Travaux dirigés 4 : fonctions et procédures (1)

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 /* Definition 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 }
```

2 Déclaration et définition 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  }
```

3 Écriture de fonctions

Pour les questions suivantes il faut donner la déclaration et la définition de chaque fonction. Vous pouvez faire l'exercice une première fois en donnant uniquement les déclarations, puis le reprendre pour les définitions. Répondre dans un seul programme, dans lequel vous écrirez une fonction principale (`main`) faisant appel à ces fonctions pour les tester.

1. Écrire la fonction `carre` qui prend en entrée un entier et qui renvoie le carré de cet entier.
2. Écrire la fonction `cube` qui prend en entrée un entier et qui renvoie le cube de cet entier.
3. Écrire la fonction `est_majeur` qui prend en entrée un entier représentant l'age en années d'une personne et renvoie `TRUE` si cette personne est majeure et `FALSE` sinon (on considérera les deux constantes utilisateurs bien déclarées).
4. Écrire la fonction `somme` qui prend en entrée un entier n et qui renvoie $\sum_{i=1}^{i=n} i$. Où faut-il déclarer la variable de boucle ?
5. Si vous ne l'avez pas fait à l'exercice précédent, écrire la fonction `factorielle` qui prend en entrée un entier n et qui renvoie $\Pi_{i=1}^{i=n} i$.
6. Écrire la procédure `afficher_rectangle` qui prend en entrée deux entiers, largeur et hauteur, et affiche un rectangle d'étoiles de ces dimensions.
7. Écrire la fonction `saisie_utilisateur` sans argument, qui demande à l'utilisateur de saisir un nombre entier et le retourne.
8. Écrire la fonction `binomial` qui prend en entrée un entier n et un entier p et retourne le nombre de tirages différents, non ordonnés et sans remise, de p éléments parmi n, c'est à dire le coefficient binomial :

$$\binom{n}{p} = \mathrm{C}_n^p = \frac{n!}{p!(n-p)!}.$$

   Faire appel à la fonction `factorielle`.
9. Optionnel. Écrire la fonction `saisie_dans_intervalle` à deux paramètre entiers a et b, qui demande à l'utilisateur de saisir un nombre entier jusqu'à ce qu'il soit dans l'intervalle $[a, b]$ et le retourne. On pourra fixer un nombre maximum de tentatives après quoi le nombre de l'intervalle le plus proche de la saisie utilisateur sera renvoyé.