Arrays im Speicher

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>

int main()
{
    float f = M_PI;
    char c = 'A';
    double d = M_E;
    int n = 15;

    printf("PI = %f\t\t addr. = %p\n", f, (void *)&f);
    printf("c = %c\t\t\t addr. = %p\n", c, (void *)&c);
    printf("E = %e\t addr. = %p\n", d, (void *)&d);
    printf("n = %d\t\t\t addr. = %p\n\n", n, (void *)&n);

    int nArr[4][3][3] = {
                          {
                             {33, 8, 0},
                             {4, 77, 90},
                             {89, 67, 1}
                          },
                          {
                              {11, 6, 7},
                              {4, 77, 20},
                              {19, 88, 11}
                          },
                          {
                              {23, 78, 90},
                              {1, 9, 90},
                              {3, 67, 33}
                          },
                          {
                              {34, 4, 99},
                              {68, 7, 88},
                              {10, 54, 5}
                          }
                        };

    for(int i = 0; i < 4; i++)
        for(int j = 0; j < 3; j++)
            for(int k = 0; k < 3; k++)
                printf("nArr[%d][%d][%d] = %d\t addr. = %p\n",
                       i, j, k, nArr[i][j][k],
                      (void *)&nArr[i][j][k]);
    return 0;
}
```

Programmausgabe:

```
Willkürliche Belegung im Speicher (wo gerade Platz ist).

                                        höchste Adresse

PI = 3.141593             addr. = 000000ff4dfff870  difference = 1
c = A                     addr. = 000000ff4dfff86f  difference = 15
E = 2.718282e+00          addr. = 000000ff4dfff860  difference = 4
n = 15                    addr. = 000000ff4dfff85c

                                        kleinste Adresse


Arrayelemente liegen immer lückenlos hintereinander.

nArr[0][0][0] = 33        addr. = 000000ff4dfff7c0  kleinste Adresse
nArr[0][0][1] = 8         addr. = 000000ff4dfff7c4
nArr[0][0][2] = 0         addr. = 000000ff4dfff7c8
nArr[0][1][0] = 4         addr. = 000000ff4dfff7cc
nArr[0][1][1] = 77        addr. = 000000ff4dfff7d0
nArr[0][1][2] = 90        addr. = 000000ff4dfff7d4
nArr[0][2][0] = 89        addr. = 000000ff4dfff7d8
nArr[0][2][1] = 67        addr. = 000000ff4dfff7dc
nArr[0][2][2] = 1         addr. = 000000ff4dfff7e0
nArr[1][0][0] = 11        addr. = 000000ff4dfff7e4
nArr[1][0][1] = 6         addr. = 000000ff4dfff7e8
nArr[1][0][2] = 7         addr. = 000000ff4dfff7ec
nArr[1][1][0] = 4         addr. = 000000ff4dfff7f0
nArr[1][1][1] = 77        addr. = 000000ff4dfff7f4
nArr[1][1][2] = 20        addr. = 000000ff4dfff7f8
nArr[1][2][0] = 19        addr. = 000000ff4dfff7fc
nArr[1][2][1] = 88        addr. = 000000ff4dfff800
nArr[1][2][2] = 11        addr. = 000000ff4dfff804
nArr[2][0][0] = 23        addr. = 000000ff4dfff808
nArr[2][0][1] = 78        addr. = 000000ff4dfff80c
nArr[2][0][2] = 90        addr. = 000000ff4dfff810
nArr[2][1][0] = 1         addr. = 000000ff4dfff814
nArr[2][1][1] = 9         addr. = 000000ff4dfff818
nArr[2][1][2] = 90        addr. = 000000ff4dfff81c
nArr[2][2][0] = 3         addr. = 000000ff4dfff820
nArr[2][2][1] = 67        addr. = 000000ff4dfff824
nArr[2][2][2] = 33        addr. = 000000ff4dfff828
nArr[3][0][0] = 34        addr. = 000000ff4dfff82c
nArr[3][0][1] = 4         addr. = 000000ff4dfff830
nArr[3][0][2] = 99        addr. = 000000ff4dfff834
nArr[3][1][0] = 68        addr. = 000000ff4dfff838
nArr[3][1][1] = 7         addr. = 000000ff4dfff83c
nArr[3][1][2] = 88        addr. = 000000ff4dfff840
nArr[3][2][0] = 10        addr. = 000000ff4dfff844
nArr[3][2][1] = 54        addr. = 000000ff4dfff848
nArr[3][2][2] = 5         addr. = 000000ff4dfff84c höchste Adresse
```