Ficha Prática 11

11.1 Objectivos

1. Leitura e escrita de informação em ficheiros: `see` e `tell`.
2. Praticar a utilização de `assert` e `retract`.

11.2 Conceitos

11.2.1 `read` e `write`

O SWI Prolog fornece diversos predicados para leitura e escrita. Listam-se de seguida alguns deles:

- `read(-Term)` — lê o próximo termo da *stream* de entrada (*input*) e unifica-o com `Term`. Se a *stream* de entrada actual já não possuir mais informação, `Term` é unificado com o átomo `end_of_file`.
- `write(+Term)` — escreve o termo `Term` na *stream* de saída (*output*) actual.
- `nl` — escreve o caracter de mudança de linha na *stream* de saída.
- `writeln(+Term)` — escreve o termo `Term` seguido de uma mudança de linha (equivalente a: `write(+Term), nl`).
- `writeq(+Term)` — escreve o termo `Term`, sempre que necessário são colocadas plicas nos átomos (um termo escrito com `writeq/1` pode depois ser lido com o `read/1`).

Analise os seguintes exemplos:

```
?- read(N), write('Olá '), write(N).
|: 'Manuel Albertino'.
Olá Manuel Albertino

N = 'Manuel Albertino'

Yes
?- read(N), write('Olá '), writeq(N).
|: 'Manuel Albertino'.
Olá 'Manuel Albertino'

N = 'Manuel Albertino'

Yes
?- read(N), write('Olá '), writeq(N).
|: jose.
```

```
Olá jose

N = jose

Yes
?- read(N), writeln('Olá '), write(N).
|: 'filomena'.
Olá
filomena

N = filomena

Yes
?- read(N), writeln('Olá '), writeq(N).
|: 'filomena'.
Olá
filomena

N = filomena

Yes
```

11.2.2 `tell` (`told`, `telling`)

- `tell(F)` — abre `F` para escrita e torna-o a *stream* de saída actual (se `F` já for uma *stream*, limita-se a torná-la a *stream* de saída actual).
- `told` — fecha a *stream* de saída actual.
- `telling(S)` — `S` é unificado com a *stream* de saída actual.

Para escrever no ficheiro `fich.txt` o facto `impar(1)` e, garantidamente, voltar a deixar o interpretador com a *stream* de saída como estava anteriormente, utiliza-se o código:

```
telling(S),
tell('fich.txt'),
write(impar(1)),
writeln('.').
told,
tell(S).
```

Escreveu-se um ponto (.) no fim da expressão para que ela corresponda a um termo Prolog bem formado. Deste modo, mais tarde será possível ler o termo para o voltar a colocar na base de conhecimento.

11.2.3 `see` (`seen`, `seeing`)

- `see(F)` — abre `F` para leitura e torna-o a *stream* de entrada actual (se `F` já for uma *stream*, limita-se a torná-la a *stream* de entrada actual).
- `seen` — fecha a *stream* de entrada actual.
- `seeing(S)` — `S` é unificado com a *stream* de entrada actual.

Para ler do ficheiro anterior o facto lá escrito e voltar a colocá-lo na base de conhecimento, deixando, garantidamente, o interpretador com a *stream* de entrada como estava anteriormente, utiliza-se o código:

```
seeing(S),
see('fich.txt'),
read(T),
assert(T),
seen,
see(S).
```

11.2.4 `repeat`

O predicado `repeat` sucede sempre, sendo utilizado para controlar o *backtracking*. Um exemplo de utilização é apresentado na próxima secção.

11.2.5 Leitura de informação a partir de ficheiros

A leitura de informação a partir de um ficheiro, e o seu carregamento para a base de conhecimento do Prolog, obedece a um esquema geral que se baseia na repetição da leitura até que se encontre o fim do ficheiro (representado pelo símbolo `end_of_file`). Em simultâneo com a leitura, a informação lida é processada (tarefa que é inerente à complexidade do problema) e gravada em memória (através do predicado `assert`).

O código padrão para este comportamento é apresentado de seguida:

```
carrega(F) :- see(F),
              repeat,
                read(T),
                processaTermoLido(T),
                T==end_of_file,
              !,
              seen.


processaTermoLido(end_of_file).
processaTermoLido(P) :- assert(P).  % P é o resultado da leitura.
                                    % Atenção que pode ser necessário
                                    % fazer algumas operações sobre P,
                                    % dependendo da lógica do problema.
```

11.3 Exercícios

1. Com base no que sabe sobre os predicados `see` e `tell` procure prever o resultado das seguintes queries:

   (a) `?- write('Escreva o seu nome:'), read(N), atom_concat('Olá ', N, S), write(S).`

   (b) `?- write('Escreva o seu nome:'), read(N), tell('Teste.txt'), atom_concat('Olá ', N, S), write(S), told.`
   Veja o conteúdo do ficheiro `Teste.txt`.

(c) Crie o ficheiro `Entrada.txt` com uma linha contendo o seu nome sob a forma de átomo prolog (por exemplo, "`'Manuel Costa'.`"):

```
?- see('Entrada.txt'), read(N), atom_concat('Olá ', N, S),
write(S), seen.
```

(d) `?- tell(t1), write(olá), tell(t2), write(olé), told, write(oli), told, write(olo).`

(e) `?- tell(t1), write(olá), telling(F), tell(t2), write(olé), tell(F), write(oli), told, write(olo).`

2. Num ficheiro encontram-se por ordem aleatória predicados que representam definições de figuras geométricas sob a forma: `quadrado(p, tam)`, `triangulo(p1, p2, p3)`, `rectangulo(p1, p2)` e `circunf(p, r)`:

   (a) Escreva o predicado `lerFiguras/1` que carrega para a Base de Conhecimento todas as figuras presentes no ficheiro.

   (b) Escreva o predicado `lerQuadrados/1` que carrega para a Base de Conhecimento todos os quadrados presentes no ficheiro.

   (c) Escreva o predicado `maiorCircunf/2` que define qual a maior circunferência presente no ficheiro (sem alterar a base de conhecimento).

   (d) Escreva ainda o predicado `listaQuad/2` que define a lista dos quadrados presentes na Base de Conhecimento que possuem como lado um valor dado.

   (e) Escreva o predicado `guardarQuad/1` que permite guardar, num ficheiro cujo nome é dado, todos os quadrados existentes na base de conhecimento.

   (f) Escreva o predicado `guardarTri/2` que permite guardar, num ficheiro cujo nome é dado, todos os triângulos existentes na base de conhecimento que possuam como um dos vértices o ponto `P` passado como parâmetro.

3. Termos da forma `telefonou(pessoa1, pessoa2)` são utilizados numa Base de Conhecimento para indicar que uma dada pessoa contactou outra. Escreva predicados que lhe permitam:

   (a) Carregar para a BC tais factos a partir de um ficheiro F.

   (b) Determinar qual a pessoa que mais vezes ligou para a pessoa X.

   (c) Determinar qual o conjunto de pessoas que a pessoa Y contactou.

   (d) Determinar a lista de pares `(pessoa, lista_de_pessoas_contactadas)`.

   (e) Sabendo que cada chamada custa P, gravar num ficheiro, para cada pessoa, o custo das suas chamadas.