2. beadandó feladat: Típus megvalósítása

Közös követelmények:

- A feladatokat saját típus segítségével kell magvalósítani, a lehetőségek maximális kihasználtsága mellett. A típust külön fordítási egységben kell létrehozni, belső megvalósítását el kell rejteni a külvilág elől (de a megfelelő értékekre biztosítani kell lekérdező, esetlegesen beállító műveleteket). Ahol értelemszerű, valósítsuk meg a műveleteket operátorok segítségével. Minden esetben gondoskodjunk a kényelmi beolvasásról és kiírásról operátorok segítségével.

- A program legyen informatív, és adjon lehetőséget az adatok billentyűzetről, valamint fájlból történő betöltésére is, továbbá ellenőrizze, hogy a beolvasott adat megfelelő-e.

- A program használjon alprogramokat az egyes funkciók (beolvasás, kiírás, feldolgozás) megvalósítására. Az alprogramok a kommunikációhoz használjanak paraméterátadást és/vagy visszatérési értéket, ne legyen a programban globális változó.

- A dokumentáció tartalmazza a feladat leírását és elemzését (a típus felületét, műveleteinek szintaxisát), a programszerkezet leírását (a forrásfájlok kapcsolatát, a típus reprezentációs módját, a típuson kívüli alprogramok szintaxisát és kapcsolatait), valamint a teszteseteket.

Feladatok:

1. Valósítsuk meg a komplex számok típusát, ahol a komplex számokat az exponenciális alakjukkal ábrázoljuk ($\lambda e^{i\varphi}$). Implementáljuk komplex számok szorzását, osztását, hatványozását. A főprogram ilyen komplex számokkal töltsön fel egy t tömböt, amelyre ezután ki kell számolni a $\prod t[i]^i$ értéket.

2. Valósítsuk meg a kvaterniók típusát, ahol a kvaterniókat az algebrai alakjukkal ($a + ib + jc + kd$) ábrázoljuk. Implementáljuk kvaterniók összeadását, szorzását valamint negálását. A főprogram ilyen kvaterniókkal töltsön fel egy tömböt, amelynek ezután ki kell számolni az összegét és szorzatát.

3. Valósítsuk meg a kör típust, ahol a kört a középpontjával és sugarával reprezentáljuk. Legyen lehetőség a terület és a kerület lekérdezésére, illetve annak megvizsgálására, hogy egy adott pont benne van-e egy körben. Ehhez valósítsuk meg a síkbeli pont típusát is. A főprogram rögzítsen egy síkbeli pontot, töltsön fel egy tömböt körökkel, majd minden olyan kör paramétereit írja ki, amely a rögzített pontot tartalmazza.

4. Valósítsuk meg a kör típust, ahol a kört a középpontjával és sugarával reprezentáljuk. Legyen lehetőség a terület és a kerület lekérdezésére, valamint a körök eltolására

adott vektorral. Ehhez valósítsuk meg a síkbeli pont, valamint a vektor típusait is, amelyek biztosítják pont vektorral való eltolását. A főprogram rögzítsen egy síkvektort, töltsön fel egy tömböt körlemezekkel, majd minden körlemezt toljon el az adott vektorral, és írja ki az eltolt körök paramétereit.

5. Valósítsuk meg az azonos állású négyzetek típusát. Ennek értékei a sík olyan négyzetei, amelyek közül bármelyik kettőnek egy-egy oldala vagy párhuzamos, vagy merőleges. A reprezentációhoz képzeljen el a négyzetek oldalaival párhuzamos, illetve merőleges tengelyű derékszögű koordináta rendszert, és tegye fel, hogy csak az első sík-negyedbe (ahol a pontok koordinátái nem negatívok) eső négyzetekkel van dolgunk. Egy négyzetet létre lehet hozni a két ellentétes csúcsa alapján, továbbá le lehet kérdezni a területét és kerületet, valamint meg lehet vizsgálni, hogy egy adott pont benne van-e. Ehhez valósítsuk meg a síkbeli pont típusát is. A főprogram rögzítsen egy síkbeli pontot, töltsön fel egy tömböt négyzetekkel, majd minden olyan négyzet paramétereit írja ki, amely a rögzített pontot tartalmazza.

6. Valósítsuk meg az azonos állású négyzetek típusát. Ennek értékei a sík olyan négyzetei, amelyek közül bármelyik kettőnek egy-egy oldala vagy párhuzamos, vagy merőleges. A reprezentációhoz képzeljen el a négyzetek oldalaival párhuzamos, illetve merőleges tengelyű derékszögű koordináta rendszert, és tegye fel, hogy csak az első sík-negyedbe (ahol a pontok koordinátái nem negatívok) eső négyzetekkel van dolgunk. Egy négyzetet létre lehet hozni az oldalhossz és az origóhoz legközelebb eső csúcsa alapján, és el lehet tolni egy vektorral. Ehhez valósítsuk meg a síkbeli pont, valamint a vektor típusait is, amelyek biztosítják pont vektorral való eltolását. A főprogram rögzítsen egy síkvektort, töltsön fel egy tömböt négyzetekkel, majd minden négyzetet toljon el az adott vektorral, és írja ki az eltolt négyzetek paramétereit.

7. Valósítsuk meg a derékszögű háromszögek típusát, amelyeket a két befogólyuk hosszával reprezentálunk. Tegyük fel, hogy csak a koordinátarendszer első sík-negyedébe (ahol a pontok koordinátái nem negatívok) eső háromszögekkel van dolgunk. A derékszög mindig a bal alsó sarokban található, a befogók párhuzamosak a koordinátatengelyekkel. Egy háromszögnek le lehet kérdezni a szögeit, az átfogóra állított magasságának hosszát, a területét és a kerületét. A program olvassa be derékszögű háromszögek sorozatát, írja ki a beolvasott háromszögek adatait, valamint keresse meg, melyik háromszög rendelkezik a legkisebb területtel.

8. Valósítsuk meg a derékszögű trapézok típusát, amelyeket három oldalhosszal reprezentálunk (a program rögzítheti, melyik három oldal értékeit használja minden trapézra). Egy trapéznek le lehet kérdezni az oldalainak hosszát, a szögeit, a területét és a kerületét. A főprogram töltsön fel egy tömböt trapézokkal, és adja meg, melyik trapéznak a legkisebb a legnagyobb szöge.

9. Valósítsuk meg a konvex deltoidok típusát, amelyeket az átlójuk hosszával, és azzal az aránnyal reprezentálunk, amely megmutatja, hogy hol metszi egyik átló a másikat.

Egy deltoidnak le lehet kérdezni az oldalainak hosszát, a szögeit, valamint a területét és a kerületét. A főprogram töltsön fel egy tömböt deltoidokkal, és keresse meg, van-e négyzet a deltoidok között.

10. Valósítsuk meg a paralelogrammák típusát, amelyeket az oldalaik hosszával, és az egyik hegyes- (vagy derék) szögével reprezentálunk. Egy paralelogrammának le lehet kérdezni a szögeit, a területét és a kerületét. A főprogram töltsön fel egy tömböt paralelogrammákkal, és adja meg, hány négyzet található a paralelogrammák között.

11. Valósítsuk meg a 3 dimenziós vektorok típusát, ahol a vektort három koordinátájával ábrázoljuk. Egy vektornak le tudjuk kérdezni a hosszát, negálhatjuk, valamint szorozhatjuk egy skalár értékkel (ekkor az a hosszát módosítja). Két vektort összeadhatunk, vehetjük a különbségüket, illetve lekérdezhetjük a távolságukat. A főprogram töltsön fel egy tömböt vektorokkal, és adja meg az összesített vektort.

12. Valósítsuk meg a kockák típusát, ahol a kockát középpontjával, valamint oldalhosszával ábrázolunk. Ehhez valósítsuk meg a térbeli pont típusát is. Egy kockának lekérdezhetjük a felszínét, térfogatát, valamint megállapíthatjuk, hogy egy kocka tartalmaz-e egy másikat, illetve érintkezik vele (részben tartalmazza). A főprogram rögzítsen egy kockát, továbbá töltsön fel egy tömböt kockákkal, és adja meg, hány kocka tartalmazza a kijelöltet, mennyi érintkezik vele, illetve mennyit tartalmaz.

13. Valósítsuk meg a Tamagochik típusát. A Tamagochi névvel és 0..100 közötti testtömeggel rendelkezik. A tömeget etetéssel és futtatással befolyásolhatjuk, amely során a Tamagochi négy állapotban lehet:

- *kövér*, ha tömege 70 fölé megy;
- *sovány*, ha 30 alá megy;
- *normál*, ha tömege 30 és 70 közötti;
- *halott*, amennyiben tömege eléri a 100-t, vagy a 0-t.

Etetés során megadott mennyiségű ételt kaphat, amely a következőképpen változtatja tömegét:

- a mennyiség felével növeli, ha kövér;
- a mennyiséggel növeli, ha normál;
- a mennyiség duplájával növeli, ha sovány.

Futtatásnál a távot adjuk meg, amely a következőképpen hat a tömegre:

- a táv felével csökkenti a tömegét, ha kövér;
- a táv harmadával csökkenti a tömegét egyébként.

Legyen lehetőség Tamagochi beolvasására, illetve kiíratására a **>>** és **<<** operátorok segítségével. A beolvasásnál a Tamagochi nevét, illetve kezdőtömegét adjuk meg, míg a kiírásnál a nevét, illetve állapotát (kövér, normál, sovány, halott) látjuk. A főprogramban lehessen létrehozni új Tamagochit, beolvasni, kiírni, valamint etetni, illetve futtatni (természetesen csak akkor, ha még életben van).

14. Valósítsuk meg az egyetemi hallgatók típusát. A hallgató névvel, Neptun-kóddal és 0..10 közötti energiával, illetve 0…10 közötti kedvvel rendelkezik. Az energia és a kedv több módon befolyásolhatóak:

- *tanulás* csökkenti az energiát 1-gyel, csökkenti a kedvet 1-gyel,
- *vizsgázás* csökkenti az energiát 3-mal, csökkenti a kedvet 2-vel,
- *bulizás* csökkenti az energiát 1-gyel, növeli a kedvet 5-tel.
- *alvás* növeli az energiát 5-tel, a kedvet nem befolyásolja.

Amennyiben a hallgató eléri a 0 kedvet, akkor csak bulizni hajlandó, ha eléri a 0 energiát, akkor csak aludni, viszont ha eléri a 10 energiát, akkor a kedve is növekszik 1-gyel. (Az értékek nem léphetnek ki a 0..10 intervallumból).

Legyen lehetőség hallgató beolvasására, illetve kiíratására a **>>** és **<<** operátorok segítségével. A beolvasásnál a hallgató nevét és Neptun kódját adjuk meg (a kezdő energia 10, kedv 5), míg a kiírásnál a nevét, Neptun-kódját, energiáját és kedvét látjuk. A főprogramban lehessen létrehozni új hallgatót, beolvasni, kiírni, valamint a fenti műveleteket elvégezni (ha a hallgató nem hajlandó elvégezni, arról is értesüljünk).