

gogocode

სახელმძღვანელო შექმნილია 10-დღიანი პროგრამირების ბანაკისთვის. ალტერნატიულად, შეიძლება მოეწყოს 8 შეხვედრისგან შემდგარი მოკლე სალექციო კურსი, სადაც თითოეული შეხვედრა გრძელდება 4 საათს (ლანჩისა და გასართობი აქტივობების ჩათვლით).

თითოეული შეხვედრისთვის გათვალისწინებულია 1.5-საათიანი ლექცია და 1.5-საათიანი პრაქტიკული სამუშაო.

წინამდებარე სახელმძღვანელოში დეტალურადაა აღწერილი ლექციის მიმდინარეობა, თემატიკა და პრაქტიკული სავარჯიშოები.

სასწავლო პროგრამა დაიგეგმა და განახლდა თბილისში მოწყობილი პროგრამირების სკოლის ფარგლებში, რომელიც 15-მა მონაწილემ გაიარა.

პროექტი დაფინანსებულია საქართველოში ამერიკის საელჩოს მიერ, პროექტის GoGo Code ფარგლებში.

პროექტი განხორციელდა ჯამპსტარტ ჯორჯიას მიერ.



ლექცია 1

(ხანგრძლივობა 90 წთ)

- გაცნობა ----- 5 წთ
- შესავალი (კურსის მიზანი)----- 5 წთ
- რა არის პროგრამირება?-----5 წთ

მონაწილეებს ჯერ ვუსვამთ კითხვებს, ვიგებთ მათ აზრს, შემდეგ ვაჯერებთ და ვაყალიბებთ სწორ განმარტებას:

პროგრამირება არის პროცესი, რომელიც მოიცავს: ამოცანის ამოხსნის გზის მოფიქრების და შემდეგ ამ ამოხსნის კომპიუტერისთვის თავის ენაზე მიწოდებას.

- რა არის პროგრამა? -----5წთ

მონაწილეებს ჯერ ვუსვამთ კითხვებს, ვიგებთ მათ აზრს, შემდეგ ვაჯერებთ და ვაყალიბებთ სწორ განმარტებას:

ამოხსნის გზის მოფიქრების შემდეგ ვიწყებთ პროგრამის წერას, რომელმაც უნდა იმუშაოს ჩვენ კომპიუტერში და გააკეთოს ის, რასაც მე მივაწვდი თავის ენაზე.

- პროგრამირების ენები (ვაკეთებთ პარალელს ენებთან, რომ მეტად გასაგები იყოს ეს ყველაფერი) -----5წთ

როგორც რეალურ ცხოვრებაში ვმეტყველებთ სხვადასხვა ენაზე, ანალოგიურად, პროგრამირებაშიც არსებობს სხვადასხვა ენა: Java, C, C++, C#, Python და ა.შ

- რა განსხვავებაა მათ შორის? -----5 წთ

არსებობს ობიექტზე ორიენტირებული ენები და ფუნქციონალური ენები

- სტაგისტიკა, თუ რომელია ყველაზე პოპულარული ენები და რომელ ენებზეა დაწერილი ცნობილი საიტები -----5 წთ

ვსაუბრობთ იმაზე, თუ როგორი პოპულარული გახდა ბოლო წლებში პროგრამირება და რამდენად საჭიროა ის თანამედროვე ცხოვრებაში

რა არის კოდი? (თან ვაჩვენებთ კოდის სქრინებს) -----5 წთ

კომპიუტერის ენაზე დაწერილი ბრძანებების ერთობლიობა, რომელიც უნდა იყოს სინგაქსურად სწორი და თანმიმდევრული

- რა სახის შეცდომები შეიძლება არსებობდეს კოდში ----- 10წთ
 - სინგაქსური, ამრობრივი, სტილური შეცდომები
 - რა უპირატესობა აქვს კარგად დაწერილ კოდს?
 - კარგი და ცუდი კოდის მაგალითები (ისეთები, ორივე რომ სწორად მუშაობდეს)

კოდში შეიძლება არსებობდეს სინგაქსური შეცდომები (სინგაქსური შეცდომა არის, სადაც ბრძანების დაწერაში არის შეცდომა დაშვებული: მაგალითად სიტყვა `print`, რომელიც არის ბეჭდვის ბრძანება, ამის ნაცვლად თუ დაწერეთ `primt`. ანუ კომპიუტერმა არ იცის `primt` რა არის და გაგვიწითლებს ამ ბრძანებას. კიდევ ერთი სინგაქსური შეცდომის მაგალითი: თითოეული ბრძანების ბოლოს იწერება `;` და თუ დაგვავიწყდა ან უბრალოდ `,` დაწერეთ ამას ვერ გაიგებს კომპიუტერი.. ეს ყველაფერი სინგაქსური შეცდომებია.. (ჩვენ ენაზე რომ ვთქვათ გრამატიკული შეცდომებია). კიდევ არსებობს შინაარსობრივი შეცდომა (ამის მაგალითია პროგრამა რომ ჩაიციკლება და მუშაობას არ დაასრულებს არასდროს). შეიძლება პროგრამა სწორად მუშაობდეს, თუმცა კოდში იყო სტილური შეცდომები (უნდა მივეჩვიოთ სწორად წერას, ეს სწორად ფიქრშიც დაგვეხმარება)

- რა ეგაპებს გაღის პროგრამა სანამ შედეგს გამოიტანს ----- 5წთ
(უბრალოდ თეორიულად მოუყევით, კომპილაციამდე რა ეგაპებს გაღის ძალიან მარტივი ენით)

ჯერ მოწმდება რამდენად სწორად არის სინგაქსურად (ანუ ჩვენ ენაზე, რომ ვთქვათ გრამატიკული შეცდომები ჰო არ არის კოდში. შემდეგ დაიწყებს თანმიმდევრულად ბრძანებების შესრულებას და უკვე შემდეგ დააკომპილირებს საბოლოო შედეგს)
შესვენება 15 წთ

- რა არის ალგორითმი ----- 10 წთ

ნებისმიერი საკითხის გადაჭრისთვის დასახული გეგმა, რომელიც იყოფა ბიჯებად. უხეშად რომ ვთქვათ ამოცანის ამოხსნა. შეგვიძლია მოვიფიქროთ ნებისმიერი საკითხისთვის მაგალითი. მაგალითად: ოთახიდან გასვლის ალგორითმი - უნდა ავღგე წავიდე კარისკენ, გავაღო კარი, გავიდე კარში, დავხურო კარი. აი ამ პროცესს ეწოდება ალგორითმი.

რა არის ბიჯი-----5 წთ

ბიჯი არის თითოეული ეტაპი (step) ამ პროცესში. მაგალითად აღგობა არის ერთი ბიჯი, კარისკენ წასვლა - მეორე და ა.შ. ალგორითმი გამოდის ბიჯების მიმდევრობა.

სხვა მარტივი ალგორითმის მოფიქრება, რომელსაც აუდიტორია მოიფიქრებს და ბიჯებად დაყოფა.

● ბიჯების მიმდევრობა-----5 წთ

ბიჯების მიმდევრობა საბოლოოდ გვაძლევს ალგორითმს, რომელიც აუცილებლად უნდა იყოს თანმიმდევრული. მაგალითად: ჯერ რომ კარი გავაღო და მერე კარისკენ წავიდე ეგ არ იქნება სწორი. ორივე ბიჯია უბრალოდ თანმიმდევრობა შევცვალეთ. შესაბამისად ალგორითმის სისწორეში დიდი მნიშვნელობა აქვს ქრონოლოგიას.

● რა როლი აქვს ალგორითმს პროგრამირებაში -----10წთ

იმისთვის, რომ დავეწეროთ პროგრამა, საჭიროა კომპიუტერს ვუთხრათ ნაბიჯ-ნაბიჯ, რა გვინდა რომ გააკეთოს. შემდეგ კომპიუტერი ამას გაიაზრებს და ჩვენ დაწერილ ბრძანებებს შეასრულებს. ეს გვაძლევს კომპიუტერულ ალგორითმს. პროგრამის დაწერაში მთავარია სწორი ალგორითმის მოფიქრება. “სწორ”-ში იგულისხმება, რამდენად ოპტიმალურია (ანუ იკავენს ნაკლებ მეხსიერებას და მეშაობს ჩქარა), რამდენად შეუძლია ალგორითმის შეცვლას აასწრაფოს ან შეანელოს პროგრამის მუშაობა. მაგალითები.

● რა არის ფსევდოკოდი-----5წთ

მოფიქრებული ალგორითმის ჩამოყალიბებით და კომპიუტერის ენასთან მიახლოებული ენით ჩაწერით ვიღებთ ფსევდოკოდს. პროგრამის დაწერამდე პირველად საჭიროა მოვიფიქროთ ალგორითმი შემდეგ ავაგოთ ფსევდოკოდი და მერე დავიწყოთ წერა. ფსევდოკოდის აგების შემდეგ, ბევრად გაგვიმარტივდება კოდის დაწერა. რეალურად ფსევდოკოდი გამოდის პაგარა გეგმა, რომელსაც სინგაქსურად გავმართავთ და მივიღებთ კოდს.

● ჩაის მომზადების ფსევდოკოდის აგება-----10 წთ

Pseudocode

- For example, for making a cup of tea:

```
Organise everything together;  
Plug in kettle;  
Put teabag in cup;  
Put water into kettle;  
Wait for kettle to boil;  
Add water to cup;  
Remove teabag with spoon/fork;  
Add milk and/or sugar;  
Serve;
```

დაწეროთ ბლინების ამოცანის ფსევდოკოდი:

გვინდა, გავაკეთოთ 3 ბლინი, მაგრამ ტაფაზე ეგევა მხოლოდ 2. იმისათვის, რომ ბლინი იყოს გემრიელი, ორივე გვერდი უნდა გამოცხვეს. თითო გვერდის გამოცხობას სჭირდება 1 წთ. მინიმუმ რამდენ წთ-ში გამოვაქსობთ 3 ბლინს?

როგორც წესი ბავშვები აქ ჯერ ამბობენ 4 წთიანი ალგორითმს და შემდეგ 3 წთიანს.



ლექცია 2

(ხანგრძლივობა 90 წთ)

- რა არის ცვლადი -----5 წთ

ცვლადი არის ობიექტი, სადაც შეგვიძლია შევინახოთ კონკრეტული ინფორმაცია.

შეგვიძლია გავცვალოთ ცვლადებში არსებული ინფორმაცია. მეხსიერებაში გამოვყოფთ ადგილს ცვლადისთვის და შემდეგ ამ ცვლადში (რეალურად გამოყოფილ მეხსიერებაში, ვწერთ მნიშვნელობას (ჩვენთვის სასურველ ინფორმაციას) აქ დაფა თუ გვექნება ძალიან კარგი იქნება ამის დახატვა მეტი თვალსაჩინოებისთვის)

- რა გიპის ცვლადები არსებობს-----5 წთ

ინფორმაცია მეტი სიმარტივისთვის დავეოთ გიპებად, მაგალითად: მთელი რიცხვები, რაციონალური რიცხვები, სიმბოლოები, გექსტი და ა.შ. ცვლადის შემოგანის დროს აუცილებელია ვიცოდეთ რა გიპის ცვლადზე ვაკეთებთ განაცხადს, რადგან კომპიუტერმა იცოდეს რა ზომის მეხსიერება გამოგვიყოს.

- Int და Float ცვლადები-----10წთ

Int - მთელი რიცხვების გიპის ცვლადი. **Int** ცვლადებში იწერება აქედან აქამდე მთელი რიცხვები.

მაგალითად: **Int** age = 24;

მაგალითები: (შეკრება გამოკლება)

Froat - რეალური რიცხვები რიცხვები. მაგალითები. გამრავლება გაყოფა

მარტივი ოპერაციები (უნაშთოდ გაყოფა)

- ცვლადებისთვის სახელების დარქმევა-----5წთ

ერთ-ერთი უმნიშვნელოვანესი ნაწილი პროგრამირებაში. იმისთვის რომ ჩვენთვისაც თვალნათელი იყოს პროგრამა და მათთვისაც ვინც ოდესმე ჩაიხედება ჩვენ კოდში, საჭიროა ცვლადებს შევურჩიოთ სწორი სახელები. შინაარსობრივი დატვირთვიდან გამომდინარე. მაგალითები ცუდი სახელებისთვის.

- ცვლადების გამოყენებით ფსევდოკოდის დაწერა (2 ამოცანა)-----20წთ

დავწეროთ ფსევდოკოდი: გვაქვს 2 ცვლადი და გვინდა რომ მნიშვნელობები გავუცვალოთ ერმანეთს. ჯერ მესამე ცვლადის გამოყენებით და შემდეგ მესამე ცვლადის გარეშე

- კონსტანტა ცვლადები -----20წთ

როდესაც ვიცით, რომ მაქვს მნიშვნელობა რომელიც არასოდეს შეიცვლება მუდმივია ანუ ესეთ ცვლადებს დავარქვით კონსტანტა ცვლადები

const int SECONDS_PER_HOUR = 3600

კონსტანტა ცვლადების სახელს ყოველთვის ვარქმევთ დიდი ასოებით. ასევე სახელი უნდა იყოს

გასაგები რადგან, კოდში ვინც ჩაიხედავს კონსტანტის მნიშვნელობას სახელიდან გამომდინარე უნდა ხვდებოდეს.

შესვენება 15 წთ:

- რა არის HTML (Hyper Text Markup Language)-----5წთ

HTML ითარგმნება როგორც, ჰიპერ ტექსტის მარკირებული ენა. არ არის პროგრამული ენა. ის გამოიყენება **web** გვერდების შექმნაში (**www World Wide Web**)

- რისგან შედგება HTML ფაილი (ტეგები) ----- 5წთ

HTML ფაილი არის მარკირებული ტეგებისგან შემდგარი ფაილი, რომლებიც ბრაუზერს გაღასკევენ გვერდის ეკრანზე გამოტანის ინფორმაციას.

- როგორ შეგვიძლია შევექმნათ HTML ფაილი
-----5წთ

მისი შექმნა შეიძლება უბრალო ტექსტური რედაქტორით. უნდა მივცეთ **htm** ან **html** გაფართოება. ლაიფწვით ფაილის შექმნა და მაგალითის წერა

- ლაიფწვით Hello World
მაგალითი-----15წთ

ავხნათ **html**, **head**, **title**, **body**, **b** ტეგები მაგალითის საშუალებით.

**** და **** ვანახოთ ზოგადი ტეგების სია, რომელსაც პრაქტიკულზე გამოვიყენებთ.

ძირითადი ტეგები	
ტეგი	აღწერა
<html>	საზღვრავს HTML დოკუმენტს
<body>	საზღვრავს დოკუმენტის სხეულს
<h1>-დან <h6>-მდე	საზღვრავს 6 ტიპის სათაურს
<p>	საზღვრავს აბზაცს
 	წყვეტავს ხაზს
<hr>	ამატებს ჰორიზონტალურ ხაზს
<!-->	საზღვრავს კომენტარებს

ტექსტის ფორმატირების ტეგები

ტეგი	აღწერა
<code></code>	საზღვრავს გამუქებულ ტექსტს
<code><big></code>	საზღვრავს დიდ ტექსტს
<code></code>	საზღვრავს ხაზგასმულ ტექსტს
<code><i></code>	საზღვრავს დახრილ ტექსტს

- დავწეროთ მაგალითი, სადაც წინა ტექსტს შევუცვლით-----10წთ
bbgcolor-ს (ფერების კოდები)
- დავწეროთ მაგალითი, სადაც წინა მაგალითის შევუცვლით-----5წთ
background-ს (როგორც ინტერნეტიდან აღებული სურათით, ისე კომპიდან)

პრაქტიკული:

ლექციაზე მიღებული თეორიული ინფორმაციის პრაქტიკაში გამოყენება
Hello World-ის გაუმჯობესება:

ბ
დავს



ლექცია 3

(ხანგრძლივობა 90 წთ)

- char ტიპის ცვლადები -----5 წთ

char ტიპის ცვლადებში შეგვიძლია შევინახოთ სიმბოლოები. ნებისმიერ სიმბოლოს, (ტექსტის შემადგენელ ნაწილს) შეესაბამება თავისი კოდი, რომელიც *char* ტიპის ცვლადებში ინახება.

- რამდენიმე მაგალითის ჩვენება როგორ ვინახავთ *char* ტიპში-----10 წთ

მაგალითად: *char letter = 'A';* ამ ჩანაწერის გაშიფრვა, ყოველი ბრძანების შემდეგ იწერება ; და სიმბოლოები მოქცეულია " " მსგავს ბრჭყალებში

- რამდენიმე ამოცანა *char* ტიპის ობიექტებზე-----15 წთ

მაგალითად: მაგალითად გვაქვს 2 ცვლადი "A" და "B". რას დაბეჭდავს "A" - "B"? მსგავსი ტიპის ამოცანები ბევრი, რადგან კარგად გაჯდეს *char* ტიპის მნიშვნელობა

- *string* ტიპის ცვლადები -----10 წთ

String ტიპის ცვლადებში შეგვიძლია შევინახოთ ნებისმიერი ზომის ტექსტი. სხვა ტიპებისგან განსხვავებით *String* ტიპის ზომა არის განუსაზღვრელი. ანუ შეგვიძლია უზარმაზარი ტექსტიც შევინახოთ ერთ ცვლადში. რასაც სხვა ცვლადებზე ვერ ვიგყვით. ყველა დანარჩენ ცვლადში განსაზღვრულია რა ზომის "ყუთები" გვაქვს

- რისგან შედგება *string* ტიპის ობიექტი-----5 წთ

String ტიპის ობიექტი შედგება *char* ობიექტებისგან. ტექსტი შედგება სიმბოლოებისგან. 2 *char* ტიპის ობიექტის შეწყობისგან მივიღებთ *string* ტიპის ობიექტს.

შესვენება 15 წთ

- რამდენიმე მაგალითის ჩვენება როგორ ვინახავთ *string* ტიპში ტექსტს-----10 წთ

მაგალითად: *string text = "Hello World!";* რა განსხვავებაა ღიდ ტექსტს თუ მივანიჭებთ.. რა მარტივი ოპერაციების ჩატარება შეგვიძლია *string* ტიპის ობიექტებზე:

- რამდენიმე ამოცანა *string* ტიპის ობიექტებზე-----15 წთ

მაგალითი: მაგალითად გვაქვს 2 ცვლადი
string name = "Tamuna";

```
string surname = "Keshelava";  
string fullname = ?
```

+ მეთოდის მნიშვნელობა (იგივე კონკატენაცია).
კონკატენაცია 3 ცვლადის შემთხვევაში;

- რა მნიშვნელობა აქვს სწორად დარქმეულ სახელს -----5 წთ

მაგალითად: მაგალითად გვაქვს 2 ცვლადი

```
string a = "Keshelava";  
string b = "Tamuna";
```

ვინმე უცხელი თუ ჩაიხედავს კოდში ვერ მისვდება მარტივად რა არის Tamuna. წინა მაგალითში კი შინაარსი ნათელია და ნებისმიერი ადამიანი გაიგებს რა მნიშვნელობები წერია ცვლადში. და რომელ ცვლადს რა დატვირთვა აქვს

- char ტიპის ცვლადები string-ში -----15 წთ

როგორც ვიცით, string შედგება char-ებისგან. თუმცა string-იც შემიძლია იყოს 1 სიმბოლოიანი.
მაგალითად: char იწერება ყოველთვის ' ' - ამაში, string " " ამაში.
ჩვენ შეგვიძლია strings დავუმატოთ char და მივიღოთ string. ვხვდებით, რომ პირიქით ვერ მოხდება.

მაგალითად:

```
string header = "Hello World";  
Char symbol = '!';  
String result = header + symbol;
```

ეს ამოცანა აგრეთვე შეგვიძლია ჩავწეროთ ასე

```
string header = "Hello World";  
Char symbol = '!';  
header = header + symbol;
```

- += ოპერაცია, ანალოგიურად -= -----5 წთ

```
header += symbol;
```

+= ოპერაცია არის შემოკლება. როდესაც თავის თავს ვუმატებთ ახალ ცვლად

პრაქტიკული:

Exercises

1. Explain in plain English what a variable is and is not.
2. Explain in plain English what a constant is and is not.
3. Declare an integer variable named `dice_roll_sum`, designed to hold very large, strictly positive numbers (please pick the most appropriate data type if more than one type is possible).
4. Declare an integer variable named `quarterly_income`, designed to hold both negative and positive numbers.
5. Initialize a constant named `ABSOLUTE_ZERO_TEMP` to -273.
6. Initialize four variables: `q1_income`, `q2_income`, `q3_income` and `q4_income` to 200, -50, 100 and 0 respectively. Then create a new variable, `annual_income`, which you set to equal the sum of all of the above variables. In the end, please also write down the value of `annual_income`.

11. Write down the ending value of `x`.

```
int x = 5;
```

```
int y = 3;
```

```
x = x + 5;
```

```
x = x - y;
```

12. Write down the ending value of `z`.

```
int x = 10;
```

```
int y = 3;
```

```
int z = 4 * x + y;
```

13. Write down the ending value of `z`.

```
int x = 8 / 2;
```

```
int y = x * 2;
```

```
int z = (y - x) / x;
```

```
x = z * x * y;
```

14. ADVANCED What mistakes can you find in the code below?

```
int x = 1;
```

```
int y = x;
```

```
int z = x + y;
```

```
Int x = z + x + y;
```

```
int q = q + x;
```

11. Explain, in plain English, what string concatenation is.

12. Please write down the ending value of the variable `result`:

```
var string a = "The";
```

```
var string b = " ";
```

```
var string c = "fantas";
var string d = "tic";
var string e = "4";
var string result = a + b + c + d + b + e
```

13. ADVANCED Please initialize a new string variable named phone_nbr and set its value to equal "+1 (212) 123 4567" without using any numbers anywhere in your code:

```
var string country_code = "1"
var string area_code = "212"
var string local_nbr = "123 4567"
```



ლექცია 4

(ხანგრძლივობა 90 წთ)

- if ცნება ყოველდღიურ ცხოვრებაში -----5 წთ

თუ კარი ღია არ არის კარი უნდა გაფაღო. ესეთი მაგალითები, რომლებსაც თვითონ მოიფიქრებენ აუდიტორიაში.

- ფსევდოკოდში if ბრძანების დაწერა. -----10 წთ

ამოცანას მოვიფიქრებთ, რომელსაც if ჭირდება. წინა ლექციის გამეორება იქნება თან ifს ჩავღვათ ფსევდოკოდში.

- არსებული ქეისების გარჩევა. შევა თუ არა ifს ტანში პროგრამა-----15 წთ

მაგალითად:

```

Int age = 19;
if(age > 18)
{
    მიეყიღოთ სასმელი;
}

```

მსგავსი მაგალითებით ცოცხა გაგართულოთ

- else ცნება ყოველდღიურ ცხოვრებაში -----5 წთ

თუ დღეს არ იწვიმებს წაველ კინოში თუ არადა დავრჩები სახლში.

- ფსევდოკოდში else ბრძანების დაწერა. -----10 წთ

გადმოვიტანოთ პროგრამირებაში if else statement . წინა გარჩეულ მაგალითში ჩავამატოთ else ბრძანებაც.

შესვენება 15 წთ

- if-ის მუშაობის პრინციპი ----- 10 წთ

ჩაეწეროთ ჯერ ცული სტილით, შემდეგ გაეცაწოროთ და გასაგები სტილით ჩაეწეროთ.

- if / else დაწერის კულტურა. ტანის გამოყოფა ----- 5 წთ

ჩაეწეროთ ჯერ ცული სტილით, შემდეგ გაეცაწოროთ და გასაგები სტილით ჩაეწეროთ.

- else-ში ჩადგმული if----- 10 წთ

```

If (CARD is RED)
    Award YOUR team 1 point
Else
    If ( CARD is higher than 9)
        Award OTHER team 1 point
    Else
        Award YOUR team the same
        number of points on the card

```

Here is the same program in pseudocode:

```

If (card.color == RED) {
    points.yours = points.yours + 1;
}

Else {
    If ( card.value > 9) {
        points.other = points.other + 1;
    }

    Else {
        points.yours = points.yours + card.value;
    }
}

```

ამ მაგალითის გარჩევა.

- ამ მაგალითის უკეთ გადაწერა. Else if (...) ----- 10 წთ

გადაწერეთ მუსტად იგივე მაგალითი მედმეტი ჩადგმების გარეშე. ავსსნათ *else if statement*.

- დავწეროთ ამოცანები if/else statement-ების გამოტენებით ----- 10
წთ

პრაქტიკული:

1. დავწეროთ დიდი კოდი და გავიაროთ ნაბიჯ-ნაბიჯ როგორ იმუშავებს პროგრამა.
მაგალითად:

```
int x = 13;  
int y = 5;  
  
x+=2;  
y+=x;  
if(x == 10)  
{  
    Print ("Yes");  
}  
Else  
{  
    Print("No");  
}
```

- 2 რიცხვს შორის მაქსიმუმის პოვნა:

```
if (x > y) max = x;  
else     max = y;
```

- 3 რიცხვს შორის მაქსიმუმის პოვნა. რამდენიმე ალგორითმის მოსმენა: ავაროთ ყველაზე
ფსევდოკოდი და შემდეგ ჩავწეროთ კოდის სახით. ერთ-ერთი მაგალითი:

```

#include <iostream>
#include <conio>

int maximum (int,int,int);

main()
{
    int a,b,c;

    cout<<"Enter Three Integers: ";
    cin>>a>>b>>c;
    cout<<"Maximum is: "<<maximum(a,b,c)<<endl;
    getch();
    return 0;
}

maximum (int a,int b,int c)
{
    if(a>b)
    {
        if(a>c)
        { cout<<a;
        }
    }
    elseif(b>a)
    {
        if(b>c)
        {
            cout<<b;
        }
    }
    elseif(c>b)
    {
        if(c>a)
        {
            cout<<c;
        }
    }
}

```

```

#include <iostream>
#include <conio>

int maximum (int,int,int);

main()
{
    int a,b,c;

    cout<<"Enter Three Integers: ";
    cin>>a>>b>>c;
    cout<<"Maximum is: "<<maximum(a,b,c)<<endl;
    getch();
    return 0;
}

maximum (int a,int b,int c)
{
    if(a>b)
    {
        if(a>c)
        { cout<<a;
        }
    }
    elseif(b>a)
    {
        if(b>c)
        {
            cout<<b;
        }
    }
    elseif(c>b)
    {
        if(c>a)
        {
            cout<<c;
        }
    }
}

```

4. The following if-else statement adds x to a sum of positive numbers and increments a count of positive numbers if it is positive. Similarly if x is negative it is added to a sum of negative numbers and a count of negative numbers is incremented.

```

if (x >= 0.0)
{
    sumpos += x;
    poscount++;
}
else
{
    sumneg += x;
    negcount++;
}

```

ლექცია 5-6

(ხანგრძლივობა 90 წთ)

- ერთი და იგივე მოძრაობის რამდენჯერმე გამეორება (ფსევდოკოდის აგება) -----10 წთ

მოვიფიქროთ ისეთი ამოცანა, სადაც გვიწევს ერთიდაიგივე მოძრაობის ბევრჯერ გამეორება. მაგალითად: ერთ ყუთში აწყვია წიგნები და მინდა მეორე ყუთში გადავაღებო. ამ ამოცანის ფსევდოკოდი

- ამოცანის გადაკეთება ისე, რომ ციკლის იდეა გამოჩნდეს.-----5 წთ

გავხსნათ ყუთი. - ამოვიღო წიგნი პირველიდან ჩავლო მეორეში - ეს მოძრაობა გავიმეორო რამდენი წიგნიცაა იმდენჯერ. ღაფხურო მეორე ყუთი

- for ციკლის ფსევდოკოდით ჩაწერა ამ ამოცანის მიხედვით. -----10 წთ

ყუთის გახსნა;

for(რამდენი წიგნიცაა იმდენჯერ)

{

წიგნის ამოღება და მეორეში გადაღება;

}

ყუთის დახურვა;

- for ციკლის სინტაქსი (ჩაწერის სტილი), ტანი და ფრჩხილებში მოქცეული ნაწილის 3 ნაწილად დაყოფა -----10 წთ

for(int i = 0; i < 10; i++)

{

}

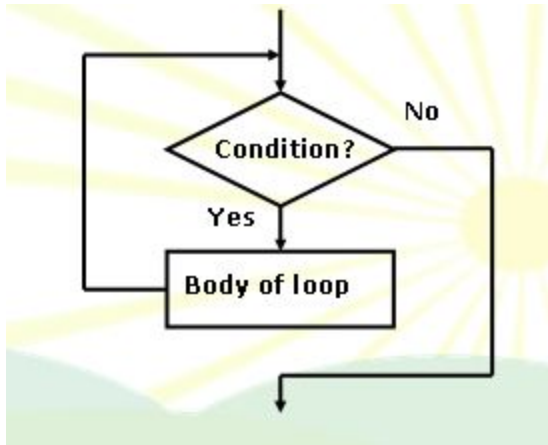
შესვენება 15 წთ:

- ახალი ამოცანის ფსევდოკოდის აგება -----20 წთ

მინდა ღაფთფალოთ პირველი 5 რიცხვის ჯამი. ჯერ ჩაეწერთ ციკლის გარეში, შემდეგ **for**-ით ფსევდოკოდი ავაგოთ და შემდეგ **for**-ით კოდი დაეწეროთ

- **while** ციკლის იდეა -----5 წთ

ციკლით ჩაწერის მეორენაირი ხერხი.



- **while** ციკლის სინტაქსი -----10 წთ

while (true)

```
{
    if(i < 5)
    {
        break;
    }
}
```

- **for**-ით ჩაწერილი კოდის **while**ში გადმოწერა -----10 წთ

ორივე ამოცანა გადმოეწეროთ **whil**-ში.

while(true)

```
{
    If (ყუთი ცარიელია)
    {
        Break;
    }
}
```

პრაქტიკული:

1. დაბეჭდეთ 1-დან 40-მდე ყველა კენტი რიცხვი.
2. დათვალოთ 10-დან 100-მდე 5-ის ჯერადი რიცხვების ჯამი.
3. დათვალოთ მოცემული მთელი რიცხვის ფაქტორიალი.

რას დაბეჭდავს?

1.

```
int i = 0;
while( i < 3)
{
    print ("hi");
    i++;
}
```

2.

```
int i = 0;
while( i < 3)
{
    print ("hi");
    i++;
}
print("by");
```

3.

```
int i = 0;
while( i < 0)
{
    print ("hi");
    i++;
}
```

4.

```
int x = 3;
int i = 0;
while( i < 3)
{
    x += 1;
}
```

```
    i +=1;
}
print(x);
```

5.

```
int i = 3;
while( i < 3)
{
    print (i);
    i += 1;
}
```

6.

```
int i = 0;
while( i < 3)
{
    print (i);
    i++;
}
```

7.

```
for(int i = 0; i < 5; i++)
{
    if(i % 2 == 0)
    {
        print(i*2);
    }
}
```

8.

```
int counter = 15;
for(int i = 0; i < 6; i++)
{
    counter -= i;
}
print(counter);
```

9.

```
int x = 1;  
int i = 1;  
while(i < 5)  
{  
    x*=i;  
    i++;  
}  
print(x);
```


ლექცია 7

(ხანგრძლივობა 90 წთ)

- კონტენერის იდეა, მათი საჭიროება-----10 წთ

გვაქვს ინფორმაცია შესანახი. მაგალითად კლასის სია მინდა ჩაწერო. რა ვარიანტები არსებობს? შემოვიტანოთ იმდენივე ცვლადი რამდენი მოსწავლეცა.. ეს მოუხერხებელია რატომ?

- ყუთების საშუალებით ავსხნათ ერთიდაიგივე ტიპის მასივი.-----10 წთ

ავიღოთ იმდენი ყუთი რამდენი მოსწავლეც მყავს. ჩაწეროთ მასში სახელები. სხვადასხვა ტიპზე მათგალითები

- ყუთების გადანომრვა. მასივის ინდექსები. მასივის ზომა. მასივიდან ობიექტის ამოღება-----10 წთ

ნუმერაცია იწყება 0-ით. პირველი ობიექტი იქნება 0 ყუთში. ბოლო ელემენტი იქნება ზომას - 1 -ე ინდექსზე.

- მასივში ინფორმაციის შენახვა და ამოღება -----10 წთ

წინასწარ უნდა ვიცოდეთ რა ზომის მასივს ვაკეთებთ, რომ გამოვყოთ მეხსიერება. შემდეგ ჩაწეროთ ინფო `array[0] = "Tamuna"`; და ა.შ ჩამოვწეროთ ყველა სახელი და ამოვიღოთ ინფო `print(array[0])`;

- როგორ მოვიქცეთ უფრო ოპტიმალური რომ გავხადოთ ჩვენი მუშაობა? როგორ შემიძლია მასივის ყველა ობიექტის ამოღება. ამდენი წერის გარეშე?-----5 წთ

ციკლით გადავუყვებო.

შესვენება 15 წთ:

- წინა მაგალითი გადავწერთ ციკლით ვნახოთ რამდენ ხაზს დავზოგავთ. -----10 წთ

```
for(int i = 0; i < 5; i++)
{
    print (array[i]);
}
```

- გვაქვს რიცხვების მასივი. როგორ ვიპოვოთ მაქსიმუმი? მინიმუმი? ორივე? -----15 წთ

```
public class HelloWorld {
    public static void main(String[] args) {
        int numbers[] = new int[]{8, 2, 7, 1, 4, 9, 5};
        int s = numbers[0];
        int l = numbers[0];

        for(int i = 1; i < numbers.length; i++) {
            if(numbers[i] > l) l = numbers[i];
            else if (numbers[i] < s) s = numbers[i];
        }
        System.out.println("Largest Number is : " + l);
        System.out.println("Smallest Number is : " + s);
    }
}
```

- გავამახვილოთ ყურადღება ინდექსების ნუმერაციაზე. For-ში size-მდე უნდა მივიდეთ-----.5 წთ

```
for(int i = 0; i < array.size(); i++)
{
    print (array[i]);
}
for(int i = 0; i <= array.size() - 1; i++)
{
    print (array[i]);
}
```

- დავწეროთ მაგალითი: მასივში მოცემული გვაქვს 20 რიცხვი (არ არის აუცილებელი მრავლობითი დალაგებული) და დავბეჭდოთ მხოლოდ 3-ის ჯერადები -----15 წთ

```
for(int i = 0; i < 20; i++)
{
    if(array[i] % 3 == 0)
    {
        print(array[i]);
    }
}
```

}
}

1. დაებეჭდოთ მასივის ყველა ელემენტი.
2. დაებეჭდოთ მასივის ელემენტებიდან მხოლოდ ლუწი რიცხვები
3. დაებეჭდოთ მასივის ელემენტებიდან მხოლოდ 3-ის ჯერადები
4. დაწეროთ მასივის ელემენტების ჯამი
5. ვიპოვოთ მასივში მაქსიმალური მნიშვნელობა და დაებეჭდოთ.
6. ვიპოვოთ მასივში მინიმალური მნიშვნელობა და დაებეჭდოთ