NOVA SCHOOL OF SCIENCE & TECHNOLOGY

MDE

Labwork2 – Class 2

Dynamic Change & I/O in Prolog

2024 - 2025

Lab2 (Class 2)

- Dynamic Change in Memory
  - Dynamic Directive
  - Assert and retract predicates
- Input / Output
  - Read, write, nl, open
  - Examples using menu

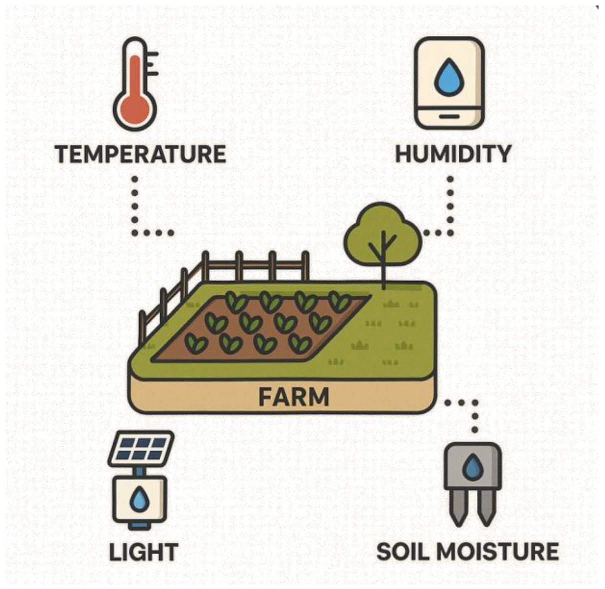
These mechanisms allow us to modify the knowledge base at runtime and interact with users or external files.

Dynamic Change of Memory

Dynamic Directive

Facts (some examples):

```
farmer(ana, alentejo).
farm(quinta_sol, ana).
sensor_reading(quinta_sol, humidity, 28).
sensor_reading(quinta_sol, temperature, 34).
```



In order to add new facts in runtime, please add in the beginning of your file:

```
% A farmer has a name and a zone
:- dynamic farmer/2.
% A farm has a name and belongs to a farmer
:- dynamic farm/2.
% A sensor reading is associated with a farm
:- dynamic sensor_reading/3.
```

In Prolog, the notation **/n** indicates the **arity** of a predicate — that is, **how many arguments** it has.

ATTENTION

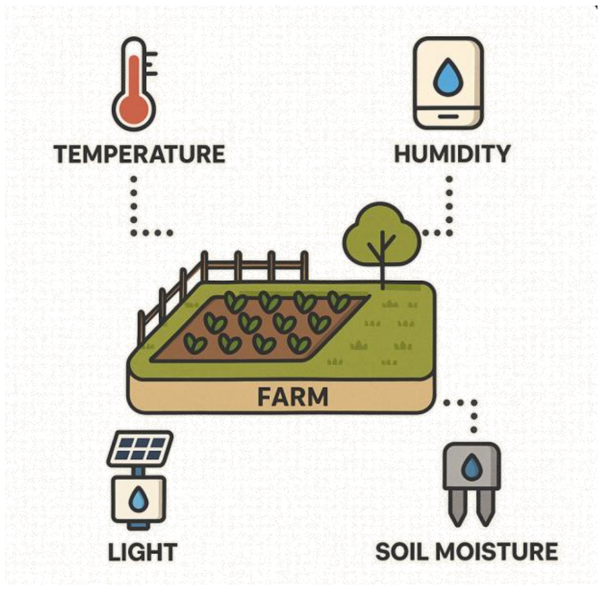
`farm/2` and `farm/1` are **different predicates!!!**

Dynamic Change of Memory

Dynamic Directive

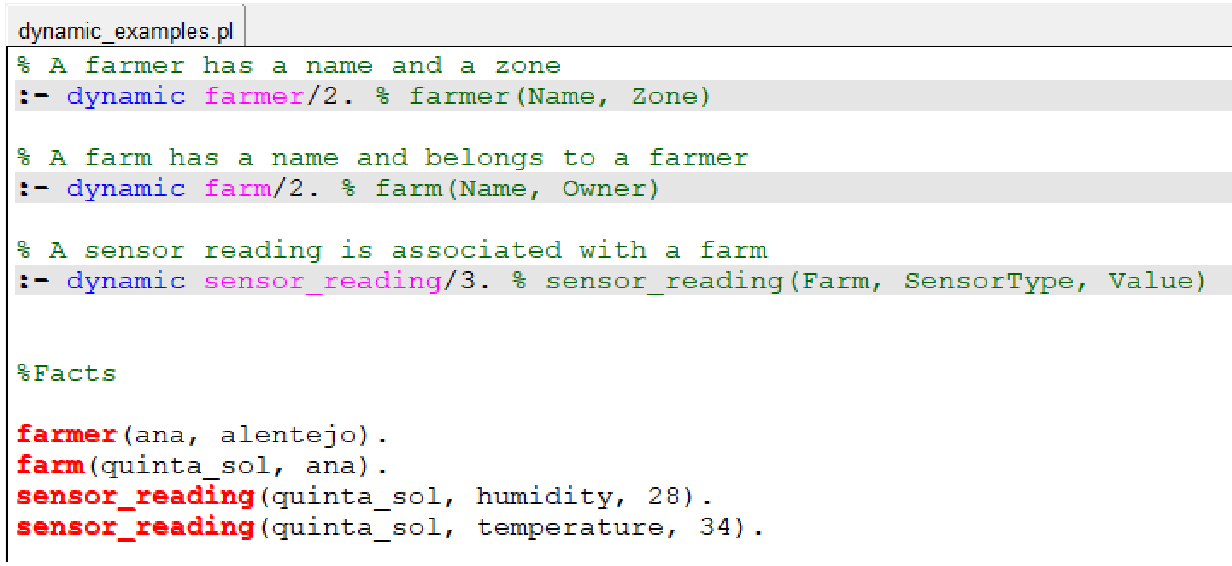
Facts (some examples):

```
farmer(ana, alentejo).
farm(quinta_sol, ana).
sensor_reading(quinta_sol, humidity, 28).
sensor_reading(quinta_sol, temperature, 34).
```



In order to add new facts in runtime, please add in the beginning of your file:

```
% A farmer has a name and a zone
:- dynamic farmer/2.
% A farm has a name and belongs to a farmer
:- dynamic farm/2.
% A sensor reading is associated with a farm
:- dynamic sensor_reading/3.
```



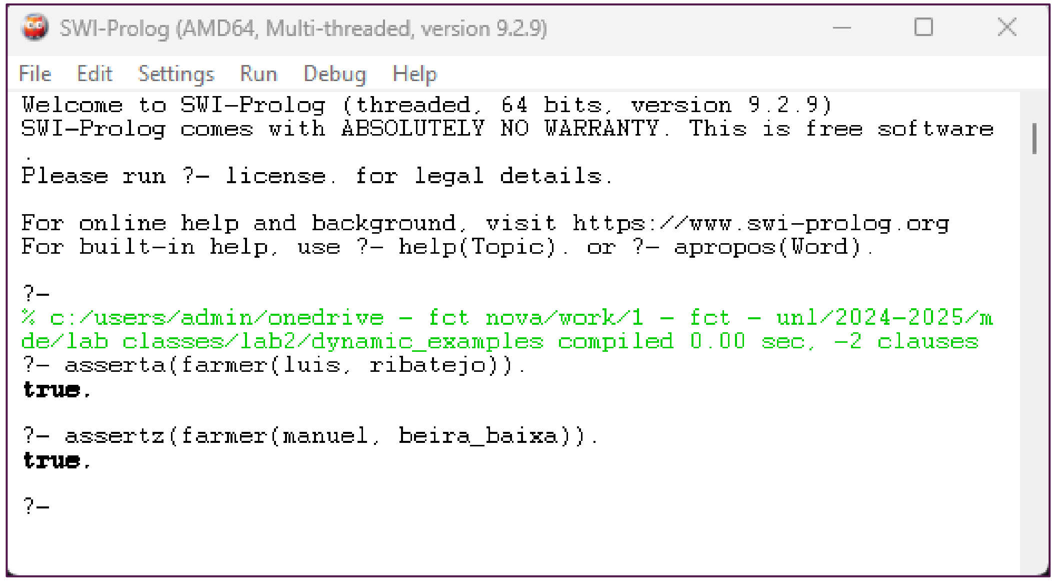
```
% A farmer has a name and a zone
:- dynamic farmer/2. % farmer(Name, Zone)

% A farm has a name and belongs to a farmer
:- dynamic farm/2. % farm(Name, Owner)

% A sensor reading is associated with a farm
:- dynamic sensor_reading/3. % sensor_reading(Farm, SensorType, Value)


%Facts

farmer(ana, alentejo).
farm(quinta_sol, ana).
sensor_reading(quinta_sol, humidity, 28).
sensor_reading(quinta_sol, temperature, 34).
```

Dynamic Change of Memory

Assert Predicate

Try the following instructions (in SWI-Prolog console):

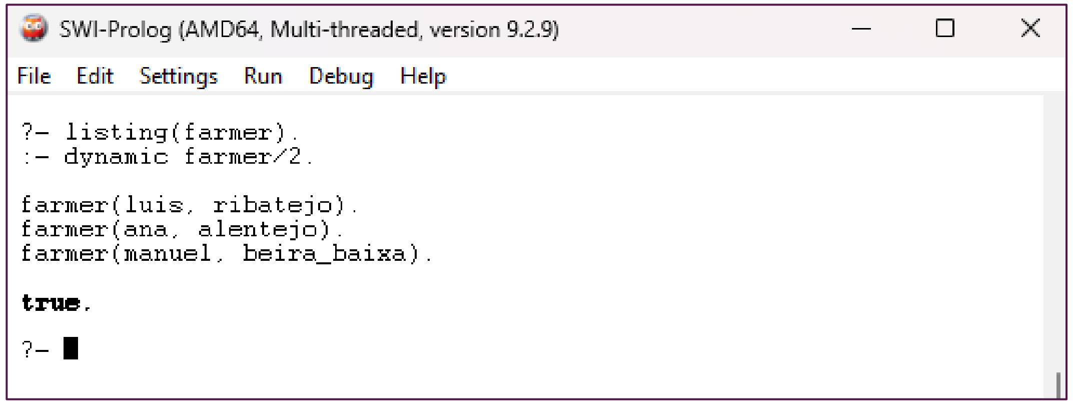
- asserta(farmer(luis, ribatejo)).
- assertz(farmer(manuel, beira_baixa)).


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software
.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/users/admin/onedrive - fct nova/work/1 - fct - unl/2024-2025/m
de/lab classes/lab2/dynamic_examples compiled 0.00 sec, -2 clauses
?- asserta(farmer(luis, ribatejo)).
true.

?- assertz(farmer(manuel, beira_baixa)).
true.

?-
```

Now try the following instruction (in SWI-Prolog console):

- listing(farmer).


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
?- listing(farmer).
:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(ana, alentejo).
farmer(manuel, beira_baixa).

true.

?- 
```

- asserta inserts the fact in the beginning
- assertz inserts the fact in the end

You can also try assert:

- assert inserts in a random position

Dynamic Change of Memory

Assert Predicate

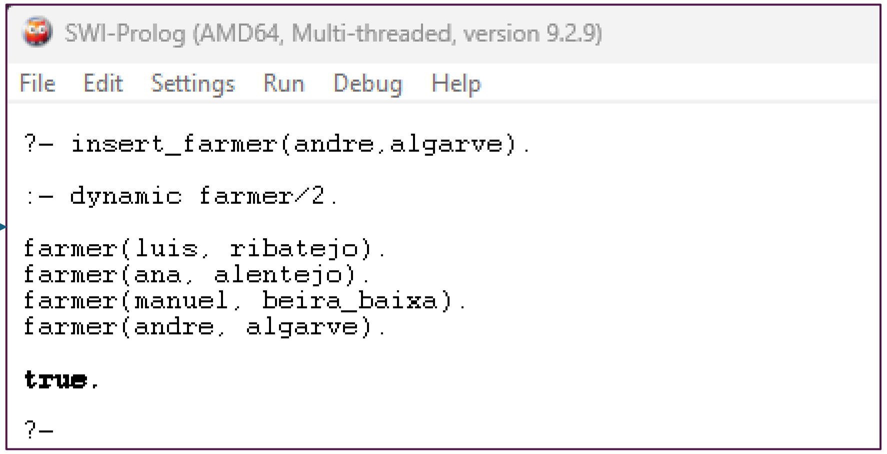
- Let’s create a rule to insert a new fact and print the current list of facts.

- Add the following rule:

```
insert_farmer(N,Z) :-
      assertz(farmer(N,Z)),
      nl,
      listing(farmer).
```

- Try:

```
insert_farmer(andre,algarve).
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
?- insert_farmer(andre,algarve).

:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(ana, alentejo).
farmer(manuel, beira_baixa).
farmer(andre, algarve).

true.

?-
```

Dynamic Change of Memory

Assert Predicate

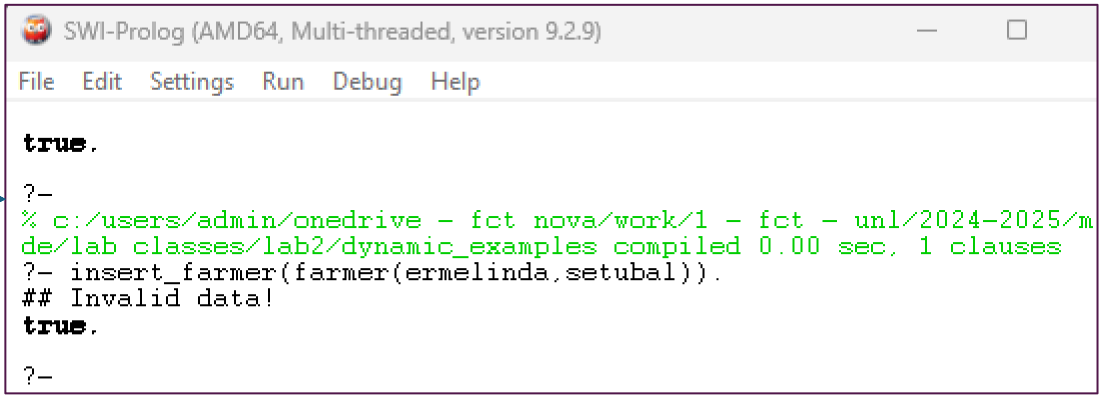
- Let's warning the user if not used correctly.

- Add:

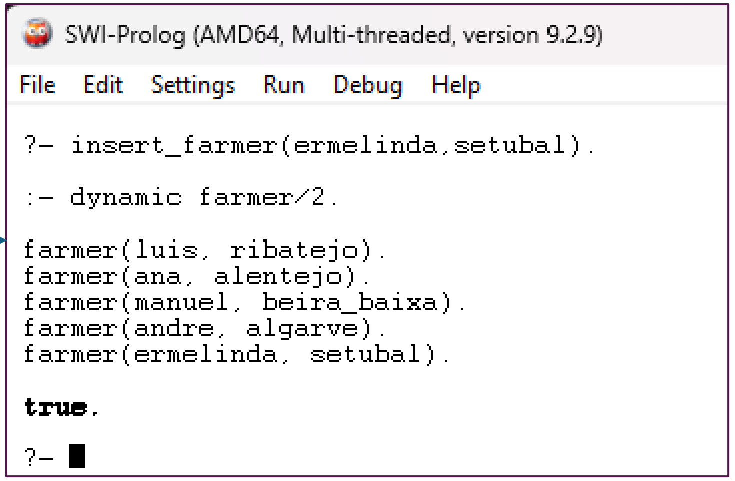
```
insert_farmer(N,Z) :-
        assertz(farmer(N,Z)),
        nl,
        listing(farmer).

insert_farmer(_) :- write('## Invalid data!').
```

- Try:

```
insert_farmer(farmer(ermelinda,setubal)).
insert_farmer(ermelinda,setubal).
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
true.

?-
% c:/users/admin/onedrive - fct nova/work/1 - fct - unl/2024-2025/m
de/lab classes/lab2/dynamic_examples compiled 0.00 sec, 1 clauses
?- insert_farmer(farmer(ermelinda,setubal)).
## Invalid data!
true.

?-
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

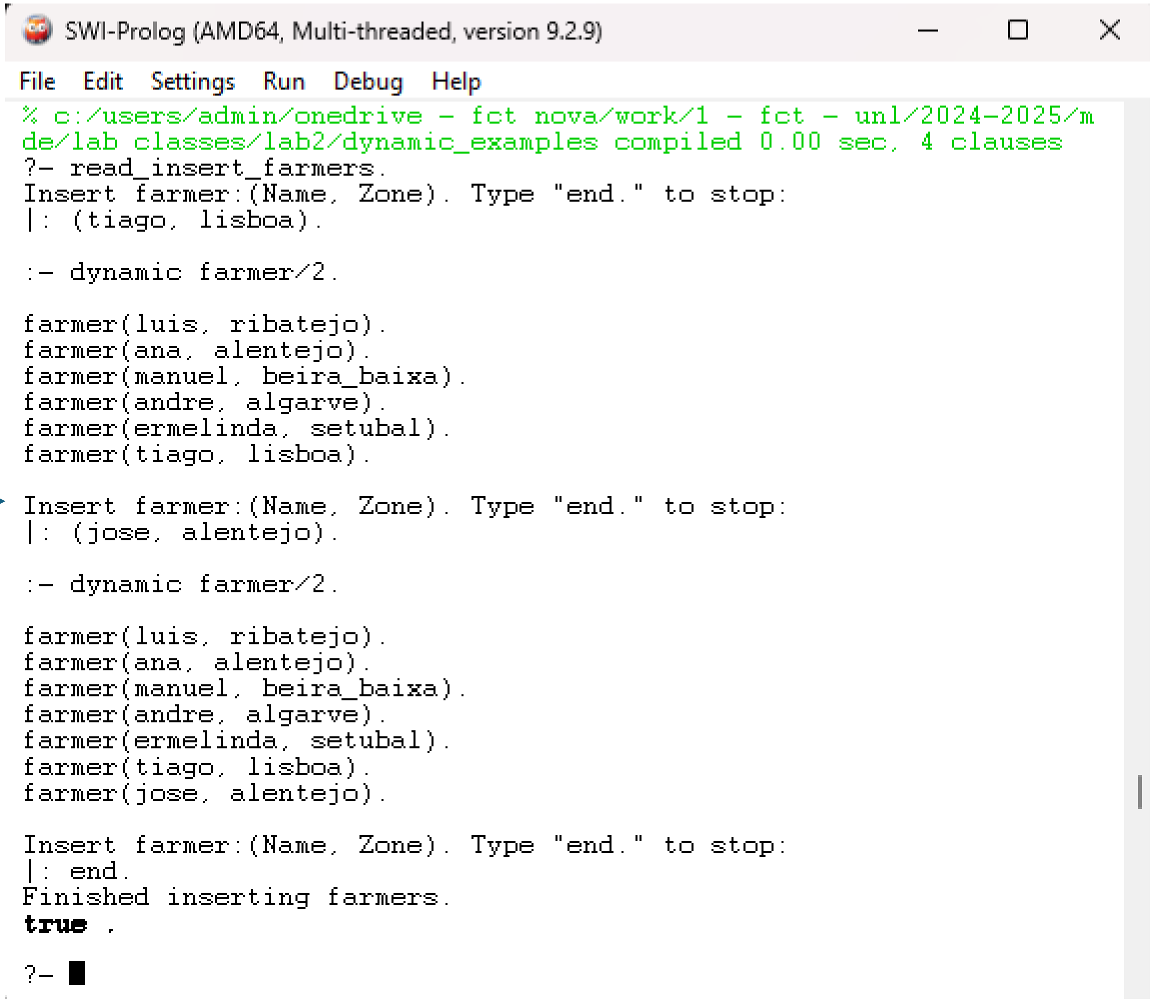
```
?- insert_farmer(ermelinda,setubal).

:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(ana, alentejo).
farmer(manuel, beira_baixa).
farmer(andre, algarve).
farmer(ermelinda, setubal).

true.

?-
```

Dynamic Change of Memory

- How about a program that lets us insert facts until we're done?
- Let's try the following rules:

```
read_insert_farmers :-
    write('Insert farmer:(Name, Zone).
           Type "end." to stop:’),
    nl,
    read(Input),
    handle_input(Input).

handle_input(end) :-
    write('Finished inserting farmers.'), nl.

handle_input((Name, Zone)) :-
    insert_farmer(Name, Zone),
    read_insert_farmers.

handle_input(_) :-
    write('## Invalid input! Try again.'), nl,
    read_insert_farmers.
```

- **Try it!!!**


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
% c:/users/admin/onedrive - fct nova/work/1 - fct - unl/2024-2025/m
de/lab classes/lab2/dynamic_examples compiled 0.00 sec, 4 clauses
?- read_insert_farmers.
Insert farmer:(Name, Zone). Type "end." to stop:
|: (tiago, lisboa).

:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(ana, alentejo).
farmer(manuel, beira_baixa).
farmer(andre, algarve).
farmer(ermelinda, setubal).
farmer(tiago, lisboa).

Insert farmer:(Name, Zone). Type "end." to stop:
|: (jose, alentejo).

:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(ana, alentejo).
farmer(manuel, beira_baixa).
farmer(andre, algarve).
farmer(ermelinda, setubal).
farmer(tiago, lisboa).
farmer(jose, alentejo).

Insert farmer:(Name, Zone). Type "end." to stop:
|: end.
Finished inserting farmers.
true .

?-
```

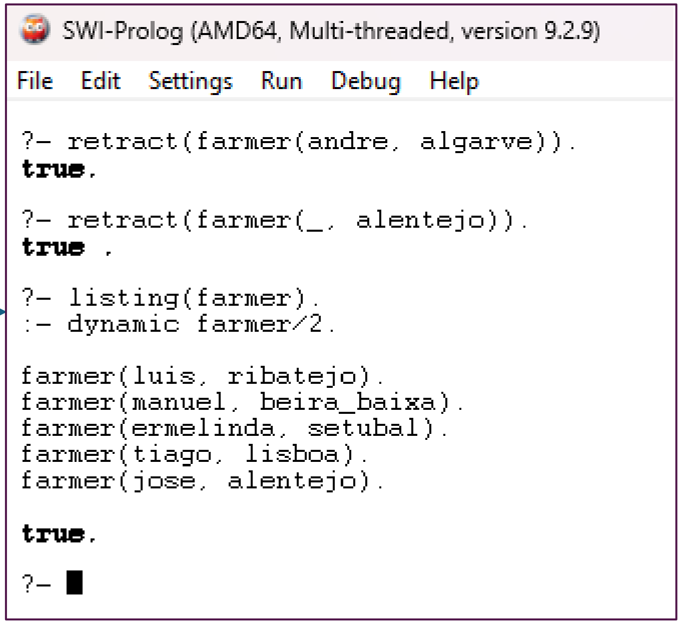
Dynamic Change of Memory

Retract Predicate

- To remove facts, we can use retract:
- Try:

```
retract(farmer(andre, algarve)).

retract(farmer(_, alentejo)).
```

- The first command removed a specific fact.
- The second command removed the first fact that can answer *farmer(_,alentejo)*.


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

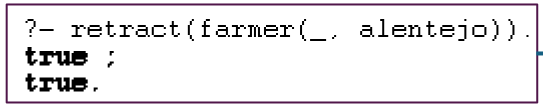
```
?- retract(farmer(andre, algarve)).
true.

?- retract(farmer(_, alentejo)).
true .

?- listing(farmer).
:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(manuel, beira_baixa).
farmer(ermelinda, setubal).
farmer(tiago, lisboa).
farmer(jose, alentejo).

true.

?-
```

Dynamic Change of Memory

Retract Predicate

- If you want to remove all the facts where *alentejo* is the farmer's zone, try:

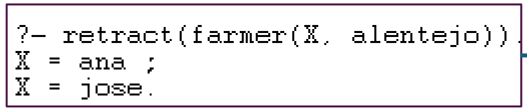
```
?- retract(farmer(_, alentejo)).
```



```
?- retract(farmer(_, alentejo)).
true ;
true.
```

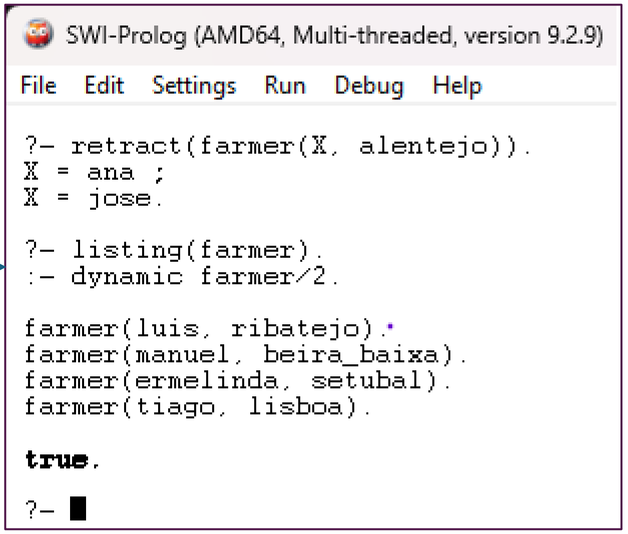
Or

```
?- retract(farmer(X, alentejo)).
```



```
?- retract(farmer(X, alentejo)).
X = ana ;
X = jose.
```

```
listing(farmer).
```



SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

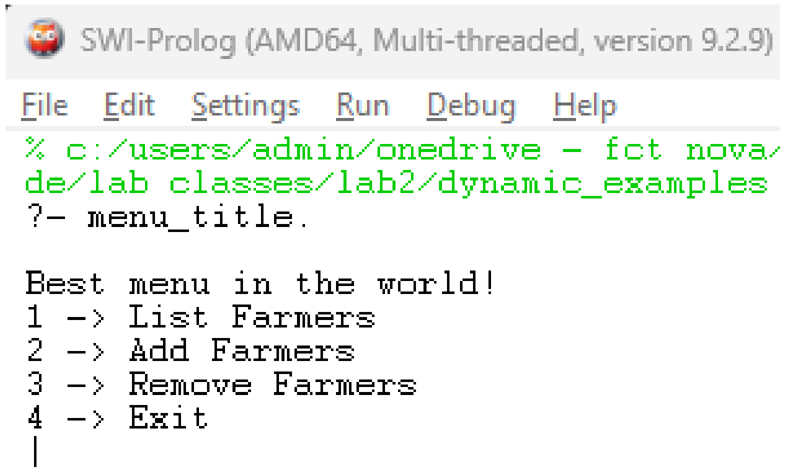
```
?- retract(farmer(X, alentejo)).
X = ana ;
X = jose.

?- listing(farmer).
:- dynamic farmer/2.

farmer(luis, ribatejo).
farmer(manuel, beira_baixa).
farmer(ermelinda, setubal).
farmer(tiago, lisboa).

true.

?- 
```

- You can also se `retractall/1` if you want to delete all matching entries safely. **Try it!!**

Menu Example

- Simple menu:

  1 -> Listing Farmers
  2 -> Add Farmers
  3 -> Remove Farmers
  4 -> Exit


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
% c:/users/admin/onedrive - fct nova/
de/lab classes/lab2/dynamic_examples
?- menu_title.

Best menu in the world!
1 -> List Farmers
2 -> Add Farmers
3 -> Remove Farmers
4 -> Exit
|
```

- **Try it!**

```
% menu

menu_title :- nl,
    write('Best menu in the world!'), nl,
    menu(Op),
    execute(Op).

menu(Op) :-
    write('1 -> List Farmers'), nl,
    write('2 -> Add Farmers'), nl,
    write('3 -> Remove Farmers'), nl,
    write('4 -> Exit'), nl,
    read(Op).

execute(4) :- !.  % finish execution
execute(Op) :-
    exec(Op), nl,
    menu(NOp),
    execute(NOp).

addFarmers:- read_insert_farmers.

removeFarmers:- !.

exec(1) :- listing(farmer).
exec(2) :- addFarmers.
exec(3) :- removeFarmers.
exec(_) :- write('Invalid option! Try again.'), nl.
```

Read Facts From Previously Saved File

- Add `read_facts` predicate to your main file…

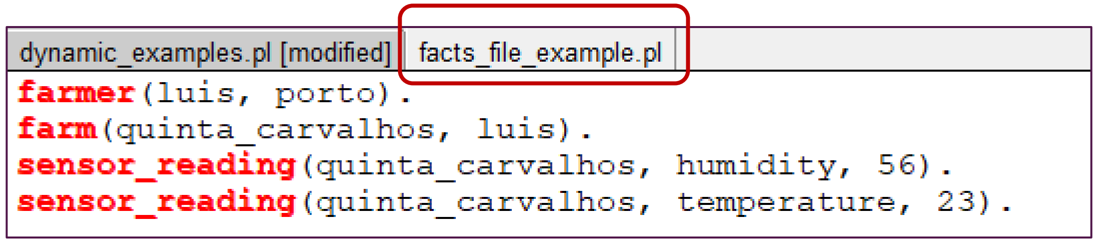
```
%Open and load facts from file
read_facts(File) :-
    open(File, read, Stream),
    repeat,
    read(Stream, Term),
    ( Term == end_of_file -> close(Stream), ! ;
      assert(Term), fail ).
```

- Compile the main file
  - do not compile the facts file!!! Save it only!

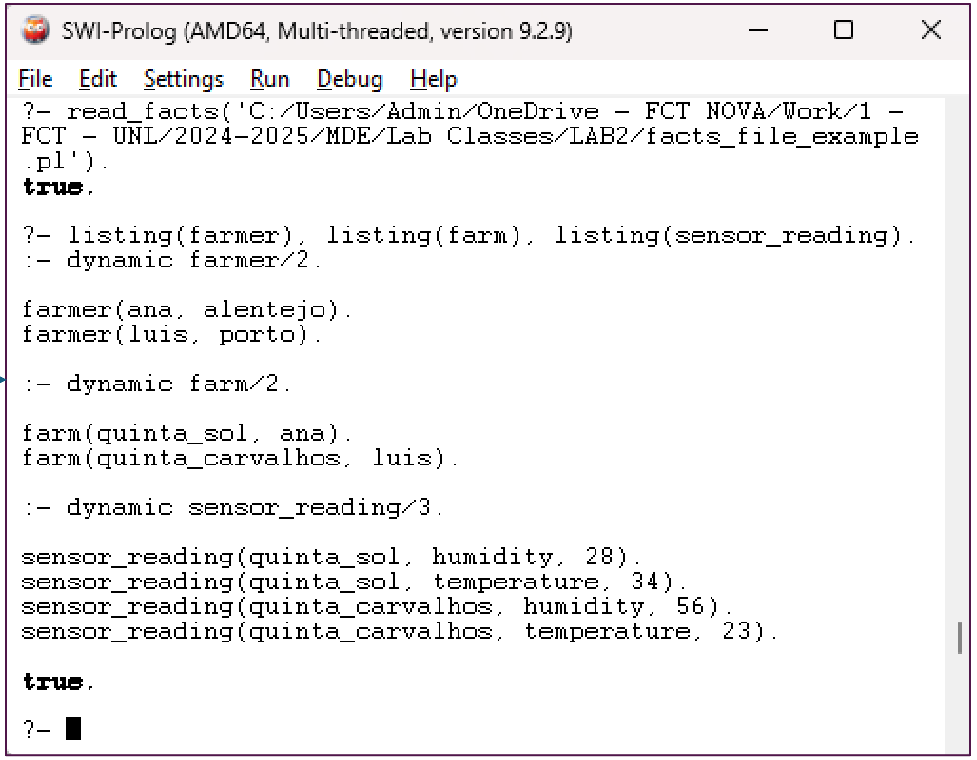
- Run `read_facts` predicate
  - The file *facts_file_example.pl* must be in the **same directory** from which you are running Prolog — or else you should use the **full path**, like this:

```
?- read_facts('c:/path/to/facts_file_example.pl').
```

- From Previously Saved File…



```
farmer(luis, porto).
farm(quinta_carvalhos, luis).
sensor_reading(quinta_carvalhos, humidity, 56).
sensor_reading(quinta_carvalhos, temperature, 23).
```



SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
?- read_facts('C:/Users/Admin/OneDrive - FCT NOVA/Work/1 -
FCT - UNL/2024-2025/MDE/Lab Classes/LAB2/facts_file_example
.pl').
true.

?- listing(farmer), listing(farm), listing(sensor_reading).
:- dynamic farmer/2.

farmer(ana, alentejo).
farmer(luis, porto).

:- dynamic farm/2.

farm(quinta_sol, ana).
farm(quinta_carvalhos, luis).

:- dynamic sensor_reading/3.

sensor_reading(quinta_sol, humidity, 28).
sensor_reading(quinta_sol, temperature, 34).
sensor_reading(quinta_carvalhos, humidity, 56).
sensor_reading(quinta_carvalhos, temperature, 23).

true.

?- 
```

Summary

✓ Declare changeable facts with `dynamic/1`

✓ Add facts using `assert/1`, `asserta/1` or `assertz/1`

✓ Remove facts using `retract/1` or `retractall/1`

✓ Interact with the user via `read/1` and `write/1`

✓ Process input loops (e.g., insert until 'end')

✓ Create and manipulate a menu.

✓ Read facts from a previously saved file.