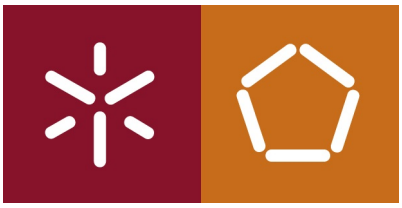

Universidade do Minho

Escola de Engenharia

Mestrado Integrado de Engenharia Informática

Sistemas de Representação Conhecimento e Raciocínio

Exercício 1 - Programação em lógica e Invariantes

Trabalho realizado por:

Adriana Guedes	A74545
Guilherme Guerreiro	A73860
Marco Barbosa	A75278
Ricardo Certo	A75315

19 março de 2017

Conteúdo

1 Introdução

A programação em lógica é um tipo específico de programação cujo objetivo é a implementação de um programa cujo conteúdo se prende em factos (registos que se sabem ser verdadeiros), predicados, associados aos factos e regras.

A programação em lógica baseia-se em dois princípios básicos para a descoberta das respostas a essas questões: **lógica**, usada para representar os conhecimentos e informação, e **inferência**, regras aplicadas à lógica para manipular o conhecimento.

O trabalho prático presente consiste no desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da prestação de cuidados de saúde pela realização de serviços de atos médicos. Para tal iremos utilizar a linguagem de programação em lógica **PROLOG**.

PROLOG é uma linguagem de programação que se enquadra no paradigma de Programação em Lógica. É uma linguagem de uso geral que é especialmente associada com a inteligência artificial e linguística computacional, pois trata-se de uma linguagem mais direcionada ao conhecimento do que aos algoritmos.

1.1 Motivação e Objetivos

Depois de adquiridos os conhecimentos da linguagem, este exercício tem como motivação principal consolidar os conhecimentos lecionados nas aulas, que são muito úteis para a resolução de problemas de programação em lógica.

O objetivo deste exercício é a elaboração de um programa capaz de armazenar conhecimento sobre um registo de atos de saúde de uma instituição realizados por um utente e através deste solucionar os problemas relativos a este tema.

1.2 Estrutura do Relatório

Este relatório encontra-se organizado em seis capítulos. Sendo que neste primeiro introduzimos a linguagem e o tema em que se baseia este exercício.

No segundo capítulo é elaborado um estudo prévio da linguagem de modo a que o leitor consiga entender o que se pretende com a realização deste exercício.

No terceiro capítulo explicamos o que foi desenvolvido para a implementação do exercício fazendo uma breve referência à base de conhecimento inicial e aos invariantes.

No quarto capítulo explicamos tudo o que foi desenvolvido para a implementação do exercício.

No quinto capítulo apresentamos as funcionalidades extras implementadas para tornarem o conhecimento o mais completo possível.

Por fim apresentamos uma conclusão com as reflexões do grupo depois de realizar o exercício.

2 Programação em Lógica e PROLOG

Uma linguagem de programação lógica utiliza a lógica para representar conhecimento e inferências para manipular informações. Como tal, um programa deste tipo de programação possui os seguintes parâmetros:

- **Factos**- algo que se conhece como e se sabe ser verdadeiro (ex: cor(preto))
- **Predicados**- implementações de relações, por exemplo, pai(pai,filho) é a implementação da relação de descendência (ser pai de).
- **Regras**- As regras PROLOG são descrições de predicados por meio de condicionais.

3 Descrição do trabalho e Análise de Resultados

3.1 Base de Conhecimento

A base de conhecimento define as bases de dados ou de conhecimento acumulado sobre determinado assunto.

Para a elaboração deste exercício foi importante definir uma base de conhecimento que responda aos pedidos do enunciado.

3.1.1 Utentes

Para a resolução do exercício foi necessário criar uma base de conhecimento dos utentes existentes. Como tal, incluímos na base de conhecimento os utentes com a assinatura **utente(IdUt, Nome, Idade, Rua , Cidade, Contacto)**. Decidimos acescentar os predicados rua, cidade e contacto de modo a que o conhecimento que vamos possuir na nossa base de conhecimento seja o mais completo possível. Apresentamos de seguida um excerto da base de conhecimento:

```
utente( 1,'Carlos',35,'Rua D.Pedro V','Braga','253456789').
utente( 2,'Joao',12,'Rua da Ramada','Guimaraes','929876543').
utente( 3,'Julio',89,'Rua das Victorias','Guimaraes','935436789').
utente( 4,'Ana',25,'Rua Conde Almoster','Lisboa','913456789').
utente( 5,'Carolina',50,'Rua do Caires','Braga','253987654').
utente( 6,'Joana',23,'Av.da Boavista','Porto','961234567').
utente( 7,'Fernando',65,'Rua do Loureiro','Viana do Castelo','966668569').
utente( 8,'Rute',18,'Av. da Liberdade','Braga','916386423').
utente( 9,'Maciel',45,'Rua das Flores','Porto','935731290').
utente( 10,'Filipa',32,'Rua Padre Vitorino','Faro','289347681').
utente( 11,'Mauro',76,'Rua Gil Vicente','Montalegre','276327904').
utente( 12,'Laura',90,'Rua Fernando Mendes','Leiria','244000045').
utente( 13,'Jaime',48,'Av. Norton Matos','Barcelos','914768180').
utente( 14,'Lourenço',26,'Rua da Boavista','Guimaraes','926306127').
utente( 15,'Tiago',16,'Rua Monsenhor de Melo','Vilamoura','936150873').
```

3.1.2 Cuidados prestados

Um requisito a incluir na base de conhecimento são os cuidados prestados. Assim, para representar os mesmos temos o predicado cuidado_prestado com a assinatura **cuidado_prestado(IdServ, Descrição, Instituição, Cidade)**. Apresentamos de seguida um excerto da base de conhecimento:

```
cuidado_prestado( 1,'Pediatria','Hospital Privado de Braga','Braga').
cuidado_prestado( 2,'Cardiologia','Hospital de Braga','Braga').
cuidado_prestado( 3,'Ortopedia','Hospital de Braga','Braga').
cuidado_prestado( 4,'Oftalmologia','Hospital de Braga','Braga').
cuidado_prestado( 5,'Oncologia','IPO','Porto').
cuidado_prestado( 6,'Urgência','Hospital de Santa Maria','Porto').
cuidado_prestado( 7,'Urgência','Hospital da Luz','Guimaraes').
cuidado_prestado( 8,'Neurologia','Centro Hospitalar Sao Joao','Porto').
cuidado_prestado( 9,'Urgência','Hospital de Braga','Braga').
cuidado_prestado( 10,'Urgência','Hospital Lusiadas','Faro').
cuidado_prestado( 11,'Otorrinolaringologia','Hospital de Vila Real','Vila Real').
```

3.1.3 Atos médicos

Por fim, incluímos os atos médicos na base de conhecimento com a assinatura **ato_medico(Data, IdUt, IdServ, CorPulseira, Médico, Custo)**. Adicionamos a cor da pulseira e o nome do médico ao predicado do ato médico de forma a que o conhecimento presente na base de conhecimento seja o mais completo possível. Apresentamos de seguida um excerto da base de conhecimento:

```
atos( '02-01-17', 15, 10, 'Verde', 'Dra.Sara', 17.5).
atos( '24-02-17', 8, 4, 'Sem_pulseira', 'Dr.Mourao', 4).
atos( '25-02-17', 1, 2, 'Sem_pulseira', 'Dr.Barroso', 12).
atos( '28-01-17', 13, 9, 'Laranja', 'Dr.Tomas', 5).
atos( '10-02-17', 4, 3, 'Sem_pulseira', 'Dr.Falcão', 20).
atos( '19-03-17', 3, 9, 'Amarela', 'Dr.Bones', 50).
atos( '11-01-17', 1, 1, 'Sem_pulseira', 'Dr.Pardal', 2).
atos( '12-02-17', 5, 8, 'Sem_pulseira', 'Dra.Teresa', 13.75).
atos( '20-03-17', 11, 11, 'Sem_pulseira', 'Dra.Marta', 13).
atos( '27-01-17', 2, 7, 'Amarela', 'Dr.Pedro Martins', 11).
atos( '01-01-17', 6, 6, 'Laranja', 'Dr.Reveillon', 16).
atos( '03-03-17', 3, 1, 'Sem_pulseira', 'Dra.Candida', 45).
atos( '08-03-17', 9, 9, 'Vermelha', 'Dr.Lima', 50).
atos( '14-02-17', 14, 7, 'Amarela', 'Dra.Mafalda', 23).
atos( '30-01-17', 7, 5, 'Sem_pulseira', 'Dr.Quimio', 10).
atos( '01-02-17', 1, 6, 'Verde', 'Dra.Luisa', 25.5).
atos( '01-03-17', 10, 6, 'Verde', 'Dra.Luisa', 25.5).
atos( '10-03-17', 12, 11, ' Sem_pulseira', 'Dr.Luis', 12.50).
```

4 Funcionalidades do programa

Depois de apresentada a base de conhecimento do exercício iremos agora apresentar e explicar a forma como resolvemos cada uma das funcionalidades e quais os resultados obtidos para as soluções que iremos apresentar.

4.1 Registar utentes, cuidados prestados e atos médicos

Para que fosse possível a inserção de conhecimento na base desenvolvemos o teorema **evolucao**, teorema este que verifica os invariantes que nós definimos, pois só assim podemos garantir que não é adicionada à base informação repetida, e caso não seja informação repetida adiciona na base. Apresentamos assim de seguida um excerto da nossa implementação das inserções na base de conhecimento:

```
registaUtentes(ID,NM,I,RU,CDD,CNT) :- evolucao(utente(ID,NM,I,RU,CDD,CNT)).

registaCuidados(ID,D,I,C) :- evolucao(cuidado_prestado(ID,D,I,C)).

registaAtos(D,IDUT,IDS,CP,MDC,C) :- evolucao(atos(D,IDUT,IDS,CP,MDC,C)).
```

Exemplo output

```
| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').
utente(15, 'Tiago', 16, 'Rua Monsenhor de Melo', 'Vilamoura', '936150873').

yes
| ?- registaUtentes(16,'Joao',12,'Rua D.Pedro V', 'Braga' ,'122435867').
yes
| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').
utente(15, 'Tiago', 16, 'Rua Monsenhor de Melo', 'Vilamoura', '936150873').
utente(16, 'Joao', 12, 'Rua D.Pedro V', 'Braga', '122435867').
```

```
yes


| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').
utente(15, 'Tiago', 16, 'Rua Monsenhor de Melo', 'Vilamoura', '936150873').
utente(16, 'Joao', 12, 'Rua D.Pedro V', 'Braga', '122435867').

yes



| ?- listing(cuidado_prestado).
cuidado_prestado(1, 'Pediatria', 'Hospital Privado de Braga', 'Braga').
cuidado_prestado(2, 'Cardiologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(3, 'Ortopedia', 'Hospital de Braga', 'Braga').
cuidado_prestado(4, 'Oftalmologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(5, 'Oncologia', 'IPO', 'Porto').
cuidado_prestado(6, 'Urgência', 'Hospital de Santa Maria', 'Porto').
cuidado_prestado(7, 'Urgência', 'Hospital da Luz', 'Guimaraes').
cuidado_prestado(8, 'Neurologia', 'Centro Hospitalar Sao Joao', 'Porto').
cuidado_prestado(9, 'Urgência', 'Hospital de Braga', 'Braga').
cuidado_prestado(10, 'Urgência', 'Hospital Lusiadas', 'Faro').

yes
| ?- registaCuidados(13, 'Pediatria', 'Hospital Privado de Braga', 'Braga').
yes
| ?- listing(cuidado_prestado).
cuidado_prestado(1, 'Pediatria', 'Hospital Privado de Braga', 'Braga').
cuidado_prestado(2, 'Cardiologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(3, 'Ortopedia', 'Hospital de Braga', 'Braga').
cuidado_prestado(4, 'Oftalmologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(5, 'Oncologia', 'IPO', 'Porto').
cuidado_prestado(6, 'Urgência', 'Hospital de Santa Maria', 'Porto').
cuidado_prestado(7, 'Urgência', 'Hospital da Luz', 'Guimaraes').
cuidado_prestado(8, 'Neurologia', 'Centro Hospitalar Sao Joao', 'Porto').
cuidado_prestado(9, 'Urgência', 'Hospital de Braga', 'Braga').
cuidado_prestado(10, 'Urgência', 'Hospital Lusiadas', 'Faro').
```

```
cuidado_prestado(13, 'Pediatria', 'Hospital Privado de Braga', 'Braga').

yes



| ?- listing(atos).
atos('02-01-17', 15, 10, 'Verde', 'Dra.Sara', 17.5).
atos('24-02-17', 8, 4, 'Sem_pulseira', 'Dr.Mourao', 4).
atos('25-02-17', 1, 2, 'Sem_pulseira', 'Dr.Barroso', 12).
atos('28-01-17', 13, 9, 'Laranja', 'Dr.Tomas', 5).
atos('10-02-17', 4, 3, 'Sem_pulseira', 'Dr.FalcÃ£o', 20).
atos('19-03-17', 3, 9, 'Amarela', 'Dr.Bones', 50).
atos('11-01-17', 1, 1, 'Sem_pulseira', 'Dr.Pardal', 2).
atos('12-02-17', 5, 8, 'Sem_pulseira', 'Dra.Teresa', 13.75).
atos('20-03-17', 11, 11, 'Sem_pulseira', 'Dra.Marta', 13).
atos('27-01-17', 2, 7, 'Amarela', 'Dr.Pedro Martins', 11).
atos('01-01-17', 6, 6, 'Laranja', 'Dr.Reveillon', 16).
atos('03-03-17', 3, 1, 'Sem_pulseira', 'Dra.Candida', 45).
atos('08-03-17', 9, 9, 'Vermelha', 'Dr.Lima', 50).
atos('14-02-17', 14, 7, 'Amarela', 'Dra.Mafalda', 23).
atos('30-01-17', 7, 5, 'Sem_pulseira', 'Dr.Quimio', 10).
atos('01-02-17', 1, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('01-03-17', 10, 6, 'Verde', 'Dra.Luisa', 25.5).

yes
| ?- registaAtos('01-03-17', 4, 6, 'Verde', 'Dra.Mafalda', 20).
yes
| ?- listing(atos).
atos('02-01-17', 15, 10, 'Verde', 'Dra.Sara', 17.5).
atos('24-02-17', 8, 4, 'Sem_pulseira', 'Dr.Mourao', 4).
atos('25-02-17', 1, 2, 'Sem_pulseira', 'Dr.Barroso', 12).
atos('28-01-17', 13, 9, 'Laranja', 'Dr.Tomas', 5).
atos('10-02-17', 4, 3, 'Sem_pulseira', 'Dr.FalcÃ£o', 20).
atos('19-03-17', 3, 9, 'Amarela', 'Dr.Bones', 50).
atos('11-01-17', 1, 1, 'Sem_pulseira', 'Dr.Pardal', 2).
atos('12-02-17', 5, 8, 'Sem_pulseira', 'Dra.Teresa', 13.75).
atos('20-03-17', 11, 11, 'Sem_pulseira', 'Dra.Marta', 13).
atos('27-01-17', 2, 7, 'Amarela', 'Dr.Pedro Martins', 11).
atos('01-01-17', 6, 6, 'Laranja', 'Dr.Reveillon', 16).
atos('03-03-17', 3, 1, 'Sem_pulseira', 'Dra.Candida', 45).
atos('08-03-17', 9, 9, 'Vermelha', 'Dr.Lima', 50).
atos('14-02-17', 14, 7, 'Amarela', 'Dra.Mafalda', 23).
atos('30-01-17', 7, 5, 'Sem_pulseira', 'Dr.Quimio', 10).
atos('01-02-17', 1, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('01-03-17', 10, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('01-03-17', 4, 6, 'Verde', 'Dra.Mafalda', 20).

yes
```

Tal como referido anteriormente, foi necessário criarmos invariantes para garantir que não era adicionada à base informação repetida.

```
+utente(I,Nome,IDD,RU,CDD,CNT) :: solucoes(I,(utente(I,_,_,_,_,_)),L),
                                   comprimento(L,N),
                                   N == 1.
```

O invariante acima apresentado foi criado de forma a que não fosse possível introduzir um utente com o mesmo id de um que já exista na nossa base. Assim este invariante irá ser útil, pois após a inserção de um utente este irá procurar esse mesmo utente na base e colocar as soluções numa lista, lista esta que posteriormente será calculado o comprimento e que este terá que ser igual a 1, pois tal significa que não foram adicionados utentes repetidos.

Os invariantes para os atos médicos e os cuidados prestados seguem a mesma linha de pensamento.

```
+cuidado_prestado(ID,D,I,X) ::(solucoes(ID,(cuidado_prestado(ID,_,_,_)),L),
                                 comprimento(L,N),
                                  N == 1).

+atos(D,IDUT,IDS,CP,MDC,C) ::  (solucoes((D,IDUT,IDS),(atos(D,IDUT,IDS,_,_,_)),L),
                               comprimento(L,N),
                               N == 1).
```

O invariante seguinte é responsável por fazer a verificação da existência nos utentes e nos cuidados prestados de ids válidos, para permitir o registo na base de conhecimento de um ato.

```
+atos(D,IDUT,IDS,CP,MDC,C) :: (utente(IDUT,_,_,_,_,_),
                               cuidado_prestado(IDS,_,_,_)).
```

4.2 Identificar os utentes por critério de seleção

Para identificar os utentes segundo um critério selecionado, isto é, segundo o ID, o nome, a idade, a morada, a rua , a cidade ou o contacto, iremos à base e com o predicado **solucoes**, predicado este que utiliza o *findall*, iremos encontrar todas as soluções para o critério pretendido.

Apresentamos de seguida os predicados **utenteID, utenteNome, utenteIdade , utenteRua , utenteCidade** e **utenteContacto**, soluções por nós desenvolvidas para responder à questão apresentada.

```
utenteID(ID,R) :- solucoes((ID,X,Y,Z,W,K),utente(ID,X,Y,Z,W,K),R).

utenteNome(NM,R) :- solucoes((X,NM,Y,Z,W,K),utente(X,NM,Y,Z,W,K),R).

utenteIdade(I,R) :- solucoes((X,Y,I,Z,W,K),utente(X,Y,I,Z,W,K),R).

utenteRua(RU,R) :- solucoes((X,Y,Z,RU,W,K),utente(X,Y,Z,RU,W,K),R).

utenteCidade(CDD,R) :- solucoes((X,Y,Z,W,CDD,K),utente(X,Y,Z,W,CDD,K),R).

utenteContacto(CNT,R) :- solucoes((X,Y,Z,W,K,CNT),utente(X,Y,Z,W,K,CNT),R).
```

Output

```
| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
```

```
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').
utente(15, 'Tiago', 16, 'Rua Monsenhor de Melo', 'Vilamoura', '936150873').

yes
| ?- utenteID(4,R).
R = [(4,'Ana',25,'Rua Conde Almoster','Lisboa','913456789')] ?
yes

| ?- utenteNome('Joao',R).
R = [(2,'Joao',12,'Rua da Ramada','Guimaraes','929876543')] ?
yes

| ?- utenteIdade(35,R).
R = [(1,'Carlos',35,'Rua D.Pedro V','Braga','253456789'),(10,'Filipa',35,'Rua Padre Vitorino','Faro',
yes

| ?- utenteRua('Rua das Flores',R).
R = [(9,'Maciel',45,'Rua das Flores','Porto','935731290')] ?
yes

| ?- utenteCidade('Guimaraes',R).
R = [(2,'Joao',12,'Rua da Ramada','Guimaraes','929876543'),(3,'Julio',89,'Rua das Victorias','Guimara
yes

| ?- utenteContacto('935436789',R).
R = [(3,'Julio',89,'Rua das Victorias','Guimaraes','935436789')] ?
yes
```

4.3 Identificar as instituições prestadoras de cuidados de saúde

Para a implementação deste predicado foi necessário encontrar todos as instituições onde foram prestados cuidados de saúde. Como tal utilizamos o predicado **solucoes** para encontrar as soluções que respondiam a esta questão.

De forma ao resultado apresentado não apresentar informação replicada procedemos a desenvolver o **retiraRep** para isso mesmo. Este utiliza o predicado **retiraEle** que irá retirar um elemento de uma lista. Assim iremos obter os resultados pretendidos.

```
instCuidSaud(R) :- solucoes(I,cuidado_prestado(_,_,I,_),L),
                   retiraRep(L,R).

retiraRep([],[]).
retiraRep([X|A],R) :- retiraEle(X,A,L),
                      retiraRep(L,T),
                      R = [X|T].

retiraEle(A,[],[]).
retiraEle(A,[A|Y],T) :- retiraEle(A,Y,T).
retiraEle(A,[X|Y],T) :- X \== A,
                        retiraEle(A,Y,R),
                        T = [X|R].
```

Exemplo output

```
| ?- instCuidSaud(R).
R = ['Hospital Privado de Braga','Hospital de Braga','IPO','Hospital de Santa Maria','Hospital da Luz
yes
```

4.4 Identificar os cuidados prestados por instituição/cidade

A implementação deste predicado é semelhante à anterior, na qual utilizamos o predicado **solucoes** para encontrarmos os cuidados prestados numa dada instituição/cidade.

```
cuidInst(I,R) :- solucoes(C,cuidado_prestado(_,C,I,_),R).

cuidCid(C,R) :- solucoes(S,cuidado_prestado(_,S,_,C),P),
                retiraRep(P,R).
```

Exemplo output

```
    | ?- cuidInst('Hospital Privado de Braga',R).
R = ['Pediatria'] ?
yes

| ?- cuidCid('Guimaraes',R).
R = ['Urgência'] ?
yes
```

4.5 Identificar os utentes de uma instituição/serviço

A implementação deste predicado para o caso da instituição baseia-se no uso do predicado **solucoes** para encontrar todos os utentes de uma dada instituição. Usamos também dois predicados auxiliares de forma a conseguirmos apresentar os resultados na forma de strings. Para o caso do serviço do serviço usamos o predicado **solucoes** para encontrar todos os serviços e depois usa-mos dois predicados auxiliares para evitarmos ter conhecimento repetido e para apresentarmos como resultado final os nomes dos utentes.

```
utentesInstituicao(I,R) :- solucoes(S, cuidado_prestado(S,_,I,_), P),
                           retiraRep(P,F),
                           utServ(F,R).

utServ([S],R) :- utentesServico(S,R).
utServ([S|Ss],R) :- utentesServico(S,F),
                    utServ(Ss,P),
                    concat(F,P,R).


utentesServico(S,R) :- solucoes(U, atos(_,U,S,_,_,_), P),
                       retiraRep(P,F),
                       idUt(F,R).

idUt([U],R) :- utenteID(U,R).
idUt([U|Us],R) :- utenteID(U,F),
                  idUt(Us,P),
                  concat(F,P,R).


concat([],L2,L2).
concat(L1,[],L1).
concat([X|L1],L2,[X|L]) :- concat(L1,L2,L).
```

Exemplo output

```
| ?- utentesInstituicao('Hospital Privado de Braga', R).
R = [(1,'Carlos',35,'Rua D.Pedro V','Braga','253456789'),(3,'Julio',89,'Rua das Victorias','Guimaraes
yes


| ?- utentesServico(1,R).
R = [(1,'Carlos',35,'Rua D.Pedro V','Braga','253456789'),(3,'Julio',89,'Rua das Victorias','Guimaraes
yes
```

4.6 Identificar atos médicos realizados por utente/instituição/serviço

Para a implementação deste predicado, tal como já visto nas questões anteriores, usamos o predicado **solucoes** para encontrar todos os atos médicos realizados por um utente/numa instituição/num serviço. Recorremos ao **retiraRep** para eliminar repetições de informação.

```
atoUte(U,R) :- solucoes((X,U,Y,Z,W,Q), atos(X,U,Y,Z,W,Q), P),
               retiraRep(P,R).


atoInst(I,R) :- solucoes(S, cuidado_prestado(S,_,I,_), F),
                retiraRep(F,P),
                servAto(P,R).

servAto([S],R) :- atoServ(S,R).
```

```
servAto([S|Ss],R) :- atoServ(S,F),
                     servAto(Ss,P),
                     concat(F,P,R)

atoServ(S,R) :- solucoes((X,Y,S,Z,W,Q), atos(X,Y,S,Z,W,Q), R).
```

Exemplo output

```
| ?- atoUte(3,R).
R = [('19-03-17',3,9,'Amarela','Dr.Bones',50),('03-03-17',3,1,'Sem_pulseira','Dra.Candida',45)] ?
yes

| ?- atoInst('Hospital Privado de Braga',R).
R = [('11-01-17',1,1,'Sem_pulseira','Dr.Pardal',2),('03-03-17',3,1,'Sem_pulseira','Dra.Candida',45)]
yes

| ?- atoServ(3,R).
R = [('10-02-17',4,3,'Sem_pulseira','Dr.FalcÃ£o',20)] ?
yes
```

4.7 Determinar todas as instituições/serviços a que um utente já recorreu

A implementação destes predicados no caso das instituições a que um utente já recorreu é realizada com a ajuda de dois predicados auxiliares que servem para apresentarem o resultado que foi pedido , sendo que para isso é feita uma invocação do predicado nServUte que corresponde a todos os serviços a que um utente já recorreu. No casos dos serviços a que um utente já recorreu a implementação é feita de uma forma mais simplista pois só é preciso verificar todos os atos realziados por um dado utente.

```
nInstUte(U,R) :- nServUte(U,F),
                 servicosInstituicao(F,P),
                 retiraRep(P,R).

servicosInstituicao([S],R) :- solucoes(I, cuidado_prestado(S,_,I,_), R).
servicosInstituicao([S|Ss],R) :-  solucoes(I, cuidado_prestado(S,_,I,_), P),
                                  servicosInstituicao(Ss,F),
                                  concat(P,F,R).


nServUte(U,R) :- solucoes(S, atos(_,U,S,_,_,_), P),
                 retiraRep(P,R).
```

Exemplos output

```
| ?- nInstUte(4,R).
R = ['Hospital de Braga'] ?
yes

| ?- nServUte(5,R).
R = [8] ?
yes
```

4.8 Calcular o custo total dos atos médicos por utente/serviço/instituição/data

A implementação destes predicados foi realizada da mesma forma , variando assim em função de que parâmetros é que calculávamos o custo total dos atos médicos. Por exemplo para o calculo do custo total dos atos médicos por utente tivemos de usar o predicado **atoUte** que nos indica os atos de um dado utente e depois usamos o predicado **atoCusto** que soma os custos de todos os atos desse utente. A implementação dos outros três predicados é realizada de maneira análoga.

```
custoUte(U,R) :- atoUte(U,F),
                 atoCusto(F,R).

atoCusto([(_,_,_,_,_,C)],R) :- R is C.
atoCusto([(_,_,_,_,_,C)|Cs],R) :- atoCusto(Cs,F),
                                  R is C+F.

custoServ(S,R) :- atoServ(S,F),
                  atoCusto(F,R).

custoInst(I,R) :- atoInst(I,F),
                  atoCusto(F,R).

custoData(D,R) :- solucoes((D,X,Y,Z,W,Q), atos(D,X,Y,Z,W,Q), F),
                  atoCusto(F,R).
```

Exemplo output

```
| ?- custoUte(4,R).
R = 20 ?
yes

| ?- custoServ(1,R).
R = 47 ?
yes

| ?- custoInst('Hospital Privado de Braga',R).
R = 47 ?
yes

| ?- custoData('19-03-17',R).
R = 50 ?
yes
```

4.9 Remover utentes, cuidados e atos médicos

A implementação de todos estes predicados tem um fator em comum que é o uso do predicado **retroceder** que encontra todos os utentes/cuidados/atos médicos que queiramos remover. O **removeUtentes** requer um cuidado extra pois quando removemos um utente da base de conhecimento todos os seus atos também têm de ser removidos, e para isso utilizamos o predicado **removeTodosAtos**.

```
removeUtentes(U) :- solucoes((D,U,IDS),atos(D,U,IDS,_,_,_),R),
```

```
removeTodosAtos(R),
retroceder(utente(U,N,I,RU,CDD,CNT)).

removeTodosAtos([(D,IDUT,IDS)]) :- removeAtos(D,IDUT,IDS).
removeTodosAtos([(D,IDUT,IDS)|As]) :- removeAtos(D,IDUT,IDS),
    removeTodosAtos(As).


removeCuidados(I) :- retroceder(cuidado_prestado(I,D,C,Cid)).


removeAtos(D,IDUT,IDS) :- retroceder(atos(D,IDUT,IDS,_,_,_)).

retroceder(E) :- solucoes(I,+E::I,L),
                 teste(L),
                 remove(E).
```

Exemplo output

```
| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').
utente(15, 'Tiago', 16, 'Rua Monsenhor de Melo', 'Vilamoura', '936150873').

yes
| ?- removeUtentes(15).
yes
| ?- listing(utente).
utente(1, 'Carlos', 35, 'Rua D.Pedro V', 'Braga', '253456789').
utente(2, 'Joao', 12, 'Rua da Ramada', 'Guimaraes', '929876543').
utente(3, 'Julio', 89, 'Rua das Victorias', 'Guimaraes', '935436789').
utente(4, 'Ana', 25, 'Rua Conde Almoster', 'Lisboa', '913456789').
utente(5, 'Carolina', 50, 'Rua do Caires', 'Braga', '253987654').
utente(6, 'Joana', 26, 'Av.da Boavista', 'Porto', '961234567').
utente(7, 'Fernando', 65, 'Rua do Loureiro', 'Viana do Castelo', '966668569').
utente(8, 'Rute', 18, 'Av. da Liberdade', 'Braga', '916386423').
utente(9, 'Maciel', 45, 'Rua das Flores', 'Porto', '935731290').
utente(10, 'Filipa', 35, 'Rua Padre Vitorino', 'Faro', '289347681').
utente(11, 'Mauro', 76, 'Rua Gil Vicente', 'Montalegre', '276327904').
utente(12, 'Laura', 90, 'Rua Fernando Mendes', 'Leiria', '244000045').
```

```
utente(13, 'Jaime', 50, 'Av. Norton Matos', 'Barcelos', '914768180').
utente(14, 'LourenÃ§o', 26, 'Rua da Boavista', 'Guimaraes', '926306127').

yes



| ?- listing(atos).
atos('02-01-17', 15, 10, 'Verde', 'Dra.Sara', 17.5).
atos('24-02-17', 8, 4, 'Sem_pulseira', 'Dr.Mourao', 4).
atos('25-02-17', 1, 2, 'Sem_pulseira', 'Dr.Barroso', 12).
atos('28-01-17', 13, 9, 'Laranja', 'Dr.Tomas', 5).
atos('10-02-17', 4, 3, 'Sem_pulseira', 'Dr.FalcÃ£o', 20).
atos('19-03-17', 3, 9, 'Amarela', 'Dr.Bones', 50).
atos('11-01-17', 1, 1, 'Sem_pulseira', 'Dr.Pardal', 2).
atos('12-02-17', 5, 8, 'Sem_pulseira', 'Dra.Teresa', 13.75).
atos('20-03-17', 11, 11, 'Sem_pulseira', 'Dra.Marta', 13).
atos('27-01-17', 2, 7, 'Amarela', 'Dr.Pedro Martins', 11).
atos('01-01-17', 6, 6, 'Laranja', 'Dr.Reveillon', 16).
atos('03-03-17', 3, 1, 'Sem_pulseira', 'Dra.Candida', 45).
atos('08-03-17', 9, 9, 'Vermelha', 'Dr.Lima', 50).
atos('14-02-17', 14, 7, 'Amarela', 'Dra.Mafalda', 23).
atos('30-01-17', 7, 5, 'Sem_pulseira', 'Dr.Quimio', 10).
atos('01-02-17', 1, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('01-03-17', 10, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('10-03-17', 12, 11, ' Sem_pulseira', 'Dr.Luis', 12.5).

yes
| ?- removeAtos('10-03-17',  12, 11).
yes
| ?- listing(atos).
atos('02-01-17', 15, 10, 'Verde', 'Dra.Sara', 17.5).
atos('24-02-17', 8, 4, 'Sem_pulseira', 'Dr.Mourao', 4).
atos('25-02-17', 1, 2, 'Sem_pulseira', 'Dr.Barroso', 12).
atos('28-01-17', 13, 9, 'Laranja', 'Dr.Tomas', 5).
atos('10-02-17', 4, 3, 'Sem_pulseira', 'Dr.FalcÃ£o', 20).
atos('19-03-17', 3, 9, 'Amarela', 'Dr.Bones', 50).
atos('11-01-17', 1, 1, 'Sem_pulseira', 'Dr.Pardal', 2).
atos('12-02-17', 5, 8, 'Sem_pulseira', 'Dra.Teresa', 13.75).
atos('20-03-17', 11, 11, 'Sem_pulseira', 'Dra.Marta', 13).
atos('27-01-17', 2, 7, 'Amarela', 'Dr.Pedro Martins', 11).
atos('01-01-17', 6, 6, 'Laranja', 'Dr.Reveillon', 16).
atos('03-03-17', 3, 1, 'Sem_pulseira', 'Dra.Candida', 45).
atos('08-03-17', 9, 9, 'Vermelha', 'Dr.Lima', 50).
atos('14-02-17', 14, 7, 'Amarela', 'Dra.Mafalda', 23).
atos('30-01-17', 7, 5, 'Sem_pulseira', 'Dr.Quimio', 10).
atos('01-02-17', 1, 6, 'Verde', 'Dra.Luisa', 25.5).
atos('01-03-17', 10, 6, 'Verde', 'Dra.Luisa', 25.5).

yes
```

```
| ?- listing(cuidado_prestado).
cuidado_prestado(1, 'Pediatria', 'Hospital Privado de Braga', 'Braga').
cuidado_prestado(2, 'Cardiologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(3, 'Ortopedia', 'Hospital de Braga', 'Braga').
cuidado_prestado(4, 'Oftalmologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(5, 'Oncologia', 'IPO', 'Porto').
cuidado_prestado(6, 'UrgÃªncia', 'Hospital de Santa Maria', 'Porto').
cuidado_prestado(7, 'UrgÃªncia', 'Hospital da Luz', 'Guimaraes').
cuidado_prestado(8, 'Neurologia', 'Centro Hospitalar Sao Joao', 'Porto').
cuidado_prestado(9, 'UrgÃªncia', 'Hospital de Braga', 'Braga').
cuidado_prestado(10, 'UrgÃªncia', 'Hospital Lusiadas', 'Faro').
cuidado_prestado(11, 'Otorrinolaringologia', 'Hospital de Vila Real', 'Vila Real').

yes
| ?- removeCuidados(11).
yes
| ?- listing(cuidado_prestado).
cuidado_prestado(1, 'Pediatria', 'Hospital Privado de Braga', 'Braga').
cuidado_prestado(2, 'Cardiologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(3, 'Ortopedia', 'Hospital de Braga', 'Braga').
cuidado_prestado(4, 'Oftalmologia', 'Hospital de Braga', 'Braga').
cuidado_prestado(5, 'Oncologia', 'IPO', 'Porto').
cuidado_prestado(6, 'Urgência', 'Hospital de Santa Maria', 'Porto').
cuidado_prestado(7, 'Urgência', 'Hospital da Luz', 'Guimaraes').
cuidado_prestado(8, 'Neurologia', 'Centro Hospitalar Sao Joao', 'Porto').
cuidado_prestado(9, 'Urgência', 'Hospital de Braga', 'Braga').
cuidado_prestado(10, 'Urgência', 'Hospital Lusiadas', 'Faro').

yes
```

5 Funcionalidades Extra

Após a conclusão das funcionalidades básicas exigidas e de forma a enriquecer o nosso exercício decidimos implementar funcionalidades extra. Assim sendo, nesta parte do relatório vamos fazer uma explicação destas novas funcionalidades.

5.1 Numero de serviços/atos/utentes

Para realizarmos estes predicados foi necessário a criação de um predicado que permitisse calcular o comprimento de uma lista, para esse efeito implementamos o predicado **comprimento**.

```
comprimento([],0).
comprimento([X|P],N) :- comprimento(P,G) ,
                        N is 1 + G.
```

Para efetuarmos as contagens das características pretendidas aplicamos em cada uma o predicado **comprimento** , como podemos verificar de seguida com o exemplo dos predicados **numeroServicos , numeroUtentes , nCorPul** e **numeroAtos**.

```
numeroServicos(I,R) :- solucoes(S, cuidado_prestado(S,_,I,_), P),
                       comprimento(P,T),
                       R is T.


numeroUtentes(R) :- solucoes(U,utente(U,_,_,_,_,_,_),L),
                    comprimento(L,T),
                    R is T.

numeroAtos(R) :- solucoes(A,atos(A,_,_,_,_,_,_),L),
                 comprimento(L,T),
                 R is T.

nCorPul(C,R) :- solucoes(C,atos(_,_,_,C,_,_),L),
                comprimento(L,T),
                R is T.
```

Exemplo output

```
| ?- numeroServicos('Hospital Privado de Braga',R).
R = 1 ?
yes

| ?- numeroUtentes(R).
R = 15 ?
yes

| ?- numeroAtos(R).
R = 18 ?
yes

| ?- nCorPul('Sem_pulseira',R).
R = 9 ?
yes
```

5.2 Cores da Pulseira

Predicado usado para retornar o conhecimento acerca das diferentes cores das pulseiras que são registadas aquando da realização de um ato. A implementação deste predicado consiste em encontrar todas as diferentes cores das pulseiras (uso do predicado **solucoes**) e depois remover as repetidas (uso do predicado **retiraRep**).

```
corPuls(R) :- solucoes(C,atos(_,_,_,C,_,_),T),
              retiraRep(T,R).
```

Exemplo output

```
| ?- corPuls(R).
R = ['Verde','Sem_pulseira','Laranja','Amarela','Vermelha'] ?
yes
```

5.3 Ato Médico Mais Caro Registado até ao Momento

Para a implementação deste predicado **msCaro**, tal como já visto nas questões anteriores, usamos o predicado **solucoes** para encontrar todos os custos dos atos médicos realizados. Recorremos ao **maxLst** para obtermos o custo do ato médico mais caro registado até ao momento com o apoio do predicado **maior** que nos permite saber qual o maior valor entre dois valores. De seguida é executado mais uma vez o predicado **solucoes** para captar os atos médicos cujo custo é o obtido anteriormente pelo predicado **maxLst**. Por último, é usado o predicado **servAtos** para exibir os serviços mais caros.

```
msCro(R) :- solucoes(Q,atos(_,_,_,_,_,Q),T),
            maxLst(T,0,C),
            solucoes((X,U,Y,Z,W,C), atos(X,U,Y,Z,W,C),K),
            servAtos(K,R).

servAtos([(_,_,S,_,_,_)],R) :- solucoes((S,B,C,D), cuidado_prestado(S,B,C,D), R).
servAtos([(_,_,S,_,_,_)|Cs],R) :- solucoes((S,B,C,D), cuidado_prestado(S,B,C,D), P),
                                  servAtos(Cs,F),
                                  concat(P,F,R).

maxLst([C],K,R) :- maior(C,K,X),
                   R is X.
maxLst([C|Cs],K,R) :- maior(C,K,X),
                      maxLst(Cs,X,R).

maior(X,Y,X) :- X >= Y.
maior(X,Y,Y) :- Y > X.
```

Exemplo output

```
| ?- msCro(R).
R = [(8,'Neurologia','Centro Hospitalar Sao Joao','Porto'),(9,'Urgência','Hospital de Braga','Braga')
yes
```

5.4 Médicos de uma dada Instituição

A implementação deste predicado, requer o uso do predicado **solucoes** que serve para encontrar os IDs dos cuidados prestados para que de seguida seja aplicado o predicado **getDoc** que retorna os doutores de uma ato médico de uma dada isntituição.

```
medInst(I,R) :- solucoes(ID,(cuidado_prestado(ID,_,I,_)),L),
                getDoc(L,R).

getDoc([],[]).
getDoc([X|XS],RS) :- solucoes(M,atos(_,_,X,_,M,_),MS),
                     getDoc(XS,TS),
                     append(MS,TS,F),
                     retiraRep(F,RS).
```

Exemplo output

```
| ?- medInst('Hospital Privado de Braga',R).
R = ['Dr.Pardal','Dra.Candida'] ?
yes
```

5.5 Média dos custos dos atos médicos

Este predicado tem como objetivo calcular a média de custos dos atos médicos, e para isso usamos o predicado **solucoes** para encontrar todos os atos médicos, de seguida usamos o predicado **atoCusto** que calcula o custo total de uma lista de atos e depois usamos o predicado **comprimento** para calcularmos o comprimento da lista dos atos e por fim calculamos a média entre os dois valores calculados anteriormente.

```
mediaCusto(R) :- solucoes((X,U,Y,Z,W,Q), atos(X,U,Y,Z,W,Q), P),
                 atoCusto(P,F),
                 comprimento(P,K),
                 R is F/K.
```

Exemplo output

```
| ?- mediaCusto(R).
R = 20.180555555555557 ?
yes
```

5.6 Ordenar os Atos Médicos Registados até ao Momento

A implementação deste predicado tem como objetivo ordenar os atos médicos registados até ao momento. Numa primeira fase utilizamos o predicado **solucoes** para obtermos uma lista de todos os atos, de seguida aplicamos o predicado **ordenaL** que tem como objetivo ordenar uma lista por ordem crescente de custo de um ato.

```
ordCst(R) :- solucoes(Q,atos(_,_,_,_,_,Q),T),
             ordenaL(T,R).


ordenaL([X],[X]).
ordenaL([H|T],OL) :- ordenaL(T,OT),
                     insr(H,OT,OL).

insr(X,[],[X]).
insr(X,[H|T],[X,H|T]) :- X =< H.
insr(X,[H|T],[H|NT]) :- X > H,
                        insr(X,T,NT).
```

Exemplo output

```
| ?- ordCst(R).
R = [2,4,5,10,11,12,12.5,13,13.75,16|...] ?
yes
```

6 Conclusão

A realização deste trabalho prático, o primeiro desta unidade curricular, permitiu-nos aprimorar e consolidar o conhecimento trabalhado nas aulas, nomeadamente no que diz respeito à programação lógica. Como tal, o suporte utilizado para desenvolvermos tal conhecimento foi o *PROLOG*, o qual utilizamos para representar um ambiente de conhecimento e construção de mecanismos de raciocínio para a resolução de um problema que simula a prestação de cuidados de saúde através de serviços/atos médicos.

Como tal, mesmo antes de tratarmos de responder aos exercícios enunciados, foi necessário analisarmos e percebermos toda a dinâmica e aspetos que envolvem tal problema, de forma a podermos melhorar e aproximar o mais possível da realidade o nosso trabalho desenvolvido.

Em jeito de conclusão, podemos afirmar que realmente tiramos proveito da realização deste trabalho prático, tendo também a consciência de que cumprimos e conseguimos melhorar o problema proposto.