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 №2.1

з дисципліни
«Алгоритми і структури даних»

Виконав

Студент групи ІМ-34
Никифоров Артем Михайлович
номер у списку групи: 17

Перевірила:

Молчанова А. А.

Київ 2023

Постановка задачі

1. Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) методом двійкового пошуку. Алгоритм двійкового пошуку задається варіантом завдання.

2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.

3. При тестуванні програми необхідно підбирати такі вхідні набори початкових значеннь 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

Варіант № 17

Задано матрицю дійсних чисел $A[m,n]$. Окремо у кожному стовпчику матриці визначити присутність будь-якого з чисел діапазону $[0,5]$ і його місцезнаходження (координати) методом двійкового пошуку (Алгоритм №1), якщо елементи кожного стовпчика окремо впорядковані за незбільшенням.

Текст програми:

```
#include <stdio.h>
int n;
void algoritm1(int A[][n], int m, int x, int j) {
    int L = 0, R = m - 1;
    while (L <= R) {
        int i = (R + L) / 2;
        if (A[i][j] == x) {
            printf("element %d found at position (%d, %d).\n", x, i, j);
            return;
        }
        if (x < A[i][j]) {
            R = i - 1;
        } else {
            L = i + 1;
        }
    }
    printf("element %d not found in column %d.\n", x, j);
}
int main() {
    int m;
    printf("rows (m): ");
    scanf("%d", &m);
    printf("columns (n): ");
    scanf("%d", &n);
    int A[m][n];
    printf("enter matrix elements:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
```

```
            scanf("%d", &A[i][j]);
        }
    }
    printf("matrix elements:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
    for (int j = 0; j < n; j++) {
        for (int element = 0; element <= 5; element++) {
            algoritm1(A, m, element, j);
        }
    }
    return 0;
}
```

Результати тестування програми

```
matrix elements:
-1 2 3 0 -5 6 1
-1 3 0 1 -3 5 6
0 4 0 0 -1 4 6
0 5 0 1 2 3 6
1 6 0 0 3 2 6
1 7 0 1 5 1 6
1 6 0 1 7 0 6
element 0 found at position (3, 0).
element 1 found at position (5, 0).
element 2 not found in column 0.
element 3 not found in column 0.
element 4 not found in column 0.
element 5 not found in column 0.
element 0 not found in column 1.
element 1 not found in column 1.
element 2 found at position (0, 1).
element 3 found at position (1, 1).
element 4 found at position (2, 1).
element 5 found at position (3, 1).
element 0 found at position (3, 2).
element 1 not found in column 2.
element 2 not found in column 2.
element 3 not found in column 2.
element 4 not found in column 2.
element 5 not found in column 2.
element 0 found at position (0, 3).
element 1 found at position (3, 3).
element 2 not found in column 3.
element 3 not found in column 3.
element 4 not found in column 3.
element 5 not found in column 3.
element 0 not found in column 4.
element 1 not found in column 4.
element 2 found at position (3, 4).
element 3 found at position (4, 4).
element 4 not found in column 4.
element 5 found at position (5, 4).
element 0 not found in column 5.
```

```
element 1 not found in column 5.
element 2 not found in column 5.
element 3 found at position (3, 5).
element 4 not found in column 5.
element 5 not found in column 5.
element 0 not found in column 6.
element 1 found at position (0, 6).
element 2 not found in column 6.
element 3 not found in column 6.
element 4 not found in column 6.
element 5 not found in column 6.
```

```
matrix elements:
1 0 -1
0 -1 -2
-1 -2 -3
element 0 found at position (1, 0).
element 1 not found in column 0.
element 2 not found in column 0.
element 3 not found in column 0.
element 4 not found in column 0.
element 5 not found in column 0.
element 0 not found in column 1.
element 1 not found in column 1.
element 2 not found in column 1.
element 3 not found in column 1.
element 4 not found in column 1.
element 5 not found in column 1.
element 0 not found in column 2.
element 1 not found in column 2.
element 2 not found in column 2.
element 3 not found in column 2.
element 4 not found in column 2.
element 5 not found in column 2.
```

```
matrix elements:
1
1
1
2
2
3
4
4
5
5
element 0 not found in column 0.
element 1 found at position (1, 0).
element 2 found at position (4, 0).
element 3 found at position (5, 0).
element 4 found at position (7, 0).
element 5 found at position (8, 0).
```

```
matrix elements:
0 -1 -1 4 5
1 -1 -1 3 6
2 -1 -1 2 6
3 -1 -1 1 6
4 -1 1 0 6
element 0 found at position (0, 0).
element 1 found at position (1, 0).
element 2 found at position (2, 0).
element 3 found at position (3, 0).
element 4 found at position (4, 0).
element 5 not found in column 0.
element 0 not found in column 1.
element 1 not found in column 1.
element 2 not found in column 1.
element 3 not found in column 1.
element 4 not found in column 1.
element 5 not found in column 1.
element 0 not found in column 2.
element 1 found at position (4, 2).
element 2 not found in column 2.
element 3 not found in column 2.
element 4 not found in column 2.
element 5 not found in column 2.
element 0 not found in column 3.
element 1 not found in column 3.
element 2 found at position (2, 3).
element 3 not found in column 3.
element 4 not found in column 3.
element 5 not found in column 3.
element 0 not found in column 4.
element 1 not found in column 4.
element 2 not found in column 4.
element 3 not found in column 4.
element 4 not found in column 4.
element 5 found at position (0, 4).
```