Programozási paradigmák és technikák
ZH minta

1. Feladat:

Írjon programot, amely List<> tárolóban tárolt körökkel statisztikákat és számításokat végez!

Definiáljon egy **EgyKor** nevű struktúrát három egész típusú adatmezővel, az első kettő szám a kör középpontjának az „x” és „y” koordinátái, a harmadik szám a kör sugara „r”.
A struktúrának legyen paraméteres konstruktora, amelyben az adatmezők értékeit az átvett paraméterekkel inicializálja.
Minden adatmezőnek legyen lekérdező tulajdonsága is.

Definiáljon egy **KorStat** nevű osztályt, amelynek privát adattagja a következő:

korList Deklaráljon egy **List<>** generikus gyűjteményt, **EgyKor** típusú struktúrák tárolására.

A metódusai pedig a következők:

paraméteres konstruktor: Hozza létre a List<> tárolót. A konstruktor paraméterként vegye át a körök számát, majd ennyi elemmel töltse fel véletlenszerűen a listát. Az x és y koordináták -20 és 20 közötti értékek legyenek, a sugár pedig 0 és 10 közötti legyen.

indexelő: Készítsen egy indexelőt, amely visszaadja a korList i-edik elemét. (EgyKor típusú struktúrát ad vissza.)

BenneOrigo(); Adja vissza, hány olyan kör van, amelyik tartalmazza az origót, vagyis a középpont távolsága az origótól kisebb vagy egyenlő, mint a sugár. Az origó és a középpont távolsága=$\sqrt{x^2 + y^2}$.

TeruletOsszeg(); Adja vissza a körök területének az összegét. (A kör területe: T = r*r*Pi)!

LegtavolabbiKorok(); A metódus keresse meg a két egymástól legtávolabbra lévő kört. Írja ki a két kör sorszámát és a távolságukat két tizedesjegyre kerekítve a minta szerint! A feladatban két kör távolságán a két kör legközelebbi pontjainak a távolságát értjük, vagyis a két középpont távolságából ki kell vonni a két sugarat.
A két középpont távolsága = $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$.

A(z) 10. és a(z) 11. kör körvonala van a legtávolabb!
Távolságuk: 287,40

A **Main()** metódusban kérje be a körök számát ellenőrzötten (try), ha a felhasználó nem számot, vagy nem pozitív számot ad meg, adjon hibajelzést és állítsa be a körök számát 10-re!

Definiáljon egy **KorStat** típusú objektumot, **korStat** néven,
majd hívja meg a metódusait! Ha a metódusnak van visszatérési értéke, azt írassa is ki!
Írassa ki az indexelő felhasználásával az összes kör adatát **(x,y) - r** formában.

2. Feladat:

Írjon programot, amely körökkel statisztikákat és számításokat végez!

Definiáljon egy **Kor** nevű struktúrát három egész típusú adattaggal, az első kettő szám a kör középpontjának az „x” és „y” koordinátái, a harmadik szám a kör sugara „r”.
A struktúrának legyen paraméteres konstruktora, amelyben az adattagok értékeit az átvett paraméterekkel inicializálja.
Minden adattagnak legyen lekérdező tulajdonsága is.

Definiáljon egy interfészt **IStatisztika** néven, amely két metódusdeklarációt tartalmaz:
Legtávolabb(); nem vesz át semmit, és egy Kor típusú struktúrát ad vissza.
Kicsik(); egy egész értéket vesz át, és egy egész értéket ad vissza.

Definiáljon egy **Korok** nevű osztályt, amely megvalósítja az **IStatisztika** nevű interfészt, és privát adattagja a következő:

korTmb egy **Kor** típusú struktúrák tárolására alkalmas egydimenziós tömbreferencia.

A metódusai pedig a következők:

paraméteres konstruktor: A konstruktor paraméterként vegye át a körök számát, hozzon létre ennyi elemű tömböt, majd töltse fel véletlenszerűen a listát. Az x és y koordináták -100 és 100 közötti értékek legyenek, a sugár pedig 0 és 150 közötti legyen.

TengelyenK(); adja vissza azoknak a pontoknak a számát, amelyek középpontja valamelyik tengelyen van (a középpont valamelyik koordinátája 0).

Legtávolabb(); nem vesz át semmit, és az origótól legtávolabbi kört adja vissza. Az origó és a kör távolsága=$\sqrt{x^2 + y^2} - r$.

Kicsik(); egy pozitív egész sugár értéket vesz át, és visszaadja azoknak a köröknek a számát, amelyek az átvett sugárnál kisebb sugarúak.

A **Main()** metódusban kérje be a körök számát ellenőrzötten (try), ha a felhasználó nem számot, vagy nem pozitív számot ad meg, adjon hibajelzést és állítsa be a körök számát 15-re!

Definiáljon egy **Korok** típusú objektumot **proba** néven,
majd hívja meg a metódusait! Ha a metódusnak van visszatérési értéke, azt írassa is ki! Ha a **Kicsik()** metódus visszatérési értéke nagyobb, mint 0, akkor írassa ki a kicsik számát, ha 0, akkor pedig írassa ki, hogy *„nincsen kicsi kör”*.

3. Feladat:

Írjon programot, amely körökkel számításokat végez, illetve a sugár szerint sorba rendezi a köröket!

Definiáljon egy **Kor** nevű osztályt három egész típusú adattaggal, az első kettő szám a kör középpontjának az „x” és „y” koordinátái, a harmadik szám a kör sugara „r”.
Az osztálynak legyen **paraméteres konstruktora**, amelyben az adattagok értékeit az átvett paraméterekkel inicializálja.
Minden adattagnak legyen lekérdező tulajdonsága is.

A metódusai pedig a következők:

CompareTo(); Az osztály objektumai legyenek összehasonlíthatóak. Adja meg a megfelelő interfészt és valósítsa meg annak CompareTo() metódusát úgy, hogy a körök a sugaruk szerint növekvő sorrendbe rendezhetők legyenek.

ToString(); Definiálja át a ToString() metódust úgy, hogy „K(3,5) R(6)” alakban írja ki a kör adatait.

OrigoTavolsag(); Adja vissza a kör origótól való távolságát. Az origó és a kör távolsága=$\sqrt{x^2 + y^2} - r$.

A **Main()** metódusban kérje be a körök számát ellenőrzötten, legalább 3 legyen (try), ha a felhasználó nem számot, vagy nem 2-nél nagyobb számot ad meg, adjon hibajelzést és addig kérje be az adatot, amíg jó értéket nem ad meg a felhasználó!
Definiáljon egy **Kor** típusú objektumtömböt a beolvasott elemszámmal, majd ennyi elemre töltse fel a tömböt billentyűzetről beolvasott értékekkel.
Az Array osztály Sort() metódusával **rendezze** a tömböt, majd foreach ciklusban írja ki a körök adatait és az origótól való távolságot is.
Keresse meg azt a kört, amelyik az origótól **legtávolabb** van, és írassa ki az adatait!

Segédlet a feladat megoldásához: