```
#include <iostream.h>

const int m_max = 50;
const int n_max = 50;
t
```

```
                        if (laby[i][j - 1] == 0)
                                dir = (dir + 1) % 4;
                        else
                                ir = (dir + 2) % 4;
                }
                else if (dir == 3)      // Cas d'un deplacement vers la droite;
                {
                        if (laby[i - 1][j] == 0)
                                dir = (dir + 3) % 4;
                        else if (laby[i][j - 1] != 0)
                                if (laby[i + 1][j] == 0)
                                        dir = (dir + 1) % 4;
                                else
                                        dir = (dir + 2) % 4;
                }
                cpt++;                  // Compte le nombre de pas franchi. Modifie les
                                        // coordonees du "curseur" en fonction de la direction
                                        // choisie precedement.
                switch (dir)
                {
                case 0:
                        i++;
                        break;
                case 1:
                        j++;
                        break;
                case 2:
                        i--;
                        break;
                case 3:
                        j--;
                        break;
                }
        }
        while (i > 0 && j > 0 && i < n - 1 && j < m - 1);        // La boucle s'execute
                                                                 // tant que l'on
                                                                 // n'atteint pas un bord.
        return cpt;
}

// Le principe est le meme pour la fonction gauche, mais on procede dans l'autre sens
int
gauche (MatrM
```

```
                        else
                                dir = (dir + 2) % 4;
                }
                else if (dir == 1)
                {
                        if (laby[i - 1][j] == 0)
                                dir = (dir + 1) % 4;
                        else if (laby[i][j + 1] != 0)
                                if (laby[i + 1][j] == 0)
                                        dir = (dir + 3) % 4;
                                else
                                        dir = (dir + 2) % 4;
                }
                else if (dir == 2)
                {
                        if (laby[i][j - 1] == 0)
                                dir = (dir + 1) % 4;
                        else if (laby[i - 1][j] != 0)
                                if (laby[i][j + 1] == 0)
                                        dir = (dir + 3) % 4;
                                else
                                        dir = (dir + 2) % 4;
                }
                else if (dir == 3)
                {
                        if (laby[i + 1][j] == 0)
                                dir = (dir + 1) % 4;
                        else if (laby[i][j - 1] != 0)
                                if (laby[i - 1][j] == 0)
                                        dir = (dir + 3) % 4;
                                else
                                        dir = (dir + 2) % 4;
                }
                cpt++;
                switch (dir)
                {
                case 0:
                        i++;
                        break #
                case 1:
                        j++;
                        break;
                case 2:
                        i--;
                        break;
                case 3:
                        j--;
                        break;
                }
        }
        while (i > 0 && j > 0 && i < n - 1 && j < m - 1);
        return cpt;
}

bool
meilleur (MatrM
```

```
{
        // La fonction meilleur renvoie vrai si le chemin de droite est plus court.
        return (droite (laby, m, n) < gauche (laby, m, n));
}

// Simple fonction destinee a afficher la matrice a l'ecran.
void
lecture (MatrM
```