ZADANIA DO SAMODZIELNEGO ĆWICZENIA

Iteratory

Iteratory są specjalnymi obiektami, które umożliwiają poruszanie się po kolekcjach z biblioteki standardowej C++. Do tej pory poznaliśmy już iterator $std :: vector < T >:: iterator$, którego czasem używaliśmy w pętli for do iterowania po kolekcji **std::vector**.

Do zrobienia zadań z tej serii wystarczy wiedzieć, że będziemy używać iteratorów jednokierunkowych, a więc takich dla których zdefiniowane są operatory inkrementacji(++), tożsamości (==, !=) oraz dereferencji (*). Operator dereferencji użyty na iteratorze pozwala uzyskać dostęp do wskazywanego obiektu w kolekcji.

Niektóre z iteratorów użytych w przykładach posiadają więcej operatorów i dodatkowe metody, ale do zrobienia zadań wystarczy użycie wymienionych 4 operatorów.

Ćwiczenie 0

Napisz szablon funkcji, który przyjmuje dwa iteratory z kolekcji standardowej i zwraca przez referencję wartości najmniejszego i największego elementu pomiędzy tymi iteratorami. Wykorzystaj poniższy szablon

```
#include <iostream>
#include <vector>

template <class T>
void findMaxMin(double &max, double &min, T start, T end);

int main()
{
        std::vector<int> v = {-5, 4, 12, 3, 0, 24};
        double max;
        double min;
        findMaxMin(max, min, v.begin(), v.end());
        for(auto val : v)
        {
                std::cout << val << " ";
        }
        std::cout<<std::endl;
        std::cout << "min="<<min<<" max="<<max<<std::endl;
        return 0;
}
```

Ćwiczenie 1

Wykorzystaj funkcję **std::rotate** z pliku nagłówkowego $< algorithm >$ aby przesunąć cyklicznie w lewo elementy $std :: vector < double >$ wypełnionego kolejnymi dziesięcioma cyframi tak, żeby cyfra 7 była pierwsza.

Ćwiczenie 2

Wykorzystaj funkcję *std* :: *distance* żeby dla wektora 100 kolejnych liczb naturalnych od 29 począwszy wypisać tylko te liczby, których odległość od początku wektora nie jest podzielna przez 2 ani 3. Przetestuj działanie programu.