



**UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

Paradigmas de Programación

Laboratorio N°2

Alumno: Israel Arias Panez.
Sección: B-2.
Profesor: Víctor Flores Sánchez.

Santiago - Chile
2-2020

TABLA DE CONTENIDOS

CAPÍTULO 1.	Introducción	5
1.1.1	Descripción del problema.	5
1.1.2	Descripción del paradigma utilizado.	5
CAPÍTULO 2.	Solución del problema.	6
2.1.1	Análisis del problema respecto a sus requisitos específicos.	6
2.1.2	Diseño de la solución.	7
2.1.3	Aspectos de implementación.	10
2.1.4	Instrucciones de uso.	11
2.2	Resultados y autoevaluación.	12
2.3	Conclusión.	13
2.4	Referencias.	13
CAPÍTULO 3.	Anexo.	14

CAPÍTULO 1. INTRODUCCIÓN

En el presente informe se describirá el desarrollo de la solución al segundo laboratorio de la asignatura de Paradigmas de programación. Se efectuará una revisión del problema, se describirá el paradigma utilizado para el desarrollo de la solución además de las herramientas utilizadas, también se hará un contraste de resultados conseguidos con este paradigma en comparación al paradigma funcional, el cual fue usado en la primera entrega de laboratorio. Además, se presentarán instrucciones de uso de la solución desarrollada.

1.1.1 Descripción del problema

En esta segunda entrega de laboratorio se continua con la misma tarea de la entrega anterior, el desarrollo de una simulación de un software de sistema de foros, tomando como referencia el conocido foro web “StackOverflow”, el cual ahora debe ser implementado usando el paradigma lógico y el lenguaje de programación Prolog, para lograrlo, se debe implementar características que tiene un foro, como por ejemplo: registrar usuarios, crear preguntas, responder preguntas, además cada usuario tiene características como username, contraseña, reputación, todas las preguntas que ha hecho, etc. Las preguntas tienen cantidad de votos, respuestas a la pregunta, fecha de publicación, recompensa por la pregunta, entre otros. En la sección 2.1.1 del informe se presentarán los requisitos específicos para esta segunda entrega de laboratorio.

1.1.2 Descripción del paradigma utilizado

Para esta segunda entrega del laboratorio, se solicita hacer uso del paradigma lógico en conjunto del lenguaje de programación Prolog, la principal característica del paradigma lógico es la de los predicados, a diferencia que en el paradigma funcional se usaban las funciones acá existen los predicados, los cuales funcionan mediante relaciones que se definen en una base de conocimientos y se “ejecutan” como consultas en este caso en la consola de Prolog, la cual efectúa consultas en la base de conocimiento.

Ya que se hará uso del paradigma lógico, todas las características a implementar serán hechas como predicados, los cuales se relacionarán con otros predicados para lograr la simulación del foro. Cabe destacar que en la implementación del foro en este paradigma la transparencia referencial también se encuentra presente, al igual que en la entrega anterior, lo cual es una ventaja debido a que no existirán errores de efectos colaterales por modificar un predicado. También se hace mucho uso del backtracking automático de Prolog para las recursiones.

CAPÍTULO 2. SOLUCIÓN DEL PROBLEMA

2.1.1 Análisis del problema respecto a sus requisitos específicos.

Para el desarrollo de la solución se solicitan los siguientes requisitos funcionales específicos:

- Implementación de distintos TDA's para cumplir con todos los siguientes requisitos funcionales, con el TDA stack como base del programa.
- Base de hechos: Implementada a partir de los TDA's, la base debe contener al menos 2 stacks, 4 usuarios registrados, 5 preguntas y 10 respuestas.
- Predicado register: permite registrar a un usuario en el stack.
- Predicado login: permite a un usuario ya registrado iniciar sesión, lo que le permitirá hacer ciertas operaciones.
- Predicado ask: permite a un usuario con sesión iniciada hacer una pregunta, la cual se agrega al stack.
- Predicado answer: permite a un usuario con sesión iniciada responder a una pregunta, la respuesta se agrega al stack.
- Predicado accept: permite a un usuario con sesión iniciada aceptar una respuesta a una de sus preguntas, modificara la reputación del usuario cuya respuesta fue aceptada.
- Predicado stackToString: transforma todo el stack a un string posible de visualizar de forma comprensible por el usuario, el string puede ser mostrado en pantalla con el predicado write, en caso de existir un usuario con sesión iniciada, mostrará solo las preguntas e información de ese usuario.

También se solicita implementar un predicado complementario, para este laboratorio se eligió implementar la función vote:

- Predicado vote: permite a un usuario con sesión iniciada, votar positiva o negativamente una pregunta o respuesta que se encuentre en el stack, al hacerlo se registrara el voto y se sumara o restara reputación a los usuarios según corresponda.

Tomando en consideración los requisitos funcionales quedan claras las funcionalidades que deben implementarse en esta entrega de laboratorio, además es posible darse cuenta que es conveniente ir realizando la implementación de los predicados bajo el mismo orden que se ha presentado arriba, debido a que por ejemplo los predicados ask, answer, accept y stackToString(cuando hay un usuario con sesión iniciada) necesitan del predicado login para funcionar, por lo cual para esta entrega, se implementaran los requerimientos siguiendo el orden entregado.

2.1.2 Diseño de la solución.

Gracias a la estructuración del laboratorio mediante los requisitos funcionales descritos en el punto 2.1.1, la solución se implementó bajo una división de problemas en subproblemas, siguiendo los mismos subproblemas de los requisitos funcionales: de implementar primero Tda's, luego una base de hechos, luego register y así sucesivamente, para luego combinar todas esas implementaciones y formar la simulación del foro, lo que también es conocido como divide y vencerás, “El esquema de dividir y vencer(también conocido como divide y vencerás)es una técnica para resolver problemas que consiste en dividir el problema original en subproblemas (de menor tamaño), resolver los subproblemas y finalmente combinar las soluciones de los subproblemas para dar una solución al problema original”.^[1]

- Implementación de Tda's:

Para la creación del foro se definieron cuatro TDA's: TDA Stack, TDA Pregunta, TDA Respuesta y TDA Usuarios. Cada TDA con sus respectivos predicados constructores, selectores, de pertenencia y modificadores según corresponda. El TDA Stack es la base de los otros TDA's ya que funciona como un contenedor de TDA's dentro de listas, siguiendo la estructura basada en listas:

[[[TDA's Pregunta]],[[TDA's Respuesta]],[[TDA's Usuarios]],[[UsuarioConectado]]].

- Base de hechos:

La base de hechos consiste en dos stacks con variedad de datos, las bases fueron definidas a través de los TDA's implementados, se pueden ver en el archivo `stackoverflowProlog_20110122_AriasPanez.pl` en la sección de base de hechos.

- Predicado register:

Desde la implementación de este predicado en adelante fue fundamental el uso de recursión, y el separar las listas como [cabeza|cola] a fin de poder ir “recorriendo” los datos, Prolog facilita esta tarea gracias a su backtracking automático, el cual busca la variable para la cual cierto predicado es verdadero. En este predicado fue necesario antes de registrar al usuario en el stack el recorrer toda la lista de usuarios para determinar que el mismo usuario no se intentará registrar dos veces.

- Predicado login:

En login al igual que en register, debió hacer uso de la recursión, se debía recorrer el stack de usuarios registrados y encontrar al usuario que se quiere loguear comparando su Username, una vez encontrado se comparaba la contraseña ingresada, en caso de ser correcta, permite el ingreso, agregándolo al final del stack en UsuarioConectado.

- Predicado ask:
En este caso se usaba el constructor de preguntas, se registraba al autor como el usuario conectado, se generaba un id de pregunta y luego se hacía una recursión la cual recorría todo el stack de usuarios, encontraba al usuario que realizó la pregunta y le agregaba el id de la pregunta recién hecha a su historial de preguntas.
- Predicado answer:
El predicado answer funciona de la misma manera que ask, con la diferencia que en la respuesta se registra el id de la pregunta que se responde y un id de respuesta, y no se registra la respuesta en un historial de respuestas del usuario.
- Predicado accept:
Para accept se fue haciendo recursiones sucesivas combinadas con los predicados de los distintos TDA's definidos, primero se verifica que la pregunta en la cual se quiere aceptar una respuesta sea una pregunta hecha por el mismo usuario logueado en ese momento en el stack, luego se buscaba la pregunta en el stack de preguntas y se modificaba su estado a "cerrada", luego se buscaba la respuesta de la pregunta en el stack de respuestas y se cambiaba su estado a "aceptada" a su vez se registraba el nombre de usuario de la persona cuya respuesta fue aceptada, finalmente se recorría la lista de usuarios, buscando al usuario cuya respuesta fue aceptada, una vez encontrado se le agregaba la reputación.
- Predicado StackToString:
Para poder transformar el stack en una string que luego al usarla en conjunto con el predicado write devolviera un stack mucho más comprensible de ver, fue necesario usar los predicados: atom_string, atom_list_concat y string_concat. Atom string transforma un átomo a string, atom_list_concat concatena una lista de atomos y los retorna como un solo átomo, y por ultimo string_concat concatena strings, o sea toma dos strings y devuelve una sola string siendo esta la unión de las dos strings ingresadas. Esos predicados se usaron en conjunto de recursión, para poder ir recorriendo todo el stack y formando una string que luego pueda ser vista de una manera mucho más comprensible.
- Predicado vote:
El predicado vote funciona de una manera muy parecida a como funciona accept, con la diferencia que hace uso de dos predicados complementarios getQuestion y getAnswer, los cuales retornan un Tda pregunta o respuesta a la cual luego debe aplicársele el voto positivo(true) o negativo(false), por lo cual para reconocer si lo que se busca es una pregunta o una respuesta, simplemente se aplican los predicados de pertenencia de los

TDA pregunta y TDA respuesta, luego se procede a aplicar el voto siguiendo un procedimiento similar al predicado accept, luego se actualiza el puntaje según corresponda.

Para lograr todo el desarrollo del foro, fue fundamental los predicados definidos en un inicio en los TDA's junto al comprender el funcionamiento de la recursión en prolog y los procesos de unificación de variables para “simular” retornos de datos.

2.1.3 Aspectos de implementación.

La implementación de este programa se hizo en el lenguaje Prolog, el foro implementado se encuentra en una base de conocimientos y las consultas se realizan en el ambiente SWI-Prolog V8.2.2, no hace uso de librerías, solo predicados nativos del lenguaje de programación Prolog.

Al abrir el archivo `stackoverflowProlog_20110122_AriasPanez.pl` Se puede apreciar que el programa se encuentra estructurado de la siguiente manera:

- 1) TDA Stack
- 2) TDA Pregunta
- 3) TDA Respuesta
- 4) TDA Usuarios
- 5) Base de Hechos
- 6) Predicados complementarios
- 7) Predicados requeridos: En el mismo orden presentado en la sección 2.1.1.

2.1.4 Instrucciones de uso.

A continuación, se explicará las instrucciones para poder usar el programa de una manera adecuada:

- En primer lugar, abrir Swi-prolog y asegurarse de seleccionar el archivo de consulta “stackOverflowProlog_20110122_AriasPanez.pl”.
-
- Al final del archivo stackOverflowProlog_20110122_AriasPanez.pl se encuentran diversos ejemplos como comentarios, son distintas consultas que pueden ser directamente ingresadas en la consola de Prolog, para probar cada funcionalidad del foro implementado.
- Un ejemplo para probar los predicados, por ejemplo, la consulta:
crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2).
Puede ser copiada y pegada directamente en la consola de prolog, lo que mostrará el resultado de registrar al usuario “israel” con contraseña “qwerty” al foro.
- Se recomienda ver los anexos al informe para ver más ejemplos de ejecución.

2.2 Resultados y autoevaluación.

En esta sección se hará una revisión de los resultados, para ello se verificarán los requisitos no funcionales y funcionales solicitados

Requerimientos no funcionales:

La implementación debe ser en el lenguaje de programación Prolog.	Logrado: el lenguaje usado en el programa es Prolog.
Versión: Usar Swi-prolog versión 8.x.x .	Logrado: la versión de Swi-prolog usada es la 8.2.2
Documentación: Todos los predicados deben estar debidamente comentados. Indicando descripción de la relación, términos de entrada y de salida.	Logrado: Todas las funciones implementadas se encuentran comentadas, se indica descripción y dominio de cada una de ellas
Historial: al menos 10 commits en un periodo mayor o igual a una semana.	Logrado: primer commit hecho el 22 de noviembre de 2020, último commit hecho el 06 de diciembre de 2020, con un total de 14 commits.
Ejemplos:	Logrado, hay un mínimo de 3 ejemplos por cada función en el programa stackoverflowProlog_20110122_AriasPanez.pl” los ejemplos se encuentran al final de la base de conocimientos.
Prerrequisitos:	Logrado, cada predicado tiene su prerrequisito implementado como se solicitó.

Requerimientos Funcionales:

Todos los requerimientos funcionales funcionan sin problemas, cumpliendo con las restricciones solicitadas para cada una de ellas, fueron probadas ingresándoles distintas entradas a cada función y verificando que la salida fuera la esperada de acuerdo con la entrada, se adjuntan ejemplos dentro del archivo
stackoverflowProlog_20110122_AriasPanez.pl

2.3 Conclusión.

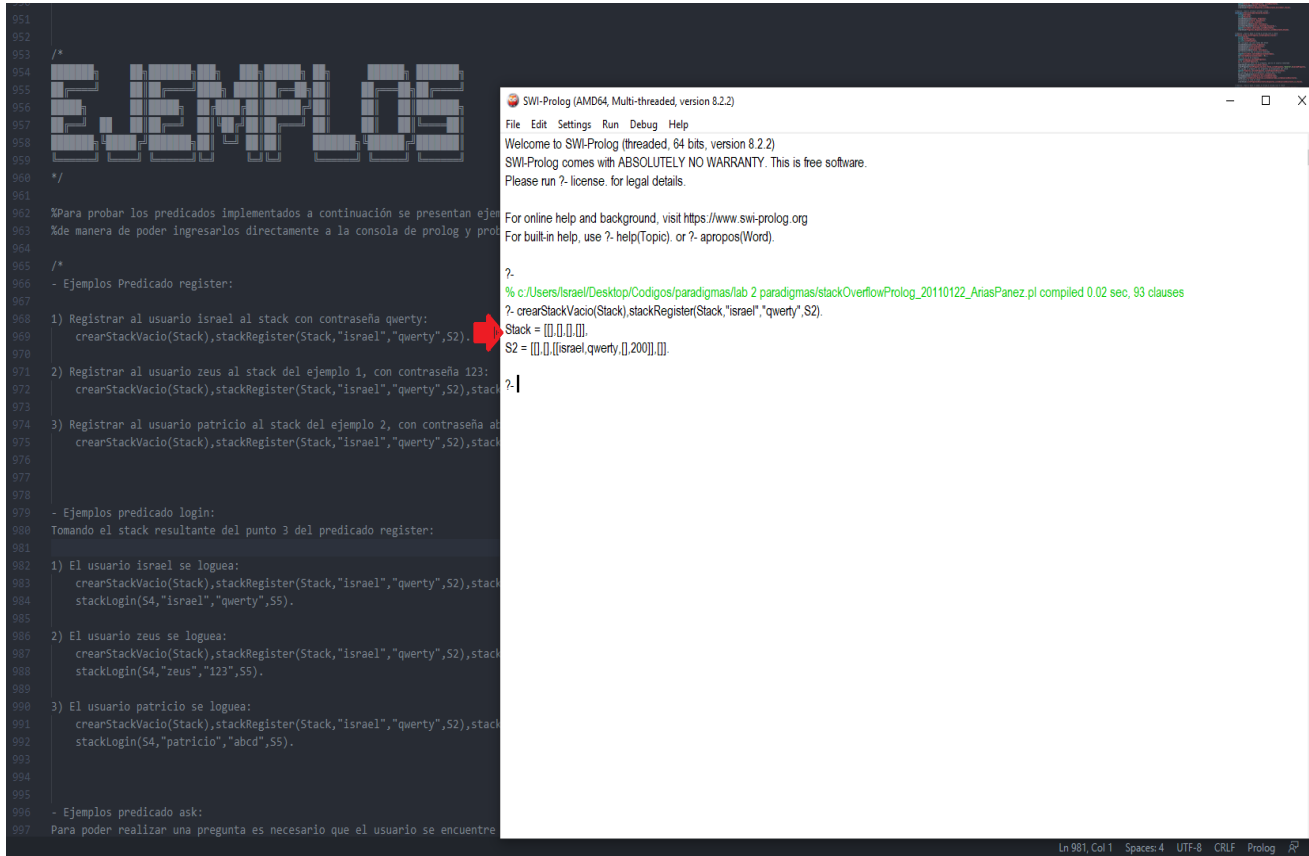
Tomando en consideración los resultados listados en la sección 2.2 de este informe, es posible concluir que se ha logrado cumplir con todos los objetivos requeridos para esta segunda entrega de laboratorio, tanto funcionales como no funcionales.

En un principio fue todo un desafío el entender cómo Prolog buscaba unificar variables a través de los hechos definidos, buscando las relaciones establecidas. Una vez entendida esa lógica junto a la recursión y backtracking en Prolog, la implementación de la simulación del foro “StackOverflow” se hizo mucho más fácil incluso que en la entrega anterior hecha en el paradigma funcional.

2.4 Referencias

- [1] Duch A. (2006). *Esquema de Dividir y Vencer*.
(Recuperado 05/12/2020).
<https://www.cs.upc.edu/~ Duch/home/duch/dyd.pdf>

CAPÍTULO 3. ANEXO



The screenshot shows a Prolog IDE with a dark theme. On the left, a Prolog program is loaded from a file. The program includes comments in Spanish and Prolog code for creating a stack, registering users, and logging them in. On the right, the SWI-Prolog console shows the execution of the program. A red arrow points to the line where the stack is created and registered. The console output shows the stack's internal representation.

```
351
352
353 /*
354
355
356
357
358
359
360 */
361
362 %Para probar los predicados implementados a continuación se presentan ejem
363 %de manera de poder ingresarlos directamente a la consola de prolog y prob
364
365 /*
366 - Ejemplos Predicado register:
367
368 1) Registrar al usuario israel al stack con contraseña qwerty:
369   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2).
370
371 2) Registrar al usuario zeus al stack del ejemplo 1, con contraseña 123:
372   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stack
373
374 3) Registrar al usuario patricio al stack del ejemplo 2, con contraseña ab
375   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stack
376
377
378 - Ejemplos predicado login:
379 Tomando el stack resultante del punto 3 del predicado register:
380
381
382 1) El usuario israel se loguea:
383   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stack
384   stackLogin(S4,"israel","qwerty",S5).
385
386 2) El usuario zeus se loguea:
387   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stack
388   stackLogin(S4,"zeus","123",S5).
389
390 3) El usuario patricio se loguea:
391   crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stack
392   stackLogin(S4,"patricio","abcd",S5).
393
394
395 - Ejemplos predicado ask:
396 Para poder realizar una pregunta es necesario que el usuario se encuentre
```

```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.2)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.2)
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/israel/Desktop/Codigos/paradigmas/lab 2 paradigmas/stackOverflowProlog_20110122_AniasPanez.pl compiled 0.02 sec, 93 clauses
?- crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2).
Stack = [],[],[],[],[]
S2 = [],[],[[israel,qwerty,[],200]],[]
?-
```

Anexo 1: Ejemplo de prueba de predicado register, logrado copiando y pegando la consulta en la consola de prolog, previamente habiendo seleccionado el archivo de consulta.

```
-Ejemplos predicado stackToString:
Se retomara el stack como se dejo en el punto 3 de accept:

1) Para crear una string que pueda ser visualizada luego de una manera mucho mas organizada del stack que se lleva hasta ahora:
crearStackVacio(Stack),stackRegister(Stack,"israel","qwerty",S2),stackRegister(S2,"zeus","123",S3),stackRegister(S3,"patricio","abcd",S4),
stackLogin(S4,"israel","qwerty",S5),ask(S5,"03/12/2020","¿Como puedo hacer hola mundo en prolog?","[prolog]",S6),
stackLogin(S6,"zeus","123",S7),ask(S7,"04/12/2020","¿Como defino una variable en C?","[C", "variables",S8),
stackLogin(S8,"patricio","abcd",S9),ask(S9,"05/12/2020","¿como puedo importar una libreria en C?","[C","librerias",S10),
stackLogin(S10,"israel","qwerty",S11),answer(S11,"06/12/2020",2,"Puedes hacer int variable = 1, es importante definir el tipo de dato","[prolog","hola mundo",S12),
stackLogin(S12,"zeus","123",S13),answer(S13,"06/12/2020",3,"debes usar #include <nombre libreria> al principio de tu codigo","[C","librerias",S14),
stackLogin(S14,"patricio","abcd",S15),answer(S15,"06/12/2020",1,"en la consola de prolog has la consulta write(hola mundo)","[C","librerias",S16),
stackLogin(S16,"zeus","123",S17),answer(S17,"06/12/2020",1,"tambien puedes usar print o display en vez de write","[prolog","print",S18),
stackLogin(S18,"israel","qwerty",S19),accept(S19,1,4,S20),
stackLogin(S20,"zeus","123",S21),accept(S21,2,1,S22),
stackLogin(S22,"patricio","abcd",S23),accept(S23,3,2,S24),
stackToString(S24,StackStr).
```

Anexo 2: Ejemplo de Consulta de predicado stackToString.

StackStr =

Pregunta: Id: 1 Fecha Publicacion: 03/12/2020 Votos: 0 Estado: cerrada

¿Como puedo hacer hola mundo en prolog?

Etiquetas: prolog Autor: israel

Respuesta: Id: 3 Id al que responde: 1 Fecha de respuesta: 06/12/2020 Votos: 0 Estado: pendiente
en la consola de prolog has la consulta write(hola mundo)

Etiquetas: C, librerias Autor: patricio

Respuesta: Id: 4 Id al que responde: 1 Fecha de respuesta: 06/12/2020 Votos: 0 Estado: aceptada
tambien puedes usar print o display en vez de write

Etiquetas: prolog, print Autor: zeus

Pregunta: Id: 2 Fecha Publicacion: 04/12/2020 Votos: 0 Estado: cerrada

¿Como defino una variable en C?

Etiquetas: C, variables Autor: zeus

Respuesta: Id: 1 Id al que responde: 2 Fecha de respuesta: 06/12/2020 Votos: 0 Estado: aceptada
Puedes hacer int variable = 1, es importante definir el tipo de dato

Etiquetas: prolog, hola mundo Autor: israel

Pregunta: Id: 3 Fecha Publicacion: 05/12/2020 Votos: 0 Estado: cerrada

¿como puedo importar una libreria en C?

Etiquetas: C, Librerias Autor: patricio

Respuesta: Id: 2 Id al que responde: 3 Fecha de respuesta: 06/12/2020 Votos: 0 Estado: aceptada
debes usar #include <nombre libreria> al principio de tu codigo

Etiquetas: C, librerias Autor: zeus

Informacion de usuarios registrados en el foro:

Usuario:

Username: israel Reputacion: 217 Preguntas Realizadas(Id's): 1

Usuario:

Username: zeus Reputacion: 232 Preguntas Realizadas(Id's): 2

Usuario:

Username: patricio Reputacion: 202 Preguntas Realizadas(Id's): 3

Anexo 3: Resultado de aplicar write en la consulta realizada en el anexo 2.