Université Paris-Nord — L1
Institut Galilée — Année 2008-2009
Licence 1 - 1er semestre

Travaux dirigés 9 : écriture et appel de fonctions (1)

Correction. Note aux chargés de TD : cette semaine les appels de fonctions n'ont lieu que dans le `main`. Il n'y a pas non plus de tableau de en paramètre des fonctions.

1 Trace de fonctions

Faire la trace du programme suivant et dire ce que calcule la fonction `est_xxx`.

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf() */
3  /* Declaration constantes et types utilisateurs */
4  #define TRUE 1
5  #define FALSE 0
6
7  /* Declaration de fonctions utilisateurs */
8  int est_xxx(int n);
9
10 /* Fonction principale */
11 int main()
12 {
13     /* Declaration et initialisation des variables */
14     int n = 9;
15
16     if (est_xxx(n))
17     {
18         printf("L'entier %d est xxx\n", n);
19     }
20     else
21     {
22         printf("L'entier %d n'est pas xxx\n", n);
23     }
24
25     /* Valeur fonction */
26     return EXIT_SUCCESS;
27 }
28
29 /* Implantation de fonctions utilisateurs */
30 int est_xxx(int n)
31 {
32     int i;
33
34     for (i = 2; i < n; i = i + 1)
35     {
36         if (n % i == 0)
37         {
38             return FALSE;
39         }
```

```
40      }
41      return TRUE;
42  }
```

Correction. Cette fonction teste si son argument est un nombre premier : elle renvoie TRUE si son argument n est premier (ou négatif ou nul – remarque : zéro n'est pas premier), et FALSE sinon.

`main()`

ligne	n	Affichage (sortie écran)
initialisation	9	
22		L'entier 9 n'est pas xxx
26	SORTIE AVEC SUCCÈS	

`est_xxx(9)`

ligne	n	i	Affichage
initialisation	9	?	
34		2	
40		3	
38	renvoie faux (0)		

TAB. 1 – Trace du programme de l'exercice 1.

2 Prototypage et implantation de fonctions

Les fonctions `valeur_absolue()`, `factorielle()` et `minimum()` ne sont pas fournies avec le programme suivant. Compléter le programme avec le prototype et le code des fonctions (le `main` doit rester inchangé) et faire la trace du programme complet.

```
14  int main()
15  {
16      int x = -3;
17      int y = 5;
18      int z;
19
20      /* Un calcul sans signification particulière */
21      x = valeur_absolue(x); /* valeur absolue de x */
22      z = minimum(x, y);     /* minimum entre x et y */
23      z = factorielle(z);    /* z! */
24      z = minimum(y, z);     /* minimum entre y et z */
25
26      /* Valeur fonction */
27      return EXIT_SUCCESS;
28  }
```

Correction.

Code.

```
1   #include <stdlib.h> /* EXIT_SUCCESS */
2
3   /* Declaration de constantes et types utilisateurs */
4
5   /* Declaration de fonctions utilisateurs */
6   /* Retourne la valeur absolue de son argument entier */
7   int valeur_absolue(int n);
8   /* Retourne le minimum des deux valeurs en argument */
9   int minimum(int a, int b);
10  /* Retourne la factorielle de l'argument entier (si positif) */
11  int factorielle(int n);
12
13  /* Fonction principale */
14  int main()
15  {
16      int x = -3;
17      int y = 5;
18      int z;
19
20      /* Un calcul sans signification particulière */
21      x = valeur_absolue(x); /* valeur absolue de x */
22      z = minimum(x, y);     /* minimum entre x et y */
23      z = factorielle(z);    /* z! */
24      z = minimum(y, z);     /* minimum entre y et z */
25
26      /* Valeur fonction */
27      return EXIT_SUCCESS;
28  }
29  /* Implantation de fonctions utilisateurs */
30  int valeur_absolue(int n)
31  {
32      if (n < 0) /* si n est négatif */
33      {
34          return -n; /* inverser le signe de n et renvoyer */
35      }
36      /* n est positif */
37      return n;
38  }
39
40  int minimum(int a, int b)
41  {
42      if (a < b) /* Si a est le minimum */
43      {
44          return a; /* renvoyer a */
45      }
46      /* a n'est pas le minimum */
47      return b; /* renvoyer b */
48  }
49
50  int factorielle(int n)
51  {
52      int i; /* Var. de boucle */
53      int res = 1; /* resultat */
54
55      for (i = 1; i <= n; i = i + 1) /* Pour i = 1, 2, ..., n */
56      {
```

```
57          res = res * i; /* mettre i dans le produit */
58      }
59
60      /* Valeur fonction */
61      return res;
62  }
63
```

`main()`

ligne	x	y	z	Affichage
initialisation	-3	5	?	
21	3			
22			3	
23			6	
24			5	
27	SORTIE AVEC SUCCÈS			

`valeur_absolue(-3)`

ligne	n	Affichage
initialisation	-3	
34	retourne 3	

`minimum(3, 5)`

ligne	a	b	Affichage
initialisation	3	5	
44	retourne 3		

`factorielle(3)`

ligne	n	i	res	Affichage
initialisation	3	?	1	
55		1		
57			1	
58		2		
57			2	
58		3		
57			6	
58		4		
61	retourne 6			

`minimum(5, 6)`

ligne	a	b	Affichage
initialisation	5	6	
44	retourne 5		

TAB. 2 – Trace du programme de l'exercice 2.

3 Dessin d'étoiles

1. Définir la fonction `motif` du programme suivant (au centre), de manière à ce que l'affichage soit le motif *demi-carré d'étoiles*, à gauche ci-dessous.

   Correction. Pour aider, on peut écrire x et y comme coordonnées (x horizontal, y vertical, du haut vers le bas) :

```
0   1234...
1   *---
2   **--
3   ***-
4   ****
5   ...
```

```
/* Implantation de fonctions utilisateurs */
char motif(int x, int y)
{
    if ( x <= y )
    {
        return '*';
    }
    return '-';
}
```

2. Même question pour le second motif (*triangle isocèle*), à droite.

   Correction.

```
/* Implantation de fonctions utilisateurs */
char motif(int x, int y)
{
    if ( (x <= y) && (LARGEUR - x < y) )
    {
        return '*';
    }
    return '-';
}
```

Les plus avancés peuvent essayer d'autres motifs (cercle, losange...).

```
*----------    ...                                                    ----------
**---------    #define LARGEUR 11                                     ----------
***--------    #define HAUTEUR 11                                     ----------
****-------                                                           ----------
*****------    /* Declaration de fonctions utilisateurs */           ----------
******-----    char motif(int x, int y);                              -----*-----
*******----                                                           ----***----
********---    /* Fonction principale */                             ---*****---
*********--    int main()                                             --*******--
**********-    {                                                      -*********-
**********        /* Declaration et initialisation des variables */  **********
                  int i; /* Var de boucle, colonne */
                  int j; /* Var de boucle, ligne */

                  for (j = 1; j <= HAUTEUR; j = j + 1)
                  {
                      for (i = 1; i <= LARGEUR; i = i + 1)
                      {
                          printf("%c", motif(i, j));
                      }
                      printf("\n");
                  }
                  /* Valeur fonction */
                  return EXIT_SUCCESS;
              }
```