# Indice fou

1. Voici un programme qui modifie pernicieusement des données à cause d'un indice incorrect (si la valeur *n* entrée par l'utilisateur est supérieure à la taille du tableau) :

```
double sigma (double t[], int n) {
  int i;
  double s = 0;

  for (i = 0; i < n; i++)
    s = s + t[i];
  return s;
}

double test[4], x, y;

void main (void) {
  int i, n;

  printf ("Entrer la taille : ");
  scanf ("%d", &n);

  y = 1;
  for (i = 0; i < n; i++)
    test[i] = i * i;

  /* premier calcul */
  x = sigma (test, n);
  printf ("x = %g, y = %g\n", x, y);

  /* second calcul */
  x = sigma (test, n);
  printf ("x = %g, y = %g\n", x, y);
}
```

   Expliquer exactement les résultats obtenus :

```
Entrer la taille : 4
x = 14, y = 1
x = 14, y = 1
Entrer la taille : 5
x = 30, y = 1
x = 44, y = 1
Entrer la taille : 6
x = 55, y = 25
x = 94, y = 25
Entrer la taille : 10000
Segmentation fault
```

2. On remplace la déclaration globale du tableau par une déclaration locale, et on exécute le programme suivant :

```
void main (void) {
  int i, n;
  double test[4];

  printf ("Entrer la taille : ");
  scanf ("%d", &n);

  for (i = 0; i < n; i++) {
    test[i] = i * i;
    printf ("i = %d, n = %x\n", i, n);
```

```
    }
}
```

Expliquer le comportement du programme, qui boucle pour $n = 5$. Indication : les variables locales sont rangées en mémoire dans l'ordre inverse de leurs déclarations.

```
Entrer la taille : 4
i = 0, n = 4
i = 1, n = 4
i = 2, n = 4
i = 3, n = 4
Entrer la taille : 5
i = 0, n = 5
i = 1, n = 5
i = 2, n = 5
i = 3, n = 5
i = 0, n = 40300000
i = 1, n = 40300000
i = 2, n = 40300000
i = 3, n = 40300000
i = 0, n = 40300000
i = 1, n = 40300000
i = 2, n = 40300000
i = 3, n = 40300000
i = 0, n = 40300000
i = 1, n = 40300000
i = 2, n = 40300000
i = 3, n = 40300000
i = 0, n = 40300000
etc.
```