Kurs Komputerowy S
System Symboliczny
Mathematica

Programowanie

Funkcje logiczne (analiza własności)

IntegerQ[wartość]
NumberQ[wartość]
NumericQ[wartość]
OddQ[wartość]
EvenQ[wartość]
PrimeQ[wartość]
VectorQ[wartość]
MatrixQ[wartość]
ArrayQ[wartość]
MemberQ[wartość,lista]
LetterQ[wartość]
DigitQ[wartość]
StringQ[wartość]
SyntaxQ[wartość]

IntegerQ[x]

```
IntegerQ[2.]
```

```
False
```

```
IntegerQ[2.5]
```

```
False
```

NumberQ[x]

```
NumberQ[2]
```

```
True
```

```
NumberQ[Sqrt[2]]
```

```
False
```

NumericQ[x]

```
NumericQ[2]
```

```
True
```

```
NumericQ[Sqrt[2]]
```

```
True
```

```
x = 3
```

```
3
```

```
x
```

```
abc
```

```
abc
```

```
NumericQ[abc]
```

```
False
```

VectorQ[x]

```
VectorQ[{1, 2, 3}]
```

```
True
```

```
VectorQ[{{1, 3}, {1, 2}, {3, 2}}]
```

```
False
```

MatrixQ[x]

```
MatrixQ[{1, 2, 3}]
```

```
False
```

```
MatrixQ[{{1, 3}, {1, 2}, {3, 2}}]
```

```
False
```

```
MatrixQ[{{{1, 3}, {1, 2}, {3, 2}}, {{1, 3}, {1, 2}, {3, 2}}, {{1, 3}, {1, 2}, {3, 2}}}]
```

```
False
```

ArrayQ[x]

```
ArrayQ[{1, 2, 3}]
```

```
True
```

```
ArrayQ[{{1, 3}, {1, 2}, {3, 2}}]
```

```
True
```

```
ArrayQ[{{1, 3}, {1}, {3, 2}}]
```

```
False
```

```
? *Q
```

▼ System`

AcyclicGraphQ	ImageQ	PermutationCyclesQ
AlgebraicIntegerQ	IndependentEdgeSetQ	PermutationListQ
AlgebraicUnitQ	IndependentVertexSetQ	PolynomialQ
ArgumentCountQ	InexactNumberQ	PositiveDefiniteMatrixQ
ArrayQ	IntegerQ	PossibleZeroQ
AtomQ	IntervalMemberQ	PrimePowerQ
BinaryImageQ	InverseEllipticNomeQ	PrimeQ
BipartiteGraphQ	IrreduciblePolynomialQ	QHypergeometricPFQ
CompleteGraphQ	IsomorphicGraphQ	QuadraticIrrationalQ
ConnectedGraphQ	LegendreQ	RootOfUnityQ
ContinuousTimeModelQ	LetterQ	SameQ
ControllableModelQ	LinkConnectedQ	SatisfiableQ
CoprimeQ	LinkReadyQ	SimpleGraphQ
DigitQ	ListQ	SquareFreeQ
DirectedGraphQ	LoopFreeGraphQ	StringFreeQ
DirectoryQ	LowerCaseQ	StringMatchQ
DiscreteTimeModelQ	MachineNumberQ	StringQ
DistributionParameterQ	MarcumQ	SymmetricMatrixQ
EdgeCoverQ	MatchLocalNameQ	SyntaxQ
EdgeQ	MatchQ	TautologyQ
EllipticNomeQ	MatrixQ	TensorQ
EmptyGraphQ	MemberQ	TreeGraphQ
EulerianGraphQ	NameQ	TrueQ
EvenQ	NumberQ	UndirectedGraphQ
ExactNumberQ	NumericQ	UnsameQ
FileExistsQ	ObservableModelQ	UpperCaseQ
FreeQ	OddQ	ValueQ
GraphQ	OptionQ	VectorQ
GroupElementQ	OrderedQ	VertexCoverQ
HamiltonianGraphQ	OutputControllableModelQ	VertexQ
HermitianMatrixQ	PartitionsQ	WeightedGraphQ
HypergeometricPFQ	PathGraphQ	

▼ PacletManager`

PacletNewerQ

Wprowadzanie danych i wypisywanie

Input[tekst]
InputString[tekst]
Pause[czas]
Print[wyrażenie]
PrintTemporary[wyrażenie]

```
a = Input["Podaj liczbę"]
```

```
$Canceled
```

```
a
```

```
345
```

```
a = Input["a=", 5]
```

```
5
```

```
a
```

```
5
```

```
a = 3
Pause[5]
b = 4
```

```
3
```

```
4
```

```
c = Print["a=", a]
```

```
a=3
```

```
c
```

```
d
```

```
d
```

```
Out[29] * 4
```

```
12
```

```
Do[PrintTemporary["poczekaj ", i, " chwile"]; Pause[0.5], {i, 1, 5}]; a
```

```
3
```

Operacje

```
Increment[x]        (*  x++  *)
PreIncrement[x]     (*  ++x  *)
Decrement[x]        (*  x--  *)
PreDecrement[x]     (*  --x  *)
AddTo               (*  x+=y  *)
SubtractFrom        (*  x-=  *)
TimesBy             (*  x*=y  *)
DivideBy            (*  x/=y  *)
```

```
x = 1; x++
```

```
1
```

```
x
```

```
2
```

```
y = 1; ++y
```

```
2
```

```
y
```

```
2
```

```
a = 3; b = 2;
a += b
```

```
5
```

```
a
```

```
5
```

■ Warunki

```
w == 5
```

```
w == 5
```

```
w === 4
```

```
False
```

If[warunek,jeżeli_prawda,jeżeli_fałsz,jeżeli_coś_innego]
Which[warunek_1,wyr_1,warunek_2,wyr_2,...]
Switch[wyrażenie,wart_1,wyr_1,wart_2,wyr_2,...]

```
b = 3
```

```
3
```

```
a = 5;
q = If[a == 5, b = b^2, b = b / a];
```

```
q
```

```
9
```

```
b
```

```
9
```

```
Out[52]
```

```
5
```

```
a = 6;
If[a == 5, b = b^2, b = b / a]
```

$$\frac{3}{2}$$

```
a =.;
If[a == 5, b = b^2, b = b / a]
```

$$\text{If}\left[a == 5,\ b = b^2,\ b = \frac{b}{a}\right]$$

```
a =.;
If[a == 5, b = b^2, b = b / a, b = 2]
```

```
2
```

```
b
```

```
2
```

```
a = 5;
Which[a == 5, b = b^2, a == 6, b = b / a, a == 2, b = 2]
```

```
4
```

```
a = 6;
Which[a < 5, b = b^2, a == 6, b = b / a, a == 2, b = 2]
```

$$\frac{2}{3}$$

```
a = 2;
Which[a < 5 && a > -3, b = b^2, a == 6, b = b / a, a == 2, b = 2]
```

```
2
```

```
a = 3;
Which[a == 5, b = b^2, a == 6, b = b / a, a == 2, b = 2]
```

```
a = 5;
Switch[a, 5, b = b^2, 6, b = b / a, 2, b = 2]
```

```
4
```

■ Pętle

Do[wyrażenie,iterator]
While[warunek,wyrażenie]
For[wart_poczatkowe,warunek,zmiana_iteratora,wyrażenie]

```
Clear[a];
```

```
q
```

```
9
```

```
Do[Print["i=", i]; q = i, {i, 5, 1, -1}]
```

```
q
```

```
5
```

```
i
```

```
i
```

```
Do[Print["i=", i]; a = i, {i, 5}]
```

```
i=1
i=2
i=3
i=4
i=5
```

```
a
```

```
5
```

```
i =.
```

```
i = 0; While[i < 5, i++; Print["i=", i]]
```

```
i=1
i=2
i=3
i=4
i=5
```

```
i
```

```
5
```

```
i = 1; While[i ≤ 5, Print["i=", i]; i++]
```

```
i=1
i=2
i=3
i=4
i=5
```

```
i =.
```

```
For[i = 1, i ≤ 5, i++, Print["i=", i]]
```

```
i=1
i=2
i=3
i=4
i=5
```

Kontrola

Break[ ]
Continue[ ]
Return[wyrażenie]
Goto[etykieta] Label[etykieta]
Monitor[wyrażenie,zmienna]
Throw[wartość]
Catch

```
Do[If[i == 3, Break[]]; Print["i=", i], {i, 1, 5}]
```

```
i=1
i=2
```

```
3
```

```
Do[If[i == 3, Continue[]]; Print["i=", i], {i, 1, 5}]
```

```
i=1
i=2
i=4
i=5
```

```
Do[If[i == 3, Return[i]]; Print["i=", i], {i, 1, 5}]
```

```
i=1
i=2
```

```
3
```

```
Monitor[i = 0; While[i < 20, Pause[0.1]; i++]; i^2, i]
```

```
400
```

```
Catch[Do[If[i! > 10^10, Throw[i]], {i, 100}]]
```

```
14
```

```
14!
```

```
87 178 291 200
```

```
13!
```

```
6 227 020 800
```

```
?*Cell*
```

▼ System`

AllowInlineCells	CellOpen
ButtonCell	CellPasswords
Cell	CellPrint
CellAutoOverwrite	CellProlog
CellBaseline	CellSize
CellBoundingBox	CellStyle
CellBracketOptions	CellTags
CellChangeTimes	CellularAutomaton
CellContents	DefaultDuplicateCellStyle
CellContext	DefaultNewCellStyle
CellDingbat	DefaultNewInlineCellStyle
CellDynamicExpression	DockedCells
CellEditDuplicate	EditCellTagsSettings
CellElementsBoundingBox	EvaluationCell
CellElementSpacings	ExpressionCell
CellEpilog	GeneratedCell
CellEvaluationDuplicate	InitializationCell
CellEvaluationFunction	InitializationCellEvaluation
CellEventActions	InitializationCellWarning
CellFrame	NotebookResetGeneratedCells
CellFrameColor	PasteBoxFormInlineCells
CellFrameLabelMargins	PrivateCellOptions

CellFrameLabels	SelectionCell
CellFrameMargins	SelectionCellCreateCell
CellGroup	SelectionCellDefaultStyle
CellGroupData	SelectionCellParentStyle
CellGrouping	SelectionCreateCell
CellGroupingRules	SelectionDuplicateCell
CellHorizontalScrolling	SelectionEvaluateCreateCell
CellID	ShowCellBracket
CellLabel	ShowCellLabel
CellLabelAutoDelete	ShowCellTags
CellLabelMargins	ShowClosedCellArea
CellLabelPositioning	TextCell
CellMargins	VisibleCell
CellObject	

▼ PacletManager`

GetVirtualCellGroup

CellEvaluationFunction is an option to Cell which gives a function to be applied to every expression from the cell that is sent to the kernel for ordinary evaluation. ≫

```
Print["a=4"]
```

```
a=4
```

```
Do[CellPrint[i], {i, 1, 10}]
```

1

2

3

4

5

6

7

8

9

10

```
t = {};
Do[AppendTo[t, {}];
 Do[AppendTo[t[[i]], 0], {j, 5}], {i, 1, 5}]
t
```

```
{{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}}
```

■ Przekazywanie zmiennych

f[zmienna_]
f[zmienna__]
f[zmienna___]
f[zmienna_:WartPocz]

```
Clear[f, x, y]
```

```
f[x] = x^4
```

$$x^4$$

```
f[1]
f[2]
f[x]
f[y]
```

```
f[1]
```

```
f[2]
```

$$x^4$$

```
f[y]
```

```
Clear[f]
```

```
x = 3;
f[x] = x^4
```

```
81
```

```
f[1]
f[2]
f[3]
f[x]
f[y]
```

```
f[1]
```

```
f[2]
```

```
81
```

```
81
```

```
f[y]
```

```
Clear[f]
```

```
x = 3;
f[x_] = x^4
```

```
81
```

```
f[1]
f[2]
f[y]
```

```
81
```

```
81
```

```
81
```

```
Clear[f]
```

```
x = 3;
f[x_] := x^4
```

```
f[1]
f[2]
f[x]
f[y]
```

```
1
```

```
16
```

```
81
```

```
y^4
```

```
Clear[f]
```

```
x = 3;
f[x_] := (y = x^4; {x, y})
```

```
f[5]
```

```
{5, 625}
```

```
f[1]
f[2]
f[y]
f[]
f[x, y]
```

```
{1, 1}
```

```
{2, 16}
```

```
{16, 65 536}
```

```
f[]
```

```
f[3, 65 536]
```

```
g[x__] := {Length[{x}], Flatten[{x}] / 2}
```

```
g[1]
g[1, 2]
g[]
```

$$\left\{1, \left\{\frac{1}{2}\right\}\right\}$$

$$\left\{2, \left\{\frac{1}{2}, 1\right\}\right\}$$

```
g[]
```

```
g[1, 2]
```

$$\left\{2, \left\{\frac{1}{2}, 1\right\}\right\}$$

```
g[{1, 2}]
```

$$\left\{1, \left\{\frac{1}{2}, 1\right\}\right\}$$

```
g[1, 2, 3, 4, 5]
```

$$\left\{5, \left\{\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}\right\}\right\}$$

```
h[x___] := Print["Ilość podanych parametrów: ", Length[{x}]]
```

```
h[1]
h[1, 2]
h[]
```

```
Ilość podanych parametrów: 1
```

```
Ilość podanych parametrów: 2
```

```
Ilość podanych parametrów: 0
```

```
Options[Roots]
```

```
{Cubics → True, Eliminate → False, EquatedTo → Null,
 Modulus → 0, Multiplicity → 1, Quartics → True, Using → True}
```

```
Remove[f]
```

```
f[x___] := Print["Ilość parametrów: ", Length[{x}]]
f[x_] := x^2
f[x__] := {x}
```

```
f[1]
f[2]
f[1, 2]
f[2, 3, 4]
f[]
```

```
1
```

```
4
```

```
{1, 2}
```

```
{2, 3, 4}
```

```
Ilość parametrów: 0
```

```
f2[x_] := 2 x
```

```
f2[1]
```

```
2
```

```
f2[]
```

```
f2[]
```

```
f1[x_: 4] := 5 x
```

```
f1[2]
f1[]
```

```
10
```

```
20
```

```
Clear[g]
```

```
g[x_: 4, y_: 7] := Print["x=", x, "  y=", y]
```

```
g[2, 5]
```

```
x=2  y=5
```

```
g[]
```

```
x=4  y=7
```

```
g[9]
```

```
x=9  y=7
```

■ Warunkowe definicje

f[x_Head]
f[x_] := wyrażenie /; test

```
Clear[f, g, h, h1]
```

```
a = 2 x + 3 y^2 - Sqrt[x^y]
```

$$2\,x - \sqrt{x^y} + 3\,y^2$$

```
? a
```

```
Global`a
```

$$a = 2\,x - \sqrt{x^y} + 3\,y^2$$

```
FullForm[a]
```

```
Plus[Times[2, x],
 Times[-1, Power[Power[x, y], Rational[1, 2]]], Times[3, Power[y, 2]]]
```

```
b = {1, 2, 3, 4, 5}
```

```
{1, 2, 3, 4, 5}
```

```
FullForm[b]
```

```
List[1, 2, 3, 4, 5]
```

```
a[[2, 2, 1, 1]] = 2
```

```
2
```

```
a
```

$$-\sqrt{2^y} + 2\,x + 3\,y^2$$

```
Head[a]
```

```
Plus
```

```
f[x_Integer] := Print["podałeś liczbę: ", x]
```

```
f[1]
```

```
podałeś liczbę: 1
```

```
a = {4.5}
```

```
{4.5}
```

```
Head[a]
```

```
List
```

```
f[1]
f[1.3]
```

```
podałeś liczbę: 1
```

```
f[1.3]
```

```
f[x_Real] := Print["Twoja liczba to: ", x]
```

```
f[1]
f[1.3]
```

```
podałeś liczbę: 1
```

```
Twoja liczba to: 1.3
```

```
f[x_List] := Length[x]
```

```
f[5]
f[5.5]
f[{5, 5}]
```

```
podałeś liczbę: 5
```

```
Twoja liczba to: 5.5
```

```
2
```

```
g[x_] := Print["Liczba parzysta: ", x] /; EvenQ[x]
```

```
g[1]
g[2]
```

```
g[1]
```

```
Liczba parzysta: 2
```

```
g[x_] := Print["A teraz nieparzysta: ", x] /; OddQ[x]
```

```
g[x_] := Print["Liczba rzeczywista (z przecinkiem): ", x] /; Not[IntegerQ[x]]
```

```
g[2.4]
g[2]
g[1]
```

```
Liczba rzeczywista (z przecinkiem): 2.4
```

```
Liczba parzysta: 2
```

```
A teraz nieparzysta: 1
```

```
?g
```

```
Global`g
```

```
g[x_] := Print[Liczba parzysta: , x] /; EvenQ[x]

g[x_] := Print[A teraz nieparzysta: , x] /; OddQ[x]

g[x_] := Print[Liczba rzeczywista (z przecinkiem): , x] /; ! IntegerQ[x]
```

```
Clear[h]
```

```
h[x_] := Print["Parzysta: ", x] /; EvenQ[x]
h[x_] := Print["Nieparzysta: ", x] /; OddQ[x]
h[x_] := Print["Liczba: ", x] /; NumberQ[x]
```

```
h[1.4]
h[1]
h[2]
```

```
Liczba: 1.4
```

```
Nieparzysta: 1
```

```
Parzysta: 2
```

```
Clear[h]
```

```
h[x_] := Print["Liczba: ", x] /; NumberQ[x]
h[x_] := Print["Parzysta: ", x] /; EvenQ[x]
h[x_] := Print["Nieparzysta: ", x] /; OddQ[x]
```

```
h[1.4]
h[1]
h[2]
```

```
Liczba: 1.4
```

```
Liczba: 1
```

```
Liczba: 2
```

Procedury i funkcje

Module[{zmienne_lokalne},wyrażenie]
Block[{zmienne_lokalne},wyrażenie]
With[{stale},wyrażenie]
Function[wyrażenie]

```
a = 5
b = 4
```

```
5
```

```
4
```

```
Module[{}, a = 10; a = b^4]
```

```
256
```

```
a
```

```
256
```

```
Clear[a]
```

```
a = 4
```

```
4
```

```
Block[{a = 10}, Print["a=", a]; a = b^4; Print["a=", a]; a]
```

```
a=10
```

```
a=256
```

```
256
```

```
a
```

```
4
```

```
Clear[i, a];
m = i^2
```

```
i^2
```

```
Module[{i = a}, i + m]
```

```
a + i^2
```

```
Block[{i = a}, i + m]
```

```
a + a^2
```

```
With[{i = 4}, i^4 + 5]
```

```
261
```

```
p = Function[u, u^3]
```

```
Function[u, u^3]
```

```
p[2]
```

```
8
```

```
Function[#^3]
```

```
#1^3 &
```

```
%[6]
```

```
216
```

```
#^3 &
```

```
#1^3 &
```

```
%[4]
```

```
64
```

```
Clear[f, g]
```

```
f[x_, y_] := x^3 + y / 2
```

```
f[2, 6]
```

```
11
```

```
g = Function[{x, y}, x^3 + y / 2]
```

$$\text{Function}\left[\{x, y\}, x^3 + \frac{y}{2}\right]$$

```
g[2, 6]
```

```
11
```

■ Właściwości nowych poleceń

```
a = x + y
```

```
x + y
```

```
b = y + x
```

```
x + y
```

```
?? Plus
```

x + y + z represents a sum of terms. ≫

```
Attributes[Plus] = {Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}

Default[Plus] := 0
```

```
Plus = 4
```

Set::wrsym : Symbol Plus is Protected. ≫

```
4
```

Attributes[polecenie]
SetAttributes[polecenie, właściwość]
ClearAttrbutes[polecenie,właściwość]

```
Attributes[Plus]
```

```
{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
```

```
Plus[2, q] = 5
```

Set::write : Tag Plus in 2 + q is Protected. ≫

```
5
```

```
Attributes[f]
```

```
{}
```

```
? f
```

Global`f

$$f[x\_, y\_] := x^3 + \frac{y}{2}$$

```
Attributes[f] = Flat
```

```
Flat
```

```
Attributes[f]
```

```
{Flat}
```

```
Attributes[f] = {Listable, Orderless}
```

```
{Listable, Orderless}
```

```
Attributes[f]
```

```
{Listable, Orderless}
```

```
SetAttributes[f, Flat]
```

```
Attributes[f]
```

```
{Flat, Listable, Orderless}
```

```
ClearAttributes[f, Orderless]
```

```
Attributes[f]
```

```
{Flat, Listable}
```

Właściwości

Constant

```
Clear[f, x, b, g]
```

```
D[f[x], x]
```

```
f′[x]
```

```
Attributes[f] = Constant
```

```
Constant
```

```
D[f[x], x]
```

```
0
```

```
ClearAttributes[f, Constant]
```

Flat

```
Clear[a];
```

```
f[a, f[b], c]
```

```
f[a, f[b], 5]
```

```
Attributes[f] = Flat
```

```
Flat
```

```
f[a, f[b], c]
```

```
f[a, b, 5]
```

```
ClearAttributes[f, Flat]
```

HoldAll

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[6, 8, 2]
```

```
Attributes[f] = HoldAll
```

```
HoldAll
```

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[Evaluate[2 + 4], 6 + 2, 5 - 3]
```

```
f[6, 6 + 2, 5 - 3]
```

```
ClearAttributes[f, HoldAll]
```

HoldAllComplete

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
Attributes[f] = HoldAllComplete
```

```
HoldAllComplete
```

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[2 + 4, 6 + 2, 5 - 3]
```

```
f[Evaluate[2 + 4], 6 + 2, 5 - 3]
```

```
f[Evaluate[2 + 4], 6 + 2, 5 - 3]
```

```
ClearAttributes[f, HoldAllComplete]
```

HoldFirst

HoldRest

Listable

```
f[{a, b, c}]
```

```
f[{a, b, 5}]
```

```
Attributes[f] = Listable
```

```
Listable
```

```
f[{a, b, c}]
```

```
{f[a], f[b], f[5]}
```

```
ClearAttributes[f, Listable]
```

Locked

```
Attributes[f] = {Constant, Flat, Listable}
```

```
{Constant, Flat, Listable}
```

```
Attributes[f]
```

```
{Constant, Flat, Listable}
```

```
Attributes[f] = {Constant, Flat}
```

```
{Constant, Flat}
```

```
Attributes[f]
```

```
{Constant, Flat}
```

```
SetAttributes[f, Locked]
```

```
Attributes[f]
```

```
{Constant, Flat, Locked}
```

```
Attributes[f] = {Constant, Flat, Listable}
```

Attributes::locked : Symbol f is locked. ≫

```
{Constant, Flat, Listable}
```

```
Attributes[f]
```

```
{Constant, Flat, Locked}
```

```
ClearAttributes[f, Locked]
```

Attributes::locked : Symbol f is locked. ≫

NHoldAll

```
g[2 + 3, 2 + Pi]
```

```
g[5, 2 + π]
```

```
N[%]
```

```
g[5., 5.14159]
```

```
Attributes[g] = NHoldAll
```

```
NHoldAll
```

```
g[2 + 3, 2 + Pi]
```

```
g[5, 2 + π]
```

```
N[%]
```

```
g[5, 2 + π]
```

```
ClearAttributes[g, NHoldAll]
```

NHoldFirst

NHoldRest

NumericFunction

```
Clear[g]
```

```
NumericQ[g[2, 3]]
```

```
False
```

```
Attributes[g] = NumericFunction
```

```
NumericFunction
```

```
NumericQ[g[2, 3]]
```

```
True
```

```
g[2, 3]
```

```
g[2, 3]
```

```
ClearAttributes[g, NumericFunction]
```

OneIdentity

Orderless

```
g[b, c, a]
```

```
g[b, 5, a]
```

```
Attributes[g] = Orderless
```

```
Orderless
```

```
g[b, c, a]
```

```
g[5, a, b]
```

```
g[5, 2, 4]
```

```
g[2, 4, 5]
```

```
ClearAttributes[g, Orderless]
```

Protected

```
Attributes[g]
```

```
{}
```

```
g[x_] := x^2
```

```
g[4]
```

```
16
```

```
g[x_] := 2 + x
```

```
g[4]
```

```
6
```

```
Attributes[g] = Protected
```

```
Protected
```

```
g[x_] := 5 x
```

SetDelayed::write : Tag g in g[x_] is Protected. ≫

```
$Failed
```

```
? g
```

```
Global`g
```

```
Attributes[g] = {Protected}

g[x_] := 2 + x
```

```
ClearAttributes[g, Protected]
```

```
g[x_] := 5 x
```

```
g[4]
```

```
20
```

ReadProtected

```
Attributes[g] = Flat
```

```
Flat
```

```
? g
```

```
Global`g
```

```
Attributes[g] = {Flat}

g[x_] := 5 x
```

```
SetAttributes[g, ReadProtected]
```

```
?g
```

```
Global`g
```

```
Attributes[g] = {Flat, ReadProtected}
```

```
??g
```

```
Global`g
```

```
Attributes[g] = {Flat, ReadProtected}
```

```
ClearAttributes[g, ReadProtected]
```

SequenceHold

```
g[a, Sequence[b, c]]
```

```
g[a, b, 5]
```

```
Attributes[g] = SequenceHold
```

```
SequenceHold
```

```
g[a, Sequence[b, c]]
```

```
g[a, Sequence[b, 5]]
```

```
ClearAttributes[g, SequenceHold]
```

Stub

Temporary

```
Module[{}, Attributes[x]]
```

```
{}
```

```
Module[{x}, Attributes[x]]
```

```
{Temporary}
```

■ Opis i komunikaty o bledach

polecenie::usage
polecenie::nazwa
Message[polecenie::nazwa]
On[polecenie::nazwa]
Off[polecenie::nazwa]

```
Remove[g]
```

```
?g
```

Information::notfound : Symbol g not found. ≫

```
Remove[a]
```

```
?a
```

Information::notfound : Symbol a not found. ≫

```
a
```

```
a
```

```
?a
```

```
Global`a
```

```
g[x_] := Sqrt[x]
```

```
?g
```

```
Global`g
```

$$g[x\_] := \sqrt{x}$$

```
g::usage =
  "Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.";
```

```
?g
```

Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.

```
??g
```

Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.

```
g[x_] := √x
```

```
g[-3]
```

```
i √3
```

```
g[x_] := If[x < 0, Message[g::neg], Sqrt[x]]
```

```
g[-3]
```

g::neg : –– Message text not found ––

```
g::neg = "Jako argument należy podać wartość nieujemną.";
```

```
g[-4]
```

g::neg : Jako argument należy podać wartość nieujemną.

```
Print["a"]
```

```
a
```

```
Sqrt[2, 3, 4, 6]
```

Sqrt::argx : Sqrt called with 4 arguments; 1 argument is expected. ≫

```
Sqrt[2, 3, 4, 6]
```

```
Sqrt[2, 3, 4]
```

Sqrt::argx : Sqrt called with 3 arguments; 1 argument is expected. ≫

```
Sqrt[2, 3, 4]
```

```
g[x_] := If[x < 0, Message[g::neg]; Return[HoldForm[g[x]]], Sqrt[x]]
```

```
g[-1]
```

g::neg : Jako argument należy podać wartość nieujemną.

```
g[-1]
```

```
g[x_] := If[x < 0, Message[g::neg1, x]; Return[HoldForm[g[x]]], Sqrt[x]]
```

```
g::neg1 = "Jako argument podałeś wartość `1`. Powinna to być wartość nieujemna.";
```

```
g[4]
```

```
2
```

```
g[-1]
```

g::neg1 : Jako argument podałeś wartość –1. Powinna to być wartość nieujemna.

```
g[-1]
```

```
g[-4]
```

g::neg1 : Jako argument podałeś wartość –4. Powinna to być wartość nieujemna.

```
g[-4]
```

```
Off[g::neg1]
g[-1]
```

```
g[-1]
```

```
On[g::neg1]
g[-1]
```

g::neg1 : Jako argument podałeś wartość –1. Powinna to być wartość nieujemna.

```
g[-1]
```

■ Różne błędy

Zmiana wartości parametrów wywołania

```
Remove[g]
```

```
g[x_] := Module[{}, x = x + 2; x]
```

```
g[1]
```

Set::setraw : Cannot assign to raw object 1. ≫

```
1
```

```
g1[x_] := Module[{y}, y = x + 2; y]
```

```
g1[1]
```

```
3
```

Brak informacji o poleceniu (definicja komunikatów w poleceniu)

```
Remove[g]
```

```
g[x_] := Module[{},
  g::usage =
   "Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.";
  g::neg = "Jako argument należy podać wartość nieujemną.";
  If[x < 0, Message[g::neg]; Return[HoldForm[g[x]]], Sqrt[x]]]
```

```
?g
```

```
Global`g
```

```
g[x_] := Module[{},
  g::usage = Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.;
  g::neg = Jako argument należy podać wartość nieujemną.;
  If[x < 0, Message[g::neg]; Return[HoldForm[g[x]]], √x ]]
```

```
g[1]
```

```
1
```

```
?g
```

Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.

Brak wyniku

ostatnie polecenie zakończone średnikiem

```
Remove[g]
```

```
g[x_] := Module[{},
  If[x < 0, Message[g::neg]; Return[HoldForm[g[x]]], Sqrt[x]];]
g::usage =
  "Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.";
g::neg = "Jako argument należy podać wartość nieujemną.";
```

```
g[1]
```

```
g[-1]
```

g::neg : Jako argument należy podać wartość nieujemną.

```
g[-1]
```

wynik generowany przez polecenie Print

```
g[x_] := Module[{},
  If[x < 0, Message[g::neg]; Return[HoldForm[g[x]]], Print[Sqrt[x]]]]
g::usage =
  "Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.";
g::neg = "Jako argument należy podać wartość nieujemną.";
```

```
Clear[a]
```

```
a = g[1]
```

```
1
```

```
a
```

```
g[-1]
```

g::neg : Jako argument należy podać wartość nieujemną.

```
g[-1]
```

Dziwne wyniki (brak średnika między poleceniami)

```
Remove[g1]
```

```
g1[x_] := Module[{a = x},
  If[Not[VectorQ[x]], Message[g1::tab]; Return[HoldForm[g1[x]]]];
  Do[a[[i]] = Table[x[[i]], {Length[a]}], {i, Length[a]}]
   a]
g1::usage =
  "Polecenie g[liczba] liczy pierwiastek drugiego stopnia z podanej liczby.";
g1::tab = "Jako parametr należy podać listę jednowymiarową.";
```

```
g1[1]
```

g1::tab : Jako parametr należy podać listę jednowymiarową.

```
g1[1]
```

```
g1[{a1, a2}]
```

```
{{a1 Null, a1 Null}, {a2 Null, a2 Null}}
```

Przykład

Przykład 1

Napisać polecenie, które wyznaczy listę różnic kwadratów i sześcianów pierwszych x liczb.

```
? lista
```

Polecenie lista[liczba] wyznacza różnice między kwadratem i sześcianem kolejnych liczb.

```
lista[4]
```

```
{0, -4, -18, -48}
```

```
lista[3.4]
```

lista::blad : Jako parametr należy podać liczbę naturalną.

```
lista[3.4]
```

Przykład 2

Zamiana dwóch wierszy miejscami w macierzy. Opcja obroc określa, czy zamiana jest dokonywana na macierzy transponowanej.

```
a = Table[RandomInteger[20], {5}, {5}]; TableForm[a]
```

```
3   8   12  4   3
16  11  7   6   17
19  19  10  17  20
12  9   18  19  11
17  3   13  13  16
```

```
? obroc
```

Opcja obroc jest opcją polecenia zamien i umożliwia transpozycję macierzy. Domyślną wartością jest False

```
? zamien
```

Polecenie zamien[macierz, nr1,nr2] zamienia w macierzy miejscami wiersze
o podanych numerach. Użycie opcji obroc umożliwia transpozycję macierzy przez zamianą.

```
?? zamien
```

Polecenie zamien[macierz, nr1,nr2] zamienia w macierzy miejscami wiersze
o podanych numerach. Użycie opcji obroc umożliwia transpozycję macierzy przez zamianą.

```
Attributes[zamien] = {Protected, ReadProtected}

Options[zamien] = {obroc → False}
```

```
q = zamien[a, 2, 4, obroc → False]; TableForm[q]
```

```
3   8   12  4   3
12  9   18  19  11
19  19  10  17  20
16  11  7   6   17
17  3   13  13  16
```

```
q = zamien[a, 2, 4, obroc → True]; TableForm[q]
```

```
3   4   12  8   3
16  6   7   11  17
19  17  10  19  20
12  19  18  9   11
17  13  13  3   16
```

Przykład 3

Dodawanie dwóch liczb i wyświetlanie wyniku w innym systemie liczenia. Opcja baza umożliwia określenie systemu pozycyjnego

```
dodaj[3, 5]
```

```
8
```

```
dodaj[3, 5, baza → 4]
```

20_4

```
dodaj[BaseForm[14, 3], BaseForm[43, 6]]
```

dodaj::int : Jako parametry należy podać liczby całkowite. ≫

```
dodaj[112_3, 111_6]
```

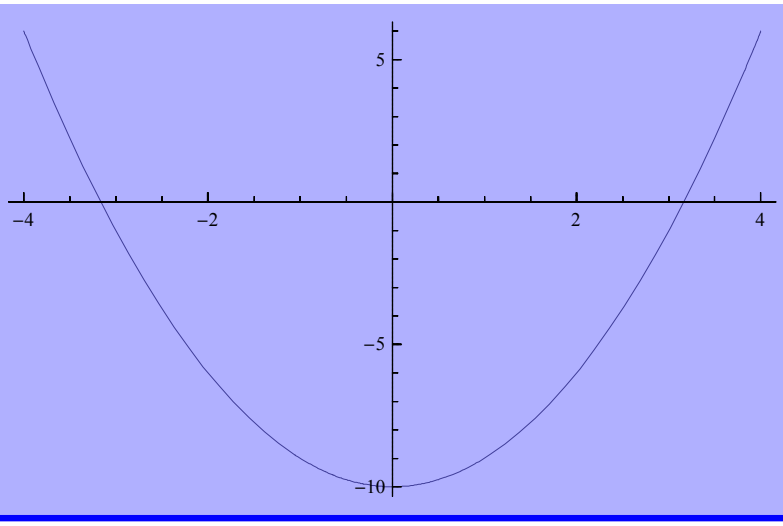
```
dodaj[2^^1001011, 4^^103]
```

```
94
```

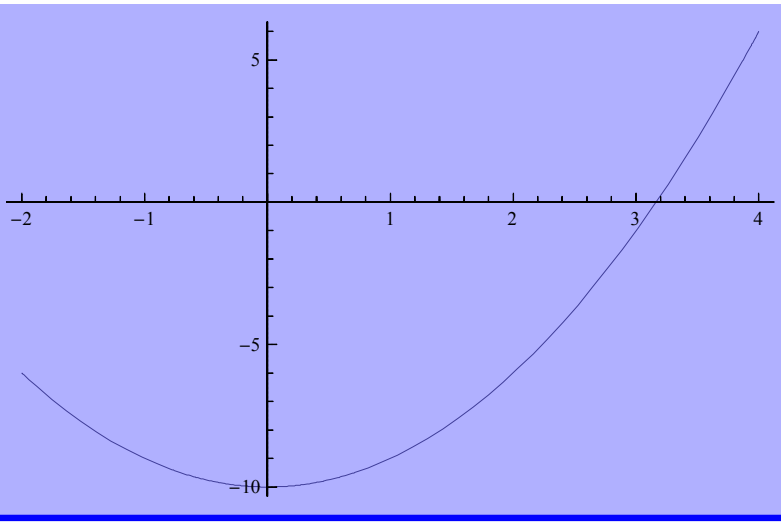
Przykład 4

Wykres funkcji kwadratowej. Opcja zakres umożliwia indywidualne określenie przedziału, na którym będzie rysowana funkcja

```
trojmian[1, 0, -10, zakres → {-4, 4}]
```



```
trojmian[1, 0, -10, zakres → {4, -2}]
```




```
trojmian[1, 0, -10, zakres → {-4, 0}]
```