Exercices

Dans la suite, ce que j'appelle procédure est une fonction qui renvoie `void`.

Toutes les fonctions des exercices doivent être testées à l'intérieur du `main()`.

1 Introduction au langage C

▶ **Exercice 1** *Écrire une fonction qui renvoie la moyenne de deux flottants.*

▶ **Exercice 2** *Lorsqu'on affecte un nombre à virgule flottante à une variable de type* `int`*, la partie décimale est automatiquement tronquée.*

1. *Écrire une fonction qui renvoie la partie entière d'un flottant.*
2. *Écrire une fonction qui renvoie la valeur arrondie à l'unité d'un flottant.*

▶ **Exercice 3** *Compléter la procédure suivante pour intervertir le contenu des deux variables* `a` *et* `b` *sans utiliser de constantes numériques :*

```
1 void permuter() {
2   int a, b;
3   a = 5;
4   b = 2;
5   printf("a contient %d et b contient %d\n", a, b);
6   ...
7   printf("a contient %d et b contient %d\n", a, b);
8 }
```

▶ **Exercice 4** *Écrire une fonction récursive* `int factorielle(int n)` *qui, étant donné un paramètre n, renvoie le produit des entiers de* `1` *à* `n` *:*

$$1 \times 2 \times 3 \times ... \times n$$

Testez votre fonction grâce à des assertions du type `assert( factorielle(4) == 24 );`

▶ **Exercice 5** *Écrire une fonction récursive* `int sommecarre(int n)` *qui, étant donné un paramètre n, renvoie la somme des carrés des entiers de* `0` *à* `n` *:*

$$0^2 + 1^2 + 2^2 + ... + n^2$$

▶ **Exercice 6** *Écrire une fonction récursive* `sommecube` *qui, étant donné un paramètre n, renvoie la somme des cubes des entiers de* `0` *à* `n`*.*

$$0^3 + 1^3 + 2^3 + ... + n^3$$

▶ **Exercice 7** *Écrire une fonction récursive* `puiss` *qui prend en paramètres deux entiers n et p et qui renvoie n^p.*

▶ **Exercice 8** *Écrire une fonction récursive* `sommepuiss` *qui renvoie la somme*

$$0^p + 1^p + 2^p + ... + n^p$$

où n et p seront des paramètres de la fonction.

▶ **Exercice 9** *Écrire une fonction récursive* `sommepairs` *qui, étant donné un paramètre n, renvoie la somme des nombres pairs jusqu'à $2n$:*

$$0 + 2 + 4 + 6 + ... + 2n$$

▶ **Exercice 10** *Écrire une fonction récursive* `fibo` *qui, étant donné un paramètre n, renvoie la valeur au rang n de la suite de Fibonacci.*

On rappelle que la suite de Fibonacci est la suite (a_n) définie par :

$$\begin{cases} a_0 = 0 \\ a_1 = 1 \\ a_n = a_{n-1} + a_{n-2} \text{ si } n \geqslant 2 \end{cases}$$

▶ **Exercice 11** *Écrire les fonctions récursives précédentes en récursif terminal.*

2 Les boucles

▶ **Exercice 12** *Refaire les exercices 4 à 10 avec des boucles.*

▶ **Exercice 13** *Écrire une fonction qui permet de faire une division entière sans utiliser l'opérateur* `/`*. Pour cela, on va déterminer combien de fois on peut soustraire le deuxième nombre du premier.*

► **Exercice 14** *Écrire une procédure avec une seule boucle, qui permet d'afficher le motif d'étoiles suivant (avec un espace entre chaque étoile) :*

```
* * * * * * * *
```

► **Exercice 15** *Écrire une procédure avec deux boucles l'une à la suite de l'autre, qui permet d'afficher le motif d'étoiles suivant :*

```
* * * * * * * *
 * * * * * * * *
```

► **Exercice 16** *En réutilisant éventuellement les procédures précédentes, écrire une nouvelle procédure qui permet d'afficher le motif d'étoiles suivant :*

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

► **Exercice 17** *Écrire une procédure avec deux boucles imbriquées, qui permet d'afficher le motif d'étoiles suivant :*

```
*
**
***
****
*****
```

Indication : *le ligne 1 contient une étoile, la ligne 2 contient deux étoiles...*

► **Exercice 18** *Écrire une procédure avec deux boucles imbriquées, qui permet d'afficher le motif d'étoiles suivant :*

```
*****
****
***
**
*
```

► **Exercice 19** *Écrire une procédure avec deux boucles imbriquées, qui permet d'afficher le motif d'étoiles suivant :*

```
    *
   **
  ***
 ****
*****
```

3 Les tableaux

► **Exercice 20** *Écrire une procédure qui affiche le contenu d'un tableau de* `n` *cases (le tableau et sa taille* `n` *seront passés en paramètres).*

► **Exercice 21** *Écrire une procédure qui multiplie par deux chaque élément d'un tableau de* `n` *cases.*

► **Exercice 22** *Écrire une procédure qui divise par deux tous les nombres pairs d'un tableau de* `n` *cases.*

► **Exercice 23** *Écrire une fonction qui renvoie le produit de tous les éléments d'un tableau de* `n` *cases.*

► **Exercice 24** *Écrire une fonction qui renvoie la moyenne d'un tableau de* `n` *cases.*

► **Exercice 25** *Écrire une fonction qui renvoie le minimum d'un tableau de* `n` *cases.*

► **Exercice 26** *Écrire une fonction qui renvoie le minimum d'un tableau de* `n` *cases, en ignorant la partie du tableau située avant la case numéro* `d`.

► **Exercice 27**

1. *Refaire les deux exercices précédents en renvoyant le numéro de la case du minimum à la place du minimum lui-même.*
2. *Écrire une fonction qui trie le tableau dans l'ordre décroissant, puis une fonction qui trie le tableau dans l'ordre croissant. (On utilisera les deux fonctions de la première question en mettant, à chaque étape, le minimum à sa place).*

► **Exercice 28** *Écrire une fonction qui renverse les éléments d'un tableau de* `n` *cases en utilisant une boucle. Par exemple, à partir du tableau :*

1	5	7	12	42

on doit obtenir à la fin :

42	12	7	5	1

4 Les chaînes de caractères

► **Exercice 29** *Écrire une fonction qui permet de tester si deux chaînes de caractères sont identiques. Cette fonction renverra 0 si c'est faux, et 1 si c'est vrai.*

► **Exercice 30** *Écrire une fonction qui copie une chaîne de caractères dans une autre.*

► **Exercice 31** *Écrire une fonction qui modifie une chaîne* `ch` *en remplaçant chaque lettre* `'e'` *ou* `'E'` *par un espace. Par exemple, si* `ch` *contient initialement* `"hello"`, *à la fin il contiendra* `"h llo"`.

► **Exercice 32** *Écrire une fonction qui extrait une sous-chaîne située au milieu d'une chaîne. Pour cela, on dispose de trois données en entrée :*

— *une chaîne* `ch1` ;
— *un entier* `n` *qui contient la longueur de la sous-chaîne voulue ;*
— *un entier* `d` *qui contient le numéro de la case à partir de laquelle on va copier la sous-chaîne.*

Par exemple, si `ch1` *contient* `"hello"`, `n` *contient 3 et* `p` *contient 1, alors la sous-chaîne au milieu sera* `"ell"`.

► **Exercice 33** *Le type* `char` *qui représente les caractères est en fait de type entier. Chaque caractère est représenté par son code ASCII. Les codes ASCII de* `'A'` *et* `'a'` *sont respectivement 65 et 97. L'écart entre les deux est donc de 32. On retrouve le même écart entre toutes les lettres de l'alphabet. Donc, si je fais* `'A'+ 32`, *j'obtiens* `'a'`.

Écrire une fonction qui met en majuscules une chaîne `ch` *ne contenant initialement que des lettres minuscules.*

► **Exercice 34** *Améliorer la fonction précédente pour qu'elle marche pour tout type de chaîne. Pour cela, on indique que les codes ASCII des minuscules sont tous compris entre 97 et 122.*

► **Exercice 35** *Écrire une fonction qui teste si une chaîne est un palindrôme (c'est-à-dire une suite de caractères qui se lit de la même manière de gauche à droite, et de droite à gauche.*

► **Exercice 36** *Écrire une fonction qui modifie une chaîne* `ch` *en faisant disparaître tous les* `'e'` *et les* `'E'`. *Par exemple, si* `ch` *contient initialement* `"hello"`, *à la fin il contiendra* `"hllo"`.

► **Exercice 37** *Écrire une fonction qui renvoie le nombre de mots d'une chaîne dans le cas où deux mots sont séparés par un et un seul espace.*

► **Exercice 38** *Écrire une fonction qui renvoie le nombre de lignes d'une chaîne.*

► **Exercice 39** *Écrire une fonction qui renvoie le nombre de mots d'une chaîne dans le cas où deux mots sont séparés par un ou plusieurs espaces, par un ou plusieurs retours à la ligne, ou toute cominaison d'espaces et de retours à la ligne.*

5 Les enregistrements

► **Exercice 40** *Quel type entier est le mieux adapté pour représenter :*

1. *le nombre d'étudiants présents au cours de C ?*
2. *le nombre d'habitants sur la Terre ?*
3. *un mois ?*
4. *une année ?*
5. *une note sur 20 ?*
6. *la température extérieure en dégrés celsius ?*
7. *une distance en km entre deux villes de France ?*

► **Exercice 41** *Écrire un enregistrement pour représenter les données suivantes :*

1. *un point de l'espace ;*
2. *une fraction ;*
3. *un nombre complexe ;*
4. *une date ;*
5. *un horaire de train ;*
6. *un vêtement dans un magasin dont on connait la référence (entier) et le prix ;*
7. *un polynôme grâce à son degré (entier au plus égal à 10) et ses coefficients (réels).*

► **Exercice 42** *Définir un type* `tvecteur` *permettant de représenter des vecteurs à deux dimensions (abscisse et ordonnée), puis écrire les fonctions qui permettent :*

1. *d'afficher un vecteur*
2. *de tester si un vecteur est nul*
3. *de calculer l'opposé d'un vecteur*
4. *de calculer la somme de deux vecteurs*
5. *de calculer le produit scalaire de deux vecteurs*
6. *de calculer le déterminant de deux vecteurs*
7. *de calculer la norme d'un vecteur.*

On testera ensuite toutes ces fonctions dans une procédure `testevecteurs`.

► **Exercice 43** *Définir un type* `tfraction` *permettant de représenter des fractions de nombres entiers (numérateur et dénominateur), puis écrire les fonctions qui permettent :*

1. *d'afficher une fraction*
2. *de tester si une fraction est nulle*
3. *de calculer l'opposé d'une fraction*
4. *de calculer l'inverse d'une fraction*
5. *de calculer le produit de deux fractions*
6. *de calculer la somme de deux fractions*
7. *de simplifier une fraction*
8. *de calculer une valeur flottante approchée d'une fraction.*

On testera ensuite toutes ces fonctions dans une procédure `testefractions`.

6 Pointeurs et allocation dynamique de la mémoire

► **Exercice 44** *Dans la fonction* `main()`, *déclarez les variables suivantes :*

```
1  int i;
2  double x;
3  int tab[10];
4  char c;
5  float y;
```

1. *À la suite de ces déclaration, écrire les instructions qui permettent d'afficher l'adresse de chaque variable.*
2. *En compilant et en exécutant ce programme, comparer les adresses les unes par rapport aux autres.*
3. *Peut-on en déduire l'espace occupé par certaines de ces variables ?*

► **Exercice 45**

1. *Écrire une fonction qui remplace la valeur de l'entier passé en argument par son carré.*
2. *Écrire une fonction qui échange les valeurs de deux entiers passés en argument.*
3. *Écrire une fonction qui échange les valeurs de deux pointeurs passés en argument.*

► **Exercice 46** *Écrire une fonction qui renvoie le minimum d'un tableau, et qui, grâce à un passage par pointeur, permettra aussi de connaître le numéro de la case du minimum. Cette fonction ne fait évidemment aucun affichage.*

► **Exercice 47** *Écrire un programme qui affiche le contenu du tableau :*

```
1  int tab[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Ce programme utilisera des pointeurs et aucun crochet. Pour cela on utilisera un pointeur qui pointera initialement sur la première case du tableau, puis on utilisera l'arithmétique des pointeurs pour parcourir tout le tableau.

► **Exercice 48** *En utilisant un tableau dynamique, écrire un programme qui :*

1. *demande à l'utilisateur combien il veut saisir de nombres ;*
2. *crée le tableau à la taille désirée, puis demande de saisir les nombres ;*
3. *affiche les nombres entrés par l'utilisateur dans l'ordre inverse.*

► **Exercice 49**

1. *Déclarer un type* `struct fraction` *permettant de représenter des fractions d'entiers.*
2. *Créer un tableau dynamique de 10 fractions et le remplir comme vous le souhaitez.*
3. *Afficher le tableau.*

► **Exercice 50** *On peut voir un tableau à deux dimensions comme un tableau de tableaux.*

Écrire un programme qui :

1. *crée un tableau dynamique à deux dimensions de 5 lignes et 10 colonnes ;*
2. *le remplit avec les nombres de 1 à 50 ;*
3. *affiche ce tableau.*

7 Ligne de commande / Entrées/sorties

► **Exercice 51** *Écrire le programme* `renverse` *qui, après compilation, fonctionne ainsi (copie du terminal) :*

```
mamachine $ gcc -W -Wall renverse.c -o renverse ↵
mamachine $ ./renverse salut tout le monde ↵
monde le tout salut
mamachine $ ▮
```

► **Exercice 52** *Écrire le programme* `miroir` *qui, après compilation, fonctionne ainsi (copie du terminal) :*

```
mamachine $ gcc -W -Wall miroir.c -o miroir ↵
mamachine $ ./miroir salut tout le monde ↵
ednom el tuot tulas
mamachine $ ▮
```

► **Exercice 53** *Écrire le programme* `begayer` *qui, après compilation, fonctionne ainsi (copie du terminal) :*

```
mamachine $ gcc -W -Wall begayer.c -o begayer ↵
mamachine $ ./begayer salut tout le monde ↵
salut salut tout tout le le monde monde
mamachine $ ▮
```

► **Exercice 54** *Écrire un programme qui compte le nombre de caractères, le nombre de mots et le nombre de lignes d'un fichier passé en paramètre.*

► **Exercice 55** *Écrire le programme* `head` *qui permet d'afficher les n premières lignes d'un fichier. Par exemple, la commande pour afficher les 10 premières lignes du fichier* `toto.txt` *sera :*

```
mamachine $ ./head -n 10 toto.txt ↵
```

► **Exercice 56**

1. *Créer un fichier* `data.txt` *contenant une dizaine de nombres flottants séparés par des retours à la ligne.*
2. *Écrire le programme* `moyenne` *qui permet de faire la moyenne des nombres contenus dans* `data.txt`