Paradigma de Linguagem de Programação

Programação Lógica

Prolog

Prolog

- Criada por Robert Kowalski, Alain Colmerauer e Phillippe Roussel, na década de 70.
- Algumas aplicações :
  - ✓ Sistemas Baseados em Conhecimento.
  - ✓ Sistemas Especialistas.
  - ✓ Processamento de Linguagem Natural.
- Não é uma linguagem procedimental como C++ ou Pascal, e sim uma linguagem declarativa (não é preciso definir passo a passo o que deve ser feito e sim descrever o problema).
- Pode provar teses a partir de hipóteses.
- Inclui predicados, conectivos lógicos e regras de inferência (princípio da resolução).

Princípio da resolução: regra de inferência que dá origem a uma técnica de demonstração por dedução de uma contradição (refutação).

Conceitos básicos sobre Prolog

- Na lógica de predicados usamos regras de inferência para demonstrar que uma tese é consequência de certas hipóteses.

- As declarações de um programa formam uma base de dados e para determinar se uma tese (consulta de um usuário à base de dados) é verdadeira ou não, Prolog aplica suas regras de inferência nesta base.

- Um programa em Prolog (***Pro****gramming* ***Log****ic*) é uma declaração de hipóteses e normalmente se divide em:

  - ✓ Declaração de fatos sobre objetos e suas relações.
  - ✓ Definição de regras.
  - ✓ Consultas.

Prolog

- As bases de dados convencionais descrevem apenas os fatos, ao passo que Prolog registra também as regras. E baseando-se nestas regras pode-se gerar novos fatos.

Fato	Regra	Dedução de um novo fato
Rufos é um vira-lata.	Todo vira-lata é um cachorro.	Rufos é um cachorro.

- Operadores:

Linguagem natural	Cálculo de predicados	Prolog
E	$\wedge$	**,**
OU	$\vee$	**;**
SE	$\leftarrow$	**:-**
NÃO	$\neg$	**not**

Prolog

- As sentenças são expressas por cláusulas e as cláusulas se dividem em fatos e regras.
  - ✓ **Fato** é a declaração de uma verdade incondicional (os fatos permitem definir os predicados).
  - ✓ **Regra** é a condição que deve ser satisfeita para que uma declaração seja considerada verdadeira.
- Exemplos:

```
rato(jerry).
```

Este é um exemplo de FATO:

"rato" é o predicado (cabeça) e "jerry" é o argumento do predicado .

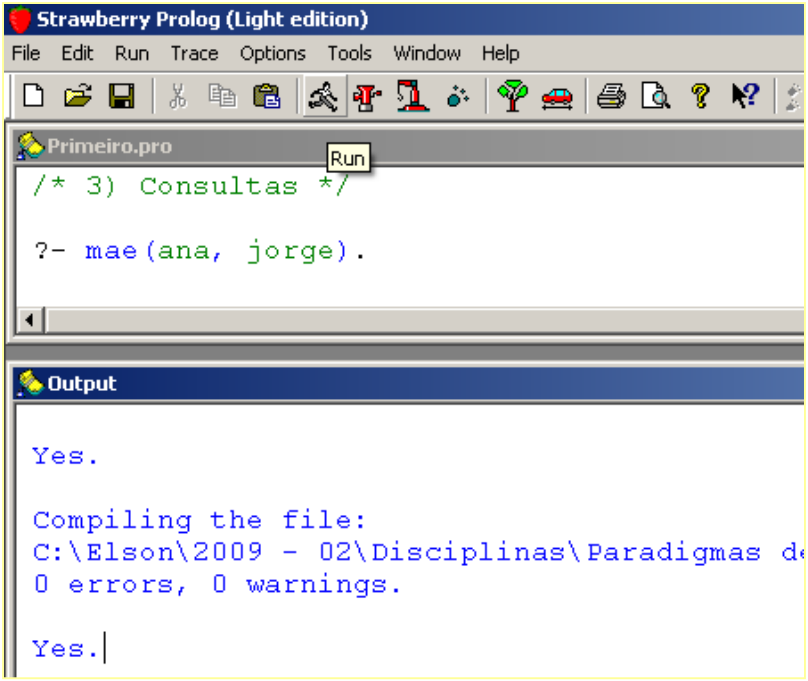
```
roedor(X) :- rato(X); toupeira(X).
```

Este é um exemplo de REGRA:

Observe o uso do operador "OU" (;) para determinar a validade da regra.

Exemplo 1

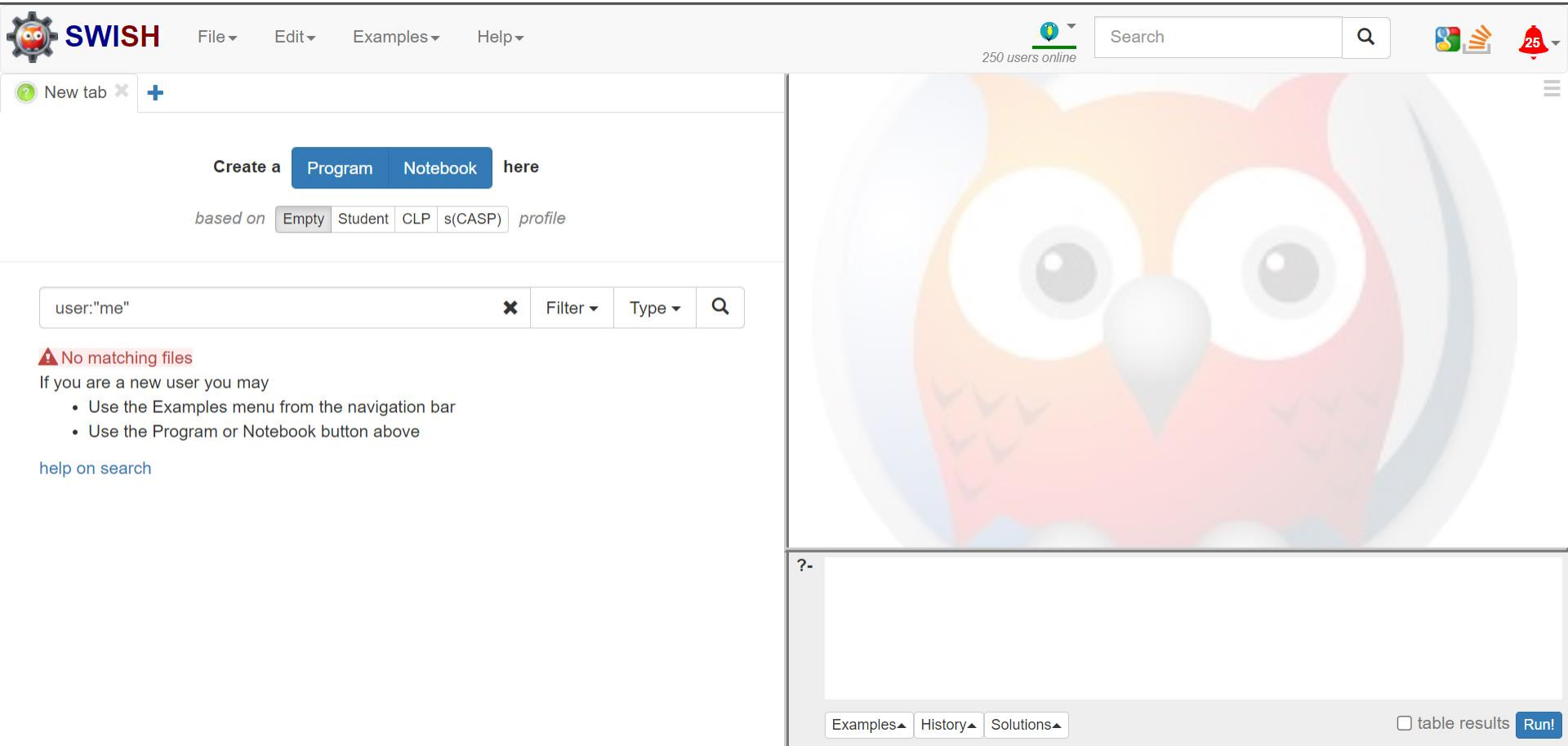
```
/* 1) Fatos */
progenitor(maria, rodrigo).      /* Predicado binário */
progenitor(emanuel, rodrigo).

progenitor(rodrigo, julia).
progenitor(silvia,  julia).

progenitor(rodrigo, jorge).
progenitor(ana, jorge).

masculino(emanuel).  /* Predicado unário */
masculino(rodrigo).
masculino(jorge).

feminino(maria).
feminino(ana).
feminino(julia).
feminino(silvia).


/* 2) Regras */
irmao(X, Y) :- progenitor(Z, X), progenitor(Z, Y), masculino(X).
irma(X, Y) :- progenitor(Z, X), progenitor(Z, Y), feminino(X).

pai(X, Y) :- progenitor(X, Y), masculino(X).
mae(X, Y) :- progenitor(X, Y), feminino(X).

/* 3) Consultas */
?- mae(ana, jorge).
```



Observe que ao executarmos o código, o interpretador do Strawberry Prolog nos diz "Yes", ou seja, "Ana é mãe de Jorge". Procure analisar os fatos e regras, assim começará a perceber a recursividade (backtracking) do mecanismo de inferência.

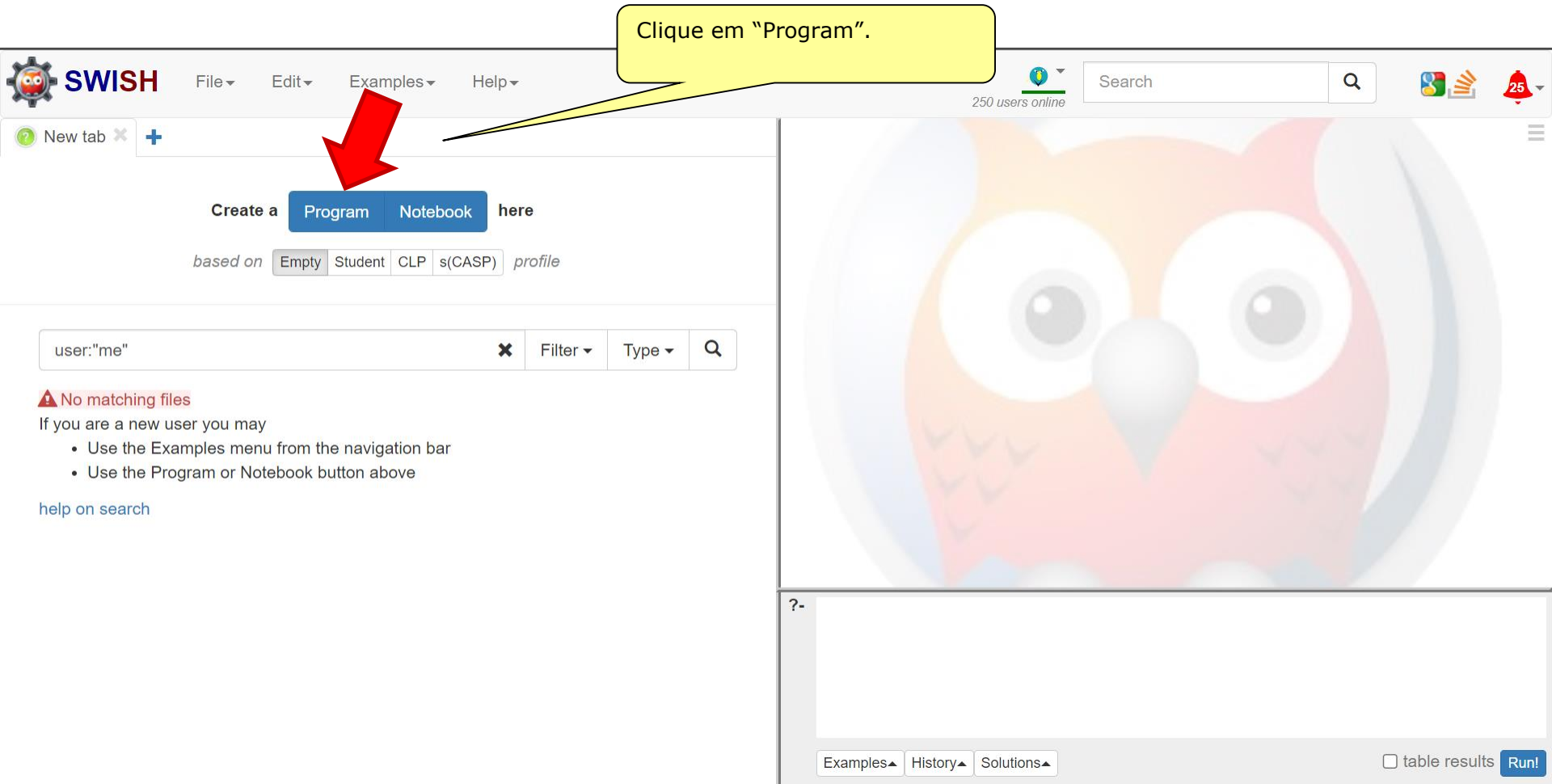
SWISH - Prolog

- https://swish.swi-prolog.org/



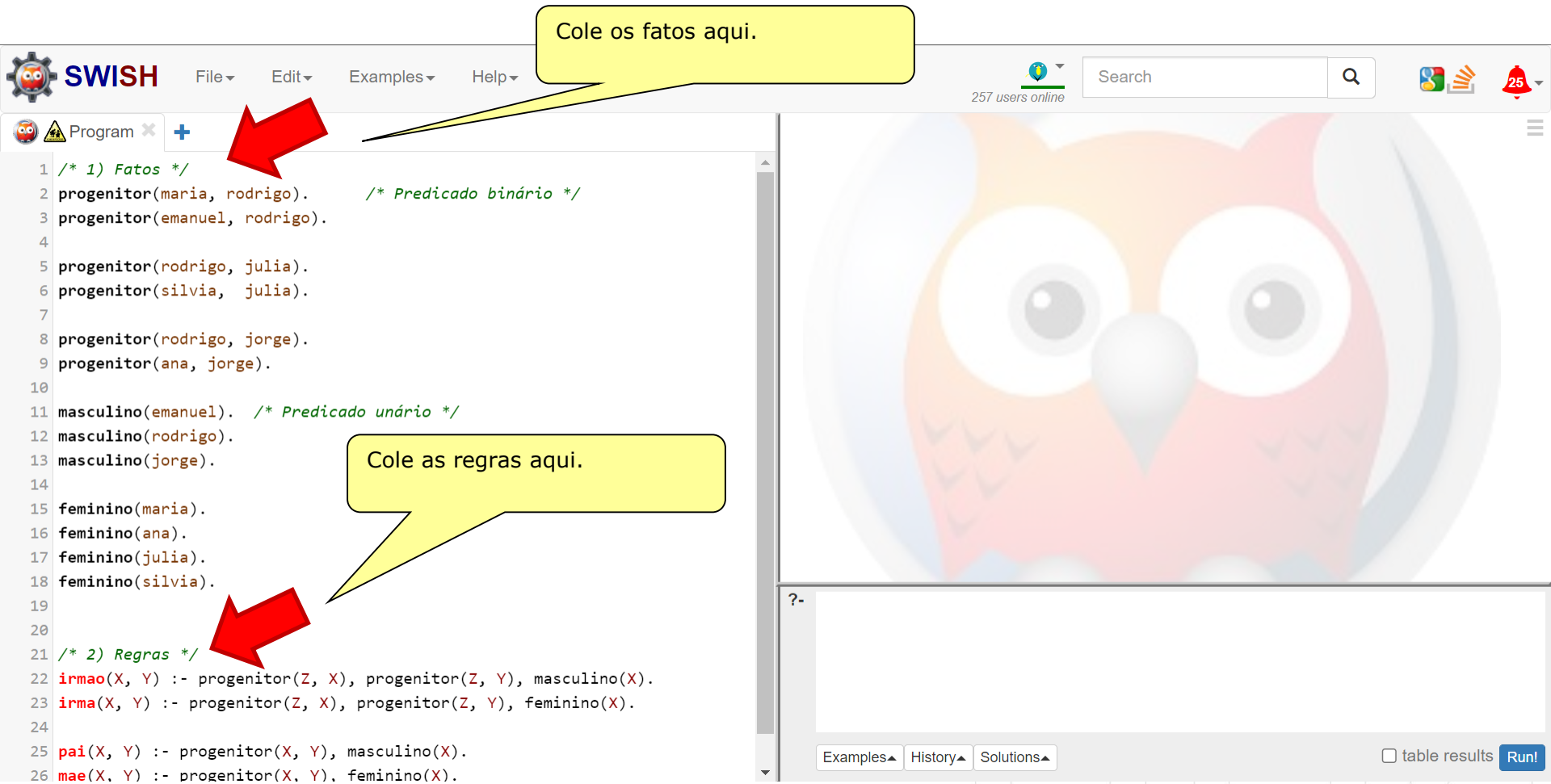
SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

- https://swish.swi-prolog.org/

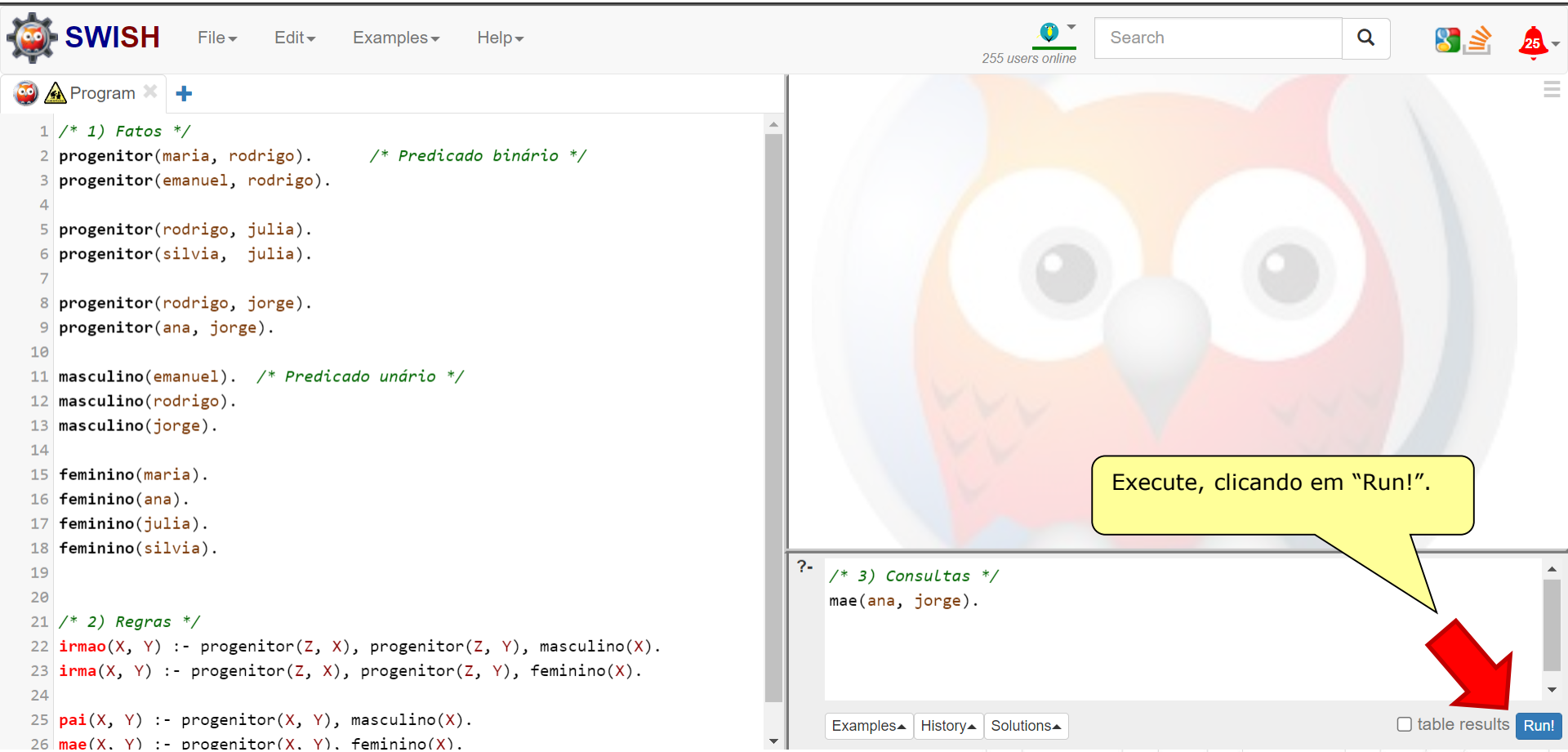


SWISH - Prolog

- https://swish.swi-prolog.org/

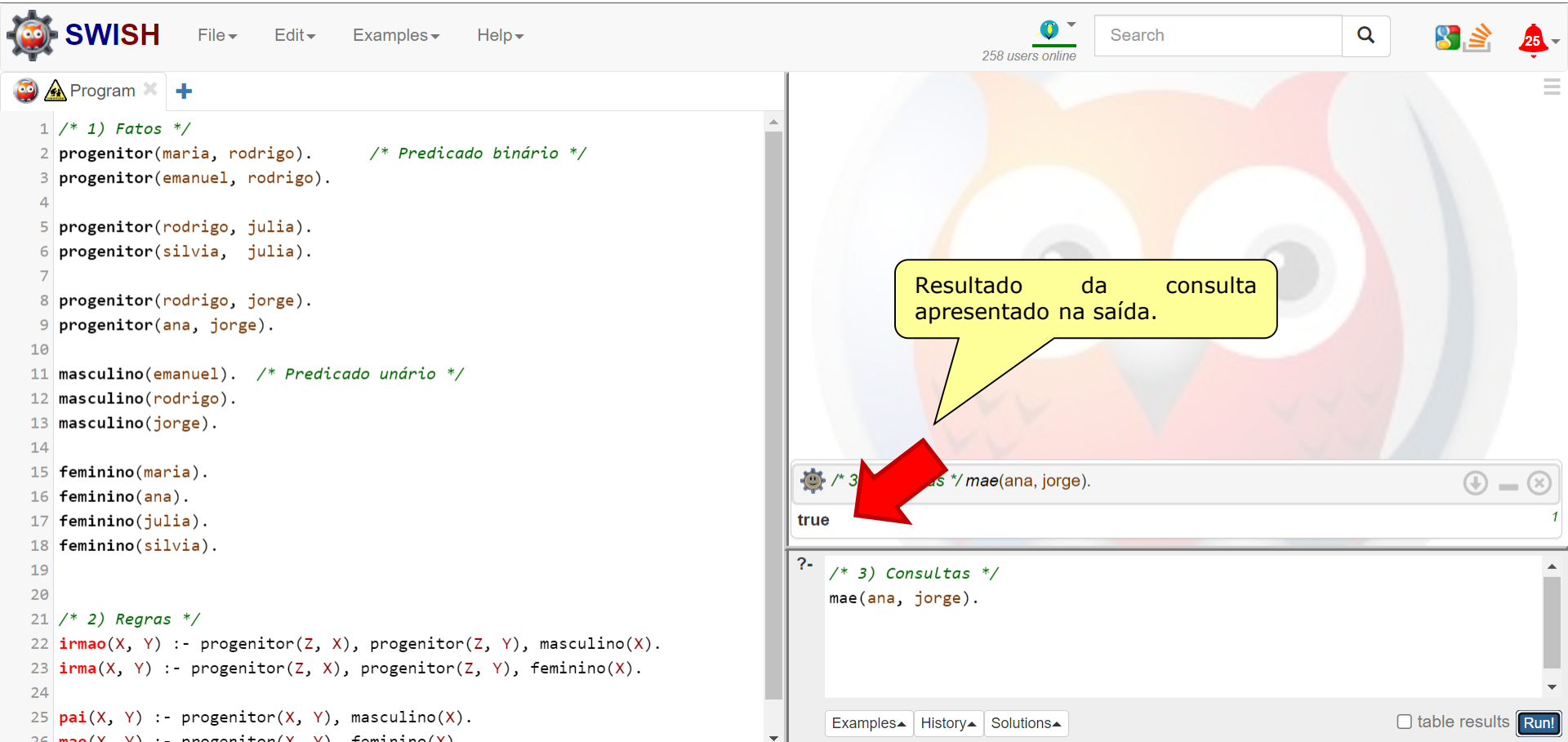
SWISH - Prolog

- https://swish.swi-prolog.org/



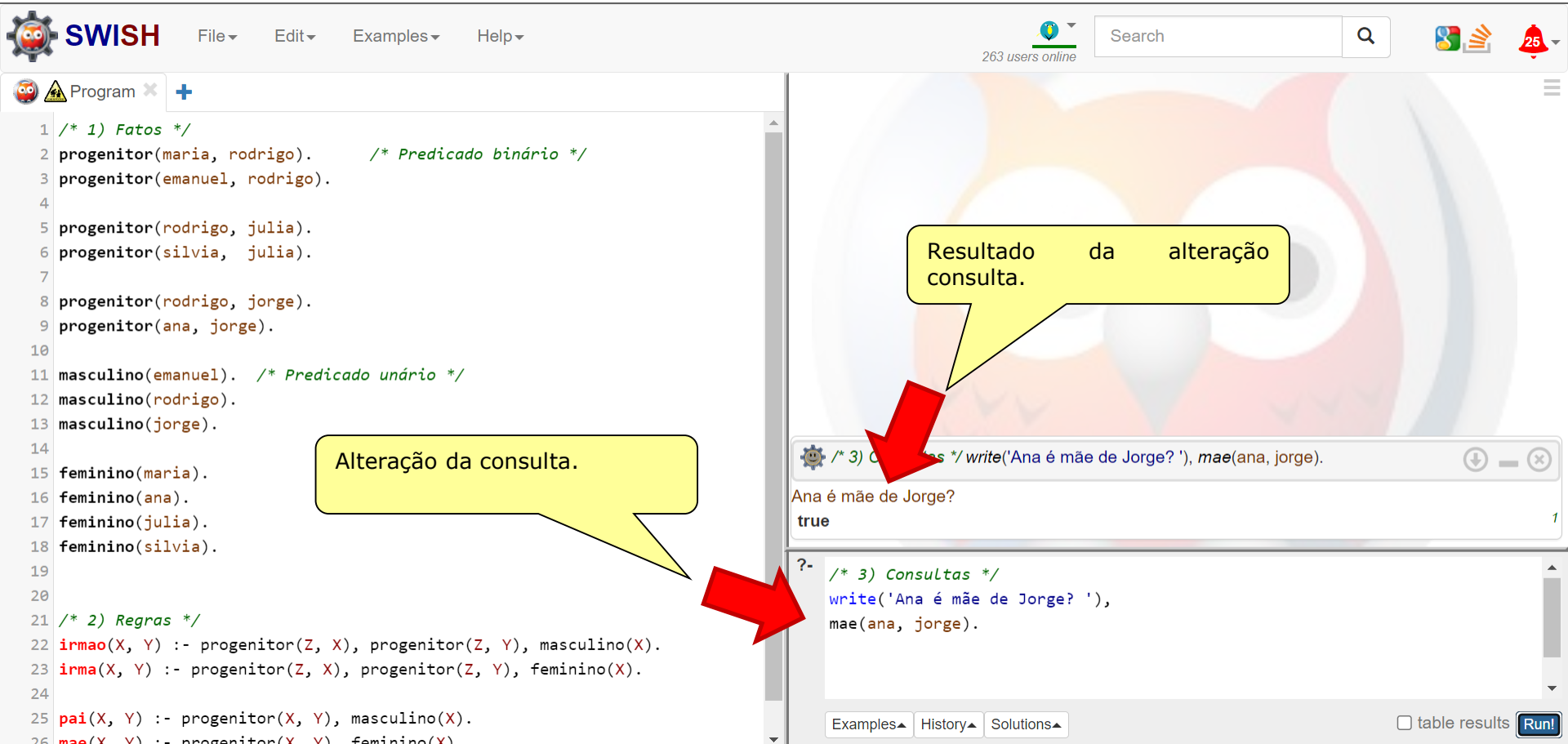
SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

- https://swish.swi-prolog.org/

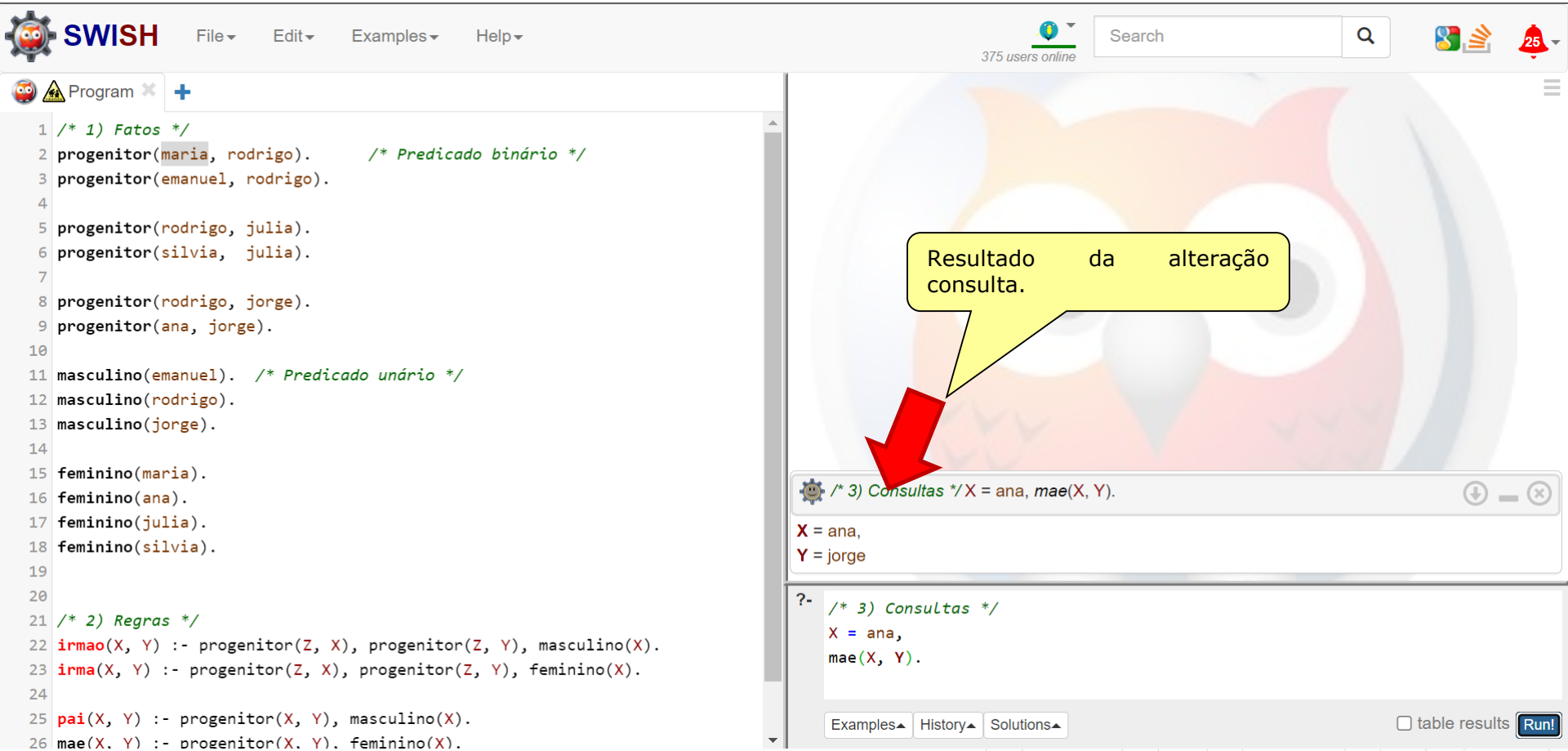


SWISH - Prolog

- https://swish.swi-prolog.org/

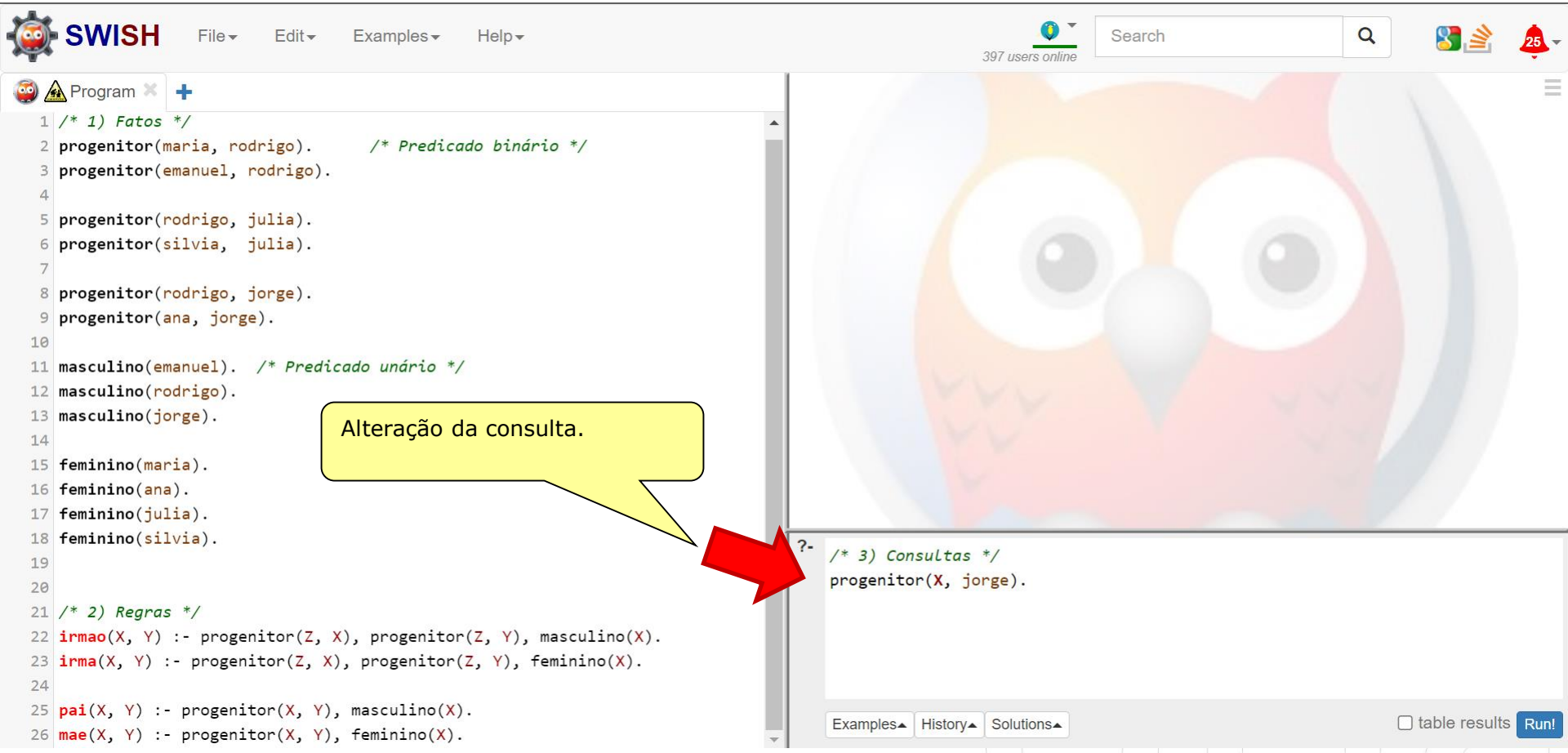
SWISH - Prolog

- https://swish.swi-prolog.org/



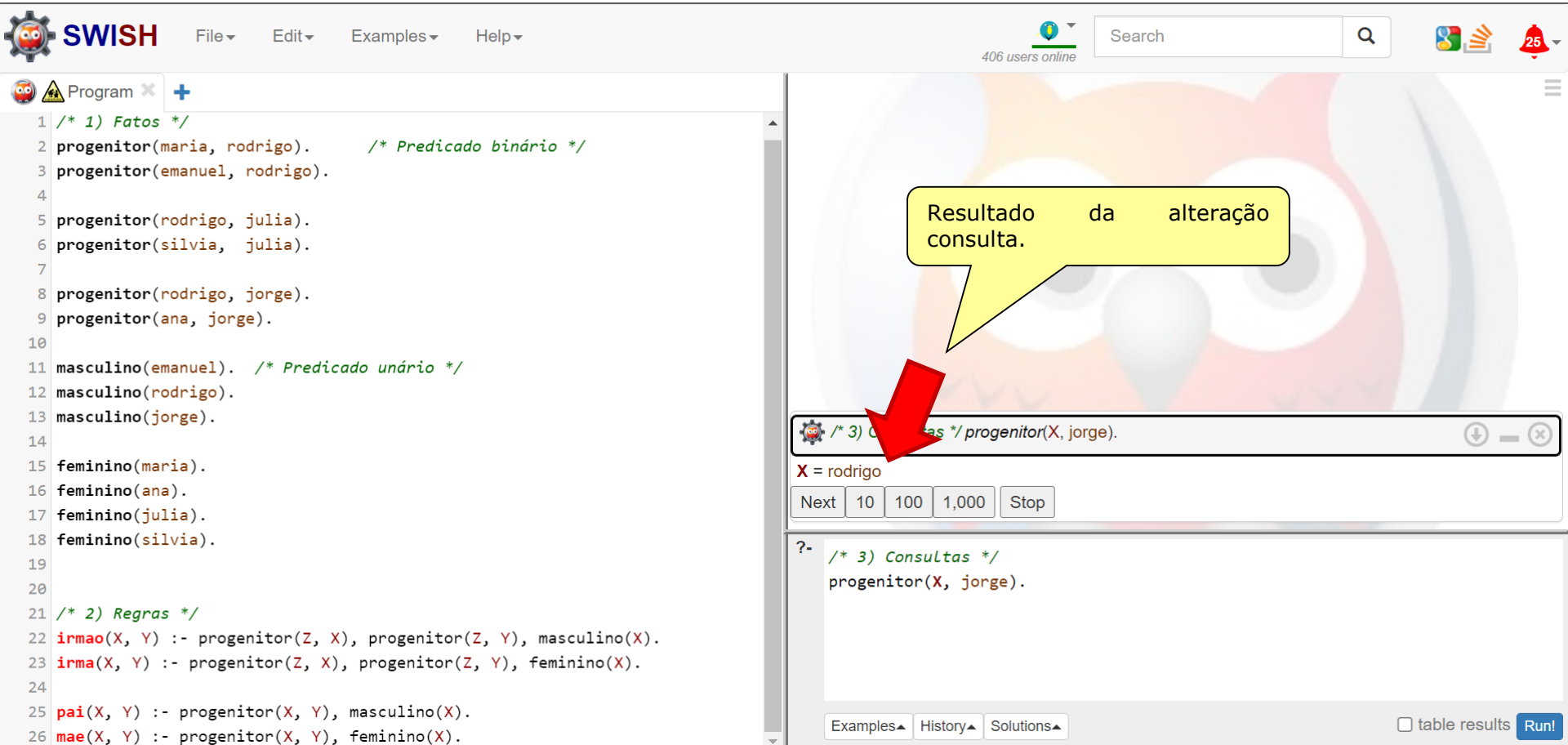
SWISH - Prolog

- https://swish.swi-prolog.org/



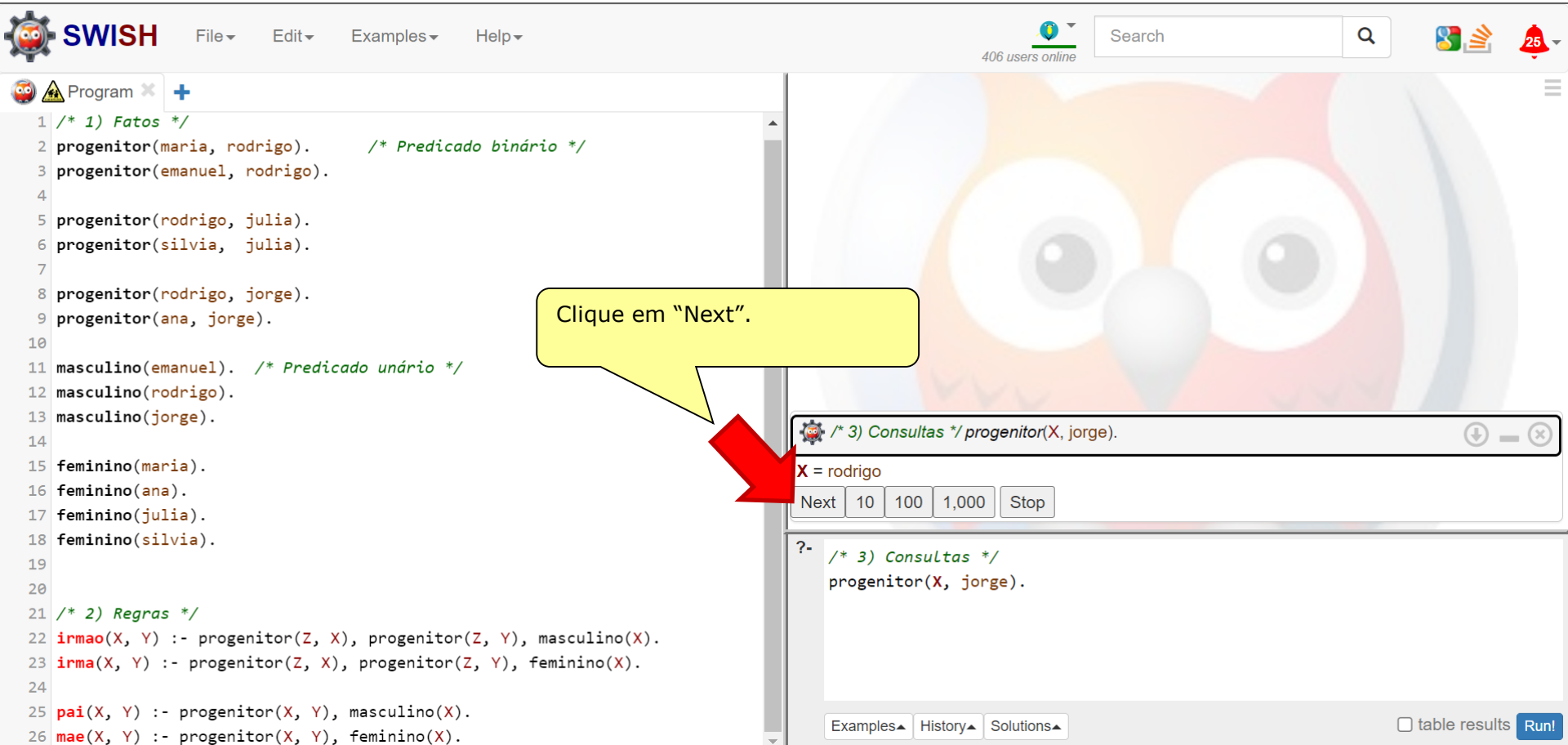
SWISH - Prolog

- https://swish.swi-prolog.org/



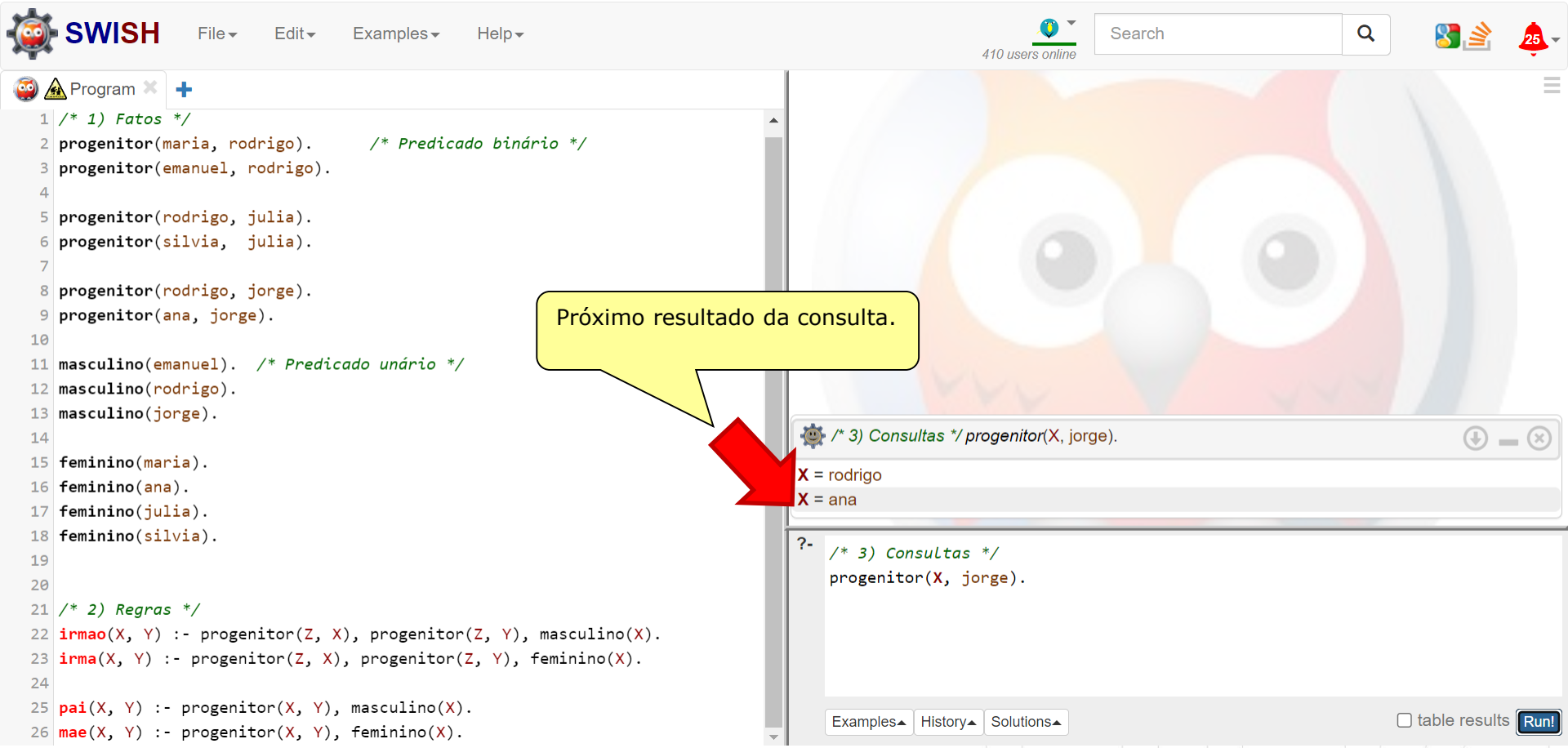
SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

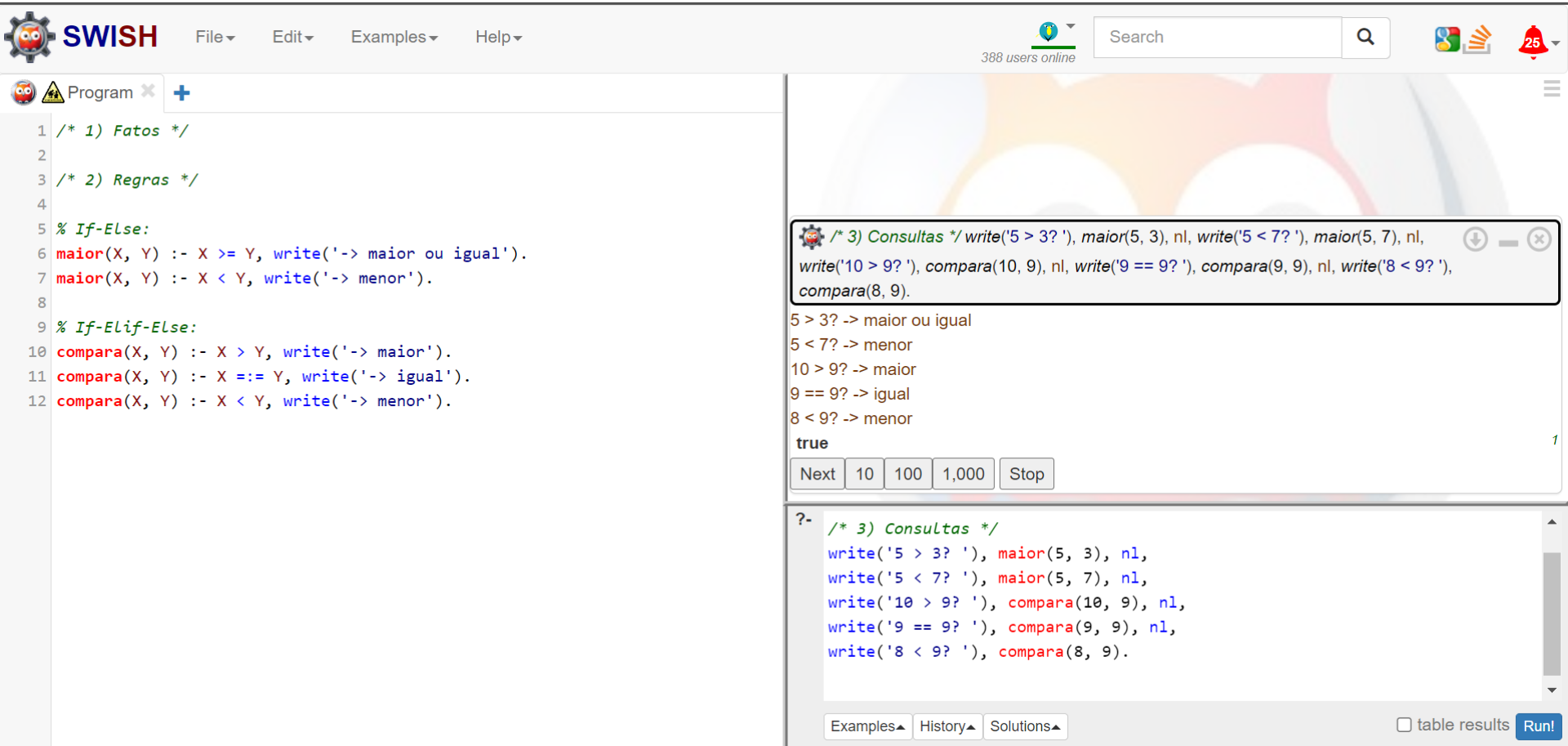
- https://swish.swi-prolog.org/

```
/* 1) Fatos */

/* 2) Regras */

% If-Else:
maior(X, Y) :- X >= Y, write('-> maior ou igual').
maior(X, Y) :- X < Y, write('-> menor').

% If-Elif-Else:
compara(X, Y) :- X > Y, write('-> maior').
compara(X, Y) :- X =:= Y, write('-> igual').
compara(X, Y) :- X < Y, write('-> menor').
```

```
/* 3) Consultas */
write('5 > 3? '), maior(5, 3), nl,
write('5 < 7? '), maior(5, 7), nl,
write('10 > 9? '), compara(10, 9), nl,
write('9 == 9? '), compara(9, 9), nl,
write('8 < 9? '), compara(8, 9).
```

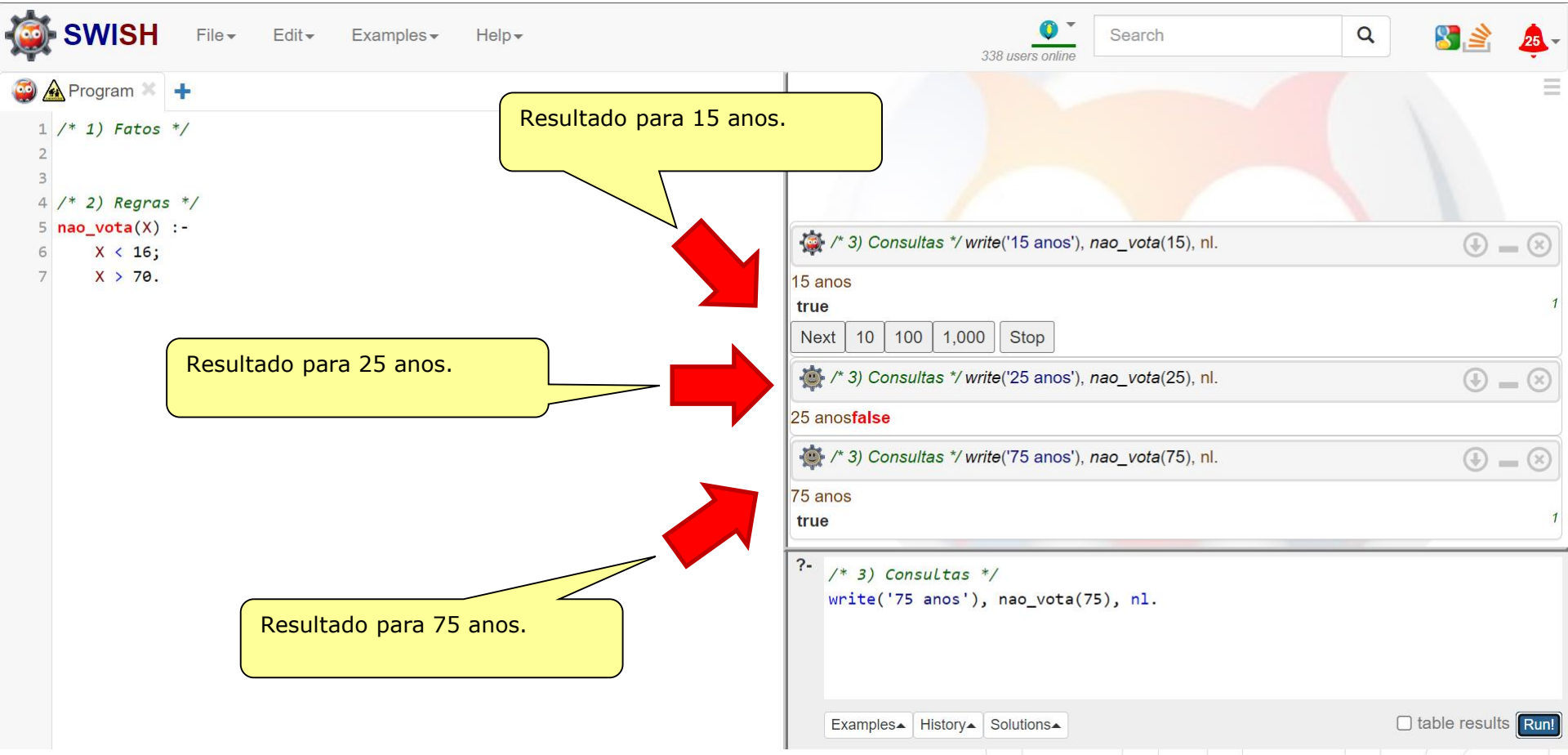
SWISH - Prolog

- https://swish.swi-prolog.org/



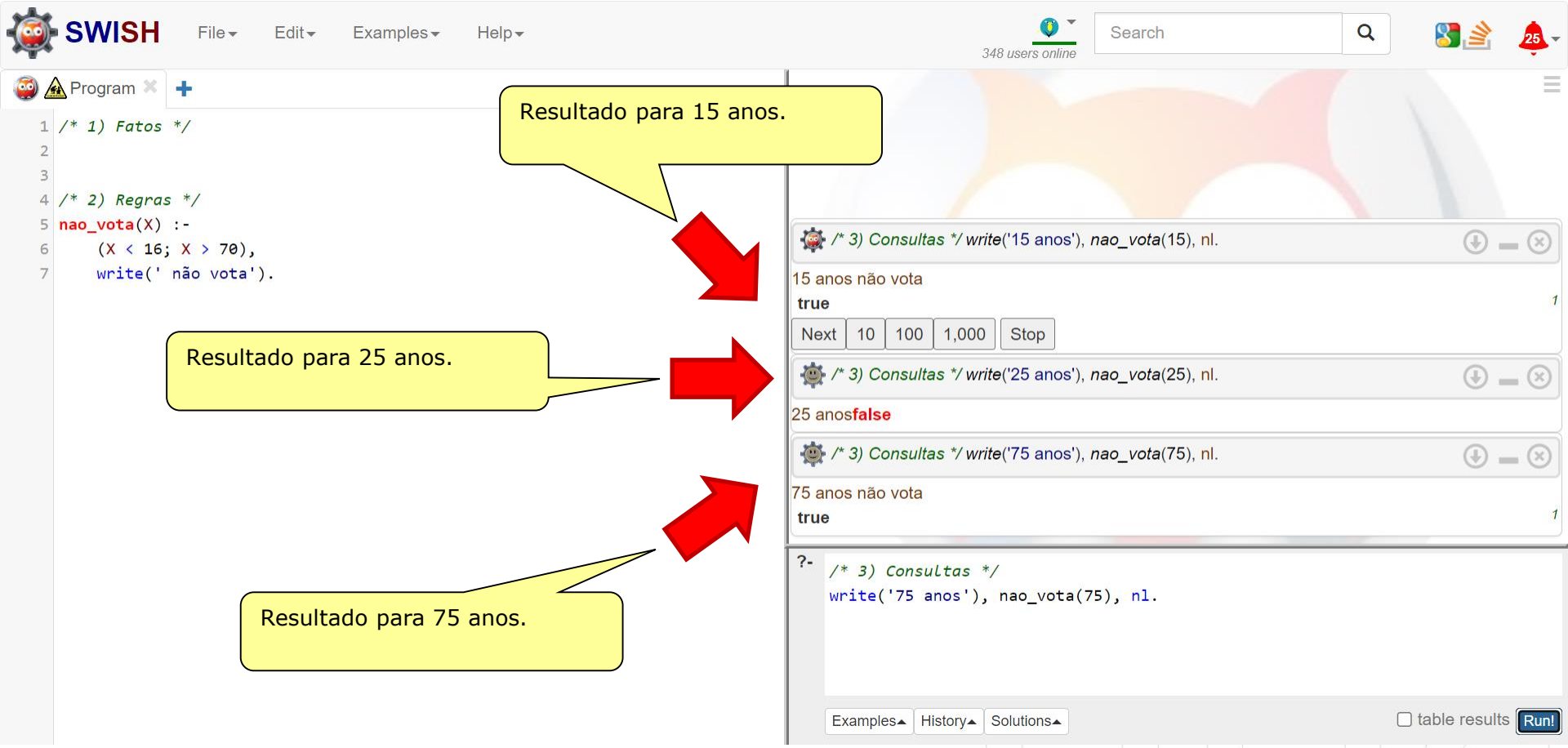
SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

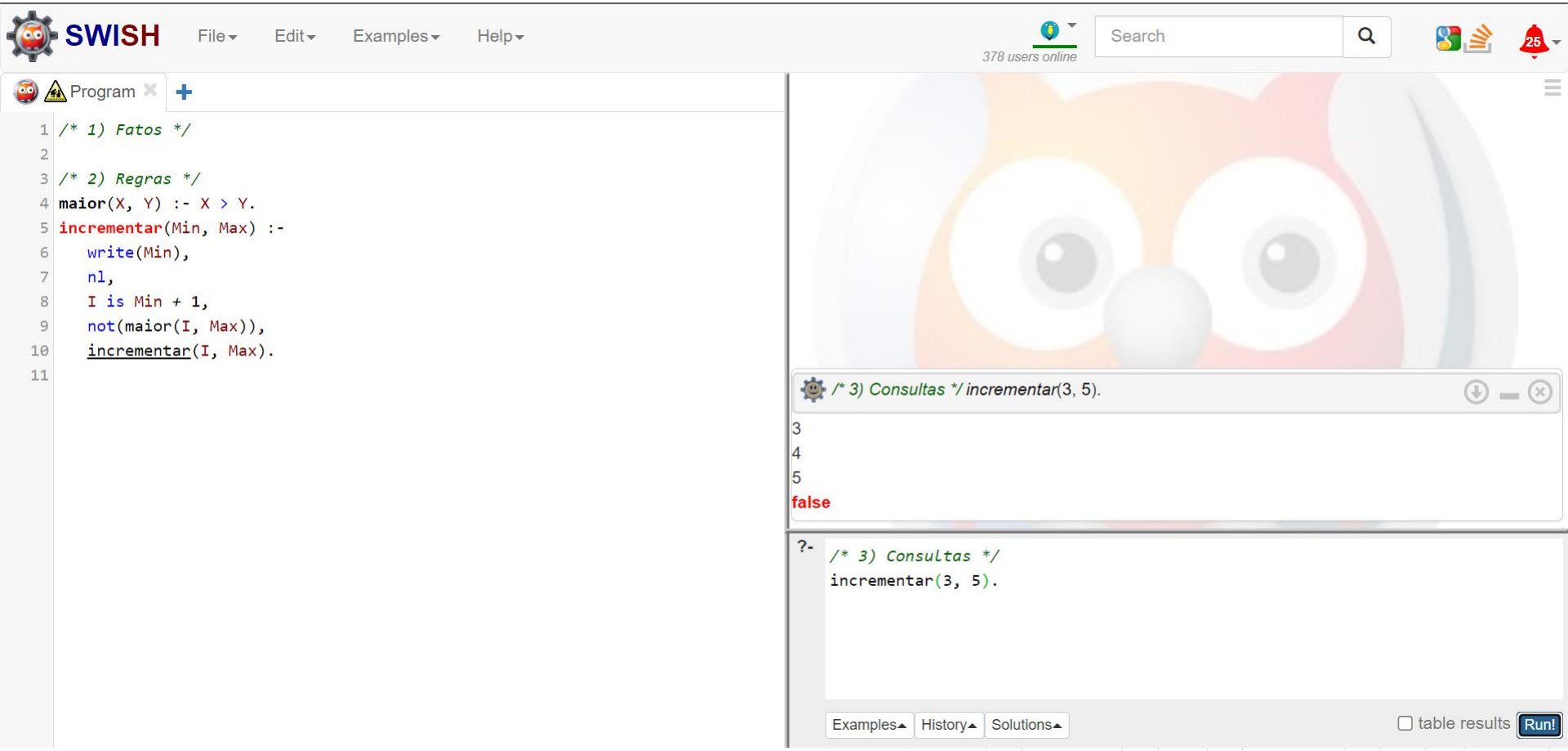
- https://swish.swi-prolog.org/

```
/* 1) Fatos */

/* 2) Regras */
maior(X, Y) :- X > Y.
incrementar(Min, Max) :-
  write(Min),
  nl,
  I is Min + 1,
  not(maior(I, Max)),
  incrementar(I, Max).
```

```
/* 3) Consultas */
incrementar(3, 5).
```

SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

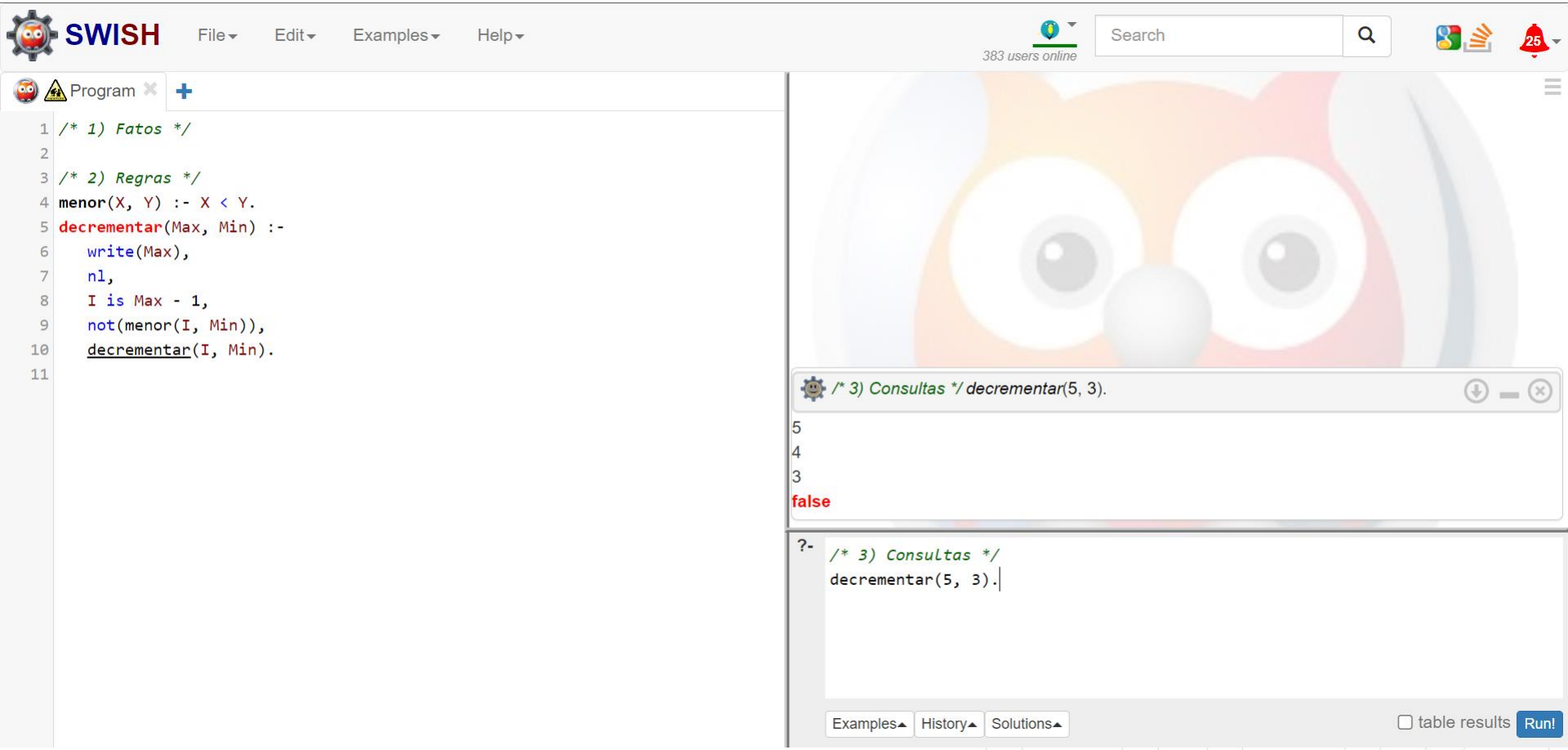
- https://swish.swi-prolog.org/

```
/* 1) Fatos */

/* 2) Regras */
menor(X, Y) :- X < Y.
decrementar(Max, Min) :-
  write(Max),
  nl,
  I is Max - 1,
  not(menor(I, Min)),
  decrementar(I, Min).
```

```
/* 3) Consultas */
decrementar(5, 3).
```

SWISH - Prolog

- https://swish.swi-prolog.org/



SWISH - Prolog

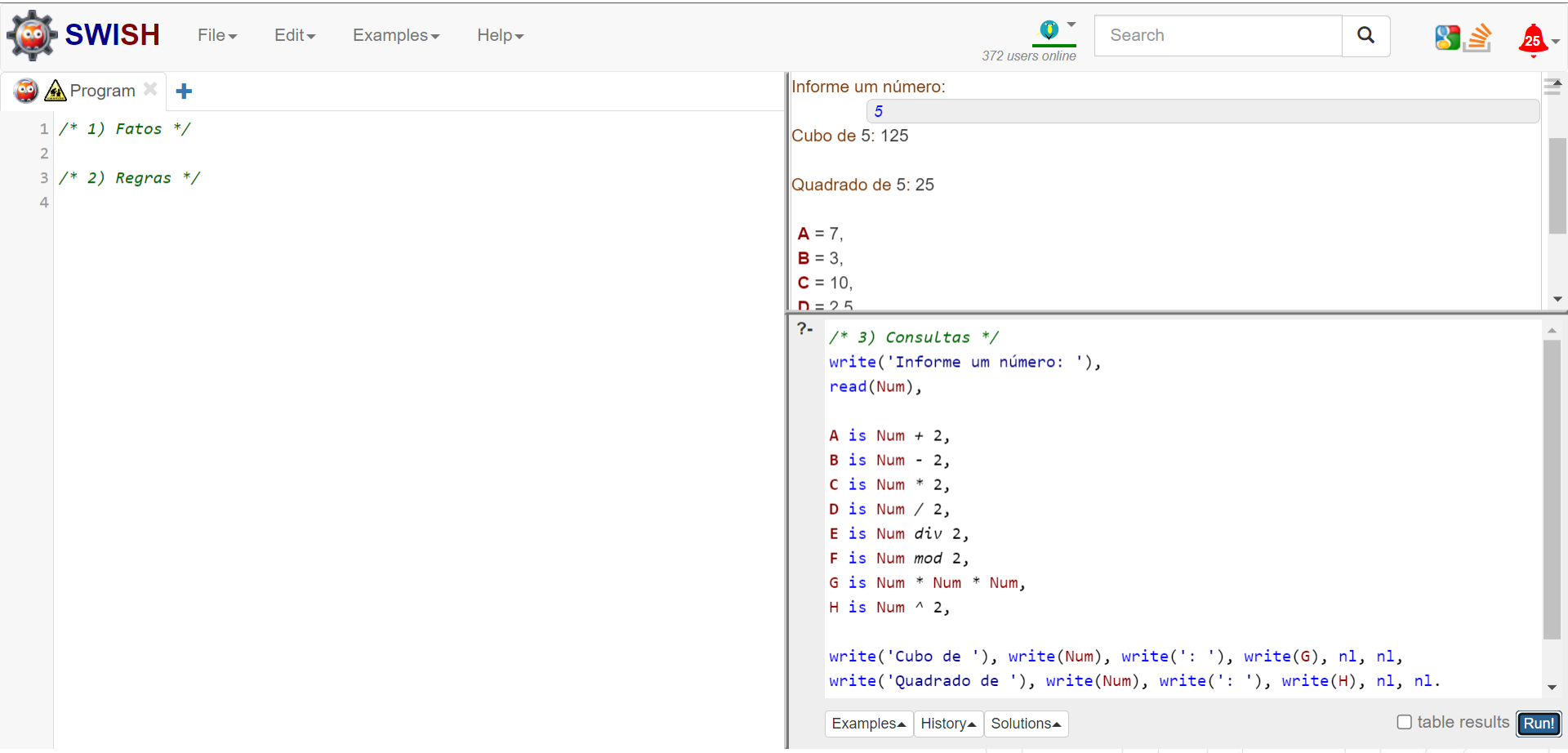
- https://swish.swi-prolog.org/

```
/* 1) Fatos */

/* 2) Regras */
```

```
/* 3) Consultas */
write('Informe um número: '),
read(Num),

A is Num + 2,
B is Num - 2,
C is Num * 2,
D is Num / 2,
E is Num div 2,
F is Num mod 2,
G is Num * Num * Num,
H is Num ^ 2,

write('Cubo de '), write(Num), write(': '), write(G), nl, nl,
write('Quadrado de '), write(Num), write(': '), write(H), nl, nl.
```

SWISH - Prolog

- https://swish.swi-prolog.org/



Exercícios

1) Considerando o código ao lado, realize as seguintes consultas em Prolog:
   a) Listar todas as mães.
   b) Listar todas as filhas.
   c) Listar todos os filhos.
   d) Listar as avós.
   e) Listar os avôs.
   f) Listar os netos.
   g) Listar as netas.
   h) Listar todas as irmãs.
   i) Listar todos os irmãos.
   j) Listar os filhos de Maria.
   k) Listar as filhas de Sílvia.
   l) Listar os filhos de rodrigo.

```
/* 1) Fatos */
progenitor(maria, rodrigo).      /* Predicado binário */
progenitor(emanuel, rodrigo).

progenitor(rodrigo, julia).
progenitor(silvia, julia).

progenitor(rodrigo, jorge).
progenitor(ana, jorge).

masculino(emanuel).  /* Predicado unário */
masculino(rodrigo).
masculino(jorge).

feminino(maria).
feminino(ana).
feminino(julia).
feminino(silvia).


/* 2) Regras */
irmao(X, Y) :- progenitor(Z, X), progenitor(Z, Y), masculino(X).
irma(X, Y) :- progenitor(Z, X), progenitor(Z, Y), feminino(X).

pai(X, Y) :- progenitor(X, Y), masculino(X).
mae(X, Y) :- progenitor(X, Y), feminino(X).

/* 3) Consultas */
?- mae(ana, jorge).
```

Exercícios

2) Considerando o código ao lado, realize a seguintes consultas em Prolog:
   a) Listar todas as classes que herdam da classe Pessoa.
   b) Listar todas as classes que herdam da classe Aluno.
   c) Listar todas as classes que herdam da classe Professor.
   d) Listar todas as classes que contratam a interface Pessoa.
   e) Listar todas as classes que contratam a interface Professor.
   f) Listar todas as classes filhas da classe Pessoa.
   g) Listar todas as classes filhas da classe Professor.
   h) Listar a interface que a classe Aluno implementa

```
interface("ItfPessoa").
interface("ItfAluno").
interface("ItfProfessor").
interface("ItfCoordenador").

classe("ClsPessoa").
classe("ClsAluno").
classe("ClsProfessor").
classe("ClsCoordenador").

implementa("ClsPessoa", "ItfPessoa").
implementa("ClsAluno", "ItfAluno").
implementa("ClsProfessor", "ItfProfessor").
implementa("ClsCoordenador", "ItfCoordenador").

herda("ClsAluno", "ClsPessoa").
herda("ClsProfessor", "ClsPessoa").
herda("ClsCoordenador", "ClsProfessor").

subclasse(X, Y) :- classe(X), herda(X, Y), classe(Y).
contrata(X, Y) :- classe(X), implementa(X, Y), interface(Y).
```

Exercícios

3) Em Prolog, apresente, no mínimo, 3 fatos, 3 regras e 5 consultas para os seguintes domínios:
   a) Aviação.
   b) Comércio.
   c) Medicina.
   d) Meio ambiente.
   e) Tecnologia.