Representação interna de polinómios

De modo a ser mais fácil trabalhar com polinómios, decidimos utilizar um conjunto de *nested structures*. Dividimos todos os elementos constituíntes de um polínomio por estruturas, e, por ordem organizacional, contruímos uma estrutura complexa que contempla todas as partes integrantes de um polinómio

Polinómio

O polinómio é uma lista de monómios

```
Polynomial = [Monomial]
```

Monómio

O monómio é um tuplo, sendo que o seu primeiro elemento é o coeficiente de multiplicação, e o segundo as variáveis constituíntes do mesmo

```
Monomial = (Float, Variable)
```

Variáveis

As variáveis são representadas por uma lista de termos, ou seja, uma lista de todas as variáveis que constituem o monómio

```
Variables = [Term]
```

Termo

O termo é um tuplo, consituído pela variável de multiplicação e pelo expoente ao qual a mesma é elevada

```
Term = (Char, Float)
```

A estrutura final de um polinómio, seguindo a construção que decidimos implementar é, portanto, a seguinte :

```
Polynomial = [Coeficiente, [(Variável, Expoente),(Variável, Expoente)]]
```

Funcionalidades

Passagem de estrutura a string

O trabalho contempla a passagem do Output de cada função para string, de modo a que o resultado seja de mais facil compreensão. Para isso as funções principais recorrem á função `polynomialToString` para realizar a conversão, que por sua vez vai recorrer ás funções auxiliares `monomialToString` e `variablesToString`.

a)

A alínea a) pedia-nos a normalização de um polinómio, o que é realizado com recurso á função `normalizar`, para a qual usamos um conjunto de funções auxiliares:

normalizePolynomial

Esta, recebe um polinómio e vai aplicar uma série de funções auxiliares que criamos de maneira a fragmentar o processo da normalização. De maneira recursiva, monómio a monómio vão ser aplicadas as seguintes funções:

addMonomialAux

Caso dois monómios do polinómio tenham as mesmas variáveis elevadas ao mesmo expoente vai somar os monómios;

removeNullMonomials

Monómios iguais a zero vão ser retirados do polinómio;

simplifyPolynomials

Monómios com variáveis de expoente 0 vão ser simplificados.

b)

A alínea b) pedia-nos que fosse feita a adição de dois polinómios, o que é realizado pela função `adicionar`; para tal, com a nossa representação interna de polinómios, isto foi possível com apenas a concatenção dos dois polinómios, que são listas de monómios, seguida pela sua normalização com recurso á função anterior `normalizePolynomial`.

c)

A alínea c) pedia-nos o produto de dois polinómios, o que é realizado com recurso á função `multiplicar`, para a qual usamos um conjunto de funções auxiliares:

multMonomial

Esta função recebe dois monómios, multiplica os seus coeficientes, e junta as duas variáveis e seus expoentes

multMonomialAux

Esta função recebe um monómio e um polinómio e multiplica o mesmo por todos os monómios constituintes do polinómio. Esta, dá uso à recurção da própria função, e vai passando por todos os monómios, aplicando a função anterios aos mesmos

multPolynomial

Esta função recebe dois polinómios e multiplica um pelo outro. dá também uso à recurção, e vai fazendo o produto de cada monómio do primeiro polinómio com cada monómio do segundo

d)

A alínea d) pedia-nos que fosse calculada a derivada de um polinómio, o que é realizado pela função `derivar`, que vai pedir uma variável e o polinómio. Para esta função existe um conjunto de funções auxiliares:

derivateMonomial

Recebe um caracter, um monómio e um tipo Variables. Vai derivar o monómio em função ao caracter dado.

derivateMonomialAux

Verifica se o tipo Variabels tem algum Term com o caracter fornecido, retornando True se tiver, caso contrário retorna False. O propósito desta função é que, no caso do monómio não tiver o caracter fornecido enquanto variável, a derivada deste vai ser igual a 0.

derivateVariabels

Recebe um tipo Variabels e um caracter, retorna Variabels derivado em função ao caracter dado.

derivatePolynomial

Recebe um caracter e um poliómio, retornando também um polinómio, Usa a função auxiliar `derivateMonomial` para derivar recursivamentes os monómios do polinómio em função ao caracter dado.

Exemplos de utlização

Para todas as operações, os polinómios devem ser inseridos no formato da representação interna que implementámos. No Output das funções monómios vão ser representados da seguinte forma: - O coeficiente vai ser seguido pela parte literal, separados por `*`. - A parte literal vai ser representada pela variável seguido do seu expoente, separados por `^`. - O conjunto das várias variáveis e os seus expoentes também vão ser separadas por `*`

Na representação dos polinómios, caso um monómio tiver coeficiente negativo, este vai ser apresentado no seu valor absoluto e o monómio vai ser antecedido pelo sinal da subtração, em oposição do sinal de adição que vai ocorrer normalmente.

Normalizar

```
 Input: normalizar [(2,[('x',2), ('y', 3)]), (3,[('x',1)]), (4,[('y',1)]), (-5,[('z',1)]), (6,[]), (7,[
Output: "7.0*x^4.0*y^5.0*z^6.0 + 6.0 - 5.0*z^1.0 + 4.0*y^1.0 + 3.0*x^1.0 + 2.0*y^3.0*x^2.0"
```

Adicionar

```
 Input: adicionar [(2,[('x',2), ('y',3)])] [(3,[]),(5,[('x',2), ('y',3)])]
Output: "3.0 + 7.0*x^2.0*y^3.0"
```

Multiplicar

```
 Input: multiplicar [(2,[('x',2), ('y',3)])] [(3,[]),(5,[('x',2), ('y',3)])]
Output: "10.0*x^2.0*y^3.0*x^2.0*y^3.0 + 6.0*x^2.0*y^3.0"
```

Derivar

```
 Input: derivar 'x' [(2,[('x',2), ('y', 3)]), (3,[('x',1)]), (4,[('y',1)]), (-5,[('z',1)]), (6,[]), (7,
Output: "28.0*x^3.0*y^5.0*z^6.0 + 3.0 + 4.0*x^1.0*y^3.0"
```

Membros

- Bernardo Campos - up202006056
- José Sousa - up202006141