Eötvös Loránd Tudományegyetem
Informatikai Kar

Alkalmazott modul: Programozás

Feladatgyűjtemény

Összeállította:
Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Frissítve:
2015. szeptember 3.

I. Procedurális programozás

1. Kifejezések

1. Döntsd el egy egész számról, hogy páros-e.
2. Döntsd el egy tetszőleges számról, hogy egy adott intervallumba esik-e.
3. a) Döntsd el egy koordinátákkal adott pontról, hogy az origó-e.
   b) Döntsd el, hogy az egyik koordinátatengelyre esik-e.
4. Számítsd ki egy adott sugarú gömb térfogatát.
5. a) Döntsd el két egész számról, hogy az első osztója-e a másodiknak.
   b) Döntsd el, hogy bármelyik osztója-e a másiknak.
6. Döntsd el három számról, hogy lehetnek-e egy háromszög oldalhosszai.
7. Döntsd el két számról, hogy megegyezik-e az előjelük.
8. a) Add meg egy számtani sorozat első két elemének ismeretében a harmadik elemét.
   b) Add meg az N-edik elemét.
   c) Mértani sorozatra is add meg az N-edik elemet.
9. Számítsd ki egy háromszög területét az oldalhosszaiból.
10. Számítsd ki két térvektor vektoriális szorzatát (koordináták használatával).
11. a) Add meg egy koordinátákkal adott pont távolságát az origótól.
    b) Két tetszőleges, koordinátáival adott pont távolságát add meg.
12. Add meg egy másodfokú egyenlet megoldásait.
13. Számítsd ki egy síkbeli koordinátákkal megadott háromszög szögeit.
14. Döntsd el egy szövegről, hogy nagybetűvel kezdődik-e.
15. Döntsd el egy szövegről, hogy számjegyre végződik-e.
16. a) Add meg egy tetszőleges szöveg első szavát.
    b) Egy tetszőleges szövegnek töröld le az első szavát.
    c) Egy tetszőleges szöveg első szavát cseréld le egy másik, adott szóra.

2. Vezérlési szerkezetek

17. a) Írj ki N darab *-ot.
    b) "Rajzolj" ki egy NxN-es négyzetet *-okból.
    c) Rajzolj ki egy N hosszú befogójú, egyenlő szárú derékszögű háromszöget *-okból.
    d) Rajzolj ki egy N oldalhosszúságú, csúcsára állított rombuszt *-okból.
    e) Rajzolj ki egy NxN-es sakktáblát, a sötét mezőket **, a világosakat szóközök jelöljék.
18. Sorold fel két pozitív egész szám közös osztóit.
19. Sorold fel az első N négyzetszámot.
20. Sorold fel a K-nál kisebb négyzetszámokat.
21. Állíts elő N darab véletlen számot.
22. Add meg az N. Fibonacci-számot. A Fibonacci sorozat egész számokból áll, az első két tagja 0 és 1, és minden további tagja az előző két tag összege.
23. Egész számhármasok tetszőleges sorozatát módosítsd úgy, hogy minden hármas növekvő sorrendben legyen.
24. Sorold fel azokat a másodfokú egyenleteket, amelyek minden együtthatója 0 és 10 közötti egész szám, és pontosan egy megoldása van.
25. Add meg a Pascal-háromszög első N sorát.
26. Add meg egy tetszőleges számsorban az ismétlődő számokat.
27. a) Egy olyan szövegből, amiben van pontosan egy zárójelpár, add meg a zárójelben levő részt.
    b) Ismerd fel, ha nincs zárójel a szövegben.
    c) Ismerd fel, ha hibás a zárójelezés.
    d) Több zárójelpár esetén add meg mindegyik tartalmát.
    e) Ha a zárójelen belül újabb zárójelpár van, akkor is a teljes külső zárójelpár taralmát add meg.
28. Egy tetszőleges szövegben alakítsd át a kisbetűket nagybetűkké, a nagybetűket pedig kisbetűkké.
29. a) Fordíts meg egy tetszőleges egész számsort.
    b) Fordíts meg egy tetszőleges szöveget.
30. Sorold fel egy tetszőleges egész számsor összes részsorozatát.
31. Add meg két tetszőleges szövegről, hogy mely pozíciókon vannak azonos karaktereik.

32. Add meg egy tetszőleges szöveg karaktereinek az összes permutációját.

33. Egy tetszőleges szöveget módosíts úgy, hogy a sorai elé írod az adott sor sorszámát.

34. Add meg egy tetszőleges szövegnek minden szavát külön-külön.

35. a) Add meg egy természetes szám prímtényezős felbontását.
    b) Az első N természetes szám felbontását add meg.

36. Adott egy szöveg, ami minden sorában szóközzel elválasztott egész számokat tartalmaz. Add meg minden sorhoz a benne található legnagyobb páros számot. (Vigyázz! Nem biztos, hogy minden sorban van páros szám!)

37. Számítsd ki a következő iterációs eljárás n. lépésének eredményét (az x pozitív valós szám négyzetgyökének az értékét közelíti): $a_0 = 1, a_{i+1} = 0.5 * (a_i + x/a_i)$

38. Adott egy tetszőleges számsorozat. Állítsd növekvő sorrendbe az elemeit.

39. Fésülj össze két monoton számsorozatot.

40. Permutált egy számsorozat elemeit véletlenszerűen.

3. Alrogramok

41. Valósítsd meg az **`int kozos(int a, int b)`** függvényt, ami a közös osztók számát adja vissza.

42. Valósítsd meg az **`bool tokeletes(int a)`** függvényt, ami visszaadja, hogy a paraméterül kapott érték tökéletes szám-e.

43. Valósítsd meg az **`bool baratsagos(int a, int b)`** függvényt, ami visszaadja, hogy a paraméterül kapott értékek barátságos számpárt alkotnak-e.

44. Valósítsd meg az int **`max(vector<int> v)`** függvényt, ami a paraméterül kapott vektor elemei közül a legnagyobbat adja vissza.

45. Valósítsd meg az **`bool vane(vector<int> v, int ez)`** függvényt, ami eldönti, hogy a paraméterül kapott vektor elemei között van-e "ez".

46. Valósítsd meg a **`double atlag(vector<double> v)`** függvényt, ami a paraméterül kapott vektor átlagát adja vissza.

47. Valósítsd meg az **`int hany(vector<double> v, double ez)`** függvényt, ami a paraméterül kapott vektorban megszámolja, hogy hány "ez" van benne.

48. Valósítsd meg az **`int hanysor(ifstream &f)`** függvényt, ami egy paraméterül kapott fájlban levő maradék sorok számát adja vissza.

49. Valósítsd meg az **`int hanysor(string fajlnev)`** függvényt, ami a paraméterként megkapott fájlnévhez tartozó fájlt megpróbálja megnyitni, ha nem létezik a fájl, akkor -1-et ad vissza, egyébként pedig a fájlban található sorok számát.

a) Valósítsd meg a **`void alahuzvakiir(string s)`** függvényt, ami a paraméterül kapott szöveget új sorba kiírja, és "=" karakterekkel aláhúzza.

50. Valósítsd meg a **`void szamparbeolvas(ifstream &f, int &a, int &b)`** függvényt, ami a kapott fájlból számpárt olvas be.

51. Valósíts meg függvényt, ami a kapott három egész szám, mint háromszög három magasságának hosszait adja meg.

52. Valósíts meg függvényt, amely egy szöveget átalakít úgy, hogy ha több whitespace karakter (szóköz, tabulátor, újsor) van benne egymás után, azt egyetlen szóközzé alakítja.

53. Valósíts meg függvényt, ami egész számból szöveges változót csinál, ami előjelet és a megfelelő számjegyeket tartalmazza, felesleges karakterek nélkül.

54. Valósíts meg függvényt, ami szöveges változóból egész számot próbál csinálni, visszaadja az eredményt, és visszaadja azt is, hogy zökkenőmentes volt-e az átalakítás. Ez utóbbi érték legyen 0, ha sikeres volt, különben hogy hányadik karakter (1-től indexelve) nem volt számjegy, illetve előjel megfelelő helyen.

4. Programozási tételek egyszerű alkalmazása

55. Számítsd ki egy szám faktoriálisát.

56. Számítsd ki egy tetszőleges számsorozat átlagát.

57. Számítsd ki egy tetszőleges számsorozat szórását.

58. Add meg egy természetes szám valódi osztóinak összegét.

59. Add meg egy tetszőleges egész szám valódi osztóinak a számát.

60. Add meg egy természetes szám legnagyobb valódi osztóját.

61. Add meg két természetes szám legnagyobb közös osztóját.

62. a) Sorold fel az első N tökéletes számot (olyan természetes számot, ami megegyezik a valódi osztóinak összegével).
    b) Sorold fel a K-nál kisebb tökéletes számokat.

63. Add meg egy tetszőleges egész számsorról, hogy hány eleme nagyobb, ill. kisebb az átlagánál.

64. Egy pozitív egész számokból álló számsorban add meg, hogy hány páros szám van.

65. Egy tetszőleges számsorban add meg a legkisebb és a legnagyobb számot.

66. Add meg egy tetszőleges pozitív számsorozat elemeinek a négyzetgyök-összegét.

67. Számítsd ki két N dimenziós vektor skaláris szorzatát.

68. a) Egy szigorúan növő egész számsorban add meg a legnagyobb ugrást (szomszédos elemek közötti legnagyobb előforduló különbséget).
    b) Nem monoton számsorra is adj helyes eredményt.

69. Add meg egy tetszőleges egész számsorban a szomszédos elemek átlagos különbségét.

70. Tetszőleges sok számról döntsd el, hogy növekvő sorrendben vannak-e.

71. Egy egész számról döntsd el, hogy prímszám-e.

72. a) Sorold fel az első N prímszámot.
    b) Sorold fel a K-nál kisebb prímszámokat.
    c) Sorold fel az A és B közé eső prímszámokat.

73. Egy tetszőleges szövegről add meg, hány kis "a" betű van benne.

74. Egy tetszőleges szövegről add meg, hány számjegy, hány nagybetű és hány kisbetű van benne.

75. Add meg egy tetszőleges szövegben, hogy melyik karakter fordul elő benne a legtöbbször.

76. Add meg egy tetszőleges szövegből a leghosszabb sort.

77. Döntsd el egy tetszőleges szövegről, hogy a sorai ABC sorrendben vannak-e.

78. a) Egy tetszőleges szövegben add meg a sorok számát.
    b) Add meg a karakterek számát is.
    c) Add meg a szavak számát is.

79. Add meg egy tetszőleges szöveg leghosszabb szavát.

80. a) Egy tetszőleges szövegben számold meg, hány sor kezdődik azzal a betűvel, amivel az előző végződött.
    b) Azt is számold meg, hogy hány szó kezdődik a megelőző szó utolsó betűjével.

81. a) Egy tetszőleges szövegről add meg, hány mondat található benne. Mondatnak tekintünk minden olyan sort, ami nagybetűvel kezdődik, és ponttal, felkiáltójellel vagy kérdőjellel végződik.
    b) A több mondatot tartalmazó sorokat és a többsoros mondatokat is kezeld helyesen.
82. a) Egy tetszőleges szövegről add meg, hány szóból áll. Felteheted, hogy két szót mindig pontosan egy szóköz választ el.
    b) Akkor is működjön, ha két szó között több szóköz is lehet.

83. Egész számhármasok tetszőleges sorozatáról add meg, hogy a hármasok közül hánynak vannak növekvő sorrendben az elemei.

84. a) Háromszögek oldalhosszainak egy tetszőleges sorozatában (pozitív számhármasok sorozata) add meg a legnagyobb kerületű háromszöget.
    b) Add meg a legnagyobb területű háromszöget is.
    c) A hibás háromszögeket szűrd ki a sorozatból.

85. Tetszőleges, a csúcsai koordinátáival adott sokszögnek add meg a kerületét.

86. Tetszőleges, koordinátákkal adott pontsorozatból add meg, hogy mennyi esik az origó körüli R sugarú körön belülre.

87. Tetszőleges, koordinátákkal adott pontsorozatban add meg az origótól legtávolabb eső pontot.

5. Programozási tételek összetett alkalmazása

88. Add meg egy tetszőleges egész számsorban a prímszámok számát.

89. Add meg, hogy az A és B közötti egész számok közül melyiknek van a legtöbb valódi osztója.

90. Adott egy szöveg, ami minden sorában egész számokat tartalmaz. Add meg, hogy melyik sorban a legnagyobb a sor legkisebb száma (és azt is, hogy mi ez a szám).

91. a) Egy több soros szövegben add meg, hány sorában található meg az "alma" szó.
    b) Az "alma" helyett tetszőleges szöveget lehessen megadni.
    c) A szó összes előfordulásának a számát add meg.

92. Mátrixban tároljuk egy osztály adatait, minden sora egy diák, minden oszlopa egy tantárgy, a mátrix értékei a jegyek. Add meg a következőket:
    a) Osztályátlag.
    b) Legjobb tanuló (átlag alapján).
    c) Legnehezebb tantárgy (legtöbb bukás).
    d) Van-e két hallgató, akiknek egyforma az átlaguk?
    e) Hányan nem buktak meg semmiből?

f) A legjobb tanuló legrosszabb jegye.
g) Ki a legrosszabb átlagú hallgató azok közül, akik nem buktak meg semmiből?
h) Melyik hallgató jegyeinek a legnagyobb a szórása?

93. Egy vonalban szabályos távolságonként megmértük a tengerszint feletti magasságot egyik szárazföldi ponttól a másikig. Add meg a következőket:
    a) Található-e hegycsúcs a területen (olyan pont, amely az előtte és utána lévőnél is magasabb).
    b) Hány sziget található a területen (sziget azon tengerszint feletti magasságok sorozata, amely előtt és után is tengerszint alatti magasság van).
    c) Milyen hosszú a legnagyobb (legtöbb mérésből álló) sziget.
    d) Van-e olyan sziget, amelyben völgy található.
    e) Hány hegycsúcs található a legnagyobb szigeten.
    f) Hány hegycsúcs alakú sziget van (ahol a hegycsúcs előtt monoton nőnek, utána monoton csökkenek az értékek).

94. Egy szöveges fájl minden sora egy mondatot tartalmaz, amelyek tetszőleges számú szóból állhatnak. Minden mondatnak van egy írásjel a végén, de ezen kívül nincs más írásjel a mondatban.
    a) Hányadik mondatban található a legtöbb szó?
    b) Melyik a leghosszabb szó a teljes szövegben?
    c) Van-e olyan mondat a szövegben, amely legalább 10 szavat tartalmaz?
    d) Hányadik mondatban található a legtöbb ’s’ betű?
    e) Van-e olyan mondat, amely évszámot (4 jegyű pozitív számot) tartalmaz?
    f) Mennyi a szavak hosszának átlaga a legtöbb szót tartalmaz mondatban?
    g) Hányadik mondatban található a legtöbb határozott névelő (a, az)?

95. Egy szöveges fájl minden sora egy egész számsorozatot tartalmaz (mindegyik sor tetszőleges hosszú, akár üres is lehet). Add meg a következőket:
    a) Melyik sor összege a legnagyobb.
    b) Melyik sorban lévő számok abszolút értékeinek összege a legnagyobb.
    c) Melyik sorban található páros sok szám.
    d) Melyik sorban található a legnagyobb páros szám.
    e) Van-e olyan sor, amelyben csak egyféle érték fordul elő.
    f) Hány olyan sor van, amelyben a számok átlaga pont nulla.
    g) Hányadik sorban van a legnagyobb szám.

II. Strukturált programozás

1. Rekordok

1. Adott egy szövegfájl, ami egy recept hozzávalóit tartalmazza. A fájl minden sora egy számmal kezdődik, ami egy összetevőből szükséges mennyiség, majd vesszővel elválasztva tőle az összetevő neve jön.
   a) Add meg azt az összetevőt, amiből a legtöbb, és amiből a legkevesebb kell.
   b) Add meg, hány olyan összetevő van, amiből kevesebb, mint egy egységnyi kell.
   c) Add meg egy tetszőleges összetevőről, hogy mennyi kell belőle.

2. Adott egy telefonkönyv egy szövegfájlban, aminek a sorai vesszővel elválasztott neveket és telefonszámokat tartalmaznak.
   a) Egy tetszőleges névhez add meg a telefonszámot.
   b) Egy tetszőleges telefonszámhoz add meg a nevet.

3. Adott egy szövegfájl, aminek a sorai neveket és születési adatokat tartalmaznak (név, év.hó.nap. alakban).
   a) Add meg egy ember születési adatait a neve alapján.
   b) Add meg a legöregebb és legfiatalabb embert a listában.
   c) Add meg, hány januári születésnap van a listában.

4. Adott egy szövegfájl, ami egy hónap minden napjának hőmérsékleti adatait tartalmazza: minden sorban három szám van, egy napon mért reggeli, déli és esti hőmérsékletet.
   a) Add meg a havi átlaghőmérsékletet.
   b) Add meg a legalacsonyabb napi középhőmérsékletet (és azt is, hogy hányadik napon volt).
   c) Add meg, hány reggel volt fagy.
   d) Add meg, melyik napon volt a legnagyobb hőmérséklet-ingadozás.

2. Típusok

5. Valósítsuk meg a komplex számok típusát úgy, hogy a komplex számot az algebrai alakkal ábrázoljuk ($x + iy$). Implementáljuk az összeadás, kivonás, szorzás, osztás, konjugálás, beolvasás és kiírás műveleteit (operátorok segítségével).

6. Valósítsuk meg a polinomok típusát, ahol a polinomot együtthatói sorozatával ábrázoljuk (amelyeket egy tömbben tárolunk). Implementáljuk az összeadás, kivonás, szorzás, beolvasás és kiírás műveleteit (operátorok segítségével), adott ponton történő helyettesítési érték kiszámítását, valamint adott fokszámú együttható lekérdezését, vagy módosítását.

7. Valósítsuk meg a valós értékű diagonális mátrixok típusát. A hatékony ábrázolás érdekében csak a nem nulla elemeket tároljuk el. Legyen lehetőség mátrixok összeadására, szorzására, beolvasására és kiírására (operátorok segítségével), a determináns lekérdezésére, továbbá tetszőleges soron és oszlopon lévő érték beállítására, valamint lekérdezésére.

8. Valósítsuk meg a valós értékű tridiagonális mátrixok típusát, ahol az elemek csak a főátlóban, illetve alatta és felette helyezkedhetnek el. A hatékony megvalósítás érdekében csak a nem nulla adatokat tároljuk el. Legyen lehetőség mátrixok összeadására, szorzására, beolvasására és kiírására (operátorok segítségével), a determináns lekérdezésére, továbbá tetszőleges soron és oszlopon lévő érték beállítására, valamint lekérdezésére.

9. Valósítsuk meg a 2×2-es egész értékű mátrixok típusát. Legyen lehetőség mátrixok összeadására, szorzására, beolvasására és kiírására (operátorok segítségével), a determináns lekérdezésére, inverz számítására, továbbá tetszőleges soron és oszlopon lévő érték beállítására, valamint lekérdezésére.

10. Valósítsuk meg a nagyon nagy természetes számok típusát, ahol a számokat számjegyeik sorozatával ábrázoljuk (amelyeket tömbben tárolunk). Implementáljuk nagy számok összeadását, szorzását, beolvasását és kiírását operátorok segítségével. A főprogram töltsön fel egy tömböt nagyon nagy számokkal, amelynek ezután ki kell számolni az összegét és szorzatát.

11. Valósítsuk meg a karakterlánc típust, ahol a karakterlánc egy karakterekből álló tömb. A karakterlánc bővíthető, azaz hozzáfűzhető tetszőleges sok karakter. Emellett lehetőség van két karrakterlánc konkatenálására, kiírására, illetve beolvasására operátorok segítségével. A főprogram olvasson be két karakterláncot, majd konkatenálja őket össze, és utána biztosítson lehetőséget annak bővítésére tetszőleges sokszor tetszőleges karakterrel.

3. Sablonok

12. Valósítsuk meg a sablonos verem adattípust, amelyet tömb segítségével reprezentálunk. Legyen lehetőség elem behelyezésére (push), kivételére (pop), valamint a tetőelem lekérdezésére (top). A veremnek létrehozáskor adjuk meg a maximális méretét. A verem jelezzen hibát, ha valamely művelet sikertelen volt.

13. Valósítsuk meg a sablonos verem adattípust, amelyet láncolt adatszerkezet segítségével reprezentálunk. Legyen lehetőség elem behelyezésére (push), kivételére (pop), valamint a tetőelem lekérdezésére (top). A verem jelezzen hibát, ha valamely művelet sikertelen volt.

14. Valósítsuk meg a sablonos sor adattípust, amelyet tömb segítségével reprezentálunk. Legyen lehetőség elem behelyezésére (in), kivételére (out),

valamint az első elem lekérdezésére (first). A sornak létrehozáskor adjuk meg a maximális méretét. A sor jelezzen hibát, ha valamely művelet sikertelen volt.

15. Valósítsuk meg a sablonos halmaz típust, amelyet tömb segítségével reprezentálunk. Implementáljuk elem behelyezését, kivételét, a tartalmazás lekérdezését, halmaz kiírását, valamint halmazok unióját, metszetét, különbségét és szimmetrikus differenciáját.

16. Valósítsuk meg a sablonos halmaz típust, amelyet láncolt adatszerkezet segítségével reprezentálunk. Implementáljuk elem behelyezését, kivételét, a tartalmazás lekérdezését, halmaz kiírását, valamint halmazok unióját, metszetét, különbségét és szimmetrikus differenciáját.

17. Valósítsuk meg a sablonos zsák adatszerkezetet, amelyet tömb segítségével reprezentálunk. Implementáljuk elem behelyezését, kivételét, a számosság lekérdezését, zsák kiírását, valamint zsákok unióját és metszetét.

III. Objektumorientált programozás

1. Készítsünk el egy geometriai alakzatokat megvalósító alkalmazást, amelybe feltölthetünk különböző típusú alakzatokat (vízszintes vonal, függőleges vonal, kör, négyzet), és azt közös adatszerkezetben tudjuk kezelni. Az alakzatokra lehessen együttesen területet és kerültet lekérdezni, valamint meghatározni, mely alakzatok tartalmaznak pontot, illetve eltolni őket egy vektorral. Minden alakzatot reprezentáljuk egy középpont és egy sugár segítségével, és ennek megfelelően a műveletek a típusnak megfelelő értékeket állítsák elő.

2. Készítsünk el egy alkalmazást, amelyben vonatokat állíthatunk össze. A vonatok kocsikból állnak, melyek lehetnek mozdonyok, személykocsik, teherkocsik, és egy speciális a bicikliszállító. Minden kocsiról ismert az azonosítószáma valamint a hossza.
   - Mozdony esetén ismert a meghajtás típusa (gőz, elektromos, …), valamint a szállítható vagonok száma.
   - Személykocsik esetén ismert az ülések száma és az osztály, teherkocsik esetén ismert a kapacitás és az áru típusa.
   - A bicikliszállító személy- és teherkocsi egyben, mindig másodosztályú és mindig kerékpárt szállít.

   A vonatoknak lehessen lekérdezni a teljes hosszát, az összes ülésszámot és árukapacitást, lehessen vagonokat csatolni és leválasztani, valamint elindítani a vonatot egy megadott célállomásra (persze csak akkor, ha a mozdonyok elbírják a szerelvényt). Mozdony csatolásánál ügyeljünk arra, hogy mindig a vonat elejére kapcsoljuk, és csak olyan mozdony engedélyezett, amelynek meghajtása megegyezik a már a vonathoz csatolt mozdonyokéval (ha van olyan).

3. Készítsünk programot, amely alkalmas egy szerelőműhelyben rendelkezésre álló és felhasznált alkatrészek adatainak nyilvántartására.

   Az alkatrészek a következők lehetnek:
   - *Csavar*: adott a mérete (átmérője), a típusa (fa/fém), kiszerelés, illetve a nettó egységára.
   - *Csapágy*: adott a mérete (átmérője), kiszerelés, illetve a nettó egységára.
   - *Gerenda*: adott a hossza, a szélessége, illetve a négyzetméterenkénti nettó egységára.

   A program olvassa be a fájlt és biztosítson lehetőséget a méret és a bruttó ár kiszámítására. A kiszerelés megadja, hogy az adott csomagban hány alkatrész található. Gerenda esetén a méret a szélesség és a hossz szorzata. A bruttó egységár 27% ÁFA-t tartalmaz.

4. Készítsünk programot, amely segítségével matematikai műveleteket tudunk elvégezni és azok eredményét kiírni a képernyőre. A program négyféle műveletet támogat, amelyek kódjai és operandusai egy szöveges fájlban találhatóak:

   - Faktoriális számítás (0-ás kód), 1 operandusú
   - Legkisebb közös többszörös számítása (1-es kód), 2 operandusú
   - Fibonacci sorozat n-edik tagja (2-es kód), 1 operandusú
   - Vektorok skaláris szorzata (3-as kód), 2 operandusú

   Vektorok esetén a vektor hossza nem adott előre, így azt a sorban található számok számából kell kikövetkeztetni.

   A program olvassa be a fájlt, majd írja ki a képernyőre a műveletek nevét, operandusait, valamint eredményét soronként. Amennyiben egy sorban nincs elég operandus, vagy hibás a sor szerkezete, úgy a hibát jelezze, de a többi műveletet a fájlból végezze el. A megvalósításnál a műveleteket készítsük el osztályok formájában, amelyek az alábbi interfészt valósítják meg:

```
class Muvelet {
public:
   std::string nev() = 0;
   std::string[] operandusok() = 0;
   double kiszamol() = 0;
}
```

5. Egy szöveges állomány egy lakás helységeinek adatait (lakószobáknál az oldalhosszakat, az ajtók számát és az ablakok számát, terasz esetén az alapterületet, egyéb helységek esetén az alapterületet, az ajtók számát és az ablakok számát) tartalmazza. Minden sorban egy-egy helység adatai találhatók. A sor első számjegye a helység fajtájára utal (1, ha szoba; 2, ha terasz, 3 ha egyéb helység), ezután szóközökkel elválasztva a helység fajtájától függően egy vagy két valós szám, majd (a terasz kivételével) az ajtók és ablakok számát megadó két egész. Definiáljuk külön-külön az egyes helység-típusok osztályait, amelyek az alábbi interfész leszármazottai.

```
class Helyiseg {
public:
   virtual std::string nev() = 0;
   virtual double terulet() = 0;
   virtual int ablakokSzama() = 0;
   virtual int ajtokSzama() = 0;
};
```

   Készítsünk olyan programot, amely a szöveges állomány alapján létrehozza a megfelelő helyiség-objektumokat. A felhasználónak legyen lehetősége megadni a fájlnevet, valamint lekérdezni a következő adatokat:

   - teljes terület, teljes belterület (terasz nélkül),

- adott helyiség összes adatainak listája,
- összes helyiség listája terület szerint növekvő sorrendben rendezve (ehhez alkalmas rendező algoritmust kell megvalósítani).

6. Készítsünk programot, amely különböző típusú testek tömegét tudjuk kiszámítani. A program háromféle test adatait tudja kezelni:

   - Hengeres márványoszlop (0): sűrűség, magasság, átmérő;
   - Tölgyfa egészben, gyökér nélkül (1): sűrűség, törzs hossza, törzs sugara, fa kora, zsugorodási tényező;
   - Raktári polc (2): sűrűség, teljes magasság, hosszúság, mélység, polcok száma, polcok vastagsága.

   A program az adatokat szöveged fájlból olvassa be, ahol az első sor a testek száma, majd ezt követik soronként egy test kódja (0,1, vagy 2) és az adatai a megadott sorrendben. Amennyiben egy sor szerkezete nem megfelelő, akkor ezt a program jelezze, de a további sorokat dolgozza fel. A testeket osztályok segítségével valósítsuk meg a következő absztrakt osztályból származtatva:

```
abstract class Test {
public:
   virtual std::string nev() = 0;
   virtual double terfogat() = 0;
   virtual double suruseg() = 0;
   double tomeg() {
      return terfogat() * suruseg();
   }
}
```

   A tömeg kiszámításához használjuk a következő képleteket:

   - tömeg: $m = V \cdot \rho$ (V: térfogat, ρ: sűrűség)
   - henger térfogata: $V = \pi \cdot r^2 \cdot h$ (r: henger sugara, h: henger magassága)
   - fa térfogata: $V = V_1$, ahol $V_n = \begin{cases} \pi \cdot r^2 \cdot h \cdot d^n + \frac{1}{d+n} \cdot V_{n+1}, ha\ n < k \\ \pi \cdot r^2 \cdot h \cdot d^n, ha\ n = k \end{cases}$

     (r: törzs sugara, h: törzs magassága, k: fa kora, d: zsugorodási tényező)
   - polc térfogata: $V = l \cdot d \cdot t \cdot n$ (l: hossz, d: mélység, t: vastagság, n: polcok száma)

7. Egy szöveges állomány háromféle síkidom adatait (négyzetnél az alapot, téglalapnál az alapot és az oldalt, rombusznál az alapot és az alapok által bezárt szöget) tartalmazza. Minden sorban egy-egy síkidom adatai találhatók. A sor első számjegye a síkidom fajtájára utal (0, ha négyzet; 1, ha téglalap; 2, ha rombusz),

ezután szóközökkel elválasztva a síkidom fajtájától függően egy vagy két valós szám. Definiáljuk külön-külön az egyes síkidomok-típusok osztályait úgy, hogy a négyzetét az alább megadott absztrakt osztályból származtatjuk, a másik kettőt pedig a négyzet osztályából.

```
class Alakzat {
public:
   virtual std::string nev() = 0;
   double terulet() {
      return alap() * magassag();
   }
   double kerulet() {
      return 2 * (alap() + oldal());
   }
   virtual double alap() = 0;
   virtual double oldal() { return alap(); }
   virtual double magassag() { return oldal(); }
};
```

Készítsünk olyan programot, amely a szöveges állomány alapján létrehozza a megfelelő síkidom-objektumokat, és eltárolja őket. Lehessen kilistázni a megadott tartalmat, valamint rendezni azt terület, illetve kerület szerint növekvő sorrendbe, ehhez valósítsunk meg alkalmas rendező eljárást.

8. Készítsünk programot, amely elősegíti termek foglalását az oktatók számára. A rendszer háromféle termet tart nyilván, amelyek a következő adatokkal rendelkeznek:

   - *Előadó terem* (0-s kód): terem neve, az ülőhelyek száma, van-e beépített projektor (0: nincs, 1: van)
   - *Szeminárium terem* (1-es kód): terem neve, az ülőhelyek száma, valamint a tábla típusa (0: krétás, 1: filces).
   - *Gépterem* (2-es kód): terem neve, az ülőhelyek száma, számítógépek száma.

   Ezen felül minden teremre kiszámolható a kapacitása a következőeknek megfelelően:

   - *Előadó terem*: ha van projektor, és azt igénybe veszik, akkor az ülőhelyek számának 115%-a, mivel ekkor úgyse fognak bejönni a hallgatók órára, különben a ülőhelyek száma.
   - *Szeminárium terem*: ha filces a tábla, az ülőhelyek száma, ha krétás, akkor az ülőhelyek száma -6 fő, mert az első sorba senki sem ül a szálló krétapor miatt.

- *Gépterem*: a számítógépek számának 90%-a (mert a többi biztos rossz) +10 fő (akik úgyis laptoppal járnak), de maximálisan az ülőhelyek száma.

A program az adatokat szöveges fájlból olvassa be, amelynek minden sora egy terem adatait tartalmazza a megadott sorrendben. Legyen lehetőség a felhasználónak megadni, mekkora kapacitású termet akar foglalni, géptermet szeretne-e, legyen-e projektor (gépteremben mindig van, szeminárium teremben nincs), illetve filces táblát szeretne-e (gépteremben mindig filces a tábla, előadóban sosem), és a program ekkor listázza ki azokat a termeket, amelyek az igénynek eleget tesznek, vagy írja ki, hogy nincs a keresésnek megfelelő terem.