1. 🔍 Uždavinio algoritmo idėja

1. **Pradiniai duomenys**:
   n – viso knygos puslapių skaičius. **a** – kiek puslapių perskaitė pirmą dieną. **b** – kiekvieną kitą dieną perskaito **b puslapių daugiau** nei prieš tai buvusią dieną.

2. **Sprendimo eiga**:

   - Sukuriame kintamąjį `dienos`, kuris skaičiuos dienų skaičių. Pradinė reikšmė 1
   - Sukuriame kintamąjį `viso`, kuris kaups perskaitytų puslapių sumą. Pradinė reikšmė a
   - Sukuriame kintamąjį `dabar`, kuris skaičiuos kiekvieną konkrečią dieną skaitomų puslapių skaičių. Pradinė reikšmė a
   - Kol `viso` yra mažiau nei `n`, kartojame:
     - Padidiname `dienos` skaičių. *(2-oji; 3-oji;....)*
     - Kiekvieną dieną skaitomų puslapių skaičių didiname `b`.
     - Pridedame prie `viso` tą dieną perskaitytų puslapių skaičių.
   - Kai `viso >= n`, sustojame ir grąžiname `dienos`.

   į Funkciją patogu iškelti skaičiavimo ciklą

Programos pvz, kita idėja: skaičiuojama kiek puslapių liko po kiekvienos dienos:

```cpp
//uždavinio Knygos skaitymas, funkcija
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int skaiciuoja (int n, int a, int b);

int n, a, b, per, dien, liko;

int main()
{
    int dienos;
    ifstream fd("Duomenys.txt");
    fd >>n >>a >>b;
    fd.close();

    dienos = skaiciuoja (n, a, b);

    ofstream fr("Rezultatai.txt");
    fr <<dienos;
    fr.close();

}
//........
int skaiciuoja (int n, int a, int b)
{
    per = a;
    dien = 1;
    liko = n - a;
  while(liko > 0)
        {
            per = per + b;
            liko = liko - per;
            dien ++;
        }
 return dien;

}
```

☐ 2. Uždavinio algoritmo idėja:

1. **Nuskaityti duomenis iš failo**:
   - Plytelės matmenis centimetrais: ilgį ir plotį.
   - Grindų matmenis metrais: ilgį ir plotį.
2. **Konvertuoti grindų matmenis į centimetrus**, nes plytelės pateiktos centimetrais.

3. **Apskaičiuoti, kiek plytelių reikia ilgio kryptimi**:
   - Padalinti grindų ilgį iš plytelės ilgio.
   - Jei lieka likutis (grindų ilgis nesidalija tiksliai), pridėti dar vieną plytelę.
4. **Apskaičiuoti, kiek plytelių reikia pločio kryptimi**:
   - Padalinti grindų plotį iš plytelės pločio.
   - Jei lieka likutis, pridėti dar vieną plytelę.
5. **Padauginti plytelių kiekį ilgio ir pločio kryptimis**, kad gautume bendrą reikalingų plytelių skaičių.

6. **Rezultatą įrašyti į rezultatų failą**.

Funkciją: plytelių kiekio skaičiavimui (3,4 dalys: du kreipiniai, viena funkcija)

Smalsiems: apvalinimo funkcijos C++

- ☐ `round(x)` – „įprastas“ matematinis apvalinimas Pvz.: round(2,7) = 3
- ☐ `trunc(x)` – pašalina skaičiaus dalį po kablelio Pvz.: trunc(2,7) = 2
- ☐ `floor(x)` – apvalina žemyn skaičių ašyje: **randa didžiausią sveikąjį skaičių, kuris yra *mažesnis arba lygus* x** Pvz.: floor(2,3) = 2 floor(2,7) = 2
- ☐ `ceil(x)` – apvalina aukštyn skaičių ašyje: **randa mažiausią sveikąjį skaičių, kuris yra *didesnis arba lygus* x** Pvz.: ceil(2,3) = 3 ceil(2,7) = 3

3. Uždavinio algoritmo idėja (naudojant Herono formulę)

1. **Nuskaityti** iš failo `Duomenys.txt` tris taškus: A(x_1, y_1), B(x_2, y_2), C(x_3, y_3).
2. **Apskaičiuoti atstumus tarp taškų** (trikampio kraštines):
   - AB = atstumas tarp A ir B
   - BC = atstumas tarp B ir C
   - CA = atstumas tarp C ir A

Naudoti formulę atstumui tarp dviejų taškų apskaičiuoti: $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Atstumo skaičiavimui patogu naudoti funkciją (3 kreipiniai)

3. **Apskaičiuoti pusperimetrį**: $$s = \frac{AB + BC + CA}{2}$$

4. **Apskaičiuoti plotą pagal Herono formulę**: $$S = \sqrt{s(s-AB)(s-BC)(s-CA)}$$
5. **Suapvalinti** rezultatą iki **dviejų skaitmenų po kablelio**.
6. **Įrašyti** rezultatą į `Rezultatai.txt`.

4. 🧠 Algoritmo idėja

🎯 Tikslas:

Rasti kiekvieno mokinio **geriausią įvertinimą** iš trijų kontrolinių važiavimų ir įrašyti jį į rezultatų failą.

1. **Atidaryti duomenų failą** `Duomenys1.txt`, iš kurio bus skaitomi mokinių duomenys.

2. **Perskaityti pirmą eilutę** – joje yra skaičius **n**, kuris reiškia, kiek mokinių yra.

3. **Kartoti veiksmus n kartų** (tiek, kiek yra mokinių):

    - Perskaityti **tris sveikus skaičius** – tai vieno mokinio trys kontrolinių važiavimų įvertinimai.
    - Naudoti **funkciją**, kuri iš šių trijų skaičių **suranda didžiausią** (geriausią įvertinimą).
    - Įrašyti tą geriausią įvertinimą į rezultatų failą `Rezultatai1.txt`.
4. **Uždaryti abu failus** – duomenų ir rezultatų.

Funkcijos veikimo principas:

- Funkcija gauna **tris skaičius** kaip argumentus.
- Ji palygina juos tarpusavyje ir **grąžina didžiausią**.

5. 🧠 Algoritmo idėja

1. Perskaityti duomenis iš failo `Duomenys.txt`:
    - Kambario ilgį `x`, plotį `y`, aukštį `a` (metrais).
    - Dažų išeigą `l` – kiek kilogramų dažų reikia 1 kvadratiniam metrui sienos.
    - Vienos dažų dėžutės kainą `k` litais.
2. Apskaičiuoti bendrą sienų plotą:
    - Kambarys turi 4 sienas: 2 sienos yra `x × a`, o kitos 2 – `y × a`.
    - Bendras plotas = `2 * x * a + 2 * y * a`.
3. Apskaičiuoti, kiek kilogramų dažų reikia:
   *Reikalingi dažai = bendras sienų plotas * dažų išeiga `l`.
4. Apskaičiuoti, kiek dėžučių reikia:
    - Vienoje dėžutėje yra 10 kg dažų.
    - Dėžučių kiekis = reikalingi dažai / 10.

- Jei lieka likutis, reikia papildomos dėžutės (naudojame sveiką dalybą su apvalinimu į viršų: *if arba ceil*).

Funkcija `ceil` C++ kalboje yra **matematinė funkcija**, kuri **apvalina skaičių į viršų** iki artimiausio sveikojo skaičiaus.

pvz. `ceil(2.1)` = `3`. `ceil(5.9)` = `6` `ceil(4.0) = 4`

5. Apskaičiuoti bendrą kainą eurais:
    - Kaina litais = dėžučių kiekis * k.
    - Kaina eurais = kaina litais / 3.4528.
    - Rezultatą apvalinti iki dviejų skaičių po kablelio.
6. Įrašyti rezultatus į failą `Rezultatai.txt`:
    - Pirmoje eilutėje – dėžučių kiekis.
    - Antroje eilutėje – bendra kaina eurais.

6. 🧠 Šį uždavinį patogu spręsti dviem pagrindinėmis dalimis:

1. Apskaičiuoti bendrą braškių kiekį.
    - Kiekvieną dieną sudedame visų keturių vaikų priskintą kiekį: Vaida + Gytis + Jonas + Rasa.
    - Šį veiksmą atliekame kiekvienai iš `n` dienų.
    - Gautas dienos kiekių sumas kaupiame bendrame kintamajame.
2. Nustatyti, kas kiekvieną dieną priskynė daugiausia.
    - Kiekvieną dieną randame didžiausią iš keturių skaičių.
    - Palyginame kiekvieno vaiko kiekį su didžiausiuoju.
    - Jei keli vaikai turi tokį pat didžiausią kiekį, išvedame visų jų vardus.

Mąstymo schema:
Pradžia → perskaitome `n` → kartojame `n` kartų:`a, b, c, d` ↓
apskaičiuojame `a+b+c+d` → pridedame prie bendro kiekio ↓
randame didžiausią iš `a, b, c, d` ↓
patikriname, kieno kiekis lygus didžiausiam ↓
išvedame vardą / vardus ↓
pabaigoje išvedame bendrą visų vaikų kiekį.

Svarbiausia čia suprasti, kad dvi užduoties dalys sprendžiamos vienu ciklu: kiekvienos dienos duomenis perskaitome, iš karto suskaičiuojame tos dienos bendrą kiekį ir nustatome daugiausiai priskynusį vaiką.

Reikalingos konstrukcijos

1. **Failo atidarymas ir duomenų skaitymas**
    - `ifstream` – skaityti iš `Duomenys5.txt`

    - `ofstream` – rašyti į `Rezultatai5.txt`
2. **`for` ciklas**
    - Kadangi žinome `n` dienų skaičių, patogiausia naudoti `for`.
    - Kiekvieno ciklo metu nuskaitomi 4 vaikų rezultatai.
3. **Kintamųjų naudojimas**
    - `n` – dienų skaičius
    - `a, b, c, d` – vaikų per dieną priskintų braškių kiekiai
    - `suma` – bendras visų vaikų priskintų braškių kiekis.
4. **`if` sąlygos**
    - Nustatyti, kuris vaikas priskynė daugiausia.
    - Svarbu: negalima naudoti vien tik `else if`, nes reikia išvesti **kelis vardus**, jei kiekiai vienodi.

Kur galima panaudoti paprogramę?

Čia labai natūraliai galima sukurti funkciją, kuri **randa didžiausią iš keturių skaičių**.