Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни

«Алгоритми і структури даних»

Виконав:

Студент групи ІМ-42

Лобань Михайло Юрійович

номер у списку групи: 21

Перевірила:

Молчанова А. А.

Київ 2024

Постановка задачі

1. Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) одним з алгоритмів методу лінійного пошуку.
2. Розміри матриці *m* та *n* взяти самостійно у межах від 7 до 10.
3. Виконати тестування та налагодження програми на комп’ютері. При тестуванні програми необхідно підібрати такі вхідні набори початкових значень матриці, щоб можна було легко відстежити коректність виконання пошуку і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

Умова

Варіант 21

Задано матрицю дійсних чисел A[m,n]. При обході матриці по рядках знайти в ній останній максимальний елемент і його місцезнаходження (координати).

Текст програми

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {

    srand(time(0));

    int m, n; //m rows and n columns
    printf("enter amount of rows and columns: ");
    scanf("%d %d", &m, &n);
    float matrix[m][n];

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
```

```
            matrix[i][j] = (float)rand()/RAND_MAX * 20.0 - 10.0;
        }
    }

    printf("\nmatrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%7.3f ", matrix[i][j]);
        }
        printf("\n");
    }

    float max_value = matrix[0][0];
    int result_row, result_column;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if(matrix[i][j] >= max_value) {
                max_value = matrix[i][j];
                result_column = j;
                result_row = i;
            }
        }
    }

    printf("the largest last element: %.3f", max_value);
    printf("\n(row, column): (%d, %d)", result_row, result_column);

    return 0;
}
```

Результати тестування

```
enter amount of rows and columns: 7 7

matrix:
 -4.747   4.658   7.031  -4.869  -5.773   9.588   7.035
  9.684  -9.962  -5.742   5.185  -6.683   1.174   5.875
  1.867   8.868  -2.854  -8.154  -2.984   0.056   6.213
 -0.869   4.569  -9.373  -9.528  -8.563  -6.632   4.143
 -1.023  -7.598  -9.869  -2.431  -0.529  -7.132  -9.882
 -8.224  -0.612  -9.615  -8.884  -7.920  -3.265  -0.932
 -3.623   1.948  -8.521  -2.071   2.111   2.983  -1.928
the largest last element: 9.684
(row, column): (1, 0)
Process returned 0 (0x0)   execution time : 1.542 s
Press any key to continue.
```

```
enter amount of rows and columns: 8 8

matrix:
 -4.687   1.467  -5.867  -4.256   3.093   1.488   4.026  -3.917
 -7.625   1.684   2.477  -5.876   6.549   8.848   8.218   6.337
  8.199   9.222  -9.463  -9.194  -4.764   2.595  -0.501  -0.931
 -2.140  -6.071   3.539   5.822   1.954   5.665   5.080   4.594
 -5.444   3.800  -5.235   7.526  -2.970   0.896  -5.040  -2.630
  1.549   9.115  -6.865   9.858  -3.470   6.423  -4.248  -5.484
 -3.369   9.084   3.287  -7.886   2.956  -6.434  -2.991  -3.179
  9.067   0.988  -1.853   1.213  -1.549   7.225  -9.945  -7.732
the largest last element: 9.858
(row, column): (5, 3)
Process returned 0 (0x0)   execution time : 3.558 s
Press any key to continue.
```

```
enter amount of rows and columns: 9 9

matrix:
 -4.612  -9.241   8.464  -6.147   2.323  -7.438  -3.786   8.190  -4.666
 -0.910  -0.954  -4.855   0.024  -7.388   2.929   8.466  -9.802  -3.435
 -7.002  -3.578   0.000   1.650   7.743   1.761  -3.448  -9.581  -2.244
  7.949  -6.276  -1.538  -5.318  -5.174  -2.337   0.312   5.986  -3.193
  4.710  -1.791   1.164   6.738   2.314   1.841   5.031  -1.458  -6.407
 -2.818   3.699  -8.210  -3.861  -7.441  -1.129  -8.704  -4.288   0.075
 -3.905   4.886   5.447   0.494   2.571  -7.548  -8.819   9.749   0.164
  9.425  -6.448  -4.844   9.235  -2.224   7.469  -0.504   8.916   7.697
  9.600   1.061   9.459   3.055  -5.520   7.808   2.272  -6.228   9.896
the largest last element: 9.896
(row, column): (8, 8)
Process returned 0 (0x0)   execution time : 1.245 s
Press any key to continue.
```

```
enter amount of rows and columns: 10 10

matrix:
 -4.470  -3.459   2.607  -3.365   9.304   2.727  -8.908   1.336  -9.137  -7.337
 -7.363  -2.946  -0.589   9.647   4.626  -0.189  -7.646  -4.979  -7.668  -4.137
 -5.309  -7.485  -1.590   5.738   9.373   9.651   1.161   1.922   6.769   1.849
  4.728  -9.214   1.364  -5.152  -5.682  -0.586  -6.205  -8.917  -0.405  -2.076
  6.376  -4.382  -0.639  -2.073  -9.787   7.283  -9.351  -2.250  -7.937  -9.369
 -4.642  -4.971  -0.978  -5.661   7.546  -2.678   9.211  -8.323  -5.480  -8.654
  6.545   8.676  -5.160   1.481   7.746  -4.386   6.350  -1.856  -0.308   7.450
  7.387   6.683  -4.746   8.826   4.274  -2.600   5.728   9.260   8.226   7.326
 -2.005   4.171  -5.209  -5.613  -5.463   7.341   8.971  -5.423   4.421  -9.559
  9.108   2.752   3.390  -4.726  -7.293   7.702  -8.816  -5.689  -2.718   2.087
the largest last element: 9.651
(row, column): (2, 5)
Process returned 0 (0x0)   execution time : 2.444 s
Press any key to continue.
```

```
enter amount of rows and columns: 10 7

matrix:
 -4.385  -1.366  -8.546   8.180   0.011   5.785  -7.222
  4.510   4.212  -8.694   8.755  -1.790   0.448   3.907
  7.062  -5.149  -3.804  -1.407   0.380  -6.730  -5.708
 -1.186  -7.524   9.838  -7.372  -0.110   0.407  -5.672
  9.036  -3.143   3.488   5.522   6.986   9.849  -0.354
  4.655   7.749  -9.852  -0.228  -5.161   4.610   0.019
  0.716   4.598   4.786   9.457   7.536   3.614  -5.336
  2.985   5.625   8.840  -6.016  -0.402  -6.647   9.080
 -5.411  -0.987  -4.158   6.171   9.371  -2.679  -9.511
  0.896   7.047   5.187   4.884   1.747  -1.638  -8.296
the largest last element: 9.849
(row, column): (4, 5)
Process returned 0 (0x0)   execution time : 1.816 s
Press any key to continue.
```

Висновки

Розв’язав поставлену задачу із використанням методу лінійного пошуку, та показав, що він працює для випадково згенерованих матриць розмірністю m*n.