Pótórák feladatai és megoldásai

1. Pótóra (2024. Márcuis 05.)

A feladat (hátultesztelő)

1. Kérj a felhasználótól egy 3 jegyű egész számot.
2. Ha nem ilyet ír be, kérd újra.
3. Ha megfelelő számot írt be, alkoss az utolsó számjegyéből egy kétjegyű számot, ami a 11 többszöröse, és az annál 7-tel nagyobbat.
   `PL 635 -> 55 és 62`
   Megoldás

```cpp
#include <iostream>

using namespace std;

int main() {
    int sz;

    /*2. Ha nem ilyet ír be, kérd újra.*/
    // azt kell átgondolni, hogy egyszer megengedhetjük-e,
    // hogy lefusson a do ciklus
    do {
        /*1. Kérj a felhasználótól egy 3 jegyű egész számot.*/
        cout << "Kérek egy háromjegyű számot! " << endl;
        cin >> sz;
    } while (sz < 100 || sz > 999);

    // ugyenezt elérhetnénk a következő módon is:
    /*
    cout << "Kérek egy háromjegyű számot! " << endl;
    cin >> number;
    while (number < 100 || number > 999) {
        cout << "Kérek egy háromjegyű számot! " << endl;
        cin >> number;
    }
    */
   // ugyanazt értük el, viszont több sorban kellett leírnunk.
   // mivel egyszer mindenképp lefut az adatbekérés (16-17. sor) (bármilyen


    /*3. Ha megfelelő számot írt be, alkoss az utolsó számjegyéből egy kétj
    ami a 11 többszöröse, és az annál  7-tel nagyobbat.  PL 635 -> 55 és 62

    // A feladat hátralevő része már cikluson kívül oldható meg,
    // mivel nem kell minden megadott adatra elvégezni a műveleteket.
    // Ha ez is a ciklusokon belül lenne,
    //akkor feleslegesen végeznénk el ezeket a műveleteket,
    //nekünk nem megfelelő számokra.
    // Például:
    // Ha szeretnél egy almát felvágni négybe, és sorba adogatnak a kezedbe
```

```
        //akkor ezeket nem fogod felvágni, mert nem almák.
        // Inkább kérsz egy másik gyümölcsöt, mindaddig, amíg az nem alma.
        //Ekkor nem kérsz több gyümölcsöt, hanem felvágod az almát.

        int tizenegyszer = number % 10 * 10 + number % 10;
        int pluszHet = tizenegyszer + 7;

        cout << "Az utolsó számjegyből alkotott 11 többszöröse " << tizenegysze
        return 0;
    }
```

B feladat (hátultesztelő)

1. Kérj a felhasználótól egy kisbetűt!
2. Ha nem kisbetűt ír be, akkor kérd újra!
3. Ha kisbetűt írt be, akkor írjuk ki a nagybetűs párját, háromszor egymás alá, úgy hogy legyen közöttük üres sor!
   Megoldás

```cpp
#include <iostream>

using namespace std;

int main() {
    char betu;
    /*2. Ha nem kisbetűt ír be, akkor kérd újra!*/
    do {
        /*1. Kérj a felhasználótól egy kisbetűt!*/
        cout << "Kérek egy kisbetűt! " << endl;
        cin >> betu;
    } while (betu < 97 || betu > 122);
    // ascii táblázatból láthatjuk, hogy a kisbetűk kódjai 97 és 122 között
    // Az első feladathoz hasonlóan a do ciklus használata itt indokolt,
    // mivel egyszer mindenképp lefut az adatbekérés.


    /*3. Ha kisbetűt írt be, akkor írjuk ki a nagybetűs párját,
    háromszor egymás alá, úgy hogy legyen közöttük üres sor!*/

    // Ahhoz, hogy a kis a-ból nagy A legyen, meg kell keresnünk,
    // hogy mekkora a kettő közt a távolság.
    // A kis a kódja 97, a nagy A kódja 65, tehát 32 a különbség.
    // Mivel párhuzmosan növekednek a számok,
    // ezért ez a különbség nem csak az a-ra igaz,
    // hanem minden betűre.
    char nagyBetu = betu - 32;

    //mivel háromszor kell kiírni, kell egy for ciklus, ami háromszor fut l
    for (int i = 0; i < 3; i++) {
        cout << nagyBetu << endl;
        // A feladat kér közéjük egy-egy üres sort is,
        // ezért nem elég egy endl, kettő kell.
        cout << endl;
    }

    return 0;
}
```

C feladat (elöltesztelő)

1. Kérj a felhasználótól egy egyjegyű számot, és őrizd meg az `n` változóban!
2. Amíg a szám kisebb mint `100`, adj hozzá véletlen számokat a `[10,50]` intervallumból.
3. Ha az `n` eléri a 100-at, lépj ki a ciklusból!
4. Menet közben írasd ki a generált számokat (nem az összeget) egymás mellé, szóközzel elválasztva
5. Menet közben számold meg, hány számot generáltál ki
6. Menet közben számítsd ki a generált számok összegét
7. Menet közben számold meg, hány `1`-gyel kezdődő számot generáltál
8. Írj üzenetet az eredményekről
9. Számítsd ki és írd ki a generált számok átlagát!
   Megoldás

```cpp
#include <iostream>

using namespace std;

int main() {
    srand(time(0));
    /*1. Kérj a felhasználótól egy egyjegyű számot, és őrizd meg az `n` vál
    cout << "Kérek egy egyjegyű számot! " << endl;
    int n;
    cin >> n;
    int szamlal = 0;
    int sum = 0;
    int szamlalEgyes = 0;

    // Mivel a feladat nem kéri, hogy ellenőrizzük a megadott számot,
    // ezért nem garantált, hogy az tényleg egyjegyű lesz.
    // Ezért ha a felhasználó nem fogad szót,
    // és rögtön 100-nál nagyobb számot ad meg,
    // akkor egyszer sem adhatunk hozzá véletlen számot.
    // Emiatt, ha do ciklust használnánk, akkor nem teljesítenénk a feladat
    /*
    do {
        int random = rand() % 41 + 10;
        szamlal++;
        sum += random;
        if (random / 10 == 1) {
            szamlalEgyes++;
        }
        cout << random << " ";
        n += random;
    } while (n < 100);
    */
    // Így még ha 100-nál nagyobb számot kapunk, akkor is adtunk hozzá véle
    // (pedig nem kellett volna), mivel a do ciklus legalább egyszer lefut.


    /*2. Amíg a szám kisebb mint `100`, adj hozzá véletlen számokat a `[10,

    // Ezért inkább while ciklust használunk,
```

```
        // mivel a feltétel nem garantáltan teljesül a ciklus első futásakor.
        /*3. Ha az `n` eléri a 100-at, lépj ki a ciklusból!*/
        while (n < 100) {
            int random = rand() % 41 + 10;
            /*Menet közben számold meg, hány számot generáltál ki*/
            counter++;
            /*Menet közben számítsd ki a generált számok összegét*/
            sum += random;
            /*Menet közben számold meg, hány 1-gyel kezdődő számot generáltál*/
            if (random / 10 == 1) {
                szamlalEgyes++;
            }
            /*Menet közben írasd ki a generált számokat (nem az összeget) egymá
            cout << random << " ";
            n += random;
        }

        cout << endl;
        /*Írj üzenetet az eredményekről*/
        cout << "Generált számok száma: " << szamlal << endl;
        cout << "Generált számok összege: " << sum << endl;
        cout << "1-gyel kezdődő számok száma: " << szamlalEgyes << endl;
        /*Számítsd ki és írd ki a generált számok átlagát!*/
        double avg = (double)sum / counter;
        cout << "Generált számok átlaga: " << avg << endl;

        return 0;
    }
```

D feladat

1. Írj függvényt, ami a kocka megadott élhosszából (cm) kiszámítja egy kocka térfogatát, (cm)!
   `int kocka (int a)`
2. A főprogramban kérd be a felhasználótól a kocka élhosszát cm-ben!
3. Hívd meg a térfogat számítására szolgáló függvényt!
4. Írasd ki, hogy hány köbcentiméter a kocka térfogata!
   Megoldás

```cpp
#include <iostream>

using namespace std;

/*Írj függvényt, ami a kocka megadott élhosszából (cm) kiszámítja egy kocka
int kocka (int a) {
    int terf = a * a * a;
    return terf;
}

int main() {
    /*A főprogramban kérd be a felhasználótól a kocka élhosszát cm-ben!*/
    cout << "Kérem a kocka élének hosszát! " << endl;
    int elhossz;
    cin >> elhossz;
    /*Hívd meg a térfogat számítására szolgáló függvényt!*/
    int terfogat = kocka(elhossz);
    /*Írasd ki, hogy hány köbcentiméter a kocka térfogata!*/
    cout << "A kocka térfogata " << terfogat << endl;
    return 0;
}
```

E feladat

1. Írj függvényt ami egy derékszögű háromszög két befogójából kiszámítja az átfogót!
   `double atfogo (double a, double b)`
2. A főprogramban kérd be a felhasználótól a befogók hosszát, tizedes számok is lehetnek!
3. Hívd meg a függvényt, és írd ki az eredményt!
   Megoldás

```cpp
#include <iostream>
#include <cmath>

using namespace std;

/*Írj függvényt ami egy derékszögű háromszög két befogójából kiszámítja az
double atfogo (double a, double b) {
    double atfogo = sqrt(a * a + b * b);
    return atfogo;
}

int main() {
    /*A főprogramban kérd be a felhasználótól a befogók hosszát, tizedes sz
    cout << "Kérem a befogók hosszait! " << endl;
    double befogo1, befogo2;
    cin >> befogo1 >> befogo2;
    /*Hívd meg a függvényt, és írd ki az eredményt!*/
    double atfogoHossz = atfogo(befogo1, befogo2);
    cout << "Az átfogó hossza " << atfogoHossz << endl;
    return 0;
}
```

2. Pótóra (2024. Március 12.)

A feladat

1. Az `a` tömb elemeinek add kezdőértékként: `{2, -10, 8, 3, 11, 20, 7}`
2. Írj függvényt, ami kiírja a tömb minden elemét egy sorba, szóközzel elválasztva! Az utolsó elem után kövekezzen 2 üres sor!
   `void kiirsorba(int *a, int n)`
   `a` - a tömb neve, `n` - az elemeinek a száma
3. A főprogramból hívd meg a függvényt és mutasd meg a tömb elemeit!
4. Deklarálj egy új tömböt, aminek a neve `haromszoros` legyen! (gondolkodj el, milyen típusú adatok kerülnek majd bele).
5. Az új tömbbe az a tömb elemeinek háromszorosa kerüljön!
6. Írasd ki a haromszoros tömb elemeit is a kész függvénnyel!

7. Írj függvényt, ami a összeadja egy tömb páros indexű elemeit!
   `int pindossz(int *a, int n)`
   `a` - a tömb neve, `n` - az elemeinek a száma
8. Hívd meg a függvényt a főprogramból és írasd ki mindkét tömb páros indexű elemeinek az összegét, üzenetek kíséretében!
9. Számítsd ki az eredeti tömb páros indexű elemeinek az átlagát is, és írasd ki!
   Megoldás

```cpp
#include <iostream>

using namespace std;
/*Írj függvényt, ami kiírja a tömb minden elemét egy sorba, szóközzel elválaszt
Az utolsó elem után kövekezzen 2 üres sor!*/
void kiirsorba(int *a, int n) {
    for (int i = 0; i < n; i++) {
        cout << a[i] << " ";
    }
    cout << endl;  //ez az endl meg a kiiras veget jelzi
    cout << endl;
    coud << endl;
}

/*Írj függvényt, ami a összeadja egy tömb páros indexű elemeit!*/
int pindossz(int *a, int n) {
    int sum = 0;
    for(int i = 0; i < n; i += 2) {
        sum += a[i];
    }
    return sum;
}

int main() {
    /*Az a tömb elemeinek add kezdőértékként: {2, -10, 8, 3, 11, 20, 7}*/
    int a[7] = {2, -10, 8, 3, 11, 20, 7};

    /*A főprogramból hívd meg a függvényt és mutasd meg a tömb elemeit!*/
    kiirsorba(a, 7);

    /*Deklarálj egy új tömböt, aminek a neve haromszoros legyen!
    (gondolkodj el, milyen típusú adatok kerülnek majd bele).*/
    int haromszoros[7];
    /*Az új tömbbe az a tömb elemeinek háromszorosa kerüljön!*/
    for (int i = 0; i < 7; i++) {
        haromszoros[i] = a[i] * 3;
    }

    /*Írasd ki a haromszoros tömb elemeit is a kész függvénnyel!*/
```

```
    kiirsorba(haromszoros, 7);

    /*Hívd meg a függvényt a főprogramból
    és írasd ki mindkét tömb páros indexű elemeinek az összegét,
    üzenetek kíséretében!*/
    cout << "Páros indexű elemek összege a tombben: " << pindossz(a, 7) << endl
    cout << "Páros indexű elemek összege a haromszoros tombben: " << pindossz(h

    /*Számítsd ki az eredeti tömb páros indexű elemeinek az átlagát is, és íras
    cout << "Paros indexu elemek atlaga az a tombben: " << (float)pindossz(a, 7

    return 0;
}
```

B feladat

1. Deklarálj tömböt 10 egész szám tárolására!
2. Töltsd fel véletlen számokkal az `[10,205]` intervallumból! Addig végezd a feltöltést, míg a tömb tele nem lesz, vagy az első 100-asig. Tehát lehet ilyen tömböd: `{56, 200, 100}` vagy `{10,20,30,40, 60, 80, 90, 200, 199, 198}`, de nem lehet `{10,20,30,40, 60, 80, 90, 100, 199, 198}`. A ciklus típusát magad válaszd meg!
3. Számold meg, hogy hány elemet vittél be a tömbbe! Írasd is ki, miután kiléptél a ciklusból!
4. Írj függvényt, ami kiírja a képernyőre a tömb elemeit 3 oszlopba!
   `void kiir(int *a, int n)`
   `a` - a tömb neve, `n` - az elemeinek a száma
   Ügyelj arra, hogy a tomb végén lévő szemetet ne írasd ki, ha kevesebb mint 10 elem van!
5. Írj függvényt, ami megszámolja, hogy a tömbben hány kétjegyű szám van! A függvényt hívd meg helyesen a főprogramból, és írasd ki az eredményt!
   `int ketj(int *a, int n)`
   `a` - a tömb neve, `n` - az elemeinek a száma
   A főprogramból hívd meg a függvényt, írasd ki a kétjegyűek számát!

6. A főprogramban végezd a következő műveletet: a tömb minden páros elemét oszd el 2-vel! A páratlanokat ne változtasd!
7. Írasd ki a tömb elemeit, a már kész függvénnyel!
8. Írasd ki megint a kétjegyűek számát is! Használd a kész függvényt!

Megoldás

```cpp
#include <iostream>

using namespace std;

/*Írj függvényt, ami kiírja a képernyőre a tömb elemeit 3 oszlopba!
Ügyelj arra, hogy a tomb végén lévő szemetet ne írasd ki, ha kevesebb mint 10 e
void kiir(int *a, int n) {
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
        if(i % 3 == 2) {
            cout << endl;
        }
    }
    cout << endl;

}
/*Írj függvényt, ami megszámolja, hogy a tömbben hány kétjegyű szám van!
A függvényt hívd meg helyesen a főprogramból, és írasd ki az eredményt! */
int ketj(int *a, int n) {
    int db = 0;
    for (int i = 0; i < n; i++) {
        if( (a[i] > 10) && (a[i] < 100)) {
            db++;
        }
    }
    return db;
}

int main() {
    /*Deklarálj tömböt 10 egész szám tárolására!*/
    int a[10];

    srand(time(0));
    /*for (int i = 0; i < 10; i++) {
        int random = rand() % 196+10;
        if(random != 100) {
            a[i] = random; // 10-205
        }
        else {
```

```
        break;
    }

}*/
/*Töltsd fel véletlen számokkal az `[10,205]` intervallumból!
Addig végezd a feltöltést, míg a tömb tele nem lesz,
vagy az első 100-asig.
Tehát lehet ilyen tömböd: `{56, 200, 100}`
vagy `{10,20,30,40, 60, 80, 90, 200, 199, 198}`,
de nem lehet `{10,20,30,40, 60, 80, 90, 100, 199, 198}`.
A ciklus típusát magad válaszd meg!*/
int random = rand() % 196+10;
int ind = 0;
while ((random != 100) && (ind < 10)) {
    int random = rand() % 196+10;
    a[ind] = random;
    /*Számold meg, hogy hány elemet vittél be a tömbbe! Írasd is ki, miután
    ind++;
}

cout << ind << " elemet vittem be\n";

kiir(a, ind);

/*A főprogramból hívd meg a függvényt, írasd ki a kétjegyűek számát!*/
cout << "Ketjegyu szamok szama: " << ketj(a, ind) << endl;

cout << "<---------------Osztas utan--------------->\n";

/*A főprogramban végezd a következő műveletet: a tömb minden páros elemét o
A páratlanokat ne változtasd!*/
for(int i = 0; i < ind; i++) {
    if (a[i] % 2 == 0) {
        a[i] = a[i] / 2;
    }
}
/*Írasd ki a tömb elemeit, a már kész függvénnyel!*/
kiir(a, ind);
```

```
    /*Írasd ki megint a kétjegyűek számát is! Használd a kész függvényt!*/
    cout << "Ketjegyu szamok szama: " << ketj(a, ind) << endl;

    return 0;
}
```

C feladat

1. Hozz létre új projektumot! A projektum mappájában hozz létre egy txt filet, aminek a neve `xx.txt` legyen! Írj bele egész számokat szóközzel elválasztva, legalább 8 db-ot! Őrizd meg a filet!
2. A programban deklarálj egy 15 elemű tömböt egész számok tárolására!
3. Nyisd meg a filet olvasásra, ellenőrizd, hogy helyesen nyílt e meg a file!
4. A file elemeit olvasd be a tömbbe! Ügyelj arra, hogy a tömb határát (15 elem) ne lépd túl!
5. Írasd ki, hogy hány elemet olvastál be a tömbbe!
6. Írj függvényt, ami kiszámítja az elemek összegét!
   `int osszeg(int *a, int n)`
   `a`-a tömb neve, `n` - az elemeinek a száma
7. A főprogramból hívd meg a függvényt, és írasd ki az összeget!
8. Számítsd ki az elemek átlagát is! Írasd ki!
9. Próbáld ki úgy is, hogy kevesebb mint 15 szám van a file-ban, és úgy is, hogy több!
   Megoldás

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int ketj(int *a, int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += a[i];
    }
    return sum;
}

int main() {
    int a[15];

    ifstream f("xx.txt");
    if(!f.is_open()) {
        cout << "Error opening file\n" << endl;
        return 1;
    }

    int i = 0;
    while(!f.eof() && (i < 15)) {
        f >> a[i];
        //cout << a[i] << " ";
        i++;
    }
    //cout << endl;

    cout << i << " darab szamot olvastam be\n";

    cout << "A szamok osszege: " << ketj(a, i) << endl;

    cout << "A szamok atlaga: " << (float)ketj(a, i) / i << endl;

    return 0;
}
```

3. Pótóra (2024. Március 19.)

X csoport

Írj programot:

1. Deklarálj egy 10 elemű tömböt egész számok tárolására, a neve legyen `elso`!
2. Add neki kezdőértékként azt a tömböt, aminek az elemei: `5, 8, 11, 2, 19, 21, 33, 42`
3. Írj függvényt, ami kiírja a tömb elemit 4 oszlopba! A kiírás után legyen két üres sor!
   `void kiir3(int *a, int n)`
4. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb elemeit 4 oszlopban! (mind a 10 elemet)
5. Újra írasd ki az elemeket 4 oszlopba, de most csak a “hasznos” elemeket!
6. Írj függvényt, ami kiszámítja a tömb `[7,21]` intervallumba eső elemeinek az összegét!
   `int osszeg721(int*a, int n)`
7. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kíséretében!
8. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen `masodik`
9. Az új tömb elemei az eredeti tömb elemeinek háromszorosánál 15-tel kisebb számok legyenek!
10. Írasd ki ezt a tömböt is 4 oszlopba!
11. Számítsd ki ennél a tömbnél is a `[7,21]` intervallumba eső elemek az összegét! Írasd ki az eredményt üzenet kíséretében!
12. Deklarálj még két, szintén 10 elemű tömböt, egész számok tárolására, `harmadik` és `negyedik` néven!
13. A `harmadik` tömbbe kerüljenek a `masodik` tömbből azok az elemek, amik párosak vagy 10-nél kisebbek, a `negyedik`-be pedig a többi elem! Ügyelj arra, hogy ne legyenek az új tömbökben “lyukak”!
14. Írasd ki, hogy hány eleme van a `harmadik` és a `negyedik` tömbnek!

15. Írasd ki csak a “hasznos” elemeiket!
16. Az `első` tömb 5. elemét csökkentsd 11-gyel!
17. Írasd ki újra az `elso` és a `masodik` tömböt, mindkettőt 3-3 oszlopba!
    Megoldás

```cpp
#include <iostream>

using namespace std;

//3. Írj függvényt, ami kiírja a tömb elemit 4 oszlopba! A kiírás után legyen k
void kiir3(int *a, int n) {
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
        if ((i+1) % 4 == 0) {
            cout << endl;
        }
    }
    cout << endl << endl;
}

//6. Írj függvényt, ami kiszámítja a tömb [7,21] intervallumba eső elemeinek az
int osszeg721(int*a, int n) {
    int osszeg = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] >= 7 && a[i] <= 21) {
            osszeg += a[i];
        }
    }
    return osszeg;
}

int main() {
// 1. Deklarálj egy 10 elemű tömböt egész számok tárolására, a neve legyen elso
// 2. Add neki kezdőértékként azt a tömböt, aminek az elemei: 5, 8, 11, 2, 19,
    int elso[10] = {5,8,11,2,19,21,33,42};
// 4. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb elemeit 4 o
    cout << "Az elso tomb elemei 4 oszlopba rendezve: \n";
    kiir3(elso, 10);
// 5. Újra írasd ki az elemeket 4 oszlopba, de most csak a “hasznos” elemeket!
    cout << "Az elso tomb hasznos elemei 4 oszlopba rendezve: \n";
    kiir3(elso, 8);
// 7. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kísé
    cout << "A tomb [7,21] intervallumba eso elemeinek az osszege: " << osszeg7
// 8. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen maso
```

```
    int masodik[10];
// 9. Az új tömb elemei az eredeti tömb elemeinek háromszorosánél 15-tel kisebb
    for (int i = 0; i < 10; i++) {
        masodik[i] = elso[i] * 3 - 15;
    }
// 10. Írasd ki ezt a tömböt is 4 oszlopba!
    cout << "A masodik (= elso * 3 - 15) tomb elemei 4 oszlopba rendezve: \n";
    kiir3(masodik, 10);
// 11. Számítsd ki ennél a tömbnél is a [7,21] intervallumba eső elemek az össz
    cout << "A masodik tomb [7,21] intervallumba eso elemeinek az osszege: " <<
// 12. Deklarálj még két, szintén 10 elemű tömböt, egész számok tárolására, har
    int harmadik[10], negyedik[10];
// 13. A harmadik tömbbe kerüljenek a második tömbből azok az elemek, amik páro
    int h = 0, k = 0;
    for (int i = 0; i < 10; i++) {
        if (masodik[i] % 2 == 0 || masodik[i] < 10) {
            harmadik[h] = masodik[i];
            h++;
        } else {
            negyedik[k] = masodik[i];
            k++;
        }
    }
// 14. Írasd ki, hogy hány eleme van a harmadik és a negyedik tömbnek!
    cout << "A harmadik tomb merete: " << h << endl;
    cout << "A negyedik tomb merete: " << k << endl;
// 15. Írasd ki csak a “hasznos” elemeiket!
    cout << "A harmadik (masodikbol paros vagy < 10) tomb hasznos elemei 4 oszl
    kiir3(harmadik, h);
    cout << "A negyedik (masodikbol paratlan es >= 10) tomb hasznos elemei 4 os
    kiir3(negyedik, k);
// 16. Az első tömb 5. elemét csökkentsd 11-gyel!
    elso[4] -= 11;
// 17. Írasd  ki újra az elso és a masodik tömböt, mindkettőt 3-3 oszlopba!
    cout << "Az elso tomb elemei 3 oszlopba rendezve: \n";
    for (int i = 0; i < 10; i++) {
        cout << elso[i] << "\t";
        if ((i+1) % 3 == 0) {
            cout << endl;
```

```
        }
    }
    cout << endl << endl;
    cout << "A masodik tomb elemei 3 oszlopba rendezve: \n";
    for (int i = 0; i < 10; i++) {
        cout << masodik[i] << "\t";
        if ((i+1) % 3 == 0) {
            cout << endl;
        }
    }
    cout << endl << endl;

}
```

Y csoport

Írj programot:

1. Deklarálj egy 15 elemű tömböt egész számok tárolására, a neve legyen `e`!
2. Az elemei legyenek véletlen számok a `[100, 120]` intervallumból! Ha megjelenik az első `110`-es érték, hagyd abba a tömb feltöltését! Ügyelj arra, hogy ne legyen több 15 elemnél!
3. Írasd ki, hogy hány elem lett végül a tömbben!
4. Írj függvényt, ami kiírja a tömb elemeit 5 oszlopba! A kiírás előtt és után is egy-egy üres sor legyen!
   `void kiir5(int *a, int n)`
5. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb elemeit 5 oszlopban!
6. Írj függvényt, ami megszámolja, hogy a tömbben hány `110`-nél nagyobb páratlan szám van!
   `int paratlan110(int*a, int n)`
7. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kíséretében!
8. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen `m`

9. Az `m` tömbbe írj az `e` tömb páratlan elemeinél `4`-gyel kisebbet, a párosakhoz viszont adj hozzá `9`-et! Az elemek sorrendje ne változzon!
10. Írasd ki ezt a tömböt is 5 oszlopba!
11. Ebben a tömbben hány hány `110`-nél nagyobb páratlan szám van? Írasd ki az eredményt üzenet kíséretében!
12. Deklarálj még egy, szintén `10` elemű tömböt, egész számok tárolására, `h` néven!
13. Ebbe a tömbbe kerüljenek az `e` tömbből azok az elemek, amik öttel oszthatóak vagy a középső számjegyük `0`! Ügyelj arra, hogy ne legyenek a tömbben “lyukak”!
14. Írasd ki, hogy hány eleme van a `h` tömbnek!
15. Írasd ki csak a “hasznos” elemeit, 5 oszlopba!
16. Az `m` tömb 10. elemét szorzod meg `3`-mal!
17. Írasd ki újra az `m` tömb elemeit, 5 oszlopba!
    Megoldás

```cpp
#include <iostream>

using namespace std;

// 4. Írj függvényt, ami kiírja a tömb elemeit 5 oszlopba! A kiírás előtt és ut
void kiir5(int *a, int n) {
    cout << endl;
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
        if ((i+1) % 5 == 0) {
            cout << endl;
        }
    }
    cout << endl;
}

// 6. Írj függvényt, ami megszámolja, hogy a tömbben hány 110-nél nagyobb párat
int paratlan110(int*a, int n) {
    int db = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] > 110 && a[i] % 2 == 1) {
            db++;
        }
    }
    return db;

}

int main() {
    srand(time(0));
// 1. Deklarálj egy 15 elemű tömböt egész számok tárolására, a neve legyen e!
    int e[15];
// 2. Az elemei legyenek véletlen számok a [100, 120] intervallumból! Ha megjel
    int i = 0;
    do {
        e[i] = rand() % 21 + 100;
        i++;
    }while ((i < 15) && (e[i - 1] != 110));
// 3. Írasd ki, hogy hány elem lett végül a tömbben!
```

```
    cout << "A tombben " << i << " elem van." << endl;

// 5. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb elemeit 5 o
    cout << "Az e tomb elemei 5 oszlopba rendezve: \n";
    kiir5(e, i);

// 7. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kísé
    cout << "A tombben " << paratlan110(e, i) << " db 110-nel nagyobb paratlan
// 8. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen m.
    int m[15];
// 9. Az m tömbbe írj az e tömb páratlan elemeinél 4-gyel kisebbet, a párosakho
    for (int j = 0; j < i; j++) {
        if (e[j] % 2 == 1) {
            m[j] = e[j] - 4;
        } else {
            m[j] = e[j] + 9;
        }
    }
// 10. Írasd ki ezt a tömböt is 5 oszlopba!
    cout << "Az m (e-bol paratlan - 4, paros + 9) tomb elemei 5 oszlopba rendez
    kiir5(m, i);
// 11. Ebben a tömbben hány hány 110-nél nagyobb páratlan szám van? Írasd ki az
    cout << "A m tombben " << paratlan110(m, i) << " db 110-nel nagyobb paratla
// 12. Deklarálj még egy, szintén 10 elemű tömböt, egész számok tárolására, h n
    int h[10];
// 13. Ebbe a tömbbe kerüljenek az e tömbből azok az elemek, amik öttel oszthat
    int j = 0;
    for (int k = 0; k < i; k++) {
        if (e[k] % 5 == 0 || (e[k] / 10) % 10 == 0) {
            h[j] = e[k];
            j++;
        }
    }

// 14. Írasd ki, hogy hány eleme van a h tömbnek!
    cout << "A h tomb merete: " << j << endl;

// 15. Írasd ki csak a “hasznos” elemeit, 5 oszlopba!
    cout << "A h (e-bol 5-tel oszthato vagy a kozepsoben 0) tomb hasznos elemei
```

```
    kiir5(h, j);

// 16. Az m tömb 10. elemét szorzod meg 3-mal!
    m[9] *= 3;

// 17. Írasd ki újra az m tömb elemeit, 5 oszlopba!
    cout << "Az m tomb elemei 5 oszlopba rendezve: \n";
    kiir5(m, i);
    return 0;


}
```

Z csoport

Írj programot:

1. Deklarálj egy 7 elemű tömböt egész számok tárolására, a neve legyen `r`!
2. Az elemeit a felhasználó vigye be, `0` végjelig! Ügyelj arra, hogy ne legyen több `7` elemnél!
3. Írasd ki, hogy hány elem került a tömbbe!
4. Írj függvényt, ami kiírja a tömb elemeit egymás mellé, vesszővel elválasztva! Az utolsó elem után ne legyen vessző! A kiírás előtt és után is egy-egy üres sor legyen!
   `void kiir(int *a, int n)`
5. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb `7` elemét a megadott módon!
6. Újra írasd ki a tömböt, de most már csak a “hasznos” elemeit, amiket a felhasználó vitt be (a végjelet sem)!
7. Írj függvényt, ami meghatározza a tömb legkisebb elemét!
   `int lkis(int*a, int n)`
8. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kíséretében!
9. Változtasd meg a tömb 4. elemét úgy, hogy 25 legyen, ha páros volt, egyébként vonj ki belőle 75-öt!

10. Most mi a tömb legkisebb eleme? Írasd ki!
11. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen `s`
12. Az `s` tömbbe írj az `r` tömb reciprok értékénél 2-vel nagyobb számot!
13. Írasd ki ezt a tömböt is! Ez nem megy a kész függvénnyel, a főprogramban dolgozz rajta!
14. Deklarálj még egy, szintén `7` elemű tömböt, egész számok tárolására, `t` néven!
15. A `t` tömbbe kerüljenek az `r` tömbből azok az elemek, `10` és `70` közé esnek! Ügyelj arra, hogy ne legyenek a tömböken “lyukak”!
16. Írasd ki, hogy hány eleme van a `t` tömbnek!
17. Írasd ki csak a “hasznos” elemeit a függvénnyel!
    Megoldás

```cpp
#include <iostream>

using namespace std;

// 4. Írj függvényt, ami kiírja a tömb elemeit egymás mellé,
// vesszővel elválasztva! Az utolsó elem után ne legyen vessző!
// A kiírás előtt és után is egy-egy üres sor legyen!
void kiir(int *a, int n) {
    for (int i = 0; i < n; i++) {
        cout << a[i];
        if (i < n - 1) {
            cout << ",";
        }
    }
    cout << endl;
}

// 7. Írj függvényt, ami meghatározza a tömb legkisebb elemét!
int lkis(int*a, int n){
    int kis = a[0];
    for(int i = 1; i < n; i++) {
        if(a[i] < kis) {
            kis = a[i];
        }
    }
    return kis;
}

int main() {
// 1. Deklarálj egy 7 elemű tömböt egész számok tárolására, a neve legyen r!
    int r[7];

// 2. Az elemeit a felhasználó vigye be, 0 végjelig!
// Ügyelj arra, hogy ne legyen több 7 elemnél!
    cout << "Kerek 7 szamot" << endl;
    int i = 0;
    do  {
        cin >> r[i];
        i++;
```

```
    } while ((i < 7) && (r[i - 1] != 0));
// 3. Írasd ki, hogy hány elem került a tömbbe!
    cout << "A tombben " << i << " elem van" << endl;
// 5. A főprogramból hívd meg a fenti függvényt, hogy lássuk a tömb 7 elemét a
    kiir(r,7);
// 6. Újra írasd ki a tömböt, de most már csak a “hasznos” elemeit, amiket a fe
    kiir(r,i);

// 8. Hívd meg a főprogramból a függvényt, és írasd ki az eredményt üzenet kísé
    cout << "A legkisebb elem: " << lkis(r, i) << endl;

// 9. Változtasd meg a tömb 4. elemét úgy, hogy 25 legyen, ha páros volt, egyéb
    if(r[4] % 2 == 0) {
        r[4] = 25;
    }
    else {
        r[4] -= 75;
    }
// 10. Most mi a tömb legkisebb eleme? Írasd ki!
    cout << "A tomb uj legkisebb eleme: " << lkis(r,i);
// 11. Hozz létre egy új tömböt, amiben ugyanennyi elem lesz! A neve legyen s.
    double s[7];
// 12. Az s tömbbe írj az r tömb reciprok értékénél 2-vel nagyobb számot!
    for(int j = 0; j < i; j++) {
        s[j] = (double)1 / r[j] + 2;
    }
// 13. Írasd ki ezt a tömböt is! Ez nem megy a kész függvénnyel, a főprogramban
    for (int j = 0; j < i; j++) {
        cout << s[i];
        if (j < i - 1) {
            cout << ",";
        }
    }
// 14. Deklarálj még egy, szintén 7 elemű tömböt, egész számok tárolására, t né
    int t[7];
// 15. A t tömbbe kerüljenek az r tömbből azok az elemek, 10 és 70 közé esnek!
    int k = 0;
    for(int j = 0; j < i; j++){
        if(r[j] > 10 && r[j] < 70) {
```

```
            t[k] = r[j];
            k++;
        }
    }
// 16. Írasd ki, hogy hány eleme van a t tömbnek!
    cout << "A t tombnek " << k << " eleme van." << endl;
// 17. Írasd ki csak a “hasznos” elemeit a függvénnyel!
    kiir(t,k);
}
```

4. pótóra (2024. Március 26.)

A feladat

> A személyi számunkról (JMBG)
> Tudjuk, hogy a JMBG 13 jegyű, és kódként használatos. A következő elemekből épül fel:
> • 1-2 számjegy a születés napja (hányadika);
> • 3-4 a születés hónapja (hanyadik);
> • 5-6-7 a születési évszám utolsó 3 számjegye;
> • 10. számjegy – ha 0, akkor a személy neme férfi, ha 5, akkor a személy neme nő;
> *példa: 1302989825066 ez a személy 1989.02.13-án született, nő*

1. Egy szöveges fileba írjuk be 3 személy JMBG-jét!
2. Olvassuk be sztringbe egy személy személyi számát (JMBG)! (például `1202928820042`).
3. Válaszd ki külön változóba a születési évet;
4. Alkosd meg belőle a teljes születési évet leíró karaktereket (4 jegyű legyen)!
5. Válaszd ki külön változóba a születés hónapját!
6. Válaszd ki külön változóba a születés napját!
7. A nemre vonatkozó karaktertől és a születési évtől függően egy változó értéke legyen fiú/lány vagy férfi/nő *(akik 2000-ben, vagy azóta születtek, azok esetén a fiú/lány, a többiekre a férfi/nő)* kerüljön a

változóba!
8. Hozz létre egy üzenetet a következő formában:
   > Ez a fiú/lány/férfi/nő eeee.hh.nn –án született.”

B feladat

1. Két sztring típusú változóba adjunk meg kezdőértékként (a tesztelés könnyítése miatt) be két személyi számot! (pl `1112978820077` és `0102003825011`).
   `string attila=”1112978820077”, anna=”0102003825011”;`
2. Határozzuk meg a születés hónapját és napját!
3. Ezekből az adatokból döntsd el, hogy az év folyamán kinek van/volt/lesz előbb a születésnapja!
4. Írj erről üzenetet!
5. Hány nappal lesz korábban a születésnap? (vedd figyelembe a hónapok valós hosszát és azt is, hogy a mostani évszázadokban minden néggyel osztható évszám szökőév).
   Ehhez a feladathoz érdemes létrehozni egy tömböt a hónapok hosszával:
   `int ho[12]={31,28,31,30,31,30,31,31,30,31,30,31}`
   vagy, lehet, hogy praktikusabb:
   `int hojobb[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}`
6. A felhasználótól kérj be egy dátumot! (`hh nn`)
7. Döntsd el, hogy az adott dátumtól számítva melyik személy fog előbb születésnapot ünnepelni!

C feladat

A projektumod mappájába másold be a szemszam.txt szöveges dokumentumot, aminek a tartalma:

```
Attila 1112978820077 Emese 0102003825011
Pityu 1311985720018 Lenke 0102005825014
Monika 1202014615045 Kitti 1102003825004
```

1. Nyisd meg a filet-olvasásra!

2. Deklarálj egy sztringet, amibe a el tudod helyezni a fileból beolvasott egy-egy sort!
3. `getline`-nal olvasd be a file első sorában lévő adatokat!
4. A kapott sztringben keresd meg az első szóköz helyét! Az eredményt tárold!
5. Válaszd ki külön sztringbe a személy nevét!
   `string nev (string sor)`
6. Válaszd ki külön sztringbe a személyi számot!
7. Válaszd ki külön sztringbe az JMBG-ből a napra vonatkozó részt! Erre kell egy változó!
8. Alakítsd ezt egész számmá! Erre is kell egy vátozó!
9. Válaszd ki sztringbe a hónapra vonatkozó részt is! Erre is kell változó!
10. Alakítsd egész számmá! Tárold!
11. Töröld a bejövő sztringből az első nevet és a az első személyi számot is! Írasd ki a maradékot, hogy lásd, jól töröltél-e?
12. Ebben a sztringben keresd meg az első szóköz helyét! Az eredményt tárold!
13. Válaszd ki külön sztringbe a személy nevét! (használd a kész függvényt!)
14. Válaszd ki külön sztringbe a személyi számot!
15. Válaszd ki külön sztringbe az JMBG-ből a napra vonatkozó részt!
16. Alakítsd ezt egész számmá!
17. Válaszd ki sztringbe a hónapra vonatkozó részt is!
18. Alakítsd egész számmá!
    Így az egy sorban lévő két személyről ki tudod íratni a következő mondatot:
    > Attila 12.11-én, Emese pedig 2.1-én ünnepli a születésnapját.

    A hónap kiírására használhatod a következő tömböt is:
    `string ho[13]={“”, “januar”, ..., “december”};`
    Ebben az esetben a mondat barátságosabb lesz:
    > Attila december 11-én, Emese pedig február 1-én ünnepli a születésnapját.

19. Állapítsd meg, hogy kinek van korábban a születésnapja, és írj róla üzenetet:

> pl. Emese ünnepel korábban.

20. A fileban lévő összes sort dolgozd fel ugyanígy.
21. Zárd be a filet!

D feladat

1. Másold a projektumod mappájába a datumok.txt filet, aminek tartalma:

```
12/1/21
21/11/93
1/1/89
23/5/02
17/9/17
15/12/10
```

   Minden sorban 1-1 születési dátum található, nap/hó/év formátumban.
2. Nyisd meg a filet olvasásra! Ellenőrizd, hpgy megnyílt-e helyesen a file! Ha nem, adj üzenetet!
3. Soronként (`getline`) olvasd be az adatokat.
4. Minden sort bonts szét részekre: napra, hónapra, évre.
5. Határozd meg a teljes évszámot: ha az évszám első számjegye kevesebb mint 3, akkor a dátumot a kétzeres évekből valónak tekintjük. A többi évszám az 1900-as ávekből való legyen!
6. Számítsd ki hogy az idén hány éves az a személy, akinek ez az születési dátuma!
7. Nyiss egy új fájlt írásra, a neve legyen `datmagyar.txt` .
8. Írd ki a képernyőre következő mondatokat: nn. szuletesnap: 2024. hhh dd.
   pl az első sor alapján: 3. születésnap: 2024. jan 12.
9. Írd ki a fájlba is a mondatokat, külön sorokba!

E feladat

1. Másold a projektumod mappájába a furcsak.txt file, aminek a tartalma:

```
Kalocsai Anna+Piroska 1956.1.11. beszerzo
Nagy+Nemedi Kelemen 2001.12.3. kereskedo
Szemeredi Ivan 2008.3.10 vizvezetekszerelo
Kispal Peter+Zalan 1994.10.4. szakacs
Toth+Ugyonka Kitti+Anita 2000.4.6. cukrasz
```

A file minden sorában egy személy teljes neve, születési dátuma és foglalkozása található.
A teljes név vezetéknév keresztnév sorrendben van a fájlban.
Bizonyos személyek vezetékneve 2 részből áll. Ilyenkor a két rész között + jel jelzi a határt.
Más személyeknek két keresztnevük is van. Ilyenkor a két keresztnév között + jel jelzi a határt.
A vezetéknév és a keresztnév között mindenképpen szóköz van.
Egy szóköz választja el egymástól a születési dátumot és foglalkozást.

2. Nyisd meg a filet olvasásra! Ellenőrizd, hpgy megnyílt-e helyesen a file! Ha nem, adj üzenetet!
3. Soronként (`getline`) olvasd be az adatokat.
4. Egy sort bonts szét több adatra: `vezeteknev` , `keresztnev` , `szul_ev` , `szul_ho` , `szul_nap` , `foglalkozas` .
5. Minden személyről a következő formában írd ki az adatait:

   > Vezetéknev: Toth Ugyonka
   > Keresztnev: Kitti Anita
   > Szuletett: 2000 aprilis 6.
   > Foglalkozasa: cukrasz

6. Akik a kétezres években születtek, azoknak a keresztnevét és a foglalkozását írd ki a `fiatalok.txt` fileba, soronként:

```
Kelemen kereskedo
Ivan vizvezetekszerelo
...
```

F feladat

1. Másold a projektumod mappájába a varosok.txt filet, aminek tartalma:

```
Zenta Szerbia Europa 18704 293
Magyarkanizsa Szerbia Europa 9871 400
Los Angeles USA Amerika 3967000 1302
Budapest Magyarorszag Europa 1756000 525
Moszkva Oroszorszag ? 11920000 2511
```

Minden sorban 1-1 városról megtaláljuk, a következő adatokat: elnevezés, ország, kontinens, lakosság száma (fő), területe (km2).

2. Nyisd meg a filet olvasásra! Ellenőrizd, hpgy megnyílt-e helyesen a file! Ha nem, adj üzenetet!
3. Soronként (`getline`) olvasd be az adatokat.
4. Minden sort bonts szét részekre, és tárold a megfelelő típusú változókban: `elnevezes` , `orszag` , `foldresz` , `lakossag` , `terulet` .
5. A képernyőre írj ki ilyen szerkezetű mondatokat:
   > Zenta egy varos Szerbia-ban, 18704 lakosa van!
6. Minden városnak számítsd ki a népsűrűségét! A mértékegység legyen `fő/km2` .
7. Számítsd ki, hogy a megnevezett városok lakossága hányszoros Zenta lakosságához képest!
8. Nyiss egy új fájlt írásra, a neve legyen `v.txt` .
9. A fileba írd ki az európai városokat és a népsűrűségüket és azt hogy hányszor van több lakosuk mint Zentának. Az információkat foglald mondatba! Ha egy város kisebb Zentánál, ott ne szerepeljen ez a szám. Pl:
   > Budapestnek 9,38-szor tobb lakosa van mint Zentanak, a nepsurusege 3344 fo/km2.

   vagy:
   > Magyarkanizsa nepsurusege 24,68 fo/km2.

   Minden város mondata kerüljön külön sorba!

G feladat

A `gyakorlat.txt` fileba másold át a következő kódokat:

```
F4215
F2319
L1321
L2222
L1303
F2110
```

A első karkater azt jelöli, hogy fiú vagy lány a személy akinek a kódot adták,
A második karakter, hogy hanyadik osztályos
A harmadik karakter: 1-matekos 2-képzős 3-infós
A 4-5. karakter: az évszám, amikor a kódot kapta
Pl az első kódból ez a mondat állítható össze:

> F4215 a kódja annak, a fiúnak, aki 2015-ben negyedikes volt a képzős szakon.

FELADAT: Mindegyik kódhoz írasd ki a megfelelő mondatot.

H feladat

A `mindenfele.txt` fileba másold át a következő időpontokat:

```
1992.8.11. 20:53:23
2023.11.12. 15:42:11
2011.3.11. 11:7:10
2013.1.1. 1:2:3
```

Írass ki minden sor alapján egy ehhez hasonló mondatot:

> 1992-ben, augusztus 11-én délután 8:53-kor történt valami.

5. Pótóra (2024. Április 16.)

A feladat

Hozz létre egy struktúra –típust:

```
typedef struct {
    int ora;
    int perc;
} Idopont;
```

1. Deklarálj két Idopont típusú változót.
2. Az egyikbe egy véletlen időpont kerüljön 8:00 és 9:59 között!
3. A másikba a felhasználótól kérj egy időpontot!
4. Írasd ki mindkét időpontot oo:pp formátumban! Írhatsz erre függvényt is!
5. Állapítsd meg, hogy a véletlen időpont korábbi, későbbi, vagy egybeesik a felhasználó által bevitt időponttal! Számítsd ki az eltérést is, és írd ki üzenet formájában: “A felhasználó időpontja hh óra pp perccel korábbi/ későbbi mint a véletlen időpont ”, vagy “A két időpont egybeesik”.
6. Hozz létre egy `20` elemű `Idopont` típusú tömböt és töltsd fel 8:00 és 14:59 közötti időpontokkal.
7. Írasd ki a tömb elemeit egymás alá oo:pp formátumban
8. Válaszd ki a legkorábbi és a legkésőbbi időpontot, és számítsd ki, hogy hány perc az eltérés közöttük, majd írd ki az eltérést oo:pp formátumban is.

B feladat

Írj programot, ami síkbeli pontok tárolására létrehoz egy típust (struktúrát) és tartalmazza a következő függvényeket:

1. Pont beírása *(függvény bemenő paraméter nélkül, kijön belőle egy pont)*
2. Pont kiírása *(függvény-void)*
3. Megállapítani, hogy két pont vízszintes szakaszt alkot-e? A függvény kimenete legyen 1 ha vízszintes, és 0 ha nem. *(függvény)*
4. Kiszámítani a szakasz hosszát (ehhez is két pont kell) *(függvény)*
   ha `A(x1,y1)` és `B(x2,y2)`, akkor $|AB| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
5. Megállapítani, hogy a pont melyik negyedhez tartozik *(függvény, 1, 2, 3 vagy 4 értéket adjon vissza)*
   FŐPROGRAM
6. A főprogramban kérj be `3` pontot (`m` , `n` , `p`) a felhasználótól amik egy

háromszög csúcsai! *(ciklus, függvényhívás)*

7. Számítsd ki az oldalak hosszát (mn, np, mp)! *(ciklus, függvényhívás)*
8. Számítsd ki a háromszög kerületét, írd ki!
9. Számítsd ki a területét is *(Heron képlettel)* $Haahároms zögoldalainakhosszaa, bésc - velvanjelölve, akkorT = \sqrt{s * (s-a) * (s-b) * (s-c)}, ahols = \frac{a+b+c}{2}$
10. Állapítsd meg, hogy van-e a háromszögnek vízszintes oldala, és írj róla üzenetet! Használd a megírt függvényt!
11. Metszi-e a háromszöget az y tengely? Állapítsd meg, és írd ki a választ!
12. És az x tengely? Állapítsd meg, és írd ki a választ!

C feladat

Misi kedvenc „étele” a madártej. Kell hozzá

- 0,8 liter tej
- 5 tojás
- 0,3 kg cukor
- 2 csomag vaníliás puding

Ebből 2 adag lesz.
A hozzávalók boltonként más-más áron kerülnek a polcra, ezért írjunk programot arra, hogy kiszámítsuk egy adag árát.

1. Hozz létre struktúrát:
    - bolt neve
    - tej ára (din/liter)
    - tojás ára (din/db)
    - cukor ára (din/kg)
    - vaníliás puding (din/csomag)

```
typedef struct {
    string nev;
    int tejAr;
    int tojasAr;
    int cukorAr;
    int pudingAr;
}
```

Az árak egész számok.

2. Vigyük be 3 boltból a termékek árait, tároljuk őket fileban!
pl.

```txt
bolt1 100 20 70        30
bolt2 125 22 65 40
bolt3 140 20 58 30
bolt4 135 19 88 42
```

3. Olvassuk vissza a fileból az adatokat, listázzuk ki a képernyőre táblázatos formában.

```
    while (!fI.eof()) {
        string line;
        getline(fI, line);
        bolt.nev = line.substr(0, line.find(" "));
        line = line.substr(line.find(" ") + 1);
        bolt.tejAr = stoi(line.substr(0, line.find(" ")));
        line = line.substr(line.find(" ") + 1);
        bolt.tojasAr = stoi(line.substr(0, line.find(" ")));
        line = line.substr(line.find(" ") + 1);
        bolt.cukorAr = stoi(line.substr(0, line.find(" ")));
        line = line.substr(line.find(" ") + 1);
        bolt.pudingAr = stoi(line.substr(0, line.find(" ")));
    }
//Modositani arra, hogy ne csak egy boltot taroljunk, hanem egy tombben legyenek
//Max 3 bolt adatait olvassuk be
//Mindenhol boltok -> boltok[i]
//a vegere egy i++;
//elotte ne felejtsuk el letrehozni i-t es a tombot

    cout << "Nev\tTej\tTojas\tCukor\tPuding" << endl;
    cout << "---------------------------------------" << endl;
    for(int j = 0 ; j < i; j++) {
        cout << bolt[j].nev << "\t";
        cout << bolt[j].tejAr << "\t";
        cout << bolt[j].cukorAr << "\t";
        cout << bolt[j].pudingAr << "\n";
    }
```

4. Számítsuk ki mindegyik esetben, hogy mennyibe kerül egy adag madártej alapanyaga.

```
int ar = 0;
for(int i = 0; i < 3; i++) {
    ar += bolt[i].tejAr * 0.8;
    ar += bolt[i].tojasAr * 5;
    ar += bolt[i].cukorAr * 0.3;
    ar += bolt[i].pudingAr * 2;
}
```

5. A munkadíj és egyéb költségek címén az eddig kiszámított összeg 70%-át számoljuk még el.
Egy másik fileba írjuk ki: a bolt nevét és egy adag költségét anyagi és tobábbi költségekre bontva.
Pl. `bolt1 261 + 182.7 = 443.7`

```
    ofstream f0;
    f0.open("out.txt");

    for(int i = 0; i < 3; i++) {
        f0 << bolt[i].nev << "\t" << ar[i] << " + " << ar[i] * 0.7 << " = " << ar[i] * 1.
    }
```

`in.txt:`

```
bolt1 100 20 70 30
bolt2 125 22 65 40
bolt3 140 20 58 30
bolt4 135 19 88 42
```

`out.txt:`

```
bolt1        261 + 182.7 = 443.7
bolt2        309 + 216.3 = 525.3
bolt3        289 + 202.3 = 491.3
```

`console:`

```
Nev     Tej     Tojas   Cukor   Puding
----------------------------------------
bolt1   100     20      70      30
bolt2   125     22      65      40
bolt3   140     20      58      30
```

6. Pótóra (2024. Április 16.)

A vasarlok.txt adatit olvasd be soronként:

- neve
- mennyit koltott
- melyik evben szuletett

Hozz létre struktúrát az adatoknak!

```
typedef struct {
    string nev;
    int penz;
    int szulev;
} Vasarlo;
```

Hozz létre tömböt, ami struktúrát tartalmaz (max 20 elem számára)

```
Vasarlo vasarlok[20];
```

Olvasd be a file sorait getline-nal

```
while(!f.eof()) {
    string line;
    getline(f, line);
}
```

Bontsd fel részeire a sort és tárold egy struktúrában

```
int vege = line.find(" ");
int ejele = vege + 1;
uj.nev = line.substr()
```

Az adatokat írd be a tömbbe

Számítsd ki minden vasárlóról, hogy hány éves az idén.

Keresd ki hogy ki a legfiatalab, írasd ki a nevét.

Kik azok, akik 2000-nél többet költöttek?

Mennyi az átlag vásárlási költség?

Kik költöttek kevesebbet az átlagnál?

fi = 0;
fiatal = a[0].hev
for(int i = 0; i < n; i++){
if(fiatal < a[i].hev) {
fiatal = a[i].hev;
fi=i
}
}

a[fi].nev

7. Pótóra (2024. Május 07.)

Tömb rendezése

1. Készíts egy 10 elemű tömböt.
2. Töltsd fel véletlen számokkal 5-100 közt.
3. Írd ki az elemeit a képernyőre, üreshelyekkel elválasztva!
4. Készíts egy másolatot a tömbről!
5. Rendezd az eredeti tömböt növekvő sorrendben, majd írasd ki.
6. Rendezd a másolt tömböt csökkenő sorrendben, majd írasd ki.

Struktúra rendezése

1. Hozz létre egy "Személy" struktúrát, amely eltárolja egy személy adatait:
   a. Vezetéknév
   b. Utónév
   c. JMBG
   d. Magasság
2. Olvasd be fileból 10 személy adatait:

```
Kovács János 1506990710011 175
Nagy Eszter 2008984405017 163
Tóth Péter 0603005610026 178
Szabó Anna 3011999925021 170
Horváth Gábor 2507950620135 185
Kovács Kinga 1404993105004 168
Molnár Balázs 1806997800012 180
Varga Zsuzsanna 1001008915010 165
Farkas Ádám 0303978900023 177
Papp Katalin 2205995305005 172
```

3. Tárold el az adatokat egy Személyekből álló tömbbe.
4. Másold le a tömböt, és az új tömböt rendezd magasság szerint.
5. Másold le a tömböt, és az út tömböt rendezd születési dátum szerint.
6. Másold le a tömböt, és az új tömböt rendezd név szerint.
7. Írd ki az eredeti tömböt, majd minden tömből a neveket és a rendezett adatot.