```c
/*
 * Übungsaufgabe 0:
 *
 * Beispiel-Vektoren und Matrizen erzeugen
 *
 * Autoren:
 *   1. Anton Prochel
 *   2. ...
 *   3. ...
 * nicht beteiligt
 *   4. ...
 *
 * Compilieren mit:
 *
 *   gcc -o example1 -lm example1.c
 */

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


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Berechnet die Fakultät von n
 *
 *  n: ganzzahlig, >= 1
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
double fak(int n)
{
    int i = 1;
    double f = 1.;

    while (i ≤ n)
    {
        f = f * i; /* impl. Typumwandlung */
        i++;
    }

    return f;
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Zweck:
 *    Erzeugt eine (nicht initialisierte !) m x n Matrix
 *    d.h. belegt den entsprechenden Speicherplatz.
 *  Eingabe:
 *    n: Spalten
 *    m: Zeilen
 *  Bem.:
 *    n, m: ganzzahlig, >= 1
 *  Rückgabe:
 *    Zeiger auf eine Matrix
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
double** mk_matrixd(int m, int n)
{
    /* Speicherplatz für n x m Matrix aus Gleitkommazahlen allokieren */
    int i;

    double** A;
    A = (double**) malloc(n * sizeof(double*));
    for (i = 0; i < n; i++)
        A[i] = (double*) malloc(m * sizeof(double));

    return A;
}
```

```c
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Zweck:
 *    Gibt den Speicherplatz einer m x n Matrix aus double-Werten frei.
 *  Eingabe:
 *    A: freizugebende Matrix
 *    n: Spalten
 *    m: Zeilen
 *  Bem.:
 *    n, m: ganzzahlig, >= 1
 *  Rückgabe:
 *    keine
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void free_matrixd(double** A, int m, int n)
{
    int i;

    /* Speicherplatz der Matrix freigeben. */
    for (i = 0; i < n; i++)
        free(A[i]);
    free(A);
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Zweck:
 *    m x n Matrix ausgeben, elementweise
 *  Eingabe:
 *    A: auszugebende Matrix
 *    n: Spalten
 *    m: Zeilen
 *  Bem.:
 *    n, m: ganzzahlig, >= 1
 *  Rückgabe:
 *    keine
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void dump_matrixd(double** A, int m, int n)
{

    int i, j;

    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            printf ("a[% 3d][% 3d] = %f\n",i, j, A[i][j]);
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Zweck:
 *    m x n Matrix ausgeben in Matrixform
 *  Eingabe:
 *    A: auszugebende Matrix
 *    n: Spalten
 *    m: Zeilen
 *  Bem.:
 *    n, m: ganzzahlig, >= 1
 *  Rückgabe:
 *    keine
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void print_matrixd(double** A, int m, int n)
{

    int i, j;

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

```
    {
        for (j = 0; j < m; j++)
            printf (" %12f", A[i][j]);
        printf("\n");
    }
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Ziel:
 *    Erzeugt eine m x n Hilbert-Matrix
 *  Eingabe:
 *    n: Spalten
 *    m: Zeilen
 *  Bem.:
 *    n, m: ganzzahlig, >= 1
 *  Rückgabe:
 *    Zeiger auf Matrix
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
double** hilbertd(int m, int n)
{
    int i, j;

    /* Speicherplatz für m x n Matrix aus Gleitkommazahlen allokieren */
    double** A = mk_matrixd(m, n);

    /* Matrix befüllen */
    for (i = 0; i < n; i++) /* Spalten */
        for (j = 0; j < m; j++) /* Zeilen */
        {
            A[i][j] = (double) ((i + 1.) + (j + 1.));
        }

    return A;
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Ziel:
 *    Bsp. für Erzeugung, Initialisierung, Anzeige und Freigeben eines Vektors
 *  Eingabe:
 *    n: Vektordimension
 *  Bem.:
 *    n: ganzzahlig, >= 1
 *  Rückgabe:
 *    nix
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void ex_vec1(int n)
{

    int i;


    /* Vektor allokieren */
    double* v = (double*) malloc(n * sizeof(double));

    /* Vektor initialisieren */
    for (i = 0; i < n; i++)
        v[i] = pow (i + 1., 2.);

    /* Vektorinhalt nochmal ausgeben */
    for (i = 0; i < n; i++)
        printf ("v[%3d] = %f\n",i, v[i]);

    /* Vektor freigeben */
    free(v);
```

```
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
int main(void)
{

    printf ("HALLO\n");

    int i,j;
    int n,m;

    double** H1;

    n = 4;
    m = 4;

    double f;
    f = fak(m);
    printf("f=%f\n", f);


    /* Hilbert-Matrix darstellen */
    /* erzeugen */
    H1 = hilbertd(m, n);
    /* anzeigen */
    print_matrixd(H1, m, n);
    /* freigeben */
    free_matrixd(H1, m, n);

    printf ("\n");

    /* Beispiel mit Vektor */
    ex_vec1(4);

}
```