Programowanie obiektowe w C++: Enum, Funkcje i Klasy zaprzyjaźnione

Ćwiczenie 1

Napisz program, zawierający zmienną wyliczeniową, zawierającą dni tygodnia.
Wypisz wartości dla trzech kolejnych dni tygodnia.

Ćwiczenie 2

Napisz program, który listuje marki samochodów. Następnie utwórz zmienną wywołania danej marki `my_car_brand`i ustaw ją na jedną z nich. Wypisz wartość dla parametru RollsRoyce.

Audi = 4,
BMW = 5,
Cadillac = 11,
Ford = 44,
Jaguar = 45,
Lexus,
Maybach = 55,
RollsRoyce = 65,
Saab = 111

Ćwiczenie 3

Napisz program, który posiada dwie klasy: *GrupaZaawawansowana* oraz *GrupaPodstawowa*, oraz dwie funkcje *inkrementujOcene*(), która podnosi stopień o jeden oraz *ustawOceneNiedostateczna*(), która ustawia ocenę na 2, przekazaną jako parametr z klasy GrupaZawawansowana, a wewnątrz ciała funkcji , ustawia parametr *ocena* przekazanego typu. Klasa *GrupaZaawawansowana* powinna zawierać funkcje zaprzyjaźnioną *inkrementujOcene*() oraz *ustawOceneNiedostateczna*(), które zawarte są w programie. Konstruktor, który ustawia domyślnie ocenę na zero podczas inicjowania klasy. Funkcję wypisującą ocenę *wypiszOcene*(), oraz zmienną ocena. Klasa *GrupaPodstawowa*, powinna zawierać funkcję *zmienOcene*() – ocena ma być przekazana parametrem do funkcji. Wywołaj program w podobny sposób:

```
int main() {
	GrupaZaawawansowana grupaZaawawansowanaSokoly;
	GrupaPodstawowa grupaPodstawowaZoltodzioby;
	grupaZawawansowanaSokoly.wypiszOcene();

	ZaawawansowanaSokoly:inkrementujOcene(5);
	ZaawawansowanaSokoly.wypiszOcene();

	grupaPodstawowaZoltodzioby:ustawOceneNiedostateczna(grupaZaawawansowanaSokoly);
	grupaZawawansowanaSokoly.wypiszOcene();

	grupaPodstawowaZoltodzioby.ocena = 7;
	cout << grupaPodstawowaZoltodzioby.ocena << endl;
}
```

Ćwiczenie 4

Napisz program, który tworzy tablice struktur różnych typów służące do przechowywania wybranych danych osobowych pracowników.

Szablony

Szablony wspomagają bezpośrednio programowanie z użyciem typów jako parametrów. Mechanizm szablonów umożliwia używanie typów i wartości jako parametrów m.in. definicji funkcji oraz klas. Szablony można łatwo reprezentować i łączyć ze sobą ogólne koncepcje programistyczne.

Szablony1 umożliwiają definiowanie funkcji uniwersalnych, bez podawania typów. Kompilator C++ podczas kompilacji tworzy funkcje odpowiednich typów. Definiując szablony funkcji, zmniejszamy liczbę funkcji, a program może używać tej samej nazwy dla funkcji realizujących konkretną operację, niezależnie od typu zwracanej wartości i typów parametrów.

Wszystkie szablony funkcji zaczynają się od słowa kluczowego template poprzedzającego zawartą w nawiasach ostrokątnych < > listę parametrów formalnych wzorca funkcji. Każdy z tych parametrów, reprezentujący typ, musi być poprzedzony słowem class (lub typename), tak jak tutaj:

template <class T> lub emplate <typename T>

Litera T oznacza uniwersalny typ szablonu. Parametry formalne w definicji wzorca wykorzystywane są (podobnie jak argumenty typów wbudowanych i zdefiniowanych przez użytkownika) do określenia typów argumentów funkcji i typu zwracanej przez nią wartości oraz do deklarowania przez nią zmiennych. Po definicji wzorca występuje definicja funkcji. Słowo kluczowe class (lub typename) wykorzystywane do określenia typów parametrów wzorca funkcji oznacza tutaj „każdy typ wbudowany lub zdefiniowany przez użytkownika”.

Ćwiczenie 5

Napisz program zawierający prosty szablon funkcji *max()*, która zwraca większą z dwóch wartości typu T.

Ćwiczenie 6

Napisz program zawierający szablon funkcji *wypiszTablice(),* który wyświetla zadeklarowane w programie elementy tablicy. W definicji szablonu typ T oznacza typ tablicy, a T1 typ licznika tablicy. Stworzona za pomocą szablonu funkcja wypisuje elementy tablic typów int, double oraz char.