Zadanie 1 (STL)

Napisz funkcję, która będzie obliczała najmniejszą i największą wartość występującą w wektorze liczb (int lub double). Wektor powinien być przekazywany do funkcji jako argument (np. przez referencję). Wynik powinien być zwracany w postaci pary liczb (szablon pair), z których pierwsza będzie zawierała obliczone minimum, a druga maksimum.

Zadanie 2 (STL)

Napisz funkcję, która przyjmie wektor liczb całkowitych (int) i zwróci drugi wektor, zawierający wyłącznie parzyste wartości z pierwszego wektora. Wartości w nowym wektorze powinny zostać uporządkowane w ten sam sposób, co w wektorze wejściowym. Przykład:

Wektor wejściowy: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (rozmiar: 11)
Wektor wynikowy: 0, 2, 4, 6, 8, 10 (rozmiar: 6)

Zadanie 3 (elementy statyczne)

Napisz klasę zawierającą metodę **bool oryginal()**. Metoda powinna zwracać true, jeżeli zostanie wywołana na pierwszym utworzonym w programie obiekcie tej klasy. Dla każdego obiektu utworzonego później, metoda powinna zawsze zwracać false. Podpowiedź: wykorzystaj możliwości oferowane przez pola statyczne.

Zadanie 4 (elementy statyczne)

Napisz klasę zawierającą statyczną metodę **bool czyIstnieje()**. Metoda powinna zwracać true, jeżeli w danej chwili w programie istnieje przynajmniej jeden obiekt tej klasy (i false w przeciwnym przypadku). Metoda powinna zwracać poprawny wynik także w sytuacji, w której wszystkie utworzone obiekty tej klasy zostaną usunięte z pamięci.

Zadanie 5 (operatory)

Napisz klasę LiczbaZespolona, zawierającą dwa pola prywatne typu double (**a** oraz **b**). Każde z pól powinno posiadać swój setter i getter. Następnie **poza klasą** zdefiniuj operatory dodawania i odejmowania obiektów tej klasy. Każdy z tych operatorów powinien zwracać nowy obiekt klasy LiczbaZespolona. Użycie konstruktorów jest opcjonalne.

Zadanie 6 (operatory)

Dla klasy z zadania 5 zdefiniuj operator przekierowania (wypisania). Wartości przechowywane w obiekcie powinny być wypisywane w formacie „**[a, b]**”.

Zadanie 7 (polimorfizm)

Zdefiniuj interfejs **Zwierze**, zawierający metodę czysto wirtualną **string opis()**.

Zaimplementuj klasy **Pies, Kot** i **Krolik**, realizujące interfejs **Zwierze**. Każda z klas powinna:

- posiadać **stałe** pole **string imie**
- posiadać konstruktor przyjmujący i ustawiający wartość pola **imię**
- nadpisywać metodę **opis()** tak, aby zwracała gatunek oraz imię zwierzęcia, np. „Pies Burek”

Zaimplementuj funkcję **void listaZwierzat()**, przyjmującą jako argument **wektor wskaźników na klasę Zwierze**. Funkcja powinna dla każdego obiektu z wektora wywołać metodę **opis()** i wypisać jej wynik na ekran.

W funkcji **main()** przetestuj działanie całości - stwórz **wektor wskaźników na klasę Zwierze** i wypełnij go dynamicznie tworzonymi obiektami konkretnych klas, następnie wywołaj funkcję **listaZwierzat()**.