

Contrôle terminal de Programmation en C avancée

Mercredi 6 janvier 2021 de 9h à 11h

Aucun document n'est autorisé.
Les calculatrices, ordinateurs, et téléphones portables sont interdits.

Exercice 1. Importance de l'indentation, malgré tout

0.75 + 1.25 = 2 pts

Contrairement à certains autres langages, l'indentation d'un programme C n'a pas d'impact sur le résultat de son exécution. Mais elle a une importance pour sa lisibilité.

- 1. Que fait le programme ci-dessous ?

```
int main(void) {float ct, cc, note; scanf("%f", &ct); scanf("%f", &cc); note = .5f * cc + .5f * ct; printf("Resultat=%f\n", note); if (note < 10) {printf("... non validee\n");} else {printf("... validee\n");}}
```

- 2. Écrire un programme qui lit une valeur entière au clavier, puis qui indique si elle est positive, négative, ou nulle (en distinguant les 3 cas).

Exercice 2. Une boucle à un point

1 pt

- 1. Écrire un programme qui lit une valeur entière $C \geq 1$, puis qui calcule de manière itérative et affiche la plus grande puissance de 2 inférieure ou égale à C .

Exercice 3. Une drôle de pyramide

1 + 1.5 = 2.5 pts

Étant donné le couple de valeurs (h, c) , avec $h, c \geq 1$, on souhaite dessiner une pyramide de la forme suivante pour $(h, c) = (3, 2)$, par exemple.

```
      XX
     XX
    XX00XX
   XX00XX
  XX00XX00XX
 XX00XX00XX
```

h= hauteur de la pyramide
c= taille d'un cube

- 1. Écrire une fonction qui prend en paramètre une caractère k et un entier $n \geq 0$, et qui affiche une ligne de n caractères k , sans retour à la ligne.
► 2. Écrire un programme qui lit h et c , et qui affiche la pyramide correspondante.

Exercice 4. À l'aide!

1.5 pts

Un étudiant a besoin d'aide. Il souhaite écrire une fonction qui, étant donnés deux entiers en paramètre, calcule puis renvoie à la fonction appelante le produit et la somme de ces deux entiers.

- 1. Écrire cette fonction, en expliquant précisément l'intérêt de chacun de ces paramètres et en illustrant son utilisation dans un programme principal.

Exercice 5. Traitement de tableau

1.5 + 0.5 = 2 pts

Soit T un tableau d'entiers de taille n . Les éléments de T sont définis de la manière suivante :

$$\forall i \in [1, n-1], T[i] = i \cdot (T[0] + \dots + T[i-1]), \text{ avec } T[0] = 1.$$

- 1. Écrire une fonction qui, étant donné un tableau vide de taille $n = 20$, remplit le tableau en respectant la contrainte ci-dessus. Illustrer l'utilisation de cette fonction dans un programme principal.
► 2. Donner le prototype de la fonction qui permet de faire le même traitement sur un tableau de taille n .

Exercice 6. Plus grande sous-chaîne commune

1 + 2 = 3 pts

Soient $A = a_1 \cdots a_n$ et $B = b_1 \cdots b_p$ deux chaînes de caractères, de longueur n et p , respectivement (avec éventuellement $n \neq p$). On souhaite déterminer la longueur de la plus grande sous-chaîne commune. Pour cela, on note $\ell_{i,j}$ la longueur d'une plus grande sous-chaîne commune aux chaînes $a_1 \cdots a_i$ et $b_1 \cdots b_j$, défini de la manière suivante :

$$\forall (i, j) \in [0, n] \times [0, p], \ell_{i,j} = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0, \text{ ou } a_i \neq b_j, \\ 1 + \ell_{i-1, j-1} & \text{sinon.} \end{cases}$$

La longueur de la plus grande sous-chaîne commune à A et B est alors le maximum des $\ell_{i,j}$.

- 1. Écrire une fonction qui prend une chaîne de caractères en paramètre, puis qui calcule et retourne la longueur de cette chaîne.
- 2. Écrire une fonction qui prend en paramètre A et B (deux chaînes de caractères), puis qui détermine et retourne la longueur de la plus grande sous-chaîne commune à A et B . Cette fonction n'utilisera pas d'allocation dynamique.

Exercice 7. Les pointeurs, comment ça marche ?

1 + 1 = 2 pts

- 1. Rappeler la définition d'un pointeur vue en cours, en illustrant éventuellement avec un schéma.
- 2. Quel est le résultat de l'exécution du programme suivant. Donner une explication (en vous aidant éventuellement des numéros de ligne et/ou d'un schéma).

```
1  int
2  main(void) {
3      int a = 5, b = a;
4      int *ptr = &a;
5      *ptr = a * (*ptr);
6      printf("--> a= %d et *ptr= %d... et b= %d\n", a, *ptr, b);
7      a = ((*ptr)--)-1;
8      printf("--> a= %d et *ptr= %d... et b= %d\n", a, *ptr, b);
9      return 0;
10 }
```

Exercice 8. Recherche dans un tableau trié 2D

2 + 1 = 3 pts

Retour en L1 : l'aspect algorithmique n'a plus de secret pour vous. Soit T un tableau 2D d'entiers de taille $\ell \times c$, c'est-à-dire, ℓ lignes et c colonnes, qui possède la propriété suivante : si on parcourt le tableau T ligne par ligne, de gauche à droite, la séquence des éléments rencontrés est strictement croissante (sans doublon). Par exemple, pour $(\ell, c) = (3, 4)$, le tableau T pourrait être :

1	5	23	24
32	47	56	57
60	71	82	90

Étant donné un entier e , si e appartient au tableau T , on souhaite rechercher puis afficher les indices de cet élément dans T , et $(-1, -1)$ si e n'appartient pas à T .

- 1. Écrire une fonction qui, étant donné un tableau T de taille 3×4 et l'élément e , recherche et renvoie les indices de e dans T , en effectuant au plus $\ell + c = 7$ comparaisons.
- 2. Illustrer son utilisation dans un programme principal, en prenant le tableau ci-dessus comme exemple.

Exercice 9. Un peu d'allocation dynamique

2 + 1 = 3 pts

- 1. Écrire un programme principal qui lit un entier $n \geq 1$ au clavier, puis qui
 - alloue dynamiquement un tableau de n entiers,
 - appelle la fonction de l'Exercice 5 sur ce tableau,
 - affiche le contenu du tableau en utilisant l'arithmétique des pointeurs,
 - puis libère enfin la mémoire allouée.
- 2. Faut-il modifier la fonction appelée. Proposer une justification.