5. óra, önálló feladat

A feladat 2D ponthalmazokat tárolni különböző szerepekben.

A *-gal jelöl feladatok megoldása nélkül is lehet tovább haladni, ezek gyakran nehezebbek is.

Pont osztály

Adott a **Pont** osztály két adattaggal: **x**, **y** (**double**), ezek konstruktorban megadandók, getterekkel lekérdezhetők, van továbbá saját *kiir* függvénye is, amely tetszőleges kimeneti folyamra használható.

1. Legyen a **Pont** osztályban paraméter nélküli konstruktor, ami az origót $(0,0)$ állítja be.
2. *Legyen a **Pont** osztályban egy **statikus M adattag** (**double**), aminek értéke kezdetben 10^6, és a *setM* **statikus** setter függvénnyel legyen beállítható, a *getM* **statikus** getter függvénnyel lekérdezhető.
3. *Az **M** egy felső korlátot határoz meg a **Pont**-ok koordinátáinak az abszolút értékére. Valahányszor olyan **Pont** objektum konstruálódik, aminek van **M**-nél nagyobb abszolút értékű koordinátája (vagyis nem teljesül rá, hogy $-M \leq x, y \leq M$), akkor a konstruktor írjon ki egy egysoros figyelmeztetést, ami ezt közli, és szerepeljen az üzenetben a pont két koordinátája is valamilyen módon.
4. *Írj másoló konstruktort is a **Pont** osztályhoz, ami szintén figyeli az **M** korlátját.
5. *Az említett figyelmeztetések legyenek kiírva egy **warnings.log** nevű fájba is. A fájl **végéhez illesztve** kell írni, tehát **ne törlődjön** a fájl addigi tartalma (tipp: **std::ios::app**).

Halmaz – általános tároló osztály

6. Legyen egy **Halmaz** osztály, aminek két adattagja van: egy megnevezés (**string**), és egy pontokat tároló **vector<Pont>**. A konstruktor csak egy paramétert, a megnevezést kapja meg, a tároló kezdetben legyen üres.
7. Legyen olyan konstruktor is a **Halmaz** osztályban, ami két paramétert kap: a megnevezésen kívül egy elemszámot is, és a ponthalmaznak kezdetben ennyi eleme legyen. Az elemeket nem kell beállítani, az alapértelmezett origó $(0,0)$ fog érvényesülni. (Megoldható az előző feladattal együtt, egyetlen konstruktorral.)
8. Legyen a **Halmaz** osztályban egy *hozzaad* függvény, ami egyetlen paraméterben egy **Pont**-ot kap és a vektor végére beszúrja azt.
9. Terheld túl a *hozzaad* függvényt: legyen olyan változata is, ami két **double** értéket kap paraméterben, az új **Pont** **x** és **y** koordinátáját.
10. Legyen a **Halmaz** osztályban egy *kiir* függvény, ami egyetlen sorban kiírja a halmaz megnevezését, a tárolt pontok számát, majd az egyes pontokat, sortöréssel a legvégén (a formátumhoz lásd a példakimenetet).
11. *Legyen a **Halmaz** osztályban egy *beolvas* függvény, ami paraméterben egy fájl elérési útját kapja meg (**string**), és onnan olvassa be a pontokat. A függvény először törölje a meglévő pontokat. A fájlban először a pontok száma szerepel, majd a pontok egyenként, mindegyiknek az **x** és **y** koordinátája, kizárólag whitespace-szel elválasztva. **Fontos:** A teszteléshez a mellékelt **halmaz3.txt** fájlt a projekt build könyvtárába kell másolni.

12. *Legyen a **Halmaz** osztályban egy *fajlbair* függvény, ami paraméterben egy fájl elérési útját kapja meg. Mentse el a pontokat a megadott fájlba (felülírva annak korábbi tartalmát) olyan formátumban, hogy ugyanaz a fájl a *beolvas* függvénnyel beolvasható legyen.

Utvonal gyerekosztály

13. Legyen egy **Utvonal** osztály a **Halmaz** osztályból származtatva. Legyen konstruktor, ami nem vár paramétert, az ősosztálynak átadott megnevezés a **"PATH"** legyen, és kezdetben a ponthalmaz üres. Legyen továbbá egy irányt jelölő logikai adattag, ami kezdetben **true**.
14. *Legyen egy *megfordit* függvény az **Utvonal** osztályban, ami az irányt jelölő logikai változót az ellentettjére állítja.
15. *Írd felül a *kiir* függvényt az **Utvonal** osztályban úgy, hogy ha az irányt jelölő logikai változó értéke **false**, akkor a pontok fordított sorrendben kerüljenek kiírásra (a tárolási sorrend nem változik).
16. *Legyen egy *hossz* függvény az **Utvonal** osztályban, ami visszaadja a reprezentált útvonal hosszát (**double**). Ha kettőnél kevesebb pont van a halmazban, akkor a hossz 0, egyébként a szomszédos pontok távolságainak az összege. **Tipp:** használható a **Pont** osztályból a statikus *tavolsag* függvény két pont távolságának kiszámítására.

Haromszog gyerekosztály

17. Legyen egy **Haromszog** osztály a **Halmaz** osztályból származtatva. Ne legyen saját adattagja. Legyen egy konstruktora, ami három **Pont** paramétert vár, ezek a háromszög csúcsai. A ponthalmaz ebből a három **Pont**-ból álljon. Az ősosztálynak átadott megnevezés a **"TRIANGLE"** legyen.
18. *Legyen a **Haromszog** osztályban egy *getA*, *getB* és *getC* függvény, amik visszaadják a háromszög csúcsait. A csúcsokat az ősosztályban tárolt háromelemű vector<Pont> tárolóból kell kinyerni.
19. *Legyen egy *alak* függvény a **Haromszog** osztályban, ami egy **string**-et ad vissza. Ez a **"hegyesszogu"**, **"derekszogu"** és **"tompaszogu"** valamelyike lehet az alábbiak szerint. Tegyük fel, hogy a háromszög oldalai v, w és z, amik közül z a leghosszabb. Számítsuk ki a $D = z^2 - (v^2 + w^2)$ kifejezést.
    - Ha $D < -10^{-8}$, akkor háromszög hegyesszögű.
    - Ha $-10^{-8} \leq D \leq 10^{-8}$, akkor a háromszög derékszögű.
    - Ha $10^{-8} < D$, akkor a háromszög tompaszögű.

    Itt is használható a **Pont** osztály statikus *tavolsag* függvénye.

Teglalap gyerekosztály

20. Legyen egy **Teglalap** osztály a **Halmaz** osztályból származtatva, ami egy **tengelypárhuzamos** téglalapot fog reprezentálni. Ne legyen saját adattagja. Legyen egy konstruktora, ami paraméterben a téglalapot meghatározó négy **double** értéket vár: x_{min}, x_{max}, y_{min}, y_{max}. Feltételezhetjük, hogy $x_{min} < x_{max}$ és $y_{min} < y_{max}$, ezt nem kell külön ellenőrizni. Az ősosztálynak átadott megnevezés legyen **"RECTANGLE"**, és a ponthalmaz a fenti határok által meghatározott téglalap négy csúcsából álljon (tetszőleges sorrendben), vagyis: (x_{min}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{max}), (x_{max}, y_{min}).
21. *Legyen a **Teglalap** osztályban egy *terulet* függvény, ami visszaadja a téglalap területét (**double**).
22. *Legyen a **Teglalap** osztályban egy *tartalmaz* függvény, ami paraméterben egy **Pont**-ot kap, és visszaad egy **bool** értéket aszerint, hogy a **Pont** a téglalapon belül van-e. A téglalap kerületén lévő pontok belsőnek számítanak.

Sokszog gyerekosztály

23. Legyen egy **Sokszog** osztály a **Halmaz** osztályból származtatva, ami tetszőleges számú csúcsból álló sokszöglapot fog reprezentálni. Ne legyen saját adattagja. Legyen egy konstruktora, aminek ezúttal egy paramétere van: várja a megnevezést (**string**), ezt továbbítja az ősosztálynak. A ponthalmaz kezdetben üres.
24. ***Legyen egy *kerulet* függvény a **Sokszog** osztályban, ami visszaadja a sokszög kerületét. A számítás módja ugyanaz, mint az **Utvonal** osztály *hossz* függvénye esetén, csak az első és utolsó pont távolságát is az összeghez kell adni.
25. ***Legyen egy *bennfoglalo* függvény a **Sokszog** osztályban, ami nem vár paramétert, és egy **minimális területű** **Teglalap** objektumot ad vissza, ami tartalmazza a **Sokszog**-et.