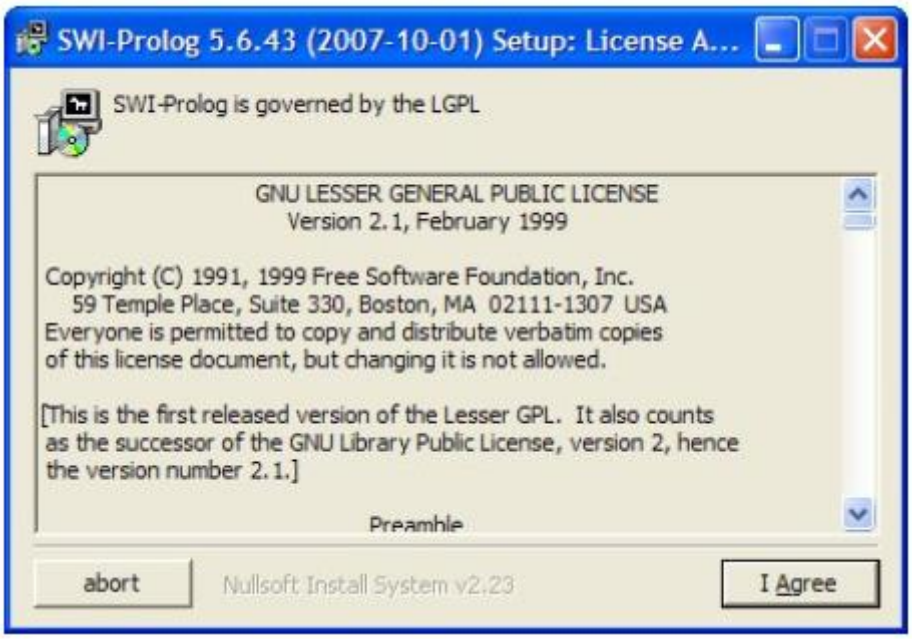
Lab 4: Introduction to Prolog

Tutorial 1: Installing SWI-Prolog

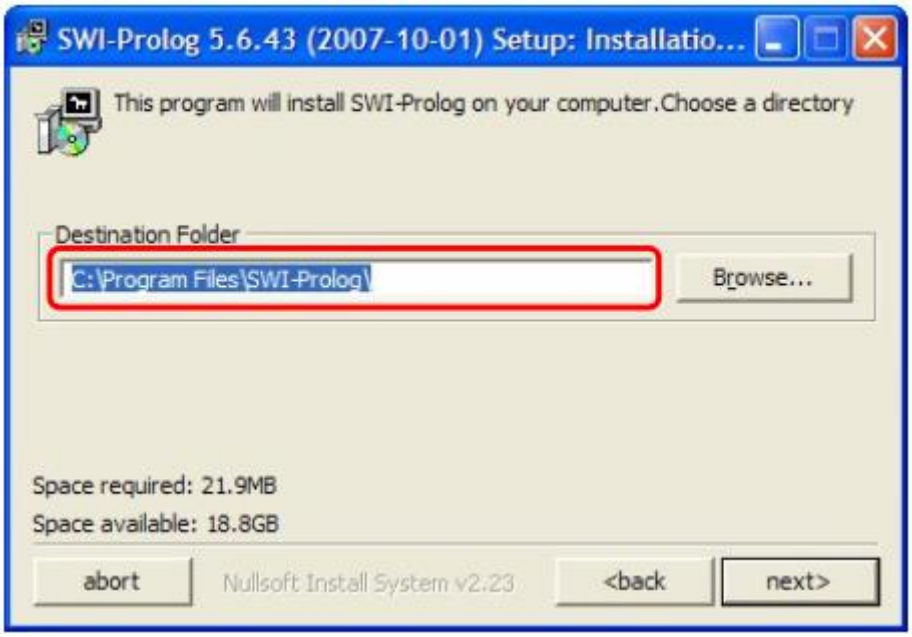
Step 1: Visit the SWI-Prolog website: http://www.swi-prolog.org/. Download the SWI-Prolog installation file for Windows. Make sure to download the correct version for Windows since SWI-Prolog has installation files for various platforms.

Step 2: Run the installation file to start the SWI-Prolog installation for Windows.

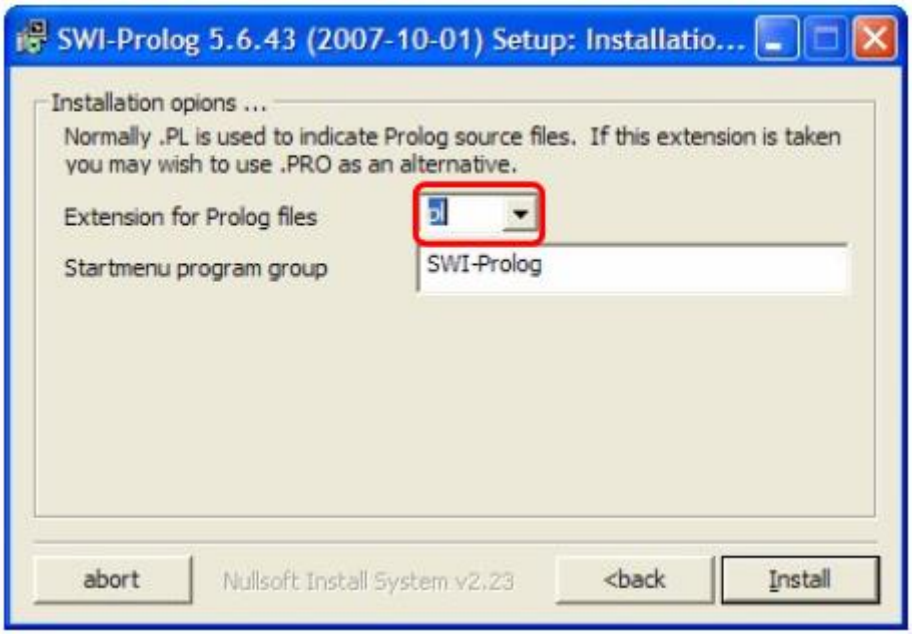


Step 3: Click **I Agree**, then click **Next** in the following dialog box.

Step 4: In the installation directory selection dialog, choose C:\Program Files\SWI-Prolog (instead of the default C:\Program Files\pl), as a full name is preferable.



Step 5: Click **Next**. In this dialog box, select the file extension for Prolog source files (pl or pro). Choose pl if no other file types have registered this extension (Perl source files also use pl). Otherwise, select pro. Here, we choose pl.



Step 6: Click **Install** and wait for SWI-Prolog to complete the installation.

For convenience, we refer to {SWI-Prolog} as the SWI-Prolog installation directory (the directory chosen in Step 4). After installation, this directory contains the entire program, DLLs, and other SWI-Prolog files. Below are some important files and directories:

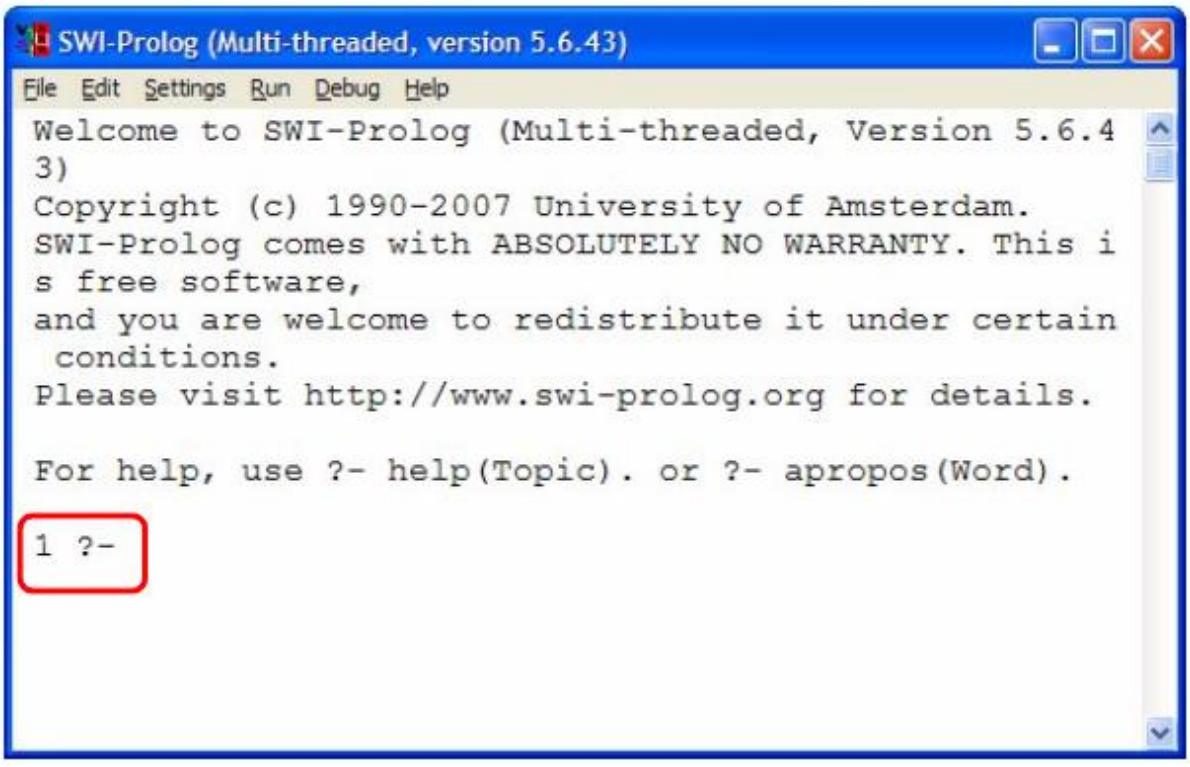
- {SWI-Prolog}\bin: Contains the executable files and DLLs of SWI-Prolog. The file {SWI-Prolog}\bin\plwin.exe is the executable program for SWI-Prolog. The file {SWI-Prolog}\bin\plcon.exe is the console-based executable.
- {SWI-Prolog}\library: Contains the source files of SWI-Prolog libraries.

- {SWI-Prolog}\doc: Contains web-based documentation. When running SWI-Prolog, access **Help → Online manual...** for assistance with similar content in the SWI-Prolog Help window. A PDF version of this guide can also be downloaded from the SWI-Prolog website.
- {SWI-Prolog}\demo: Contains demo source files for SWI-Prolog, including {SWI-Prolog}\demo\likes.pl, which is a demo source file.

SWI-Prolog source files use the .pl extension (as selected in Step 5) and contain declarations of facts and rules for the **Knowledge Base (KB)**. These files are associated with plwin.exe. When opening a Prolog source file, plwin.exe will launch, set the working directory to the file's location, and load the facts and rules into the KB.

Tutorial 2: Querying

Run the SWI-Prolog program (plwin.exe). The main interface of SWI-Prolog appears as shown below.



After the welcome message, a prompt appears, awaiting queries. Enter a query after the Prolog prompt (n ?- where n is the query number). Type the query, followed by a period (.), and press **Enter** to execute. If there are syntax errors or runtime errors, SWI-Prolog will display an error message. Otherwise, it will return the query result. If a query has multiple solutions, type ; to see the next solution. Otherwise, press **Enter** to finish and move to the next query.

SWI-Prolog queries can be categorized into two types:

1. **Queries expecting a unification result**: These queries usually contain variables and may have multiple unification values.
2. **Queries with side effects**: These queries typically do not contain variables and execute specific operations. They are often referred to as **commands**.

Example Queries

Step 1: Double-click the file {SWI-Prolog}\demo\likes.pl to launch SWI-Prolog and load the facts and rules from this source file into the KB.

Step 2: Query listing. to view the facts and rules in the KB.

```
1 ?- listing.

likes(sam, A) :-
        indian(A),
        mild(A).
likes(sam, A) :-
        chinese(A).
likes(sam, A) :-
        italian(A).
likes(sam, chips).

chinese(chow_mein).
chinese(chop_suey).
chinese(sweet_and_sour).

mild(dahl).
mild(tandoori).
mild(kurma).

italian(pizza).
italian(spaghetti).

indian(curry).
indian(dahl).
indian(tandoori).
indian(kurma).

Yes
```

Step 3: Query mild(dahl).

```
2 ?- mild(dahl).

Yes
```

Step 4: Query indian(X).

```
3 ?- indian(X).
X = curry
```

Step 5: Press ; to view more solutions.

```
3 ?- indian(X).
X = curry ;
X = dahl
```

Step 6: Press **Enter** to stop the current query.

```
3 ?- indian(X).
X = curry ;
X = dahl

Yes
```

Step 7: Query likes(sam, X). and use ; to view all solutions.

```
4 ?- likes(sam, X).
X = dahl ;
X = tandoori ;
X = kurma ;
X = chow_mein ;
X = chop_suey ;
X = sweet_and_sour ;
X = pizza ;
X = spaghetti ;
X = chips
5 ?-
```

Tutorial 3: Working with SWI-Prolog Source Files

Step 1: Launch SWI-Prolog via plwin.exe (without loading a source file into the KB). Select **File → New...** to create a new source file. In the **Create new Prolog source** dialog, name the file hello.pl. SWI-Prolog will open Notepad for editing.

Step 2: Enter the following rule in hello.pl:

```
hello :- write('Hello World').
```

Save and close Notepad.

Step 3: Select **File → Consult...** to load a source file. Choose hello.pl. SWI-Prolog will load its facts and rules into the KB.

Step 4: Query listing. to verify that the rule has been added to the KB.

```
1 ?- listing.
```

```
hello :-
        write('Hello World').

Yes
```

Step 5: Query hello. to execute the rule.

```
2 ?- hello.

Hello World


Yes
```

Step 6: Select **File** → **Edit...** to modify hello.pl. Change its contents to:

```
hello(X) :- write('Hello '), write(X).
```

Save and close Notepad.

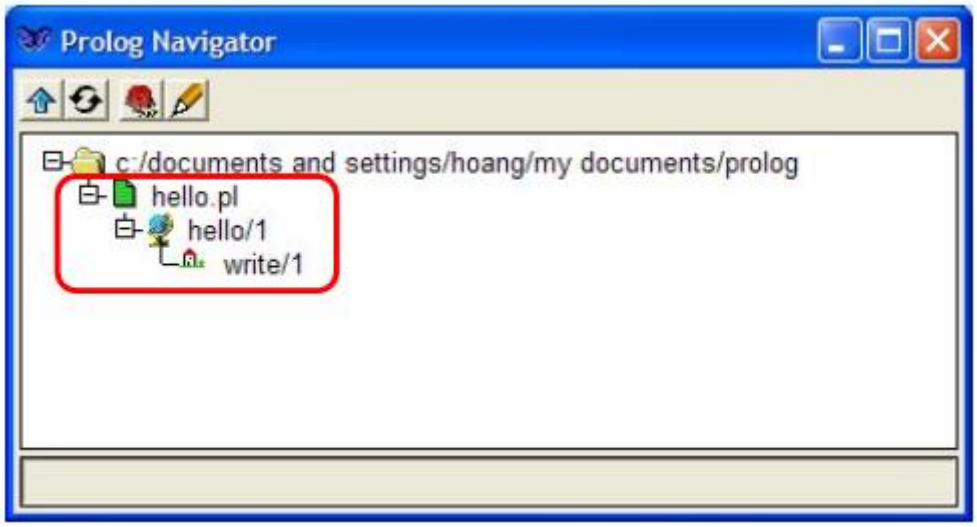
Step 7: Query listing. to see that the changes are automatically updated in the KB.

```
3 ?- listing.
hello(A) :-
        write('Hello '),
        write(A).

Yes
```

Step 8: Query hello(nam). to execute the new rule.

```
4 ?- hello(nam).
Hello nam

Yes
```

Step 9: Open the **Prolog Navigator** window by selecting **File** → **Navigator...**. Locate the node corresponding to the hello.pl source file. This node contains the hello/1 predicate, and it also shows its dependency on the write/1 predicate.



Step 10: Open Notepad independently and modify hello.pl to contain only the following rule:

```
hello :- write('Hello World').
```

Save and close Notepad.

Step 11: Query listing. to verify that the KB has not changed.

```
5 ?- listing.
hello(A) :-
        write('Hello '),
        write(A).

Yes
```

Step 12: Select **File → Reload modified files** to reload hello.pl in SWI-Prolog. Query listing. again to confirm the changes in the KB.

```
6 ?- listing.

hello :-

   write('Hello World').


Yes
```

Note:

- When loading a source file, its facts and rules are added to the KB without erasing the previous KB contents. SWI-Prolog maintains a list of loaded files, and selecting **File → Reload modified files** will reload all these files into the KB.

- Some SWI-Prolog predicates perform similar functions as the **File** menu commands. These are frequently used and will be covered in future tutorials:
    - make/0 reloads all loaded files.
    - edit/0, edit/1 edits a source file.
    - edit(file(File)) creates a new source file.
    - consult/1 or its shorthand [File] loads a source file.
- **Edit** → **Copy** and **Edit** → **Paste** allow copying and pasting text. Note that Ctrl + V works for pasting, but Ctrl + C is reserved for interrupting a running query (**Run** → **Interrupt**).

Ref: Nhóm GVHD TH Lập trình Logic, Hướng dẫn SWI-Prolog, ĐH KHTN, 2007.

Exercises

Exercise 1: Installing and Verifying SWI-Prolog

Objective: Ensure that you have successfully installed SWI-Prolog and can navigate the interface.

1. Download and install SWI-Prolog following the instructions from the tutorial.
2. Open SWI-Prolog (plwin.exe) and verify the installation by executing a simple query:

   write('SWI-Prolog Installed Successfully').
3. If the message appears correctly, your installation is complete.

Exercise 2: Querying the Knowledge Base

Objective: Learn how to query facts and rules in a loaded Prolog source file.

1. Open likes.pl from the {SWI-Prolog}\demo folder.
2. Run the listing. query to display all stored facts and rules.
3. Perform the following queries and record the outputs:

   mild(X).

   indian(X).

   likes(sam, X).
4. Use ; to find all possible answers for likes(sam, X).
5. Explain the results.

Exercise 3: Writing and Executing a Simple Rule

Objective: Create a new Prolog source file and define a simple rule.

1. Open SWI-Prolog and create a new file named greeting.pl.
2. Add the following rule in the file:

   greet :- write('Hello, welcome to Prolog!').
3. Save the file and consult it in SWI-Prolog using:

   [greeting].
4. Execute the query:

   greet.
5. Observe and record the output.
6. Explain the results.

Exercise 4: Modifying and Reloading a Rule

Objective: Modify an existing rule and reload it without restarting SWI-Prolog.

1. Edit greeting.pl and modify the rule to accept a name:

   greet(Name) :- write('Hello, '), write(Name), write('! Welcome to Prolog!').
2. Save the file and reload it in SWI-Prolog using:

   make.
3. Execute the query:

   greet('Alice').
4. Observe the change in output and ensure the modification was successfully updated.
5. Explain the results.

Exercise 5: Creating and Querying a Custom Knowledge Base

Objective: Define and query relationships in a custom knowledge base.

1. Create a new Prolog file named family.pl.
2. Define the following family relationships:

   parent(john, mary).

   parent(mary, susan).

   parent(john, peter).

parent(peter, james).

ancestor(X, Y) :- parent(X, Y).

ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

3. Save the file and consult it in SWI-Prolog.
4. Run the following queries and analyze the results:

   parent(john, X).

   ancestor(john, X).

   ancestor(mary, james).
5. Use ; to retrieve all possible answers for the ancestor queries.
6. Explain the results.

Exercise 6: Implementing a Recursive List Processing Predicate

Objective: Write and test a recursive predicate to calculate the sum of elements in a list.

1. Create a new Prolog file named list_operations.pl.
2. Define a predicate sum_list/2 that calculates the sum of a list of numbers:

   sum_list([], 0).

   sum_list([H|T], Sum) :- sum_list(T, Rest), Sum is H + Rest.
3. Save and consult the file.
4. Execute the following queries:

   sum_list([1, 2, 3, 4, 5], X).

   sum_list([], X).
5. Verify that the output matches the expected sum.
6. Explain the results.