1 Szablony - definicja, zastosowania

Szablony są jedną z głównych cech języka *C++*. Dzięki nim możemy dostarczać generyczne typy i funkcje, bez kosztów czasu wykonania. Skupiają się na pisaniu kodu w sposób niezależny od konkretnego typu, dzięki czemu wspierają programowanie generyczne. *C++* to bogaty język wspierający polimorficzne zachowania zarówno w czasie wykonania jak i kompilacji. W tym pierwszym używa hierarchii klas i wywołań funkcji wirtualnych by wspierać praktyki zorientowane obiektowo, gdzie wywoływana funkcja zależy od typu obiektu docelowego podczas czasu wykonania. Natomiast w czasie kompilacji szablony wspierają programowanie generyczne, gdzie wywoływana funkcja zależy od statycznego typu czasu kompilacji argumentów szablonu.

Polimorfizm czasu kompilacji był w języku od bardzo dawna. Polega on na dostarczeniu szablonu, który umożliwia kompilatorowi wygenerowanie kodu w czasie kompilacji.

Grają kluczową rolę w projektowaniu obecnych, znanych i popularnych bibliotek i systemów. Stanowią podstawę technik programowania w różnych dziedzinach, począwszy od konwencjonalnego programowania ogólnego przeznaczenia do oprogramowywania wbudowanych systemów bezpieczeństwa.

Szablon to coś w rodzaju przepisu, z którego translator *C++* generuje deklaracje.

```
template<typename T>
T kwadrat (T x) {
    return x * x;
}
```

Kod ten deklaruje rodzinę funkcji indeksowanych po parametrze typu. Można odnieść się do konkretnego członka tej rodziny przez zastosowanie konstrukcji `kwadrat<int>`. Mówimy wtedy, że żądana jest specjalizacja szablonu dla funkcji `kwadrat` z listą argumentów szablonu `<int>`. Proces tworzenia specjalizacji nosi nazwę inicjalizacji szablonu, potocznie zwany inicjalizacją. Kompilator C++ stworzy stosowny odpowiednik definicji funkcji:

```
int kwadrat(int x) {
    return x * x;
}
```

Argument typu `int` jest podstawiony za parametr typu `T`. Kod wynikowy jest sprawdzany pod względem typu, by zapewnić brak błędów wynikających z podmiany. Inicjalizacja szablonu jest wykonywana tylko raz dla danej specyfikacji nawet jeśli program zawiera jej wielokrotne żądania.

W przeciwieństwie do języków takich jak *Ada* czy *System F*, lista argumentów szablonu może być pominięta z żądania inicjalizacji szablonu funkcji. Zazwyczaj, wartości parametrów szablonu są dedukowane.

```
double d = kwadrat(2.0);
```

Argument typu jest dedukowany na `double`. Warto zauważyć, że odmiennie niż w językach takich jak *Haskell* czy *System F*, parametry szablonu w *C++* nie są ograniczone względem typów.

Szablonów używa się do zmniejszania kar abstrakcji i zjawiska *code bloat* w systemach wbudowanych w stopniu, który jest niepraktyczny w standardowych systemach obiektowych. Robi się to z dwóch powodów:

- Po pierwsze, inicjalizacja szablonu łączy informacje zarówno z definicji, jak i z kontekstu użycia. To oznacza, że pełna informacja zarówno z definicji jak i z wywołanych kontekstów (włączając w to informacje o typach) jest udostępniana generatorowi kodu. Dzisiejsze generatory kodu dobrze sobie radzą z używaniem tych informacji w celu zminimalizowania czasu wykonania i przestrzeni kodu. Różni się to od zwykłego przypadku w języku obiektowym, gdzie wywołujący i wywoływany są kompletnie oddzieleni przez interfejs, który zakłada pośrednie wywołania funkcji.

- Po drugie, szablon w C++ jest zazwyczaj domyślnie tworzony tylko jeśli jest używany w sposób niezbędny dla semantyki programu, automatycznie minimalizując miejsce w pamięci, które wykorzystuje aplikacja. W przeciwieństwie do języka *Ada* czy *System F*, gdzie programista musi wyraźnie zarządzać inicjalizacjami.