W programowaniu zdarza się tak, że zdefiniowano kilka klas o takiej samej implementacji i jednocześnie wspomniane klasy różnią się od siebie wyłącznie typami danych. Podobny problem występuje wówczas, gdy w obrębie jednej klasy zdefiniowano zmienne i funkcje członkowskie należące do różnych typów, ale o analogicznej funkcjonalności. Wymienione przypadki powodują niepotrzebne rozbudowanie kodu źródłowego oraz zapowiadają duże trudności w przyszłej konserwacji — utrzymaniu tego kodu.

Omówione powyżej problemy można rozwiązać z wykorzystaniem tzw. szablonów — wzorców klas, które wspomagają zastosowanie tego samego kodu do różnych typów danych. **Szablon klasy** (ang. *class template*) to **klasa uogólniona** (ang. *generic class*), która pozwala zdefiniować na jej podstawie całą rodzinę klas w zależności od typu/typów danych, do których należą jej zmienne członkowskie i na których operują jej metody.

Szablony klas — analogicznie do szablonów funkcji — są **parametryzowane** (ang. *parametrized*) za pomocą **parametrów szablonowych** (ang. *template parameters*). Rolę parametrów szablonowych odgrywają **uogólnione typy danych** (ang. *generic data types*).

Postać ogólna definicji szablonu klasy jest następująca:

```
template <class T>
class nazwa_klasy {
    deklaracje_i_definicje_elementów_członkowskich;
}
```

```
//Definicja szblonu klasy Prostokąt
template<class T>
//UWAGA:
//Parametr szablonu klasy prostokąt jest uogólnionym typem danych T. To jedyny parametr tego szablonu.
class Prostokat {
public:
    //Deklaracje szablonów zmiennych członkowskich
    T bok1, bok2;
    //Definicja (szablonu) metody pole()
    T pole() {
        return bok1 * bok2;
    }
    //Deklaracja (szablonu) metody obwod()
    T obwod();
};
```

Ćwiczenie 18.3

Zmodyfikuj program zawarty w przykładzie 18.3 — zamiast szablonu klasy `Prostokat` zdefiniuj szablon klasy `Kwadrat` zawierający definicje szablonu zmiennej członkowskiej `bok` oraz szablonów metod `pole()` i `obwod()`. Utwórz instancje szablonu klasy `Kwadrat` w programie głównym skonkretyzowane dla typów danych `long` i `float`. Przyjmij, że zadana długość boku kwadratu wynosi `1`.

Ćwiczenie 18.4

Zmodyfikuj program zawarty w przykładzie 18.4 — uzupełnij szablon klasy `Oceny` o definicję (szablonu) funkcji członkowskiej pozwalającej wyznaczyć średnią arytmetyczną elementów zapisanych w (szablonie) zmiennej członkowskiej `oceny`. Wykorzystaj tę funkcję w programie głównym. Szablon klasy `Oceny` skonkretyzuj tam dla typu danych `float` oraz `n` równego `4`.

Specjalizacja szablonu klasy (ang. *class template specialization*) stanowi wersję specjalną szablonu klasy „wyspecjalizowaną” dla określonego typu lub typów danych. Ten drugi przypadek dotyczy sytuacji, w której szablon klasy ma wiele parametrów szablonowych. Specjalizacje klas nie mają żadnych parametrów szablonowych (ang. *template parameters*).

Specjalizacje szablonów klas są traktowane przez kompilator jako klasy zupełnie niezależne od szablonów, z którymi są skojarzone. Tym samym implementacja klasy specjalizowanej może być, o czym była mowa już wcześniej, całkowicie odmienna od implementacji (a więc również funkcjonalności) szablonu.

Jeśli w procesie kompilacji programu kompilator napotyka wyrażenie, którego zadaniem jest utworzenie obiektu na podstawie szablonu klasy, w pierwszej kolejności poszukuje w kodzie źródłowym definicji wersji specjalizowanej tego szablonu. Wspomniana specjalizacja szablonu dotyczy typu argumentu, który odpowiada parametrowi szablonowemu. W przypadku gdy specjalizacja szablonu klasy dla zadanego typu danych zostaje znaleziona, to właśnie ona jest wykorzystywana do utworzenia obiektu. W przeciwnym razie, tj. jeśli wersja specjalizowana szablonu klasy dla zadanego typu danych nie zostaje znaleziona, kompilator tworzy instancję szablonu klasy skonkretyzowaną dla tego typu danych.

```cpp
//Definicja specjalizacji szablonu klasy Oceny dla typu double
template<>
class Oceny<double> {
public:
    //Definicja kontruktora domyślnego
    Oceny() {
        cout << "Wywołanie konstruktora klasy specjalizowanej..." << endl;
    }
    //Definicje szablonów zmiennych członkowskich
    double wyklad;
    double cwiczenia;
    double laboratorium;
    double seminarium;
    //Definicja szablonu funkcji członkowskiej srednia()
    double srednia() {
        return (round(wyklad) + round(cwiczenia) + round(laboratorium) + round(seminarium)) / 4;
        /*UWAGA:
        Kod (implementacja) metody średnia w klasie specjalizowanej różni się od kodu szablonu metody o tej
        samej nazwie
        zdefiniowanej w klasie uogólnionej.*/
    }

};
```

Ćwiczenie 18.5

Zmodyfikuj program zawarty w przykładzie 18.5 — zdefiniuj specjalizacje klasy uogólnionej `Oceny` dla typów danych `long` i `float`. Wykorzystaj wymienione specjalizacje w programie głównym.

Szablony a polimorfizm

Stosowanie szablonów funkcji oraz szablonów klas to kolejna możliwość implementacji mechanizmu polimorfizmu (wielopostaciowości) w języku C++. W szczególności wykorzystanie szablonów funkcji pozwala definiować funkcje o tych samych nazwach i tej samej implementacji, ale operujących na różnych typach danych. To samo dotyczy szablonów klas umożliwiających tworzenie rodziny klas o analogicznej implementacji, ale różniących się od siebie typem lub typami danych, do których należą zmienne członkowskie i na których operują funkcje członkowskie klas — członków tej rodziny klas.

Generowanie przez kompilator instancji szablonów funkcji skonkretyzowanych dla określonych typów danych jest podobne do mechanizmu przeciążania funkcji (ang. *function overloading*), które jest jednym z najważniejszych narzędzi polimorfizmu. Różnica pomiędzy instancjami szablonów funkcji a funkcjami przeciążonymi polega na tym, że instancje szablonów funkcji mają identyczną implementację — ten sam kod, a implementacja funkcji przeciążonych może być różna. Przy tym uwzględnienie dodatkowo specjalizacji szablonów funkcji jeszcze bardziej zbliża instancje szablonów do funkcji przeciążonych.

Pytania kontrolne

1. W jakim celu stosuje się szablony funkcji (ang. *function templates*)? Jaka jest różnica pomiędzy przeciążaniem funkcji (ang. *function overloading*) a wykorzystaniem szablonów funkcji?
2. Wyjaśnij pojęcia: funkcja szablonowa (ang. *template function*), parametr szablonowy (ang. *template parameter*), uogólniony typ danych (ang. *generic data type*).
3. W jakim celu stosuje się szablony klas (ang. *class templates*)?
4. Co to jest instancja szablonu (ang. *template instance*)?
5. Na czym polega specjalizacja szablonów (ang. *template specialization*)?
6. Czy wykorzystanie szablonów (funkcji i klas) ma związek z polimorfizmem?

Zadania

1. Napisz program pozwalający obliczać pola powierzchni i obwody figur płaskich: kwadratu, prostokąta i koła. Dane wejściowe — parametry wymienionych figur — mają być wprowadzane z klawiatury, a wyniki wyświetlane na ekranie monitora. Uwzględnij założenie, że parametry figur mogą być zarówno liczbami całkowitymi, jak i rzeczywistymi. Wykorzystaj zdefiniowane samodzielnie szablony funkcji. Obsługę każdego z zastosowanych typów danych całkowitych i rzeczywistych zrealizuj za pomocą instancji szablonów funkcji skonkretyzowanych dla tych typów.
2. Zrób tak jak w zadaniu 1., lecz dodatkowo zdefiniuj specjalizacje szablonów funkcji dla parametrów figur będących liczbami rzeczywistymi.
3. Zrób tak jak w zadaniu 1., lecz zamiast szablonów funkcji wykorzystaj zdefiniowane samodzielnie szablony klas. Uwzględnij wersje specjalizowane tych szablonów dla danych należących do typów rzeczywistych.

4. Napisz program pozwalający wyznaczyć średnią ocen semestralnych uzyskanych z następujących przedmiotów: języka polskiego, języka angielskiego, matematyki, informatyki. Uwzględnij średnią arytmetyczną oraz średnią ważoną z wagami: język polski — `0.1`, język angielski — `0.2`, matematyka — `0.4`, informatyka — `0.3`. Przyjmij założenie, że oceny semestralne mogą być liczbami całkowitymi o wartościach: `1`, `2`, `3`, `4`, `5`, `6` oraz liczbami rzeczywistymi o wartościach: `1`, `1.5`, `2`, `2.5`, `3`, `3.5`, `4`, `4.5`, `5`, `5.5`, `6`. Dane wejściowe, czyli oceny semestralne uzyskane z wymienionych przedmiotów, mają być wprowadzane z klawiatury. Wyniki, tj. średnia arytmetyczna i średnia ważona, powinny być wyświetlane na ekranie monitora. Wykorzystaj zdefiniowane samodzielnie szablony funkcji. Obsługę każdego z użytych w programie typów danych zrealizuj za pomocą instancji szablonów funkcji skonkretyzowanych dla tych typów.

5. Zrób tak jak w zadaniu 4., lecz dodatkowo zdefiniuj specjalizacje szablonów funkcji dla danych wejściowych będących liczbami rzeczywistymi.

6. Zrób tak jak w zadaniu 4., lecz zamiast szablonów funkcji wykorzystaj zdefiniowane samodzielnie szablony klas. Uwzględnij wersje specjalizowane tych szablonów dla danych należących do typów rzeczywistych.