Lista zadań nr 5: ***Sortowanie (algorytmy proste)***
(zadania oznaczone „*” wyznaczają minimalny wymagany zbiór zadań do realizacji)

***Zadanie 1.**

Implementując każdy w omawianych na wykładzie *prosty algorytm sortowania* (BubbleSort, SelectSort, InsertSort, …) zbadaj (na stosownych, samodzielnie zdefiniowanych zbiorach danych testowych) liczbę porównań i przesunięć (przepisań) wykonanych w każdym z algorytmów.

W celu przeprowadzenia badań utwórz program pomocniczy, który wspomoże przeprowadzenie badań, poprzez umożliwienie:

- wykonania badań różnych algorytmów sortowania na jednakowych danych (ciągach liczb); uwzględnić przypadki: ciąg „prawie uporządkowany” (zgodnie z kryterium sortowania), ciąg „odwrotnie (prawie) uporządkowany”, ciąg „losowy”,
- gromadzenia informacji o liczbie porównań (przepisań) i przesunięć dla każdego przypadku danych i każdego algorytmu sortowania,
- prezentacji uzyskanych wyników badań, umożliwiających sformułowanie stosownych wniosków.

Uwaga: Program przyda się także do badania zaawansowanych algorytmów sortowania (następna lista zadań).

***Zadanie 2.**

Zaproponuj implementację realizującą wielość alternatywnych porządków zbioru obiektów danego (zaproponowanego przez siebie) typu tak, by możliwe było uzyskiwanie kilku alternatywnych porządków, wyznaczanych przez wartości różnych atrybutów (pól), z użyciem komparatora, komparatora naturalnego oraz komparatora złożonego.

Zadanie 3.

Zaproponuj ogólną, ulepszoną w stosunku do wersji podstawowej, implementację sortowania bąbelkowego z *przechodzeniem* ciągu, na przemian, w obu kierunkach (ShakerSort) tak, by po każdym przejściu (w wyniku którego element maksymalny/minimalny trafia na swoje docelowe miejsce) skracać zakres *przechodzenia pozostałego (nieuporządkowanego) fragmentu* ciągu. Wprowadź zapamiętywanie miejsca ostatniego przestawienia elementów, by przyspieszyć sortowanie.