Sistema de Diagnóstico de doenças Respiratórias, Cardiovasculares, Crônicas e Viroses.

Graduandos: Diogo Fernando e Eduardo Bluthgen

RA: 93814 e 99164

Estrutura Geral e Entrada.

- Base de conhecimento (Prolog).
- Sistema baseado no predicado sintoma(Sintoma, Doença).
  - Sintoma é um átomo e Doença também.
- A base de conhecimento foi povoada manualmente com as sentenças na forma acima.
  - Ex:

```
sintoma(febre,hiv).
sintoma(tosse,hiv).
```

Estrutura Doenças

- As doenças foram cadastradas em suas classes e generalizadas como doença(X).
  - Ex:

```
doenca(X) :- doencaRespiratoria(X).
doenca(X) :- doencaCardiovascular(X).
doenca(X) :- doencaCronica(X).
doenca(X) :- doencaVirose(X).

doencaCronica(diabetes).
doencaRespiratoria(pneumonia).
doencaCardiovascular(miocardite).
doencaVirose(dengue).
```

Algoritmos

- Interface para interação com o usuário. Contém as referências para as funcionalidades do sistema.

```
switchDoMenu(In) :- (In == 0 -> halt;
                     In == 1 -> login();
                     In == 2 -> imprimeTodasDoencas();
                     In == 3 -> imprimeTodosSintomas();
                     In == 4 -> imprimeDoencasDeEmergencia();
                     In == 5 -> write('Qual classe de doenças? Cronica, respiratoria, cardiovascular ou virose: '),
                      read(C), imprimeDoencaPorClasse(C);
                     In == 6 -> write('Qual o nome da doença? '), read(D), indicacaoProfissional(D);
                     In == 7 -> diagnostico();
                     In == 8 -> adicionarDoenca();
                     In == 9 -> excluirDoenca();
                    adicionarUsuario()), menu().
```

Algoritmos

- Funções que são chamadas pelo menu

```
imprimeDoencaPorClasse(C):- doencaPorClasse(C, L), pd([]), pd(L).
imprimeTodosSintomas() :- listaSintomas(L), ps(L, L), !.
imprimeTodasDoencas() :- listaDoencas(L), pd(L), !.
diagnostico() :- confirmaSintomas().
imprimeDoencasDeEmergencia() :- listaEmergencia(L), pd(L).
adicionarDoenca() :- get_flag(logado, usuario), escreveDoencaNoArquivo(), make.
excluirDoenca():- get_flag(logado, usuario), escreveNoArquivo(), make.
adicionarUsuario() :- get_flag(logado, usuario), escreveUsuarioNoArquivo(), make.
indicacaoProfissional(D) :- indicaMedico(M, D), phh(['Procure', um, M]), !.
login() :- verificaLogin().
```

Algoritmos

```
% Função que concatena com o arquivo fonte um
% aqrvuivo que contem um predicado de uma doença que foi
% excluida.
escreveNoArquivo() :- current_output(Terminal), open('trabalho.pl', append, Arq),
excluiDoenca(Arq, Terminal, D),  set_output(Terminal), close(Arq).

excluiDoenca(Arq, Terminal, Doenca) :- set_output(Terminal),
write('Digite o nome da doenca a ser excluida: '),
read(Doenca),set_output(Arq), write('doencaExcluida('),
write(Doenca), write(').').
%--------------
```

Algoritmos

```
% ! Função que escreve em um arquivo uma nova doença e seus
% sintomas no padrao do arquivo e o concatena com o arquivo fonte.
%  para pegar uma doença é necessario passar o nome da doença e
%  depois a classe dela, para pegar os sintomar é passado o numeros dos
%  sintomas e é feito um loop com essa quantidade.

escreveDoencaNoArquivo() :- current_output(Terminal), open('trabalho.pl', append, Arq),
escreveDoenca(Arq, Terminal, D), write('Digite o numero de sintomas a serem cadastrados: '),
read(Num), escreveSintomas(Arq, Terminal, D, 0, Num), close(Arq).

escreveDoenca(Arq, Terminal, Doenca) :- set_output(Terminal),
write('Digite o nome da doenca a ser cadastrada: '), read(Doenca),
write('Digite a classe da doenca (cronica, cardiovascular, respiratoria, virose): '),
read(Classe),set_output(Arq), write('doenca'),
(Classe == 'cronica' -> write('Cronica(')
; Classe == 'respiratoria' -> write('Respiratoria(')
; Classe == 'cardiovascular' -> write('Cardiovascular(')
; write('Virose(')), write(Doenca), write(').'), nl, set_output(Terminal).

escreveSintomas(_, _, _, I, N) :- I == N.

escreveSintomas(Arq, Terminal, D, I, N) :- set_output(Terminal),
write('Digite o nome do sintoma: '), read(S), set_output(Arq),
write('sintoma('), write(S), write(','), write(D), write(').'), nl,
set_output(Terminal), I1 is I + 1, escreveSintomas(Arq, Terminal, D, I1, N).
%! ------------
```

Algoritmos

```
% ! Funções que validam o usuario e senha passados para o sistema. Se o
% usario e senha passados unificarem com os existentes no banco o login
% é efetuado.


confirmaSenha(Nome) :- write('Digite sua senha: '), read(Senha),
(senha(Nome, Senha) -> flag(logado, _, usuario), write('Login efetuado com sucesso!');
write('Senha incorreta, tente novamente'), nl, confirmaSenha(Nome)).

verificaLogin() :- write('Digite seu nome de usuário: '), read(Nome),
(usuario(Nome) -> confirmaSenha(Nome) ;
write('Usuário não encontrado. Entrando como convidado.'), flag(logado, _, convidado)).
```

Algoritmos

```
% ! Função que escreve em um arquivo um novo usuario e senha para login
% no sistema e concatena no final do arquivo fonte.

escreveUsuarioNoArquivo() :- current_output(Terminal),
open('trabalho.pl', append, Arq), escreveUsuario(Arq, Terminal),
set_output(Terminal), close(Arq).

escreveUsuario(Arq, Terminal) :- set_output(Terminal),
write('Digite o nome de login do novo usuario: '), read(User),
write('Escreva a senha de '), write(User), write(': '), read(Senha),
set_output(Arq), write('usuario('), write(User), write('),'), nl,
write('senha('), write(User), write(','), write(Senha), write(').').
```

Algoritmos

```
%Funções usadas para mostrar na tela informções formatadas.
% pd->função que printa uma lista um elemento em cada linha.
% ps->função que printa o indice e o elemento da lista separando-os com
%um espaço e separando os elementos com uma nova linha.
% phh->função que printa uma lista separando cada elemento
%com um espaço.

pd([]) :- nl.
pd([H|T]) :- write(H), pd([]), pd(T).

ps([], _) :- nl.
ps([H|T], L) :-  indexOf(L, H, I), write(I), spaces(1), write(H), pd([]), ps(T, L).

phh([]) :- nl.
phh([H|T]) :- write(H), spaces(1), phh(T).
```