```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
#include <stdbool.h>

const int MAX=100;

/*
 * Antécédent : m,n>0, resp. #lignes et de col de la matrice mat
 * Rôle : initialise aléatoirement (sur [0;MAX[) la matrice mat
 */
void initAléatoire(const int m, const int n, int mat[][n]) {
  for (int i=0; i<m; i++)
    for (int j=0; j<n; j++)
      mat[i][j]=rand()%MAX;
}

/*
 * Antécédent : m,n>0, resp. #lignes et de col de la matrice mat
 * Rôle : écrit la matrice mat sur la S/S
 */
void écrireMatrice(const int m, const int n, const int mat[][n]) {
  for (int i=0; i<m; i++) {
    // écrire la ième ligne
    for (int j=0; j<n; j++)
      printf("%4d", mat[i][j]);
    //
    printf("\n");
  }
}

/*
 * Antécédent : m,n>0, resp. #lignes et de col de la matrice mat
 * Rôle : renvoie le minimum de la matrice
 */
int minimum(const int m, const int n, const int mat[][n]) {
  int min = mat[0][0];
  for (int i=0; i<m; i++)
    for (int j=0; j<n; j++)
      if (mat[i][j]<min) min=mat[i][j];
  // invariant:
  return min;
}

/*
 * Antécédent : A[m][p] et B[p][n] deux matrices d'entiers initialisées
 * Conséquent : C[m][b] = A[m][p]  $\times$ B[p][n]
 */
void produit(const int m,  const int n,  const int p,
             const int A[][p], const int B[][n], int C[][n]) {
  for (int i=0; i<m; i++)
    for (int j=0; j<n; j++) {
      // calculer les $C_{i,j} = \sum_{k=1}^{p}{A_{i,k}\times{}B_{k,j}}$
      int som = 0;
      for (int k=0; k<p; k++) som+=A[i][k]*B[k][j];
      C[i][j] = som;
    }
}

/*
 * Antécédent : m,n>0, resp. #lignes et de col de la matrice mat
 *              e entier à rechercher dans mat
 * Rôle : teste si e est présent ou pas dans mat
 */
bool rechercher(const int e, const int m, const int n,
```



```c
                const int mat[][n])
{
  for (int i=0; i<m; i++)
    for (int j=0; j<n; j++)
      if (mat[i][j] == e)
        // trouvé
        return true;
  // on a parcouru la matrice mat sans trouver e
  return false;
}

int main(void) {
  int  m, n;
  scanf("%d %d", &m, &n);
  assert(m>0 && n>0);

  int matrix[m][n];

  srand(time(NULL));

  initAléatoire(m, n, matrix);
  écrireMatrice(m, n, matrix);
  printf("min = %d\n", minimum(m, n, matrix));
  // recherche
  printf("e = ");
  int e;
  scanf("%d", &e);
  printf("trouvé = %d\n", rechercher(e, m, n, matrix));
  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>


/*
 * antécédent : n>0
 * conséquent : m est la matrice identité
 * Note : version peu intelligente avec n² tests !
 */
void identité0(const int n, int m[][n]) {
  for (int i=0; i<n; i++) {
    // mettre les 0 de part et d'autre de la diagonale
    for (int j=0; j<n; j++)
      m[i][j] = i==j;
  }
}


/*
 * antécédent : n>0
 * conséquent : m est la matrice identité
 * Note : smart version sans test et exactement n² affectation
 */
void identité(const int n, int m[][n]) {
  // algo : on parcourt la demi-matrice inférieure
  //        et on met les 0 dans le 2 demi-matrices
  //        puis le 1 sur la diagonale
  // écrire le 1er 1
  m[0][0] = 1;
  for (int i=1; i<n; i++) {
    // mettre les 0 de part et d'autre de la diagonale
    for (int j=0; j<i; j++)
      m[i][j] = m[j][i] = 0;
    // mettre 1 sur la diagonale à la ligne i
    m[i][i] = 1;
  }
}



/*
 * antécédent : n>0
 * conséquent : m est écrite sur la S/S
 */
void écrireMat(const int n, const int m[][n]) {
  for (int i=0; i<n; i++) {
    // écrire la ième ligne sur la S/S
    for (int j=0; j<n; j++)
      printf("%3d", m[i][j]);
    //
    printf("\n");
  }
}

int main(void) {
  int n;
  scanf("%d", &n);
  int mat[n][n];
  // créer la matrice identité
  identité(n, mat);
  // puis l'afficher sur la S/S
  écrireMat(n, mat);

  return EXIT_SUCCESS;
}
```



```c
/**
 * Ce programme affiche le triangle de Pascal des
 * coefficients binomiaux d'ordre n (<N)
 *
 *    @author: Vincent Granet "vg@unice.fr"
 *  Creation @date:  6-Nov-2015 07:12
 * Last file update: 16-Jan-2024 09:52
 */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

/*
 * Antécédent : n>0 ordre du triangle de Pascal
 *              tp contient les coefficients binomiaux
 * Rôle : écrit le triangle de Pascal d'ordre n
 *        sur la sortie standard
 */
void écrireTriangleDePascal(const int n, const int tp[][n+1]) {
  for (int i=0; i<=n; i++) {
    // écrire la ligne i
    for (int j=0; j<=i; j++)
      printf("%4d", tp[i][j]);
    //
    printf("\n");
  }
}

/*
 * Antécédent : n>0 ordre du triangle de Pascal
 * Conséquent : tp contient les coefficients binomiaux
 * Rôle : calcule le triangle de Pascal d'ordre n
 */
void triangleDePascal(const int n, int tp[][n+1]) {
  // écrire les 1 sur la première et deuxième ligne
  tp[0][0]= tp[1][0]=tp[1][1] = 1;
  // construire les coeff binomiaux
  for (int i=2; i<=n; i++) {
    // calculer (par récurrence) les coeff binomiaux de la ligne i
    tp[i][0] = tp[i][i] = 1;
    for (int j=1; j<i; j++)
      tp[i][j] = tp[i-1][j-1]+tp[i-1][j];
  }
}

int main(void) {
  int n;

  printf("n = ");
  scanf("%d", &n);
  assert(n>0);
  // n>0 => calculer le triangle de Pascal d'ordre n
  int tp[n+1][n+1];
  triangleDePascal(n, tp);
  écrireTriangleDePascal(n, tp);

  return EXIT_SUCCESS;
}
```