



R1.08 Informatique 2022 – 2023	TD n°2 Programmation modulaire	 
-----------------------------------	-----------------------------------	---

Exercice 1. Prototypes

La déclaration du prototype d'une fonction permet en langage C au programme principal de connaître les entrées-sortie de celle-ci avant qu'elle ne soit appelée.

a) Récupérer le programme td2exo16.c sur Moodle. Compilez-le. Corrigez l'erreur puis exécutez-le.

Donner sous forme graphique puis écrire en langage C lorsque c'est possible les prototypes des sous-programmes suivants :

- b) Module qui renvoie un entier au hasard entre 1 et 1000
- c) Module qui renvoie un entier au hasard entre 2 nombres passés comme arguments
- d) Module qui calcule l'accélération d'une particule animée d'un mouvement circulaire uniforme en fonction de sa vitesse et du rayon de sa trajectoire
- e) Module qui renvoie le cube d'un nombre entier
- f) Module qui affiche le cube d'un nombre entier
- g) Module qui remplace un nombre entier par son cube
- h) Module qui calcule le module et l'argument d'un nombre complexe en fonction de ses parties réelles et imaginaires
- i) Module qui, en fonction d'une distance exprimée en pieds et pouces, la convertit en mètres
- j) Module qui, en fonction d'une distance exprimée en mètres, la convertit en pieds et pouces

Exercice 2. Appel des fonctions

Pour qu'une fonction soit exécutée, elle doit être appelée par le programme principal. Il faut pour cela donner un ou des valeurs aux entrées de la fonction, et la sortie de la fonction peut être manipulée comme n'importe quelle valeur du même type. Il est évidemment possible d'appeler plusieurs fois la même fonction dans le programme principal, les entrées 'et donc la sortie) pouvant bien sûr changer d'un appel à l'autre.



- ✓ Récupérer le programme td2exo17.c sur Moodle
- ✓ Complétez ce programme pour que celui calcule et affiche également :
 - Le cosinus du double de x ,
 - Le double du cos de x ,
 - La tangente de x ,
 - La tangente de x mais en n'utilisant que les fonction cos et sin,
 - L'exponentielle du cosinus de $5x + 3$,
 - $\sqrt[4]{3x + 4}$

Exercice 3. Instruction return

L'instruction `return` permet de préciser la valeur en sortie de la fonction (on parle de valeur renvoyée). Elle est donc nécessaire à la fin de la définition de la fonction (à l'exception des fonctions de type `void`, ne renvoyant pas de valeur). Par ailleurs, l'instruction `return` a également pour effet de terminer immédiatement l'exécution de la fonction et de revenir au programme principal.

- ✓ Récupérer le programme td2exo18.c sur Moodle qui permet de calculer la résistance équivalente à 2 résistances données en parallèle.
- ✓ Compilez et exécutez le programme puis corrigez l'instruction `return` pour que le résultat correct s'affiche ;

- ✓ En ne faisant que déplacer l'instruction `return`, sans modifier ni supprimer aucune autre ligne, faites en sorte que ce soit la résistance équivalente en série qui soit calculée.
- ✓ Modifiez une dernière fois la fonction pour qu'elle ne contienne plus qu'une seule instruction

R1.08 Informatique 2022 – 2023	TD n°3 Structures conditionnelles	 
-----------------------------------	--------------------------------------	---

Exercice 4. Structure if

- ✓ Récupérer le programme `td3exo19.c` sur Moodle. Compilez-le et testez-le.
- ✓ Modifiez-le pour que le nom de la planète Venus soit affiché uniquement si la valeur entrée par l'utilisateur est strictement positive ;
- ✓ Modifier de nouveau le programme pour que Venus et Terre ne soient affichés que si `a` est strictement positif
- ✓ Modifier encore le programme : Venus et Terre seront affichés si `a` est strictement positif, Mars et Jupiter si `a` est négatif ou nul.

Exercice 5. Positif ou négatif ?

- Récupérer le programme `td3exo20.c` sur Moodle. Le compléter pour qu'il informe si le produit des 2 nombres entrés est positif ou négatif (on ne prendra pas en compte le cas où le produit est nul).
- Reprendre cet exercice mais cette fois-ci **sans** calculer le produit des deux nombres. Le programme testera les 4 cas possibles.
- Reprendre l'exercice, toujours sans calculer le produit mais en ne conservant qu'un seul test (ce dernier contiendra alors des opérateurs relationnels)

Exercice 6. Inscription sportive

Récupérer le programme `td3exo21.c` sur Moodle. Ecrire la fonction `Sport` qui, en fonction de l'âge d'un enfant, affiche sa catégorie :

- « Poussin » de 6 à 7 ans,
- « Pupille » de 8 à 9 ans,
- « Minime » de 10 à 11 ans,
- « Cadet » après 12 ans



Exercice 7. Opérateurs logiques

- Récupérer le programme `td3exo22a.c` sur Moodle
 - ✓ Compiler et tester le programme en rentrant des valeurs différentes pour `a`. Que constatez-vous ? (Pour information, la réponse à la grande question sur la vie, l'univers et le reste est bien 42). Pour vous aider à comprendre les erreurs contenues dans le code, vous pouvez vous aider des avertissements (warnings) émis par le compilateur et vous pouvez également afficher la valeur de `a` à la fin du programme.
 - ✓ Corriger le programme pour qu'il fonctionne correctement
- Récupérer le programme `td3exo22b.c` sur Moodle.
 - ✓ Essayez de deviner ce qui va être affiché à l'écran

- ✓ Compilez et exécutez le programme pour comparer avec ce que vous aviez prédit à la question précédente.

Exercice 8. Expressions booléennes

- ✓ Récupérer le programme td3exo23.c
- ✓ Exécutez-le en entrant des valeurs différentes pour a et b.
- ✓ Que font les 2 fonctions ? Renommez-les pour que le nom corresponde à leur usage.
- ✓ Réécrivez-les en utilisant une structure `if` classique

R1.08 Informatique 2022 – 2023	TD n°4 Structures répétitives	 
-----------------------------------	----------------------------------	---

Exercice 9. Saisie valide – Boucle `do ... while`

La structure `do ... while` permet de répéter l'instruction ou le bloc d'instructions situé entre le `do` et le `while` tant que la condition spécifiée au niveau du `while` est vraie.

- ✓ Récupérez le fichier td4ex24.c sur Moodle. L'objectif est de demander à l'utilisateur de rentrer un nombre compris entre 1 et 20 et de répéter cette demande jusqu'à ce que la réponse convienne. Compilez et corrigez ce programme pour que le programme exécutable soit généré.
- ✓ Testez ce programme et identifiez le dysfonctionnement. En utilisant simplement l'opérateur « non logique », corrigez le problème.
- ✓ Ajoutez une nouvelle variable entière `i` qui permet de savoir combien de nombres auront été saisis avant qu'une valeur correcte soit entrée. Comment se nomme ce type de variable ?

Exercice 10. Tirages aléatoires – Boucle `while`

La structure `while` permet de répéter l'instruction ou le bloc d'instructions situé juste après le `while` tant que la condition est vraie

- ✓ Récupérez le fichier td4ex25.c sur Moodle. Celui-ci doit générer et afficher des nombres aléatoires compris entre 1 et une valeur maximale NMAX (fixée à 20 par une définition de constante symbolique en début de programme) jusqu'à ce qu'une certaine valeur fixée par l'utilisateur soit obtenue.
- ✓ Compilez et testez ce programme sans le modifier. Que constatez-vous ?
- ✓ Modifiez le programme pour qu'il fonctionne correctement.
- ✓ Ajoutez un compteur `i` permettant de savoir combien de tirages ont été réalisées avant d'obtenir la valeur ciblée

Exercice 11. Table de multiplication – Boucle `for`

La structure `for` permet de répéter l'instruction ou le bloc d'instructions situé juste après le `for` un nombre de fois déterminé à l'avance. Elle utilise pour cela un compteur `i` qui s'incrémente à chaque tour de boucle.

- ✓ Récupérez le fichier td4ex26.c sur Moodle
- ✓ Complétez ce programme en plaçant une boucle `for` de façon à écrire la table de multiplication du nombre saisi par l'utilisateur, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de : 7

```
-----  
7 x 1 = 7  
7 x 2 = 14  
7 x 3 = 21  
...  
7 x 10 = 70
```

Exercice 12. Somme et moyenne

Reprenez l'exercice 25 et complétez le programme de façon à ce qu'il affiche également la somme et la moyenne des nombres aléatoires qui ont été générés. Pour cela :

- vous créez une variable entière `somme` que vous initialiserez à 0 et une variable réelle `moyenne` ;
- dans la boucle, vous incrémenterez la variable `somme` de la valeur aléatoire qui vient d'être générée ;
- après la fin de la boucle, vous diviserez la somme par le nombre total de valeurs générées pour obtenir la moyenne (vous penserez à forcer la division réelle pour que la valeur obtenue soit bien correcte).



Exercice 13. Structures imbriquées

- a) Récupérez le fichier `td4ex28.c` sur Moodle. Ce programme est censé afficher le nombre de tours de circuits faits, ainsi que, à chaque tour le nombre de kilomètres parcourus selon le modèle ci-dessous :

```
vous allez faire 5 tours de circuit  
chaque tour fait 3 km  
  
tour 1  
    km 1 distance totale 1  
    km 2 distance totale 2  
    km 3 distance totale 3  
tour 2  
    km 1 distance totale 4  
    km 2 distance totale 5  
    km 3 distance totale 6  
tour 3  
    km 1 distance totale 7  
    km 2 distance totale 8  
    km 3 distance totale 9  
tour 4  
    km 1 distance totale 10  
    km 2 distance totale 11  
    km 3 distance totale 12  
tour 5  
    km 1 distance totale 13  
    km 2 distance totale 14  
    km 3 distance totale 15
```

- ✓ Compilez et exécutez ce programme sans le modifier. Que constatez-vous ?
- ✓ Ajoutez une paire d'accolades de façon à avoir 2 boucles imbriquées à la place des 2 boucles successives.
- ✓ Retestez le programme. Identifiez le dysfonctionnement.
- ✓ Modifiez la longueur d'un tour pour qu'il fasse maintenant 3 km au lieu de 8. Identifiez le dysfonctionnement
- ✓ Corrigez le programme en utilisant un indice **différent** pour chaque boucle.
- ✓ Rajoutez un nouveau compteur pour pouvoir afficher la distance totale parcourue tous les km

- b) Reprenez le programme de l'exercice 24, en ajoutant un message « Trop petit ! » si le nombre saisi est inférieur à 1 et « Trop grand ! » s'il est supérieur à 20
- ✓ Pour commencer, vous ajouterez une structure conditionnelle à l'intérieur de la boucle `do ... while`
 - ✓ Vous remplacerez ensuite la boucle `do ... while` par une boucle `while` infinie, la sortie de boucle se faisant alors au niveau de la structure conditionnelle grâce à une instruction `break` bien placée.

R1.08 Informatique 2022 – 2023	TD n°5 Les tableaux	 
-----------------------------------	------------------------	---

Exercice 14. Manipulation d'un tableau

- ✓ Récupérez le fichier `td5ex29.c` sur Moodle. Ce programme est censé remplir un tableau de 8 cases, où la case d'indice i contient $i!$ (on rappelle $i! = 1 \times 2 \times 3 \times \dots \times (i-1) \times i$ avec $0! = 1$)
- ✓ Compilez et corrigez uniquement les erreurs de compilation : définissez une constante symbolique N pour la taille du tableau et déclarez la variable T
- ✓ Testez-le et modifiez la ligne permettant d'afficher le tableau en la remplaçant par une boucle pour afficher le tableau élément par élément
- ✓ Modifiez ensuite le programme pour le rendre plus efficace en supprimant les deux boucles imbriquées utilisées pour remplir le tableau. On pourra remarquer que $i! = (i-1)! \times i$
- ✓ Modifiez une dernière fois le programme pour aller jusqu'à 15 ! Que constatez-vous ?

Exercice 15. Taille d'un tableau

- ✓ Récupérez le fichier `td5ex30.c` sur Moodle. Ce programme doit laisser l'utilisateur choisir la taille du tableau puis le remplir avec des valeurs au hasard entre 1 et N_{MAX} (où N_{MAX} est une constante symbolique définie à 12 en début de programme).
- ✓ Compilez et testez le programme. Que constatez-vous ?

Lors de la déclaration d'un tableau, la taille de celui-ci doit être connue et donnée sous la forme d'une constante numérique ou symbolique.

- ✓ Modifiez le programme pour que la taille du tableau soit donnée par une constante symbolique N que vous définirez à 10. Testez cette nouvelle version du programme pour différentes valeurs de n . que constatez-vous ?
- ✓ Rajoutez une boucle autour de la lecture de n pour s'assurer que n reste inférieur ou égal à N .

Exercice 16. Tableaux et fonctions

- ✓ Récupérez le fichier `td5exo31.c` sur Moodle. Ce programme doit permettre de remplir un tableau de taille choisie par l'utilisateur avec des nombres aléatoires, de l'afficher puis d'afficher sa moyenne ainsi que sa valeur minimale.
- ✓ Compilez et testez ce programme
- ✓ Corrigez l'erreur au niveau du calcul de la moyenne pour que le résultat soit correct.

Lorsque l'on passe un tableau en argument d'une fonction, deux choix sont possibles. Soit (comme pour la fonction `MinTab`), on précise la taille du tableau, ce qui permet de ne passer que le tableau en argument mais dans ce cas la fonction ne fonctionnera que pour des tableaux de cette taille précise. Soit (comme pour

la fonction `RempTab`) on ne précise pas la taille du tableau mais on est alors contraint de rajouter un deuxième argument qui correspond à la taille du tableau manipulé lors de l'appel de la fonction.

- ✓ Modifiez le prototype de la fonction `MinTab` sur le modèle de la fonction `RempTab` de façon à ce que le calcul du minimum soit correct.
- ✓ Créez une fonction `MoyTab` qui calcule la moyenne d'un tableau passé en argument. Dans le programme principal, appelez cette fonction pour faire le calcul, seul l'affichage restera dans le `main`.
- ✓ Modifiez la fonction `MinTab` de façon à ce qu'elle renvoie l'indice de la case contenant le minimum plutôt que sa valeur. Dans le programme principal, modifiez l'affichage pour afficher l'indice **et** la valeur du minimum
- ✓ Rajoutez une boucle autour de la lecture de `n` pour s'assurer que `n` reste inférieur ou égal à `N`.

Exercice 17. Chaines des caractères

Une chaine de caractère est un tableau de caractère qui comporte le caractère spécial de fin de chaine `'\0'`. Tous les caractères situés après celui-ci seront ignorés par les fonctions gérant les chaines de caractères.

- ✓ Récupérez le fichier `td5exo33.c` sur Moodle.
- ✓ Compilez-le et corrigez l'erreur pour pouvoir l'exécuter. Vous pourrez initialiser la chaine message soit lors de sa déclaration, soit après en utilisant la fonction `strcpy` (pour *string copy*).
- ✓ Testez le programme
- ✓ Modifiez la lecture de la chaine à la ligne 27 pour que la totalité du texte saisi par l'utilisateur soit lue, y compris ce qui suit l'espace. Vous pourrez remplacer l'appel de la fonction `scanf` par la fonction `gets` (*get string*).
- ✓ Ajoutez une instruction à la ligne 26 pour afficher la chaine de caractères saisie par l'utilisateur, donc en s'arrêtant au caractère de fin de chaine sans atteindre la fin du tableau (vous pourrez pour cela utiliser la fonction `puts` (*put string*) ou la fonction `printf` avec le format `"%s"`).
- ✓ Ajoutez une « vraie » condition dans le `if` de la ligne 37 pour afficher le nombre de caractères dans la chaine, le même que celui indiqué par la fonction `strlen` (*string length*)
- ✓ Ajoutez une instruction ligne 44, avant le `printf` final pour que ce dernier permette d'afficher le message « Bienvenue a » suivi du texte saisi par l'utilisateur. Vous pourrez utiliser la fonction `strcat` (*string concatenation*) qui permet d'accoler 2 chaine de caractères

Exercice 18. Permutation d'un tableau

- ✓ Récupérez le programme `td5exo33.c` sur Moodle. Ce programme doit remplir un tableau de taille choisie par l'utilisateur avec des valeurs aléatoires, l'afficher, permuter les cases (la 1^{ère} avec la dernière, la 2^{ème} avec l'avant-dernière, ...) puis le réafficher.
- ✓ Dans la fonction `PermTab`, ajoutez les instructions permettant de permuter à chaque tour de boucle la case n° `i` avec la case n° `(n-1) - i`. Testez le programme, que constatez-vous ?
- ✓ Modifiez de nouveau la fonction `PermTab` pour qu'elle fonctionne correctement.
- ✓ Modifiez votre programme pour que le tableau soit rempli avec des valeurs réelles aléatoires comprises entre 0 et `NMAX`. Vous pourrez pour cela utiliser la fonction `RandVal` et vous ferez en sorte de n'afficher qu'un seul chiffre après la virgule lorsque vous affichez le tableau.

Exercice 19. Opérateurs d'incrémentation

En plus des opérateurs arithmétiques classiques, on peut définir en C deux opérateurs unaires (ils ne s'appliquent que sur une seule opérande) d'incrémentation : '++' et '--'. Ils servent à incrémenter ou décrémenter l'opérande de 1. Ainsi l'instruction "i++" ou "++i" est strictement équivalente à l'instruction "i = i + 1". De même l'instruction "i--" ou "--i" est strictement équivalente à l'instruction "i = i - 1". La particularité de cet opérateur est qu'il est parfois utilisé à l'intérieur d'une autre expression. Il est alors important de se rappeler qu'il possède non seulement une valeur mais qu'il réalise de surcroît une action. Si l'opérateur est placé après l'opérande (i.e. "i++"), c'est la valeur avant incrémentation qui est utilisée dans le reste de l'expression, si l'opérateur est placé avant l'opérande (i.e. "++i"), c'est la valeur après incrémentation qui est utilisée.

Application : soient les variables

```
int i = 6, j = 7, p 10;
```

Donner la valeur de i, j et n après les expressions suivantes

```
n = 3 * i++ - j;          n = 3 * ++i - j;
n = (i++ >= j) ;          n = (++i >= j) ;
```

Exercice 20. Opérateur séquentiel

En C, la notion d'expression est beaucoup plus large que dans la plupart des autres langages (par exemple à travers les opérateurs d'incrémentation et d'affectation). L'opérateur séquentiel permet d'élargir encore ces possibilités en permettant d'effectuer plusieurs opérations en une seule instruction. Le symbole représentant cet opérateur est la virgule (','). Sous sa forme générale, il permet de séparer plusieurs opérations qui seront effectuées séquentiellement, de la gauche vers la droite.

Exemple :

`a*b , j=i+1` (ici la valeur de `a*b` sera déterminée, bien que dans ce cas elle ne soit affectée nulle part et que cette opération soit inutile, puis la valeur de `i+1` est affectée à `j`).

Si la valeur de sortie de l'instruction est utilisée (par exemple à l'intérieur d'un `if`), c'est le résultat de l'expression la plus à droite qui sera utilisée.

Cet opérateur est couramment utilisé dans les structures conditionnelles et répétitives pour effectuer plusieurs calculs au sein de l'instruction.

Exemple :

```
if (c=a*b , c<a)
for (i=0, a=0; i<10, i++)
```

Cet opérateur peut se révéler très puissant mais à utiliser avec parcimonie au risque de rendre le programme rapidement illisible.

Voici une série d'exemples d'utilisation de cet opérateur, vous proposerez une écriture n'utilisant pas l'opérateur séquentiel et pour les trois derniers vous donnerez le résultat de l'expression concernée.

```
if (i++,j>0)
do ; while (printf( "entrez un nombre positif"), scanf("%d",&n), n<0) ;
while (printf( "entrez un nombre positif"), scanf("%d",&n), n<0) ;
for(i=0,a=1;i<10;a=2*a,i++) ;
for(i=0;a=1,i<10;a=2*a,i++) ;
for(i=0;i++,i<5;printf("i = %d",i)) ;
```

Exercice 21. Opérateurs d'affectation abrégés

En programmation, on utilise couramment des opérations du type :

variable = variable opérateur expression

pour remplacer la valeur contenue dans **variable** par une nouvelle valeur calculée à partir de l'ancienne valeur, par exemple :

i = i + 5;

i = i * 4;

En C, il est possible d'écrire les instructions de ce type de façon abrégée sous la forme:

variable opérateur = expression

Par exemple, les instructions précédentes peuvent s'écrire :

i+=5;

i*=4;

Ceci est possible pour tous les opérateurs arithmétiques (+, -, *, /, %).

On considère deux variables entières i=7 et j=5.

Indiquez la valeur de i et j après chaque instruction (prises séparément) :

```
i/=2;
i%=j-3;
i--=(2*j++-3);
i-=2*++j-3;
j*=(i++-j)/3;
j*=(++i-j)/3;
i+=(i+j)>=10;
i+=i++;
i+=++i;
i+=(3*++j-j++);
i+=(-j+++3*++j);
i++=6;
```