Programowanie I R

Zadania – seria 2.

Kontenery: listy i krotki.

Zadanie 1. `mostfrequent` – Najczęściej występujący element listy.

Napisz funkcję `most_frequent` przyjmującą jako argument dowolną listę i zwracającą dwuelementową krotkę, której pierwszym elementem ma być ten element tej listy, który występuje najczęściej, zaś drugim – liczba jego wystąpień. W przypadku, gdy elementów takich jest kilka, wynik działania funkcji powinien być listą krotek zbudowanych w analogiczny sposób.

Korzystając z tej funkcji, napisz program `mostfrequent`, który wczytuje ze standardowego wejścia oddzielone spacjami liczby całkowite i wypisuje na standardowe wyjście tę z nich, która powtarza się najczęściej (lub wszystkie takie liczby, gdy jest ich kilka), oraz liczbę jej (ich) wystąpień.

Zadanie 2. `bubblesort` – Sortowanie bąbelkowe.

Sortowanie bąbelkowe to jeden z algorytmów sortowania kolekcji danych. Polega on na wielokrotnym przechodzeniu przez kolekcję i porównywaniu dwóch stojących obok siebie elementów – gdy są one ustawione w złej kolejności, zostają zamienione miejscami. Proces sortowania kończy się, gdy podczas przejścia przez kolekcję nie dokonano ani jednej zamiany.

Napisz funkcję `bubble_sort` przyjmującą jako argument dowolną listę i zwracającą listę złożoną z posortowanych w kolejności rosnącej elementów wyjściowej listy. Załóż, że elementy listy dopuszczają operację porównywania (`<`). Funkcje powinna wykorzystywać algorytm sortowania bąbelkowego.

Korzystając z tej funkcji, napisz program `bubblesort`, który wczytuje ze standardowego wejścia oddzielone spacjami liczby całkowite i wypisuje na standardowe wyjście te same liczby w kolejności rosnącej.

Zadanie 3. `binsearch` – Wyszukiwanie binarne.

Wyszukiwanie binarne to jeden z algorytmów wyszukiwania elementu w kolekcji danych, oparty na metodzie *divide et impera* (łac. *dziel i rządź*). Można go stosować wyłącznie dla posortowanych kolekcji. Polega on w uproszczeniu na dzieleniu kolekcji na coraz mniejsze przedziały tak długo, aż długość przedziału będzie równa 1 – wówczas pojedynczym sprawdzeniem można ustalić, czy szukany element należy do konkretnego przedziału. Algorytm ten można realizować rekurencyjnie.

Napisz funkcję `bin_search` przyjmującą cztery argumenty: pierwszym ma być poszukiwany element, drugim – dowolna lista, zaś trzecim i czwartym – liczby całkowite `min` i `max`. Funkcja powinna posortować listę, a następnie zwrócić indeks, pod którym szukany element występuje na tej liście, biorąc pod uwagę tylko indeksy od `min` do `max` włącznie. Jeśli szukany element występuje wielokrotnie, funkcja może zwrócić dowolny z jego indeksów, jeśli zaś nie występuje wcale, funkcja powinna zwrócić `-1`.

Korzystając z tej funkcji, napisz program `binsearch`, który przyjmuje jako argument wywołania jedną liczbę całkowitą oraz wczytuje ze standardowego wejścia oddzielone spacjami liczby całkowite i wypisuje na standardowe wyjście położenie przekazanej jako argument wywołania liczby wśród posortowanych rosnąco liczb wczytanych ze standardowego wejścia.

Opracowanie: Bartłomiej Zglinicki.