Defining operators in Prolog

- ✔ Some built-in functors in Prolog can be written either with standard structure notation or with operator notation:

  `+(3, *(/(Z, Pi), 77))` is equivalent to `3 + Z/Pi * 77`

- ✔ In many cases, the operator notation is easier to understand, and provides a better user interface

- ✔ New operators can be defined by the user in Prolog programs

- ✔ User-defined operators do not perform data operations; they just change the syntax for expressing Prolog structures

Operators in Prolog

- An operator is defined by giving its name, its type, and its precedence

- This is done by asserting a fact of the form

  `op(precedence, type, opname).`

- `precedence` is an integer in the range 1-1200

- `type` is one of `xfx, xfy, yfx, fx, fy, xf, yf`

- `opname` is an atom, or a list of atoms

Operators in Prolog

- Each argument to an operator has a precedence defined as follows:
  - If the argument is an atom or is enclosed in parentheses, it has precedence 0
  - If the argument is a structure, it has the precedence of its principal functor

- The operator type defines how operators combine

  - **`xfx, yfx, xfy`** specify 2-ary infix operators
    - **`xfx`** means both arguments must have precedence less than the operator itself
    - **`yfx`** means the first argument can have precedence equal to the operator itself
    - **`xfy`** means the second argument can have precedence equal to the operator itself

  - **`fx, fy`** specify 1-ary prefix operators
    - **`fx`** means the argument must have precedence less than the operator itself
    - **`fy`** means the argument can have precedence equal to the operator itself

  - **`xf, yf`** specify 1-ary postfix operators
    - **`xf`** means the argument must have precedence less than the operator itself
    - **`yf`** means the argument can have precedence equal to the operator itself

Operators in Prolog

✔ Definitions of some built-in Prolog operators:

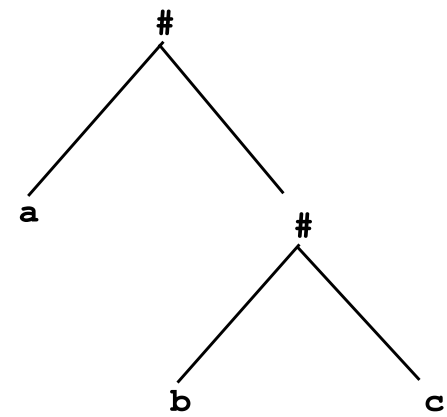
```
:- op(1200, xfx, ':-').

:- op(1000, xfy, ',').

:- op(700, xfx, [=,is,<,>,=<,>=,==,=\=,=:=]).

:- op(500, yfx, [+, -]).

:- op(500, fx, [+, -, not]).

:- op(400, yfx, [*, /, //]).

:- op(300, xfx, mod).
```

✔ In a parenthesis-free expression where every functor is an operator, the principal functor will have the highest precedence

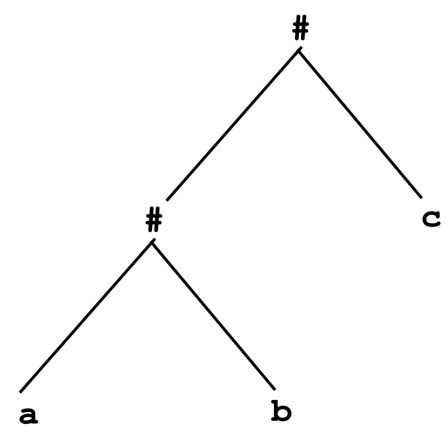
Operators in Prolog

✔ The difference between **xfy, yfx, xfx** operators

```
:- op(500, xfy, #).
a # b # c
```




```
:- op(500, yfx, #).
a # b # c
```




```
:- op(500, xfx, #).
a # b # c

Syntax error
```

Operators in Prolog

- ✔ With these operator definitions

```
:- op(300,xfx,plays).
:- op(200,xfy,and).
```

what is the structure corresponding to

```
susan plays tennis and basketball and volleyball
```

```
plays(susan,and(tennis,and(basketball,volleyball)))
```

Operators in Prolog

- What operator definitions would work for these clauses defining the append relation?

```
appending [] to List gives List.

appending [First|Rest] to List gives [First|List2] :-
    appending Rest to List gives List2.
```

```
:- op(300,fx,appending).
:- op(200,xfx,gives).
:- op(100,xfx,to).
```

```
:- op(300,xfy,to).
:- op(200,fx,appending).
:- op(100,yfx,gives).
```

```
:- op(300,yfx,gives).
:- op(200,xfx,to).
:- op(100,fx,appending).
```