Capítulo 9

Abstração de dados

1. Suponha que desejava criar o tipo *racional*. Um número racional é qualquer número que possa ser expresso como o quociente de dois inteiros: o numerador (um inteiro positivo, negativo ou nulo) e o denominador (um inteiro positivo). Os racionais a/b e c/d são iguais se e só se $a \times d = b \times c$.

   As operações básicas para o tipo *racional* são as seguintes:

   - *Construtores*:
     - $cria\_rac : \mathbb{Z} \times \mathbb{N} \mapsto racional$
       $cria\_rac(n, d)$ tem como valor o número racional cujo numerador é n e cujo denominador é d ($d > 0$).
   - *Seletores*:
     - $num : racional \mapsto \mathbb{Z}$
       $num(r)$ tem como valor o numerador do racional r.
     - $den : racional \mapsto \mathbb{N}$
       $den(r)$ tem como valor o denominador do racional r.
   - *Reconhecedores*:
     - $eh\_racional : universal \mapsto lógico$
       $eh\_racional(arg)$ tem o valor $verdadeiro$ se arg é um número racional e tem o valor $falso$ em caso contrário.
     - $eh\_rac\_zero : racional \mapsto lógico$
       $eh\_rac\_zero(r)$ tem o valor $verdadeiro$ se r é o racional 0 e tem o valor $falso$ em caso contrário.
   - *Testes*:
     - $rac\_iguais : racional \times racional \mapsto lógico$
       $rac\_iguais(r_1, r_2)$ tem o valor $verdadeiro$ se r_1 e r_2 correspondem ao mesmo número racional e tem o valor $falso$ em caso contrário.

   (a) Escolha uma representação para o tipo *racional* usando dicionários.

(b) Escreva as operações básicas utilizando a representação escolhida.

(c) Escreva o transformador de saída para o tipo *racional*. Por exemplo,

```
>>> escreve_rac(cria_rac(1, 3))
'1/3'
```

(d) Escreva a função `produto_rac` que calcula o produto de dois racionais. Se $r_1 = a/b$ e $r_2 = c/d$ então $r_1 \times r_2 = ac/bd$. Por exemplo,

```
>>> escreve_rac(produto_rac(cria_rac(1,3), cria_rac(3,4)))
3/12
```

Note que esta função é uma *função de alto nível*, pois não pertence às operações básicas e, como tal, não pode usar a representação.

2. Suponha que desejava criar o tipo *relógio* para representar um instante de tempo dentro de um dia. Suponha que um relógio é caracterizado por um triplo de inteiros positivos, correspondentes às horas (entre 0 e 23), aos minutos (entre 0 e 59) e aos segundos (entre 0 e 59).

   As operações básicas para o tipo *relógio* são as seguintes:

   - *Construtores*:
     - $cria\_rel : \mathbb{N}_0^3 \mapsto relógio$
       $cria\_rel(h, m, s)$ tem como valor o *relógio* cujas horas são h, os minutos são m e os segundos são s.
   - *Seletores*:
     - $horas : relógio \mapsto \mathbb{N}_0$
       $horas(r)$ tem como valor as horas do *relógio* r.
     - $minutos : relógio \mapsto \mathbb{N}_0$
       $minutos(r)$ tem como valor os minutos do *relógio* r.
     - $segs : relógio \mapsto \mathbb{N}_0$
       $segs(r)$ tem como valor os segundos do *relógio* r.
   - *Reconhecedores*:
     - $eh\_relógio : universal \mapsto lógico$
       $eh\_relógio(arg)$ tem o valor $verdadeiro$ se arg é um *relógio* e tem o valor $falso$ em caso contrário.
     - $eh\_meia\_noite : relógio \mapsto lógico$
       $eh\_meia\_noite(r)$ tem o valor $verdadeiro$ se r corresponde à meia noite $00:00:00$ e tem o valor $falso$ em caso contrário.
     - $eh\_meio\_dia : relógio \mapsto lógico$
       $eh\_meio\_dia(r)$ tem o valor $verdadeiro$ se r corresponde ao meio dia $12:00:00$ e tem o valor $falso$ em caso contrário.
   - *Testes*:
     - $mesmas\_horas : relógio^2 \mapsto lógico$
       $mesmas\_horas(r_1, r_2)$ tem o valor $verdadeiro$ se r_1 e r_2 correspondem às mesmas horas e tem o valor $falso$ em caso contrário.

(a) Escolha uma representação interna para o tipo *relógio* recorrendo a listas.

(b) Escreva as operações básicas, utilizando a a representação escolhida.

(c) Suponha que a representação externa para os elementos do tipo *relógio* é `hh:mm:ss`, em que `hh` são os dois dígitos que representam as horas, `mm` são os dois dígitos que identificam os minutos e `ss` são os dois dígitos que identificam os segundos. Escreva o transformador de saída para o tipo *relógio*. Por exemplo,

```
>>> escreve_relogio(cria_relogio(9, 2, 34))
'09:02:34'
```

(d) Escreva o predicado `depois_rel` que recebe dois *relógios* e devolve *verdadeiro* apenas se o segundo *relógio* corresponder a um instante de tempo posterior ao primeiro *relógio*.

(e) Escreva a função `dif_segs` que calcula o número de segundos entre dois instantes, representados por dois relógios. Esta função apenas deve produzir um valor se o segundo instante de tempo for posterior ao primeiro, gerando uma mensagem de erro se essa condição não se verificar. Por exemplo,

```
>>> dif_segs(cria_rel(10, 2, 34), cria_rel(11, 21, 34))
4740
>>> dif_segs(cria_rel(10, 2, 34), cria_rel(9, 21, 34))
ValueError: dif_segs: primeiro arg posterior ao segundo
```

(f) Suponha que altera a representação interna do tipo relógio para um dicionário com as chaves `'horas'`, `'min'` e `'seg'`. O que deverá fazer às funções `escreve_relogio` e `dif_segundos` para que estas sejam usadas com esta nova representação? Justifique a sua resposta.

3. Suponha que desejava criar o tipo *data*. Uma *data* é caracterizada por um dia (um inteiro entre 1 e 31), um mês (um inteiro entre 1 e 12) e um ano (um inteiro que pode ser positivo, nulo ou negativo). Para cada *data*, deve ser respeitado o limite de dias de cada mês, incluindo o caso de Fevereiro nos anos bissextos. Recorde que um ano é bissexto se for divisível por 4 e não for divisível por 100, a não ser que seja também divisível por 400. Por exemplo, 1984 é bissexto, 1100 não é, e 2000 é bissexto.

   O tipo *data* tem as seguintes operações básicas:

   - *Construtores*:
     - $cria\_data : \mathbb{N} \times \mathbb{N} \times \mathbb{Z} \mapsto data$
       $cria\_data(d, m, a)$ tem como valor a data com dia d, mês m e ano a.
   - *Seletores*:
     - $dia : data \mapsto \mathbb{N}$
       $dia(dt)$ tem como valor o dia da data dt.

  - $mes : data \mapsto \mathbb{N}$
    $mes(dt)$ tem como valor o mês da data dt.
  - $ano : data \mapsto \mathbb{Z}$
    $ano(dt)$ tem como valor o ano da data dt.

- *Reconhecedores*:
  - $eh\_data : universal \mapsto lógico$
    $eh\_data(arg)$ tem o valor $verdadeiro$ se arg é uma $data$ e tem o valor $falso$ em caso contrário.

- *Testes*:
  - $mesma\_data : data^2 \mapsto lógico$
    $mesma\_data(d_1, d_2)$ tem o valor $verdadeiro$ se d_1 e d_2 correspondem à mesma data e tem o valor $falso$ em caso contrário.

(a) Escolha uma representação interna para o tipo *data* usando dicionários.

(b) Escreva as operações básicas para a representação escolhida.

(c) Supondo que a representação externa para um elemento do tipo *data* é `dd/mm/aaaa ee` (em que `dd` representa o dia, `mm` o mês, `aaaa` o ano e `ee` representa a era, a qual é omitida se o ano for maior ou igual a 0 e é escrita `AC` se o ano for menor que zero), escreva o transformador de saída para o tipo *data*. Por exemplo,

```
>>> escreve_data (cria_data (5, 9, 2018))
'05/09/2018'
>>> escreve_data (cria_data (5, 9, -10))
'05/09/10 AC'
```

(d) Defina o predicado `data_anterior` que recebe como argumentos duas *datas* e tem o valor verdadeiro apenas se a primeira *data* é anterior à segunda.

```
>>> data_anterior(cria_data(2, 1, 2003), \
                  cria_data(2, 1, 2005))
True
```

(e) Defina a função `idade` que recebe como argumentos a data de nascimento de uma pessoa e outra data posterior e devolve a idade da pessoa, em anos, na segunda data.

```
>>> idade(cria_data(2, 1, 2003), cria_data(1, 1, 2005))
2
>>> idade(cria_data(2, 1, 2003), cria_data(1, 2, 2005))
3
>>> idade(cria_data(2, 1, 2003), cria_data(1, 2, 2000))
ValueError: idade: a pessoa ainda não nasceu
```

4. Considere o tipo *time_ stamp* para representar um instante de tempo. Um *time_ stamp* corresponde a um par constituído por uma *data* e por um *relógio*.

   As operações básicas para o tipo *time_ stamp* são:

   - *Construtores*:
     - $cria\_time\_stamp : data \times relógio \mapsto time\_stamp$
       $cria\_time\_stamp(dt, rel)$ tem como valor o *time_ stamp* com *data* dt e *relógio* rel.
   - *Seletores*:
     - $data : time\_stamp \mapsto data$
       $data(ts)$ tem como valor a *data* de ts.
     - $relógio : time\_stamp \mapsto relógio$
       $relógio(ts)$ tem como valor o *relógio* do *time_ stamp* ts.
   - *Reconhecedores*:
     - $eh\_time\_stamp : universal \mapsto lógico$
       $eh\_time\_stamp(arg)$ tem o valor $verdadeiro$ se arg é um *time_ stamp* e tem o valor $falso$ em caso contrário.
   - *Testes*:
     - $mesmo\_time\_stamp : time\_stamp^2 \mapsto lógico$
       $mesma\_time\_stamp(ts_1, ts_2)$ tem o valor $verdadeiro$ se ts_1 e ts_2 correspondem ao mesmo *time_ stamp* e tem o valor $falso$ em caso contrário.

   (a) Escolha uma representação para o tipo *time_ stamp*.

   (b) Escreva as operações básicas com base na representação escolhida.

   (c) Escreva o predicado
   $depois\_ts : time\_stamp^2 \mapsto lógico$
   $depois\_ts(ts_1, ts_2)$ tem o valor $verdadeiro$ apenas se ts_1 corresponder a um instante posterior a ts_2.

5. Suponha que desejava criar o tipo *vetor*. Um vetor num referencial cartesiano pode ser representado pelas coordenadas da sua extremidade (x, y), estando a sua origem no ponto $(0, 0)$, ver Figura 9.1. Podemos considerar as seguintes operações básicas para vetores:

   - *Construtor*:
     - $vetor : \mathbb{R}^2 \mapsto vetor$
       $vetor(x, y)$ tem como valor o vetor cuja extremidade é o ponto (x, y).
   - *Seletores*:
     - $abcissa : vetor \mapsto \mathbb{R}$
       $abcissa(v)$ tem como valor a abcissa da extremidade do vetor v.

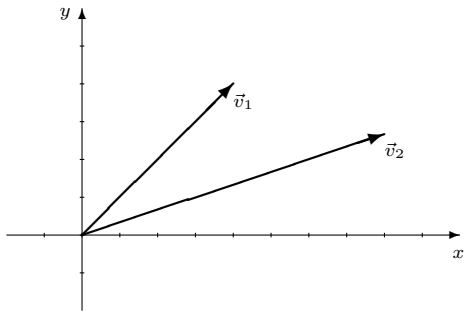

Figura 9.1: Exemplo de vetores.

  - $ordenada : vetor \mapsto \mathbb{R}$
    $ordenada(v)$ tem como valor a ordenada da extremidade do vetor v.

- *Reconhecedores*:
  - $eh\_vetor : universal \mapsto lógico$
    $eh\_vetor(arg)$ tem valor $verdadeiro$ apenas se arg é um vetor.
    $eh\_vetor\_nulo : vetor \mapsto lógico$
    $eh\_vetor\_nulo(v)$ tem valor $verdadeiro$ apenas se v é o vetor $(0, 0)$.
- *Teste*:
  - $vetores\_iguais : vetor \times vetor \mapsto lógico$
    $vetores\_iguais(v_1, v_2)$ tem valor verdadeiro apenas se os vetores v_1 e v_2 são iguais.

(a) Defina uma representação para vetores utilizando tuplos.

(b) Escreva as operações básicas, de acordo com a representação escolhida.

(c) Escreva uma função para calcular o produto escalar de dois vetores. O produto escalar dos vetores representados pelos pontos (a, b) e (c, d) é dado pelo real $a \times c + b \times d$.