Algorytmy i struktury danych I – laboratorium, zajęcia 2

Poprzednie zajęcia:

- Napisać funkcję **`SequentialSearch`** realizującą algorytm wyszukiwania sekwencyjnego według schematu blokowego z prezentacji. Wartość elementu do wyszukiwania, nazwa tablicy oraz rozmiar tablicy powinny być podawane jako argumenty funkcji.

Zadania zajęć 2

- 1. Uzupełnić napisaną funkcję **`SequentialSearch`** realizującą algorytm wyszukiwania sekwencyjnego o zwracanie (przez wskaźnik lub przez referencję) liczby porównań elementów tablicy z elementem poszukiwanym wykonanych podczas pracy danego wywołania funkcji. Zliczane mają być tylko porównania `tab[i]=s`. Zademonstrować w programie głównym (w funkcji `main`) jak działa tak zmodyfikowana funkcja - ilu porównań potrzeba do znalezienia podanego z klawiatury elementu.

- 2. Napisać funkcję **`LosujIndeks`** losującą liczbę pseudolosową typu int z zakresu od 0 do length-1 (gdzie length jest argumentem funkcji typu int) i zwracającą wylosowaną liczbę.

Uwaga: Proszę nie używać `r = rand() % length`, mimo że takie rozwiązanie wydaje się najprostsze!
Losowanie powinno polegać na wywoływaniu wewnątrz pętli **do...while** polecenia `r = rand();` tak długo jak długo otrzymywane wartości r są **większe** od length-1. Kiedy wylosujemy liczbę mieszczącą się w zakresie 0...length-1, kończymy pętlę i zwracamy liczbę. To umożliwia poprawne losowanie z podanego zakresu (chodzi o jednakowe prawdopodobieństwo uzyskania każdej spośród liczb z wybranego zakresu).

Wyjaśnienie: celem pisania dalszych funkcji będzie badanie zachowania funkcji **`SequentialSearch`**. Chodzi o to, aby określić jak liczba porównań potrzebnych do znalezienia elementu w tablicy metodą przeszukiwania sekwencyjnego zależy od rozmiaru tablicy (załóżmy, że chodzi nam o tablice o rozmiarach od 10 do 1000, z krokiem równym 10, czyli length = 10, 20, 30, ... , 100, 110, ... , 990, 1000). Aby określić to nie wystarczy zbadać po razie tablice o każdym z rozmiarów. Niezbędne jest, aby dla konkretnego rozmiaru tablicy (powiedzmy length = 1000) wielokrotnie (powiedzmy max = 1000 razy) wylosować tablicę o rozmiarze length, następnie

wywołać funkcję **SequentialSearch** i wyszukać w tej tablicy element. Wartość szukanego elementu powinna być zawartością komórki tablicy o indeksie zwróconym przez funkcję **LosujIndeks**. Chodzi o to, aby za każdym razem szukany element na pewno był w tablicy! Każde wywołanie funkcji **SequentialSearch** zwróci przez wskaźnik lub referencję liczbę wykonanych porównań. Trzeba z tych liczb policzyć średnią (czyli policzyć sumę liczb porównań i podzielić przez max – pamiętając, aby wynik był liczbą typu `float`). Następnie trzeba policzyć z tych liczb średni kwadrat (czyli policzyć sumę kwadratów liczb porównań i podzielić przez max – pamiętając, aby wynik był liczbą typu `float`).
Uwaga: proszę do zliczeń używać zmiennych typu `long long int`.
Na podstawie średniej i średniego kwadratu należy policzyć odchylenie standardowe. Jeśli `mean` jest średnią liczbą porównań, a `mean2` średnim kwadratem liczby porównań, to odchylenie standardowe wyraża się następującym wzorem: `sd = sqrt(mean2-mean*mean)`. Uwaga: do użycia `sqrt` potrzebna jest biblioteka `cmath`.

Zatem kolejne zadanie jest następujące:

- 3. Napisać funkcję o nazwie **SequentialSearchStatistics** której zadaniem będzie wielokrotne powtórzenie losowania tablicy zadanego rozmiaru oraz wielokrotne wyszukanie elementu w niej przez wywołanie funkcji **SequentialSearch**.

Funkcja **SequentialSearchStatistics** powinna przyjmować argument max określający, ile razy ma być powtórzone losowanie tablicy i wyszukiwanie w niej elementu, a także argument length określający rozmiar badanej tablicy. Funkcja **SequentialSearchStatistics** powinna być typu `void` i zwracać średnią wartość liczby porównań oraz odchylenie standardowe wartości liczby porównań dla tablicy o danym rozmiarze.

Uwaga raz jeszcze: istotne jest, jakiego elementu szukać w każdej z tablic. Proszę zawsze szukać elementu równego elementowi z tablicy pobranemu z losowo wybranej pozycji (czyli spod losowo wybranego indeksu). W tym celu w punkcie 2 potrzebne było napisanie funkcji **LosujIndeks.**

Dalej chodzi jedynie o to, aby zebrać dane dla rozmiarów tablic z zakresu 10...1000. Czyli trzeba wywołać funkcję **SequentialSearchStatistics** dla każdego z rozmiarów tablic i zapisać w pliku tekstowym w kolejnych wierszach następujące dane:

rozmiar tablicy średnia średnia-odchylenie standardowe średnia+odchylenie standardowe

W tym celu napiszemy kolejną funkcję:

- 4. Napisać funkcję **`TestSearch`**, która wywoła funkcję **`SequentialSearchStatistics`** dla rozmiarów tablic równych 10, 20, 30, ... ,1000. Dla każdego z tych rozmiarów tablic powinno zostać wylosowane po 1000 tablic i policzone średnie liczby porównań oraz odchylenie standardowe liczby porównań. Dane wyjściowe dla każdego rozmiaru tablicy powinny zostać zapisane w pliku tekstowym o nazwie „wyszukiwanie.dat", którego poszczególne wiersze powinny zawierać po kolei: rozmiar tablicy, średnią liczbę porównań, średnią liczbę porównań-odchylenie standardowe, średnią liczbę porównań+odchylenie standardowe.

Przykładowy plik danych powinien zawierać tego typu dane:

10	4,9873	2,39139	7,58321
20	9,9865	4,55199	15,421
30	14,9719	6,6006	23,3432

Mając plik można sporządzić w arkuszu kalkulacyjnym (Excel/Libre Office) wykres z użyciem każdej z kolumn danych. Proszę do wykresu używającego kolumny pierwszej i drugiej dodać linię trendu (liniową, z opcją pokazania równania linii trendu – współczynnik przy x powinien być bliski 0,5).

Celem tych zajęć laboratoryjnych jest ostatecznie wykonanie wykresu prezentującego otrzymane dane. Powinien on przedstawiać **zależność średniej liczby operacji porównań potrzebnej do odnalezienia elementu w tablicy metodą przeszukiwania sekwencyjnego od rozmiaru tej tablicy**. Przykładowy wykres o który chodzi jest przedstawiony poniżej.