Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
Linguagem de Programação

Carteira de Ações

1º Trabalho Prático em Prolog

Bruno de Oliveira Jucá - MAT 201965013AC

Rômulo Chrispim de Mello - MAT 201935038

Professor: Leonardo Vieira dos Santos Reis

Relatório do primeiro trabalho da disciplina DCC019 - Linguagem de Programação.

Juiz de Fora

Fevereiro de 2022

1 Introdução

O presente relatório tem como objetivo descrever o programa desenvolvido em Prolog para representar uma carteira de ações. Serão apresentadas as principais decisões de projeto e as estratégias utilizadas para resolução dos problemas apresentados.

O trabalho está estruturado da seguinte maneira: a Seção 2 descreve como o problema foi modelado; a Seção 3 descreve como a base de dados utilizada foi gerada e como os dados são persistidos pelo programa; a Seção 4 apresenta como cada funcionalidade foi implementada, descrevendo os predicados utilizados em cada uma delas; a Seção 5 apresenta um resumo de como utilizar as funções e interagir com o programa; por fim, a Seção 6 traz as conclusões e as considerações finais do trabalho.

2 Modelagem do Problema

Para modelar o problema, essencialmente considera-se que o usuário pode cadastrar operações que representam negociações na bolsa de valores. O usuário informa a data, o *ticket*, o preço da ação, e a quantidade comprada ou vendida. Isto representa uma operação, que quando registrada, além dos dados informados, também calcula as taxas da negociação e o valor total da transação.

A operação é registrada como compra ou venda, por meio de um dos campos do fato definido. No caso de venda os valores da quantidade e do custo total são representados internamente com valores negativos para facilitar a visualização e cálculos futuros.

A Seção 3 decorre mais especificamente sobre as representações internas dos dados.

3 Base de Dados e Persistência dos Dados

Para gerar a base de dados utilizada, foi criado um *script* em Python que gera entradas para o programa.

Esse *script* define operações aleatórias, mas com alguns controles. As operações foram geradas entre 01/01/2020 e 27/11/2022. A ação escolhida é uma dentre as ações presentes numa lista inicial e a operação pode ser de compra ou de venda. O *script* garante que as operações de venda só possam ser realizadas se houver quantidade suficiente daquela ação em posse para que possa ser vendida. Os preços foram definidos para variar entre 30 e 50 reais e a quantidade entre 1 e 500. Este código está disponível dentre os arquivos enviados.

Um exemplo de operações de compra e venda são:

```
compra(10, 11, 2022, 'B3SA3', 32.0, 12).
venda(25, 10, 2022, 'IRBR3', 42.17, 144).
```

Internamente os dados são representados pelos fatos `operacao/7`. As operações exemplificadas acima são registradas por meio dos predicados `compra/6` e `venda/6` que serão especificados na Subseção 4.1:

```
operacao(date(10,11,2022), 'B3SA3', 'COMPRA', 32.0, 12, 0.11, 384.11).
operacao(date(25,10,2022), 'IRBR3', 'VENDA', 42.17, -144, 1.82,
-6070.66).
```

Para persistir esses dados, utilizou-se a biblioteca `persistency`. Na inicialização do programa, o predicado `init/0` é unificado e cuida de definir o caminho do arquivo de persistência e de carregá-lo.

Por meio da definição de `operacao/7` como um fato que deve ser persistido pela biblioteca.

```
:- persistent operacao(date:any, ticket:any, tipo:any, preco_unid:any,
    quantidade:any, taxas:any, valor_total:any).
```

Os dados tiveram tipo `any` definidos para todos os campos para facilitar o desenvolvimento.

A biblioteca cuida de fazer `assert` para os fatos e guardá-los no arquivo especificado com o uso do predicado `assert_operacao/7`.

O programa já contém as operações registradas no arquivo 'operacoes-gen.db'. Além disso, há um arquivo 'operacoes-especs.db' que contém as operações que serviram de exemplo nas especificações do trabalho. o código pode ser alterado para abrir este arquivo e verificar que os valores obtidos nas tabelas geradas pelas funcionalidades correspondem aos valores especificados.

4 Desenvolvimento das Funcionalidades

4.1 Funcionalidade Cadastro de Operações de Compra e Venda de Ações

Os predicados `compra/6` e `venda/6` registram respectivamente as compras e as vendas de ações do usuário. São estruturalmente semelhantes e calculam a taxa e o total da compra antes de unificar com `assert_operacao/7` para registrar o fato e colocá-lo no banco de dados.

A taxa `X` foi calculada de forma que tivesse apenas 2 casas decimais como pedido através da fórmula `floor(10^2*(0.00025 * N * P + 0.00005 * N * P))/10^2`.

O valor da operação `V` foi calculado como `N * P + X`.

4.2 Funcionalidade Carteira de Ações

Para exibir a carteira de ações do usuário foi definido o predicado `carteira/0`. Este predicado começa unificando em uma lista `Ts` todos os *tickets* existentes. Isto é feito por meio do `setof/3`, que lista todos os *tickets* (T) de forma ordenada e elimina repetições. Com a lista de *tickets* (Ts), é utiliza-se o predicado `maplist/2` para aplicar o predicado `calcula_acao/1` para cada item da lista.

Por sua vez, o predicado `calcula_acao/1`, unifica com um dado *ticket* (T). Aplica-se o predicado `bagof/3` para listar todas as todas as operações daquele *ticket* numa lista (Os). Essa lista é de um tipo composto separado por hifens para que os dados possam ser processados pelo `foldl/4` e passados adiante

O `foldl/4` é aplicado sobre a lista `Os` tendo o predicado `sum_carteira/3` como objetivo de unificação. Desse modo, o dado inicial que foi definido como `T-_-0-0-0-0` unifica e vai sendo acumulado até gerar o resultado `S`.

O predicado `sum_carteira/3` possui duas definições. Um que unifica quando a operação é de compra e outro quando é de venda, para tratar esses dois casos de forma diferente. As duas definições tratam de acumular o valor total das ações e a quantidade delas, entretanto a de compra vai atualizando o preço médio da ação e a de venda não. Além disso, a de venda faz uma verificação na quantidade da ação após a venda para ver se a posição foi zerada. Se sim, volta o preço médio e o valor total para 0, se não, apenas repassa o preço médio atual e calcula o valor total normalmente. O valor total precisa ser zerado neste caso para não considerar lucros e prejuízos das vendas integrais de ações.

4.3 Funcionalidade Lucro / Prejuízo no Ano

A funcionalidade de lucro ou prejuízo funciona através do predicado `lucro/1`, que recebe o ano de cálculo desejado como parâmetro. Este predicado funciona de maneira muito similar ao predicado `carteira/0`, com apenas algumas alterações. O predicado `setof/3` continua sendo usado para gerar uma lista ordenada e sem repetição dos *tickets* (Ts). Em vez de aplicar o predicado `maplist/3` foi aplicado o predicado `foldl/4`, para unificar o valor total do lucro de todas as ações. Pode parecer estranho o primeiro argumento ser `calcula_lucro(Y)`, com um argumento já incluso, mas isso funciona pois a chamada do obejetivo faz um "*append*"dos argumentos, e então é possível definir argumentos fixos no começo, como é necessário fazer para filtrar o lucro pelo ano.

O predicado `calcula_lucro/4` lista as operações do ano especificado de cada *ticket*. Usa o `foldl/4` da mesma forma que `calcula_acao/1`, e unifica os itens da lista `Os` com o predicado `sum_lucro_carteira/3`. Este predicado também trabalha em cima de um argumento composto e vai acumulando o resultado, a diferença aqui encontra-se na definição

```
?- operacoes('ITSA4', 'VENDA', 2020).

|    DATA    | TICKET | OPERAÇÃO | PREÇO UNITÁRIO | QUANTIDADE | TAXAS | CUSTO TOTAL |
|------------------------------------------------------------------------------------|
| 07/04/2020 | ITSA4  |  VENDA   |           9.10 |       -100 |  0.27 |     -909.73 |

true.
```

Figura 1: Exemplo do uso do predicado operacoes.

feita para unificar com a venda, que define e acumula o lucro de cada venda. O lucro de uma venda parcial é definido como o valor ganho na venda menos o preço médio multiplicado pela quantidade de ações que foram vendidas. No código essa conta se encontra em outra ordem por conta dos sinais negativos utilizados na representação das operações.

Os lucros de cada ação são, então impressos, e o lucro total que foi sendo acumulado é unificado e exibido também.

4.4 Funcionalidade Operações Realizadas por Ação

O predicado `operacoes/0` lista todas as operações realizadas. O predicado `format/2` foi utilizado para imprimir os resultados de maneira formatada.

O predicado `forall/2` foi usado para imprimir todas as operações já registradas executando o `format/2` para todas elas e exibindo as informações.

Alternativamente a `operacoes/0`, foi definido `operacoes/3`, que unifica o *ticket* (T), o tipo da operação (Z) e o ano (Y). Desse modo, o usuário pode filtrar as operações de acordo com esses três parâmetros. Para isso, basta tentar unificar o predicado com os três parâmetros. Caso algum deles seja irrelevante para a busca basta passar '_' como algum desses parâmetros.

5 Resumo de Funcionalidades e Interação com Usuário

`compra(D, M, Y, T, P, N).` - cadastra uma nova compra no dia D do mês M do ano Y. T é o *ticket*, P é o preço unitário e N a quantidade.

`venda(D, M, Y, T, P, N).` - cadastra uma nova venda no dia D do mês M do ano Y. T é o *ticket*, P é o preço unitário e N a quantidade.

`operacoes.` - lista todas as operações realizadas.

`operacoes(T, Z, Y).` - lista as operações unificando com o *ticket* T, o tipo de operação Z e o ano Y.

`carteira.` - visualiza quais títulos de ações estão disponíveis naquele instante na sua carteira.

`lucro(Y).` - verifica o lucro obtido no ano Y.

6 Conclusões e considerações finais

Dados os resultados obtidos neste relatório, acreditamos ter atingido o objetivo final do trabalho, de desenvolver um sistema de controle de carteira de ações, que permitisse realizar transações, visualizar lucros e prejuízos obtidos ao longo das operações, utilizando um paradigma de Programação em Lógica Matemática, o Prolog.

Vale ressaltar que o sistema desenvolvido não está otimizado para ponto flutuante. Alguns cálculos podem sofrer pequenas variações por conta da precisão das casas decimais nas presentes bibliotecas deste trabalho. Ainda assim, o sistema é capaz de exemplificar uma carteira de ações hipotética para o propósito do trabalho.

Outro ponto a destacar diz respeito à ordenação das operações, pois para a lógica funcionar da forma correta, a ordem cronológica das operações deve ser respeitada. O *script* em Python que foi anexado ao projeto nos garante que as operações são anexadas ao banco de dados de forma ordenada.