Processamento de ficheiros

O processamento de ficheiros baseia-se, normalmente, na repetição da leitura até que se encontre o fim de ficheiro. Obedece normalmente a um dos seguintes esquemas:

Usando apenas a recursão

```
processfile(F) :- seeing(S), see(F),
                  read(Term), process(Term),
                  seen, see(S).

process(end_of_file) :- !.
process(Term) :- treat(Term), read(T),
                 process(T).
```

Usando *repeat loops* (mais eficiente)

```
processfile(F) :- seeing(S), see(F),
                  repeat,
                    read(T),
                    process(T),
                    T == end_of_file,
                  !,
                  seen, see(S).

process(end_of_file).
process(Term) :- treat(Term).
```

Nota: *treat* deverá fazer o processamneto do termo actal.

Interacção com o utilizador

Exemplo: Implementação de um menu.

```
menu :- repeat,
          write('==== MENU ===='), nl,
          write('1. - Opção A'), nl,
          write('2. - Opção B'), nl,
          write('0. - Sair'), nl,
          read(X),
          opcao(X),
          X==0,
        !.

opcao(0) :- !.
opcao(1) :- write('Escolheu a opção A ...'), nl, !.
opcao(2) :- write('Escolheu a opção B ...'), nl, !.

opcao(_) :- write('Opção inválida!'), nl, !.
```

Exemplo: Validar a entrada de dados.

```
le_int(X) :- repeat,
               read(X),
               integer(X),
             !.
```

```
| ?- le_int(X).
|: abc.
|: 1.4.
|: 3.
X = 3 ?
yes
```


Exercícios:

1. Relembre o problema apresentado anteriormente, em que a informação referente aos horários das salas de aula está guardada na base de conhecimento em factos da forma: `sala(`*num,dia,inicio,fim,discipl,tipo*`).`
   Defina um predicado `salva(+Ficheiro)` que guarda no `Ficheiro` os factos com a informação sobre as salas que são válidas (i.e., em que a hora de início é inferior à hora de fim).

2. Defina o predicado `findterm(+Term)` que escreve o écran o primeiro termo lido que unifica com `Term`.

3. Defina o predicado `findalltermsfile(+Term,+FileName)` que escreve no écran todos os termos do ficheiro que unificam com `Term` (garanta que `Term` não é instanciado).

4. Defina o predicado `to_upper(+FileIn,+FileOut)` que recebe um ficheiro de texto `FileIn` e gera o ficheiro `FileOut` com o mesmo texto de entrada mas convertido para letras maiúsculas. (Note que apenas as letras minúsculas são alteradas, o resto deverá ser mantido.)

5. Defina o predicado `numera_linhas(+FileIn,+FileOut)` que recebe o ficheiro `FileIn` e produz o ficheiro `FileOut`, que contém as mesmas linhas de `FileIn`, mas com as linhas numeradas.

6. Relembre o problema do cálculo da nota final à disciplina de *Lógica Computacional*, apresentado anteriormente, em que as notas do alunos estão guardadas na base de conhecimento em factos da forma:

   `modalidadeA(`*numero, nome, fichas, exame*`)`
   `modalidadeB(`*numero, nome, fichas, trabalho, exame*`)`

   Pretende-se agora poduzir a pauta final, tendo a informação sobre os alunos inscritos num ficheiro com factos da forma: `aluno(`*numero, nome, tipo*`)`

   Defina o predicado `gera_pauta(+Inscritos,+Pauta)` que dado o ficheiro `Inscritos`, vai lendo a informação sobre os alunos inscritos à disciplina e, sem alterar a base de conhecimento, produz no ficheiro `Pauta` o texto com a pauta devidamente preenchida. Note que:

   - -- se o aluno está inscrito mas não se submeteu a avaliação, a sua nota será Faltou;
   - -- se o aluno tem alguma das componentes de avaliação inferior a 9, ou a média final arredondada inferior a 10, a sua nota será Reprovado;
   - -- nos restantes casos, será o arredondamento da média pesada (se quiser, pode escrever também a nota por extenso).