emter

INFORMATIKA – tétel

2021

JAVÍTÓKULCS

ELIGAZÍTÁS:

- 1 pont hivatalból;
- Használj alprogramot, valahányszor célszerűnek találod. Törekedj **hatékony** megoldásra az algoritmusok időigénye tekintetében. **Lásd el beszédes kommentekkel programjaidat**.
- A bemeneti adatok helyesnek tekinthetők.
- Minden egész érték eltárolható 32 bites egészként.

FELADATOK:

1. [2 pont] Írj Pascal vagy C/C++ programot, amely billentyűzetről beolvas egy `n` értéket (`3≤n≤20`) majd kiírja a képernyőre az alábbi példák szerinti (2n-1) `x n` méretű bináris mátrixot („nyílhegy mátrix”).

Példák:

n=5 esetre

```
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
1 1 1 1 0
1 1 1 0 0
1 1 0 0 0
1 0 0 0 0
```

n=4 esetre

```
1 0 0 0
1 1 0 0
1 1 1 0
1 1 1 1
1 1 1 0
1 1 0 0
1 0 0 0
```

```
int n;
scanf("%i", &n);
for( int i = 0 ; i < 2*n-1 ; ++i ){
    for( int j = 0 ; j < n ; ++j ){
        if( i >= j && i+j < 2*n-1 ){
            printf("1 ");
        }
        else{
            printf("0 ");
        }
    }
    printf("\n");
}
```

2. Írj Pascal vagy C/C++ programot, amely a `matrix.txt` állományból beolvassa az `n` értéket (`3≤n≤10`), valamint egy `nxn` méretű mátrixot (elemei *különböző* egész számok), majd megvalósítja a következő algoritmust: (1) az 1. sorban minimumot keresünk, (2) e minimum érték oszlopában maximumot keresünk, (3) e maximum érték sorában minimumot, (4) e minimum érték oszlopában maximumot keresünk, stb. Addig folytatjuk ezt, amíg ciklusba nem kerülünk, azaz újra egy olyan sorban kell minimumot keressünk, ahol korábban már kerestünk, vagy újra egy olyan oszlopban kell maximumot keressünk, ahol korábban már kerestünk.

(a) [2 pont] Írasd ki a képernyőre, hogy hány keresésre került sor.

Példa (minimumkeresés az 1. sorban; maximumkeresés a 2. oszlopban; minimumkeresés az 3. sorban):

INPUT	OUTPUT
3	3

```
7 1 3
4 2 6
9 5 8
```

```
int minpoz(int a[][11], int n, int i){
    int minp = 0;
    for( int j = 1 ; j < n ; ++j ){
        if( a[i][j] < a[i][minp] ){
            minp = j;
        }
    }
    return minp;
}
```

```
int maxpoz(int a[][11], int n, int j){
    int maxp = 0;
    for( int i = 1 ; i < n ; ++i ){
        if( a[i][j] > a[maxp][j] ){
            maxp = i;
        }
    }
    return maxp;
}
```

```
    int n, a[11][11];
    FILE *fin = fopen("matrix2.txt", "rt");
    fscanf(fin, "%i", &n);
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
            fscanf(fin, "%i", &a[i][j]);
        }
    }
    int x[11] = {0}, y[11] = {0};
    int i = 0, j, k = 0;
    x[i] = 1;
    for( ; ; ){
        j = minpoz(a,n,i); ++k;
        ++y[j]; if( y[j] > 1 ) { break; }
        i = maxpoz(a,n,j); ++k;
        ++x[i]; if( x[i] > 1 ) { break; }
    }
    printf("%i", k);
```

(b) [1 pont] A beolvasott `n`-re, jeleníts meg a képernyőn egy olyan `nxn` méretű, különböző elemeket tartalmazó mátrixot, amely esetében $2n$ keresésre kerül sor (indokold meg a válaszod).

```
    int n, a[11][11] = {0}, k = 0;
    scanf("%i", &n);
    for( int i = 0 ; i < n ; ++i ){
        a[i][i] = ++k;
    }
    k = 100;
    for( int i = 1 ; i <= n ; ++i ){
        a[i%n][i-1] = ++k;
    }
    k = 10;
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
            if( a[i][j] == 0 ){
```

```
                a[i][j] = ++k;
            }
        }
    }
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
            printf("%4i", a[i][j]);
        }
        printf("\n");
    }
```

3. [2 pont] Írj Pascal vagy C/C++ programot, amely a `matrix.txt` állományból beolvassa az `n` értéket (`3≤n≤51`), valamint egy `nxn` méretű *bináris* mátrixot. Írasd ki a képernyőre, hogy a mátrix hány sora, oszlopa, balátlója (főátlóval párhuzamos átló), illetve jobbátlója (mellékátlóval párhuzamos átló) „foglalt”, azaz tartalmaz 1-es értéket.

Példa:

INPUT	OUTPUT
4 0 1 1 0 1 1 0 1 0 0 0 0 0 1 0 0	3 4 5 3

```
    int n, x, sor[51]={0}, oszlop[51]={0};
    int balatlo[101]={0}, jobbatlo[101]={0};
    FILE *fin = fopen("matrix3.txt", "rt");
    fscanf(fin, "%i", &n);
    int k_sor=0, k_oszlop=0, k_balatlo=0, k_jobbatlo=0;
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
            fscanf(fin, "%i", &x);
            if( x == 0 ) { continue; }
            ++sor[i]; if( sor[i] == 1 ){ ++k_sor; }
            ++oszlop[j]; if( oszlop[j] == 1 ){ ++k_oszlop; }
            ++balatlo[i-j+n-1];
            if( balatlo[i-j+n-1] == 1 ){ ++k_balatlo; }
            ++jobbatlo[i+j];
            if( jobbatlo[i+j] == 1 ){ ++k_jobbatlo; }
        }
    }
    printf("%i %i %i %i", k_sor, k_oszlop, k_balatlo, k_jobbatlo);
```

4. [2 pont] Adottak az `n` és `m` értékek (`3≤n,m≤100`), valamint egy `nxm` méretű *bináris* mátrix. Az 1es értékek focijátékosokat szemléltetnek, akik balról jobbra (az első oszlop felől az utolsó oszlop felé) támadnak. Az első oszlop egyetlen 1est tartalmaz, amely a kapust szemlélteti. Mindenik játékos csak előre passzolhat, azaz tőle jobbra lévő oszlopban lévő játékosnak. Az utolsó oszlop nem tartalmaz 1eseket, ide képzeljük el az ellenfél kapuját. Minden játékos (a kapus is) dönthet úgy, hogy passzol, vagy közvetlenül kapura lő. Írj Pascal vagy C/C++ programot, amely a `matrix.txt` állományból beolvassa az `n` és `m` értékeket és a mátrixot, majd kiírja a képernyőre, hogy hány különböző módon juthat el a labda a kapustól az ellenfél kapujába.

Példa (a félkövér 1es a kapus; a szürke hátterű oszlop, az ellenfél kapujának oszlopa):

INPUT	OUTPUT
4 6 0 1 0 0 0 0 **1** 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0	12

Magyarázat: a 12 lehetséges útvonal a kapustól ([2,1]) az ellenfél kapujáig ([x,6])

1. [2,1], [x,6]

2. [2,1], [1,2], [x,6]
3. [2,1], [4,3], [x,6]
4. [2,1], [1,2], [4,3], [x,6]
5. [2,1], [2,4], [x,6]
6. [2,1], [1,2], [2,4], [x,6]
7. [2,1], [4,3], [2,4], [x,6]
8. [2,1], [1,2], [4,3], [2,4], [x,6]
9. [2,1], [4,4], [x,6]
10. [2,1], [1,2], [4,4], [x,6]
11. [2,1], [4,3], [4,4], [x,6]
12. [2,1], [1,2], [4,3], [4,4], [x,6]

```
int n, m, a[101][101];
FILE *fin = fopen("matrix4b.txt", "rt");
fscanf(fin, "%i%i", &n, &m);
for( int i = 0 ; i < n ; ++i ){
    for( int j = 0 ; j < m ; ++j ){
        fscanf(fin, "%i", &a[i][j]);
    }
}
int kumulalt = 1, kurrens;
for( int j = 1 ; j < m-1 ; ++j ){
    kurrens = 0;
    for( int i = 0 ; i < n ; ++i ){
        if( a[i][j] ) { kurrens += kumulalt; }
    }
    kumulalt += kurrens;
}
printf("%i", kumulalt);
```