

## ECUE «Introduction à la programmation » - Session 2

17 juin 2016 - Bruno Bouzy  
sans document - durée 1 heure 30

**Exercice 1 (4 points)**

Donner la sortie du programme ci-dessous. Pour chaque ligne en **caractères gras**, tenir compte de la **couleur de votre copie (bleu, rouge, vert, jaune)** pour **valoriser a et b** avec les valeurs précisées dans le commentaire de la ligne.

```
#include <stdio.h>
int main() {
    int a = 2;      // Couleur de la copie : Bleu:2, Rouge:3, Vert:4, Jaune:5
    int * p = &a;
    int b = *p;    printf("1: a = %d, b = %d, *p = %d.\n", a, b, *p);
    a *= 3;       // Couleur de la copie : Bleu:3, Rouge:4, Vert:5, Jaune:2
                 printf("2: a = %d, b = %d, *p = %d.\n", a, b, *p);
    b += 4;       // Couleur de la copie : Bleu:4, Rouge:5, Vert:2, Jaune:3
                 printf("3: a = %d, b = %d, *p = %d.\n", a, b, *p);
    int * q = &b; printf("4: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
    *q += (*p)++; printf("5: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
    *q *= ++(*p); printf("6: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
    p = q;        printf("7: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
    q = &a;       printf("8: a = %d, b = %d, *p = %d, *q = %d.\n", a, b, *p, *q);
    return 0;
}
```

**Exercice 2 (4 points)**

1) Ecrire une procédure void afficheDiviseurs(int n) affichant les diviseurs de n, différents de 1 et n, pour n>1. **(1 point)**. Par exemple, pour n valant 2015, 2016 ou 2017, la procédure affiche :

```
Diviseurs de 2015 = 5 13 31 65 155 403
Diviseurs de 2016 = 2 3 4 6 7 8 9 12 14 16 18 21 24 28 32 36 42 48 56 63
72 84 96 112 126 144 168 224 252 288 336 504 672 1008
Diviseurs de 2017 =
```

2) Ecrire une procédure void afficheDFP(int n) affichant la Décomposition en Facteurs Premiers de n, pour n>1. Par exemple, pour n valant 2015, 2016 ou 2017, la procédure affiche :

```
2015 = 5 x 13 x 31
2016 = 2^5 x 3^2 x 7
2017 = 2017
```

afficheDFP affiche la suite des nombres premiers diviseurs de n avec un ^ et un exposant si celui-ci est strictement supérieur à 1 (cf 2^5 dans 2016), ou sans ^ si l'exposant vaut 1 (cf 5, 13, 31 par exemple). afficheDFP affiche des x pour séparer les nombres premiers si il en y a au moins deux (cf 2015 et 2016) ou pas si le nombre est premier (cf 2017).

afficheDFP utilisera une boucle for avec une variable de boucle d allant de 2 à n. Si d divise n, la valeur de n sera divisée par d autant de fois que nécessaire. Pour d divisant n, afficheDFP utilisera une variable exposant pour conditionner l'affichage éventuel du ^ suivi de la valeur de l'exposant. Enfin, afficheDFP utilisera une variable deja pour conditionner l'affichage des éventuels x. **(3 points)**.

### Exercice 3 (12 points)

Dans cet exercice, on veut programmer une mise à jour *simplifiée* d'une ligne du jeu de 2048 lorsque le joueur joue vers la gauche. Une ligne contient 4 nombres. Un nombre est soit 0, soit un entier puissance de 2.

Quand le joueur effectue l'action à gauche sur une « ligne entrée », cela transforme la ligne en une « ligne sortie » avec un décalage des nombres non nuls vers la gauche, suivi d'une fusion éventuelle de deux nombres voisins égaux, et enfin suivi d'un autre éventuel décalage vers la gauche. Cf exemples ci-dessous.

« ligne entrée »:	« ligne sortie »:	Commentaire :
(0 0 2 0)	(2 0 0 0)	le 2 est venu à gauche. Un 0 est vu comme du vide.
(4 0 0 0)	(4 0 0 0)	le 4 reste à gauche.
(0 2 0 4)	(2 4 0 0)	le 2 et le 4 sont venus à gauche.
(2 32 4 16)	(2 32 4 16)	les nombres n'ont pas bougé.
(16 16 0 2)	(32 2 0 0)	les deux 16 ont fusionné en 32, le 2 est venu à gauche.
(16 0 16 2)	(32 2 0 0)	les deux 16 ont fusionné en 32, le 2 est venu à gauche.
(4 4 4 0)	(8 4 0 0)	les deux 4 ont fusionné en 8, le dernier 4 est venu à gauche.
(4 4 8 0)	(8 8 0 0)	les deux 4 ont fusionné en 8, le dernier 8 est venu à gauche.
(4 4 4 4)	(8 8 0 0)	les deux 1ers 4 ont fusionné en 8, les deux derniers 4 aussi.
(1024 0 0 1024)	(2048 0 0 0)	les deux 1024 ont fusionné en 2048.

On représente la ligne par un tableau de quatre entiers `int`. On va programmer deux fonctions de base : `decalageGauche(int * tab)` et `fusionGauche(int * tab)`. Ces fonctions seront utilisées par la fonction `miseAJour(int * tab)` appelée par le `main`.

- 1) Programmer une fonction `int zeroOuPuissanceDe(int n, int p)` retournant 1 si `n` est égal à 0 ou à une puissance de `p`, et retournant 0 sinon. **(2 points)**.
- 2) Programmer une procédure d'initialisation `void init(int * tab)`. Cette procédure remplit le tableau avec des nombres entrés au clavier. Avec la fonction `zeroOuPuissanceDe`, elle vérifie que les nombres entrés sont soit un 0 soit une puissance de 2. **(2 points)**.
- 3) Programmer une procédure d'affichage à l'écran `void affichage(int * tab)`. Elle respectera le format des exemples ci-dessus avec quatre nombres entiers entre parenthèses. **(1 point)**.
- 4) Programmer la procédure `void decalageGauche(int * tab)`. **(3 points)**.
- 5) Programmer la procédure `void fusionGauche(int * tab)`. **(2 points)**.
- 6) Programmer la procédure `void miseAJour(int * tab)`. Elle transforme le tableau en respectant les exemples de transformations « ligne entrée » et « ligne sortie » ci-dessus. Elle utilisera des appels convenables à `decalageGauche` et à `fusionGauche`. Elle affiche la ligne après chaque appel à `decalageGauche` et à `fusionGauche`. **(1 point)**.
- 7) Programmer le `main`. Le `main` contient la déclaration du tableau `ligne`. Il initialise la ligne avec `init`, affiche la ligne avec `affichage`, et met à jour la ligne avec `miseAJour`. **(1 point)**.