**შეხვედრა 3: პირველი საუბარი არდუინოსთან C++ ენაზე**

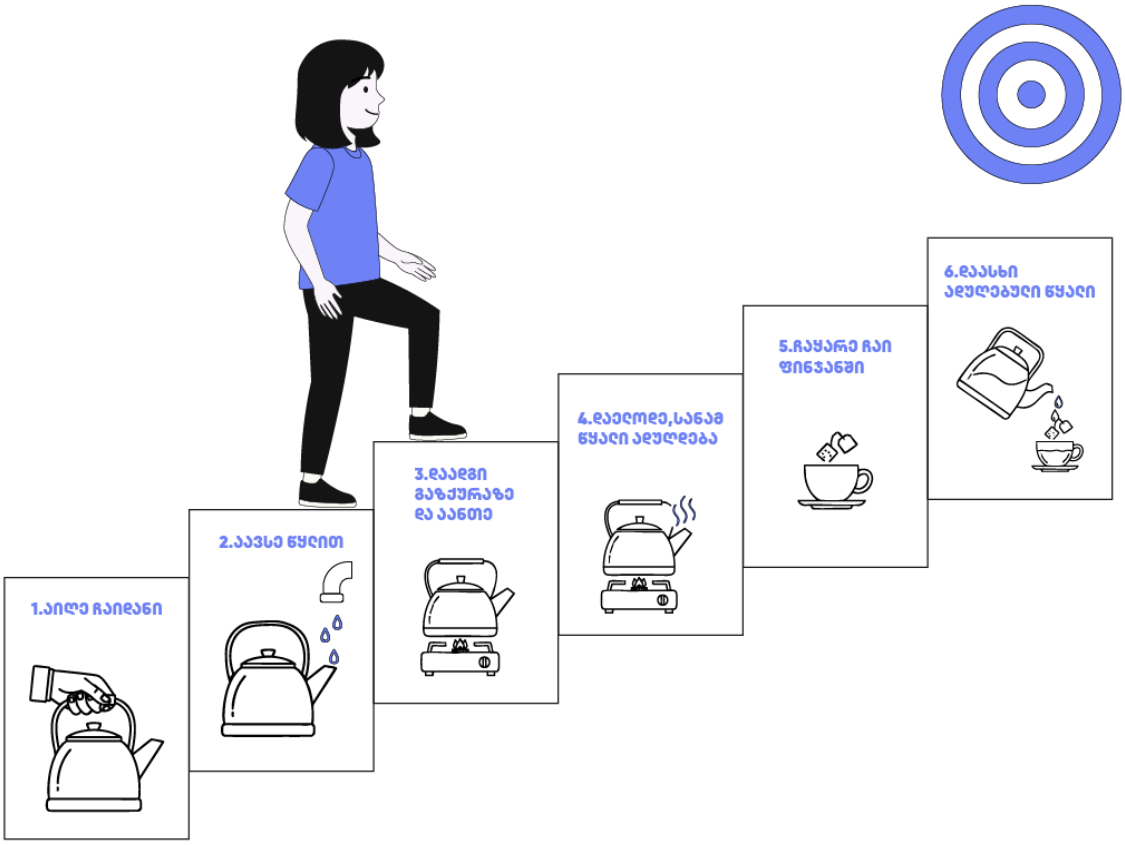
გამარჯობა! წინა შეხვედრაზე ჩვენ გავეცანით არდუინოს, ჩვენი მომავალი პროექტების „ტვინს“. დღეს კი ვისწავლით, როგორ ვესაუბროთ ამ ტვინს, როგორ მივცეთ მას ბრძანებები და ვასწავლოთ სხვადასხვა მოქმედების შესრულება. ამ საუბრის ენა იქნება C++, ერთ-ერთი ყველაზე პოპულარული პროგრამირების ენა მსოფლიოში.

## 1. პროგრამირების საფუძვლები

### 1.1. რა არის პროგრამირება, როგორც ინსტრუქციების ერთობლიობა

წარმოიდგინე, რომ შენს მეგობარს უხსნი თუ როგორ მოამზადოს ჩაი. შენ მას ეუბნები მოქმედებების ზუსტ თანმიმდევრობას:

1. აიღე ჩაიდანი;
2. აავსე წყლით;
3. დადგი გაზქურაზე;
4. აანთე გაზქურა;
5. დაელოდე, ვიდრე წყალი ადუღდება;
6. ჩაყარე ჩაი ფინჯანში;
7. დაასხი ადუღებული წყალი.




პროგრამირება ძალიან ჰგავს ამ პროცესს. ეს არის კომპიუტერისთვის (ჩვენს შემთხვევაში, არდუინოსთვის) გასაგებ ენაზე დაწერილი ინსტრუქციების (ბრძანებების) ზუსტი თანმიმდევრობა. კომპიუტერი ჭკვიანია, მაგრამ მას არ შეუძლია დამოუკიდებლად ფიქრი. ის ზუსტად იმას აკეთებს, რასაც ჩვენ ვეუბნებით. ამიტომ ჩვენი ინსტრუქციები ძალიან მკაფიო და ლოგიკური უნდა იყოს.

### 1.2. რა არის ალგორითმი, როგორც პრობლემის გადაჭრის გეგმა

სანამ კოდის წერას დავიწყებთ, ჯერ პრობლემის გადაჭრის გეგმა უნდა შევიმუშაოთ. სწორედ ამ გეგმას ეწოდება **ალგორითმი**. ჩაის მომზადების ჩვენი ინსტრუქცია, ფაქტობრივად, ალგორითმი იყო. ალგორითმი არის ნაბიჯ-ნაბიჯ გეგმა, რომელიც პრობლემის დასაწყისიდან საბოლოო შედეგამდე მიგვიყვანს.

მაგალითად, თუ ჩვენი ამოცანაა **შუქდიოდის** აციმციმება, ჩვენი ალგორითმი ასეთი იქნება:

1. ჩართე ნათურა;
2. დაიცადე ერთი წამი;
3. გამორთე ნათურა;
4. დაიცადე ერთი წამი;
5. გაიმეორე ყველაფერი თავიდან.

ჩვენ საჭიროებებზე მორგებული ალგორითმის შედგენა პროგრამირების ყველაზე მნიშვნელოვანი ნაწილია. როცა გეგმა გვაქვს, მისი კოდად ქცევა უკვე ბევრად მარტივია.

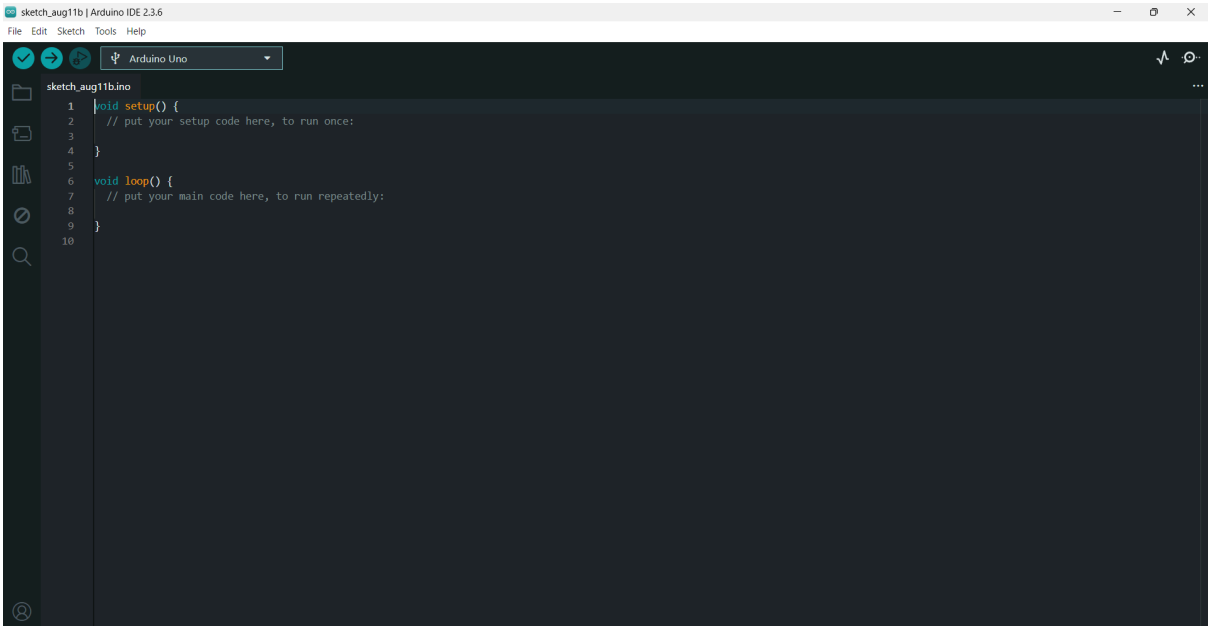
### 1.3. Arduino IDE-ის (პროგრამირების გარემოს) და სკეტჩის (პროგრამის) კონცეფცია

**IDE (Integrated Development Environment):** ეს არის ჩვენი სამუშაო სივრცე, პროგრამა, სადაც ჩვენ ვწერთ კოდს, ვამოწმებთ შეცდომებს და ვტვირთავთ არდუინოს დაფაში. AnITa-ს პლატფორმაში ჩაშენებული კოდის რედაქტორი, სწორედ, ასეთი გარემოს როლს ასრულებს. ის გვეხმარება, გავხადოთ კოდის წერის პროცესი უფრო მარტივი და ორგანიზებული.

Arduino IDE

**სკეტჩი (Sketch):** არდუინოს სამყაროში ჩვენ მიერ დაწერილ პროგრამას, ანუ კოდს, „სკეტჩი“ ეწოდება. ეს სახელი იმიტომ შეურჩიეს, რომ ის ხშირად არის სწრაფი იდეის,

პროტოტიპის ჩანახატი. თითოეული ჩვენი პროექტი ახალი სკეტჩით დაიწყება.




## 2. C++-ის ძირითადი ელემენტები

მოდი, გავიცნოთ ჩვენი ახალი ენის, C++-ის, ძირითადი სამშენებლო ბლოკები.

### 2.1. მონაცემთა ტიპები: int, float, boolean, char, String

არდუინომ რომ იმუშაოს, მას ინფორმაცია სჭირდება. ეს ინფორმაცია შეიძლება იყოს სხვადასხვა სახის (მონაცემთა ტიპის). წარმოიდგინე, რომ გაქვს სხვადასხვა ზომის და ფორმის ყუთები სხვადასხვა ნივთისთვის.

- int (Integer) – **მთელი რიცხვების ყუთი:** ამ ტიპის ცვლადში ვინახავთ მთელ რიცხვებს, მაგალითად: -10, 0, 5, 150.
- float – **წილადი რიცხვების ყუთი:** აქ ვინახავთ ათწილადებს, მაგალითად: 3.14, -0.5, 99.9.
- boolean (ლოგიკური ტიპი) – **„კი/არა“ყუთი:** ამ ტიპის ცვლადს მხოლოდ ორი მნიშვნელობა შეიძლება ჰქონდეს: true (ჭეშმარიტი) ან false (მცდარი). მას ვიყენებთ ისეთი მდგომარეობების აღსაწერად, როგორიცაა „ნათურა ჩართულია?“ (true) თუ „ღილაკს დააჭირეს?“ (false).
- char (Character) – **ერთი სიმბოლოს ყუთი:** აქ ვინახავთ მხოლოდ ერთ სიმბოლოს, მაგალითად: 'A', 'b', '7', '!'.
- String – **ტექსტის ყუთი:** ამ ტიპის ცვლადში ვინახავთ ტექსტს, ანუ სიმბოლოების ერთობლიობას, მაგალითად: "გამარჯობა, ნინო!".

### 2.2. ცვლადების დეკლარაცია (სახელის მინიჭება) და ინიციალიზაცია (მნიშვნელობის მინიჭება)

დეკლარაცია: ვიღრე ცვლადს გამოვიყენებთ, ის უნდა შევქმნათ, ანუ მოვახდინოთ მისი დეკლარირება. ამ დროს ჩვენ ვუთითებთ მის ტიპს და ვაძლევთ სახელს.
int age; (შევქმენით მთელი რიცხვის ტიპის ცვლადი სახელად age).

ინიციალიზაცია: ეს არის ცვლადისთვის საწყისი მნიშვნელობის მინიჭება.
age = 14; (ცვლად age-ს მივანიჭეთ მნიშვნელობა 14).

ამ ორი მოქმედების გაერთიანებაც შეგვიძლია:
String name = "ნინო"; (შევქმენით ტექსტური ტიპის ცვლადი name და მაშინვე მივანიჭეთ მნიშვნელობა "ნინო").

```
void setup() {
  Serial.begin(9600);
      int age=14;
      string name ="nino";
      boolean isAdult=false;

}

void loop() {
}
```

მოცემული კოდის შემდეგ კომპიუტერის მეხსიერებაში გამოიყოფა 3 “ყუთი”,სადაც შენახული იქნება ნინოს მონაცემები:


ცვლადის მნიშვნელობები, როგორც პატარა “ყუთები”, უკვე წარმოვიდგინეთ ახლა რა მოხდება თუ age ცვლადს ჯერ 14-ს მივანიჭებთ, ხოლო შემდეგ 15-ს?

```
void setup() {
  Serial.begin(9600);
       int age=14;// age ს ყუთში ჩაწერილია 14
       age=15;//age ს ყუთიდან წაიშალა 14 და ჩაიწერა 15

}
void loop() {
}
```

როცა ერთ ცვლადს თავიდან მნიშვნელობას ვანიჭებთ და მერე ისევ ახალ მნიშვნელობას მივანიჭებთ, ისინი **არ ემატება ერთმანეთს**. უბრალოდ ძველი მნიშვნელობა იშლება და **ბოლოს მინიჭებული** რიცხვი რჩება. საბოლოოდ, age-ის მნიშვნელობა იქნება 15.

### 2.3. არითმეტიკული, შედარების და ლოგიკური ოპერატორები

**არითმეტიკული ოპერატორები:** ეს არის მათემატიკური მოქმედებები, რომლებსაც რიცხვებზე ვასრულებთ.

- + (მიმატება), - (გამოკლება), * (გამრავლება), / (გაყოფა).
- მაგალითი: int result = 10 + 5; (result ცვლადის მნიშვნელობა იქნება 15).

**შედარების ოპერატორები:** მათ ვიყენებთ ორი მნიშვნელობის შესადარებლად. შედეგი ყოველთვის არის boolean ტიპის (true ან false).

- == (უდრის), != (არ უდრის), < (ნაკლებია), > (მეტია), <= (ნაკლებია ან უდრის), >= (მეტია ან უდრის).
- მაგალითი: boolean isAdult = (age >= 18); (თუ age ცვლადის მნიშვნელობა 18-ზე მეტია ან ტოლია, isAdult იქნება true, სხვა შემთხვევაში – false).

**ლოგიკური ოპერატორები:** მათ ვიყენებთ რამდენიმე ლოგიკური პირობის გასაერთიანებლად.

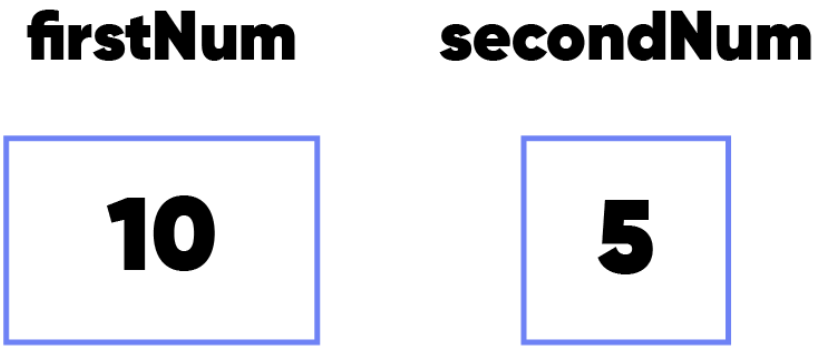
- && (AND - „და“): შედეგი true არის მხოლოდ მაშინ, თუ ორივე პირობა true-ა.
- || (OR - „ან“): შედეგი true არის, თუ ერთ-ერთი პირობა მაინც true-ა.
- ! (NOT - „არა“): ცვლის boolean მნიშვნელობას საპირისპიროთი (true-ს ხდის false-დ და პირიქით).

  ლოგიკური ოპერატორების უკეთ გასააზრებლად გადავხედოთ ქვემოთ მოცემულ კოდს:

```
void setup() {
  Serial.begin(9600);
      int firstNum=10;
      int secondNum =5;
      int result=firstNumber+secondNumber;
boolean isSumEqual= (firstNum+secondNum!=secondNum+firstNum);
}

void loop() {
}
```

მოცემული კოდი გვითვლის ორი რიცხვის ჯამს. თავდაპირველად მეხსიერებაში გამოიყოფა 2 integer ტიპის ყუთი, რომელთაც თავიანთი სახელი და მნიშვნელობა გააჩნია:




შემდეგ იქმნება კიდევ ერთი integer ტიპის ყუთი სახელად `result`, რომელიც ინახავს ამ ორის ჯამს (`firstNum + secondNum`). შედეგად `result` = 15.


გარდა ამ სამი ცვლადისა, კოდში კიდევ არის ერთი ცვლადი შექმნილი. თქვენი პირველი სავარჯიშო კი სწორედ ამ ცვლადზე და მის მნიშვნელობაზეა.

**სავარჯიშო 1:**

# გაიხსენე boolean ტიპის ცვლადები და მისი შესაძლო მნიშვნელობები, ასევე ყურადღება მიაქციე ტოლობაში გამოყენებულ ლოგიკურ ოპერატორებს და გვითხარი isSumEquals ცვლადის მნიშვნელობა

პასუხი:

boolean isSumEqual= (firstNum+secondNum!=secondNum+firstNum) ამ ბრძანებაში იქმნება ცვლადი `isSumEqual`. მას უნდა ეწეროს, მართალია თუ არა, რომ `firstNum + secondNum` არ უდრის `secondNum + firstNum.`

რადგან ორი რიცხვის შეკრება ყოველთვის ერთნაირ შედეგს იძლევა (მაგ: 3 + 5 = 8 და 5 + 3 = 8), ეს შედარება გამოდის **არასწორი**. ამიტომ ცვლადს მიენიჭება მნიშვნელობა **`false`** (ე.ი. „მცდარი“).

### 3. არდუინოს სკეტჩის სტრუქტურა

ყველა არდუინოს სკეტჩი (პროგრამა) შედგება ორი მთავარი ფუნქციისგან: `setup()` და `loop()`.

### 3.1. `setup()` ფუნქცია: კოდი, რომელიც ერთხელ სრულდება

```
void setup() { ... }
```

წარმოიდგინე, რომ სპექტაკლისთვის ემზადები. სანამ წარმოდგენა დაიწყება, შენ უნდა მოამზადო სცენა, დაალაგო რეკვიზიტები, ჩართო განათება. `setup()` ფუნქცია სწორედ ამას აკეთებს.

კოდი, რომელიც `setup()` ფუნქციის ფიგურულ ფრჩხილებში `{ }` იწერება, სრულდება მხოლოდ ერთხელ, როცა არდუინოს ჩავრთავთ ან Reset ღილაკს დავაჭერთ. აქ ჩვენ ვწერთ მოსამზადებელ ბრძანებებს, მაგალითად, ვუთითებთ, რომელი პინი რისთვის უნდა გამოვიყენოთ, ან ვრთავთ სერიულ მონიტორს (`Serial.begin(9600);`).

### 3.2. `loop()` ფუნქცია: კოდი, რომელიც უსასრულოდ მეორდება

```
void loop() { ... }
```

როცა სცენა მზად არის, სპექტაკლი იწყება. მსახიობები ასრულებენ თავიანთ როლებს, მოქმედება ვითარდება. `loop()` ფუნქცია არის ჩვენი სპექტაკლი.

კოდი, რომელიც `loop()` ფუნქციის ფიგურულ ფრჩხილებში `{ }` იწერება, სრულდება განუწყვეტლივ, უსასრულოდ. როგორც კი `loop`-ის ბოლო ბრძანება შესრულდება, არდუინო ისევ თავიდან იწყებს `loop`-ის პირველი ბრძანების შესრულებას. ეს პროცესი გრძელდება მანამ, სანამ არდუინოს კვება **მიეწოდება**. სწორედ აქ ვწერთ ჩვენი პროექტის ძირითად ლოგიკას, მაგალითად, სენსორის მონაცემების მუდმივად წაკითხვას ან ნათურის ციმციმს.

### 3.3. კომენტარების (`//` და `/* */`) გამოყენება

კომენტარი არის ტექსტი კოდში, რომელსაც არდუინო სრულად აიგნორებს. ის მხოლოდ ადამიანებისთვისაა განკუთვნილი. კომენტარები გვეხმარება, ავხსნათ, რას აკეთებს კოდის ესა თუ ის ნაწილი. ეს ძალიან მნიშვნელოვანია, რადგან რამდენიმე კვირის შემდეგ შეიძლება ჩვენ თვითონაც დაგვავიწყდეს **ის**, **თუ** რატომ დავწერეთ კოდი ასე და არა სხვაგვარად.

ასევე, მნიშვნელოვანია **შევარჩიოთ** ცვლადების სახელები სწორად. თუ ცვლადს დავარქმევთ ისეთ სახელს, რომელიც აღწერს მის რეალურ დანიშნულებას (მაგალითად: astronautas მიგვანიშნებს ასტროვნატების რაოდენობას), მოგვიანებით კოდის გაგება გაცილებით ადვილი იქნება. ეს კიდევ ერთგვარი "კომენტარის" როლს ასრულებს, რადგან ცვლადის სახელი თავად გვახსენებს **იმას თუ** რისთვის ვიყენებთ მას.

ერთხაზიანი კომენტარი: იწყება ორი ხაზით //. ყველაფერი, რაც ამ სიმბოლოების შემდეგ იმავე ხაზზე წერია, კომენტარია.
// ეს ბრძანება რთავს **შუქდიოდს.**

**მრავალხაზიანი კომენტარი:** იწყება /*-ით და მთავრდება */-ით. ყველაფერი, რაც ამ სიმბოლოებს შორისაა მოქცეული, კომენტარია, თუნდაც რამდენიმე ხაზზე იყოს გადანაწილებული.

```
/*
  ეს არის ჩემი პირველი პროექტი.
  მისი მიზანია, გამოთვალოს ორი რიცხვის ჯამი.
*/
```

### დავალება 3: „კოსმოსური მისიის კალკულატორი“

წარმოიდგინე, რომ კოსმოსური მისიის ინჟინერი ხარ. შენი ამოცანაა, დაწერო პროგრამა, რომელიც გამოითვლის ასტრონავტების საკვების მარაგს. შექმენი ორი int ტიპის ცვლადი: astronauts (ასტრონავტების რაოდენობა) და days (მისიის ხანგრძლივობა დღეებში). შემდეგ გამოთვალე მათი ჯამი, სხვაობა, ნამრავლი,განაყოფი და დაბეჭდე შედეგები სერიულ მონიტორზე შესაბამისი ტექსტური აღწერებით.

### დავალება 3.1: შეცდომის პოვნა და გასწორება

პროგრამისტს კოდის წერისას შეცდომა მოუვიდა. ერთ-ერთი ბრძანების ბოლოს მას გამორჩა აუცილებელი სიმბოლო, რის გამოც პროგრამა არ მუშაობს. შენი ამოცანაა, იყო კოდის დეტექტივი, იპოვო და გაასწორო შეცდომა, რათა პროგრამამ სწორად იმუშაოს.

**მინიშნება:** C++ ენაში ბრძანებების უმეტესობა წერტილ-მძიმით (;) უნდა მთავრდებოდეს.

```
void setup() {
  Serial.begin(9600);

  // ცვლადების შექმნა
  int astronauts = 4
  int days = 30;
```

```
  // გამოთვლები
  int sum = astronauts + days;
  int difference = days - astronauts;
  int product = astronauts * days;
  float division = (float)days / astronauts;

  // შედეგების დაბეჭდვა
  Serial.print("number of astronauts:");
  Serial.println(astronauts);
  Serial.print("duration of the mission");
  Serial.println(days);
  Serial.println("---calculations---");
  Serial.print("Sum: ");
  Serial.println(sum);
  Serial.print("difference:");
  Serial.println(difference);
  Serial.print("product: ");
  Serial.println(product);
  Serial.print("division (days on an astronaut");
  Serial.println(division);
}

void loop() {
}
```

### დავალება 1.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, ცვლადები შექმნილია და საწყისი მონაცემები იბეჭდება. თუმცა, მთავარი ნაწილი – გამოთვლები და მათი შედეგების დაბეჭდვა – გამორჩენილია. შენი ჯერია! დაამატე გამოტოვებული კოდის ნაწილები, რათა პროგრამა დასრულდეს.

```
void setup() {
  Serial.begin(9600);

  // ცვლადების შექმნა და მნიშვნელობების მინიჭება
  int astronauts = 4;
  int days = 30;

  // საწყისი მონაცემების დაბეჭდვა
  Serial.print("number of astronauts: ");
  Serial.println(astronauts);
  Serial.print("duration of the mission ");
```

```
  Serial.println(days);
  Serial.println("--- calculations---");

  // --- ჩაამატე კოდი აქ ---
  // 1. შექმენი ცვლადები ჯამის, სხვაობის, ნამრავლისა და განაყოფისთვის.
  // 2. გამოთვალე შედეგები.
  // 3. დაბეჭდე თითოეული შედეგი შესაბამისი ტექსტით.
  //    (მაგ: Serial.print("sum: "); Serial.println(sum);)

}

void loop() {
}
```

**დავალება 1.3: შეიმუშავე პროგრამული კოდი**
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შეასრულებს შემდეგ სამუშაოს:

1. მოამზადებს სერიულ მონიტორს სამუშაოდ;
2. შექმნის ორ int ტიპის ცვლადს (astronauts და days) და მიანიჭებს მათ მნიშვნელობებს (4 და 30);
3. გამოითვლის ამ ცვლადების ჯამს, სხვაობას, ნამრავლსა და განაყოფს;
4. დაბეჭდავს ყველა შედეგს სერიულ მონიტორზე შესაბამისი ტექსტური აღწერებით.

```
void setup() {
  // დაწერე მთლიანი კოდი აქ.

}

void loop() {
  // ეს ფუნქცია ცარიელი რჩება.
}
```

**სწორი პასუხი (პროგრამული კოდი სრულად):**

```
void setup() {
  // სერიული პორტის ჩართვა
  Serial.begin(9600);

  // ცვლადების შექმნა და მნიშვნელობების მინიჭება
```

```
  int astronauts = 4;
  int days = 30;

  // გამოთვლების შესრულება
  int sum = astronauts + days;
  int difference = days - astronauts;
  int product = astronauts * days;
  float division = (float)days / astronauts; // ყურადღება მიაქციე (float)-ს, ეს საჭიროა წილადი პასუხის მისაღებად

  // შედეგების დაბეჭდვა
  Serial.print("number of astronauts: ");
  Serial.println(astronauts);

  Serial.print("duration of the mission ");
  Serial.println(days);

  Serial.println("--- calculations ---");

  Serial.print("Sum: ");
  Serial.println(sum);

  Serial.print("difference: ");
  Serial.println(difference);

  Serial.print("product: ");
  Serial.println(product);

  Serial.print("division (days on an astronaut ");
  Serial.println(division);
}

void loop() {
  // ეს ფუნქცია ცარიელი რჩება.
}
```

JSON

```
{
  "version": 1,
```

```
  "board": "arduino:avr:uno",
  "steps": [
    {
      "type": "compile"
    },
    {
      "type": "static",
      "rules": [
        {
          "id": "serial_begin",
          "name": "Serial initialization",
          "kind": "require_regex",
          "pattern":
"Serial\\.begin\\s*\\(\\s*9600\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "სერიული პორტი უნდა ჩაირთოს:
Serial.begin(9600);",
          "msg_pass": "სერიული პორტი სწორად არის ჩართული."
        },
        {
          "id": "var_astronauts",
          "name": "Astronauts variable",
          "kind": "require_regex",
          "pattern": "int\\s+astronauts\\s*=\\s*\\d+\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "შექმენი int ტიპის ცვლადი astronauts: int
astronauts = 4;",
          "msg_pass": "astronauts ცვლადი სწორადაა შექმნილი."
        },
        {
          "id": "var_days",
          "name": "Days variable",
          "kind": "require_regex",
          "pattern": "int\\s+days\\s*=\\s*\\d+\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
```

```
          "msg_fail": "შექმენი int ტიპის ცვლადი days: int days =
30;",
          "msg_pass": "days ცვლადი სწორადაა შექმნილი."
        },
        {
          "id": "calc_sum",
          "name": "Sum calculation",
          "kind": "require_regex",
          "pattern":
"int\\s+sum\\s*=\\s*astronauts\\s*\\+\\s*days",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოთვალე ჯამი: int sum = astronauts +
days;",
          "msg_pass": "ჯამი სწორად არის გამოთვლილი."
        },
        {
          "id": "calc_difference",
          "name": "Difference calculation",
          "kind": "require_regex",
          "pattern":
"int\\s+difference\\s*=\\s*days\\s*-\\s*astronauts",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოთვალე სხვაობა: int difference = days -
astronauts;",
          "msg_pass": "სხვაობა სწორად არის გამოთვლილი."
        },
        {
          "id": "calc_product",
          "name": "Product calculation",
          "kind": "require_regex",
          "pattern":
"int\\s+product\\s*=\\s*astronauts\\s*\\*\\s*days",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოთვალე ნამრავლი: int product =
astronauts * days;",
```

```
          "msg_pass": "ნამრავლი სწორად არის გამოთვლილი."
        },
        {
          "id": "calc_division",
          "name": "Division calculation with float",
          "kind": "require_regex",
          "pattern":
"float\\s+division\\s*=\\s*\\(\\s*float\\s*\\)\\s*days\\s*/\\s*as
tronauts",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოთვალე განაყოფი float-ად: float
division = (float)days / astronauts;",
          "msg_pass": "განაყოფი სწორად არის გამოთვლილი float-ად."
        },
        {
          "id": "print_results",
          "name": "Print results",
          "kind": "require_regex",
          "pattern": "Serial\\.print(?:ln)?\\s*\\(",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "დაბეჭდე შედეგები სერიულ მონიტორზე
Serial.print-ით.",
          "msg_pass": "შედეგები იბეჭდება სერიულ მონიტორზე."
        },
        {
          "id": "correct_sequence",
          "name": "Correct calculation sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "Serial\\.begin\\s*\\(\\s*9600\\s*\\)",
            "int\\s+astronauts\\s*=",
            "int\\s+days\\s*=",
            "int\\s+sum\\s*=\\s*astronauts\\s*\\+\\s*days",
            "int\\s+difference\\s*=\\s*days\\s*-\\s*astronauts",
            "int\\s+product\\s*=\\s*astronauts\\s*\\*\\s*days",
```

```
"float\\s+division\\s*=\\s*\\(\\s*float\\s*\\)\\s*days\\s*/\\s*as
tronauts"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "თანმიმდევრობა უნდა იყოს: Serial.begin →
ცვლადები (astronauts, days) → გამოთვლები (sum, difference,
product, division).",
          "msg_pass": "ოპერაციების თანმიმდევრობა სწორია."
        }
      ]
    }
  ]
}
```