1 Exemples

1.1 Portée

Qu'affiche le programme suivant ?

```
1  #include <stdio.h>
2  int a;
3
4  int main()
5  {
6    int b;
7
8    a=0;
9    b=a;
10   printf("a: %d, b: %d\n",a,b);
11   a += 5;
12   {
13     int a;
14     printf("a: %d, b: %d\n",a,b);
15   }
16   a += 5;
17   {
18     int b;
19     b=a;
20     printf("a: %d, b: %d\n",a,b);
21     b += 5;
22     {
23       int b;
24       b=0;
25       printf("a: %d, b: %d\n",a,b);
26     }
27     printf("a: %d, b: %d\n",a,b);
28   }
29   printf("a: %d, b: %d\n",a,b);
30   return 0;
31 }
```

1.2 Passage des paramètres

Qu'affiche le programme ci-dessous (attention au piege) ?

```
1  #include <stdio.h>
2  int a;
3
4  void affiche_ab(int b, int a)
5  {
6    printf("a: %d, b: %d\n",a,b);
7  }
8
9  int main()
10 {
11   int b;
12
13   a=0;
14   b=a;
15   affiche_ab(a,b);
16   a += 5;
17   {
18     int a;
```

```
19        affiche_ab(a,b);
20      }
21      a += 5;
22      {
23        int b;
24        b=a;
25        affiche_ab(a,b);
26        b += 5;
27        {
28          int b;
29          b=0;
30          affiche_ab(a,b);
31        }
32        affiche_ab(a,b);
33      }
34      affiche_ab(a,b);
35      return 0;
36    }
```

2 Exercice à la maison

Nous souhaitons réaliser une bibliothèque permettant la création et la destruction dynamique d'un matrice, ainsi que l'accès à ses éléments. La structure de donnée choisie est constituée d'un seul bloc de mémoire alloué. Le fichier d'en-tête de la bibliothèque, vecteur.h est déja écrit, son contenu est donné ci-dessous. Question : Ecrivez le fichier vecteur.c contenant le corps des fonctions vecteur_alloc, vecteur_free et acces_vecteur.

```
#ifndef __VECTEUR__
#define __VECTEUR__

typedef struct vecteur_data {
  int l;        /* Nombres de composantes du vecteur */
  double *data;   /* Pointeur sur les elements du vecteur     */
} *vecteur;

/* vecteur_alloc
   description : alloue un vecteur en memoire
   parametres : la taille du vecteur
   valeur de retour : un vecteur ou NULL sur erreur
   effets de bord : alloue de la memoire
*/
vecteur vecteur_alloc(int l);

/* vecteur_free
   description : libere un vecteur precedemment allouee par vecteur_alloc
   parametres : le vecteur
   valeur de retour : aucune
   effets de bord : libere de la memoire
*/
void vecteur_free(vecteur m);

/* acces_vecteur
   description : retourne un pointeur vers l'element (i) du vecteur
   parametres : le vecteur
   valeur de retour : un pointeur vers double
   effets de bord : aucun
*/
double *acces_vecteur(vecteur m, int i);
#endif
```