ROZDZIAŁ 3

Niezmienne struktury danych

W tym rozdziale przedstawię typowe zadania dotyczące struktur danych w języku F# wraz z przykładowymi rozwiązaniami.

Struktury danych pomagają programistom w grupowaniu i reprezentowaniu powiązanych wartości w użyteczne, logiczne jednostki. Język programowania F# ma bardzo dużo wbudowanych niezmiennych struktur danych, które m.in. zawierają:

- **typ opcji** (*option type*);
- **krotki** (*tuples*);
- **listy** (*lists*);
- **sekwencje** (*sequences*);
- **zbiory** i **mapy** (*sets and maps*);
- **rekordy** (*records*), **unie** i wiele innych.

Niektóre z nich (typ opcji, krotki, listy, sekwencje oraz zbiory) omówiono w przykładowych zadaniach w tym rozdziale.

3.1 Opcje

W języku F# **typ opcji** (*option type*) może przyjmować dwie wartości: albo jakąś wartość (oznaczoną jako `Some`), albo jej brak (oznaczoną

jako `None`). Opcje są bardzo wygodne, np. do zwracania wartości wyszukiwania, gdy wyniki mogą być dostępne, ale nie jest to pewne.

Oto przykład programu, w którym zastosowano bezpieczne dzielenie, aby uniknąć dzielenia przez zero.

Zadanie 3.1

Napisz program, w którym zastosowano przykład bezpiecznego dzielenia, wykorzystujący typ `Option`.

Przykładowe rozwiązanie – listing 3.1

```
// Zadanie 3.1

open System

let bezp_dzielenie (x, y) = // Funkcja bezp_dzielenie
"obsługuje" dzielenie przez 0
    match  y with
    | 0 -> None
    | _ -> Some (x/y)

let wynik = bezp_dzielenie(6, 2) // Próba normalnego
dzielenia
let wynik1 = bezp_dzielenie(6, 0) // Próba dzielenia przez
0

if Option.isSome wynik then Console.WriteLine("Wynik
dzielenia = {0}.", bezp_dzielenie(6, 2))
if Option.isNone wynik1 then Console.WriteLine("Uwaga:
próba dzielenia przez 0.")

// Zatrzymanie konsoli
Console.WriteLine("Naciśnij dowolny klawisz.")
Console.ReadKey(true) |> ignore
```

W programie do wykonywania operacji z użyciem opcji skorzystaliśmy z właściwości modułu `Option`.