免費電子書

學習

Prolog Language

Free unaffiliated eBook created from
Stack Overflow contributors.

#prolog

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: prolog-language

It is an unofficial and free Prolog Language ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Prolog Language.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Prolog

1. SWI-Prolog swi-prolog
    - c
2. SICStus sicstus-prolog
3. YAP yap
4. GNU Prolog gnu-prolog
5. XSB xsb
6. B b-prolog
7. IF
8. Ciao
9.
10. ECLiPSe-CLP eclipse-clp
11. Jekejeke Prolog
12. Prolog IV
13. Prolog
    - c javascriptphyton
14. Visual Prolog

Examples

SWI-Prolog

WindowsMac

- SWI-Prolog
- 。

LinuxPPA

- PPA `ppa:swi-prolog/stableppa:swi-prolog/devel`
    - Ctrl + Alt + T `sudo add-apt-repository ppa:swi-prolog/stable`
    - `sudo apt-get update`
- SWI-Prolog `sudo apt-get install swi-prolog`
- `swipl`SWI-Prolog

/ 3

```
append([], Bs, Bs).
append([A|As], Bs, [A|Cs]) :-
    append(As, Bs, Cs).
```

`append/3`Prolog。 true。

Prolog `append/3`

- ```
  ?- A = [1, 2, 3], B=[4, 5, 6], append(A, B, Y)
  Output:
  A = [1, 2, 3],
  B = [4, 5, 6],
  Y = [1, 2, 3, 4, 5, 6].
  ```

- ```
  ?- A = [1, 2, 3], B = [4, 5], C = [1, 2, 3, 4, 5, 6], append(A, B, C)
  Output:
  false
  ```

- ```
  ?- append(A, B, [1, 2, 3, 4]).
  Output:
  A = [],
  B = [1, 2, 3, 4] ;
  A = [1],
  B = [2, 3, 4] ;
  A = [1, 2],
  B = [3, 4] ;
  A = [1, 2, 3],
  B = [4] ;
  A = [1, 2, 3, 4],
  B = [] ;
  false.
  ```

CLPFD

CLPFDProlog。 。

```
?- X #= 1 + 2.
X = 3.

?- 5 #= Y + 2.
Y = 3.
```

Prolog

- - 。 - a b okay
- - 。 1 22 35.8
- - _ 。 X Y Abc AA
- - 。 。 father(john,doe) relative(a) mother(X,Y) 。

。

。

Prolog

```
father_child(fred, susan).
mother_child(hillary, joe).
```
```

Prolog

```
child_of(X,Y):-
    father_child(Y,X)
    ;
    mother_child(Y,X).
```

;or◦

Prolog

◦

XY XYYXYX◦

/◦

◦ ◦

```
?- [nameofdatabase].
```

nameofdatabase .pl◦

```
?- child_of(susan,fred).
true

?- child_of(joe,hillary).
true

?- child_of(fred,susan).
false

?- child_of(susan,hillary).
false

?- child_of(susan,X).
X = fred

?- child_of(X,Y).
X = susan,
Y = fred ;
X = joe,
Y = hillary.
```

-

-

-

-

-

Prolog。 。

*father_child*fredsusan

mother_child

XY father_childYXmother_childYXchild_ofXY。

HelloWorld in the interactive interpreter

“HelloWorld”Prolog`swipl` SWI Prologshell

```
$ swipl
<...banner...>
?- write('Hello, World!'), nl.
```

`?-`a `.`。

`write('Hello World!'), nl`

- `write('Hello World!')` `'Hello World!'` ,
- `nl` 。

`write/1` `/1``nl/0` Prolog。 Prolog。

`yes`。 `true`yes 。

`hello_world.pl`

```
:- initialization hello_world, halt.

hello_world :-
    write('Hello, World!'), nl.
```

`initialization`hello_world, halt 。 `halt`。

Prolog。 Prolog。 SWI Prolog

```
$ swipl -q -l hello_world.pl
```

`Hello, World!` 。 `-q`run `swipl`。 `-l`。

Prolog https://riptutorial.com/zh-TW/prolog/topic/1038/prolog

2: Prolog

Examples

Prolog。Prolog。PrologProlog。

CLPFD

Prolog`is=:=`。PrologCLPFD。CLPFD。

CLPFDProlog。`#=is=:=`。SWI-Prolog

```
?- X is 2+2.
X = 4.

?- use_module(library(clpfd)).
?- X #= 2+2.
X = 4.
```

`is` `#=`

```
?- 4 is 2+X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- 4 #= 2+X.
X = 2.
```

CLPFD。

```
?- between(1,100,X).
X = 1;
X = 2;
X = 3...

?- X in 1..100.
X in 1..100.
```

。`label`

```
?- X in 1..100, label([X]).
X = 1;
X = 2;
X = 3..
```

CLP。

```
?- trace.
?- between(1,10,X), Y is X+5, Y>10.
...
Exit: (8) 6 is 1+5 ? creep
```

```
Call: (8) 6 > 10 ? creep
...
X = 6, Y = 11; ...
```

Prolog1-5。CLPFD

```
?- X in 1..10, Y #= X+5, Y #> 10.
X is 6..10,
X+5 #= Y,
Y is 11..15.
```

CLPFD。`label([Y])`X6..10。1-10;。

CLPFDProlog。CLPFDSICStus Prolog。SWI-PrologProlog。Visual PrologCLPFD。

Forall

“”Prolog`fail`。

```
fdl(X) :- between(1,X,Y), print(Y), fail.
fdl(_).
```

Modern Prolog。

```
nicer(X) :- forall(between(1,X,Y), print(Y)).
```

- 。

Visual Prolog

```
vploop(X) :- foreach Y = std::fromTo(1,X) do
                 console::write(X)
             end foreach.
```

Prolog*foreach*。

Prolog“”

```
mangle(X,Y) :- Y is (X*5)+2.
```

“”。

```
multimangle(X,Y) :- mangle(X,A), mangle(A,B), mangle(B,Y).
```

Prologs`is`。

```
% Define the new infix operator
:- op(900, xfy, <-).
```

```
% Define our function in terms of the infix operator - note the cut to avoid
% the choice falling through
R <- mangle(X) :- R is (X*5)+2, !.

% To make the new operator compatible with is..
R <- X :-
    compound(X),            % If the input is a compound/function
    X =.. [OP, X2, X3],     % Deconstruct it
    R2 <- X2,               % Recurse to evaluate the arguments
    R3 <- X3,
    Expr =.. [OP, R2, R3],  % Rebuild a compound with the evaluated arguments
    R is Expr,              % And send it to is
    !.
R <- X :- R is X, !.        % If it's not a compound, just use is directly
```

```
multimangle(X,Y) :- X <- mangle(mangle(mangle(Y))).
```

Prolog。Visual Prolog

```
mangle(X) = Y :- Y = ((X*5)+2).
multimangle(X,Y) :- Y = mangle(mangle(mangle(X))).
```

<- - 。 。 Visual Prolog - `X = (std::fromTo(1,10))*10`X = 10 X = 20X = 30X = 40。

/

Prolog。 `between(X,Y,Z)`ZXY。 XYZZXYXYZZXYY <X;XZY。 。

。 between

```
%! between(+X,+Y,+Z) is semidet.
%! between(+X,+Y,-Z) is nondet.
```

。 +-。 after

- det 。 `add(+X,+Y,-Z)det`XY。
- semidet。 `between(+X,+Y,+Z)semidet`ZXY。
- multi。 `factor(+X,-Y)multi` - - 。
- nondet 。 `between(+X,+Y,-Z)nondet`XYZY <X。

/。 `between(+From:Int, +To:Int, +Mid:Int) is semidet` 。

Prolog。

Mercury。 。

Visual Prolog。

```
between : (int From, int To, int Mid) determ (i,i,i) nondeterm (i,i,o).
```

- //;
- `io+-`;
- ◦ `detprocedure semidetdeterm nondetnondeterm multimulti` ◦

Prolog https://riptutorial.com/zh-TW/prolog/topic/5499/prolog

3:

Examples

。

Prolog

1.
2.
3. 。

case2。 。 。

- `(=)/2unify_with_occurs_check/2`
- `dif/2`
- **CLPFD** `(#=)/2(#>)/2` 。

Prolog。

1. 。
2. 。

- `var/1 integer/1`。
- `(@<)/2(@>=)/2`
- `!/0 (\+)/1`
- `findall/3setof/3` 。

。

。 。

```
?- var(X), X = a.
X = a.

?- X = a, var(X).
false.
```

`findall/3` 。 。

- `dif/2(\=)/2`
- CLPFDCLPQ
- `!/0` 。
- 。

Prolog。

。 Prolog。 `if_/3if_/3` 。 **dif / 2** 。

```
pred(L, Ls) :-
    condition(L),
    then(Ls).
pred(L, Ls) :-
    \+ condition(L),
    else(Ls).
```

if_/3

```
pred(L, Ls) :-
    if_(condition(L),
        then(Ls),
        else(Ls)).
```

。

https://riptutorial.com/zh-TW/prolog/topic/3989/

4:

Prolog。。 。。

Definite Clause GrammarsDCG。

Examples

sumDif/2。 。

sumDif/2*n*。

```
sumDif([X, +|OpenList], Hole) :-
    integer(X),
    sumDif(OpenList, Hole).
```

X +。 OpenList。 Hole。

sumDif/2

```
sumDif([X|Hole], Hole) :-
    integer(X).
```

X。 Hole。

```
?- sumDif([1,+,2,+,3], []).
true
```

[]。

。 。

- →
- →'+'
- →
- →'*'
- →“”
- →''

3arity。

```
expression(Value, OpenList, FinalHole) :-
    times(Value, OpenList, FinalHole).

expression(SumValue, OpenList, FinalHole) :-
    times(Value1, OpenList, ['+'|Hole1]),
    expression(Value2, Hole1, FinalHole),
```

```
    plus(Value1, Value2, SumValue).

times(Value, OpenList, FinalHole) :-
    element(Value, OpenList, FinalHole).

times(TimesValue, OpenList, FinalHole) :-
    element(Value1, OpenList, ['*'|Hole1]),
    times(Value2, Hole1, FinalHole),
    TimesValue is Value1 * Value2.

element(Value, [Value|FinalHole], FinalHole) :-
    integer(Value).

element(Value, ['('|OpenList], FinalHole) :-
    expression(Value, OpenList, [')'|FinalHole]).
```

expression

```
expression(SumValue, OpenList, FinalHole) :-
    times(Value1, OpenList, ['+'|Hole1]),
    expression(Value2, Hole1, FinalHole),
    plus(Value1, Value2, SumValue).
```

OpenList◦ + ◦ Hole1 ◦ Hole1expression Hole1expression◦

◦ Value1 Value2 ◦

◦

```
?- expression(V, [1,+,3,*,'(',5,+,5,')'], []).
V = 31
```

https://riptutorial.com/zh-TW/prolog/topic/9414/

5:

Examples

Prolog 。

Prolog。

- **WAM** *Warren*
- **TOAM** B-Prolog。
- **ZIP** SWI-PrologVM
- **VAM** 。

Prolog。

arity。 。

JIT。

Prolog TCO。 。

TRO。

https://riptutorial.com/zh-TW/prolog/topic/4205/

6:

Examples

DisjunctionORexplict

Prolog

```
likes(alice, music).
likes(bob, hiking).

// Either alice likes music, or bob likes hiking will succeed.
```

OR；

```
likes(P,Q) :-
    ( P = alice , Q = music ) ; ( P = bob , Q = hiking ).
```

。 ,; 。

AND

AND,。

```
?- X = 1, Y = 2.
```

```
triangleSides(X,Y,Z) :-
    X + Y > Z, X + Z > Y, Y + Z > X.
```

Prolog。 。 。

```
% (percent signs mean comments)
% a is the parent of b, c, and d.
parent(a,b).
parent(a,c).
parent(a,d).
```

`parent/2`

```
?- parent(a,X).
```

。 prologcut。

```
?- parent(a,X), !.
```

Xb。

https://riptutorial.com/zh-TW/prolog/topic/4479/

7:

Examples

◦

- atom `[]` ◦
- `Ls'.'(L, Ls)` ◦

Prolog

1. `'.'(a, '.'(b, '.'(c, [])))[a,b,c]` ◦
2. `'.'(L, Ls)[L|Ls]` ◦

◦ `[a,b|Ls]`*iff* `Ls`◦

List

```
?- List = [1,2,3,4].
List = [1, 2, 3, 4].
```

consing

```
?- Tail = [2, 3, 4], List = [1|Tail].
Tail = [2, 3, 4],
List = [1, 2, 3, 4].
```

`length/2`

```
?- length(List,5).
List = [_G496, _G499, _G502, _G505, _G508].
```

Prolog

```
?- List = [1, 2>1, this, term(X), 7.3, a-A].
List = [1, 2>1, this, term(X), 7.3, a-A].
```

```
List = [[1,2],[3,[4]]].
```

`(-)/2`Prolog◦ `-(A, B)AB`◦ Prolog `(-)/2` ◦ `AB` ◦

◦ `keysort/2pairs_keys_values/3` ◦

Prolog ◦ **AVL** ◦ `library(assoc)`Prolog*OlogN*◦

Prolog**term** ◦ PrologProlog◦

-

    - ◦ x test'quotes and space' ◦
  - ◦ _◦
  - ◦ 4242.42 ◦
  - T1 T2 ... T_n *F* T1 T2 ... T_n *F*◦

[record][1]◦ :- record/1 <spec><spec>◦

xypoint

```
:- use_module(library(record)).

:- record point(x:integer=0,
                y:integer=0).

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

?- default_point(Point), point_x(Point, X), set_x_of_point(10, Point, Point1).
Point = point(0, 0),
X = 0,
Point1 = point(10, 0).

?- make_point([y(20)], Point).
Point = point(0, 20).

?-  is_point(X).
false.

?- is_point(point(_, _)).
false.

?- is_point(point(1, a)).
false.

?- is_point(point(1, 1)).
true.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

https://riptutorial.com/zh-TW/prolog/topic/2417/

8:

Examples

Prolog。Prolog。。。true / falseProlog。

```
% Facts
father_child(paul,chris).        % Paul is the father of Chris and Ellen
father_child(paul,ellen).
mother_child(ellen,angie).       % Ellen is the mother of Angie and Peter
mother_child(ellen,peter).


% Rules
grandfather_grandchild(X,Y) :-
    father_child(X,Z),
    father_child(Z,Y).

grandfather_grandchild(X,Y) :-
    father_child(X,Z),
    mother_child(Z,Y).
```

```
?- grandfather_grandchild(paul,peter).
```

```
                                  ?- grandfather_grandchild(paul,peter).
                                      /                                \
                                     /                                  \
  ?- father_child(paul,Z1),father_child(Z1,peter).            ?-
father_child(paul,Z2),mother_child(Z2,peter).
             /                    \                                    /
\
      {Z1=chris}             {Z1=ellen}                        {Z2=chris}
{Z2=ellen}
           /                        \                               /
\
?- father_child(chris,peter).  ?- father_child(ellen,peter).  ?- mother_child(chris,peter). ?-
mother_child(ellen,peter).
         |                          |                             |
/              \
       fail                      fail                          fail
fail(*)          success
```

*'angie''peter'`mother_child(ellen,angie)`

https://riptutorial.com/zh-TW/prolog/topic/3097/

9: DCG

Examples

`... // 0`

DCG`... //0` “”

```
... --> [] | [_], ... .
```

`E` via`Ls`

```
phrase(( ..., [E], ... ), Ls)
```

DCG

DCG。 DCG。

```
sentence --> article, subject, verb, object.

article --> [the].

subject --> [woman] | [man].

verb --> [likes] | [enjoys].

object --> [apples] | [oranges].
```

```
?- phrase(sentence, Ls).
Ls = [the, woman, likes, apples] ;
Ls = [the, woman, likes, oranges] ;
Ls = [the, woman, enjoys, apples] .

?- phrase(sentence, [the,man,likes,apples]).
true .
```

DCG。

DCG。

```
% DCG clause requiring an integer
int --> [X], {integer(X)}.
```

```
?- phrase(int, [3]).
true.

?- phrase(int, [a]).
false.
```

DCG。。

```
% Extra arguments are passed between parenthesis after the name of the DCG clauses.
exp(C) --> int(A), [+], exp(B), {plus(A, B, C)}.
exp(X) --> int(X).
int(X) --> [X], {integer(X)}.
```

```
?- phrase(exp(X), [1,+,2,+,3]).
X = 6 ;
```

DCG https://riptutorial.com/zh-TW/prolog/topic/2426/-dcg-

10:

Examples

CLPFD

CLPFD 。Prolog。

CLPFD

-
- 。

```
?- X #= 1+2.
X = 3.

?- 3 #= Y+2.
Y = 1.
```

`is/2`

```
?- 3 is Y+2.
ERROR: is/2: Arguments are not sufficiently instantiated
```

CLPQ

CLPQ。

```
?- { 5/6 = X/2 + 1/3 }.
X = 1.
```

CLPH

Prolog**CLPH** *Herbrand*。Prolog。

```
?- X = f(Y), Y = a.
X = f(a),
Y = a.
```

https://riptutorial.com/zh-TW/prolog/topic/2057/

11:

Examples

Prolog

-
- 。

。 。 underscores_keep_even_longer_names_readable mixingTheCasesDoesNotDoThisToTheSameExtent 。

- parent_child/2
- person_likes/2
- route_to/2

。 。 Prolog。

。 BestSolutions MinElement GreatestDivisor。 S0 S1 S2 ... S S。

Prolog。

(;)/2。 ;,,。

```
(  Goal1
;  Goal2
)
```

Prolog。 。 。

- 。 。
- p(..., State0, State, ...)
- 。 。

https://riptutorial.com/zh-TW/prolog/topic/4612/

12:

Examples

◦

- `writeq/1`
- `read/1`
- `format/2`

◦ ◦

◦

- `var/1`
- `ground/1`
- `integer/1`

◦

- `arg/3`
- `functor/3`
- `(=..)/2`

◦

◦

- `setof/3`
- `findall/3`
- `bagof/3`

/ 0

Prolog◦

- `!/0`
- `(->)/2`if-then-else
- `(\+)/1`

◦

https://riptutorial.com/zh-TW/prolog/topic/2282/

13:

Examples

ISO / IEC 13211-113211-2

		s	
1200	XFX	`:- -->`	
1200	FX	`:- ?-`	
1100	XFY	`;`	
1050	XFY	`->`	
1000	XFY	`','`	
900	FY	`\+`	
700	XFX	`= \\=`	
700	XFX	`== \\== @< @=< @> @>=`	
700	XFX	`=..`	
700	XFX	`is =:= =\= < > =< >=`	
600	XFY	`:`	
500	YFX	`+ - /\ \/`	
400	YFX	`* / div mod // rem << >>`	
200	XFX	`**`	
200	XFY	`^`	
200	FY	`+ - \`	;

		s	
1150	FX	`dynamic multifile discontiguous initialization`	
1150	FX	`mode public volatile block meta_predicate`	
900	FY	`spy nospy`	

Prolog`op/3`

```
op(+Precedence, +Type, :Operator)
```

- 。 。
- 012000。
- `xf yf xfx xfy yfx fyfxfyfx f xy。 yx。`
    - `fx fy`
    - `xfx xfy yfx`
    - `xf yf`

```
:- op(900, xf, is_true).

X_0 is_true :-
  X_0.
```

```
?- dif(X, a) is_true.
dif(X, a).
```

@ <numbers @ <atoms @ <strings @ <structures @ <lists

- arity。
- 。

X @ <Y	XY.
X @> Y.	XY.
X @ = <Y	XY.
X @> = Y.	XY.

```
?- alpha @< beta.
true.

?- alpha(1) @< beta.
false.

?- alpha(X) @< alpha(1).
true.

?- alpha(X) @=< alpha(Y).
true.

?- alpha(X) @> alpha(Y).
false.
```

```
?- compound(z) @< compound(inner(a)).
true.
```

X = Y.	XY
X \ = Y.	XY
X == Y.	XY
X \ == Y.	XY
X == Y.	XY
X = \ = Y.	XY

https://riptutorial.com/zh-TW/prolog/topic/2479/

14:

Examples

DIF / 2

`dif/2` ◦

CLPFD

CLPFD◦

```
?- X #= 1+2.
X = 3.

?- 3 #= Y+2.
Y = 1.
```

◦ ◦

Prolog

- `(=)/2unify_with_occurs_check/2`
- ◦

https://riptutorial.com/zh-TW/prolog/topic/2058/

15:

Examples

◦

◦

◦

Prolog Prolog ISO◦

`throw/1catch/3`◦

ISO◦ `error(E,_) E`◦ `instantiation_error` `domain_errortype_error` ◦

`setup_call_cleanup/3` ◦

`setup_call_cleanup/3` Prolog ISO◦

```
setup_call_cleanup(open(File, Mode, Stream), process_file(File), close(Stream))
```

`open/3`◦ `Setup`◦ `call_cleanup/2` Prolog◦

◦

- `integer`
- `atom`
- `list` ◦

◦

01520◦

◦

https://riptutorial.com/zh-TW/prolog/topic/7114/

16:

+。 。

“”Prolog。 “”“”。

Examples

Prolog。 。 Prolog

- **Base** - '。 。
- **continue** 。

append/3 。 L3L1L2 append(L1,L2,L3)。 。 。

```
% Base case
append([],L,L).

% Recursive clause
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

“LL”L - Prolog

```
?- append(X,some_term(a,b),Z).
X = [],
Z = some_term(a, b).
```

Prolog -

```
    append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

“append(L1,L2,L3)”

```
    append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

“append([X|L1],L2,[X|L3]) ”

L3L1L2[XL3][XL1]L2。

“[1,2,3][1][2,3][a1,2,3][a1][2,3]。 “

。 Prolog

```
?- append(L1,L2,L3).
L1 = [],
L2 = L3 ;                                   % Answer #1
L1 = [_G1162],
```

```
L3 = [_G1162|L2] ;                          % Answer #2
L1 = [_G1162, _G1168],
L3 = [_G1162, _G1168|L2] ;                  % Answer #3
L1 = [_G1162, _G1168, _G1174],
L3 = [_G1162, _G1168, _G1174|L2] ;          % Answer #4
...
```

`_G1162` like

```
?- append(L1,L2,L3).
L1 = [],
L2 = L3 ;                                   % Answer #1
L1 = [_A],
L3 = [_A|L2] ;                              % Answer #2
L1 = [_A, _B],
L3 = [_A, _B|L2] ;                          % Answer #3
L1 = [_A, _B, _C],
L3 = [_A, _B, _C|L2] ;                      % Answer #4
...
```

Prolog`L1L2L3L3`L2。

2Prolog`L1L2L3` 。 `_A`L1L3`[_A|L2]` 。

`L1 = [_A]`。 Prolog`L1 L2L3`。 `_AL1`Prolog`_AL1 = [_A,_B]L3 = [_A,_B|L2]`

100

```
L1 = [X1,X2,..,X99],
L3 = [X1,X2,..,X99|L2]
```

Prolog `append/3`。 。

`member/2member(?Elem, ?List) ElemListtrue` 。 。

```
?- member(X, [1,2,3]).
X = 1 ;
X = 2 ;
X = 3.

?- member(X,[Y]).
X = Y.

?- member(X,Y).
Y = [X|_G969] ;
Y = [_G968, X|_G972] ;
Y = [_G968, _G971, X|_G975] ;
Y = [_G968, _G971, _G974, X|_G978]
...
```

```
third([_,_,X|_], X).
fourth([_,_,_,X|_], X).
```

https://riptutorial.com/zh-TW/prolog/topic/2005/

17:

Examples

call / N

`call/N`Prolog

```
?- G=true, call(G).
true.

?- G=(true,false), call(G).
false.
```

MAPLIST / [2,3]

`maplist/2maplist/3 ◦ call/2call/3`Prolog◦

```
?- maplist(dif(a), [X,Y,Z]).
dif(X, a),
dif(Y, a),
dif(Z, a).
```

Prolog◦ PrologProlog

```
?- Goal = dif(X, Y), Goal.
dif(X, Y).
```

Prolog◦

foldl / 4

- 3
-
-
- ◦

`foldl/4`

```
?- foldl(plus, [2,3,4], 0, S).
S = 9.
```

call / 1maplist / 2

```
?- Gs = [X = a, Y = b], maplist(call, Gs).
Gs = [a=a, b=b],
X = a,
```

```
Y = b.
```

https://riptutorial.com/zh-TW/prolog/topic/2420/

S. No		Contributors
1	Prolog	coder, Community, Eyal, Limmen, manlio, mat, Nick the coder, Norman Creaney, Rajat Jain, Ruslan López Carro, SeekAndDestroy, Willem Van Onsem, Xevaquor
2	Prolog	Mark Green
3		manlio, mat
4		ZenLulz
5		mat
6		hardmath, Nick the coder
7		Eyal, mat, SeekAndDestroy
8		SeekAndDestroy
9	DCG	false, mat, Will Ness, ZenLulz
10		mat, SeekAndDestroy
11		mat
12		false, mat
13		false, mat, SeekAndDestroy
14		mat
15		mat
16		Daniel Lyons, manlio, mat, SeekAndDestroy
17		4444, Eyal, mat