

## Capítulo 9

# Abstração de dados

1. Suponha que desejava criar o tipo *racional* em Python. 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  $a \times d = b \times c$ .

- (a) Especifique as operações básicas para o tipo *racional*.
- (b) Escolha uma representação interna para o tipo *racional*.
- (c) Escreva as operações básicas, de acordo com a representação escolhida.
- (d) Escreva o transformador de saída `escreve_rac` para o tipo racional. Por exemplo,

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

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

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

- (f) Escreva a função `soma_rac` que calcula a soma de dois racionais. Se  $r_1 = a/b$  e  $r_2 = c/d$  então  $r_1 + r_2 = (ad + bc)/bd$ .

```
>>> escreve_rac(soma_rac(cria_rac(1,3), cria_rac(3,4)))
13/12
```

2. Suponha que desejava criar em Python 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).

- (a) Especifique as operações básicas para o tipo relógio.

- (b) Escolha uma representação interna para o tipo relógio recorrendo a listas.
- (c) Escreva as operações básicas, de acordo com a representação escolhida.
- (d) Suponha que a representação externa para os elementos do tipo relógio é `hh:mm:ss`, em que `hh` são os dígitos que representam as horas, `mm` são os dígitos que identificam os minutos e `ss` são os dígitos que identificam os segundos. Escreva o transformador de saída, `escreve_relogio`, para o tipo relógio. Por exemplo,

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

- (e) Escreva a função `diferenca_segundos` 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,

```
>>> diferenca_segundos(cria_relogio(10, 2, 34), \
                        cria_relogio(11, 2, 34))
3600
```

- (f) Suponha que altera a representação interna do tipo relógio para um dicionário com as chaves `'horas'`, `'min'` e `'seg'`. Escreva em Python as operações básicas, de acordo com esta nova representação.
- (g) O que deverá fazer às funções `escreve_relogio` e `diferenca_segundos` para que estas sejam usadas com esta nova representação? Justifique.

3. Suponha que desejava criar o tipo *data* em Python. 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.

O tipo *data* tem as seguintes operações básicas:

- *Construtores:*

- $cria\_data : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mapsto data$   
 $cria\_data(d, m, a)$  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:*

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

- *Testes:*

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

- Escolha uma representação interna para o tipo data usando dicionários.
- Escreva em Python as operações básicas, de acordo com a representação escolhida.
- 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 que 0 e 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, 2014))
05/09/2014
>>> escreve_data (cria_data (5, 9, -10))
05/09/0010 AC
```

- Defina a função `data_anterior` que recebe como argumentos duas datas e tem o valor verdadeiro se a primeira data é anterior à segunda e falso caso contrário.

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

- 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 na segunda data.

```
>>> idade(cria_data(2, 1, 2003), cria_data(2, 1, 2005))
2
>>> idade(cria_data(2, 1, 2003), cria_data(2, 3, 2006))
3
```

- Suponha que pretendia representar pontos num espaço cartesiano. Cada ponto é representado por duas coordenadas, a do eixo dos  $xx$  e a do eixo dos  $yy$ , ambas contendo valores reais.

- (a) Especifique as operações básicas do tipo *ponto*.
  - (b) Escolha uma representação para o tipo *ponto*.
  - (c) Escreva em Python as operações básicas, de acordo com a representação escolhida.
  - (d) Escreva uma função que recebe duas entidades do tipo *ponto* e que determina a distância entre esses pontos. A distância entre os pontos  $(x_1, y_1)$  e  $(x_2, y_2)$  é dada por  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ .
  - (e) Escreva uma função que recebe como argumento um *ponto* e que determina o quadrante em que este se encontra. A sua função deve devolver um inteiro entre 1 e 4.
5. Considere o tipo *timestamp* para representar um instante de tempo. Um *timestamp* corresponde a um par constituído por uma data e por um relógio.
- (a) Especifique as operações básicas para o tipo *timestamp*.
  - (b) Escolha uma representação para o tipo *timestamp*.
  - (c) Escreva as operações básicas com base na representação escolhida.
  - (d) Com base no tipo *timestamp*, escreva as seguintes funções:
    - i.  $depois : timestamp \times timestamp \mapsto \text{lógico}$   
 $depois(ts_1, ts_2)$  tem o valor *verdadeiro*, se  $ts_1$  corresponder a um instante posterior a  $ts_2$ .
    - ii.  $num\_segundos : timestamp \mapsto \text{inteiro}$   
 $num\_segundos(ts)$  tem como valor o número de segundos entre o *timestamp* com data 01/01/0000 e relógio 00:00:00 e o *timestamp*  $ts$ .
6. O tipo *pilha* corresponde a uma pilha de objetos físicos, à qual apenas se pode aceder ao elemento no topo da pilha e apenas se podem adicionar elementos ao topo da pilha.
- O tipo *pilha* é caracterizado pelas operações: *nova\_pilha* (cria uma pilha sem elementos), *empurra* (adiciona um elemento à pilha), *topo* (indica o elemento no topo da pilha), *tira* (devolve uma pilha igual ao seu argumento mas sem o elemento no topo da pilha), *e\_pilha* (decide se uma entidade é uma pilha), *e\_pilha\_vazia* (testa a pilha sem elementos) e *pilhas\_iguais* (testa a igualdade de pilhas).
- (a) Especifique formalmente estas operações, e classifique-as em construtores, seletores, reconhecedores e testes.
  - (b) Escolha uma representação para o tipo *pilha*.
  - (c) Escreva em Python as operações básicas, de acordo com a representação escolhida.