```c
/*********************************************************************
 * Uebung 6, Aufgabe 1
 *
 * Matrizen erstellen, besetzen und ausgeben
 *********************************************************************/


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


int** alloc_matrix(int m, int n)
{
  int** mat_ptr;
  int i;

  /* Speicherplatz fuer eine Spalte von int-Pointern reservieren */
  mat_ptr = (int**) malloc(m * sizeof(int*));

  /* m-mal eine Zeile von int-Werten reservieren */
  for(i=0; i<m; ++i)                                                    for
    mat_ptr[i] = (int*) malloc(n * sizeof(int));

  return mat_ptr;
}


void print_matrix(int** matr_ptr, int m, int n)                        print_matrix
{
  int i, j;

  printf("\n");
  for(i=0; i<m; ++i)
  {
    for(j=0; j<n; ++j)
      printf("%6d", matr_ptr[i][j]);
    printf("\n");
  }
  printf("\n");
}


void set_matrix(int** matr_ptr, int m, int n)                          set_matrix
{
  int i, j;

  for(i=0; i<m; ++i)
```



```c
    for(j=0; j<n; ++j)
      matr_ptr[i][j] = i+j;
}

main(int argc, char *argv[])                                           main
{
  int** matrix;
  int m, n;

  if (3 != argc)
  {
    printf("\n** ACHTUNG: Funktionsaufruf ist 'matrix m n' ! **\n\n");
    return 10;
  }

  /* String in int umwandeln */
  m = atoi(argv[1]);
  n = atoi(argv[2]);

  matrix = alloc_matrix(m, n);
  set_matrix(matrix, m, n);
  print_matrix(matrix, m, n);

  /* Speicherplatz wieder loeschen:           */
  /* Zuerst werden die einzelnen Zeilen geloescht */
  /*** Beispiel fuer 'schlechten' Code:                                ***/
  /*** Die zwei folgenden Zeilen machen was sie sollen, sie sind SEHR kurz ***/
  /*** und benoetigen keine zusaetzliche Variable (sind also echt cool),   ***/
  /*** ABER: sie sind nicht gut lesbar !!!!  %-(                          ***/
  for(;--m>=0;)
    free(matrix[m]);

  /* Dann noch der Spaltenvektor              */
  free(matrix);
}
```