
Einführung in die Programmierung für Physiker

Andere Programmiersprachen

Marc Wagner

Institut für theoretische Physik
Johann Wolfgang Goethe-Universität Frankfurt am Main

WS 2013/14

Liste von Programmiersprachen

- Es existieren hunderte von Programmiersprachen.



The screenshot shows a web browser window displaying the Wikipedia page for 'Liste von Programmiersprachen'. The browser's address bar shows the URL 'de.wikipedia.org/wiki/Liste_von_Programmiersprachen'. The page features the Wikipedia logo and navigation links on the left, and a list of programming languages under the heading 'A [Bearbeiten]'. The list includes items like 'A', 'A#', 'A+', 'A-0, A-1, A-2, A-3 (ARITH-MATIC)', 'ABAP', 'ActionScript', 'Ada', 'ADbasic', 'Agilent VEE', 'AHDL', 'Aldor', 'Alef', 'Aleph', 'ALGOL (ALGOL 60, ALGOL W, ALGOL 68)', 'AML', 'AMOS', 'AMPL', 'AngelScript', 'APL', and 'AppleScript'. A search bar and navigation tabs are visible at the top of the page content.

Fortran

- **Fortran** ist eine imperative Programmiersprache, die sich insbesondere für numerische Berechnungen eignet; sie ist damit im wissenschaftlichen Bereich ebenfalls recht verbreitet.
- Starke Ähnlichkeiten zu **C**.
- In ihrer neuesten Version ist auch objektorientierte Programmierung möglich.

```
1.      PROGRAM factorial
2.      IMPLICIT NONE
3.
4.      INTEGER i1, n, result
5.
6. C   Berechne n! .
7.      n = 4
8.
9.      result = 1
10.
11.     DO i1 = 1, n
12.         result = result * i1
13.     END DO
14.
15. C   "6" entspricht Ausgabe am Bildschirm, "(I10)" entspricht Ausgabe eines Integers mit 10 Zeichen.
```

```
16.      WRITE(6, '(I10)') result
```

```
17.
```

```
18.      END
```

```
mwagner@laptop-tigger:~/lecture_ProgPhys/slides/tmp$ ls -l
```

```
insgesamt 4
```

```
-rw-rw-r-- 1 mwagner mwagner 365 Jan 23 12:44 factorial.f
```

```
mwagner@laptop-tigger:~/lecture_ProgPhys/slides/tmp$ gfortran -o factorial factorial.f
```

```
mwagner@laptop-tigger:~/lecture_ProgPhys/slides/tmp$ ls -l
```

```
insgesamt 16
```

```
-rwxrwxr-x 1 mwagner mwagner 9077 Jan 23 12:45 factorial
```

```
-rw-rw-r-- 1 mwagner mwagner 365 Jan 23 12:44 factorial.f
```

```
mwagner@laptop-tigger:~/lecture_ProgPhys/slides/tmp$ ./factorial
```

```
24
```

Lisp

- **Lisp** ist eine Familie von **funktionalen Programmiersprachen** (d.h. Programme bestehen ausschließlich aus Funktionen); eines ihrer Haupteinsatzgebiete ist die künstliche Intelligenz; in der Numerik spielt **Lisp** kaum eine Rolle.
- Geringe Ähnlichkeit zu **C**.

```
1. (defun factorial (n)
2.   (if (> n 1)
3.       (* n (factorial (- n 1)))
4.       1))
```

```
mwagner@teefax:~$ sbcl
This is SBCL 1.0.55.0.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at .

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.
* (load "factorial.lsp")

T
* (factorial 4)

24
* (quit)
mwagner@teefax:~$
```


Maple

- **Maple** ist ein Computeralgebrasystem, das vor allem als Unterstützung bei aufwändigen analytischen Rechnungen hilfreich ist.

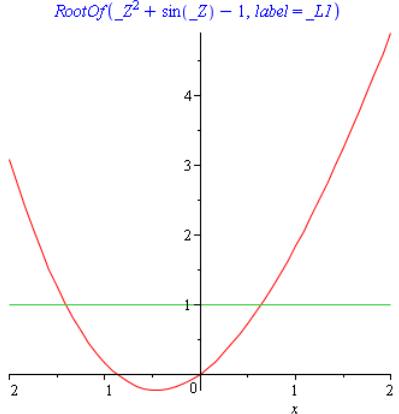
*Maple 12 - /home/mwagner/maple/C_lecture.mw - [Server 1]

File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

0.66, 4.7

Text Math Drawing Plot Animation

0.66, 4.7

> restart:
 > # *****
 # Integration
 # *****
 > int(x, x=0..1); (1)
 $\frac{1}{2}$
 > int(sin(x), x=0..Pi); (2)
 $\frac{2}{2}$
 > int(sin(x)^2, x=0..Pi); (3)
 $\frac{1}{2} \pi$
 > # *****
 # Loesen von Gleichungen
 # *****
 > solve(x^2=4, x); (4)
 $2, -2$
 > solve(x^2+sin(x)=1, x);
 plot([x^2+sin(x), 1], x=-2..2);
 fsolve(x^2+sin(x)=1, x);
 $\text{RootOf}(_Z^2 + \sin(_Z) - 1, \text{label} = _L1)$

 0.6367326508 (5)
 > # *****
 # Linear Algebra

Ready Memory: 1.49M Time: 0.72s Text Mode

*Maple 12 - /home/mwagner/maple/C_lecture.mw - [Server 1]

File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

1.05, -0.57

x

0.6367326508 (5)

```

> # *****
# Linear Algebra
# *****
> with(linalg);
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, enternmatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, invaryprod, inrbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, mataadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential, randommatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, subbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian]
> A := matrix(3, 3, [
  [1, 2, 3],
  [2, 4, 5],
  [3, 5, 6]
]);
y := vector(3, [-1, -2, -3]);
multiply(A, y);

```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

$$y = \begin{bmatrix} -1 & -2 & -3 \\ -14 & -25 & -31 \end{bmatrix} \quad (7)$$

```

> inverse(A);
multiply(inverse(A), A);
multiply(inverse(A), y);

```

$$\begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \quad (8)$$

```

> # *****
# Taylor-Entwicklung

```

• Ready Memory: 1.49M Time: 0.72s Text Mode

*Maple 12 - /home/mwagner/maple/C_lecture.mw - [Server 1]

File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

1.13, -0.36

```

[1, 2, 3],
[2, 4, 5],
[3, 5, 6]
]);

y := vector(3, [-1, -2, -3]);

multiply(A, y);

```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

$$y = \begin{bmatrix} -1 & -2 & -3 \\ -14 & -25 & -31 \end{bmatrix} \quad (7)$$

```

> inverse(A);
multiply(inverse(A), A);
multiply(inverse(A), y);

```

$$\begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \quad (8)$$

```

> # *****
> # Taylor-Entwicklung
> # *****
> cos_0 := series(cos(x), x, 4);
> cos_1 := series(cos(x), x=1, 4);

```

$$\cos_0 = 1 - \frac{1}{2}x^2 + O(x^4)$$

$$\cos_1 = \cos(1) - \sin(1)(x-1) - \frac{1}{2}\cos(1)(x-1)^2 + \frac{1}{6}\sin(1)(x-1)^3 + O((x-1)^4) \quad (9)$$

```

> # *****
> # Lösen von Differentialgleichungen
> # *****
> dsolve(diff(diff(x(t), t), t) = - omega^2*x(t));

```

$$x(t) = _C1 \sin(\omega t) + _C2 \cos(\omega t) \quad (10)$$

Ready Memory: 1.49M Time: 0.72s Text Mode

