Trabalho Prático 1: Prolog

Hudson Rogério Proença Júnior
Ra108756

Início do sistema

```
Bem vindo ao sistema de gerenciamento medico :)
Para as opçoes dos menus, a informação de doenças e sintomas, digite a entrada com um ponto no final
, já para as informações do paciente, pode digitar normalmente sem essas considerações
Para cada um desses casos ha um aviso antes!
-- Sistema médico --

0. Sair
1. Cadastrar paciente
2. Alterar dados paciente
3. Vizualizar dados paciente
4. Remover paciente
5. Listar pacientes
6. Informar sintomas
```

Para iniciar o sistema basta digitar:
?- iniciar.

Estrutura do arquivo

```
'Jorge,20/06/2000, 123456'.
'Hudson,20/06/0, 123456'.
'Maria,20/04/2010, 123456'.
```

Cadastro de pacientes

```
Digite o nome do paciente: |: Hudson
Digite a data de nascimento do paciente:20/06/2000
Digite o CPF do paciente: 12345678

Paciente cadastrado com sucesso!

   Sistema médico
```

```
% funcao para cadastrar um paciente no arquivo pacientes.txt
cadastrarPaciente :-
    write('Digite o nome do paciente: '),
    ler_string_input(NomePaciente),
    write('Digite a data de nascimento do paciente:'),
    ler_string_input(DataNasc),
    write('Digite o CPF do paciente: '),
    ler_string_input(CpfPaciente),
    inserir_paciente(NomePaciente, DataNasc, CpfPaciente),
    write('\nPaciente cadastrado com sucesso!'), nl.
```

Cadastro de pacientes

```
% funcao para inserir um paciente no arquivo pacientes.txt
inserir_paciente(Nome, Data, Cpf) :-
    working_directory(CWD, CWD),
    PacienteFile = '/pacientes.txt',
    atom_concat(CWD,PacienteFile, Arquivo),
    open(Arquivo, append, Out),
    write(Out, '\''),
    write(Out, Nome),
    write(Out, ','),
    write(Out, Data),
    write(Out, ','),
    write(Out, Cpf),
    write(Out, '\''),
    write(Out, '.'),
    write(Out, '\n'),
    close(Out).
```

Alterar paciente

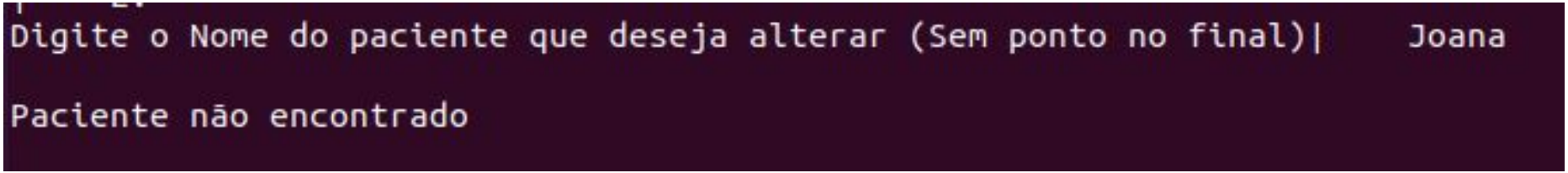
```
Digite o Nome do paciente que deseja alterar|    Hudson

Paciente encontrado no sistema:

Nome: Hudson
Data de nascimento: 20/06/2000
Cpf do paciente:  123456

Dentre as opções a seguir, digite qual informação você deseja alterar do paciente:
0. Voltar
1. Nome
2. Data de nascimento
3. Cpf
|    1.
Digite o novo nome do paciente: |    Jorge
Paciente atualizado!
```

Alterar paciente



Quando o paciente não é encontrado para alteração

Alterar paciente

```
% funcao para alterar dados do paciente
alterarPaciente :-
    write('Digite o Nome do paciente que deseja alterar'),
    ler_string_input(NomePaciente),
    get_paciente(NomePaciente, Paciente),
    Paciente \== '' ->
    get_all_pacientes(Pacientes), nl,
    write('Dentre as opções a seguir, digite qual informação você deseja alterar do paciente: '),
    quest_alterar_paciente(Pacientes, Paciente); menu.
```

```
% questionario para alteracao de dados do paciente
quest_alterar_paciente(Pacientes, Paciente) :-
    split_string(Paciente, ",", "", [Nome, Data, Cpf]), nl,
    write('0. Voltar'), nl,
    write('1. Nome'), nl,
    write('2. Data de nascimento'), nl,
    write('3. Cpf'), nl,
    read(Opcao),
    alterar_opcao(Opcao,Pacientes, Paciente, Nome, Data, Cpf).
```

Alterar paciente

```
% questionario alterar opcao 1
alterar_opcao(1,Pacientes, Paciente, _ ,Data, Cpf) :-
    write('Digite o novo nome do paciente: '),
    ler_string_input(NewNome),
    delete(Pacientes, Paciente, Result),
    escrever_arquivo(Result),
    inserir_paciente(NewNome, Data, Cpf),
    write('Paciente atualizado!').
```

```
% questionario alterar opcao 2
alterar_opcao(2, Pacientes, Paciente, Nome, _, Cpf) :-
    write('Digite a nova data de nascimento do paciente: '),
    ler_string_input(NewData),
    delete(Pacientes, Paciente, Result),
    escrever_arquivo(Result),
    inserir_paciente(Nome, NewData, Cpf),
    write('\nPaciente atualizado!\n').
```

```
%questionario alterar opcao 3
alterar_opcao(3, Pacientes, Paciente, Nome, Data, _) :-
    write('Digite o novo CPF do paciente: '),
    ler_string_input(NewCpf),
    delete(Pacientes, Paciente, Result),
    escrever_arquivo(Result),
    inserir_paciente(Nome, Data, NewCpf),
    write('Paciente atualizado!').
```

Consultar paciente

```
Digite o nome do paciente a ser consultado|    Jorge

Paciente encontrado no sistema:

Nome: Jorge
Data de nascimento: 20/06/2000
Cpf do paciente:  123456
```

```
% funcao que realiza a consulta de um paciente
consultarPaciente :-
    get_nome_paciente(Nome),
    get_paciente(Nome, Paciente).
```

```
% recupera um paciente dado o nome
get_paciente(Nome, Paciente) :-
    get_all_pacientes(Pacientes),
    check_paciente_exists(Pacientes, Nome, Paciente).
```

O fluxo para quando não encontra o paciente é o mesmo que visto anteriormente na alteração

Remover paciente

```
Digite o nome do paciente que deseja remover: |    Jorge

Paciente encontrado no sistema:

Nome: Jorge
Data de nascimento: 20/06/2000
Cpf do paciente:  123456

Paciente removido com sucesso!
```

```
% funcao que realiza a remoçao de um paciente
removerPaciente :-
    write('Digite o nome do paciente que deseja remover: '),
    ler_string_input(NomePaciente),
    get_paciente(NomePaciente, Paciente),
    Paciente \== '' ->
    get_all_pacientes(Pacientes),
    delete(Pacientes, Paciente, Result),
    write('\nPaciente removido com sucesso!\n'),
    escrever_arquivo(Result); menu.
```

Listar pacientes

```
Listando todos os pacientes...

Paciente : Jorge
Data de nascimento: 20/06/2000
CPF:  123456

Paciente : Hudson
Data de nascimento: 20/06/1990
CPF:  123456

Paciente : Maria
Data de nascimento: 20/04/2010
CPF:  123456
```

```
% listagem de todos os pacientes salvos no arquivo
listarPacientes :-
    write('\nListando todos os pacientes...\n'),
    get_all_pacientes(Pacientes),
    printaListPaciente(Pacientes).
```

Listar pacientes

```
% funcoes para printar uma lista de pacientes
printaListPaciente([]).

printaListPaciente([X|Y]) :-
    split_string(X, ",", "", Z),
    printaPaciente(Z),
    printaListPaciente(Y).

printaPaciente(List) :-
    length(List, 1), !;
    nl, write('Paciente : '),
    nth0(0, List, Nome),
    write(Nome), nl,
    nth0(1, List, DataNasc),
    nth0(2, List, Cpf),
    write('Data de nascimento: '),
    write(DataNasc), nl,
    write('CPF: '),
    write(Cpf), nl.
```

Diagnóstico

```
Informe os sintomas que está sentindo:
Por exemplo alguns sintomas sao: nausea,febre, hemorragia, calafrios, diarreia, vomito, dor generali
zada, tosse prolongada, fadiga, garganta inflamada, suor excessivo, hemorragia, dor nas constas, cor
iza, dor de cabeca, garganta inflamada, olhos amarelados, dor no peito, pouco apetito, formigamento,
 ulcera... etc .

Digite os seus sintomas, adicionando um por um com um ponto no final, entre aspas e apertando a tecl
a enter. Ao fim informar "parar." para finalizar:
```

Diagnóstico

```
|    'suor excessivo'.
|    'dor de cabeca'.
|    parar.

As seguintes doenças e suas probabilidades de ocorrência foram encontradas dado os sintomas informados:

 Meningite 61,7%

 Pneumonia 60,3%

 Dengue Classica 30,8%

 Rubeola 18,7%

O resultado do prótipo é apenas informativo e o paciente deve consultar um médico para obter um diagnóstico correto e preciso

Deseja verificar todos os sintomas de alguma doença? Se sim, digite o nome da doença entre aspas ' ', digite 0 caso contrario
|
```

Diagnóstico

```
Todos os sintomas da doença Dengue Classica podem ser encontrados a seguir:
Sintomas que você não informou:

 hemorragia

 dor generalizada

Sintomas que você informou:

 dor de cabeca
```

Diagnóstico

```
% funcao que recupera os sintomas do usuario e mostra as doencas
informarSintomas :-
    write('Informe os sintomas que está sentindo:'), nl,
    write('Por exemplo alguns sintomas sao: nausea,febre, hemorragia, calafrios, diarreia, vomito, dor generalizada, tosse prolongada, fadiga, gargant
a inflamada, suor excessivo, hemorragia, dor nas constas, coriza, dor de cabeca, garganta inflamada, olhos amarelados, dor no peito, pouco apetito, fo
rmigamento, ulcera... etc .'),
    write("\n\nDigite os seus sintomas, adicionando um por um com um ponto no final, entre aspas e apertando a tecla enter. Ao fim informar \"parar.\"
 para finalizar:\n"),
    add_read_list(ResultList, []),
    getDoencas(ResultList).
```

Diagnóstico

```
% recuperar as possiveis doenças do paciente dado seus sintomas
getDoencas(Sintomas) :-
    findall(A, doencaSintomasProb(A, Sintomas), Z),
    remove_duplicates(Z, Lista),
    length(Lista, Tamanho),
    (   (Tamanho = 1,
    write('\nFoi encontrado apenas uma doença relacionada com os sintomas informados, a doença pode ser vista a seguir:\n'),
    printaUmaDoenca(Lista),
    write('\nO resultado do prótipo é apenas informativo e que o paciente deve consultar um médico para obter um diagnóstico correto e preciso\n')
    );
    (write('\nAs seguintes doenças e suas probabilidades de ocorrência foram encontradas dado os sintomas informados:\n'), printaList(Lista),
     write('\nO resultado do prótipo é apenas informativo e o paciente deve consultar um médico para obter um diagnóstico correto e preciso\n'))),
    verificarSintomas(Sintomas).
```

Diagnóstico

```
% verificar sintomas adicionais dada uma doença
verificarSintomas(SintomasPaci) :-
    write('\nDeseja verificar todos os sintomas de alguma doença? Se sim, digite o nome da doença entre aspas e com um ponto no final, digite 0 caso c
ontrario\n'),
    read(Escolha),
    dif(Escolha, 0) ->
    (   doenca(Escolha, Sintomas)->
    write('\nTodos os sintomas da doença '),
    write(Escolha),
    write(' podem ser encontrados a seguir:\n'),
    printaListSintomas(Sintomas, SintomasPaci),
    verificarSintomas(SintomasPaci);
    write('\nNão foi encontrado nenhuma doença com esse nome, por favor, tente novamente!\n'),
    verificarSintomas(SintomasPaci)); !.
```

Diagnóstico

```
% mostrar pro usuario a lista de sintomas que informou/nao informou
printaListSintomas(SintomasDoenca, SintomasPaci) :-
    subtract(SintomasDoenca, SintomasPaci, Result),
    write('Sintomas que você não informou:\n'),
    printaList(Result),
    write('\nSintomas que você informou:\n'),
    intersection(SintomasDoenca, SintomasPaci, ResultInformado),
    printaList(ResultInformado).
```

Funções auxiliares

```
% recupera todos os pacientes do arquivo
get_all_pacientes(Pacientes) :-
    working_directory(CWD, CWD),
    PacienteFile = '/pacientes.txt',
    atom_concat(CWD, PacienteFile, Arquivo),
    open(Arquivo, read, Pfile),
    current_input(Stream),
    set_input(Pfile),
    ler_arquivo_pacientes(Pfile, Pacientes), !,
    close(Pfile),
    set_input(Stream).
```

```
% funcao auxiliar para verificar se um paciente existe
check_paciente_exists([X | Y], Nome, Paciente) :-
    split_string(X, ",", "", [NomeP, DataP, CpfP]),
    Nome == NomeP ->
    Paciente = X,
    write('\nPaciente encontrado no sistema:\n'),nl,
    write('Nome: '), write(NomeP), nl,
    write('Data de nascimento: '), write(DataP), nl,
    write('Cpf do paciente: '), write(CpfP), nl;
    check_paciente_exists(Y, Nome, Paciente).

check_paciente_exists([], _, '') :- write('\nPaciente não encontrado\n'), !.
```

Funções auxiliares

```
% funcao padrao para ler entradas do usuario sem necessidade do '.'
ler_string_input(Var) :-
    get_char(_),
    read_string(user_input, "\n","\t ", _, Var).
```

```
% funcao para printar uma lista
printaList([]).

printaList([X|Y]):-
  X == end_of_file, !;
  format('\n ~w', X), nl,
  printaList(Y).
```

```
% funcao auxiliar para escrever uma lista no arquivo
escrever_arquivo(Lista) :-
    working_directory(CWD, CWD),
    PacienteFile = '/pacientes.txt',
    atom_concat(CWD, PacienteFile, Arquivo),
    open(Arquivo, write, Pfile),
    write_file(Lista, Pfile),
    close(Pfile).
```

Funções auxiliares

```
% funcao auxiliar para escrever uma lista no arquivo
write_file([P | Pacientes], Out) :-
    length(Pacientes, 0), !;
    split_string(P, ",", "", [Nome, Data, Cpf]),
    write(Out,'\''),
    write(Out,Nome),
    write(Out,','),
    write(Out,Data),
    write(Out,','),
    write(Out,Cpf),
    write(Out,'\''),
    write(Out,'.'),
    write(Out,'\n'),
    write_file(Pacientes, Out).
```

```
% funcao auxiliar que le uma lista de entradas do usuario e salva em uma
% lista
add_read_list(Resultlist, Entrylist) :-
  read(Input),
  (  Input = parar
  -> reverse(Resultlist, Entrylist)
  ;  add_read_list(Resultlist, [Input|Entrylist])
  ).
```

Funções auxiliares

```
% funcao que realiza a leitura do arquivo de pacientes e salva em uma
% lista
ler_arquivo_pacientes(Stream,[]) :-
    at_end_of_stream(Stream).

ler_arquivo_pacientes(Stream,[X|L]) :-
    \+ at_end_of_stream(Stream),
    read(Stream,X),
    ler_arquivo_pacientes(Stream,L).
```

```
% remover elemento duplicados dada uma lista
remove_duplicates([],[]).

remove_duplicates([H],[H]).

remove_duplicates([H ,H| T], List) :-remove_duplicates( [H|T], List).

remove_duplicates([H,Y | T], [H|T1]):- Y \= H,remove_duplicates( [Y|T], T1).
```

Base

```
doenca('Tuberculose', ['hemorragia', 'dor no peito', 'perda de peso', 'pouco apetite', 'febre', 'calafrios']).

doenca('Dengue Hemorragica', ['hemorragia', 'vomito', 'dificulade respiratoria', 'tontura']).

doenca('Dengue Classica', ['hemorragia', 'dor de cabeca', 'dor generalizada']).

doenca('Pneumonia', ['dor nas costas', 'dor no peito', 'tosse prolongada', 'calafrios', 'fadiga', 'febre', 'suor excessivo', 'dificuldade respiratoria
']).

doenca('Gastrite', ['dor no estomago', 'queimacao no estomago', 'enjoo', 'pouco apetite', 'vomito']).

doenca('Catapora', ['manchas avermelhadas no corpo', 'fadiga', 'febre', 'garganta inflamada', 'pouco apetite']).

doenca('Leptospirose', ['dor generalizada', 'diarreia', 'nausea', 'vomito', 'calafrios', 'fadiga', 'febre']).

doenca('Rubeola', ['coriza', 'febre', 'manchas avermelhadas no corpo']).

doenca('Caxumba', ['dor no testiculo', 'fadiga', 'febre', 'pouco apetite', 'dor de cabeca']).

doenca('Bronquite', ['tosse prolongada', 'calafrios', 'fadiga', 'febre', 'dificuldade respiratoria', 'coriza']).

doenca('Sarampo', ['dor generalizada', 'tosse prolongada', 'fadiga', 'erupcoes cutaneas', 'febre', 'manchas avermelhadas no corpo']).
```

Base

```
doencaSintomasProb('Gastrite 74,7%', A):-
    member('dor no estomago', A);
    member('queimacao no estomago', A);
    member('enjoo', A);
    member('pouco apetite', A);
    member('vomito', A).


doencaSintomasProb('Catapora 63,8%', A):-
    member('manchas avermelhadas no corpo', A);
    member('fadiga', A);
    member('febre', A);
    member('garganta inflamada', A);
    member('pouco apetite', A).

doencaSintomasProb('Sarampo 63,4%', A):-
    member('dor generalizada', A);
    member('tosse prolongada', A);
    member('fadiga', A);
    member('febre', A);
    member('erupcoes cutaneas', A);
    member('manchas avermelhadas no corpo', A).
```

```
doencaSintomasProb('Meningite 61,7%', A):-
    member('dor generalizada', A);
    member('calafrios', A);
    member('manchas avermelhadas no corpo', A);
    member('erupcoes cutaneas', A);
    member('fadiga', A);
    member('febre', A);
    member('dor de cabeca', A);
    member('nausea', A);
    member('vomito', A).

doencaSintomasProb('Pneumonia 60,3%', A):-
    member('dor nas costas', A);
    member('dor no peito', A);
    member('tosse prolongada', A);
    member('calafrios', A);
    member('fadiga', A);
    member('febre', A);
    member('suor excessivo', A);
    member('dificuldade respiratoria',A).

doencaSintomasProb('Herpes 58,8%', A):-
    member('formigamento', A);
    member('erupcoes cutaneas', A);
    member('manchas avermelhadas no corpo', A);
    member('ulcera', A).
```

Testes unitários

```
%% Execução dos testes unitarios

:- begin_tests(main).
:- use_module(library(main)).

test(doenca):-
    doenca('Rubeola', ['coriza', 'febre', 'manchas avermelhadas no corpo']).

test(doenca):-
doenca('Sifilis', ['dor generalizada', 'fadiga', 'febre', 'pouco apetite', 'ulcera']).

test(doenca):-
doenca('Herpes', ['formigamento', 'erupcoes cutaneas', 'manchas avermelhadas no corpo', 'ulcera']).

test(doenca):-
doenca('Bronquite', ['tosse prolongada', 'calafrios', 'fadiga', 'febre', 'dificuldade respiratoria', 'coriza']).

test(doenca):-
doenca('Catapora', ['manchas avermelhadas no corpo', 'fadiga', 'febre', 'garganta inflamada', 'pouco apetite']).

test(doencaSintomaProb) :-
    doencaSintomasProb('Caxumba', ['fadiga', 'febre', 'pouco apetite']).

test(doencaSintomaProb) :-
    doencaSintomasProb('Bronquite', ['tosse prolongada', 'febre', 'coriza']).

test(doencaSintomaProb) :-
    doencaSintomasProb('Meningite', ['vomito', 'calafrios', 'fadiga']).

end_tests(main).
```