# **Szablony funkcji i klas (templates)**

Szablony (ang. Templates) nazywane też Typami Generycznymi (ang. Generic Types) lub Typami Uogólnionymi to element języka C++ pozwalający na tworzenie kodu niezależnego od typów, algorytmów oraz struktur danych. Szablony są techniką realizacji polimorfizmu na innym poziomie niż za pomocą funkcji wirtualnych i dziedziczenia.

Mechanizm szablonów można rozumieć jako pewną formę makrodefinicji (preprocesora). Szablony ze względu na szerokie rozpowszechnienie się zastosowań biblioteki STL (ang. Standard Template Library) uważane są za jedną z najważniejszych właściwości języka C++. Dzięki szablonom mamy możliwości programowania ogólnego (ang. Generic programming). Tworząc szablon funkcji (podobnie jak szablon klasy) tworzymy kod działający na nieopisanych jeszcze typach – funkcja działa na typach ogólnych (które nie istnieją w języku C++), w momencie wywołania funkcji pod te typy ogólne podstawiane są konkretne typy, jak np. int czy char. Przekazujemy szablonowi typy jako parametry, podczas kompilacji następuje tak zwana konkretyzacja szablonu (ang. template instantiation), podczas której kompilator na podstawie typów danych przekazanych wzorcowi generuje docelowy kod do obsługi danego typu.

**Szablony funkcji**

Szablon funkcji to mechanizm umożliwiający automatyczną generację funkcji. Jest to schemat, według którego postępuje kompilator. Programista dostarcza jedynie ogólny zarys ciała funkcji bez określania konkretnych argumentów wywołania. Na tej podstawie kompilator jest w stanie wytworzyć dowolną ilość funkcji działających identycznie, a różniących się jedynie typem argumentów funkcji. Szablony zezwalają na definiowanie całych rodzin funkcji, które następnie mogą być używane dla różnych typów argumentów.

Definicja szablonu funkcji wygląda następująco:

template <typename T>

T MojaFunkcja(T a, T b, ...) {

//instrukcje

...

};

lub

template <class T>

T MojaFunkcja(T &a, T &b, ...) {

//instrukcje

...

};

Definicja umieszczona bezpośrednio przed definicją funkcji informuje że funkcja będzie korzystała z fikcyjnego typu o nazwie T. Definicja dotyczy tylko pojedynczej funkcji zdefiniowanej bezpośrednio po frazie „template”.

Przykłady implementacji funkcji dla różnych typów z użyciem szablonów:

Przyklad1

template <typename T>

T Sprawdz(T a, T b) {

return (a>b)?a:b;

};

Przyklad2

template <typename T>

T Minimum(T a, T b, T c) {

if (a <= b && a <= c) return(a);

if (b <= a && b <= c) return(b);

return(c);

};

Wyrażenie template<typename T> lub template<class T> oznacza, że mamy do czynienia z szablonem, który posiada jeden parametr formalny nazwany T. Słowo kluczowe typename/class oznacza, że parametr ten jest typem (nazwą typu) lub klasy. Nazwa tego parametru może być następnie wykorzystywana w definicji funkcji w miejscach, gdzie spodziewamy się nazwy typu/klasy. I W przykładzie1 mamy zdefiniowaną funkcję, która przyjmuje dwa argumenty typu T i zwraca wartość typu T, będącą wartością większego z dwu argumentów. Typ T jest na razie niewyspecyfikowany. W tym sensie szablon definiuje całą rodzinę funkcji. Konkretną funkcję z tej rodziny wybieramy poprzez podstawienie za formalny parametr T konkretnego typu będącego argumentem szablonu. Takie podstawienie nazywamy konkretyzacją szablonu. Argument szablonu umieszczamy w nawiasach <> za nazwą szablonu.

**Szablony klas**

Szablony klas (inne określenia: wzorce klas, klasy ogólne) podobne są do makr, tyle że wykonywane są przez kompilator, a nie przez preprocesor. Cechują je pewne własności, których jednak nie ma preprocesor, np. można tworzyć rekurencyjne wywołania. Podobnie jak w przypadku szablonów funkcji, szablon klasy definiuje nam w rzeczywistości całą rodzinę klas.

Definicja szablonu klasy wygląda następująco:

template <class T>

class MojaKlasa {

//definicja klasy

};

Definicja umieszczona bezpośrednio przed definicją klasy informuje że klasa będzie korzystała z fikcyjnego typu o nazwie T. Definicja dotyczy tylko pojedynczej klasy zdefiniowanej bezpośrednio po frazie „template”.

Przykłady implementacji i użycia klas z użyciem szablonów:

Przyklad1

template <class T>

class c\_klasa {

public:

T zmienna;

c\_klasa(T l) { zmienna = l; }

void wyswietl() { cout << zmienna << endl; }

};

int main() {

c\_klasa<int> calkowita(2);

calkowita.wyswietl();

c\_klasa<char> znak('b');

znak.wyswietl();

}

Przyklad2

#include <iostream>

using namespace std;

// Deklaracja klasy szablonu c\_klasa

template <class T>

class c\_klasa {

public:

T zmienna; // Pole przechowujące zmienną typu T

// Konstruktor klasy c\_klasa, inicjalizuje zmienną zmienna

c\_klasa(T l);

void wyswietl();

};

// Definicja konstruktora klasy c\_klasa

template <class T>

c\_klasa<T>::c\_klasa(T l) : zmienna(l) {}

// Definicja metody wyswietl() klasy c\_klasa

template <class T>

void c\_klasa<T>::wyswietl() {

cout << zmienna << endl;

}

int main() {

// Tworzenie obiektu calkowita klasy c\_klasa przechowującego liczbę całkowitą 2

c\_klasa<int> calkowita(2);

calkowita.wyswietl();

// Tworzenie obiektu znak klasy c\_klasa przechowującego znak 'b'

c\_klasa<char> znak('b');

znak.wyswietl();

return 0;

}

**Zadania**

1. Napisz szablon funkcji, która otrzymuje jako argumenty dwie referencje do zmiennych typu będącego parametrem szablonu i zamienia wartościami zmienne, do których referencje otrzymała w argumentach
2. Napisz szablon funkcji wypisz, która dostaje jako argument wsk wskaźnik wskazujący na wartość o typie będącym parametrem szablonu i wypisującą na standardowym wyjściu wartość wskazywaną przez argument.
3. Napisać program, w którym zostaną zaimplementowane 3 funkcje sortujące (algorytmem bąbelkowym, przez wstawianie oraz quicksort) z wykorzystaniem szablonów funkcji. Należy zaprezentować ich zastosowanie dla różnych rodzajów danych wejściowych. Funkcje powinny przyjmować jako parametr tablice do posortowania oraz liczbę elementów w tablicy.
4. Napisz szablon klasy Kalkulator, który umożliwia wykonywanie podstawowych operacji arytmetycznych na dwóch zmiennych dowolnego typu.