  $\binom{n}{p}$ , où  $0 \leq p \leq n$ , correspond au coefficient du terme  $x^p y^{n-p}$  dans le développement de  $(x+y)^n$ . La

valeur de ce coefficient est  $\frac{n!}{p!(n-p)!}$  et correspond à l'élément  $(n, p)$  du triangle de Pascal qui est calculée à partir des formules suivantes :

Elément	Valeur
$(n, 0)$	1
$(n, p)$	0 si $p > n$
$(n, p)$	valeur de $(n-1, p-1)$ + valeur de $(n-1, p)$ si $0 < p \leq n$

### Exemple.

	p=0	p=1	p=2	p=3
n=0	1	0	0	0
n=1	1	1	0	0
n=2	1	2	1	0
n=3	1	3	3	1

Ecrire une fonction récursive qui calcule la valeur du coefficient binomial  $\binom{n}{p}$  en utilisant les règles de calcul des éléments du triangle de Pascal.

### Exercice 11.

Ecrire une fonction récursive qui utilise l'algorithme d'Euclide pour calculer le pgcd de deux entiers non nuls. L'algorithme d'Euclide est basé sur les propriétés suivantes.

- $pgcd(a, b) = pgcd(b, a \% b)$  si  $b \neq 0$ .
- $pgcd(a, 0) = a$ .

Le théorème de Bezout affirme que pour deux entiers  $a, b$  avec  $a \neq 0$  ou  $b \neq 0$ , il existe deux entiers  $u$  et  $v$  tels que  $a.u + b.v = pgcd(a, b)$ . En d'autres termes, le pgcd de deux entiers peut s'écrire comme une combinaison linéaire à coefficients entiers de ces deux nombres. Il est facile de calculer les nombres  $u$  et  $v$  à partir de l'algorithme d'Euclide. Considérons le tableau suivant qui illustre le calcul du pgcd de deux entiers  $a$  et  $b$  par l'algorithme d'Euclide.

$\text{pgcd}(a, b) = \text{pgcd}(b, r_1)$	$a = bq_1 + r_1$	$r_1 = a - bq_1$
$\text{pgcd}(b, r_1) = \text{pgcd}(r_1, r_2)$	$b = r_1q_2 + r_2$	$r_2 = b - r_1q_2$
$\text{pgcd}(r_1, r_2) = \text{pgcd}(r_2, r_3)$	$r_1 = r_2q_3 + r_3$	$r_3 = r_1 - r_2q_3$
...	...	...
$\text{pgcd}(r_k, r_{k+1}) = \text{pgcd}(r_{k+1}, r_{k+2})$	$r_k = r_{k+1}q_{k+2} + r_{k+2}$	$r_{k+2} = r_k - r_{k+1}q_{k+2}$
...	...	...
$\text{pgcd}(r_{n-2}, r_{n-1}) = \text{pgcd}(r_{n-1}, r_n)$	$r_{n-2} = r_{n-1}q_n + r_n$	$r_n = r_{n-2} - r_{n-1}q_n$
$\text{pgcd}(r_{n-1}, r_n) = \text{pgcd}(r_n, 0) = r_n$	$r_{n-1} = r_nq_{n+1} + 0$	

A chaque étape, on calcule le reste de la division d'un nombre par un autre. La seconde colonne du tableau donne les relations qui lient les différents restes à chaque étape du calcul. Les nombres  $q_i$  et  $r_i$  sont les quotients et restes des divisions entières effectuées. Dans la troisième colonne, on a simplement réécrit différemment la colonne 2 pour mettre en évidence la méthode permettant de calculer les entiers  $u$  et  $v$  intervenant dans le théorème de Bezout. Cette colonne donne la valeur d'un reste  $r_i$  comme combinaison linéaire à coefficients entiers des restes  $r_{i-1}$  et  $r_{i-2}$ . En partant de la dernière ligne de la colonne 3,  $r_n = r_{n-2} - r_{n-1}q_n$ , si on remplace  $r_{n-1}$  par sa valeur en fonction de  $r_{n-2}$  et  $r_{n-3}$  ( $r_{n-1} = r_{n-3} - r_{n-2}q_{n-1}$ ) on obtient  $r_n = r_{n-2}(1 + q_{n-1}q_n) - r_{n-3}q_n$ . En remplaçant dans cette expression  $r_{n-2}$  par sa valeur donnée dans la troisième colonne du tableau et en continuant d'opérer ces substitutions jusqu'à arriver à la première ligne du tableau, on finit par obtenir la valeur de  $r_n$  comme combinaison linéaire à coefficients entiers de  $a$  et  $b$ . Observons les relations qui résultent d'une substitution d'un reste  $r_k$  par sa valeur donnée par la colonne 3. Supposons qu'on ait  $r_n = r_{k-1}u_i + r_k v_i$  si on remplace  $r_k$  par sa valeur  $r_k = r_{k-2} - r_{k-1}q_k$  on obtient  $r_n = r_{k-2}u_{i+1} + r_{k-1}v_{i+1}$  avec  $u_{i+1} = v_i$  et  $v_{i+1} = u_i - v_iq_k$  et  $u_0=1$  et  $v_0=0$  car  $r_n = r_n \cdot 1 + r_{n-1} \cdot 0$ .

A partir de ces relations, écrire une fonction  $\text{bezout}(a, b, u, v)$  qui retourne comme valeur le pgcd des entiers non nuls  $a$  et  $b$  et place dans  $u$  et  $v$  les nombres entiers tels que  $a.u + b.v = \text{pgcd}(a, b)$ .

L'algorithme permettant de calculer les nombres  $u$  et  $v$  s'appelle l'algorithme d'Euclide étendu et est utilisé en cryptographie.

### Exercice 12.

Ecrire une fonction récursive qui calcule la valeur d'une expression arithmétique entière représentée par une chaîne de caractères.

Les expressions considérées par la fonction ne comporteront que des nombres entiers positifs, les opérateurs  $+$ ,  $-$ ,  $*$ ,  $/$  (qui représente la division entière) et des parenthèses. Les opérations seront systématiquement entourées par des parenthèses. On pourra supposer que l'expression fournie à la fonction est correctement écrite.

#### Exemple.

Pour l'expression

$((23-13)*(17/2))+3$

La fonction retournera la valeur entière 83.

### Exercice 13.

Le jeu du baguenaudier est un casse-tête qui se joue avec un ensemble de  $n$  pions placés dans  $n$  cases numérotées de 1 à  $n$ . Le but du jeu est d'enlever tous les pions des cases en respectant les règles suivantes :

1. une case ne peut contenir qu'au plus un pion,
2. on peut toujours poser ou enlever un pion dans la case 1,
3. on peut poser ou enlever un pion de la case  $k$  ( $1 < k \leq n$ ) s'il y a un pion dans la case  $k-1$  et si les cases précédentes (1, 2, ...,  $k-2$ ) sont vides. En particulier on pourra retirer ou placer un pion dans la case 2 si la case 1 contient un pion.



**Exemple.**

case 1	case 2	case 3	opération
•	•	•	On enlève le pion de la case 1 (règle 2)
	•	•	On enlève le pion de la case 3 (règle 3)
	•		On pose un pion dans la case 1 (règle 2)
•	•		On enlève le pion de la case 2 (règle 3)
•			On enlève le pion de la case 1 (règle 2) et le problème est résolu

Une variante du jeu consiste à partir d'un ensemble de  $n$  cases vides et à les remplir en respectant les mêmes trois règles énoncées plus haut. On pourrait appeler ce jeu le baguenaudier inverse.

Il existe une stratégie qui permet de résoudre le problème du baguenaudier pour  $n$  quelconque. La marche à suivre est la suivante :

1	2	...	$n-2$	$n-1$	$n$	opérations
•	•	...	•	•	•	on joue au baguenaudier avec les cases de 1 à $n-2$
				•	•	une fois les cases de 1 à $n-2$ vidées, on retire le pion de la case $n$ (règle 3)
				•		on joue au baguenaudier inverse avec les cases de 1 à $n-2$
•	•	...	•	•		Une fois les cases de 1 à $n-2$ remplies, on recommence la première étape avec $n-1$ cases et ainsi de suite jusqu'à ce qu'il ne reste plus de pion

Ecrire une fonction récursive qui mette en pratique cette stratégie. La fonction recevra comme paramètre le nombre de cases du problème et devra afficher à l'écran la suite des opérations à effectuer pour vider ces cases de leur pion comme indiqué dans la dernière colonne de l'exemple.

*Indication* : écrire une fonction récursive qui résout le problème du baguenaudier inverse en s'inspirant de la stratégie donnée ci-dessus et utiliser une récursivité croisée.

**Exercice 14.**

Ecrire une fonction récursive qui affiche à l'écran tous les  $n$ -uplets d'entiers compris entre 1 et  $k$ . La fonction recevra comme paramètre les entiers  $n$  et  $k$ .

**Exemple.**

Si  $n=2$  et  $k=3$ , la fonction devra afficher

(1, 1) (2, 1) (3, 1)  
 (1, 2) (2, 2) (3, 2)  
 (1, 3) (2, 3) (3, 3)


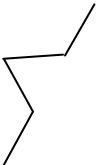

Ecrire cette même fonction de façon non récursive.

**Exercice 15.**

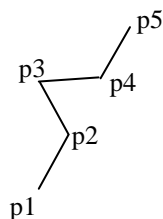
Ecrire une fonction qui calcule le déterminant d'une matrice  $n \times n$  passée en paramètre par la méthode des mineurs et cofacteurs.

**Exercice 16.**

La courbe de Von Koch, aussi appelée courbe du flocon de neige est une courbe fractale obtenue comme la limite d'un ensemble de courbes. La première courbe est un segment de droite. La deuxième courbe est obtenue en remplaçant ce segment par un motif constitué de plusieurs segments comme indiqué dans la figure ci-dessus. La troisième courbe est obtenue en remplaçant les segments de la deuxième par le même motif et ainsi de suite.

Courbe 1	Courbe 2	Courbe 3
		

Les 4 segments du motif ayant la même longueur, il est facile de trouver les coordonnées des extrémités de ces segments à partir des coordonnées des extrémités du segment d'origine.



Si  $p1(x1, y1)$  et  $p5(x5, y5)$  sont les extrémités du segment d'origine, alors les points  $p2(x2, y2)$ ,  $p3(x3, y3)$  et  $p4(x4, y4)$  sont donnés par :

$$x2 = x1 + \Delta x$$

$$y2 = y1 + \Delta y$$

$$x3 = (x1 + x5 - \sqrt{3}\Delta y) / 2$$

$$y3 = (y1 + y5 + \sqrt{3}\Delta x) / 2$$

$$x4 = x5 - \Delta x$$

$$y4 = y5 - \Delta y$$

avec

$$\Delta x = (x5 - x1) / 3$$

$$\Delta y = (y5 - y1) / 3$$

Ecrire une fonction récursive qui écrit dans un fichier les coordonnées des extrémités des segments pour une courbe de "profondeur"  $n$ , c'est-à-dire pour laquelle on a appliqué  $n$  fois le processus de remplacement des segments par les motifs à partir du segment d'origine. La fonction recevra donc comme paramètre les coordonnées du segment, la profondeur  $n$  et le fichier dans lequel elle doit écrire les coordonnées des points. Si on écrit sur chaque ligne du fichier les coordonnées  $x$  et  $y$  d'un point séparées par un ou plusieurs espaces, ce fichier peut être utilisé par l'utilitaire de tracé de courbe, *Gnuplot*, disponible sous Unix et Windows. La démarche à suivre est la suivante :

1. Créer un fichier baptisé "vonKoch" contenant la ligne suivante:

```
plot "vonKoch.dat" with lines
```

Où `vonKoch.dat` est le nom du fichier généré par le programme.

2. Dans une fenêtre de terminal, taper `gnuplot` pour lancer `gnuplot` puis `load "vonKoch"`.

L'adresse du site officiel de Gnuplot est

<http://www.gnuplot.info/>

## Exercice 17.

Ecrire une fonction récursive qui indique si deux nombres entiers positifs non nuls  $n$  et  $m$  sont premiers entre eux. On se basera sur les propriétés suivantes.

Si  $n$  ou  $m$  vaut 1 alors  $n$  et  $m$  sont premiers entre eux.

Si  $n = m \neq 1$  alors ils ne sont pas premiers entre eux.

Si  $n$  et  $m$  sont pairs ils ne sont pas premiers entre eux.

Si  $n$  est pair et  $m$  impair alors  $n$  et  $m$  sont premiers entre eux si et seulement si  $n/2$  et  $m$  le sont (idem en inversant les rôles de  $n$  et  $m$ ).

Si  $n$  et  $m$  sont impairs et  $n > m$  alors  $n$  et  $m$  sont premiers entre eux si et seulement si  $(n - m)/2$  et  $m$  le sont (idem en inversant les rôles de  $n$  et  $m$ ).

### Exercice 18.

Ecrire une fonction récursive qui affiche à l'écran toutes les suites strictement croissantes d'entiers compris entre 1 et  $k$ .

#### Exemple.

Si  $k=4$ , la fonction devra afficher

(1) (1, 2) (1, 2, 3) (1, 2, 3, 4) (1, 2, 4) (1, 3) (1, 3, 4) (1, 4) (2) (2, 3) (2, 3, 4) (2, 4) (3) (3, 4) (4)

Ecrire cette même fonction de façon non récursive.

### Exercice 19.

On souhaite étudier l'évolution d'une population de poules. On suppose que chaque poule pond chaque nuit un œuf unique. Un nombre fixé de jours après sa ponte, un œuf éclot et donne naissance à une poule qui la nuit même commence à pondre. On souhaite disposer de fonctions retournant le nombre de poules, le nombre d'œufs âgés de 0 jours, de 1 jour, ... après un nombre de jours donné; ces fonctions disposeront de paramètres donnant le nombre initial de poules et le nombre de jours entre la ponte d'un œuf et son éclosion.

*Indications:* déterminer le nombre de poules, d'œufs âgés de 0 jour, de 1 jours, ... un jour donné en fonction des ces mêmes nombres le jour précédent.

# Les pointeurs

---

## Exercice 1.

Dans le programme suivant, supprimer les lignes incorrectes et ajouter les instructions de libération de mémoire (`delete`) puis écrire les lignes affichées par le programme ainsi corrigé.

```
#include <iostream.h>

typedef int *IntPtr;
typedef IntPtr *IntPtrPtr;

main()
{
    IntPtr p, q;
    IntPtrPtr pp;
    int t[3] = {1, 2, 3};

    p = new int;
    *p = 7;
    q = 100; q = p;
    cout << *q << endl;
    q = new int;
    *q = 11;
    cout << *p << " " << *q << endl;
    pp = new int;
    pp = &q;
    cout << **pp << endl;
    pp = new IntPtr;
    *pp = new int;
    **pp = 20;
    *p = *q + **pp;
    cout << *p << endl;
    q = *pp;
    p = q;
    if (*p = **pp)
        cout << "*p = **pp" << endl;
    delete pp;
    cout << *q << endl;
    delete p;
    cout << *q << endl;
    p = t+1;
    *(++p) = 8;
    cout << t[2] << endl;
    cout << p[-1] << endl;
}
```

## Exercice 2.

Ecrire une fonction à laquelle on passe un tableau de  $n$  pointeurs sur des float et qui retourne un nouveau tableau contenant ces  $n$  valeurs float.

## Exercice 3.

Ecrire une fonction qui reçoit comme paramètres un tableau d'entiers et sa taille et retourne deux pointeurs, l'un vers l'élément contenant la plus petite valeur du tableau, l'autre vers l'élément contenant la plus grande.

#### Exercice 4.

Ecrire une fonction qui reçoit comme paramètres un tableau d'entiers et sa taille et retourne un tableau de pointeurs sur des entiers. Un élément du tableau de pointeurs pointera vers un entier égal à l'élément correspondant dans le tableau d'entiers. Si deux éléments du tableau d'entiers contiennent la même valeur, les deux pointeurs correspondants du tableau de pointeurs pointeront vers la même variable dynamique.

#### Exercice 5.

Ecrire une fonction qui reçoit comme paramètres une chaîne de caractères et deux entiers  $i$  et  $j$  et retourne la sous-chaîne de caractères contenant les caractères entre les positions  $i$  et  $j$ . Si  $i$  est négatif, si  $j$  est plus grand que la longueur de la chaîne ou si  $j < i$ , la fonction retournera le pointeur NULL. La chaîne retournée sera allouée dans la fonction.

#### Exercice 6.

Ecrire une fonction qui crée et retourne un tableau de chaînes de caractères dont chaque ligne contient un mot d'une chaîne de caractères passée en paramètre. On considère qu'un mot est une suite quelconque de caractères délimités par un espace, le début ou la fin de la phrase. Le tableau des mots sera défini comme un tableau de pointeurs vers des chaînes de caractères (char \*\*). On utilisera le pointeur NULL pour marquer la fin du tableau.

#### Exercice 7.

Ecrire une fonction qui crée et retourne une chaîne de caractères contenant l'écriture décimale (c-à-d la suite des chiffres décimaux) d'un entier passé en paramètre.

#### Exercice 8.

Ecrire une fonction qui retourne une "matrice triangulaire inférieure" à  $n$  lignes représentant un triangle de Pascal. La première ligne de la matrice aura un seul élément, la deuxième deux, ...

Pour rappel, l'élément  $(n, p)$  du triangle de Pascal correspond au coefficient du terme  $x^p y^{n-p}$  dans le développement de  $(x+y)^n$  et peut être calculée en utilisant les propriétés suivantes :

Elément	Valeur
$(n, 0)$	1
$(n, p)$	0 si $p > n$
$(n, p)$	Valeur de $(n-1, p-1)$ + valeur de $(n-1, p)$ si $0 < p \leq n$

#### Exemple.

	p=0	p=1	p=2	p=3
n=0	1			
n=1	1	1		
n=2	1	2	1	
n=3	1	3	3	1

#### Exercice 9.

Implémenter une fonction d'intégration de fonction par les sommes de Riemann. Utiliser la formule:

$$\int_a^b f(x) dx = \left( \sum_{i=1}^n f(a + ih) \right) h$$

avec  $h=(b-a)/n$ . La fonction Riemann( ) est telle que son premier argument est un pointeur sur la fonction  $f$  que l'on intègre. Dans cet exercice, intégrer la fonction

$$f(x) = x^3$$

#### Exercice 10.

Ecrire une fonction à laquelle on passe un tableau de  $n$  float et qui renvoie un pointeur sur le plus grand de ces  $n$  float.

#### Exercice 11.

On peut calculer une approximation de la dérivée d'un fonction  $f$  en utilisant

$$f'(x) \approx \frac{f(x + \varepsilon) - f(x)}{\varepsilon}$$

avec  $\varepsilon$  petit.

En appliquant la même méthode à l'approximation de la dérivée, on peut obtenir une approximation de la dérivée seconde et ainsi de suite.

Ecrire une fonction récursive qui reçoit comme paramètres une fonction  $f$  à un paramètre double retournant un double, un double  $x$  et un entier  $n$  et retourne la dérivée  $n^{\text{ième}}$  de  $f$  en  $x$  en appliquant plusieurs fois la méthode décrite ci-dessus.  $\varepsilon$  sera une constante définie dans le programme.

### Exercice 12.

Ecrire une fonction qui reçoit comme paramètres une fonction  $f$  à un paramètre double et à valeur double et 3 doubles  $a$ ,  $b$  et  $y$ . On pourra supposer que  $f$  est croissante et que  $f(a) \leq y \leq f(b)$ . La fonction retournera un double  $x$  tel que  $|f^{-1}(y) - x| \leq \varepsilon$  où  $\varepsilon$  est une constante double définie dans la fonction. En d'autres termes,  $x$  est une approximation de l'image inverse de  $y$ ,  $f^{-1}(y)$ , par la fonction. Pour calculer  $x$ , on utilisera la méthode dichotomique suivante :

1. Calculer  $m = (a + b) / 2$ .
2. Si  $f(m) > y$  alors  $b = m$  sinon  $a = m$ .
3. Recommencer en 1 tant que  $b - a > \varepsilon$ .

Retourner  $m$  comme approximation de  $f^{-1}(y)$ .

### Exercice 13.

On peut représenter un polynôme à coefficients réels à l'aide d'un tableau de doubles en plaçant dans les éléments du tableau les coefficients du polynôme par ordre décroissant du degré.

#### Exemple.

Le polynôme de degré 5 suivant:

$$4x^5 + 6x^3 + 3x^2 + 7x + 4$$

sera représenté par le tableau à 6 éléments suivants:

4	0	6	3	7	7
---	---	---	---	---	---

En utilisant cette représentation, on demande d'écrire une fonction qui calcule de façon approchée les racines d'un polynôme dans un intervalle donné.

Pour calculer les racines d'un polynôme non constant on demande d'utiliser la démarche suivante. Si  $a$  et  $b$  sont les bornes de l'intervalle dans lequel on cherche les racines, on calcule les racines de la dérivée du polynôme dans l'intervalle  $[a, b]$ . Si on note  $x_1, x_2, \dots, x_k$  ces racines,  $x_0 = a$  et  $x_{k+1} = b$ , la dérivée du polynôme ne s'annule pas dans les intervalles  $]x_i, x_{i+1}[$  avec  $i = 0, 1, 2, \dots, k$  et sur l'intervalle  $[x_i, x_{i+1}]$  le polynôme est donc strictement croissant ou strictement décroissant. Si les valeurs du polynôme en  $x_i$  et  $x_{i+1}$  sont non nulles et de même signe, le polynôme ne s'annule pas sur cet intervalle, si elles sont non nulles et de signes opposés le polynôme a une racine unique à l'intérieur de cet intervalle. Si le polynôme s'annule sur une des extrémités de l'intervalle, cette extrémité est une racine du polynôme et le polynôme ne s'annule pas à l'intérieur de l'intervalle. Pour trouver la racine d'un polynôme sur un intervalle  $[x_i, x_{i+1}]$  où les valeurs en  $x_i$  et  $x_{i+1}$  sont non nulles et de signes opposés, on peut mettre en œuvre un algorithme inspiré de la recherche dichotomique d'une valeur dans un tableau croissant ou décroissant. On calculera cette racine avec une précision  $\varepsilon$  fixée dans le programme.

La fonction devra renvoyer les racines approchées dans un tableau de doubles alloué dynamiquement dans la fonction.

#### Remarque.

Du fait de la précision limitée des calculs et des approximations effectuées, il est possible d'obtenir avec cette méthode dans certains cas marginaux des racines approchées qui ne correspondent pas à des racines du polynôme ou au contraire de manquer certaines racines.

On demande également d'utiliser cette fonction dans un programme qui proposera à l'utilisateur d'entrer le degré, les coefficients d'un polynôme et un intervalle et affichera à l'écran les racines de ce polynôme comprises dans cet intervalle.

## Les classes.

---

### Exercice 1.

Implémenter une classe *Person*. Chaque objet de cette classe représente un être humain. Les données membres sont le nom de la personne, son année de naissance et son année de décès. Définir un constructeur, un destructeur, des fonctions d'accès et une fonction d'affichage.

On vous propose de travailler non pas avec une chaîne de caractères pour le nom, mais avec un pointeur sur une chaîne de caractères allouée dynamiquement. La taille de la chaîne de caractères vers laquelle le pointeur pointe est déterminée durant la phase de construction. Un paramètre du constructeur sera un pointeur vers une chaîne de caractère contenant le nom à copier dans les données de la personne. Donc, on détermine sa taille, ensuite seulement on alloue dynamiquement l'espace mémoire nécessaire.

Ajoutez une fonction membre *age* qui prend l'année courante en paramètre et qui retourne l'âge de la personne.

### Exercice 2.

Ecrire une classe *Point* pour représenter des points en deux dimensions  $(x,y)$ . Cette classe devra contenir un constructeur sans paramètre qui initialise les coordonnées à 0 et un constructeur à 2 paramètres pour initialiser  $x$  et  $y$  ainsi que des méthodes permettant de changer les coordonnées du point et de connaître leur valeur. La classe contiendra également deux méthodes retournant les coordonnées polaires du point (module et argument) et enfin une fonction qui donne la distance du point à un autre point.

Ecrire une classe *Cercle* pour représenter des cercles. Un cercle sera caractérisé par son centre (du type *Point* vu ci-dessus) et son rayon. La classe *Cercle* devra contenir une méthode retournant l'aire du cercle.

En utilisant les deux classes ci-dessus, écrire une fonction qui détermine si deux cercles sont disjoints.

### Exercice 3.

Implémenter une classe *Cours* contenant le code d'un cours de la section MATH en polytech, le nombre d'heures de cours. Implémenter une classe *Etudiant* contenant le numéro de matricule de l'étudiant, le nombre maximum de cours qu'ils peut suivre, le nombre de cours suivis par l'étudiant ainsi que la liste de ces cours (utiliser un pointeur sur un tableau de pointeurs vers des objets *Cours*), la liste des notes obtenues pour chacun de ces cours (utiliser un pointeur sur un tableau d'entiers). Définissez également la fonction *moyenne()* renvoyant la note moyenne de l'étudiant, et une fonction *totalHeures()* renvoyant le nombre total d'heures de cours suivies par l'étudiant.

### Exercice 4.

Implémenter une classe monome contenant un entier représentant le degré du monôme, et un double représentant le coefficient du monôme. Un polynôme sera donc un array de  $n$  monômes. Dans ce tableau, le degré égal à  $-1$  est une valeur sentinelle.

Ecrire une fonction *eval* qui retourne la valeur d'un polynôme  $P$  en  $x$ .

Ecrire une fonction *somme* qui calcule la somme de deux polynômes et retourne cette somme dans un nouveau polynôme. On suppose que les monômes sont classés par ordre croissant de degré dans chacun des polynômes.

Ecrire une fonction *div\_par\_X\_moins\_1* qui calcule le quotient et le reste de la division d'un polynôme par  $x-a$  par la méthode de Horner ( $a$  est donc un paramètre de la fonction). On suppose que les polynômes utilisés sont de degrés  $> 1$ . Utiliser une fonction **ecrire** pour afficher le polynôme quotient à l'écran.

### Exercice 5.

Supprimer les lignes erronées du programme suivant puis écrire les lignes affichées par le programme corrigé.

```
#include <iostream.h>

class C
{
public:
    C(int x);
    ~C();
    C(const C& c);
    void f();
private:
    int v;
};

C::C(int x)
{
    C c(1);
    cout << "Constructeur " << x << endl;
    v = x;
}
```

```

C::~C()
{
    cout << "Destructeur " << v << endl;
}

C::C(const C& c)
{
    cout << "Constructeur de copie " << c.v << endl;
    v = 10 * c.v;
}

void C::f()
{
    cout << "Méthode f " << v << endl;
}

void f(C c)
{
    C a(2), *b = new C(3);
    cout << "Fonction f" << endl;
    c.f();
}

void g(C &c)
{
    cout << "Fonction g" << endl;
}

main()
{
    C c1;
    C c2(1), c3 = c2;
    C *p;

    cout << c2.v << endl;
    cout << endl;
    f(c2);
    cout << endl;
    g(c2);
    cout << endl;
    p = new C(4);
    cout << endl;
    f(*p);
    cout << endl;
    c2 = *p;
    cout << endl;
    delete p;
    cout << endl;
}

```

## Exercice 6.

On demande d'implémenter une classe permettant de représenter un ensemble d'entiers. Cette classe contiendra un constructeur sans paramètre qui crée un ensemble vide et un constructeur qui reçoit comme paramètre un tableau d'entiers ainsi que sa taille et qui crée un ensemble d'entiers contenant les éléments de ce tableau. La classe contiendra également des méthodes permettant d'ajouter un entier à l'ensemble (s'il ne lui appartient pas déjà), de supprimer un entier de l'ensemble et de tester l'appartenance d'un entier à l'ensemble. On demande aussi d'écrire une méthode qui fournit le nombre d'éléments de l'ensemble et une autre qui retourne un tableau d'entiers alloué dynamiquement contenant exactement les éléments de l'ensemble (si l'ensemble est vide cette fonction doit retourner NULL). Ne pas oublier d'implémenter le destructeur et le constructeur de copie.

On impose ici l'utilisation d'un tableau d'entiers comme donnée membre de la classe pour contenir les éléments de l'ensemble. A tout instant, la taille du tableau sera égale au nombre d'éléments de l'ensemble, il faudra donc trouver une méthode pour faire varier dynamiquement la taille de ce tableau. En particulier, quand l'ensemble est vide, aucun tableau ne doit être alloué.

Dans un premier temps on utilisera un tableau non trié pour contenir les éléments de l'ensemble. On implémentera ensuite la même classe avec un tableau trié en essayant d'optimiser les méthodes. On discutera de l'efficacité relative des deux implémentations.

En utilisant la classe décrite ci-dessus, écrire une fonction qui affiche à l'écran les nombres premiers entre 2 et  $n$  en utilisant la technique du crible d'Eratosthène. Cette technique consiste à créer l'ensemble des nombres de 2 à  $n$  puis à supprimer tous les multiples de 2 de cet ensemble. On passe alors à 3 dont on supprime tous les multiples. Puisque 4 n'est plus dans l'ensemble, on passe à 5 dont on supprime tous les multiples, puis à 7 puisque 6 a été supprimé, puis à 11 car 8, 9 et 10 ont été supprimés, ...



## Exercice 7.

On demande d'implémenter une classe permettant de représenter sans gaspillage de mémoire des matrices carrées symétriques. Une matrice symétrique est connue si on connaît les éléments de sa partie triangulaire inférieure c-à-d les éléments d'indices  $(i, j)$  tels que  $i \geq j$ . La classe contiendra un constructeur permettant de créer une matrice de dimension donnée dont tous les éléments sont nuls. Une méthode devra permettre d'obtenir la valeur d'un élément et une autre de la changer. On prévoira aussi une méthode qui donne la dimension de la matrice. On n'oubliera pas le destructeur et le constructeur de copie.

## Exercice 8.

Supprimer les lignes erronées du programme suivant puis écrire les lignes affichées par le programme corrigé.

```
#include <iostream.h>

class Test
{
public:
    Test(int id);
    Test(const Test &t);
    ~Test();
    void afficher();
private:
    int identificateur;
    int *noCopie;
};

Test::Test(int id)
{
    identificateur = id;
    noCopie = new int(0);
    cout << "Création de" << endl;
    cout << " Identificateur: " << identificateur << endl;
    cout << " Copie: " << *noCopie << endl;
}

Test::Test(const Test &t)
{
    cout << "Copie de" << endl;
    cout << " Identificateur: " << t.identificateur << endl;
    cout << " Copie: " << *t.noCopie << endl;
    identificateur = t.identificateur;
    noCopie = new int(*t.noCopie + 1);
}

Test::~~Test()
{
    cout << "Destruction de" << endl;
    cout << " Identificateur: " << identificateur << endl;
    cout << " Copie: " << *noCopie << endl;
    delete noCopie;
}

void Test::afficher()
{
    cout << "Affichage" << endl;
    cout << " Identificateur: " << identificateur << endl;
    cout << " Copie: " << *noCopie << endl;
}

void f1(Test t)
{
    t.afficher();
}

void f2(Test &t)
{
    t.afficher();
}

void f3(Test *t)
{
    t->afficher();
}

Test f4()
{
    Test x(3);
    return x;
}
```

```

main()
{
    Test t1(1);
    Test t2(2);
    t2 = t1;
    Test t3;
    Test t4 = t1;
    f1(t4);
    f2(t4);
    Test *t5 = new Test(t4);
    f3(t5);
    delete t5;
    Test t6(2);
    t6 = f4();
    t6.afficher();
    cout << "****" << endl;
    Test t7 = f4();
    t7.afficher();
    cout << "****" << endl;
    Test tab[100];
}

```

## Exercice 9.

Ecrire une classe "Chrono" permettant de représenter des chronomètres avec les fonctionnalités suivantes:

- Lors de la création d'un chronomètre, on peut indiquer via un paramètre booléen s'il démarre automatiquement (*true*) ou pas (*false*). Par défaut le chronomètre est créé à l'arrêt.
- On met (ou remet s'il est arrêté) en marche le chronomètre par la méthode *start*. Cette méthode n'a pas d'effet si le chronomètre est déjà en marche.
- On arrête le chronomètre à l'aide de la méthode *stop*. Cette méthode n'a pas d'effet si le chronomètre est déjà arrêté.
- On remet à zéro le chronomètre par la méthode *reset*. Cette méthode peut être appelée que le chronomètre soit en marche ou arrêté. S'il est en marche, le chronomètre sera remis à zéro et arrêté.
- On obtient le temps (en secondes) de marche du chronomètre par la méthode *getTime*.

L'appel de la fonction *time* de la librairie standard *time.h* avec comme paramètre 0 (c'est-à-dire l'appel *time(0)*) retourne le nombre de secondes écoulées depuis le 1er janvier 1970, 0 heures G.M.T.

## Exercice 10.

Le langage C++ ne permet pas de retourner comme valeur d'une fonction une fonction créée dynamiquement. Il n'est, par exemple, pas possible d'écrire une fonction qui reçoit comme paramètre deux fonctions à un paramètre double et à valeur double et retourne une fonction du même type qui serait la somme de ces deux fonctions.

On peut toutefois simuler cette possibilité en utilisant les classes et en écrivant une classe *Fonction* permettant de représenter des fonctions, de les additionner, soustraire, multiplier et diviser. La classe "Fonction" aura un constructeur ayant pour paramètre une fonction à un paramètre double et à valeur double. La classe disposera d'une méthode *val* ayant comme paramètre un double. Si on construit un objet de la classe *Fonction* (appelons de tels objets des *objets fonctionnels*) en passant au constructeur une fonction *f* et si on utilise la méthode *val* sur cet objet avec comme paramètre *x* alors cette méthode retournera  $f(x)$ .

### Exemple.

```

Fonction of(sin);
cout << of.val(x); //affichera sin(x)

```

La classe disposera d'un autre constructeur à 3 paramètres: un caractère représentant un opérateur ('+', '-', '\*' ou '/') et deux objets fonctionnels. L'objet construit sera l'objet fonctionnel correspondant à la fonction obtenue en appliquant l'opérateur au deux fonctions représentées par les deux objets fonctionnels passés en paramètres.

### Exemple.

```

Fonction of1(sin);
Fonction of2(cos);
Fonction of3('+', of1, of2);
cout << of3.val(x); //affichera sin(x) + cos(x)

```

## Exercice 11.

On vous demande de concevoir un système informatique ayant une structure orientée objet permettant de s'attaquer au problème suivant. Le secrétariat de la faculté doit gérer les cours donnés en candidature. Chaque cours possède un nom, un identificateur entier, un professeur titulaire, un local, ainsi que différentes séries d'élèves qui suivent le cours. Un cours peut être suivi par quatre séries d'élèves au maximum. Chaque série est composée d'un identificateur entier, du nombre d'élèves qui la composent (maximum 20) ainsi que des coordonnées de chacun de ces élèves. Le local est identifiable par un bâtiment (une lettre), un étage et un numéro de local. Toutes les personnes sont identifiées par leur nom et un identificateur entier. Le système informatique devra dans un premier temps être capable de constituer les données (créer un local, un cours, une personne, ...). Il devra ensuite pouvoir attacher ces données les unes aux autres (ajouter un élève dans une série, ajouter une série à un cours, modifier le titulaire d'un cours, ...). Finalement il faudra avoir la possibilité de tirer les renseignements utiles de cette base de données (« quels sont les élèves qui suivent le cours MATH305 ? », « l'élève 212 est-il dans la série 2 ? », « le cours INFO161 a-t-il un local ? », ...). De plus, il serait souhaitable que les données relatives à chaque élément puissent être affichées à l'écran.

Quelles sont les classes qui interviennent dans le problème, quels sont leurs attributs, quelles sont les méthodes qui y sont attachées ? Rédigez le code C++ correspondant, ainsi qu'un petit programme permettant de créer quelques données et de tester les différentes méthodes.

Comment feriez-vous pour que le système attribue lui-même les identificateurs entiers pour chacun des objets ? Attention, deux objets appartenant à la même classe doivent avoir des identificateurs différents !

Quelles modifications faudrait-il apporter pour pouvoir obtenir facilement la liste des cours qui se donnent dans un local ?

Indépendamment de l'exercice 3, quelles modifications faudrait-il apporter pour qu'un même cours puisse être donné dans des locaux différents, selon la série ?