Primeiro trabalho prático de Linguagens de Programação

Sistema de Gestão Acadêmico
na linguagem *Prolog*

Leonardo Azalim de Oliveira
Marina Nunes Silva

Sob orientação do Professor Leonardo Vieira dos Santos Reis

Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora
Minas Gerais - Brasil
ERE 2021.3

Sumário

1 Especificação técnica do trabalho - Sistema de Gestão Acadêmico

O presente documento tem por objetivo apoiar o projeto desenvolvido durante o primeiro trabalho da disciplina DCC019 - Linguagens de Programação. O programa foi implementado na linguagem *Prolog* e todo o código fonte foi inserido em um arquivo que representa a Base de Conhecimento (em inglês, KB - *Knowledge Base*).

As citações dentro do texto estão organizadas de forma que as referências internas encontram-se entre parênteses e as externas entre colchetes. Todas elas são clicáveis de forma a encaminhar a navegação para o ponto exato do texto onde se encontram.

As caixas com código fonte foram feitas utilizando em suas representações a sintaxe descrita por David Matuszek[1].

1.1 Execução do programa

Abaixo seguem detalhes importantes para a correta execução do programa.

1.1.1 Pré-requisitos

Abaixo estão os pré-requisitos para o correto funcionamento do programa. Por favor tenha certeza de que os possui para correta execução do programa.

- Sistema operacional GNU/Linux ou Windows (10 ou superior)
- Interpretador SWI-Prolog[10] (versão 8.4.0 ou superior)

1.1.2 Como carregar o programa

O processo de carregamento do programa no sistema é bem simples: basta obter uma cópia do arquivo *main.pl* e então abrí-lo usando o interpretador, como por exemplo no comando abaixo:

```
swipl main.pl
```

1.2 Predicados, funções e bibliotecas do interpretador

Esta seção descreve, brevemente, alguns dos predicados provenientes das bilbiotecas *Prolog built-in* do interpretador SWI-Prolog que foram utilizados no programa.

1.2.1 Predicados *findall/3*, *bagof/3* e *setof/3*

Os três predicados fazem parte dos predicados de manipulação de bases da dados, sendo responsáveis por tentar unificar um objetivo de modo que os resultados sejam agrupados em formato de lista e possam ser filtradas quais partes de um certo objetivo serão armazenadas ou não.

Apesar de possuírem um processo de funcionamento bem similar e sintaxe idêntica, sutis diferenças ainda existem entre eles. Como por exemplo, ao tentar unificar em busca de um objetivo e não encontrar nenhum predicado que satisfaça esse objetivo, o predicado *findall/3* retorna uma lista vazia, já o predicado *bagof/3* retorna um status de falha (*fail*) - que pode ser interpretado como **false.** ou **no.** a depender das configurações do interpretador utilizado.

Mais informações sobre estes predicados podem ser encontradas no manual do interpretador[3].

1.2.2 Predicados *format/2* e *print/1*

Um outro predicado que foi utilizado pelo programa, foi o predicado *format/2*[5]. Ele foi escolhido (em alguns dos casos) em detrimento do predicado *print/1*[8] porque aquele é mais flexível quanto aos formatos de impressão de conteúdo, permitindo maior precisão e personalização da exibição dos dados da tela.

1.2.3 Predicados *intersection/3*, *append/3* e *subtract/3*

Outra tríade de predicados usados pelo programa implementado neste trabalho foram os predicados *intersection/3*, *append/3* e *subtract/3*, que são predicados específicos para a manipulação de estruturas em forma de lista. Como seus próprios nomes revelam, respectivamente, eles têm como função a de produzir uma intersecção, uma junção (ou contatenação) e uma subtração (diferença) entre duas listas quaisquer passadas como parâmetro de entrada, sendo o terceiro parâmtro responsável por armazenar o resultado das respectivas operações.

Mais informações sobre estes predicados também podem ser encontradas no manual do interpretador[4].

1.2.4 Predicados *append/1*, *writeq/3*, *put/1*, *nl/0* e *told/0*

Um grupo importante de predicados que também foi utilizado no programa foram os predicados *append/1*, *writeq/3*, *put/1*, *nl/0* e *told/0*. Eles foram utilizados na seção (2.7) onde ocorreu a manipulação direta dos dados contidos na KB.

Os predicados *append/1* e *writeq/3* foram responsáveis por gravar as modificações no arquivo da KB. Os predicados *put/1* e *nl/0* foram necessários para formatação dos novos predicados inseridos na KB para, desta forma, eles seguirem o padrão de sintaxe requerido pela linguagem e não interferirem nos demais predicados. Por fim, o predicado *told/0* foi utilizado para o fechamento do *stream* de dados que foi aberto pelos predicados *append/1* e *writeq/3* anteriormente.

Mais informações sobre estes predicados podem ser encontradas em suas respectivas seções do manual do interpretador[11, 9, 2].

1.2.5 Predicados *open/3*, *close/3* e *retract/3*

Os predicados *open/3*, *close/3* e *retract/3* foram utilizados nas operações de manipulação dos fatos da KB. Mais especificamente eles são parte dos predicados de remoção de fatos da KB(22).

Mais informações sobre estes predicados podem ser vistas no manual do interpretador[6, 7].

1.3 Decisões de projeto

Nesta seção estão descritas algumas das decisões de projeto tomadas durante a implementação do programa em Prolog.

Nos predicados *estudanteCurso_ComIRA/3*(17) e *estudanteCurso_ComNotaMaior/3*(16), para realizar o agrupamento dos resultados foi utilizado o predicado *setof/3* em detrimento do *findall/3* por conta das consultas onde se retornava uma lista vazia e o predicado ainda exibia **true.** como resposta. Foi também aplicado um operador de corte por conta da tripla unificação que estava ocorrendo com o parâmetro X e, assim, ocorria a geração de várias listas contendo tuplas como saída, o que não era o desejado naquele momento.

No predicado *getMatriz/1*(11), foi utilizado o predicado *format/2*, exatamente por conta dos motivos explicados na seção 1.2.2, já que o predicado *format/2* permite mais controle e customização do *stream* que será apresentado na saída padrão do usuário.

No predicado *faltaCursar/1*(14), houve a necessidade de se usar dois cortes dentro do predicado por conta da unificação em duplicidade de algumas das variáveis envolvidas na função daquele, que provavelmente ocorreu por conta da presença de mais de um objetivo ao mesmo tempo. Outra decisão tomada foi a de se utilizar o predicado *findall/3*(1.2.1) ao invés de dois predicados *bagof/3* por conta do processo de unificação que não ocorria corretamente ao se utilizar dois *bagof/3*.

No predicado *cursoContem/1*(18), também houve a necessidade de se aplicar um corte devido a presença de um duplo objetivo no operador *bagof/3*(1.2.1).

No predicado *adicionaMateriaCurso/3*(19), foi utilizado o predicado *writeq/1*(1.2.4) em detrimento ao predicado *write/1*, porque havia a necessidade de parsear também os caracteres especiais e, nesse ponto, o predicado *writeq/1* leva vantagem já que ele permite inserir os parênteses, colchetes e aspas onde é necessário.

Nos predicados *removeMateriaCurso/3*, *removeCurso/2* e *removeAluno/2*, que são os predicados utilizados para remover fatos da KB, foi utilizada a abordagem de "cancelamento forçado"do fato que se desejava apagar, o que não interfere nos predicados de adição de fatos na KB.

Outras duas decisões pertinentes ao prejeto e, de certa forma, relacionadas, foram as de representar as disciplinas por meio de seus códigos identificadores, porém não aplicar o mesmo para os alunos. A representação

de matérias por meio de seus códigos foi importante por conta das diversas consultas que fazem o uso desses, evitando assim erros de digitação. Porém, para os alunos, foi considerado que criar um outro identificador além do nome estaria fora do escopo deste projeto, porque acabaria criando uma *overhead* de dados sobre os alunos populando desnecessáriamente a KB, já que não seria necessário dispor deles no estágio em que se encontra o programa implementado.

1.4 Predicados e fatos da KB

Nesta seção são descritos todos os predicados e fatos, na ordem em que aparecem no código fonte, presentes na KB e que são necessários para o programa funcionar corretamente.

1.4.1 Fato *nomeCurso*

O fato representa a relação de todos os nomes dos cursos cadastrados na KB com suas respectivas siglas.

Parâmetros do fato:

- X: Representa a sigla do curso
- Y: Representa o nome do curso

Código Prolog:

```
1 nomeCurso(X, Y).
2 nomeCurso(si, "Sistemas de Informacao").
```

Listing 1: Representação do fato nomeCurso

1.4.2 Predicado *getCursos/1*

O predicado retorna a lista com todos os cursos cadastrados na KB. Ele foi implementado pois um uso dele seria o de consultar a lista de cursos já cadastrados para assim se evitar o cadastro duplicado de cursos.

Parâmetros de entrada do predicado:

- R: Receberá a lista de cursos presentes na KB, no formato [Nome, Código]

Estruturas utilizadas no predicado:

- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- nomeCurso/2: Representa o predicado que contém os cursos e suas respectivas siglas

Código Prolog:

```
1 getCursos(R) :- bagof([Y,X], nomeCurso(X, Y), R).
```

Listing 2: Código do predicado *getCursos/1*

1.4.3 Fato *cursou*

O fato representa todas as matérias que um determinado aluno cursou até o momento no curso ao qual está matriculado.

Parâmetros do fato:

- X: Representa o nome do aluno
- Y: Representa o nome da disciplina
- Z: Representa a nota do aluno na disciplina

Código Prolog:

```
cursou(X, Y, Z).
cursou("Fatima Martins Matias", "Algoritmos", 60).
```

Listing 3: Representação do fato *cursou*

1.4.4 Fato *nomeDisciplina*

O fato representa o relacionamento entre os nomes das disciplinas e seus respectivos códigos de identificação.

Parâmetros do fato:

- X: Representa o código de identificação da disciplina
- Y: Representa o nome da disciplina

Código Prolog:

```
nomeDisciplina(X, Y).
nomeDisciplina(dcc119, "Algoritmos").
```

Listing 4: Representação do fato *nomeDisciplina*

1.4.5 Fato *matriz*

O fato representa todas as disciplinas que estão incluídas em um determinado curso.

Parâmetros do fato:

- X: Representa o nome da disciplina
- Y: Representa a sigla do curso ao qual a matéria pertence

Código Prolog:

```
matriz(X, Y).
matriz("Calculo I", cc).
```

Listing 5: Representação do fato *matriz*

1.4.6 Fato *alunoDe*

O fato representa todos os alunos que estão matriculados em um determinado curso.

Parâmetros do fato:

- X: Representa o nome do aluno
- Y: Representa a sigla do curso ao qual o aluno está matriculado

Código Prolog:

```
alunoDe(X, Y).
alunoDe("Gustavo Melo Cavalcanti", cc).
```

Listing 6: Representação do fato *alunoDe*

1.4.7 Predicado *somaNotas/3*

O predicado retorna a soma das notas de todas as matérias cursadas por um determinado aluno. Ele foi implementado com o objetivo de apoiar o predicado *calculaIRA/2*, facilitando, assim, os cálculos envolvidos nele.

Parâmetros de entrada do predicado:

- $[X|Xs]$: Representa a lista de notas a serem somadas
- L1: Representa o comprimento da lista X (ou a quantidade de elementos dessa)
- N1: Receberá a soma dos valores contidos em X

Código Prolog:

```
somaNotas([], 0, 0). % se a lista for vazia, nao ha o que somar
somaNotas([X|Xs], L1, N1) :- somaNotas(Xs, L, N), N1 is N + X, L1 is L + 1.
```

Listing 7: Código do predicado *somaNotas/1*

1.4.8 Predicado *calculaIRA/2*

O predicado retorna o valor do IRA de um determinado aluno. Ele foi implementado com o objetivo de apoiar o predicado *estudanteCurso_ComIRA/3*, provendo uma faixa de corte nos resultados obtidos a partir desse.

Parâmetros de entrada do predicado:

- X: Representa o nome do aluno do qual se deseja calcular o IRA
- V: Receberá o valor do IRA calculado

Estruturas utilizadas no predicado:

- findall/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- cursou/3: Representa o predicado que contém a relação de qual aluno cursou qual disciplina(3)
- somaNotas/3: Representa o predicado que retorna a soma das notas de todas as matérias cursadas por um determinado aluno(7)

Código Prolog:

```
calculaIRA(X, V) :- findall(Z, cursou(X, Y, Z), R),
                    somaNotas(R, L, N), V is N / L.
```

Listing 8: Código do predicado *calculaIRA/2*

1.4.9 Predicado *subtraiListas/3*

O predicado retorna a diferença - onde os itens que pertencem a ambas são removidos - entre duas listas. Ele foi implementado com o objetivo de apoiar o predicado *faltaCursar/2*, fornecendo uma lista que é o resultado da remoção dos itens repetidos em ambas (no caso, como a lista de matérias cursadas sempre é menor ou igual a lista de matérias da matriz curricular, a diferença entre ambas basta).

Parâmetros de entrada do predicado:

- L1: Representa a lista de disciplinas da matriz curricular
- L2: Representa a lista de disciplinas já cursadas pelo aluno
- R: Receberá a lista com as matérias restantes a cursar

Estruturas utilizadas no predicado:

- intersection/3: Função de manipulação de listas presente nas bibliotecas do interpretador(1.2.3)
- append/3: Outra função de manipulação de listas presente nas bibliotecas do interpretador(1.2.3)
- subtract/3: Mais uma função de manipulação de listas presente nas bibliotecas do interpretador(1.2.3)

Código Prolog:

```
subtraiListas(L1, L2, R) :-
                            intersection(L1, L2, Intersec),
                            append(L1, L2, AllItems),
                            subtract(AllItems, Intersec, R).
```

Listing 9: Código do predicado *subtraiListas/3*

1.5 Lista de cursos

Esta seção lista todos os cursos previamente cadastrados na KB.

- Ciência da Computação (cc)
- Sistemas de informação (si)

1.6 Lista de alunos

Esta seção lista todos os alunos previamente cadastrados na KB.

1.6.1 Alunos do curso de Ciência da Computação:

- Fátima Martins Matias
- José Valverde Coimbra
- Jorge Marcos Ortega
- Davi Silva Araujo
- Gustavo Melo Cavalcanti
- Cauã Souza Fernandes
- Márcio Receputi Faria
- Arthur Correia Barbosa
- Vitor Alves Correia
- Enzo Castro Oliveira

1.6.2 Alunos do curso de Sistema Informação:

- Estefany Toscano Canário
- Nayla Belchior Salgado
- Clara da Granja Teodoro
- Sasha Nolasco Oliveira
- Davi Goncalves Oliveira
- Lucas Barros Azevedo
- Gabriel Correia Castro
- Kaike Dias Melo
- Marcelo Dias Souza
- Daniel Lima Fernandes

2 Código

Nesta seção estão descritos os predicados alvo da avaliação do programa, presentes no documento de requisitos do trabalho.

2.1 Predicado getHistorico/1

O predicado retorna todas as matérias e suas respectivas notas que fazem parte do histórico de um determinado aluno.

Parâmetros de entrada do predicado:

- X: Representa o nome do aluno do qual o histórico deverá ser consultado

Estruturas utilizadas no predicado:

- findall/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- cursou/3: Representa o predicado que contém as matérias cursadas pelo aluno(3)
- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
1 getHistorico(X) :- findall([Y, Z], cursou(X, Y, Z), R),
2                    alunoDe(X, A), nomeCurso(A, B),
3                    format('O historico do aluno(a) ~w,~nmatriculado(a) no curso de ~w contem
      as seguintes materias:~n~n~p',
4                    [X, B, R]).
```

Listing 10: Código do predicado *getHistorico/1*

2.2 Predicado getMatriz/1

O predicado retorna todas as matérias que fazem parte da matriz curricular de um determinado curso.

Parâmetros de entrada do predicado:

- Y: Representa a sigla do curso do qual se deseja retornar a matriz curricular

Estruturas utilizadas no predicado:

- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- matriz/2: Representa o predicado que contém as matérias cadastradas em um determinado curso(5)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
1 getMatriz(Y) :- bagof(X, matriz(X, Y), R),
2                 nomeCurso(Y, Z),
3                 format(
4                 'A matriz do curso de ~w contem as seguintes materias:~n~n~p',
5                 [Z, R]).
```

Listing 11: Código do predicado *getMatriz/1*

2.3 Predicado jahCursou/1

O predicado retorna os nomes e notas de todos os alunos que já cursaram uma determinada disciplina.

Parâmetros de entrada do predicado:

- Y: Representa o nome da disciplina da qual se deseja saber quem cursou

Estruturas utilizadas no predicado:

- setof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- cursou/3: Representa o fato que contém as matérias cursadas por um determinado aluno(3)
- nomeDisciplina/2: Representa o predicado que relaciona um código de disciplina ao nome dessa(4)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
jahCursou(Y) :- setof([X,Z], cursou(X, Y, Z), R),
                nomeDisciplina(Y, A),
                format('Os seguintes alunos ja cursaram a disciplina de~n~w~nco>
                [A, R]).
```

Listing 12: Código do predicado *jahCursou/1*

2.3.1 Predicado jahCursouNotaMaior/2

O predicado retorna os nomes e notas de todos os alunos que já cursaram uma determinada disciplina, com um filtro de nota.

Parâmetros de entrada do predicado:

- Y: Representa o nome da disciplina da qual se deseja saber quem cursou
- A: Representa o ponto de corte que se deseja aplicar à nota dos alunos

Estruturas utilizadas no predicado:

- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- cursou/3: Representa o fato que contém as matérias cursadas por um determinado aluno(3)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
jahCursouNotaMaior(Y, A) :- bagof([X, Z], (cursou(X, Y, Z), Z >= A), R),
                            format('Os seguintes alunos ja cursaram a disciplina de~n~w~ncom
    uma nota menor ou igual a ~w:~n~n~p',
                            [Y, A, R]).
```

Listing 13: Código do predicado *jahCursouNotaMaior/2*

2.3.2 Predicado jahCursouNotaMenor/2

Exatamente idêntico à seção acima(13), com a única diferença de que o ponto de corte A da nota é menor ou igual, ao invés de maior ou igual.

2.4 Predicado faltaCursar/1

O predicado retorna as disciplinas que ainda faltam ser cursadas pelo aluno.

Parâmetros de entrada do predicado:

- X: Representa o nome do aluno do qual se deseja saber quais disciplinas restam para cursar

Estruturas utilizadas no predicado:

- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- matriz/2: Representa o predicado que contém as matérias cadastradas em um determinado curso(5)
- findall/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- cursou/3: Representa o fato que contém as matérias cursadas por um determinado aluno(3)
- subtraiListas/3: Representa o predicado que retorna a diferença entre duas listas(9)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
1 faltaCursar(X) :-
2                   alunoDe(X, C),
3                   bagof(Y, matriz(Y, Z), R1),
4                   findall(A, cursou(X, A, B), R2),
5                   subtraiListas(R1, R2, R3), !,
6                   nomeCurso(B, Y),
7                   format('O aluno(a) ~w,~nmatriculado(a) no curso de ~w,~nainda precisa cursar
    as seguintes disciplinas:~n~n~p',
8                   [X, Y, R3]), !.
```

Listing 14: Código do predicado *faltaCursar/1*

2.5 Predicado estudanteCurso/1

O predicado retorna uma lista com todos os alunos que estão matriculados em um determinado curso.

Parâmetros de entrada do predicado:

- Y: Representa a sigla do curso do qual se deseja saber quem são os alunos

Estruturas utilizadas no predicado:

- setof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- print/1: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
1 estudanteCurso(Y) :- setof(X, alunoDe(X, Y), R), print(R).
```

Listing 15: Código do predicado *estudanteCurso/1*

2.5.1 Predicado estudanteCurso_ComNotaMaior/3

O predicado retorna todos os alunos de determinado curso que cursaram determinada disciplina e que obtiveram uma nota maior ou igual a determinado valor.

Parâmetros de entrada do predicado:

- B: Representa a sigla do curso a ser consultado
- Y: Representa o código da disciplina a ser consultada
- A: Representa o valor da nota a ser usada como corte nos resultados

Estruturas utilizadas no predicado:

- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- cursou/3: Representa o fato que contém as matérias cursadas por um determinado aluno(3)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- nomeDisciplina/2: Representa o predicado que relaciona um código de disciplina ao nome dessa(4)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
1 estudanteCurso_ComNotaMaior(B, Y, A) :-
2                                         bagof([X, Z],
3                                         (alunoDe(X, B),
4                                         cursou(X, Y, Z), Z >= A),
5                                         R),
6                                         nomeCurso(B, C), nomeDisciplina(Y, D),
7                                         format('Os alunos do curso de ~w e ~nque cursaram a
    disciplina de ~w~ncom a nota maior ou igual a ~w, sao os seguintes:~n~n~p',
8                                         [C, D, A, R]), !.
```

Listing 16: Código do predicado *estudanteCurso_ComNotaMaior/3*

2.5.2 Predicado estudanteCurso_ComIRA/2

O predicado retorna todos os alunos de determinado curso que cursaram determinada disciplina e que possuem um IRA maior ou igual a determinado valor. Os resultados são retornados na forma de lista de todos os alunos que são do curso B, fizeram a matéria Y e atendem ao requisito A de IRA.

Parâmetros de entrada do predicado:

- B: Representa a sigla do curso a ser consultado
- A: Representa o valor do IRA a ser usado como corte nos resultados

Estruturas utilizadas no predicado:

- setof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- calculaIRA/2: Retorna o valor do IRA de um determinado aluno(8)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)

- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
estudanteCurso_ComIRA(B, A) :-
                                setof([X, V],
                                (alunoDe(X, B),
                                (calculaIRA(X, V), V >= A)), R),
                                nomeCurso(B, C),
                                format('Os alunos do curso de ~w e~nque possuem o IRA maior ou
    igual a ~w, sao:~n~n~p',
                                [C, A, R]), !.
```

Listing 17: Código do predicado *estudanteCurso_ComIRA/2*

2.6 Predicado cursoContem/1

O predicado retorna todos os cursos que contém uma determinada disciplina.

Parâmetros de entrada do predicado:

- X: Representa o código da matéria a ser consultada

Estruturas utilizadas no predicado:

- bagof/3: Função de agrupamento de termos presente nas bibliotecas do interpretador(1.2.1)
- matriz/2: Representa o predicado que contém as matérias cadastradas em um determinado curso(5)
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- nomeDisciplina/2: Representa o predicado que relaciona um código de disciplina ao nome dessa(4)
- format/2: Função de impressão de *streams* na tela(1.2.2)

Código Prolog:

```
cursoContem(X) :- bagof([Z, A], (matriz(X, Z), nomeCurso(Z, A)), R),
                  nomeDisciplina(X, Y),
                  format('A materia de ~w esta contida nos cursos de:~n~n~p',
                  [Y, R]), !.
```

Listing 18: Código do predicado *cursoContem/1*

2.7 Predicados para manipulação da Base de Conhecimento

Esta seção contém os predicados relacionados com a manipulação direta do arquivo que contém a KB.

2.7.1 Predicado adicionaMateriaCurso/3

O predicado adiciona uma determinada matéria em um determinado curso. Ele foi criado para que fosse possível adicionar uma matéria nova em qualquer um dos cursos na KB.

Parâmetros de entrada do predicado:

- X: Representa o código da matéria que será criada
- Y: Representa o nome da matéria que será criada
- Z: Representa a sigla do curso ao qual a matéria será vinculada

Estruturas utilizadas no predicado:

- append/1: Adiciona um *term* ao final do arquivo alvo

- writeq/1: Escreve um *term* no arquivo, incluindo os caracteres especiais
- matriz/2: Representa o predicado que contém as matérias cadastradas em um determinado curso(5)
- put/1: Adiciona um caractere ao *stream*, neste caso ao final dele
- nl/0: Adiciona uma quebra de linha ao *stream*
- told/0: Fecha o *stream* aberto pelo *append*/1 e pelo *writeq*/1
- nomeDisciplina/2: Representa o predicado que relaciona um código de disciplina ao nome dessa(4)
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
adicionaMateriaCurso(X, Y, Z) :- append('main.pl'),
                                 writeq(
                                 matriz(X, Z)
                                 ),
                                 put('.'),
                                 nl,
                                 writeq(
                                 nomeDisciplina(X, Y)
                                 ),
                                 put('.'),
                                 nl, told,
                                 ["main.pl"].
```

Listing 19: Código do predicado *adicionaMateriaCurso/3*

2.7.2 Predicado adicionaCurso/2

O predicado adiciona um novo curso na KB. Ele foi criado com o objetivo de tornar possível cadastrar outros cursos além dos previamente cadastrados no código fonte.

Parâmetros de entrada do predicado:

- X: Representa o nome da matéria que será criada
- Y: Representa a sigla do curso a qual a matéria será vinculada

Estruturas utilizadas no predicado:

- append/1: Adiciona um *term* ao final do arquivo alvo
- writeq/1: Escreve um *term* no arquivo, incluindo os caracteres especiais
- nomeCurso/2: Representa o predicado que relaciona uma sigla a um curso(1)
- put/1: Adiciona um caractere ao *stream*, neste caso ao final dele
- nl/0: Adiciona uma quebra de linha ao *stream*
- told/0: Fecha o *stream* aberto pelo *append*/1 e pelo *writeq*/1
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
adicionaCurso(X, Y) :- append('main.pl'),
                       writeq(
                       nomeCurso(X, Y)
                       ),
                       put('.'),
                       nl, told,
                       ["main.pl"].
```

Listing 20: Código do predicado *adicionaCurso/2*

2.7.3 Predicado adicionaAluno/2

O predicado adiciona um novo aluno na KB. Ele foi criado com o objetivo de tornar possível cadastrar outros alunos além dos previamente cadastrados.

Parâmetros de entrada do predicado:

- X: Representa o nome do novo aluno a ser cadastrado na KB
- Y: Representa a sigla do curso ao qual o aluno será vinculado

Estruturas utilizadas no predicado:

- append/1: Adiciona um *term* ao final do arquivo alvo
- writeq/1: Escreve um *term* no arquivo, incluindo os caracteres especiais
- alunoDe/2: Representa o fato que relaciona um determinado aluno com o nome do curso no qual ele está matriculado(6)
- put/1: Adiciona um caractere ao *stream*, neste caso ao final dele
- nl/0: Adiciona uma quebra de linha ao *stream*
- told/0: Fecha o *stream* aberto pelo *append*/1 e pelo *writeq*/1
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
adicionaAluno(X, Y) :- append('main.pl'),
                       writeq(
                       alunoDe(X, Y)
                       ),
                       put('.'),
                       nl, told,
                       ["main.pl"].
```

Listing 21: Código do predicado *adicionaAluno/2*

2.7.4 Predicado removeMateriaCurso/3

O predicado remove uma determinada matéria de um determinado curso. Ele foi criado para que fosse possível remover uma matéria em qualquer um dos cursos na KB.

Parâmetros de entrada do predicado:

- X: Representa o código da matéria que será removida
- Y: Representa o nome da matéria que será removida
- Z: Representa a sigla do curso do qual a matéria será removida

Estruturas utilizadas no predicado:

- open/3: Abre um determinado arquivo e uma *stream* de dados(1.2.5)
- format/3: Função de impressão de *streams* no arquivo(1.2.2)
- retract/1: Realiza a remoção de um fato da KB(1.2.5)
- close/1: Fecha o arquivo e o *stream* de dados anteriormente abertos(1.2.5)
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
removeMateriaCurso(X, Y, Z) :- open('main-test.pl', append, S),
                               format(S,
                               ':- retract(matriz(~q,~q)).~n',
                               [X, Z]),
                               format(S,
                               ':- retract(nomeDisciplina(~q,~q)).~n',
                               [X, Y]),
                               close(S),
                               ["main-test.pl"].
```

Listing 22: Código do predicado *removeMateriaCurso/3*

2.7.5 Predicado removeCurso/2

O predicado remove um curso da KB. Ele foi criado com o objetivo de tornar possível remover cursos já cadastrados.

Parâmetros de entrada do predicado:

- X: Representa o nome do curso que será removido
- Y: Representa a sigla do curso que será removido.

Estruturas utilizadas no predicado:

- open/3: Abre um determinado arquivo e uma *stream* de dados(1.2.5)
- format/3: Função de impressão de *streams* no arquivo(1.2.2)
- retract/1: Realiza a remoção de um fato da KB(1.2.5)
- close/1: Fecha o arquivo e o *stream* de dados anteriormente abertos(1.2.5)
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
removeCurso(X, Y) :- open('main.pl', append, S),
                     format(S, ':- retract(nomeCurso(~q,~q)).~n', [X, Y]),
                     close(S),
                     ["main.pl"].
```

Listing 23: Código do predicado *removeCurso/2*

2.7.6 Predicado removeAluno/2

O predicado remove um aluno da KB. Ele foi criado com o objetivo de tornar possível remover alunos cadastrados da KB.

Parâmetros de entrada do predicado:

- X: Representa o nome do aluno a ser removido da KB
- Y: Representa a sigla do curso do qual o aluno será removido

Estruturas utilizadas no predicado:

- open/3: Abre um determinado arquivo e uma *stream* de dados(1.2.5)
- format/3: Função de impressão de *streams* no arquivo(1.2.2)
- retract/1: Realiza a remoção de um fato da KB(1.2.5)

- close/1: Fecha o arquivo e o *stream* de dados anteriormente abertos(1.2.5)
- ["*nome_arquivo.pl*"]: Recarrega a KB que contém as alterações no interpretador Prolog

Código Prolog:

```
removeAluno(X, Y) :- open('main.pl', append, S),
                     format(S, ':- retract(alunoDe(~q,~q)).~n', [X, Y]),
                     close(S),
                     ["main.pl"].
```

Listing 24: Código do predicado *removeAluno/2*

3 Exemplos de funcionamento

Esta seção contém exemplos de funcionamento do programa e de alguns de seus predicados.

3.1 Consultar histórico

Para consultar o histórico de um determinado aluno, basta inserir o nome deste no predicado *getHistorico/1* (10).

```
?- getHistorico("José Valverde Coimbra").
O histórico do aluno(a) José Valverde Coimbra,
matriculado(a) no curso de Ciência da Computação contém as seguintes matérias:

[[dcc119,70],[qui126,60],[ice001,90],[mat154,62],[dcc120,60],[qui126,79],[fis122
,61],[qui125,68],[mat156,70],[est028,80],[fis077,72],[dcc013,76],[fis073,67],[dc
c107,76],[qui126,70],[dcc179,85]]
true.
```

Figura 1: Consulta mostrando um exemplo de histórico de um aluno

3.2 Consultar matriz de curso

Para consultar a matriz de um determinado curso, basta inserir a sigla dele no predicado *getMatriz/1* (11).

```
?- getMatriz(cc).
A matriz do curso de Ciência da Computação contém as seguintes matérias:

[mat155,mat154,qui125,fis122,dcc119,dcc120,qui126,ice001,mat156,est028,fis073,fi
s077,dcc013,dcc107,qui162,dcc179,mat157,dcc160,mat143,dcc059,dcc025,dcc122,mat15
8,mat029,dcc008,dcc012,dcc117,dcc070,dcc065,dcc014,est029,dcc060,dcc061,dcc062,d
cc163,dcc063,dcc174,eaddcc044,dcc042,dcc064,dcc123,dcc055,dcc001,dcc075,dcc110,d
cc045,dcc019]
true.
```

Figura 2: Consulta mostrando um exemplo da matriz de um curso

3.3 Consultar quem já cursou uma disciplina

Para consultar os alunos que já cursaram determinada disciplina, basta inserir a sigla da disciplina no predicado *jahCursou/1* (12).

```
?- jahCursou(dcc119).
Os seguintes alunos já cursaram a disciplina de
Algoritmos
com as respectivas notas:

[["Arthur Correia Barbosa",84],["Cauã Souza Fernandes",70],["Clara da Granja Teo
doro",73],["Daniel Lima Fernandes",89],["Davi Goncalves Oliveira",77],["Davi Sil
va Araujo",70],["Enzo Castro Oliveira",69],["Estefany Toscano Canário",67],["Fát
ima Martins Matias",60],["Gabriel Correia Castro",62],["Gustavo Melo Cavalcanti"
,80],["Jorge Marcos Ortega",88],["José Valverde Coimbra",70],["Kaike Dias Melo",
63],["Lucas Barros Azevedo",70],["Marcelo Dias Souza",63],["Márcio Receputi Fari
a",92],["Nayla Belchior Salgado",73],["Sasha Nolasco Oliveira",90],["Vitor Alves
 Correia",89]]
true .
```

Figura 3: Consulta mostrando um exemplo de alunos que já cursaram uma determinada disciplina

Para consultar os alunos que já cursaram determinada disciplina com nota maior do que a nota de corte, basta inserir a sigla da disciplina e a nota de corte no predicado *jahCursouNotaMaior/2*(13).

```
?- jahCursouNotaMaior(dcc119, 60).
Os seguintes alunos ja cursaram a disciplina de
Algoritmos
com uma nota maior ou igual a 60:

[["Fátima Martins Matias",60],["José Valverde Coimbra",70],["Jorge Marcos Ortega
",88],["Davi Silva Araujo",70],["Gustavo Melo Cavalcanti",80],["Cauã Souza Ferna
ndes",70],["Márcio Receputi Faria",92],["Arthur Correia Barbosa",84],["Vitor Alv
es Correia",89],["Enzo Castro Oliveira",69],["Estefany Toscano Canário",67],["Na
yla Belchior Salgado",73],["Clara da Granja Teodoro",73],["Sasha Nolasco Oliveir
a",90],["Davi Goncalves Oliveira",77],["Lucas Barros Azevedo",70],["Gabriel Corr
eia Castro",62],["Kaike Dias Melo",63],["Marcelo Dias Souza",63],["Daniel Lima F
ernandes",89]]
true.
```

Figura 4: Consulta mostrando um exemplo de alunos que já cursaram uma determinada disciplina com nota maior do que a nota de corte

Para consultar os alunos que já cursaram determinada disciplina com nota menor do que a nota de corte, basta inserir a sigla da disciplina e a nota de corte no predicado *jahCursouNotaMenor/2*(2.3.2).

```
?- jahCursouNotaMenor(dcc119, 80).
Os seguintes alunos ja cursaram a disciplina de
Algoritmos
com uma nota menor ou igual a 80:

[["Fátima Martins Matias",60],["José Valverde Coimbra",70],["Davi Silva Araujo",
70],["Gustavo Melo Cavalcanti",80],["Cauã Souza Fernandes",70],["Enzo Castro Oli
veira",69],["Estefany Toscano Canário",67],["Nayla Belchior Salgado",73],["Clara
 da Granja Teodoro",73],["Davi Goncalves Oliveira",77],["Lucas Barros Azevedo",7
0],["Gabriel Correia Castro",62],["Kaike Dias Melo",63],["Marcelo Dias Souza",63
]]
true.
```

Figura 5: Consulta mostrando um exemplo de alunos que já cursaram uma determinada disciplina com nota menor do que a nota de corte

3.4 Consultar quais matérias faltam cursar

Para consultar quais matérias determinado aluno falta cursar, basta inserir o nome do aluno no predicado *faltaCursar/1*(14).

```
?- faltaCursar("Fátima Martins Matias").
O aluno(a) Fátima Martins Matias,
matriculado(a) no curso de Ciência da Computação,
ainda precisa cursar as seguintes disciplinas:

[mat155,mat156,est028,fis073,fis077,dcc013,dcc107,qui162,dcc179,mat157,dcc160,ma
t143,dcc059,dcc025,dcc122,mat158,mat029,dcc008,dcc012,dcc117,dcc070,dcc065,dcc01
4,est029,dcc060,dcc061,dcc062,dcc163,dcc063,dcc174,eaddcc044,dcc042,dcc064,dcc12
3,dcc055,dcc001,dcc075,dcc110,dcc045,dcc019]
true.
```

Figura 6: Consulta mostrando um exemplo de quais matérias faltam para determinado aluno cursar

3.5 Consultar quem são os alunos de um curso

Para consultar quais são os alunos de um determinado curso, basta inserir a sigla do curso no predicado *estudanteCurso/1*(15).

```
?- estudanteCurso(si).
["Clara da Granja Teodoro","Daniel Lima Fernandes","Davi Goncalves Oliveira","Es
tefany Toscano Canário","Gabriel Correia Castro","Kaike Dias Melo","Lucas Barros
 Azevedo","Marcelo Dias Souza","Nayla Belchior Salgado","Sasha Nolasco Oliveira"
]
true.
```

Figura 7: Consulta mostrando um exemplo de todos os alunos de um curso

Para consultar quais são os alunos de um determinado curso que possuem nota maior que a nota de corte em uma disciplina, basta inserir a sigla do curso, o código da discplina e a nota de corte no predicado *estudanteCurso_ComNotaMaior/3*(16).

```
?- estudanteCurso_ComNotaMaior(si, dcc119, 60).
Os alunos do curso de Sistemas de Informação e
que cursaram a disciplina de Algoritmos
com a nota maior ou igual a 60, são os seguintes:

[["Clara da Granja Teodoro",73],["Daniel Lima Fernandes",89],["Davi Goncalves Ol
iveira",77],["Estefany Toscano Canário",67],["Gabriel Correia Castro",62],["Kaik
e Dias Melo",63],["Lucas Barros Azevedo",70],["Marcelo Dias Souza",63],["Nayla B
elchior Salgado",73],["Sasha Nolasco Oliveira",90]]
true.
```

Figura 8: Consulta mostrando um exemplo de todos os alunos de um curso com nota de corte em determinada disciplina

Para consultar quais são os alunos de um determinado curso que possuem determinado IRA maior que o valor de corte, basta inserir a sigla do curso e o valor de corte no predicado *estudanteCurso_ComNotaIRA/3*(17).

```
?- estudanteCurso_ComIRA(si, 80).
Os alunos do curso de Sistemas de Informação e
que possuem o IRA maior ou igual a 80, são:

[["Daniel Lima Fernandes",113.41176470588235],["Davi Goncalves Oliveira",82.5],[
"Sasha Nolasco Oliveira",89.66666666666667]]
true.
```

Figura 9: Consulta mostrando um exemplo de todos os alunos de um curso com determinado IRA

3.6 Consultar quais são os cursos que contém determinada disciplina

Para consultar quais são os cursos que contém determinada disciplina, basta inserir o código da disciplina no predicado *cursoContem/3*(18).

```
?- cursoContem(dcc120).
A matéria de Laboratório de Programação está contida nos cursos de:

[[cc,"Ciência da Computação"],[si,"Sistemas de Informação"]]
true.

?- cursoContem(dcc133).
A matéria de Introdução à Sistemas de Informação está contida nos cursos de:

[[si,"Sistemas de Informação"]]
true.

?- cursoContem(dcc065).
A matéria de Computação Gráfica está contida nos cursos de:

[[cc,"Ciência da Computação"]]
true.
```

Figura 10: Consulta mostrando um exemplo dos cursos que contém determinada disciplina

3.7 Manipular os dados da base

Para adicionar um curso, basta inserir a sigla e o nome do curso no predicado *adicionaCurso/2*(20). E, para remover um curso da KB, basta inserir a sigla e o nome do curso no predicado *removeCurso/2*(23).

```
?- getCursos(R).
R = [["Ciência da Computação", cc], ["Sistemas de Informação", si]].

?- adicionaCurso(qui, "Química").
true.

?- getCursos(R).
R = [["Ciência da Computação", cc], ["Sistemas de Informação", si], ["Química",
qui]].

?- removeCurso(qui, "Química").
true.

?- getCursos(R).
R = [["Ciência da Computação", cc], ["Sistemas de Informação", si]].

?- _
```

Figura 11: Consultas de inserção e remoção de fatos na KB

4 Referências

[1] David Matuszek. *A Concise Introduction to Prolog*. 2012. URL: https://www.cis.upenn.edu/~matuszek/Concise%20Guides/Concise%20Prolog.html#syntax.

[2] SWIProlog. *Edinburgh-style I/O*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=edinburghIO.

[3] SWIProlog. *Finding all Solutions to a Goal*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=allsolutions.

[4] SWIProlog. *Finding all Solutions to a Goal*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=lists.

[5] SWIProlog. *Format*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=format-predicates.

[6] SWIProlog. *ISO Input and Output Streams*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=isoIO.

[7] SWIProlog. *Managing (dynamic) predicates*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=dynpreds.

[8] SWIProlog. *Predicate print/1*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/doc_for?object=print/1.

[9] SWIProlog. *Primitive character I/O*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=chario.

[10] SWIProlog. *SWI-Prolog website homepage*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/.

[11] SWIProlog. *Term reading and writing*. Acesso em 02 Dezembro de 2021. URL: https://www.swi-prolog.org/pldoc/man?section=termrw.