08. Szablony

Szablony są innym mechanizmem realizacji polimorfizmu niż metody wirtualne i dziedziczenie. Nie są też bezpośrednio związane z paradygmatem obiektowym.

1 Szablony funkcji

Funkcje szablonowe są sparametryzowanymi definicjami funkcji i umożliwiają automatyczne ich „generowanie".

```
template<class T>
void print_array(T *tab, int size) {
  for (int i = 0; i < size; i++)
    std::cout << tab[i] << " ";
}
```

Użycie funkcji szablonowej:

```
int t[] = { 2, 0, 1, 3 };
print_array<int>(t, 4);
```

Zapoznaj się z kodem zawartym w pliku źródłowym *pob08-sort.cpp*.

Zadanie 01 Napisz funkcję szablonową `print_pretty_array(T *tab, int size)`, która wypisze na ekran tablicę w postaci:

```
{
    [0] : 2,
    [1] : 0,
    [2] : 1,
    [3] : 3,
}
```

Użyj powyższej funkcji na dwóch tablicach, przechowujących odpowiednio elementy typu `float` oraz `std::string`.

Zadanie 02 Z funkcji `bubble_sort` uczyń funkcję szablonową. Przetestuj jej działanie na typach `float` oraz `std::string` wykorzystując funkcję `print_pretty_array`.

2 Szablony klas

Zadanie 03 Z klasy `Stack` uczyń szablon klasy. Przetestuj tworząc dwa stosy zawierające elementy typu `float` oraz `std::string`.

Zadanie 04 Do klasy szablonowej `Stack` dodaj jako kolejny parametr szablonu maksymalny rozmiar tablicy przechowującej elementy. Powinien on zastąpić stałą `MAX_SIZE`:

```
1 static const int MAX_SIZE = 100;
```

Dostosuj kod w funkcji `main` do powyższych zmian.

Zadanie 05 Napisz szablonową klasę `Pair<T1, T2>`, która będzie reprezentować parę elementów. Następnie stwórz przykładowy stos elementów typu `Pair`.

3 Przyjaźń

Klasa (lub funkcja) zaprzyjaźniona *P* (ang. *friend*) to klasa (funkcja), która ma dostęp do składowych prywatnych i chronionych klasy *K*, z którą to klasą jest zaprzyjaźniona. Przyjaźń definiuje się w klasie *K* poprzez słowo kluczowe `friend`.

Zadanie 06 Chcemy użyć funkcji `print_pretty_array` do wypisania elementów stosu, bez zmieniania widoczności składowych tej klasy. Dodaj zaprzyjaźnioną z klasą `Stack` funkcję (zdefiniowaną globalnie) `print_pretty_array(const Stack& stack)`, aby umożliwić funkcji dostęp do prywatnych składowych stosu.

Podobnie uczyń z funkcją `bubble_sort`. Przetestuj działanie obu funkcji na klasie `Stack`.

Zadanie 07 Co należy zrobić, aby móc użyć funkcji sortującej na stosie zawierającym elementy typu `Pair`? Zdefiniuj odpowiednie metody i/lub operatory, aby móc sortować obiekty typu `Pair` w następujący sposób: para jest mniejsza, jeżeli jej pierwszy element jest mniejszy, lub w przypadku gdy pierwsze elementy par są identyczne, porównywany jest drugi element. Przetestuj odpowiednim fragmentem kodu.