DATOVE TYPY

Posloupnost

```
> #typ posloupnost (sequence)
> polynomial:=x^3-6*x^2+11*x-6;
```

$$polynomial := x^3 - 6\,x^2 + 11\,x - 6$$

```
> sequence:=op(polynomial);
```

$$sequence := x^3, -6\,x^2, 11\,x, -6$$

Jednotlive objekty jsou oddeleny carkami, objekty nemusi byt stejneho typu.

```
> whattype(sequence);
```

exprseq

```
> s:=1,4,9,16,25;
```

$$s := 1, 4, 9, 16, 25$$

```
> t:=sin, cos, tan;
```

$$t := \sin, \cos, \tan$$

```
> s,s;
```

1, 4, 9, 16, 25, 1, 4, 9, 16, 25

```
> empty:=NULL; #prazdna posloupnost
```

empty :=

Prikazy **op** a **nops** nemuzeme aplikovat na posloupnosti.

```
> op(s);

Error, invalid input: op expects 1 or 2 arguments, but received 5
```

Posloupnosti muzeme vytvaret (generovat) pomoci funkce seq:
Syntaxe: **seq**(f(i), i=m..n); generuje posloupnost f(m), f(m+1), .., f(n).

```
> seq(i^2, i=1..5);
```

1, 4, 9, 16, 25

```
> seq(ithprime(i), i=1..9);
```

2, 3, 5, 7, 11, 13, 17, 19, 23

(generuje prvnich 9 prvocisel)

Alternativne muzeme pouzit posloupnostniho operatoru $:

```
> x$4;
```

$$x, x, x, x$$

```
> seq(seq([i,j], i=1..2), j=3..4);
```

$$[1, 3], [2, 3], [1, 4], [2, 4]$$

Volani seq(f(i), i=vyraz) generuje posloupnost aplikaci funkce f na kazdy operand vyrazu.
Prikaz je ekvivalentni volani
seq(f(op(i,a)), i=1..nops(a)).

```
> seq(i^2, i={1,2,3,4,5});
```

$$1, 4, 9, 16, 25$$

```
> seq(i^2, i=x+y+z);
```

$$x^2, y^2, z^2$$

Jednotlive cleny posloupnosti muzeme vybirat pomoci operatoru [].

```
> %[2];
```

$$y^2$$

Prvni zprava.

```
> %%[-1];
```

$$z^2$$

Odkaz na vice clenu posloupnosti naraz:

```
> sequence:=v,w,x,y,z: sequence[2..4];
```

$$w, x, y$$

Jiny priklad pouziti: (generovani posloupnosti jmen)

```
> p||(1..5);
```

$$p1, p2, p3, p4, p5$$

```
> seq(p||i, i=1..5);
```

$$p1, p2, p3, p4, p5$$

Vyraz 1..5 je typu rozsah.

Mnozina

```
> restart;
```

"Posloupnost ve slozenych zavorkach"
Zadna data se nesmi vyskytovat vic nez jednou, system pouziva vnitrni system usporadani, ktere uzivatel nemuze ovlivnit (zavisi na adrese objektu v pameti).

```
> #typ mnozina (set)
  s:={1, 3, five, 2,4};
```

$$s := \{1, 2, 3, 4, five\}$$

```
> whattype(s);
```

$$set$$

```
> ‘empty set‘:={};
```

$$empty\ set := \{\ \}$$

Jednotlive prvky muzeme vybirat pomoci funkce op nebo operatoru [].

Pouziti op je mene efektivni, protoze vyhodnocuje celou mnozinu.

```
> op(1,s);
```

$$1$$

```
> s[1];
```

$$1$$

```
> op(1..3,s);
```

$$1, 2, 3$$

Zakladni mnozinove operace

Sjednoceni:

```
> {0,1,2,3} union {0,2,4,6};
```

$$\{0, 1, 2, 3, 4, 6\}$$

Rozdil:

```
> {0,1,2,3} minus {0,2,4,6};
```

$$\{1, 3\}$$

Prunik:

```
> {0,1,2,3} intersect {0,2,4,6};
```

$$\{0, 2\}$$

Pridani prvku do mnoziny:

```
> s union {x};
```

$$\{1, 2, 3, 4, \mathit{five}, x\}$$

Odstraneni prvku z mnoziny:

```
> s minus {1};
```

$$\{2, 3, 4, \mathit{five}\}$$

Nahrazeni prvku mnoziny:

```
> subsop(1=x,s);
```

$$\{2, 3, 4, \mathit{five}, x\}$$

Zjisteni, zda dany prvek patri do mnoziny.

```
> member(2, {0,1,2,3}, 'pos');
```

$$\mathit{true}$$

```
> pos;
```

$$3$$

Funkce powerset z baliku combinat generuje vsechny podmonoziny dane mnoziny ({1,2,3}).

```
> collection:=combinat[powerset](3);
```

$$\mathit{collection} := \{\{\,\}, \{1, 3\}, \{1\}, \{1, 2, 3\}, \{2, 3\}, \{3\}, \{2\}, \{1, 2\}\}$$

```
> nops(collection);
```

$$8$$

```
> collection[4];
```

$$\{1, 2, 3\}$$

```
> collection[6..8];
```

$$\{\{3\}, \{2\}, \{1, 2\}\}$$

```
> op(8, collection);
```

$$\{1, 2\}$$

K vybirani prvku podle nejakeho kriteria slouzi funkce select.

```
> die:=rand(-10..10): #generuje nahodne zvolena cela cisla z
  intervalu -10..10
> numberset:={seq(die(), i=1..10)};
```

$$numberset := \{-8, -6, -5, -4, 6, 7, 8, 10\}$$

Syntaxe: select(kriterium, mnozina, zvlastni_argumenty);
kde kriterium musi vzdy vracet true nebo false.

```
> select(isprime, numberset); #vybira prvocisla
```

$$\{7\}$$

```
> select(type, numberset, 'nonnegint');
```

$$\{6, 7, 8, 10\}$$

```
> select(x->x>-5, numberset);
```

$$\{-4, 6, 7, 8, 10\}$$

```
> remove(x->x>-5, numberset);
```

$$\{-8, -6, -5\}$$

Prikaz **map** aplikuje zadanou funkci na vsechny prvky mnoziny

```
> numbers:={0,Pi/3,Pi/2,Pi};
```

$$numbers := \{0, \pi, \frac{\pi}{3}, \frac{\pi}{2}\}$$

```
> map(g, numbers);
```

$$\{g(0), g(\pi), g\left(\frac{\pi}{3}\right), g\left(\frac{\pi}{2}\right)\}$$

```
> map(sin, numbers);
```

$$\{0, 1, \frac{\sqrt{3}}{2}\}$$

Seznam

```
> restart;
#typ SEZNAM  (list)
> s:=[x, x, x*y, x*(x-1)];
```

$$s := [x, x, x\,y, x\,(x-1)]$$

Rozdily oproti typu mnozina: stejne objekty se mohou vyskytovat vice nez jednou, zachovava se zadane
usporadani. Tedy [a,b,c], a [b,c,a] jsou dva ruzne seznamy.

```
> whattype(s);
```

$$list$$

Prevod na typ mnozina.

```
> m:=convert(s, 'set');
```

$$m := \{x, x\,y, x\,(x-1)\}$$

a zpet na seznam (odstraneni duplicit).

```
> convert(m, 'list');
```

$$[x, x\,y, x\,(x-1)]$$

```
> `empty list`:=[];
```

$$empty\ list := [\ ]$$

Prevod na typ posloupnost:

```
> op(s);
```

$$x, x, x\,y, x\,(x-1)$$

```
> [x$5];
```

$$[x, x, x, x, x]$$

Prace se seznamy:

```
> cl:=[black, red, green, yellow, blue, white];
```

$$cl := [black, red, green, yellow, blue, white]$$

Pocet prvku seznamu (delka seznamu):

```
> nops(cl);
```

$$6$$

Test, zda dany element patri do seznamu:

```
> member(indigo, cl);
```

$$false$$

```
> member(blue, cl, 'position');
```

$true$

```
> position;
```

$$5$$

Vyber jednotlivych elementu seznamu:

```
> cl[5];
```

$blue$

```
> op(5, cl);
```

$blue$

je mene efektivni zpusob.

```
> cl[3..6];
```

$[green, yellow, blue, white]$

```
> cl[-1];cl[-2];
```

$white$

$blue$

Posledni a predposledni prvek.

```
> cl[];
```

$black, red, green, yellow, blue, white$

```
> L := [1,[2,3],[4,[5,6],7],8,9];
```

$$L := [1, [2, 3], [4, [5, 6], 7], 8, 9]$$

```
> L[3,2,1];
```

$$5$$

```
> L[3][2][1];
```

$$5$$

Pridani prvku do seznamu:

```
> cl:=[op(cl),pink];
```

$$cl := [black, red, green, yellow, blue, white, pink]$$

Odstraneni prvku ze seznamu:

```
> cl:=subsop(7=NULL,cl);
```

$$cl := [black, red, green, yellow, blue, white]$$

Nahrazeni prvku v seznamu jinym:

```
> cl[5]:=purple;cl;
```

$$cl_5 := purple$$

$$[black, red, green, yellow, purple, white]$$

Tento zpusob prirazeni hodnoty prvku v seznamu je efektivnejsi, nez:

```
> cl:=subsop(5=indigo, cl);
```

$$cl := [black, red, green, yellow, indigo, white]$$

ale prime prirazeni hodnot prvkum seznamu je omezeno na seznamy o maximalne 100 polozkach.

```
>
```

Setrideni:

```
> sort(cl, lexorder);
```

$$[black, green, indigo, red, white, yellow]$$

Obraceni poradi prvku seznamu

```
> [seq(cl[-i], i=1..nops(cl))];
```

$$[white, indigo, yellow, green, red, black]$$

Secteni prvku seznamu:

```
> x:=[1,2,3,4,5,5];
```

$$x := [1, 2, 3, 4, 5, 5]$$

```
> add(i, i=x);
```

$$20$$

(add secte posloupnost prvku)

Slouceni dvou seznamu:

```
> X:=[seq(ithprime(i), i=1..6)];
```

$$X := [2, 3, 5, 7, 11, 13]$$

```
> Y:=[seq(i^2, i=1..6)];
```

$$Y := [1, 4, 9, 16, 25, 36]$$

```
> pair:=(x,y)->[x,y];
```

$$pair := (x, y) \to [x, y]$$

```
> P:=zip(pair, X,Y);
```

$$P := [[2, 1], [3, 4], [5, 9], [7, 16], [11, 25], [13, 36]]$$

Pokud maji seznamy rozdilnou delku, prikaz **zip** vytvori seznam delky odpovidajici kratsimu ze seznamu.

```
> zip(igcd, [7567,342,876], [34,756,213,346]);
```

$$[1, 18, 3]$$

```
> zip(‘+‘, [1,2,3], [4,5,6]);
```

$$[5, 7, 9]$$

Pole (array, Array)

```
> squares:=array(1..3);
```

$$squares := \text{array}(1 .. 3, [\ ])$$

Ve verzi 9 je k dispozici i datova struktura Array.

```
> sq:=Array(1..3);
```

$$sq := [0, 0, 0]$$

```
> squares[1]:=1; squares[2]:=2.0; squares[3]:=c;
```

$$squares_1 := 1$$

$$squares_2 := 2.0$$

$$squares_3 := c$$

```
> squares;
```

$$squares$$

```
> eval(squares);
```

$$[1, 2.0, c]$$

```
> sq[1]:=1;sq[2]:=2.0;sq[3]:=c;
```

$$sq_1 := 1$$

$$sq_2 := 2.0$$

$$sq_3 := c$$

```
> sq;
```

$$[1, 2.0, c]$$

Pro array plati pravidlo Last name evaluation, pro Array ne.

Proc ma Maple dve datove struktury pro pole? Hlavnim duvodem je to, ze balicek linalg je zalozen na pouziti array.

```
> cubes:=array(1..3, [1,8,27]);
```

```
> squares[2];
```

$$2.0$$

```
> squares;
```

$$squares$$

```
> eval(squares);
```

$$[1, 2.0, c]$$

```
> whattype(eval(squares));
```

$$array$$

Definice dvojdimenzionalniho pole (3 x 3).

```
> pwrs:=array(1..3,1..3,[]);
```

$$pwrs := \mathrm{array}(1 .. 3, 1 .. 3, [\ ])$$

```
> pwrs[1,1]:=1; pwrs[1,2]:=1;pwrs[1,3]:=1;
```

$$pwrs_{1,1} := 1$$

$$pwrs_{1,2} := 1$$

$$pwrs_{1,3} := 1$$

```
> pwrs[2,1]:=2: pwrs[2,2]:=4: pwrs[2,3]:=8:
  pwrs[3,1]:=3: pwrs[3,2]:=9: pwrs[3,3]:=27:
  print(pwrs);
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

```
> pwrs[2,3];
```

$$8$$

Nejdrive specifikujeme rozsah dvoudimenzionalniho pole, pote jeho prvky (neni nutno specifikovat vsechny).

```
> array(1..3, 1..2, {(1,1)=a, (1,2)=b, (2,1)=c, (2,2)=d});
```

$$\begin{bmatrix} a & b \\ c & d \\ ?_{3,1} & ?_{3,2} \end{bmatrix}$$

Pole muze byt konstruovano ze seznamu:

```
> restart;
> M:=array([[1-p, 2-q], [1-r, 2-s]]);
```

$$M := \begin{bmatrix} 1-p & 2-q \\ 1-r & 2-s \end{bmatrix}$$

```
> whattype(eval(M));
```

$$array$$

```
> pwrs2:=array(1..3, 1..3, [[1,1,1], [2,4,8], [3,9,27]]);
```

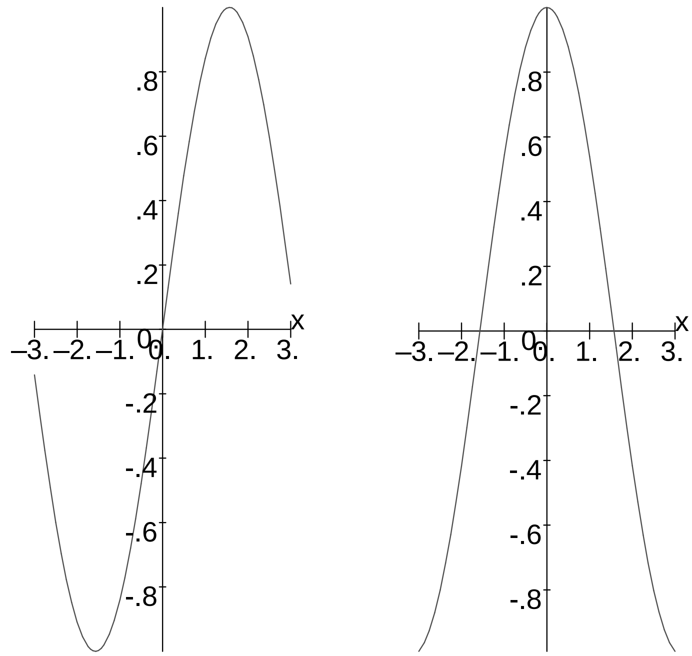
$$pwrs2 := \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

```
> obr:=array(1..2);
```

$$obr := \mathrm{array}(1 .. 2, [\ ])$$

```
> obr[1]:=plot(sin(x), x=-3..3):
> obr[2]:=plot(cos(x), x=-3..3):
> with(plots):display(obr);
```

```
Warning, the name changecoords has been redefined
```



Tabulka

Tabulky jsou zobecnenim datove struktury pole. Rozdil je v tom, ze indexem muze byt cokoliv (u pole pouze cela cisla).

```
> translate:=table([one=jedna,two=dve,three=tri]);
```

$$translate := \mathrm{table}([one = jedna, two = dve, three = tri])$$

```
> whattype(eval(translate));
```

$$table$$

```
> translate[two];
```

$$dve$$

```
> barva[cervena]:=red,rot;
```

$$barva_{cervena} := red, rot$$

```
> barva[modra]:=blue,blau;
```

$$barva_{modra} := blue, blau$$

```
> barva[zluta]:=yellow,gelb;
```

$$barva_{zluta} := yellow, gelb$$

```
> whattype(eval(barva));
```

table

```
> indices(barva);
```

[*zluta*], [*cervena*], [*modra*]

```
> entries(barva);
```

[*yellow*, *gelb*], [*red*, *rot*], [*blue*, *blau*]

Retezec

(string)

```
> "Toto je retezec.";
```

"Toto je retezec."

```
> r:="Toto je retezec.";
```

r := "Toto je retezec."

```
> r[6..-2];
```

"je retezec"

```
> length(r);
```

16

```
> whattype(r);
```

string

Linearni algebra

```
> restart;
> with(linalg);

Warning, the protected names norm and trace have been redefined
and unprotected
```

[*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*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stackmatrix*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

Matice muzeme zadavat bud primo jako dvou dimenzionalni pole nebo pomoci prikazu matrix z baliku linalag. V Maplu verze 9.5 je k dispozici i modernejsi balicek LinearAlgebra (definuje prikaz Matrix).

```
> with(LinearAlgebra);
Warning, the name GramSchmidt has been rebound
```

[*&x*, *Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, *ConditionNumber*, *ConstantMatrix*, *ConstantVector*, *Copy*, *CreatePermutation*, *CrossProduct*, *DeleteColumn*, *DeleteRow*, *Determinant*, *Diagonal*, *DiagonalMatrix*, *Dimension*, *Dimensions*, *DotProduct*, *EigenConditionNumbers*, *Eigenvalues*, *Eigenvectors*, *Equal*, *ForwardSubstitute*, *FrobeniusForm*, *GaussianElimination*, *GenerateEquations*, *GenerateMatrix*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*, *GramSchmidt*, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*, *HilbertMatrix*, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*, *IsOrthogonal*, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *LA_Main*, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *Map*, *Map2*, *MatrixAdd*, *MatrixExponential*, *MatrixFunction*, *MatrixInverse*, *MatrixMatrixMultiply*, *MatrixNorm*, *MatrixPower*, *MatrixScalarMultiply*, *MatrixVectorMultiply*, *MinimalPolynomial*, *Minor*, *Modular*, *Multiply*, *NoUserValue*, *Norm*, *Normalize*, *NullSpace*, *OuterProductMatrix*, *Permanent*, *Pivot*, *PopovForm*, *QRDecomposition*, *RandomMatrix*, *RandomVector*, *Rank*, *RationalCanonicalForm*, *ReducedRowEchelonForm*, *Row*, *RowDimension*, *RowOperation*, *RowSpace*, *ScalarMatrix*, *ScalarMultiply*, *ScalarVector*, *SchurForm*, *SingularValues*, *SmithForm*, *SubMatrix*, *SubVector*, *SumBasis*, *SylvesterMatrix*, *ToeplitzMatrix*, *Trace*, *Transpose*, *TridiagonalForm*, *UnitVector*, *VandermondeMatrix*, *VectorAdd*, *VectorAngle*, *VectorMatrixMultiply*, *VectorNorm*, *VectorScalarMultiply*, *ZeroMatrix*, *ZeroVector*, *Zip*]

```
> M:=array([[1-p, 2-q], [1-r, 2-s]]);
```

$$M := \begin{bmatrix} 1-p & 2-q \\ 1-r & 2-s \end{bmatrix}$$

```
> M:=matrix([[1-p, 2-q], [1-r, 2-s]]);
```

$$M := \begin{bmatrix} 1-p & 2-q \\ 1-r & 2-s \end{bmatrix}$$

```
> MM:=Matrix([[1-p, 2-q], [1-r, 2-s]]);
```

$$MM := \begin{bmatrix} 1-p & 2-q \\ 1-r & 2-s \end{bmatrix}$$

```
> v:=vector([1+a, 2+b, 3+c]);
```

$$v := [1+a, 2+b, 3+c]$$

```
> VV:=vector([1+a, 2+b, 3+c]);
```

$$VV := [1+a, 2+b, 3+c]$$

Definovani indexacni funkce

```
> h:=(i,j)->1/(i+j-x);
```

$$h := (i, j) \rightarrow \frac{1}{i+j-x}$$

```
> matrix(4,4,h);
```

$$\begin{bmatrix} \frac{1}{2-x} & \frac{1}{3-x} & \frac{1}{4-x} & \frac{1}{5-x} \\ \frac{1}{3-x} & \frac{1}{4-x} & \frac{1}{5-x} & \frac{1}{6-x} \\ \frac{1}{4-x} & \frac{1}{5-x} & \frac{1}{6-x} & \frac{1}{7-x} \\ \frac{1}{5-x} & \frac{1}{6-x} & \frac{1}{7-x} & \frac{1}{8-x} \end{bmatrix}$$

```
> matrix(3,3,0);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> v;
```

$$v$$

```
> eval(v);
```

$$[1+a, 2+b, 3+c]$$

```
> op(eval(v));
```

$$1\,..\,3,\ [1 = 1 + a,\ 2 = 2 + b,\ 3 = 3 + c]$$

```
> A:=matrix(2,2, [[a,b],[c,d]]);
```

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> AA:=Matrix(2,2, [[a,b],[c,d]]);
```

$$AA := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> B:=toeplitz([alpha, beta]);
```

$$B := \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix}$$

```
> BB:=ToeplitzMatrix([alpha, beta], symmetric);
```

$$BB := \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix}$$

Pro aritmetiku s maticemi pouzivame funkci evalm (evaluate using matrix arithmetic).

```
> evalm(A+B);
```

$$\begin{bmatrix} a + \alpha & b + \beta \\ c + \beta & d + \alpha \end{bmatrix}$$

```
> AA+BB;
```

$$\begin{bmatrix} a + \alpha & b + \beta \\ c + \beta & d + \alpha \end{bmatrix}$$

```
> evalm(3*A-2/7*B);
```

$$\begin{bmatrix} 3\,a - \frac{2\,\alpha}{7} & 3\,b - \frac{2\,\beta}{7} \\ 3\,c - \frac{2\,\beta}{7} & 3\,d - \frac{2\,\alpha}{7} \end{bmatrix}$$

```
> 3*AA-2/7*BB;
```

$$\begin{bmatrix} 3\,a - \frac{2\,\alpha}{7} & 3\,b - \frac{2\,\beta}{7} \\ 3\,c - \frac{2\,\beta}{7} & 3\,d - \frac{2\,\alpha}{7} \end{bmatrix}$$

```
> evalm(A-1);
```

$$\begin{bmatrix} a-1 & b \\ c & d-1 \end{bmatrix}$$

```
> AA-1;
```

$$\begin{bmatrix} a-1 & b \\ c & d-1 \end{bmatrix}$$

Pro nasobeni matic se pouziva operator &*. Za operatorem je nutno zadat mezeru.

```
> evalm(B &* A);
```

$$\begin{bmatrix} \alpha a+\beta c & \alpha b+\beta d \\ \beta a+\alpha c & \beta b+\alpha d \end{bmatrix}$$

```
> BB.AA;
```

$$\begin{bmatrix} \alpha a+\beta c & \alpha b+\beta d \\ \beta a+\alpha c & \beta b+\alpha d \end{bmatrix}$$

Mocniny:

```
> evalm(B^3);
```

$$\begin{bmatrix} (\alpha^2+\beta^2)\alpha+2\alpha\beta^2 & (\alpha^2+\beta^2)\beta+2\alpha^2\beta \\ (\alpha^2+\beta^2)\beta+2\alpha^2\beta & (\alpha^2+\beta^2)\alpha+2\alpha\beta^2 \end{bmatrix}$$

```
> BB^3;
```

$$\begin{bmatrix} (\alpha^2+\beta^2)\alpha+2\alpha\beta^2 & (\alpha^2+\beta^2)\beta+2\alpha^2\beta \\ (\alpha^2+\beta^2)\beta+2\alpha^2\beta & (\alpha^2+\beta^2)\alpha+2\alpha\beta^2 \end{bmatrix}$$

```
> toeplitz([1,2,3]);
```

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

Vypocet determinantu.

```
> det(%);
```

$$8$$

```
> A:=matrix([[1,0,0,1], [1,0,1,1], [0,0,1,0]]);
```

$$A := \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> AA:=<<1|0|0|1>, <1|0|1|1>, <0|0|1|0>>;
```

$$AA := \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> <1,2,3>; #sloupcovy vektor
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> <1|2|3>; #radkovy vektor
```

$$[1, 2, 3]$$

```
> < <1,2,3> | <4,5,6>>; #matice zadana sloupci
```

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

```
> <<1|2|3>, <4|5|6>>; #matice zadana radky
```

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

Hodnost matice:

```
> rank(A);
```

$$2$$

```
> Rank(AA);
```

$$2$$

Gaussova eliminace:

```
> A:=matrix([[1,1,3,-3],[5,5,13,-17],[3,1,7,-11]]);
```

$$A := \begin{bmatrix} 1 & 1 & 3 & -3 \\ 5 & 5 & 13 & -17 \\ 3 & 1 & 7 & -11 \end{bmatrix}$$

```
> gausselim(A);
```

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

```
> gaussjord(A);
```

$$\begin{bmatrix} 1 & 0 & 0 & -6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Rozmery matice a vektoru zjistime prikazy

```
> rowdim(A);
```

$$3$$

```
> coldim(A);
```

$$4$$

```
> v:=vector([1,2,3]):vectdim(v);
```

$$3$$

```
> submatrix(A, 2..3, 1..2);
```

$$\begin{bmatrix} 5 & 5 \\ 3 & 1 \end{bmatrix}$$

Last Name Evaluation

```
> restart;
> R:=matrix(2,2,[[cos(alpha),-sin(alpha)],[sin(alpha),cos(alpha)]]
  );
```

$$R := \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

```
> R;
```

$$R$$

```
> whattype(R);
```

$$symbol$$

```
> eval(R);
```

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

```
> whattype(eval(R));
```

$$array$$

```
> alpha:=Pi/4;
```

$$\alpha := \frac{\pi}{4}$$

```
> eval(R);
```

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

```
> R[1,2];
```

$$-\frac{\sqrt{2}}{2}$$

```
> map(eval,R);
```

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

```
>
> T:=S;
```

$$T := S$$

```
> S:=R;
```

$$S := R$$

```
> eval(T,1);#hodnota T
```

$$S$$

```
> eval(T,2);#hodnota S
```

$$R$$

```
> eval(T,3);#hodnota R
```

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

```
> map(eval,T);#vyhodnoceni jednotlivych prvku
```

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

```
> matrix([[1/2*2^(1/2), -1/2*2^(1/2)], [1/2*2^(1/2),
  1/2*2^(1/2)]]);
```

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

```
> T;
```

$$R$$

```
> alpha:='alpha';
```

$$\alpha := \alpha$$

```
> R[1,2];S[1,2];T[1,2];
```

$$-\sin(\alpha)$$
$$-\sin(\alpha)$$
$$-\sin(\alpha)$$

```
> S[2,1]:=0:
> eval(R), eval(S), eval(T);
```

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ 0 & \cos(\alpha) \end{bmatrix}, \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ 0 & \cos(\alpha) \end{bmatrix}, \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ 0 & \cos(\alpha) \end{bmatrix}$$

```
>
> S:=copy(R);
```

$$S := \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ 0 & \cos(\alpha) \end{bmatrix}$$

```
> S[1,2]:=1:
> eval(R), eval(S);
```

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ 0 & \cos(\alpha) \end{bmatrix}, \begin{bmatrix} \cos(\alpha) & 1 \\ 0 & \cos(\alpha) \end{bmatrix}$$

```
>
>
```