

---

## Travaux dirigés 3 : variables impératives et structure de contrôle *if*

---

L'objectif de ce TD est de vous familiariser avec l'écriture de programmes simples en langage C : calcul arithmétique, affichage de résultats et exécution conditionnelle d'instructions. Il aborde les mêmes notions que le TD 1, mais en utilisant le langage C.

**Correction.** Note aux chargés de TD.

- En cours, ils ont vu les variables impératives, le `printf` et le `if`. Leur sémantique a été donnée par leur traduction en langage amil.
- Ils doivent savoir résoudre/reproduire les exos marqués exercices types et faire leur trace sur un exemple quelconque.
- On garde la procédure des TD précédents :
  - on se donne des exemples
  - on trouve un algorithme en français
  - on traduit l'algorithme en C, en s'aidant de commentaires
  - on teste sur les exemples
- Le code de la fonction `main` vide en C a été présente, commente avec les différents points qu'ils vont voir au semestre. Il est long et peut être raccourci mais il faut s'assurer qu'ils sachent écrire ce genre de préambule du C avant d'écrire leurs programmes.
- Le code leur est toujours donné avec les commentaires, en suivant scrupuleusement l'indentation choisie. Ils doivent bien comprendre que le code et les commentaires sont indissociables. N'hésitez pas à ajouter des commentaires en fonction des difficultés rencontrées dans votre groupe.

## 1 Déclaration et affectation de variables impératives

### 1.1 Trace de programme en C

Soit le programme suivant :

```
1  /* Declaration de fonctionnalités supplémentaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Declaration des constantes et types utilisateurs */
6
7  /* Declaration des fonctions utilisateurs */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* Declaration et initialisation des variables */
13     int x;
14
15     x = 3;
16     x = x + 1;
17     printf("x = %d\n", x);
```

```

18
19     /* valeur fonction */
20     return EXIT_SUCCESS;
21 }
22
23 /* Definitions des fonctions utilisateurs */
24

```

1. Que fait ce programme ?

**Correction.**

- Il déclare la variable impérative `x` (occupe un espace memoire)
  - Il affecte 3 à la variable `x`
  - Affecte à `x` la valeur de l'expression  $(x + 1)$  qui vaut 4 : la sous-expression `x` s'évalue comme la valeur de la variable (3), plus un.
  - Affiche la valeur de l'expression `x` qui s'évalue comme la valeur de la variable, cad 4.
2. Donner la traduction des instructions aux lignes 15 et 16 en langage `amil`.

**Correction.**

```

CP vaut 1
# Soit la var x correspondant à la case mémoire d'adresse 10 (initialisation).
# affecte 3 a la var x
1 valeur 3 r0
2 ecriture r0 10
# x = x + 1
3 lecture 10 r0
4 valeur 1 r1
5 add r1 r0 # r0 vaut l'expression x + 1
6 ecriture r0 10 # affecte la valeur de l'expression à x

```

3. Donner la trace du programme C. Pour cela vous utiliserez un tableau comportant 1 colonne pour le numéro de ligne + autant de colonnes que de variables utilisées dans le programme + 1 colonne pour l'affichage éventuel du programme.

**Correction.** Il est très important qu'ils sachent faire une trace d'un programme C. Ils seront évalués dessus pendant les colles et aux examens. Le choix de la numérotation des lignes n'est pas imposé. On va d'habitude au plus court. Par contre, la numérotation doit être donnée explicitement dans le code.

trace :

ligne	x	sortie (affichage a l'ecran)
-----		
initialisation	?	
15	3	
16	4	
17		x = 4
renvoie EXIT_SUCCESS		

## 2 Execution conditionnelle d'instructions : *if*

### 2.1 Majeur ou mineur ?

Soit la variable `age`, contenant l'âge d'une personne. Écrire un programme qui affiche si cette personne est majeure ou mineure.

### Correction.

```
algorithme :
si age >= 18 alors
    affiche majeur
sinon
    affiche mineur

1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Declaration des constantes et types utilisateurs */
6
7  /* Declaration des fonctions utilisateurs */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* Declaration et initialisation des variables */
13     int age = 16; /* age de la personne */
14
15     if(age >= 18) /* majeur */
16     {
17         /* affiche majeur */
18         printf("Vous êtes majeur.\n");
19     }
20     else /* mineur (age < 18) */
21     {
22         /* affiche mineur */
23         printf("Vous êtes mineur.\n");
24     }
25
26     /* valeur fonction */
27     return EXIT_SUCCESS; /* renvoie OK */
28 }
29
30 /*Definitions des fonctions utilisateurs */
```

## 2.2 Exercice type : Le minimum de 3 valeurs

Soient 3 variables *a*, *b*, *c*, initialisées à des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran le minimum des 3 valeurs.

**Correction.** Nous donnons une version “standard” pour la recherche d’un extremum : une val par défaut. On parcourt les autres valeurs et on modifie en conséquence. Servira plus tard quand calcul extremum d’une série, bien qu’en général il soit plus difficile d’écrire le code initialisant l’extremum avec la première valeur de la série et on préfère initialiser avec une valeur limite (i.e.  $\min = +\infty$ ).

```
algorithme :
soit min = a /* valeur par defaut */
si b < min /* b plus petit que min courant */
    /* b est le min courant */
    min = b
/* min contient min(a,b) */
si c < min /* c plus petit que min courant */
```

```

/* c est le min courant */
min = c
/* min contient min(min(a,b),c) = min(a,b,c) */
affiche min

```

### 2.3 Exercice type : Quel temps fait-il ?

En vous inspirant du codage du genre vu en cours (1 pour MASCULIN, 2 pour FÉMININ), proposez un codage pour indiquer si le temps est COUVERT, ENSOLEILLÉ ou PLUVIEUX. Écrivez un programme principal qui, étant donné le temps affecté à une variable, affiche le temps qu'il fait.

#### Correction.

Exemple : Soit temps = 1 avec 0 COUVERT, 1 ENSOLEILLÉ et 2 PLUVIEUX.  
Affiche "Le temps est ensoleillé."

Algo :

```

/* Codage :
   0 COUVERT
   1 ENSOLEILLE
   2 PLUVIEUX
*/
si temps = 0 alors affiche couvert
si temps = 1 alors affiche ensoleille
si temps = 2 alors affiche pluvieux

les cas sont mutuellement exclusifs.

/* Declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* Declaration des constantes et types utilisateurs */

/* Declaration des fonctions utilisateurs */

/* Fonction principale */
int main()
{
    /* Declaration et initialisation variables */
    int temps = 1; /* temps qu'il fait */

    /* si temps = 0 alors affiche couvert */
    if(temps == 0) /* COUVERT */
    {
        printf("Le temps est couvert.\n");
    }

    /* si temps = 1 alors affiche ensoleille */
    if(temps == 1) /* ENSOLEILLE */
    {
        printf("Le temps est ensoleille.\n");
    }
}

```

```

    /* si temps = 2 alors affiche pluvieux */
    if(temps == 2) /* PLUVIEUX */
    {
        printf("Le temps est pluvieux.\n");
    }

    /* valeur fonction */
    return EXIT_SUCCESS;
}

/* Definitions des fonctions utilisateurs */

```

## 2.4 Exercice type : Dans 1 seconde, il sera exactement...

Écrire un programme principal qui, étant donnée une heure représentée sous la forme de 3 variables pour les heures, *h*, les minutes, *m* et les secondes, *s*, affiche l'heure qu'il sera 1 seconde plus tard. Il faudra envisager tous les cas possibles pour le changement d'heure. Deux exemples de sortie sont :

```

L'heure actuelle est : 23h12m12s
Dans une seconde, il sera exactement : 23h12m13s

```

```

L'heure actuelle est : 23h59m59s
Dans une seconde, il sera exactement : 00h00m00s

```

### Correction.

algo :

- affiche l'heure actuelle
- ajoute une seconde
- si tour du cadran des secondes alors
  - remise a 0 des secondes
  - il est une minute de plus
  - si tour du cadran des minutes alors
    - remise a 0 des minutes
    - il est une heure de plus
    - si tour du cadran des heures alors
      - remise a zero des heures
- affiche la nouvelle heure

```

1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Declaration des constantes et types utilisateurs */
6
7  /* Declaration des fonctions utilisateurs */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* soient 23h59m59s */
13     int h = 23;
14     int m = 59;
15     int s = 59;
16
17     /* affiche l'heure actuelle */

```

```

18     printf("L'heure actuelle est : %dh%dm%ds\n",h,m,s);
19
20     /* une seconde de plus */
21     s = s + 1;
22
23     if(s == 60) /* tour du cadran des secondes */
24     {
25         /* remise a 0 */
26         s = 0;
27
28         /* une minute de plus */
29         m = m + 1;
30
31         if(m == 60) /* tour du cadran des minutes */
32         {
33             /* remise a 0 */
34             m = 0;
35
36             /* une heure de plus */
37             h = h + 1;
38
39             if(h == 24) /* tour du cadran des heures */
40             {
41                 /* remise a zero */
42                 h = 0;
43             }
44         }
45     }
46     /* h,m,s contiennent l'heure une seconde plus tard */
47
48     /* affiche l'heure */
49     printf("Dans une seconde, il sera exactement : %dh%dm%ds\n",h,m,s);
50
51     /* valeur fonction */
52     return EXIT_SUCCESS;
53 }
54
55 /* Definitions des fonctions utilisateurs */
56

```