ნ.თოდუა ზ.ისააკიანი

კომპიუტერული ტექნოლოგიების საფუძვლები და დაპროგრამება ბეისიკზე

დამტკიცებულია სტუ-ს
სარედაქციო საგამომცემლო
საბჭოს მიერ

თბილისი
2006

ᲣᲐᲙ 681.3.06

დამხმარე სახელმძღვანელოში “კომპიუტერული ტექნოლოგიების საპუძვლები და დაპროგრამება ბეისიკზე” განხილულია კომპიუტეტერული ტექნოლოგიების ძირითადი ცნებები და ალგორითმიზაციისა და ბეისიკზე დაპროგრამების საფუძვლები. თეორიული მასალის ადვილად ათვისებისათვის სახელმძღვანელოში მოცემულია მრავალი პრაქტიკული მაგალითი. ლაბორატორიული სამუშაოების შესრულების მაგალითებში მოცემულია ამოცანების ამოხსნის ალგორითმის ბლოკ-სქემა და შესაბამისი პროგრამის ტექსტი QBasic-ზე.

დამხმარე სახელმძღვანელო განკუთვნილია სამშენებლო ფაკულტეტის სტუდენტებისათვის და იმათთვის, ვინც აპირებს ალგორითმიზაციისა და დაპროგრამების შესწავლას.

რეცენზენტები: პროფ. მ.კუბლაშვილი
პროფ. ო.შონია

ISBN

ს ა რ ჩ ე ვ ი

შესავალი 4
I თავი. კომპიუტერული ტექნოლოგიების ძირითადი ცნებები 5
II თავი. ბეისიკის ძირითადი ელემენტები და კონსტრუქციები 12
III თავი. გამოსახულება 17
IV თავი. QBasic სისტემის დაპროგრამების გარემო 23
V თავი. წრფივი სტრუქტურის ალგორითმის დაპროგრამება 27
VI თავი. განშტოებადი სტრუქტურის ალგორითმის დაპროგრამება 34
VII თავი. ციკლური სტრუქტურის ალგორითმის დაპროგრამება 42
VIII თავი. მასივების დამუშავება 62
IX თავი. დაპროგრამება ქვეპროგრამის გამოყენებით 79
ლიტერატურა 96

შესავალი

ბეისიკი (**BASIC**) შეიქმნა 1964 წელს ამერიკის შეერთებულ შტატებში იმ ადამიანებისათვის, რომლებიც არ ფლობდნენ კომპიუტერთან ურთიერთობის ენებს. **BASIC**-ი ინგლისური წინადადების "**Beginner's All-purpose Symbolic Instruction Code**" აბრევიატურაა, შემადგენელი სიტყვების პირველი ასოებისგან შედგენილი სიტყვაა. ამ წინადადების ქართული თარგმანია: „სიმბოლური ინსტრუქციების მრავალმიზნობრივი ენა დამწყებთათვის". ეს ენა ფირმა "**General Electric**"-ის შეკვეთით შექმნეს დორტმუტის კოლეჯის თანამშრომლებმა ჯ.კემენიმ და თ.კურცმა.

ბეისიკმა სულ მალე დიდი პოპულარობა მოიპოვა კომპიუტერის მომხმარებლებს შორის თავისი სიმარტივითა და შესაძლებლობებით. **ბეისიკის** მეორე მნიშვნელოვანი მხარე ის არის, რომ იგი საშუალებას გვაძლევს ამოვხსნათ ამოცანები მანქანასთან დიალოგში. რეალიზაციის სიმარტივემ დიდი იმპულსი მისცა **ბეისიკის** ფართო გამოყენებას დაპროგრამების შესწავლის პრაქტიკაში.

ბეისიკზე შეიძლება შეიქმნას საკმაოდ რთული და მრავალგვარი პროგრამები, ასეთია, მაგალითად: სათამაშო პროგრამები, ინფორმაციის სტატისტიკური დამუშავების, დაპროგრამების ავტომატიზაციისა და ობიექტის მართვის პროგრამები და სხვა. **ბეისიკი** ითვლება პერსონალური კომპიუტერისათვის დიალოგური პროგრამების შექმნის საუკეთესო საშუალებად.

ბეისიკის ძირითადი ელემენტები და კონსტრუქციები, რომლებსაც ჩვენ განვიხილავთ, წარმოადგენს **QBASIC**, **Turbo-BASIC**, **Visual BASIC (VB)** და **Visual BASIC for Applications (VBA)** ვერსიების ძირითად ბირთვებს.

I თავი. კომპიუტერული ტექნოლოგიების ძირითადი ცნებები

1.1. კომპიუტერის აპარატურის შედგენილობა (Hardware)

პროცესორი კომპიუტერის ცენტრალური კვანძია. იგი წარმართავს სხვა კვანძების მუშაობას და ინფორმაციის დამუშავების პროცესს. პროცესორი ხასიათდება ტიპით და სწრაფქმედებით, რაც მეგაჰერცობით იზომება. მაგალითად, **Pentium 4**, **2**Ghz (2000Mhz).

მეხსიერება – სივრცე, სადაც ინახება დასამუშავებელი ინფორმაცია – სამი ტიპისაა:

1) ოპერატიული მეხსიერება არის ელექტრონული სქემა, სადაც ინახება ის ინფორმაცია, რომელიც დამუშავების პროცესშია.
2) მეხსიერება მყარ დისკოზე ანუ ე.წ. ვინჩესტერი. ეს კომპიუტერის ძირითადი მეხსიერებაა და წარმოადგენს დამცავ კოლოფში მოთავსებულ დისკოს (**Hard Disc**). აქ ინახება ინფორმაცია, რომლითაც გამუდმებით ვსარგებლობთ.
3) ცვლადი მეხსიერება. ამ ტიპის მეხსიერებას მიეკუთვნება: მაგნიტური დისკეტები, სხვადასხვა ტექნოლოგიის კომპაქტდისკები (**CD**). მათი მეშვეობით ხდება ინფორმაციის გაცვლა კომპიუტერებს შორის და ჩვენ კომპიუტერზე ახალი პროგრამების დაყენება.

ყველაზე სწრაფქმედია ოპერატიული მეხსიერება. ყველაზე დიდი მოცულობა ვინჩესტერს აქვს.

მონიტორი. ეკრანიანი მოწყობილობაა, რომელზეც კომპიუტერი გასცემს იმფორმაციის დამუშავების შედეგებს და მასთან ურთიერთობისათვის აუცილებელ დამხმარე ინფორმაციას. გარდა ამისა, მონიტორი საშუალებას გვაძლევს თვალნათლივ ვაკონტროლოთ ჩვენ მიერ კლავიატურაზე აკრებილი ინფორმაციის სისწორე და მაუსის (**mouse**) მუშაობის პროცესს. მონიტორი ხასიათდება ეკრანის დიაგონალის ზომით და ე.წ. ეკრანის გარჩევადობის მაჩვენებლით (**dpi**), რითაც განისაზღვრება გამოსახულების შესაძლო ხარისხი.

კლავიატურა. კლავიატურის მეშვეობით კომპიუტერს მიეწოდება დასამუშავებელი ინფორმაცია და კომპიუტერთან ურთიერთობისათვის აუცილებელი ბრძანებები.

პრინტერი. საბეჭდი მოწყობილობაა რომლის მეშვეობითაც ხდება ინფორმაციის გამობეჭდვა ქაღალდზე. პრინტერის მახასიათებლებია: ბეჭდვის სისწრაფე, ფურცლის ერთ დიუმზე წერტილების რაოდენობა ანუ ნაბეჭდის გარჩევადობის მაჩვენებელი (**dpi**) და ფურცლის შესაძლო ფორმატი. არსებობს როგორც შავ-თეთრი, ისე ფერადი პრინტერები. მოქმედების პრინციპის მიხედვით განასხვავებენ მატრიცულ, ჭავლურ და ლაზერულ პრინტერებს.

სკანერი. ინფორმაციის ვიზუალურად წამკითხავი მოწყობილობაა, რომლის მეშვეობითაც კომპიუტერის მეხსიერებაში სწრაფად შეიტანება ტექსტური, გრაფიკული და ფოტოდოკუმენტები.

პლოტერი. საბეჭდი მოწყობილობაა, რომლის მეშვეობითაც ხდება ქაღალდზე დიდი ზომის ნახაზების გამოხაზვა.

პროცესორი, ოპერატიული მეხსიერება, ვინჩესტერი და ცვლადი მეხსიერების მოწყობილობები განთავსებულია ერთ აპარატულ ბლოკში, რომელსაც სისტემური ბლოკი ეწოდება, სისტემურ ბლოკში აგრეთვე მოთავსებულია:

ვიდეოპლატა. ვიდეოპლატის ანუ ვიდეოადაპტერის მეშვეობით პროცესორი მართავს მონიტორის ეკრანზე გამოსახულების გადაცემის პროცესს. მისი მახასიათებლებია: სწრაფქმედება და ვიდეომეხსიერების მოცულობა. ამ მახასიათებლებს განსაკუთრებული მნიშვნელობა ენიჭება გრაფიკული ინფორმაციის ინტენსიური დამუშავებისას.

ხმოვანი პლატა. ემსახურება ხმის გენერირებას. ბევრი თანამედროვე პროგრამა (მათ შორის, **Windows**) გახმოვანებულია. თუ ეს პლატა არ გვაქვს, ხმოვანი თანხლება არ მუშაობს.

სისტემური პლატა. სისტემური ანუ დედაპლატა (**Motherboard**) ელექტრონული სქემაა, რომელიც ქმნის საერთო საფუძველს კომპიუტერის აპარატული კვანძების ერთმანეთთან დასაკავშირებლად. პლატაზე გვაქვს ბუდეები, რომლებშიც ჩადგმულია პროცესორის, ოპერატიული მეხსიერების, ვიდეო და ხმოვანი პლატები. სხვა მოწყობილობებთან (ვინჩესტერთან, ცვლად მეხსიერებასთან) კავშირი ხდება სპეციალური სადენების მეშვეობით. სისტემური პლატა ხასიათდება გამოყენებულ მიკროსქემათა ტიპით, სწრაფქმედებით (**Bus speed**), აგრეთვე ბუდეთა ტიპითა და რაოდენობით.

1.2. კომპიუტერული ქსელი

ცალკეული კომპიუტერები შეიძლება გავაერთიანოთ ფუნქციური შესაძლებლობების გაფართოებისა და რესურსების უკეთ გამოყენების მიზნით. ასეთ გაერთიანებას კომპიუტერული ქსელი ეწოდება. ქსელი შეიძლება იყოს ლოკალური (ვთქვათ, ერთი დაწესებულების ფარგლებში) ან გლობალური. გლობალური ქსელის მაგალითია საერთაშორისო საინფორმაციო ქსელი **Internet**-ი. კავშირი ქსელის კომპიუტერებს შორის სხვადასხვა სახის საინფორმაციო არხებით ხორციელდება. ქსელში მუშაობისათვის კომპიუტერი უნდა აღიჭურვოს სპეციალური აპარატურული კვანძებით – მოდემებითა და ქსელური ადაპტორებით.

1.3. კომპიუტერული სისტემის სწრაფქმედება

ზემომოყვანილიდან ჩანს, რომ კომპიუტერი ცალკეული კვანძებისაგან შედგენილი სისტემაა. ამიტომ გასაგებია, რომ კომპიუტერული სისტემების სწრაფქმედებას და, შესაბამისად, ინფორმაციის დამუშავების სისწრაფეს განსაზღვრავს ერთდროულად რამდენიმე ფაქტორი: პროცესორის სწრაფქმედება, ოპერატიული მეხსიერების მოცულობა და სწრაფქმედება, ვინჩესტერის სწრაფქმედება, სისტემური პლატის ტიპი, ვიდეოადაპტერის პარამეტრები. აქედან გამომდინარე, მაღალი სწრაფქმედების პროცესორის გამოყენება არ ნიშნავს ავტომატურად კომპიუტერული სისტემის ფუნქციონირებას იმავე სისწრაფით. ამისათვის შესაბამისობაში უნდა იქნეს მოყვანილი სისტემის სხვა პარამეტრები, რომლებიც ზეგავლენას ახდენს სწრაფქმედებაზე. ეს გარემოება კომპიუტერული სისტემის კომპლექტაციისას მიიღება მხედველობაში.

1.4. კომპიუტერების კლასიფიკაცია

ფუნქციური დანიშნულების, მწარმოებლობისა და კონსტრუქციული შესრულების მიხედვით შეიძლება გამოვყოთ თანამედროვე კომპიუტერების შემდეგი ძირითადი კლასები:

- პერსონალური კომპიუტერები,
- სერვერები,
- ჯიბის კომპიუტერები,
- სპეციალიზებული კომპიუტერები.

1) პერსონალური კომპიუტერები საშუალო მწარმოებლობისაა და ფართოდ გამოიყენება ოფისებსა და ყოფა-ცხოვრებაში. კონსტრუქციული

შესრულების თვალსაზრისით, განასხვავებენ სამაგიდო (**Desktop**) და გადასატან (**NoteBook**) პერსონალურ კომპიუტერებს.

2) სერვერები მაღალი მწარმოებლობისა და საიმედოობის მქონე კომპიუტერებია, რომლებიც ძირითადად კომპიუტერულ ქსელებში (მაგალითად, ინტერნეტის **Web** – კვანძებში) მმართველი კომპიუტერების სახით გამოიყენება.
3) ჯიბის კომპიუტერები (**Pocket PC**) მინიატურული კომპიუტერებია მათთვის დამახასიათებელი ყველა ატრიბუტით.
4) სპეციალიზებული კომპიუტერები გამოიყენება კონკრეტული დარგის ამოცანების გადასაწყვეტად. ამ კლასის კომპიუტერები ძირითადად იხმარება რეალურ დროში ობიექტებისა და პროცესების სამართავად, რობოტებში და სხვ.

კომპიუტერთა უმრავლესობა უნივერსალური დანიშნულებისაა. მათი გამოყენების სფერო განისაზღვრება იმ პროგრამებით, რომლებსაც ამ კომპიუტერში მოათავსებთ.

1.5. პროგრამული მართვის პრინციპი

კომპიუტერზე ინფორმაციის დამუშავებას საფუძვლად უდევს პროგრამული მართვის პრინციპი.

პროგრამა ბრძანებათა ერთობლიობაა, რომელსაც ასრულებს კომპიუტერი ინფორმაციის დამუშავების პროცესში.

პროგრამა და დასამუშავებელი მონაცემები თავიდან ინახება ვინჩესტერზე ან ცვლადი მეხსიერების დისკოზე. პროცესორი თანამიმდევრულად დისკური მეხსიერებიდან ირჩევს პროგრამის ბრძანებებს, შესაბამის მონაცემებს და ათავსებს მათ ოპერატიულ მეხსიერებაში, სადაც სრულდება ბრძანებით გათვალისწინებული ოპერაცია. ოპერაციის შედეგი კვლავ დისკოს მეხსიერებაში გადაეცემა. პროცესი მეორდება მანამ, სანამ აირჩევა და შესრულდება პროგრამის ყველა ბრძანება.

1.6. კომპიუტერის პროგრამული შედგენილობა

კომპიუტერის პროგრამული ნაწილი ანუ პროგრამული უზრუნველყოფა (**Software**) ორი სახის პროგრამას შეიცავს: გამოყენებით პროგრამებსა და სისტემურ პროგრამებს.

1.6.1 გამოყენებითი პროგრამები

გამოყენებითი პროგრამების მეშვეობით ხდება კონკრეტული ამოცანების გადაწყვეტა. მათ მიეკუთვნება:

1) სპეციალური დანიშნულების პროგრამები, მაგალითად: სამეცნიერო-ტექნიკური, ეკონომიკურ და ფინანსურ გაანგარიშებათა პროგრამები და პროგრამული პაკეტები, სასწავლო პროგრამები, დიაგნოსტიკური პროგრამები, პროგრამა თამაშები და სხვ.
2) ზოგადი დანიშნულების გამოყენებითი პროგრამები. მაგალითად: დოკუმენტების მომზადების პროგრამები, რომელთაც ტექსტურ, გრაფიკულ და ცხრილურ რედაქტორებს უწოდებენ. ესენია: **Word, Excel, Corel Draw, Photoshop** და სხვ.
3) მონაცემთა ბაზის მართვის სისტემა. მონაცემთა ბაზა ინფორმაციული სისტემაა, რომელსაც მონაცემთა მიღების, შენახვისა და გაცემის გარკვეული წესი აქვს. იგი ორ კომპონენტს შეიცავს: საკუთრივ მონაცემთა ერთობლიობას და პროგრამას, რომელიც მას მართავს. მაგალითად: **Access, FoxPro** და სხვ.

1.6.2. სისტემური პროგრამები

სისტემური პროგრამები უზრუნველყოფს კომპიუტერის აპარატურის ფუნქციონირებას, მათ მომსახურება-დიაგნოსტიკას, სხვა პროგრამის გამოყენებას და კომპიუტერთან მომხმარებლის კავშირს. სისტემური პროგრამული უზრუნველყოფის ძირითადი შემადგენელი ნაწილია ოპერაციული_სისტემა. ამჟამად ყველაზე გავრცელებული ოპერაციული სისტემაა **Windows.** არსებობს **Windows**-ის სხვადასხვა ვერსია: **Windows95 Windows98, Windows2000, WindowsNT, WindowsXP** და სხვ. წლების მანძილზე, პერსონალურ კომპიუტერში გამოიყენებოდა ოპერაციული სისტემა **MS DOS**. მომხმარებლის კავშირი კომპიუტერთან ხორციელდებოდა მისი ბრძანებების მეშვეობით. **MS DOS** ბრძანებათა უფრო სწრაფად და მოხერხებულად გამოყენებისათვის შექმნილი იყო სპეციალური პროგრამა **Norton Commander**.

მნიშვნელოვანი სისტემური პროგრამებია დრაივერები, რომლებიც უზრუნველყოფს ოპერაციული სისტემის მიერ პერიფერიული მოწყობილობების (მონიტორის, პრინტერის, კლავიატურის, მაუსის, მოდემის და სხვ.) მართვას. ყველა პერიფერიულ მოწყობილობას თავისი დრაივერი აქვს.

სისტემური პროგრამები მუდმივად არის ვინჩესტერზე.

7.1. ინფორმაცია და კომპიუტერი

სიტყვა “ინფორმაცია” ლათინურია და ნიშნავს “ცნობას”. ინფორმაციის ვრცელი განმარტება ასეთია:

“ინფორმაცია არის ცნობა ანუ შეტყობინება, რომელიც ზრდის ჩვენ ცოდნას ამა თუ იმ ფაქტის, სიტუაციის ან მოვლენის შესახებ და რომელიც შენახვის, გადაცემისა და გარდაქმნის ობიექტია”.

ნებისმიერი შეტყობინება დაკავშირებულია სამ ცნებასთან: შეტყობინების გამგზავნი, შეტყობინების მიმღები და გამგზავნსა და მიმღებს შორის დამაკავშირებელი არხი.

ინფორმაციის რაოდენობის საზომი ერთეულია **ბიტი**. ერთი **ბიტი** არის ინფორმაცია, რომელსაც შეიცავს შეტყობინება ელემენტარული მოვლენის შესახებ, რომელსაც აქვს ორი არჩევანი. **ბიტის** გარდა, საზომ ერთეულებად გამოიყენება აგრეთვე უფრო დიდი ერთეულები:

1 ბაიტი (**B**) = 2^3 ბიტი = **8** ბიტი

1 კილობაიტი (**KB**) = 2^{10} ბაიტი = **1024 B**

1 მეგაბაიტი (**MB**) = 2^{10} კილობაიტი = **1024 KB**

1 გიგაბაიტი (**GB**) = 2^{10} მეგაბაიტი = **1024 MB**

საზომ ერთეულად ბაიტის გამოყენება განპირობებულია იმით, რომ კომპიუტერში სიმბოლოთა წარმოდგენა ხორციელდება რვათანრიგა ციფრული კოდით, რომლის თითოეული თანრიგი ერთ **ბიტ** ინფორმაციას შეიცავს. მაგალითად: **! - 00100001(033)**, **F - 01000110(070)**.

შეტყობინებას ან ცნობას (ინფორმაციას), რომელიც კომპიუტერს შესანახად ან დასამუშავებლად გადაეცემა, მონაცემი ჰქვია, იგი წარმოდგენილია ცალკეული სიდიდეების სახით.

კომპიუტერზე ინფორმაციის დასამუშავებლად (ამოცანის ამოსახსნელად) ადამიანმა კომპიუტერს ზუსტად და ამომწურავად უნდა მიუთითოს, ამოცანის ამოხსნის გზა. ამოცანის ამოხსნის გზა – მოქმედებების მიმდევრობა (ალგორითმი) კომპიუტერს უნდა მივაწოდოთ პროგრამის სახით, რომელიც იწერება მისთვის “გასაგებ” ენაზე. პროგრამაში უნდა იყოს მითითებული, კონკრეტულად რა მოქმედებები (ოპერაციები) უნდა შესრულდეს და რა სიდიდეებზე. ამ პროგრამის შესრულება მოგვცემს ამოცანის ამოხსნის შედეგს. სხვა სიტყვებით, კომპიუტერზე ამოცანის ამოსახსნელად გასავლელია შემდეგი ეტაპები:

1) ამოცანის დასმა ანუ ამოცანის პირობის ჩამოყალიბება;
2) ამოცანის პირობიდან ცნობილი (საწყისი) და საძებნი სიდიდეების გამოყოფა და მათ შორის ცალსახა თანადობის დადგენა;
3) ამოცანის მათემატიკური აღწერილობის აგება;
4) ალგორითმის შედგენა;
5) პროგრამის შედგენა (დაპროგრამება);
6) პროგრამის გამართვა;
7) გამართული პროგრამის შესრულება;
8) მიღებული შედეგების ანალიზი.

II თავი. ბეისიკის ძირითადი ელემენტები და კონსტრუქციები

2.1. ანბანი

ანბანი განისაზღვრება შესაბამისი გამომყვანი მოწყობილობის – კლავიატურის შესაძლებლობებით.

ბეისიკის ანბანს შეადგენს შემდეგი სახის სიმბოლოები:

1. ლათინური ანბანის ასოები.
2. ქართული და რუსული ანბანის ასოები (გამოიყენება შეტყობინების გამოსატანად).
3. ციფრები **0**-დან **9**-ის ჩათვლით (ასო **O**-საგან განსასხვავებლად ნულს გადასმული აქვს დახრილი ხაზი).
4. სპეციალური სიმბოლოები:

\+ პლუსი

\- მინუსი

* ვარსკვლავი

. ათობითი წერტილი

, მძიმე

; წერტილ-მძიმე

: ორწერტილი

! ძახილის ნიშანი

? კითხვის ნიშანი

' აპოსტროფი

" ორმაგი აპოსტროფი

/ დახრილი ხაზი

\ შებრუნებული დახრილი ხაზი

␣ ხარვეზი

(გახსნილი და) დახურული მრგვალი ფრჩხილი

< ნაკლებია

= ტოლია

> მეტია

^ სახურავი

$ დოლარის ნიშანი

% პროცენტის ნიშანი

ნომერი

& ამპერსანტი

2.2. მუდმივა

მუდმივა (კონსტანტა) ეწოდება სიდიდეს, რომელიც პროგრამის შესრულების პროცესში არ იცვლის მნიშვნელობას. ბეისიკში გამოიყენება ორი ტიპის მუდმივა: რიცხვითი და სტრიქონული.

2.2.1 რიცხვითი მუდმივა

რიცხვითი მუდმივა ბეისიკში წარმოდგენილია მთელი და ნამდვილი ტიპის მუდმივებით.

მთელი ტიპის მუდმივა ჩაიწერება ათობით ციფრთა მიმდევრობის სახით. ამ მიმდევრობას წინ შეიძლება უძღოდეს ნიშანი „+“ ან „-“, შესაბამისად დადებითი ან უარყოფითი რიცხვების წარმოსადგენად. ამასთან „+“-ის ჩაწერა სავალდებულო არ არის. მთელი ტიპის მუდმივების გამოყენება ნებადართულია -32768 - 32767.

მაგალითი 1:

+28, 11030, -275.

ნამდვილი ტიპის მუდმივა, ისევე როგორც ათწილადი რიცხვი, შეიძლება შეიცავდეს მთელ და ათწილად ნაწილებს. თანამედროვე ალგორითმულ ენებზე, მათ შორის ბეისიკზეც, რიცხვის მთელი ნაწილის ათწილადი ნაწილისაგან გამოსაყოფად მძიმის ნაცვლად გამოიყენება წერტილი, რომელსაც ათობითი წერტილი ეწოდება.

მაგალითი 2:

რიცხვი	ბეისიკზე ჩაწერა
12,5	12.5
0,125	.125
-1,25	-1.25
125,00	125

დიდი და მცირე რიცხვების ჩასაწერად გამოიყენება რიცხვის წარმოდგენის ექსპონენციალური ფორმა, ე. ი. რიცხვი გამრავლებული 10-ის ხარისხზე:

$$X \cdot 10^n$$

სადაც X არის რიცხვის მანტისა, გამოისახება ნებისმიერი ათწილადით;

n –რიგი, რომელიც წარმოადგენს დადებით ან უარყოფით მთელი ტიპის ერთ- ან ორთანრიგიან მუდმივას (-38 ≤ n ≤ 38).

ექსპონენციალური ფორმა გამოიყენება იმ შემთხვევაში, როდესაც რიცხვი:

1) აღემატება ან ტოლია 9 999 999,5;
2) ნაკლებია ან ტოლია 0,00 000 001=1/100 000 000;
3) ნაკლებია 0,1, მისი ბოლო ციფრი 0-საგან განსხვავებულია და მოშორებულია მძიმეს შვიდ პოზიციაზე მეტად.

მაგალითი 3:

136 960 000 1,3696 10^8

-0,000 356 258 -3,56 258 10^{-4}

ბეისიკზე ათვლის ათობითი ფუძის აღმნიშვნელი 10-ის ნაცვლად გამოიყენება სიმბოლო, ხოლო რიგი იწერება სიმბოლოს შემდეგ.

მაგალითი 4:

რიცხვი	ბეისიკზე ჩაწერა
$1,349*10^8$	1.349E+08
$-5,338963*10^{-4}$	-5.338963E-04
$-2,56*10^{11}$	-2.56E+11
$1,25*10^{-12}$	1.25E-12
0.00000321	3.21E-06
87654321	8.7654321E+7
-0,08736025	-8.736025 E-02

2.2.2. სტრიქონული მუდმივა

სრიქონული მუდმივა (ლიტერალი) წარმოადგენს ასო-ციფრულ და სპეციალურ სიმბოლოთა ერთობლიობას. სტრიქონული მუდმივას შემადგენელი სიმბოლოების ერთობლიობა უნდა მოთავსდეს ორმაგ აპოსტროფს შორის.

მაგალითი 5:

”namravli =”, ”ამოცანის ამონახსნის შედეგი:”, ”x+y”, ”Z =”, ”BMW”, ”60”, ”60 წუთი”.

2.3. ცვლადი

სიდიდეს, რომელიც პროგრამის შესრულების პროცესში იცვლის მნიშვნელობას, ცვლადი ეწოდება. ბეისიკში განასხვავებენ ორი ტიპის ცვლადს – რიცხვითსა და სტრიქონულს.

ბეისიკზე შესაძლებელია ცვლადების სახელით აღნიშვნა. სახელი შეიძლება შეიცავდეს 1-დან 40 სიმბოლომდე. პირველი მათგანი აუცილებლად უნდა იყოს ლათინური ანბანის ასო, ხოლო დანარჩენი ან ლათინური ანბანის ასო, ან ციფრი.

რიცხვითი ტიპის ცვლადები, თავის მხრივ, იყოფა მთელი და ნამდვილი ტიპის ცვლადებად. უმეტეს შემთხვევაში მათ სახელებს ერთი და იგივე სახე აქვთ, ამიტომ მათი განსხვავების მიზნით მთელი ტიპის ცვლადის ბოლოს დაიწერება "%" ან "&", ნამდვილი ტიპის ცვლადის ბოლოს კი "!" ან . "#"

სტრიქონული ტიპის ცვლადის ბოლოს იწერება "$" ნიშანი.

მაგალითი 1:

	სწორია	არაა სწორი
რიცხვითი ტიპის ცვლადები:	N	И
(მთელი ან ნამდვილი)	x1	2z
	jami	x+y
მთელი ტიპის ცვლადები	n%	%n
	T&	10&
ნამდვილი ტიპის ცვლადები	y!	y!!
	dro#	a? #
სტრიქონული ტიპის ცვლადები	gvari$	name
	N$.	N g$

2.4. სტანდარტული ფუნქციები

კომპიუტერის მეშვეობით ამოცანის ამოხსნისას ხშირად გამოიყენება ელემენტარულ ფუნქციათა გამოთვლა. ამ ფუნქციებს, რომელთა გამოთვლის პროგრამები ყველა კომპიუტერს აქვს, სტანდარტული ფუნქციები ეწოდება. სტანდარტული ფუნქციების გამოსაყენებლად საჭიროა პროგრამაში სათანადო ადგილას ჩაიწეროს ფუნქციის შესაბამისი სტანდარტული პროგრამის სახელი, რასაც მოსდევს მრგვალ ფრჩხილებში მოთავსებული არგუმენტი. ქვემოთ მოყვანილია ზოგიერთი სტანდარტული ფუნქცია და მისი შესაბამისი განმარტება:

სტანდარტული ფუნქცია	სტანდარტული ფუნქციის განმარტება	
SIN(X)	sin(x)	(სინუსი)
COS(X)	cos(x)	(კოსინუსი)
TAN(X)	tg(x)	(ტანგენსი)

ATN(X)	arctg(x)	(არკტანგენსი)
SQR(X)	$\sqrt{x}$	(კვადრ. ფესვი)
EXP(X)	e^x	(სადაც e≈2,71828)
LOG(X)	lnx	(ნატურალური ლოგარითმი)
ABS(X)	|x|	(მოდული)
INT(X)	რიცხვის მთელი ნაწილის გამოყოფა	

სტანდარტული ფუნქციის არგუმენტი შეიძლება წარმოდგენილი იყოს რიცხვითი მუდმივით, ცვლადით, სხვა ფუნქციით ან არითმეტიკული გამოსახულებით.

ტრიგონომეტრიული ფუნქციის არგუმენტი გამოსახული უნდა იყოს რადიანებით:

$$1^\circ = \frac{\pi}{180}\text{rad.} \approx 0.017\text{rad.} \qquad 1\ \text{rad} = \frac{180^\circ}{\pi} \approx 57.3^\circ$$

სადაც $\pi \approx 3{,}141593$

ათობითი ლოგარითმის (lgx) გამოსათვლელად უნდა გამოიყენოთ მიღებული გარდაქმნა $\lg x = \frac{\ln x}{\ln 10}$ რაც ბეისიკზე შეესაბამება შემდეგ ჩანაწერს LOG(X)/ LOG(10).

III თავი. გამოსახულება

ბეისიკში განიხილავენ არითმეტიკულ, სტრიქონულ და ლოგიკურ გამოსახულებებს.

3.1. არითმეტიკული გამოსახულება

არითმეტიკულ გამოსახულებას წარმოადგენს მუდმივა, ცვლადი, სტანდარტული ფუნქცია და მათი ნებისმიერი კონსტრუქცია, რომლებიც ერთმანეთთან დაკავშირებულია არითმეტიკული ოპერაციების ნიშნებითა და აგრეთვე მრგვალი ფრჩხილებით.

არითმეტიკულ გამოსახულებებში გამოიყენება არითმეტიკული მოქმედებების შემდეგი სპეციალური ნიშნები:

+ შეკრება	/ გაყოფა
– გამოკლება	\ მთელნიშნა გაყოფა
* გამრავლება	^ ახარისხება

(და) გახსნილი და დახურული მრგვალი ფრჩხილები

მაგალითი 1:

რიცხვითი მუდმივების, ცვლადებისა და სტანდარტული მათემატიკური ფუნქციების კონსტრუქციები:

არითმეტიკული გამოსახულება	ბეისიკზე ჩანაწერი
$\frac{-b+\sqrt{b^2-4*a*c}}{2*a}$	(–b + SQR(b^2 – 4 * a * c)) / (2 * a)
$\ln\lvert\sin^2 x+\cos x^2\rvert$	LOG(ABS(SIN(x)^2 + COS(x^2)))
$\frac{tg(x+1)}{\sqrt[3]{y}-e^{-2z}}$	TAN(x+1) / (y^(1/3) – EXP(–2 * z))

არითმეტიკული გამოსახულების შედგენისას დაცული უნდა იყოს შემდეგი წესები:

1. არითმეტიკული გამოსახულება უნდა იყოს ჩაწერილი ერთ სტრიქონში.
2. არითმეტიკული გამოსახულების გამოთვლა წარმოებს მოქმედებათა (ოპერაციათა) პრიორიტეტის (შესრულების რიგის) მიხედვით:
   1) ფუნქციის მნიშვნელობის გამოთვლა;
   2) ფრჩხილებში მოთავსებული გამოსახულების გამოთვლა;
   3) ახარისხება;
   4) გამრავლება და გაყოფა;

5) შეკრება და გამოკლება.

მაგალითი 2: მოცემულია გამოსახულება:

$$8 + (21 - 9) \cdot \sin^2 30^\circ \Rightarrow 8 + (21 - 9) * \sin(30 * 0.017) \wedge 2$$

1) SIN(30 *0.017)=SIN(0.51)=0.5	8 + (21 - 9) * 0.5^2
2) 21 – 9 = 12	8 + 12*0.5^2
3) 0.5^2 = 0.25	8 + 12* 0.25
4) 12 *0 .25 = 3	8 + 3
5) 8 + 3 = 11	11

3. ერთნაირი პრიორიტეტის მქონე ოპერაციები სრულდება მოყვანილი მიმდევრობით, ე.ი. მარცხნიდან მარჯვნივ.

მაგალითი 3:

1) 45 – 20 + 15 = 25 + 15 = 40 და არა ასე

45 – 20 + 15 ≠ 45 – 35 = 10

2) 8 / 2 * 6 = 4 * 6 = 24 და აღნიშნავს (8/2)*6 და არა 8/(2*6) რასაც ბეისიკზე შეესაბამება 8/(2*6) = 8/12 = 2/3.

4. თუ ერთმანეთის მომდევნო ოპერაციები ახარისხებაა, მაშინ გამოთვლები სრულდება მარჯვნიდან მარცხნივ.

მაგალითი 4:

2^{3^2} ჩანაწერს ექნება შემდეგი ფორმა: $2\wedge(3\wedge2) = 2\wedge9 = 512$.

5. თუ გვაქვს რამდენიმე ერთმანეთის შემცველი ფრჩხილებიანი გამოსახულება, მაშინ გამოთვლები იწყება ყველაზე შიგა ფრჩხილებიდან.

მაგალითი 5:

$$d(a + b(c - g)) \Rightarrow d * (a + b * (c - g))$$

1) x1 = c * g	d * (a + b * x1)
2) x2 = b * x1	d * (a + x2)
3) x3 = a + x2	d * x3
4) x4 = d * x3	x4

6. ყველა გახსნილ ფრჩხილს უნდა შეესაბამებოდეს დახურული ფრჩხილი, ე.ი. რამდენი გახსნილი ფრჩხილიცაა გამოსახულებაში, იმდენივე უნდა იყოს დახურული ფრჩხილი.

3.2. სტრიქონული გამოსახულება

სტრიქონულ გამოსახულებას წარმოადგენს სტრიქონული მუდმივა, ცვლადი, ფუნქცია და მათი ნებისმიერი კონსტრუქცია, რომელიც შეიძლება იყვნენ დაკავშირებული კონკატენაციის (გაერთიანების) ოპერაციის ნიშნებით.

კონკატენაციის ოპერაცია გამოიყენება სტრიქონული გამოსახულებების გაერთიანებისათვის და ხორციელდება “+” ან “**&**” ნიშნებით.

მაგალითი:

« 2 » + « 2 » = « 22 »

« 50 » + « 50 » = « 5050 »

« 20 » & « 02 » = « 2002 »

« ლომი » + « ძე » მივიღებთ « ლომიძე »

თუ a$ = « ხე » და b$ = « ხილი »

მაშინ a$ + b$ = « ხეხილი »

« ნაძვის » & a$ = « ნაძვისხე »

თუ mamr$ = « ს ძე »

და mdedr$ = « ს ასული »

მაშინ « გიორგი » + mamr$ = « გიორგის ძე »

« ირაკლი » & mamr$ = « ირაკლის ძე »

« გიორგი » + mdedr$ = « გიორგის ასული »

jg$ = « 1025 »

gvari1$ = « კიკნაძე »

gvari2$ = « ჯავახიშვილი »

jg$ + « ჯგუფის სტუდენტი » + gvari1$ = « 1025 ჯგუფის სტუდენტი კიკნაძე »

jg$ + « ჯგუფის სტუდენტი » + gvari2$ = « 1025 ჯგუფის სტუდენტი ჯავახიშვილი »

3.3. ლოგიკური გამოსახულება

ლოგიკური გამოსახულებაა ლოგიკური მუდმივა, ცვლადი და მათი შესაძლო კონსტრუქცია. კონსტრუქციაში მუდმივები და ცვლადები დაკავშირებულია შედარებისა და ლოგიკური ოპერაციებით.

შედარება არის არითმეტიკულ, ლოგიკურ და სტრიქონულ გამოსახულებებს შორის თანაფარდობათა დადგენა. შედარების ოპერაციები **ბეისიკზე** წარმოდგენილია შემდეგი ნიშნებით:

= ტოლია,

> მეტია,

>= მეტი ან ტოლია,

< ნაკლებია,

<= ნაკლები ან ტოლია,

<> ტოლი არ არის.

მაგალითი 1:

5 > 3.1

-0.1 <= 0

დავუშვათ a = 4 და x = -3,

მაშინ a*x < a+x.

სტრიქონების შედარება სრულდება სტრიქონებში შემავალი, შესაბამის ადგილზე მდგომი სიმბოლოების ორობითი კოდების შედარებით მარცხნიდან მარჯვნივ. სხვადასხვა სიგრძის სტრიქონების შედარებისას წინასწარ ხდება მათი სიგრძეების გატოლება მცირე სიგრძის სტრიქონის მარჯვნივ პრობელების მიწერის საშუალებით.

მაგალითი 2:

ვთქვათ a$= « აბგ », b$= « აბგდ » და c$= « აბგ »,

მაშინ a$ = c$ სხვა სიტყვებით a$ და c$ სიმბოლური სტრიქონები ტოლია ან ეკვივალენტურია;

b$ > c$ ე.ი. სიმბოლური b$ სტრიქონი მეტია c$ სიმბოლურ სტრიქონზე, ვინაიდან « დ » ასოს კოდი მეტია « » (პრობელის) კოდზე, შესაბამისად « აბგდ » > « აბგ ».

ლოგიკურ გამოსახულებას შეუძლია მიიღოს შემდეგი მნიშვნელობებიდან ერთერთი :

ჭეშმარიტი (აღინიშნება T-თი (True), ლოგიკური « 1 »-ით) და

მცდარი (აღინიშნება F-ით (False), ლოგიკური « 0 »-ით).

მაგალითი 3:

2 > 1 ლოგიკური გამოსახულება ჭეშმარიტია,

-2 > -1 ლოგიკური გამოსახულება მცდარია,

« გივი » = « გია » ლოგიკური გამოსახულება მცდარია.

ბეისიკზე გამოიყენება შემდეგი ლოგიკური ოპერაციები:

NOT – ლოგიკური უარყოფა, « არა »,

AND – კონიუნქცია, ლოგიკური გამრავლება, « და »,

OR – დიზიუნქცია, ლოგიკური შეკრება, « ან »,

XOR – გამომრიცხავი **OR,**

EQV – ლოგიკური ეკვივალენტობა,

IMP – იმპლიკაცია.

ლოგიკური ოპერაცია **NOT** უნარულია და გამოიყენება ერთ ლოგიკურ გამოსახულებასთან მიმართებაში.

მაგალითი 4:

NOT a <= 0,

NOT nishani$ = « არადამაკმაყოფილებელი ».

დანარჩენი ლოგიკური ოპერაციები ბინარულია და გამოიყენება ორ ლოგიკურ გამოსახულებასთან მიმართებაში.

მაგალითი 5:

2 <= c*z AND c*z < 5,

a$ = «YES » OR a$ = « OK ».

ლოგიკური ოპერაციების გამოყენების წესი მოყვანილია შემდეგ ცხრილში

I ლოგიკური გამოსახულება	II ლოგიკური გამოსახულება	**NOT**	**AND**	**OR**	**XOR**	**EQV**	**IMP**
T	T	F	T	T	F	T	T
T	F	F	F	T	T	F	F
F	T	T	F	T	T	F	T
F	F	T	F	F	F	T	T

იმ შემთხვევაში, როდესაც გამოსახულებაში გამოყენებულია არითმეტიკული, შედარებისა და ლოგიკური ოპერაციები ისინი უნდა შესრულდნენ შემდეგი მიმდევრობით:

უნარული ოპერაცია **NOT**,

არითმეტიკული ოპერაციები,

შედარების ოპერაციები,

ბინარული ლოგიკური ოპერაციები.

ბინარულ ოპერაციათა შორის ყველაზე დიდი პრიორიტეტი აქვს **AND** ოპერაციას. დანარჩენი ლოგიკური ოპერაციები ტოლფასოვანია და სრულდება მარცხნიდან მარჯვნივ. ლოგიკურ ოპერაციათა სასურველი მიმდევრობის დადგენისათვის ლოგიკურ გამოსახულებებში შეიძლება მრგვალი ფრჩხილების გამოყენება.

მაგალითი 6:

გამოთვალეთ შემდეგი გამოსახულების მნიშვნელობა:

NOT (c > 0) OR a > 10 AND a / b^2 < 5

დავუშვათ, a=45, b=3, c=7 არითმეტიკული ოპერაციების შედეგები აღვნიშნეთ x1-ით, x2-ით, და ა.შ. ლოგიკური ოპერაციების შედეგები კი L1-ით, L2-ით, და ა.შ.

გამოთვლები შესრულდება შემდეგი მიმდევრობით:

1) c > 10 ==> L1 = F
2) **NOT** L1 ==> L2 = T
3) b^2 ==> x1 = 9
4) a/x1 ==> x2 = 5
5) a > 10 ==> L3 = T
6) x1 < 5 ==> L4 = F
7) L3 AND L4 ==> L5 = F
8) L2 OR L5 ==> L6 = T

IV თავი. QBasic სისტემის დაპროგრამების გარემო

4.1. QBasic სისტემის დაპროგრამების გარემოში შესვლა

QBasic სისტემის დაპროგრამების გარემოში შესასვლელად, შესაბამის საქაღალდეში უნდა მოვნიშნოთ **qbasic.exe** ფაილი და დავაჭიროთ **Enter** კლავიშს. ეკრანზე გამონათდება რედაქტირების ფანჯარა მუშაობის დაწყების თავსართით. თავსართის მოშორება შეიძლება **Esc** კლავიშზე დაჭერით. სუფთა ეკრანზე მარცხენა ზემო კუთხეში გამონათდება მოციმციმე კურსორი. ამის შემდეგ ხდება შესაძლო პროგრამის ტექსტის შეტანა ეკრანული რედაქტორის საშუალებით.

4.2. QBasic სისტემის ეკრანული რედაქტორი

კურსორის გადაადგილება პროგრამის ტექსტში.

კურსორის გადაადგილებას ემსახურება შემდეგი კლავიშები :

–—> ან <—– – გადაადგილება ხორციელდება პროგრამის ტექსტის ერთი პოზიციით შესაბამისი მიმართულებით,

↓ ან ↑ – გადაადგილება ხორციელდება პროგრამის ტექსტის ერთი სტრიქონით ქვემოთ ან ზემოთ,

Home – კურსორი გადაინაცვლებს სტრიქონის თავში,

End – კურსორი გადაინაცვლებს სტრიქონის ბოლოში,

Page Up – კურსორი გადაინაცვლებს ეკრანზე მოთავსებული პროგრამის ტექსტის წინა გვერდზე,

Page Down – კურსორი გადაინაცვლებს ეკრანზე მოთავსებული პროგრამის ტექსტის შემდეგ გვერდზე,

Ctrl+Home – კურსორი გადაინაცვლებს პროგრამის ტექსტის დასაწყისში,

Ctrl+End – კურსორი გადაინაცვლებს პროგრამის ტექსტის ბოლოში.

პროგრამის ტექსტის მონაკვეთის გამოყოფა

რედაქტორს საშუალება აქვს გამოყოს რამდენიმე გვერდით მდგომი სიმბოლო ან სტრიქონი. გამოყოფა ხორციელდება ერთდროულად **Shift** კლავიშზე დაჭერით და კურსორის გადაადგილებით შესაბამისი გადაადგილების კლავიშზე დაჭერით. გამოყოფილ მონაკვეთზე პროგრამის ტექსტის ასოები და ეკრანის ფონი იცვლის ფერს.

გამოყოფის გაუქმებისათვის საჭიროა კურსორის გადაადგილება ერთი პოზიციით ნებისმიერი მიმართულებით **Shift** კლავიშზე დაჭერის გარეშე.

პროგრამის ტექსტის შეტანა

პროგრამის ტექსტის სიმბოლოების შეტანა ხორციელდება კლავიატურით მოთავსებულ შესაბამის კლავიშებზე დაჭერით. პრობელის (სიმბოლოებს შორის დაშორების) შეტანა ხორციელდება განიერ, უწარწერო კლავიშზე დაჭერით. პროგრამის ტექსტის ახალ სტრიქონზე გადასვლა ხორციელდება **Enter** კლავიშზე დაჭერით. მოციმციმე კურსორი მიუთითებს მომხმარებელს ეკრანის იმ ადგილს, სადაც განთავსდება პროგრამის ტექსტის მორიგი სიმბოლო. ყოველი სიმბოლოს შეტანის შემდეგ, კურსორი ავტომატურად გადაადგილდება ერთი პოზიციით მარჯვნივ.

ეკრანის პოზიციის შემცველობის შეცვლას, რომელზეც მიუთითებს კურსორი, შეიძლება ძველი შემცველობის წაშლით, ახალი სიმბოლოს ჩასმით ან ძველი შემცველობის ახალი სიმბოლოთი შეცვლით.

წაშლა ხორციელდება შემდეგ კლავიშებზე დაჭერით.

Delete – იშლება სიმბოლო, რომელზეც მიუთითებს კურსორი.

Backspase – (← გრძელი ისარი) – იშლება სიმბოლო, რომელიც მოთავსებულია კურსორის მარცხნივ.

ჩასმისა და შეცვლის რეჟიმები გარეგნულად განსხვავდება ერთმანეთისაგან კურსორის ფორმით :

ჩასმისას კურსორს აქვს ხაზგასმის ნიშნის ფორმა, ხოლო შეცვლისას – შეფერადებული მართკუთხედის ფორმა.

გადართვა ერთი რეჟიმიდან მეორეში ხორციელდება **Insert** კლავიშზე დაჭერით.

ჩასმის რეჟიმში მუშაობისას კურსორით მითითებულ პოზიციაში თავსდება ახალი სიმბოლო, ხოლო ძველი და ყველა მას შემდეგ მდგომი სიმბოლო გადაინაცვლებს ერთი პოზიციით მარჯვნივ.

შეცვლის რეჟიმის დროს კურსორით მითითებულ პოზიციაში ძველი სიმბოლოს მაგივრად თავსდება ახალი სიმბოლო.

4.3. QBasic სისტემის მთავარი მენიუ

რედაქტორის ფანჯრის ზედა სტრიქონში განლაგებულია QBasic სისტემის მთავარი მენიუ. იგი შეიცავს შემდეგ მენიუებს:

File – პროგრამის ფაილთან მუშაობა;

Edit – პროგრამის ანუ სხვა ტექსტის რედაქტირება;

View – პროგრამის ანუ სხვა ტექსტის გადათვალიერება;

Search – კონტექსტური ძებნა და შეცვლა;

Run – პროგრამის შესრულება;

Debug – პროგრამის გამართვა;

Options – დამატებითი რეჟიმების მიცემა;

Help – დახმარების მიღება.

მთავარი მენიუს გამოძახება-გააქტიურება ხორციელდება **Alt** კლავიშზე დაჭერით. ცალკეული მენიუს გააქტიურება ხდება მისი მონიშვნით ← ან → გადაადგილების კლავიშით და <**Enter**> კლავიშზე დაჭერით, ან შესაბამისი გამოყოფილი ასოს (F, E, R და ა. შ.) კლავიშზე დაჭერით.

გააქტიურებული მენიუს ქვემოთ **Qbasic** სისტემა ხსნის ვერტიკალურად განლაგებულ შესაბამისი მენიუს საშუალებების ჩამონათვალს. ცალკეული საშუალების ამორჩევა ხორციელდება ისევე, როგორც ცალკეული მენიუს გააქტიურება, ოღონდ ← და → გადაადგილების ისრების ნაცვლად გამოიყენება ↑ და ↓ ისრები.

ქვემოთ მოცემულია მთავარი მენიუს საშუალებების ჩამონათვალი მათი დანიშნულების მოკლე აღწერით.

File მენიუ

New – ახალი ფაილის შექმნა;

Open – კომპიუტერის მეხსიერებაში ძველი ფაილის ჩატვირთვა winჩესტერიდან;

Save – მიმდინარე ფაილის შენახვა winჩესტერზე;

Save As – მიმდინარე ფაილის შენახვა winჩესტერზე ახალი სახელით;

Print... – მიმდინარე ფაილის დაბეჭდვა;

Exit – Qbasic სისტემიდან გამოსვლა.

Edit მენიუ

Cut (**Shift+Del**) – შლის გამოყოფილ ტექსტს და იღებს მის ასლს ბუფერში;

Copy (**Ctrl+Ins**) – ბუფერში იღებს გამოყოფილი ტექსტის ასლს;

Paste (**Shift+Ins**) – კურსორის მიმდინარე პოზიციიდან ათავსებს ბუფერის შემცველობას;

Clear (Del) – შლის გამოყოფილ ტექსტს;

New Sub... – იძლევა ახალი ქვეპროგრამის შექმნის საშუალებას;

New Function... – იძლევა ახალი ფუნქციის შექმნის საშუალებას.

View მენიუ

Subs (F2) – გამოაქვს ეკრანზე ჩატვირთული ქვეპროგრამები და ფუნქციები ;

Split – ყოფს ეკრანის სივრცეს ორ ნაწილად ;

Output Screen (F4) – გამოაქვს ეკრანზე ბოლო პროგრამის შესრულების შედეგი.

Search მენიუ

Find... – პოულობს ტექსტში სიმბოლოების მოცემულ მიმდევრობას;

Repeat Last Find (F3) – პოულობს წინასწარ დადგენილი სიმბოლოების მიმდევრობას პროგრამის ტექსტის დანარჩენ ნაწილში;

Change... – პოულობს და ცვლის სიმბოლოების ერთ მიმდევრობას მეორეთი.

Run მენიუ

Start (Shift+F5) – უშვებს შესრულებაზე ჩატვირთულ პროგრამას;

Restart – ასუფთავებს მონაცემებით დაკავებულ კომპიუტერის მეხსიერებას და ბრუნდება საწყის პოზიციაში;

Continue (F5) – აგრძელებს შეჩერებული პროგრამის შესრულებას.

V თავი. წრფივი სტრუქტურის ალგორითმის დაპროგრამება

5.1. ალგორითმის განმარტება

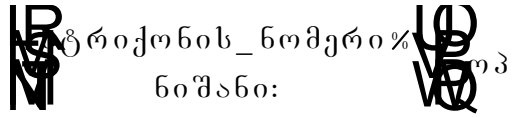
ალგორითმი არის მოქმედებების მიმდევრობა, რომელიც უნდა შესრულდეს რაიმე ამოცანის ამოსახსნელად.

ალგორითმის ბლოკ–სქემა არის ალგორითმის ნაწილების ბლოკებით (გეომეტრიული ფიგურებით) გამოსახვა.

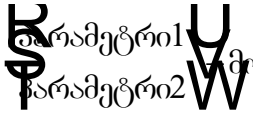
თუ ალგორითმში მოცემული ყველა მოქმედება მიყოლებით სრულდება ასეთ ალგორითმს **წრფივი** ალგორითმი ეწოდება.

5.2. ბეისიკ-ოპერატორის ზოგადი სახე

ბეისიკზე დაწერილი პროგრამა წარმოადგენს სტრიქონების მიმდევრობას, რომელსაც აქვთ შემდეგი სახე:



[ ] - არააუცილებელი მონაცემები



გამოყვანილ პარამეტრთა შორის ერთ-ერთი (პარამეტრი1 ან პარამეტრი2)

"_" – დასახელებაში გამაერთიანებელი ნიშანი

სტრიქონის_ნომერი% და ნიშანი: – მიუთითებს ბეისიკ-ოპერატორის ადგილს პროგრამის ტექსტში და ამით საშუალებას იძლევა მივმართოთ მათ პროგრამის სხვა ადგილიდან. სტრიქონის_ნომერი% წარმოადგენს მთელ დადებით რიცხვს, ნიშანი: – სიმბოლოების სტრიქონს.

ოპერატორი – ბეისიკ–ოპერატორის დასახელება იწერება დიდი ასოებით. ოპერატორი არის ბეისიკის ძირითადი ელემენტი, რომელიც მიუთითებს კომპიუტერს ამა თუ იმ მოქმედების შესრულების შესახებ.

პროგრამის თითოეული სტრიქონი შეიძლება შეიცავდეს ერთ ან რამდენიმე ოპერატორს. თუ სტრიქონი შეიცავს რამდენიმე ოპერატორს, ისინი უნდა განვაცალკეოთ " : " ნიშნით.

5.3. REM ან ' (აპოსტროფი) ოპერატორი

გამოიყენება პროგრამის ტექსტში კომენტარის ან შენიშვნის განთავსებლად. ოპერატორის ფორმატია :

კომენტარი ან შენიშვნა

შენიშვნები:

1) კომენტარი არ მონაწილეობს პროგრამის შესრულების პროცესში.

2) კომენტარი შეიძლება იყოს შედგენილი ბეისიკის ანბანის ნებისმიერი სიმბოლოებისაგან.

3) თუ REM (ან ') ოპერატორი გამოიყენება ერთ სტრიქონში სხვა ოპერატორებთან ერთად, ის იწერება სტრიქონის ბოლოში.

მაგალითი:

REM Jg. <ჯგუფის ნომერი> Kapanadze 1-14,

S = v * t : ' მანძილის გამოანგარიშება.

5.4. CLS ოპერატორი

მუშაობის პროცესში კომპიუტერი ეკრანზე არ შლის წინა პროგრამების მუშაობის შედეგებს. ამიტომ მორიგი პროგრამის გაშვებისას ეკრანზე შეიძლება დარჩენილი იყოს მანამდე გაშვებული პროგრამების მუშაობის შედეგები.

CLS ოპერატორი ასუფთავებს ეკრანს ყველა ძველი შეტყობინებისაგან და ათავსებს კურსორს მისი პირველი სტრიქონის პირველ პოზიციაში ანუ ეკრანის ზედა მრცხენა კუთხეში. ოპერატორის ფორმატია:

CLS

5.5. END *ოპერატორი*

აღნიშნავს ბეისიკზე დაწერილი პროგრამის დასასრულს. ოპერატორის ფორმატია :

END

გარდა ამისა, ის გამოიყენება IF...THEN...ELSE ოპერატორის, SELECT CASE ოპერატორისა და ქვეპროგრამების ბოლოში შემდეგი სახით:

END IF

END SELECT

END SUB

END FUNCTION

5.6. " = " (მინიჭება) ოპრატორი

ამოიყენება მაშინ, როდესაც საჭიროა ცვლადისათვის გამოსახულების მნშვნელობის მინიჭება. ოპერატორის ფორმატია:

S = G ,

სადაც S არის ცვლადის სახელი,

G – გამოსახულება.

შენიშვნა: ცვლადი და გამოსახულება უნდა იყოს ერთი და იმავე ტიპის მაგალითი:

სწორია	არასწორია
pi = 3.14	10 = x
e$ = “Epsilon”	a + b = c
n = n + 1	e = “Epsilon”
S = v * t	e$ = Epsilon
x1 = (-b + SQR(b^2 – 4*a*c))/(2*a)	e$ = .001

5.7. INPUT ოპერატორი

გამოიყენება კლავიატურიდან მონაცემის შესატანად პროგრამის შესრულებისას. ოპერატორის ფორმატია:

INPUT [" შეტყობინება "]  , ცვლადების_სია,

სადაც შეტყობინება წარმოადგენს სტრიქონულ მუდმივას, რომელიც გამოტანილი იქნება დისპლეიზე მონაცემების შეტანის წინ. მის შემდეგ მოყვანილი “ ; “ ნიშნავს რომ შეტყობინების შემდეგ დაიბეჭდება “ ? “. “ , “ – შემთხვევაში “ ? “ არ იწერება.

ცვლადების_სია – გამოყოფილი მძიმეების ერთი ან მეტი ცვლადი, რომელშიც იწერება კლავიატურიდან შეტანილი ცვლადების მნიშვნელობა.

შენიშვნები:

1) პროგრამის შესრულებისას ოპერატორი INPUT აკეთებს პაუზას იმისათვის რომ მომხმარებელს მიეცეს მონაცემების შეტანის საშუალება.
2) შეტანილი მონაცემების რიცხვი და ტიპი უნდა შეესაბამებოდეს INPUT ოპერატორში აღნიშნული ცვლადების რიცხვსა და ტიპს.

მაგალითი:

```
INPUT   a, b, c
INPUT   “x“; x
INPUT   “შეიტანეთ x ცვლადის მნიშვნელობა“, x
INPUT   “[ a; b ] “; a, b
```

INPUT “epsilon”; eps

INPUT “სახელი“; sakheli$

5.8. PRINT ოპერატორი

გამოიყენება მონაცემების გამოსატანად დისპლეიზე ან ფაილში. ოპერატორის ფორმატია:

PRINT [# ფაილის_ნომერი%,] [გამოსახულებათა_სია] [;] ,

სადაც ფაილის_ნომერი% ნიშნავს გახსნილი ფაილის ნომერს. თუ ეს ნომერი არ არის მოყვანილი, მაშინ მონაცემები იწერება დისპლეის ეკრანზე; გამოსახულებათა_სია – სია, რომელიც შეიცავს ერთ ან მეტ გამოსატან გამოსახულებას;

; – ნიშნავს მორიგი მონაცემის გამოტანას უშუალოდ ბოლო გამოსახულების მნიშვნელობის შემდეგ, თუ “; “ მოცემული არ არის მორიგი გამოსახულების მნიშვნელობა იბეჭდება შემდეგი სტრიქონიდან.

PRINT ოპერატორის გამოსახულებათა სიაში გამოიყენება TAB ფუნქცია. ის უთითებს კომპიუტერს TAB ფუნქციის შემდეგ გამოსახულების მნიშვნელობის ბეჭდვის საწყის პოზიციას. ფუნქციის ფორმატია:

TAB (n%) ,

სადაც n% არის ბეჭდვის საწყისი პოზიციის მთელი რიცხვითი მნიშვნელობა.

მაგალითი :

1) PRINT 1; 2; 3

   შედეგი:

   _1__2__3

2) PRINT “1“ ; “2“ ; “3“

   შედეგი:

   123

3)

```
PRINT 1; 2; 3
PRINT 4; 5; 6
შედეგი:
_1__2__3
_4__5__6
```

```
PRINT 1; 2; 3;
PRINT 4; 5; 6
შედეგი:
_1__2__3__4__5__6
```

4) PRINT -1; 2; -3

   შედეგი:

-1__2_-3

5) PRINT –1; TAB (6); 2; TAB (11); -3

შედეგი:
-1____2____-3

6) a = 25: b = 75
c = a + b
PRINT “a + b = “; c

a = 25: b = 75
PRINT “a+b“; a + b

შედეგი:
a + b = 100

7) v = 80
t = 3
s = v * t
PRINT ”S = “; S ; “კმ“

v = 80: t = 3
PRINT ”S = “;v * t; “კმ“

შედეგი:
S = _240_კმ

8) x = 300: y = 50000

PRINT TAB(4); “ x “; TAB(12); “ y “; TAB(20); “x+y“

PRINT TAB(2); x; TAB(9); y; TAB(19); x+y

შედეგი:

___x_____y_____x+y
__300____50300____50300

5.9. №1 ლაბორატორიული სამუშაოს შესრულების მაგალითი.

წრფივი სტრუქტურის ალგორითმის დაპროგრამება.

ლაბორატორიული სამუშაო 1–31

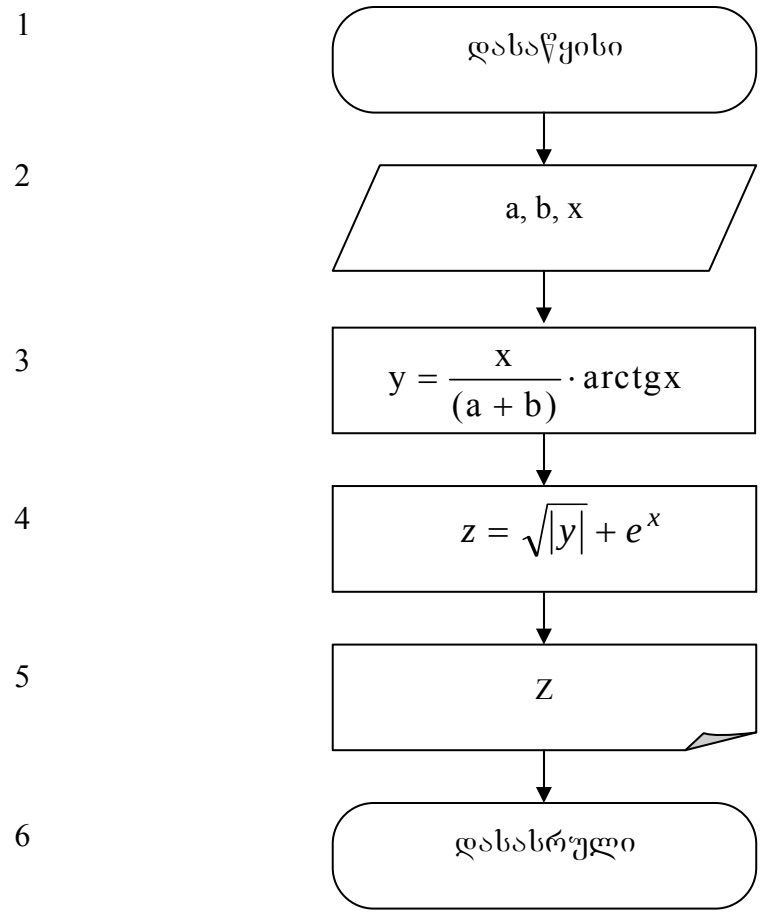
1. ამოცანის პირობა

გამოვთვალოთ $z=\sqrt{|y|}+e^{x}$ გამოსახულების მნიშვნელობა, თუ $y=\frac{x}{(a+b)}\cdot \mathrm{arctg}x$.

2. საწყისი მონაცემები

a = 10; b= -6; x = 1.8

3. ალგორითმის ბლოკ–სქემა



ბლოკ-სქემის აღწერა

ბლოკი 1 – ალგორითმის დასაწყისი

ბლოკი 2 – კლავიატურიდან საწყისი მონაცემების შეტანა კომპიუტერის მეხსიერებაში

ბლოკი 3 – y ცვლადის გამოთვლა

ბლოკი 4 – z ცვლადის გამოთვლა

ბლოკი 5 – z ცვლადის გამოტანა

ბლოკი 6 – ალგორითმის დასასრული

4. ბეისიკ–პროგრამა

```
REM  JG <ჯგუფის ნომერი> Kapanadze 1-31
OPEN  ”kap1-31.dat”  FOR  OUTPUT  AS  #1
```

```
CLS
INPUT   ”a”;  a
INPUT   “b”;  b
INPUT   ”x”;  x
y  =  x / (a + b) * ATN(x)
z  =  SQR(ABS(y)) + EXP(x)
PRINT  ”z  = ”; z
PRINT   #1,  ”z  = ”; z
CLOSE   #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
z  =  6.741503
```

VI თავი. განშტოებადი სტრუქტურის ალგორითმის დაპროგრამება

პირველ ლაბორატორიულ სამუშაოში განხილული იყო წრფივი სტრუქტურის ალგორითმის დაპროგრამება. ასეთ პროგრამაში ყოველი მომდევნო ოპერატორი სრულდება წინა ოპერატორის შესრულების შემდეგ. პრაქტიკაში ხშირად გვხვდება ისეთი ამოცანები, რომელთა ამოხსნისათვის საჭიროა განშტოებადი ალგორითმის აგება. შესაბამის პროგრამებში გამოიყენება უპირობო ან პირობითი გადასვლის (მართვის გადაცემის) ოპერატორი.

6.1. უპირობო გადასვლის GOTO ოპერატორი

GOTO-ს მეშვეობით ხდება გადასვლა მითითებულ სტრიქონზე. ოპერატორის ფორმატია:

GOTO ჭდე,

სადაც ჭდე არის შესაბამისი სტრიქონის ნომერი ან ნიშანი.

მაგალითი 1:

GOTO 90

6.2. პირობითი გადასვლის IF...THEN...ELSE ოპერატორი

ასრულებს ოპერატორის ან ოპერატორების ბლოკის შესრულებას დადგენილი პირობის შესაბამისად. ეს ოპერატორი შეიძლება იყოს ერთსტრიქონიანი და მრავალსტრიქონიანი. ერთსტრიქონიანის ფორმატი ასეთია:

IF პირობა THEN ოპერატორები [ELSE ოპერატორები]

მრავალსტრიქონიან ოპერატორს აქვს ასეთი ფორმატი:

```
IF   პირობა1   THEN
        [ოპერატორების_ბლოკი1]
[ELSEIF  პირობა2   THEN]
        [ოპერატორების_ბლოკი2]]...
[ELSE
        [ოპერატორების_ბლოკიn]]
END IF,
```

სადაც პირობა1, პირობა2 და ა.შ. არის ლოგიკური გამოსახულება, რომელმაც შეიძლება მიიღოს ერთ-ერთი მნიშვნელობა : ჭეშმარიტი – კი ან მცდარი – არა. ოპერატორები – ერთი ან მეტი ოპერატორი განლაგებული ერთ სტრიქონზე და განცალკევებული “ : “ ნიშნით

ოპერატორების_ბლოკი1
ოპერატორების_ბლოკი2

} ერთი ან მეტი ოპერატორი განლაგებული ერთ ან მეტ სტრიქონზე

...

ოპერატორების_ბლოკი

GOTO და IF...THEN...ELSE ოპერატორები ცვლიან პროგრამის სხვა ოპერატორების შესრულების მიმდევრობას. IF...THEN...ELSE ოპერატორი ამას აკეთებს გარკვეული პირობის შესაბამისად.

მაგალითი 2:

```
IF  y > 0  THEN  k = k + 1
IF  p >= 3.14  THEN  y = SIN(x)  ELSE  y = COS(x)
IF  b * x <= 9.8  THEN  z = SQR(p + x)  ELSE  z = (p + x) ^ (1 / 3)
IF  C > B  THEN  B = 0  ELSE  C = 0
IF  A * D < ABS(D – F)  THEN  min = A * D  ELSE  min = ABS(D – F)
IF  x < Y  THEN
        X = 0
        PRINT  “X = “; X,  “Y =”; y
ELSE
        Y = 0
        PRINT  “X = “; X,  “Y =”; y
END  IF
```

რთული ლოგიკური პირობები შეიძლება შედგენილი იყოს AND(და) და OR(ან) ლოგიკური ოპერაციების საშუალებით.

მაგალითი 3:

```
IF  x >= 2  AND  x < 10  THEN  y = k * x + b
IF  x < 0  OR  x >= 10  THEN  y = x ^ 2
IF  ANS$ = “Y”  OR  ANS$ = “OK“  THEN
        PRINT  “YES”
ELSE
        PRINT  “NO”
END  IF
```

6.3. №2 ლაბორატორიული სამუშაოს შესრულების მაგალითი

გაშტოებადი სტრუქტურის ალგორითმის დაპროგრამება

ლაბორატორიული სამუშაო 2–31

1. ამოცანის პირობა

იპოვეთ C + D და C * D სიდიდეებს შორის უდიდესი, თუ

$$C=\sqrt[11]{y^2+e^x},$$

$$D=\cos(x^2\cdot y+a\cdot b),$$

სადაც $y = b * x^3 + \ln|a|$.

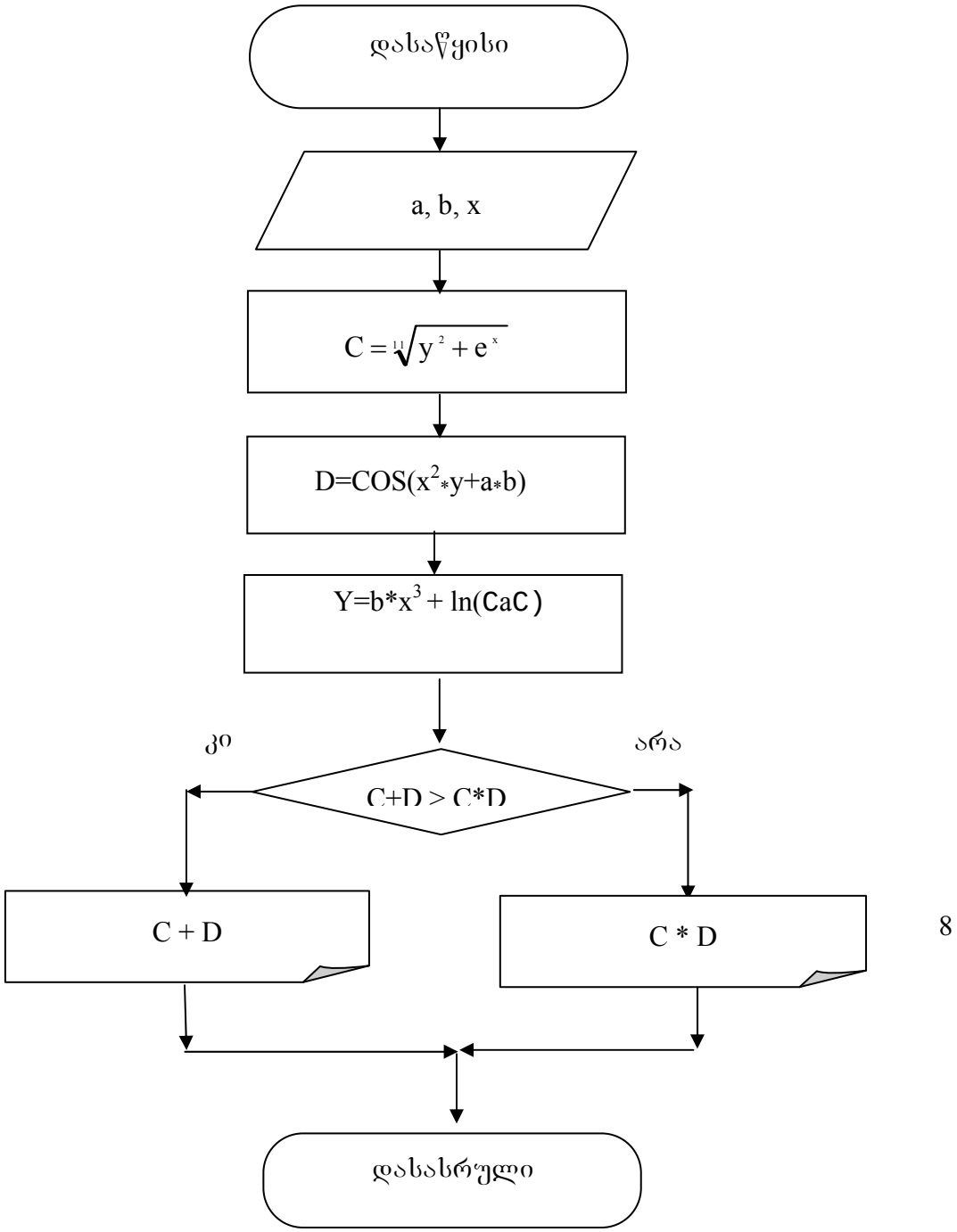
2. საწყისი მონაცემები

a = -6 ;

b = 0,2 ;

x = 3,7.

3. ალგორითმის ბლოკ–სქემა



ბლოკ-სქემის აღწერა

ბლოკი 2 – კლავიატურიდან საწყისი მონაცემების შეტანა

ბლოკი 3-5 – y, C და D ცვლადების მნიშვნელობების გამოთვლა შესაბამისი ფორმულებით

ბლოკი 6 – C+D > C*D პირობის შემოწმება.

პირობის დაკმაყოფილების შემთხვევაში შესრულდება მე-7 ბლოკის შესაბამისი მოქმედება,

წინააღმდეგ შემთხვევაში – მე-8 ბლოკის შესაბამისი მოქმედება.

ბლოკი 7 – C+D მნიშვნელობის გამოტანა

ბლოკი 8 – C*D მნიშვნელობის გამოტანა

4. ბეისიკ-პროგრამა

```
REM  JG. <ჯგუფის ნომერი>  Kapanadze 2-31
OPEN  ”kap2-31.dat”  FOR  OUTPUT  AS  #1
CLS
INPUT  ”a”;  a
INPUT  ”b”;  b
INPUT  ”x”;  x
y  =  b * x ^ 3 + LOG(ABS(a))
C  =  (y ^ 2  +  EXP(x)) ^ (1 / 11)
D  =  COS(x ^ 2 * y  =  a * b)
IF  C + D  >  C * D  THEN
        PRINT  ”C + D  =”;  C + D
        PRINT  #1,  ”C + D  =”;  C + D
ELSE
        PRINT  ”C * D  =”;  C * D
        PRINT  #1,  ”C * D  =”;  C * D
END IF
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
C + D  =  2.606421
```

ლაბორატორიული სამუშაო 2-32

1. ამოცანის პირობა

გამოიანგარიშეთ

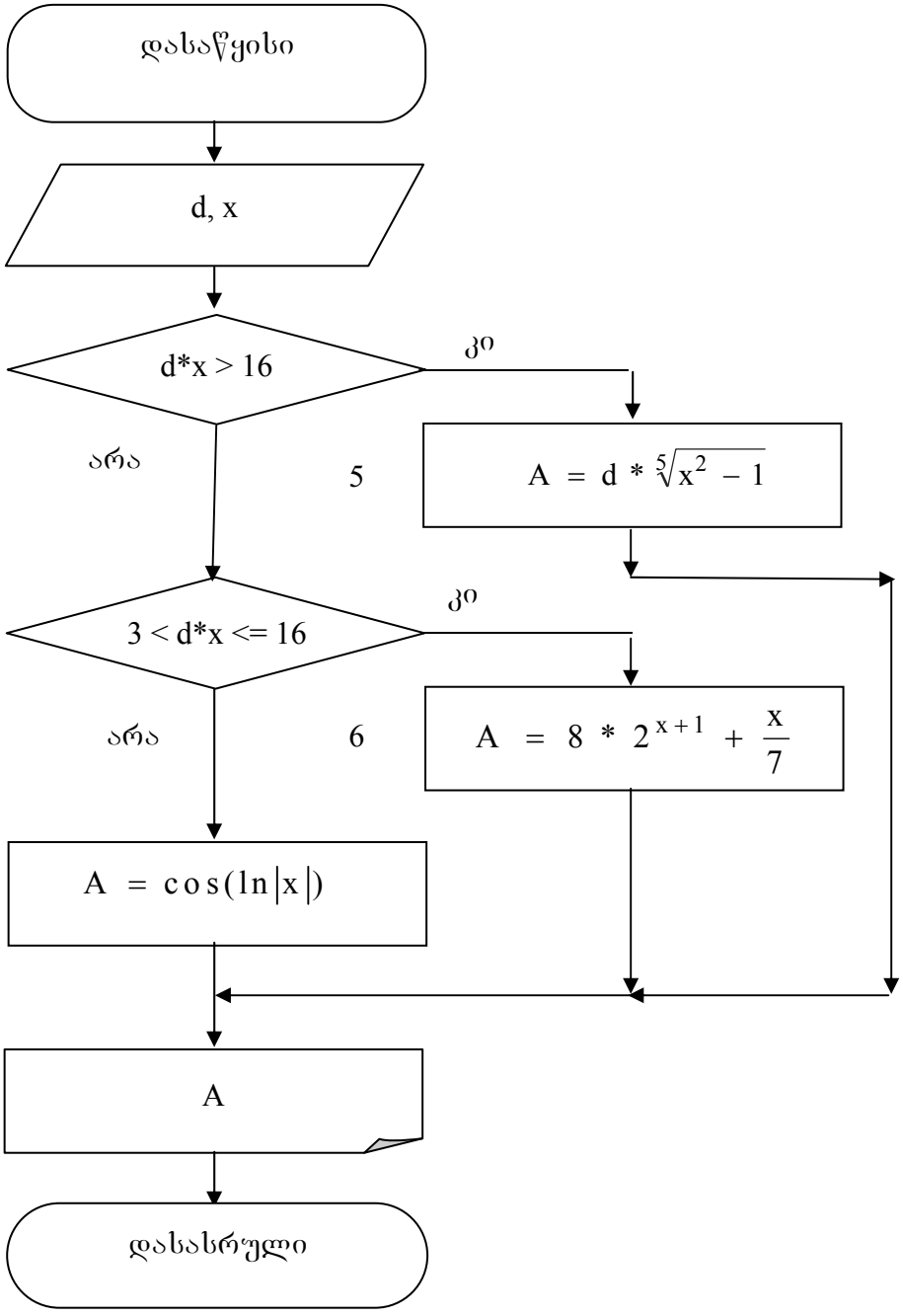
$$A = \begin{cases} d * \sqrt[5]{x^2 - 1}, & \text{როდესაც } d * x > 16 \\ 8 * 2^{x+1} + \frac{x}{7}, & \text{როდესაც } 3 < d * x \le 16 \\ \text{tosin}|x|, & \text{როდესაც } d * x \le 3 \end{cases}$$

2. საწყისი მონაცემები

d = 5,2 ;

x = -2,45

3. ალგორითმის ბლოკ-სქემა



ბლოკ-სქემის აღწერა

ბლოკი 2 – კლავიატურიდან d და x ცვლადების მნიშვნელობების შეტანა

ბლოკი 3 – d * x > 16 პირობის შემოწმება.

პირობის დაკმაყოფილების შემთხვევაში შესრულდება მე-5 ბლოკის შესაბამისი მოქმედება,

წინააღმდეგ შემთხვევაში – მე-4 ბლოკის შესაბამისი მოქმედება

ბლოკი 4 – $3 < d * x \leq 16$ პირობის შემოწმება.

პირობის დაკმაყოფილების შემთხვევაში შესრულდება მე-6 ბლოკის შესაბამისი მოქმედება,

წინააღმდეგ შემთხვევაში – მე-7 ბლოკის შესაბამისი მოქმედება.

ბლოკი 5 - 7 – A ცვლადის მნიშვნელობის გამოთვლა შესაბამისი ფორმულით

ბლოკი 8 – A ცვლადის მნიშვნელობის გამოტანა.

4 ბეისიკ-პროგრამა

```
REM  Jg. <ჯგუფის ნომერი>  Kapanadze 2-32
OPEN  ”kap2-32.dat”  FOR  OUTPUT  AS  #1
CLS
INPUT  ”d”; d
INPUT  ”x”; x
IF  d * x > 16  THEN
        A  =  d * (x ^ 2 – 1) ^ (1 / 5)
ELSEIF  d * x  >  3  AND  d * x  <=  16  THEN
        A  =  8 * 2 ^ (x  + 1)  +  x / 7
ELSE
        A  =  COS(LOG(ABS(x)))
END  IF
PRINT  ”A  =”;  A
PRINT  #1,  ”A  =”;  A
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

A = .6246692

VII თავი. ციკლური სტრუქტურის ალგორითმის დაპროგრამება

პრაქტიკული ამოცანის ამოხსნა ხშირად ხერხდება ერთი და იმავე მოქმედების (გამოთვლის) მრავალჯერ თანამიმდევრული განმეორებით. მაგალითად: გამოთვლები ერთი და იმავე ფორმულით ცვლადების სხვადასხვა მნიშვნელობისათვის. მსგავსი ამოცანის ამოხსნა შეიძლება იყოს აღწერილი ციკლური სტრუქტურის ალგორითმით.

ციკლს ეწოდება ალგორითმის (ანუ პროგრამის) ნაწილის მრავალჯერადი თანამიმდევრული განმეორება.

ციკლი შეიძლება იყოს მარტივი ან რთული. რთული ციკლი თავის მოქმედების არეში მოიცავს ერთ ან მეტ სხვა რთულ ან მარტივ ციკლს. განასხვავებენ არითმეტიკულ და იტერაციულ ციკლებს.

7.1. არითმეტიკული ციკლი

არითმეტიკულ ციკლში ამოცანის ამოხსნისათვის მოქმედებების (გამოთვლების) განმეორება ან მოცემულია, ან შეიძლება გამოვთვალოთ ამოცანის მონაცემების გამოყენებით. არითმეტიკული ციკლების დაპროგრამებისათვის ბეისიკში გათვალისწინებულია FOR...NEXT ოპერატორი. ოპერატორის ფორმატია:

FOR x = x1 TO xn [STEP h]

[ოპერატორების_ბლოკი]

NEXT x ,

სადაც x არის ციკლის მთვლელი, რომელიც წარმოადგენს რიცხვით ცვლადს და გამოიყენება ციკლის განმეორების რაოდენობის გამოსათვლელად.

x1, xn – ციკლის მთვლელის საწყისი და ბოლო მნიშვნელობები,

h – ციკლის მთვლელის მომატების ან დაკლების სიდიდე (ციკლის ბიჯი). თუ ბიჯი არ არის მითითებული ის ითვლება +1-ის ტოლად.

ოპერატორების_ბლოკი წარმოადგენს ციკლის ტანს და შეიცავს ერთ ან რამდენიმე ოპერატორს.

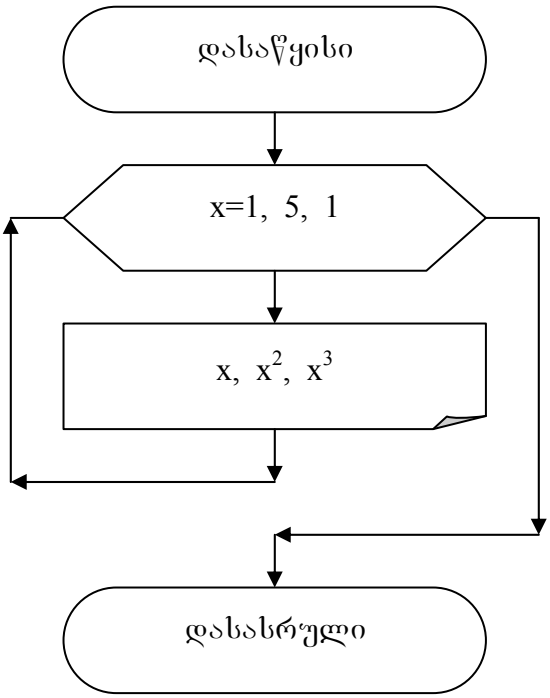
FOR ოპერატორის შესრულებისას გამოითვლება ციკლის პარამეტრები: ციკლის მთვლელის x1 საწყისი და xn საბოლოო მნიშვნელობები და ციკლის ბიჯი – h. x მთვლელს მიენიჭება x1 საწყისი მნიშვნელობა. ამის შემდეგ სრულდება ციკლის ტანში შემავალი ოპერატორები. NEXT ოპერატორის შესრულების შემდეგ x მთვლელის მნიშვნელობას დაემატება ციკლის ბიჯის მნიშვნელობა და

ხორციელდება ციკლის დამთავრების ანალიზი. ციკლი მეორდება მანამ, სანამ ციკლის მთვლელის x მნიშვნელობა არ გახდება მეტი (დადებითი ბიჯის შემთხვევაში) ან ნაკლები (უარყოფითი ბიჯის შემთხვევაში) მთვლელის საბოლოო მნიშვნელობაზე (xn-ზე)

მაგალითი 1:

გამოიანგარიშეთ და დაბეჭდეთ 1-დან 5-მდე მთელი რიცხვების კვადრატები და კუბები

ალგორითმის ბლოკ-სქემა



ბეისიკ-პროგრამა

```
FOR x = 1 TO 5
PRINT x, x^2, x^3
NEXT x
END
```

კომპიუტერზე მიღებული შედეგი

```
1         1         1
2         4         8
3         9         27
4         16        64
5         25        125
```

მთვლელის საწყისი მნიშვნელობა, ბოლო მნიშვნელობა და ბიჯი შეიძლება წარმოდგენილი იყოს ცვლადების ან არითმეტიკული გამოსახულებების მეშვეობით.

მაგალითი 2:

```
FOR   x = a   TO   b   STEP   h
FOR   x = A / 2   TO   b
FOR   x = A   TO   2 * B   STEP   h / 5
FOR   x = 10 * C   TO   D   STEP   -h
FOR   x =  -1.2   TO   2   STEP   0.3
FOR   i = 2   TO   10   STEP   2
FOR   j = i + 1   TO   m
FOR   i = 1   TO   n – 1
```

7.2. №3 ლაბორატორიული სამუშაოს შესრულების მაგალითები

არითმეტიკული ციკლური სტრუქტურის ალგორითმის დაპროგრამება

ლაბორატორიული სამუშაო N3-32

1. ამოცანის პირობა

მოცემულია $y = \frac{6*x}{x^2 - 1.5} + \sin x$ ფუნქცია, რომელიც გამოითვლება [-4.5;-1] სეგმენტზე, ბიჯი h=0,25. გამოთვალეთ ამ ფუნქციის 3-ზე ნაკლები მნიშვნელობების ნამრავლი და ამავე ფუნქციის უდიდესი მნიშვნელობა.

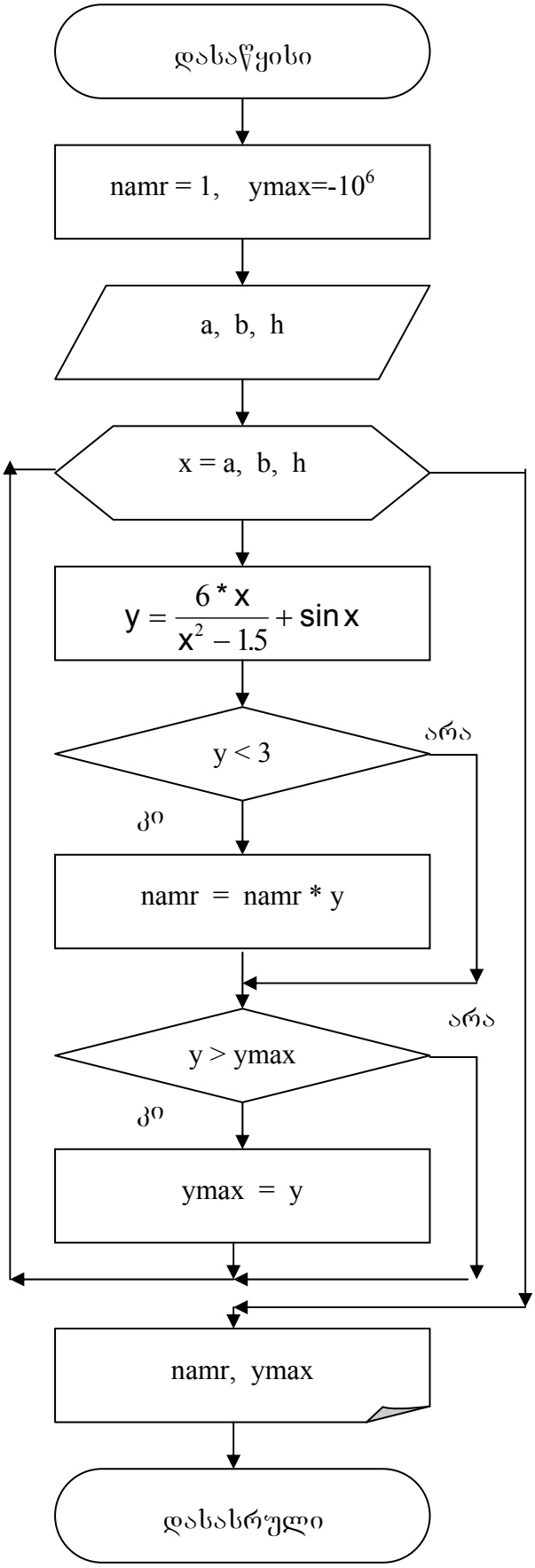
2. საწყისი მონაცემები

a = -4,5;

b = -1;

h = 0,25.

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ – პროგრამა

```
REM   Jg. <ჯგუფის ნომერი>   Kapanadze  3-32
OPEN   “kap3-32.dat”   FOR   OUTPUT    AS  #1
CLS
namr  = 1:  ymax  =  -1E + 06
INPUT   “a”; a
INPUT   “b”; b
INPUT   “h”; h
FOR    x  =  a   TO   b   STEP   h
        y  =  6 * x / (x ^ 2 – 1.5) +  SIN(x)
        IF   y < 3   THEN   namr  =  namr * y
        IF   y > ymax   THEN   ymax  =  y
NEXT   x
PRINT   “namravli  =”; namr,   “maximumi  =”; ymax
PRINT   #1,  “namravli  =”; namr,   “maximumi  =”; ymax
CLOSE   #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
        namravli  =  1.0007444E+07          maximumi  =  11.15853
```

ლაბორატორიული სამუშაო N 3 – 51

1. ამოცანის პირობა

მოცემულია $y = (3 - x^2 * \sqrt{|x|}) \cos x$ ფუნქცია, რომელიც გამოითვლება [-5; 5] მონაკვეთზე, ბიჯი h=0.1. გამოთვალეთ $z = e^R + e^{-R}$, სადაც R არის y ფუნქციის უარყოფითი მნიშვნელობების საშუალო არითმეტიკული.

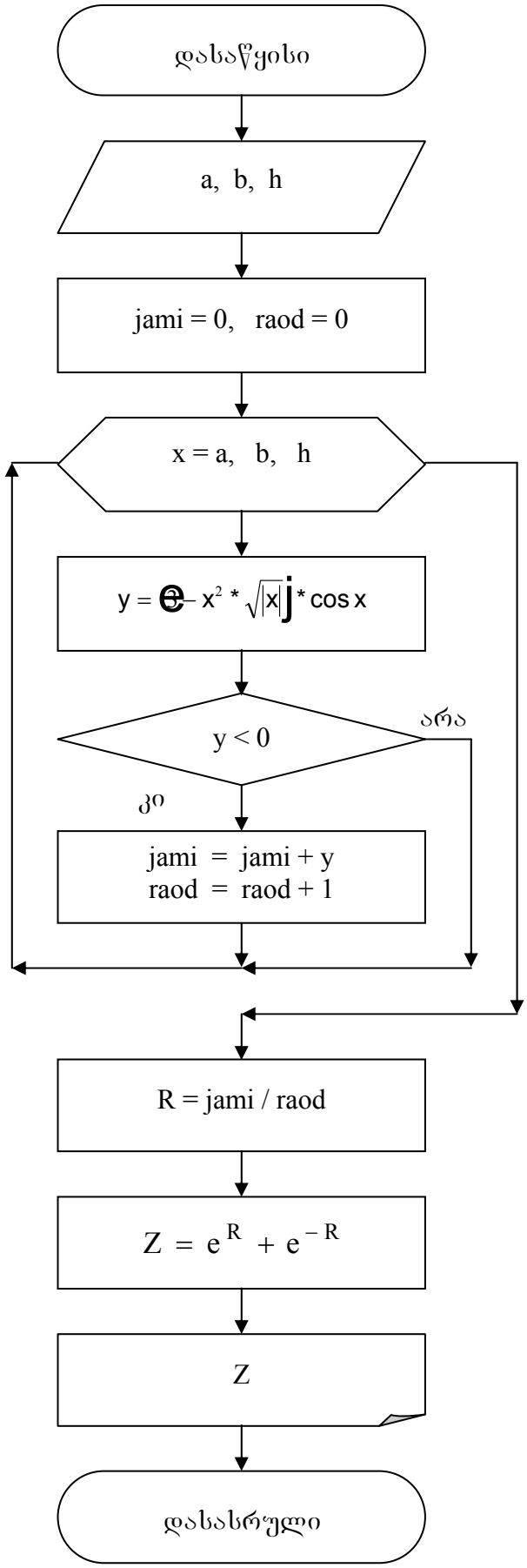
2. საწყისი მონაცემები

a = -5;

b = 5;

h = 0,1.

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ – პროგრამა

```
REM   Jg. <ჯგუფის ნომერი>   Kapanadze  3-51
OPEN   “kap3-51.dat”   FOR   OUTPUT    AS #1
CLS
INPUT   “ a “; a
INPUT   “ b “; b
INPUT   “ h “; h
jami  =  0
raod  =  0
FOR   x  =  a   TO   b   STEP   h
        y  =  ( 3 – x ^ 2 * SQR(ABS(x))) * COS(x)
        IF   y < 0   THEN
                jami  =  jami + y
                raod  =  raod + 1
        END  IF
NEXT   x
R  =  jami / raod
Z = EXP( R ) + EXP(-R)
PRINT   “Z =”; Z
PRINT   #1,   “Z =”; Z
CLOSE   #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
        Z  =  13419.63
```

7.3. იტერაციული ციკლი

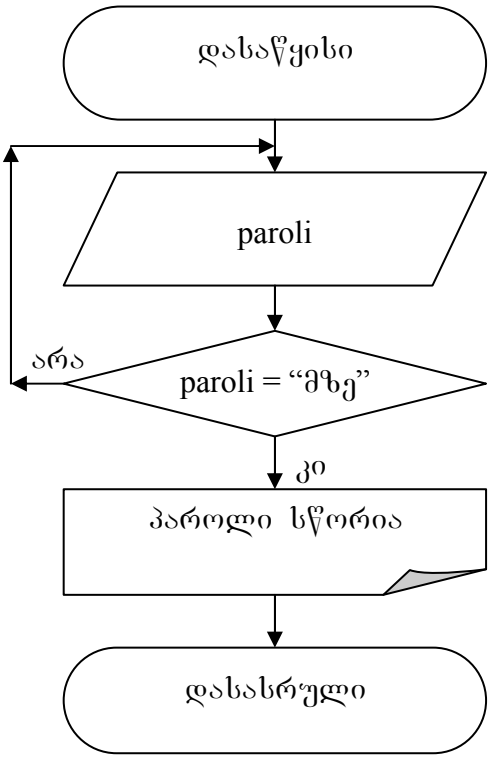
იტერაციულ ციკლში ამოცანის ამოხსნის მიღებისათვის გამოთვლების რაოდენობა დამოკიდებულია იმაზე, სრულდება ან შესრულდება თუ არა მოცემული პირობა.

ბეისიკში იტერაციული ციკლური ალგორითმებისათვის გამოიყენება ოპერატორი DO…LOOP (მარყუჟი). ეს ოპერატორი იმეორებს ოპერატორების ბლოკის შესრულებას მანამ, სანამ პირობა სრულდება, ამ შემთხვევაში გამოიყენება დამატებითი ოპერატორი WHILE, ან სანამ პირობა შესრულდება ამ შემთხვევაში გამოიყენება დამატებითი ოპერატორი UNTIL. ოპერატორის ფორმატი:

პირობა არის ლოგიკური გამოსახულება, რომელმაც შეიძლება მიიღოს ერთ-ერთი მნიშვნელობა: ჭეშმარიტი (კი) ან მცდარი (არა).

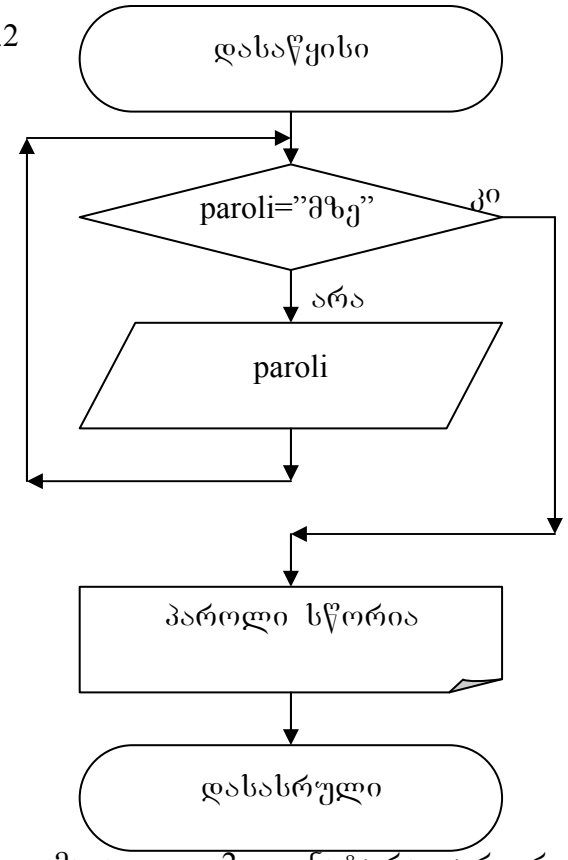
მაგალითი 1: პაროლის შემოწმება

1.1




```
REM  პაროლის შემოწმება
DO
   INPUT  შეიტანეთ პაროლი"; paroli$
LOOP  UNTIL  paroli$  =  "მზე"
PRINT  "პაროლი სწორია"
END
```

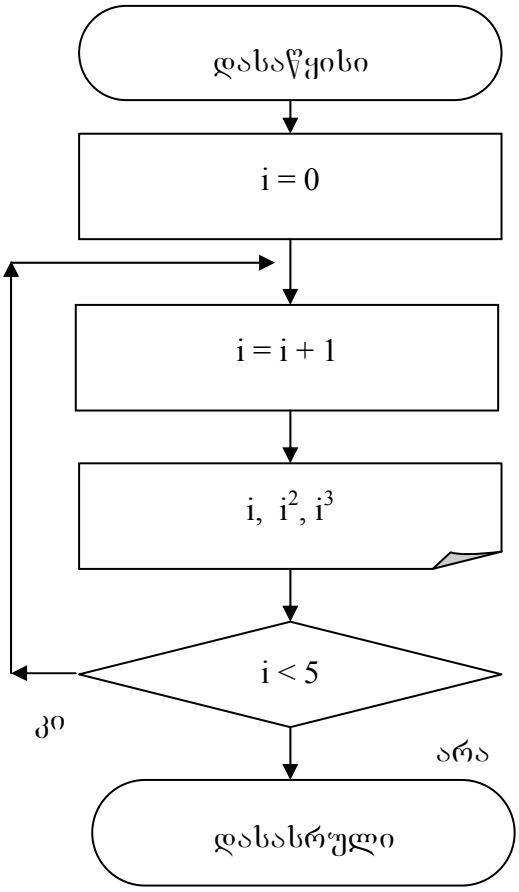
1.2




```
REM პაროლის შემოწმება
DO UNTIL  paroli$=მზე"
   INPUT "შეიტანეთ პაროლი"; paroli$
LOOP
PRINT "პაროლი სწორია"
END
```

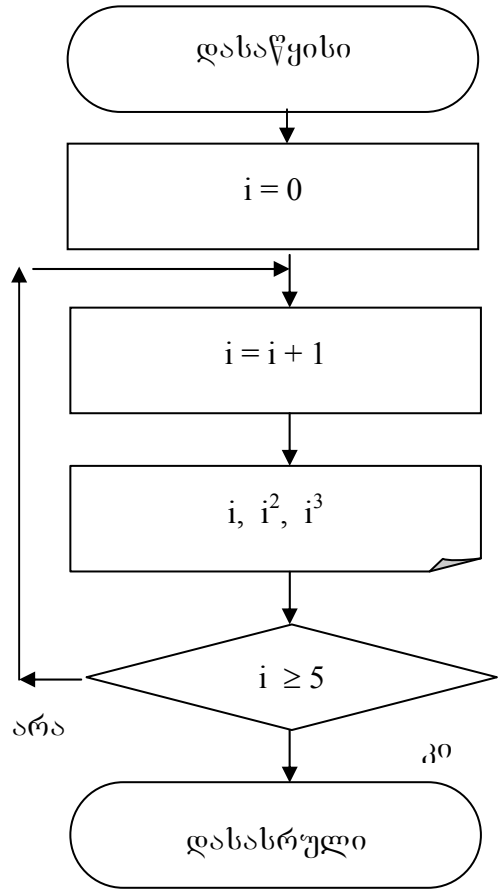
მაგალითი 2: ნატურალური რიცხვის კვადრატისა და კუბის გამოთვლა

2.1



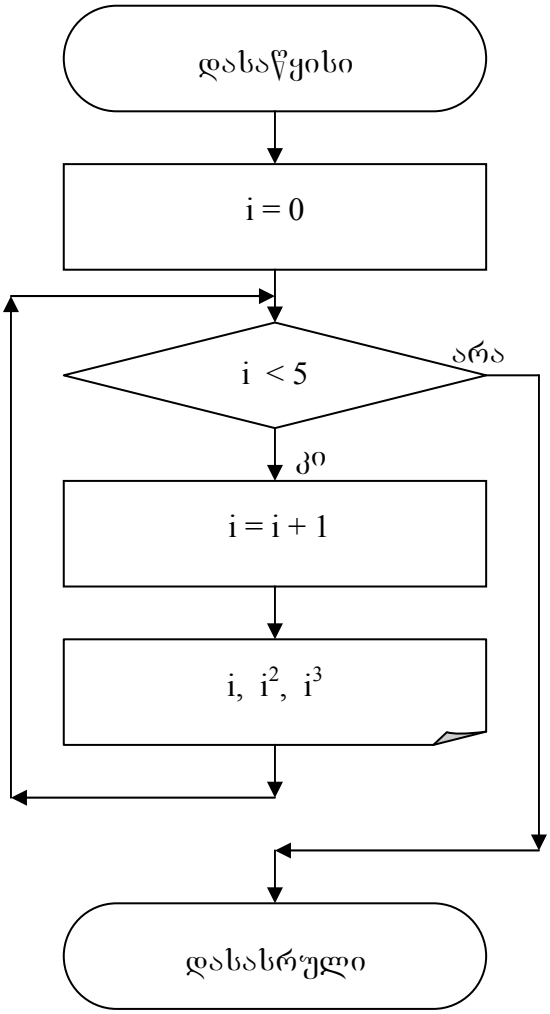

```
REM ნატურალური რიცხვის
REM კვადრატისა და კუბის
REM გამოთვლა
i = 0
DO
    i = i + 1
   PRINT  i, i ^ 2, i ^ 3
LOOP  WHILE  i < 5
END
```

2.2



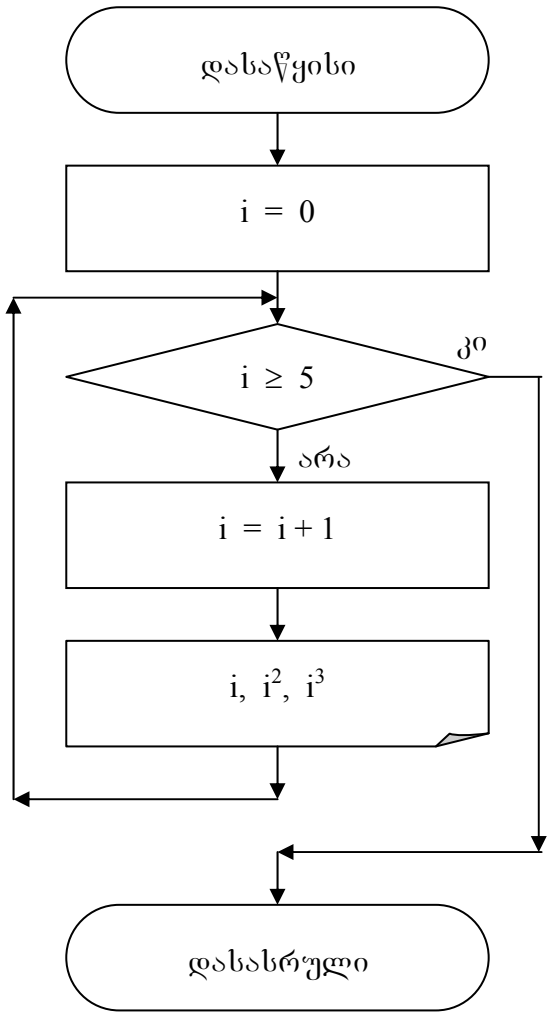

```
REM ნატურალური რიცხვის
REM კვადრატისა და კუბის
REM გამოთვლა
i = 0
DO
        i = i + 1
        PRINT  i, i ^ 2, i ^ 3
LOOP  UNTIL  i >= 5
END
```

2.3




```
REM  ნატურალური რიცხვის
REM  კვადრატისა და კუბის
REM  გამოთვლა
DO  WHILE  i < 5
        i = i + 1
        PRINT  i,  i ^ 2,  i ^ 3
LOOP
END
```

2. 4




```
REM  ნატურალური რიცხვის
REM  კვადრატისა და კუბის გამოთვლა
i = 0
DO UNTIL i >= 5
        i = i + 1
        PRINT  i, i^2, i^3
LOOP
END
```

2.1 და 2.3 პროგრამებში გამოიყენება WHILE ოპერატორი, რაც ნიშნავს რომ ციკლი მეორდება მანამ, სანამ სრულდება $i < 5$ პირობა. როგორც კი ეს პირობა დაირღვევა, ციკლის განმეორება მთავრდება. 2.2 და 2.4 პროგრამებში, სადაც გამოიყენება UNTIL ოპერატორი, პირიქით, ციკლის განმეორება მთავრდება, როგორც კი მოყვანილი $i \geq 5$ პირობა შესრულდება და მანამ, სანამ ეს პირობა არ სრულდება კომპიუტერი იმეორებს ციკლის ტანში მოყვანილ ოპერატორებს.

2.2 და 2.4 პროგრამებში პირობა მოწმდება ციკლის თავში, ე.ი. მანამ, სანამ დაიწყება ციკლის ტანში შემავალი ოპერატორების შესრულება. შეიძლება ისე მოხდეს, რომ ეს ოპერატორები არც ერთხელ არ შესრულდეს და პროგრამა გადავა ციკლის შემდეგ მოყვანილი ოპერატორის შესრულებაზე. 2.1 და 2.2 პროგრამებში პირობა მოწმდება ციკლის ბოლოში, ე.ი. ციკლის ტანში მოყვანილი ოპერატორები ერთხელ მაინც შესრულდება.

7.4. №4 ლაბორატორიული სამუშაოს შესრულების მაგალითები

იტერაციული ციკლური სტრუქტურის ალგორითმის დაპროგრამება

ლაბორატორიული სამუშაო № 4-55

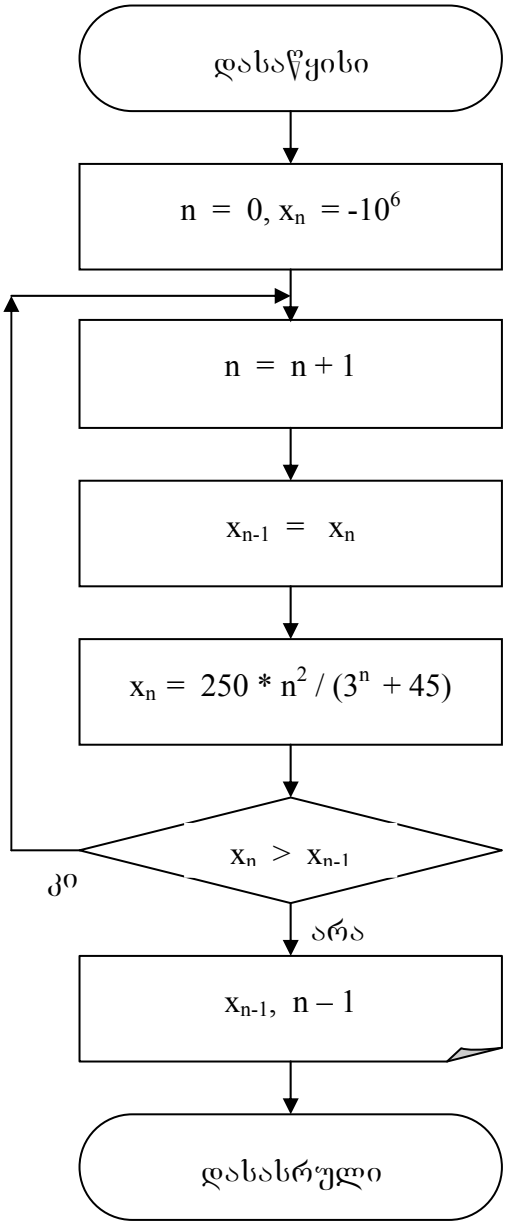
1. ამოცანის პირობა

მოძებნეთ $x_n = \frac{250 * n^2}{3^n + 45}$ $n = 1,2,...$ მიმდევრობის უდიდესი წევრი და მისი შესაბამისი n ნომერი. ფუნქცია ისეა შერჩეული, რომ შესაბამის მიმდევრობაში არის ერთადერთი უდიდესი წევრი.

2. საწყისი მონაცემები

—

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ – პროგრამა

```
REM  Jg. <ჯგუფის ნომერი> Kapanadze  4 – 55
OPEN  “kap4-55.dat”  FOR  OUTPUT  AS  #1
CLS
n = 0:  xn = -1E6
DO
        n = n + 1
        xn1 = xn
        xn = 250 * n ^ 2 / (3 ^ n + 45)
LOOP  WHILE  xn > xn1
PRINT  “xn =”;  xn1,  “n =”; n – 1
PRINT  #1,  “xn =“;  xn1,  “n =”;  n – 1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
xn = 31.74603                n = 4
```

ლაბორატორიული სამუშაო 4-31

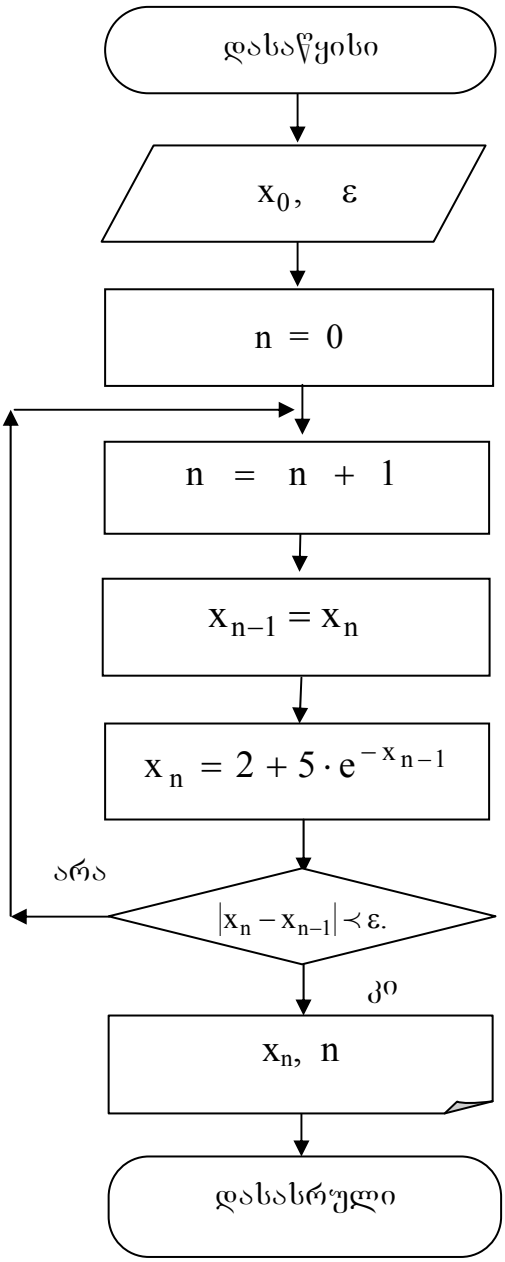
1. ამოცანის პირობა

იპოვეთ $x_n = 2 + 5 * e^{x_{n-1}}$ $(n = 1,2,\ldots)$ რეკურენტული ფორმულით მოცემული მიმდევრობის ისეთი x_n წევრი და შესაბამისი n ნომერი, რომ შესრულდეს $\left|x_n - x_{n-1}\right| < \varepsilon$ პირობა.

2. საწყისი მონაცემები

$x_0 = 1;\ \varepsilon = 0{,}001$

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ-პროგრამა

```
REM   Jg. <ჯგუფის ნომერი>  Kapanadze  4-31
OPEN  "kap4-31.dat"  FOR  OUTPUT  AS  #1
CLS
INPUT  "x0"; xn
INPUT  "epsilon" ;  eps
n  =  0
DO
        n  =  n + 1
        xn1  =  xn
        xn  =  2 + 5 * EXP(-xn1)
LOOP  UNTIL  ABS(xn - xn1) < eps
PRINT  "xn ="; xn,  "n  ="; n
PRINT  #1,  "xn ="; xn,  "n  ="; n
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

xn = 2.437293 n = 11

VIII თავი. მასივების დამუშავება

8.1. მასივები

მასივი ეწოდება ერთნაირი ტიპის ცვლადების ერთობლიობას (მიმდევრობას). ამ მიმდევრობის თითოეულ წევრს (ცვლადს) მასივის ელემენტი ან კომპონენტი ეწოდება. მასივის თითოეული ელემენტის მისათითებლად გამოიყენება ერთი ან რამდენიმე ინდექსი, რომელიც აღნიშნავს მასივის ელემენტის რიგით ნომერს. ინდექსი მოთავსებულია მრგვალ ფრჩხილებში და იწერება მასივის ელემენტის სახელის შემდეგ.

მაგალითი 1:

x(5) გამოსახავს x მასივის მეხუთე ელემენტს,

a(0) გამოსახავს a მასივის ნულოვან ელემენტს,

b(i) გამოსახავს b მასივის i-ურ ელემენტს,

t$(5) გამოსახავს სტრიქონული t$ მასივის მეხუთე ელემენტს.

მასივს, რომელიც ერთი ინდექსით გამოისახება, ერთგანზომილებიანი მასივი ანუ ვექტორი ეწოდება.

მასივი, შეიძლება ორი ინდექსით გამოვსახოთ. მაშინ მას ორგანზომილებიანი მასივი, ანუ მატრიცა ეწოდება. ამ შემთხვევაში ინდექსები ერთმანეთისაგან მძიმით გამოიყოფა.

პირველი ინდექსი განსაზღვრავს მატრიცის სტრიქონის ნომერს, ხოლო მეორე – სვეტის ნომერს.

მაგალითი 2:

a(4,2) გამოსახავს ორგანზომილებიანი a მასივის მე-4 სტრიქონის და მე-2 სვეტის გადაკვეთის ადგილზე მდგომ ელემენტს.

x(0,3) გამოსახავს ორგანზომილებიანი x მასივის ნულოვანი სტრიქონის და მე-3 სვეტის გადაკვეთის ადგილზე მდგომ ელემენტს.

პროგრამაში შეიძლება გამოყენებულ იქნას ცვლადი და მასივი, რომლებიც აღნიშნული არიან ერთი და იმავე სახელებით.

მაგალითი 3:

b არის მარტივი ტიპის ცვლადი,

b(6) – b მასივის მეექვსე ელემენტი

ინდექსის უმცირესი მნიშვნელობა ბეისიკზე ნულის ტოლია.

8.2 DIM (განზომილება) ოპერატორი

მასივი, რომელიც მონაწილეობს პროგრამაში, წინასწარ უნდა იქნას აღწერილი DIM ოპერატორით. ამ ოპერატორის საშუალებით ხორციელდება ოპერატიულ მეხსიერებაში რიცხვითი და სტრიქონული მასივებისათვის ადგილების გამოყოფა. ერთი DIM ოპერატორით შეიძლება აღწერილ იქნას რამოდენიმე მასივი.

მაგალითი 1:

DIM a(9)

DIM a(n,m)

DIM a(5,5), b(4,4), c(12)

მაგალითი 2:

DIM a(2,3) ოპერატორი განსაზღვრავს a მატრიცის მეხსიერებაში შემდეგ განლაგებას : a(0,0); a(0,1); a(0,2); a(0,3); a(1,0); a(1,1); a(1,2); a(1,3); a(2,0); a(2,1); a(2,2); a(2,3)

ეს შეესაბამება შემდეგ მატრიცას:

i \ j	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{23}	a_{23}	a_{23}	a_{23}

8.3 №5 ლაბორატორიული სამუშაოს შესრულების მაგალითები

ვექტორის დამუშავება

ლაბორატორიული სამუშაო №5-31

1. ამოცანის პირობა

გამოთვალეთ $A=(a_i)_{i=1,\ 2,\ \ldots,8}$ ვექტორის 2-ზე ნაკლები მნიშვნელობის მქონე კომპონენტების რაოდენობა

2. საწყისი მონაცემები

$a_1 = 5$

$a_2 = 3$

$a_3 = 1$

$a_4 = -1$

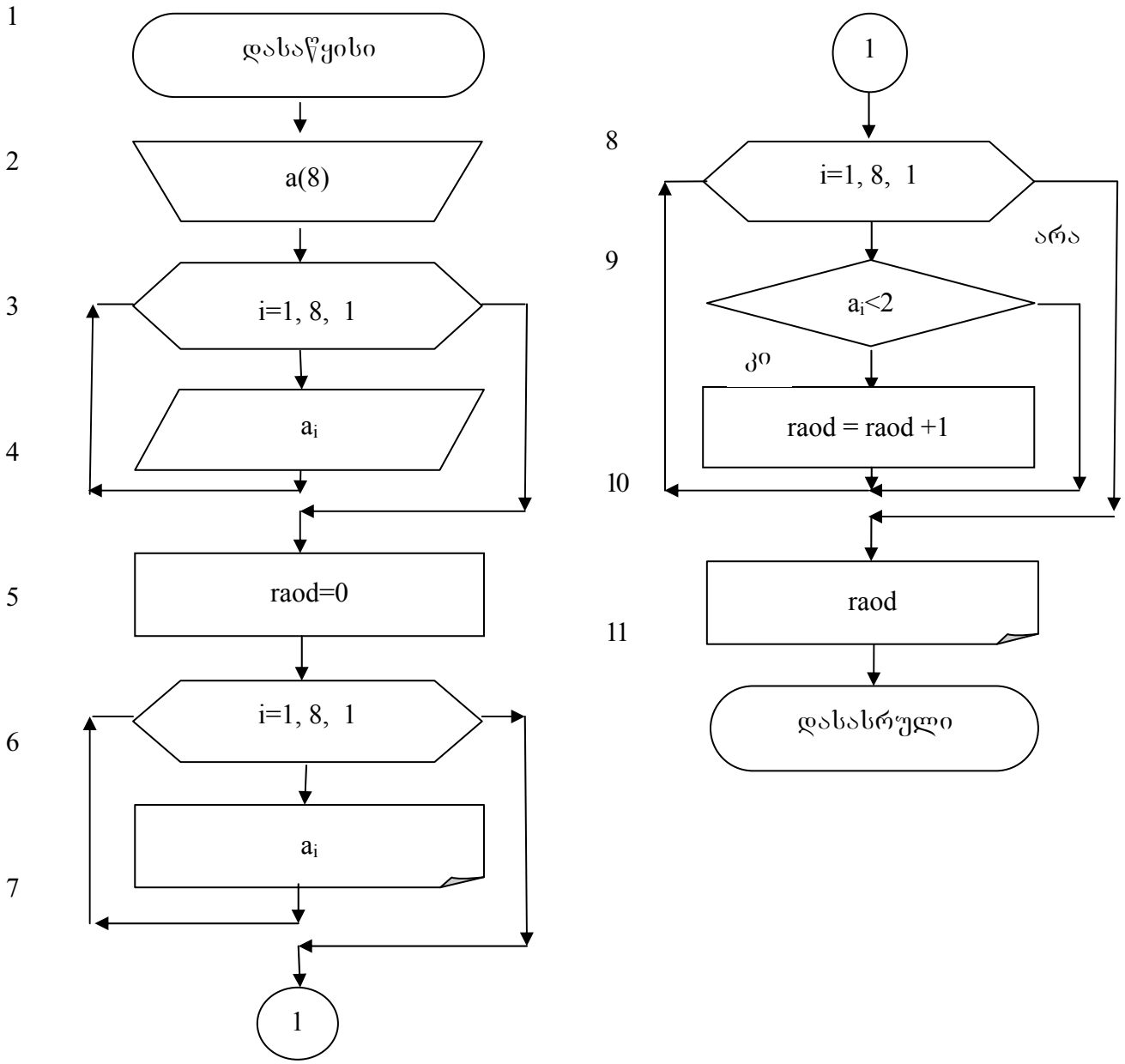
$a_5 = 0$

$a_6 = 0.5$

$a_7 = 2$

$a_8 = 2.7$

3. ალგორითმის ბლოკ-სქემა



ალგორითმის ზოგიერთი ბლოკის აღწერა

ბლოკი 2 – A ვექტორის ელემენტებისათვის ადგილის გამოყოფა კომპიუტერის მეხსიერებაში.

ბლოკი 4 – ციკლის დასაწყისი, სადაც ციკლის i ცვლადი აღნიშნავს A ვექტორის ელემენტის ინდექსს (ნომერს). ინდექსის საწყისი მნიშვნელობა ტოლია 1-ისა, ბოლო მნიშვნელობა – 8-სა, ხოლო ბიჯი ტოლია 1-ისა.

4. ბეისიკ-პროგრამა

```
REM  Jg. <ჯგუფის ნომერი>  Kapanadze  5-31
DIM  a(8)
OPEN  “kap5-31.dat”  FOR  OUTPUT  AS  #1
CLS
FOR  i  =  1  TO  8
        INPUT  “a(i) =”; a(i)
NEXT  i
raod  =  0
FOR  i  = 1  TO  8
        PRINT  #1, “a(i) =”; a(i)
NEXT  i
FOR  i  =  1  TO  8
        IF  a(i) < 2  THEN  raod  =  raod + 1
NEXT  i
PRINT  “raodenoba =”; raod
PRINT  #1,  “raodenoba =”; raod
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
raodenoba  =  4
```

ვექტორის დამუშავება

ლაბორატორიული სამუშაო №5-32

1. ამოცანის პირობა

მიიღეთ $B = (b_i)$ $i=1,2,...,9$ ვექტორის კომპონენტები $b_i = a_i + \sqrt{|R|}$ ფორმულით, სადაც R არის $A = (a_i)$ $i=1,2,..,9$ ვექტორის ელემენტებს შორის უმცირესი. დაბეჭდეთ ორივე ვექტორი.

2. საწყისი მონაცემები

$a_1 = 5$

$a_2 = 3$

$a_3 = 0$

$a_4 = 1$

$a_5 = -2$

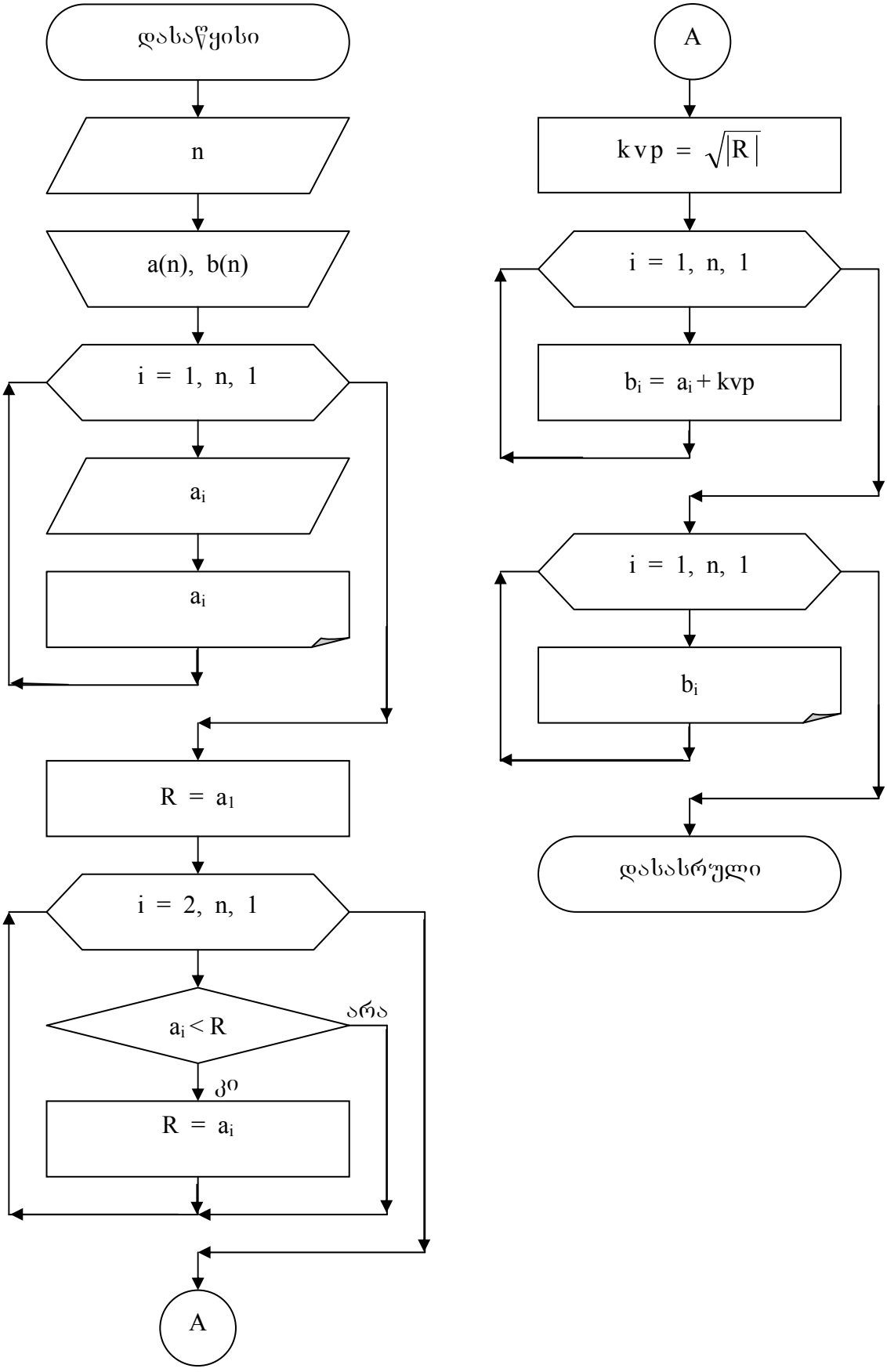
$a_6 = -1$

$a_7 = -4$

$a_8 = -1$

$a_9 = 2$

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ-პროგრამა

```
REM  Jg.<ჯგუფის ნომერი> Kapanadze 5-32
OPEN  “kap5-32.dat”  FOR  OUTPUT  AS  #1
CLS
INPUT  “n  =”; n
DIM  a(n),  b(n),
FOR  i  =  1  TO  n
        PRINT  “a(“;  i; ”) =”;
        INPUT  a(i)
        PRINT  #1,  “a(“;  i; ”) =”;  a(i)
NEXT  i
R  =  a(i)
FOR  i  =  2  TO  n
        IF  a(i) < R  THEN  R = a(i)
NEXT  i
KVP  =  SQR(ABS(R))
FOR  i  =  1  TO  n
        b(i)  =  a(i) + KVP
NEXT  i
FOR  i = 1  TO  n
        PRINT  “b(“;  i; ”) =”;  b(i)
        PRINT  #1,  “b(“;  i; ”) =”;  b(i)
NEXT  i
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

$a_1 = 5$

$a_2 = 3$

$a_3 = 0$

$a_4 = 1$

$a_5 = -2$

$a_6 = -1$

$a_7 = -4$

$a_8 = -1$

$a_9 = 2$

$b_1 = 5$

$b_2 = 3$

$b_3 = 0$

$b_4 = 1$

$b_5 = -2$

$b_6 = -1$

$b_7 = -4$

$b_8 = -1$

$b_9 = 2$

8.3 №6 ლაბორატორიული სამუშაოს შესრულების მაგალითები

მატრიცის დამუშავება

ლაბორატორიული სამუშაო №6-31

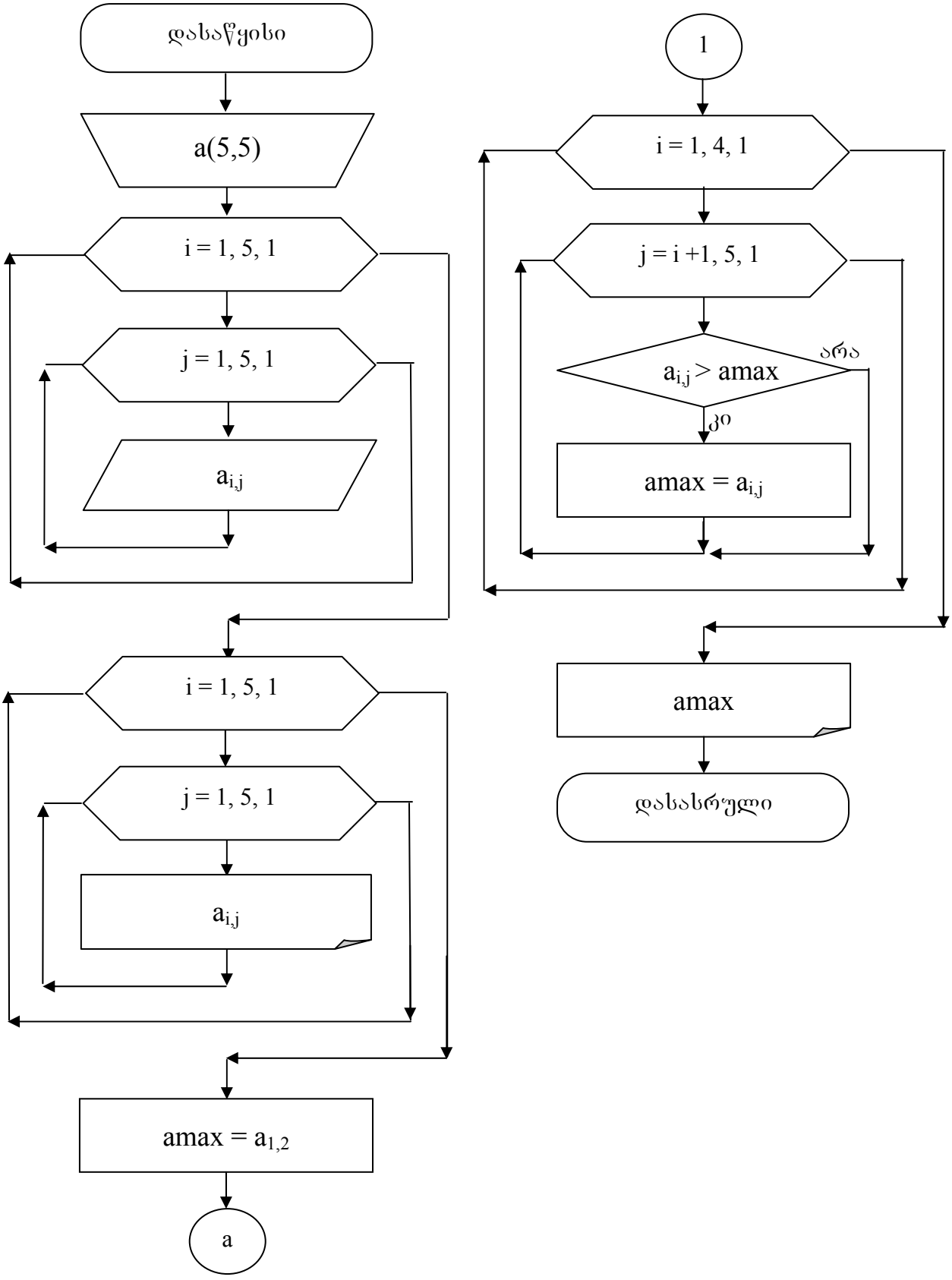
1. ამოცანის პირობა

იპოვეთ $A = (a_{i,j})$ $i,j=1,2,..,5$ მატრიცის მთავარი დიაგონალის ზემოთ მდებარე ელემენტებს შორის უდიდესი და დაბეჭდეთ იგი.

2. საწყისი მონაცემები

j / i	1	2	3	4	5
1	0	1	6	3	1
2	-5	2	5	0	3
3	9	-6	2	-7	-8
4	-1	4	1	4	2
5	8	3	9	2	-5

3. ალგორითმის ბლოკ-სქემა



4. ბეისიკ-პროგრამა

```
REM  Jg. <ჯგუფის ნომერი> Kapanadze 6-31
OPEN  “kap6-31.dat”  FOR  OUTPUT  AS  #1
CLS
DIM  a(5, 5)
PRINT  “A”
FOR  i = 1  TO  5
      FOR  j = 1  TO  5
            LOCATE  i + 3,  j * 8
            INPUT  a(i,  j)
      NEXT  j
NEXT  i
FOR  i = 1  TO  5
      FOR  j = 1  TO  5
            PRINT  #1,  a(i, j);
      NEXT  j
      PRINT  #1,
NEXT  i
amax  =  a(1,2)
FOR  i = 1  TO  4
      FOR  j = i + 1  TO  5
            IF  a(i, j) > amax  THEN  amax  =  a(i, j)
      NEXT  j
NEXT  i
PRINT  “amax  =”; amax
PRINT  #1,  “amax  =”; amax
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
      amax  =  6
```

ლაბორატორიული სამუშაო №6-32

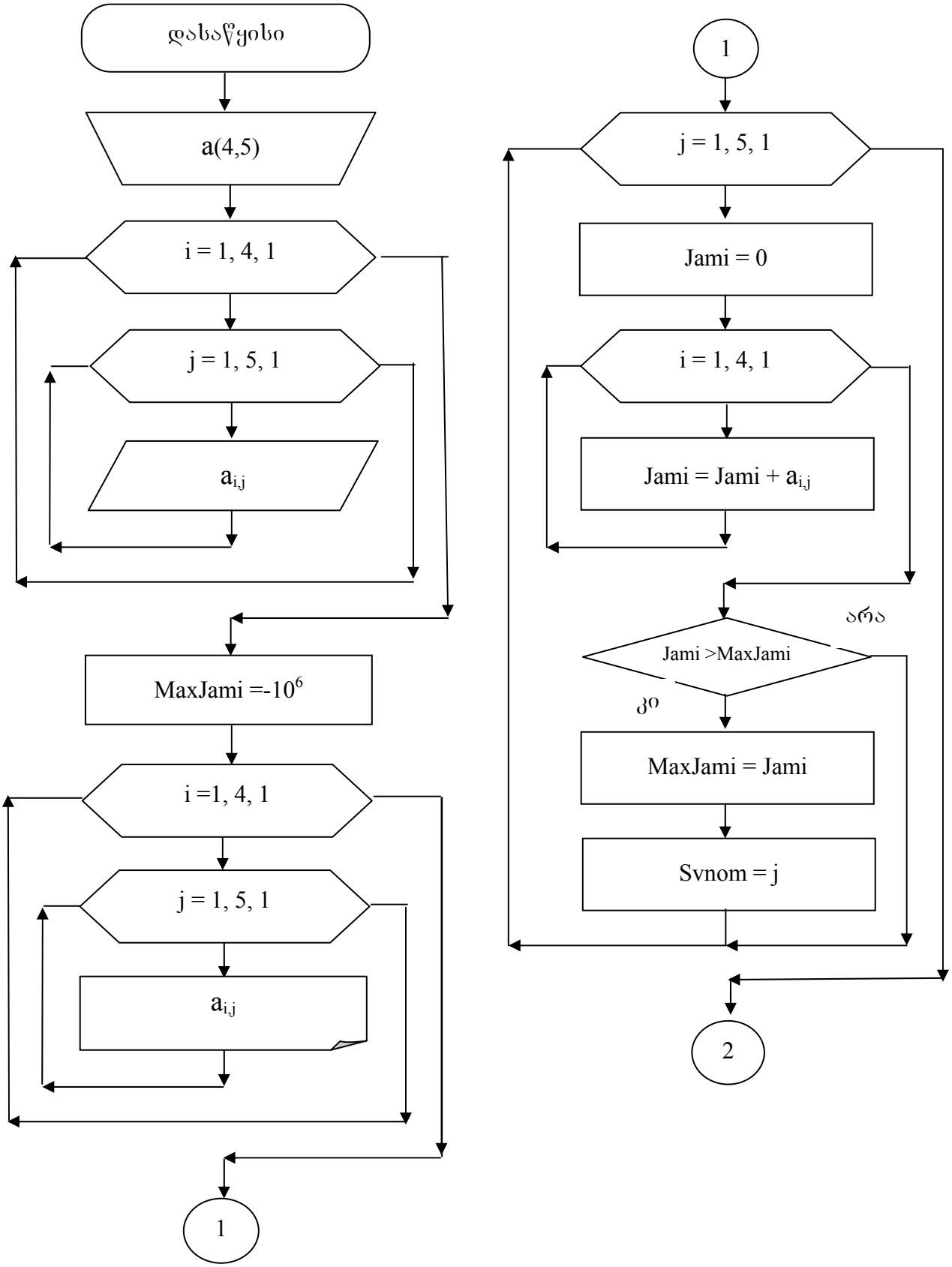
1. ამოცანის პირობა

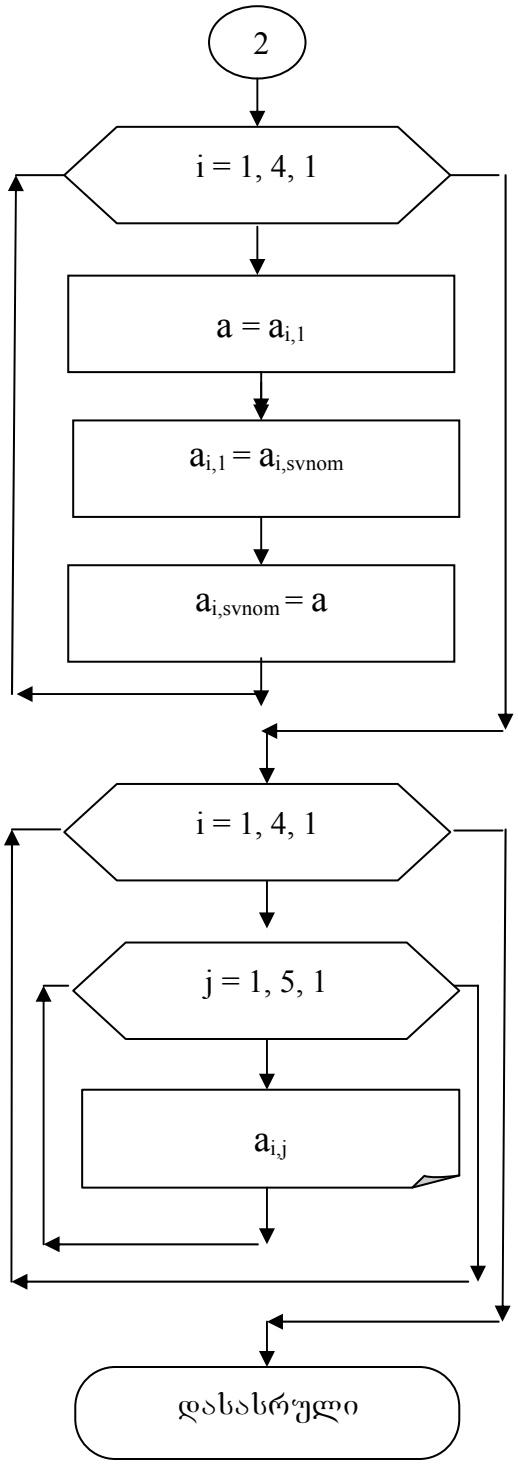
შეუცვალეთ ადგილები $A = (a_{i,j})$ $i = 1,2,..,4$, $j = 1,2,\ldots,5$ მატრიცის პირველ და მაქსიმალური ელემენტების ჯამის შემცველ სვეტებს. დაბეჭდეთ საწყისი და მიღებული მატრიცები.

2. საწყისი მონაცემები

i \ j	1	2	3	4	5
1	5	3	4	6	5
2	6	4	6	7	6
3	4	3	5	7	6
4	5	4	6	6	6

2. ალგორითმის ბლოკ-სქემა




2
i = 1, 4, 1
a = a_{i,1}
a_{i,1} = a_{i,svnom}
a_{i,svnom} = a
i = 1, 4, 1
j = 1, 5, 1
a_{i,j}
დასასრული

```
REM  Jg. <ჯგუფის ნომერი> Kapanaze 6-32
OPEN  "kap6-32.dat"  FOR  OUTPUT  AS  #1
DIM  a(4,5)
CLS
FOR  i = 1  TO  4
        FOR  j = 1  TO  5
                LOCATE  i + 2, j * 10
                INPUT  a(i, j)
        NEXT  j
NEXT  i
FOR  i = 1  TO  4
        FOR  j = 1  TO  5
                PRINT  #1,  a(i, j);
        NEXT  j
        PRINT  #1,
NEXT  i
MaxJami  = -1E6
FOR  j = 1  TO  5
        Jami  =  0
        FOR  i = 1  TO  4
                Jami  =  Jami + a(i, j)
        NEXT  i
        IF  Jami > MaxJami  THEN
                MaxJami  =  Jami
                svnom  =  j
        END  IF
NEXT  j
FOR  i = 1  TO  4
        a  =  a(i, 1)
        a(i, 1)  =  a(i, svnom)
        a(i, svnom)  =  a
```

```
NEXT  i
PRINT
PRINT  #1,
FOR  i = 1  TO  4
        FOR  j = 1  TO  5
                PRINT  a(i, j);
                PRINT  #1,  a(i, j);
        NEXT  j
        PRINT  #1,
        PRINT
NEXT  i
CLOSE  #1
END
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
5       3       4       6       5
6       4       6       7       6
4       3       5       7       6
5       4       6       6       6

6       3       4       5       5
7       4       6       6       6
7       3       5       4       6
6       4       6       5       6
```

IX თავი. დაპროგრამება ქვეპროგრამის გამოყენებით

ქვეპროგრამა წარმოადგენს სპეციალურად გაფორმებულ ოპერატორთა ჯგუფს, რომელზეც შესაძლებელია მრავალჯერადი მიმართვა. მიმართვა ხორციელდება მათი სახელების მიხედვით. განასხვავებენ ორი სახის ქვეპროგრამას პროცედურასა და ფუნქციას.

9.1. პროცედურა

პროცედურის ორგანიზებისათვის გამოიყენება შემდეგი ოპერატორები: DECLARE SUB, CALL, SUB … END SUB.

DECLARE SUB ოპერატორი განსაზღვრავს პროცედურას ძირითად პროგრამაში. ოპერატორს აქვს შემდეგი ფორმატი:

DECLARE SUB S(x1, x2, …, xn),

სადაც SUB აღნიშნავს პროცედურის განსაზღვრას, S პროცედურის სახელია, ხოლო x1, x2, …, xn - არგუმენტთა სია.

CALL ოპერატორი ახორციელებს პროცედურის გამოძახებას. მას აქვს შემდეგი ფორმატი:

CALL S(a1, a2, …, an),

სადაც S არის გამოძახებული პროცედურის სახელი, ხოლო a1, a2, … , an – არგუმენტთა სია.

SUB და END SUB ოპერატორები პროცედურის პირველი და ბოლო ოპერატორებია. მათ აქვთ შემდეგი ფორმატი:

SUB S(x1, x2, …, xn)

ოპერატორების_ბლოკი

END SUB,

სადაც S არის პროცედურის სახელი, x1, x2, …, xn – არგუმენტთა სია, ხოლო ოპერატორების_ბლოკი – ერთი ან რამდენიმე ოპერატორი.

ზემოთ მოყვანილ ოპერატორებში არგუმენტთა სიაში მოცემული არგუმენტთა მიმდევრობა უნდა იყოს ერთი და იგივე.

9.2. ფუნქცია

ფუნქციის ორგანიზებისათვის გამოიყენება ოპერატორები: DECLARE FUNCTION და FUNCTION … END FUNCTION.

DECLARE FUNCTION ოპერატორი განსაზღვრავს ფუნქციას ძირითად პროგრამაში. ოპერატორს აქვს შემდეგი ფორმატი:

DECLARE FUNCTION S(x1, x2, …, xn),

სადაც FUNCTION აღნიშნავს ფუნქციის განსაზღვრას. S არის ფუნქციის სახელი, ხოლო x1, x2, … ,xn – არგუმენტთა სია.

FUNCTION და END FUNCTION ოპერატორები ფუნქციის პირველი და ბოლო ოპერატორებია. მათ აქვთ შემდეგი ფორმატი:

FUNCTION S(x1,x2,…,xn)

ოპერატორების_ბლოკი

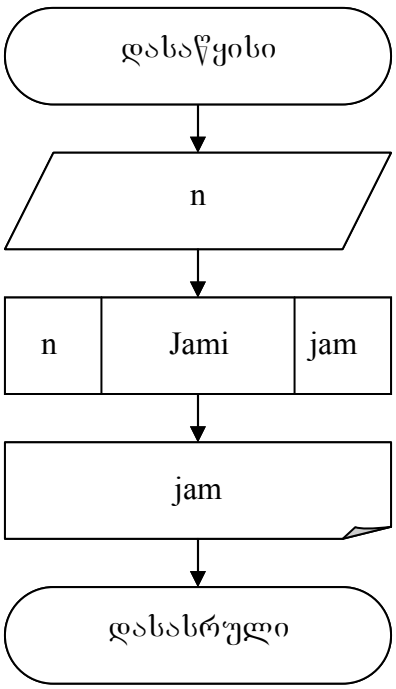
S = x

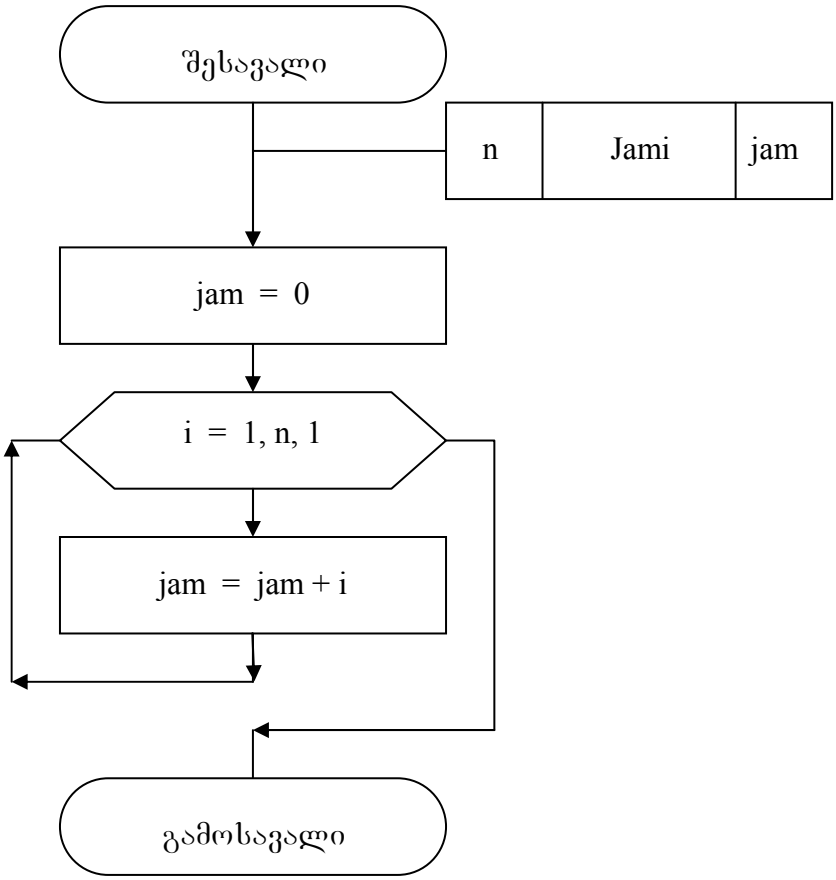
END FUNCTION,

სადაც S არის ფუნქციის სახელი, x – მათემატიკური გამოსახულება, ხოლო ოპერატორების_ბლოკი – ერთი ან რამდენიმე ოპერატორი, რომელიც ანგარიშობს x გამოსახულებაში შემავალ ცვლადებს.

მაგალითი 1: მოცემულია ნატურალური რიცხვები 1-დან n-მდე. გამოვთვალოთ ამ რიცხვების ჯამი პროცედურის გამოყენებით.

ალგორითმის ბლოკ-სქემა



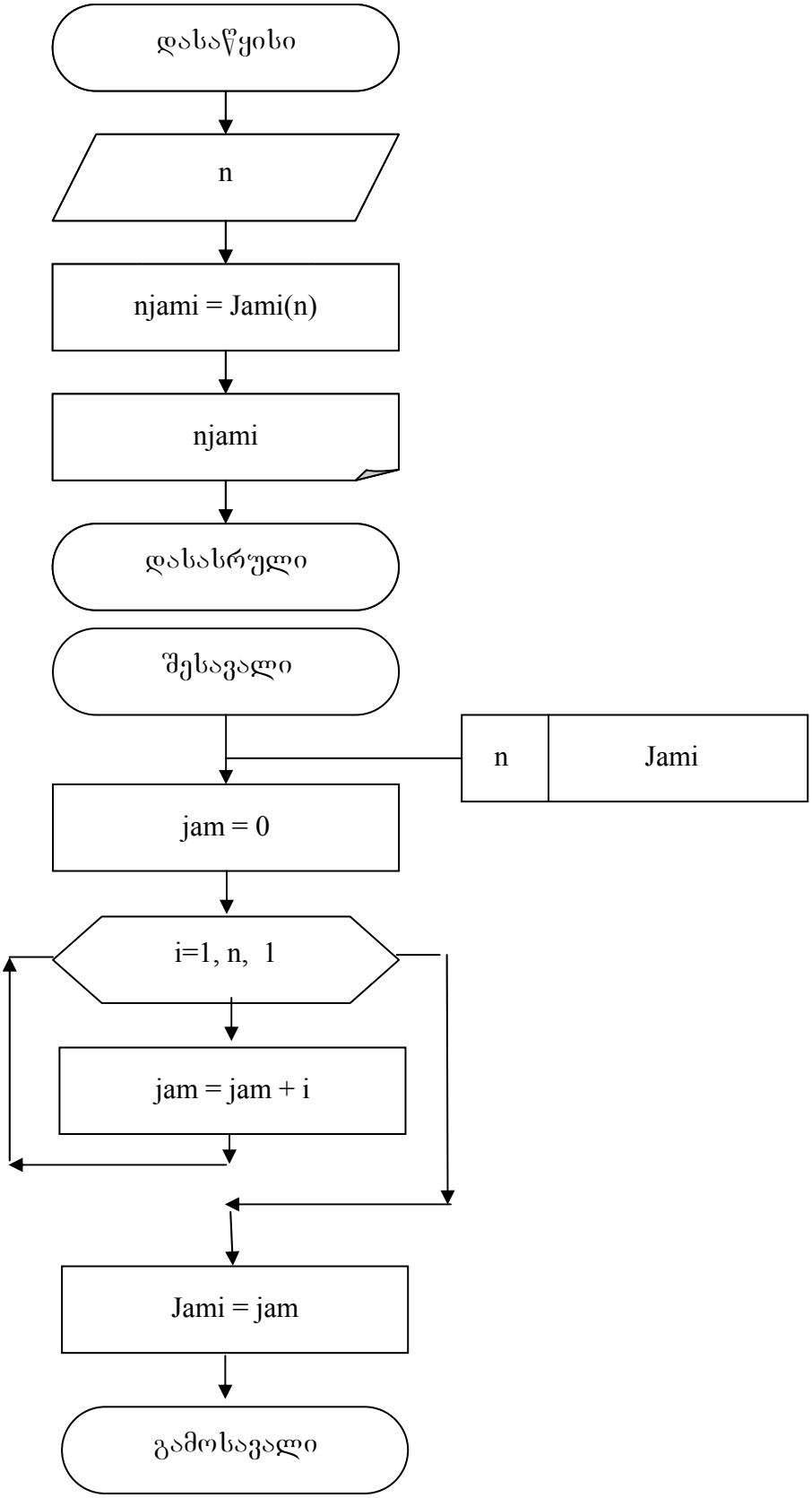


ბეისიკ-პროგრამა

```
REM   ნატურალური რიცხვების ჯამის გამოთვლა
DECLARE  SUB  Jami(n, jam)
CLS
INPUT  "n"; n
CALL  Jami(n, jam)
PRINT  "Jami  ="; jam
END
SUB  Jami(n, jam)
jam  =  0
FOR  i = 1  TO  n
        jam  =  jam + i
NEXT  i
END  SUB
```

მაგალითი 2: მოცემულია ნატურალური რიცხვები 1-დან n-მდე. გამოვთვალოთ ამ რიცხვების ჯამი ფუნქციის გამოყენებით.

ალგორითმის ბლოკ-სქემა



```
REM  ნატურალური რიცხვების ჯამის გამოთვლა
DECLARE  FUNCTION  Jami(n)
CLS
INPUT  "n"; n
njami  =  Jami(n)
PRINT  "Jami  ="; njami
END
FUNCTION  Jami(n)
jam  =  0
FOR  i  = 1  TO  n
        jam  =  jam + i
NEXT  i
Jami  =  jam
END  FUNCTION
```

9.3 №7 ლაბორატორიული სამუშაოს შესრულების მაგალითები

დაპროგრამება ქვეპროგრამის გამოყენებით

ლაბორატორიული სამუშაო №7-34

1. ამოცანის პირობა

მოცემულია $A=(a_{i,j})$ i,j=1,2,…,5 და $B=(b_{i,j})$ i,j=1,2,…,4 მატრიცები. გამოთვალეთ

$$Z = \frac{p^{ka} \cdot t^{kb}}{e^{sa+sb}},$$

სადაც p და t მოცემული რიცხვებია, ხოლო sa, ka და sb, kb, შესაბამისად A და B მატრიცების დადებითი ელემენტების ჯამი და რაოდენობაა.

sa, ka და sb, kb რიცხვების გამოთვლის პროგრამა გააფორმეთ ქვეპროგრამის სახით.

2. საწყისი მონაცემები

p = 2
t = 3

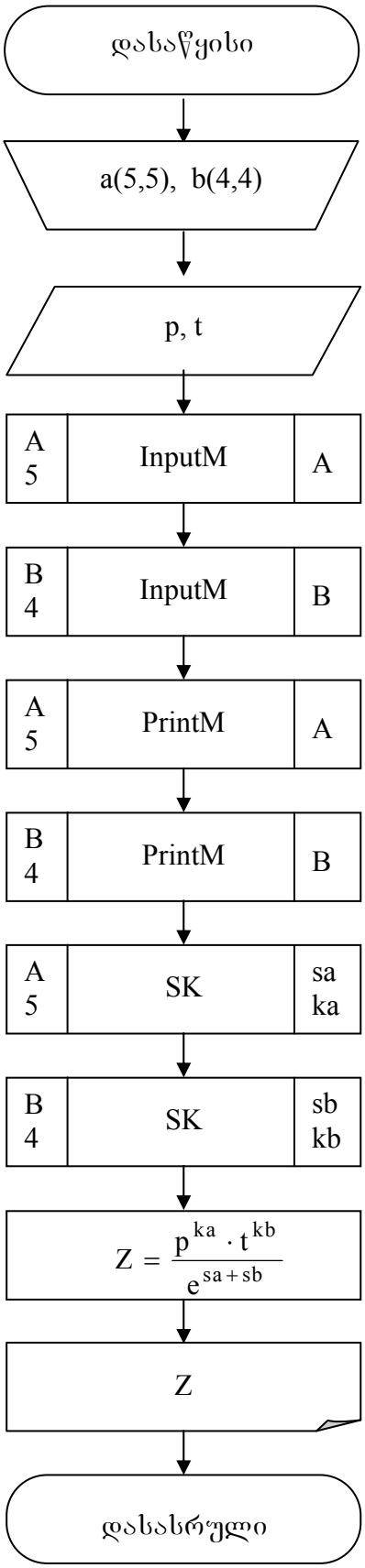
A მატრიცა

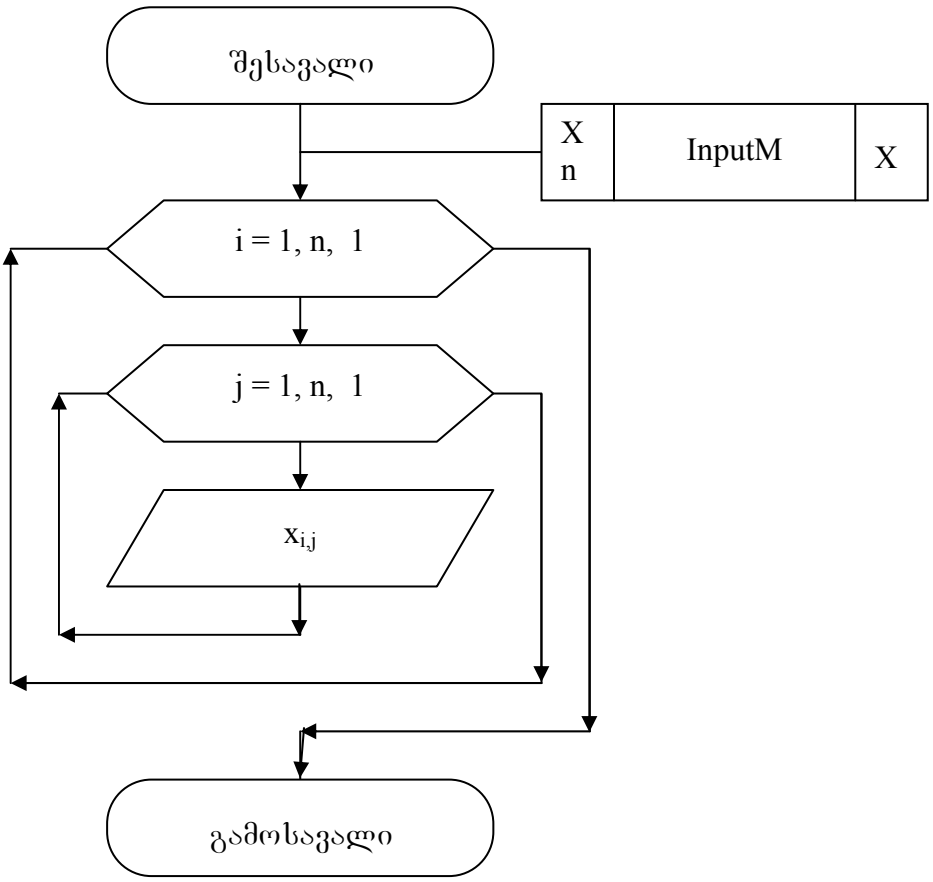
5	-7	-8	3	1
2	3	-1	-4	-6
-9	-3	2	0	6
0	4	-4	2	5
-5	-3	1	-3	-8

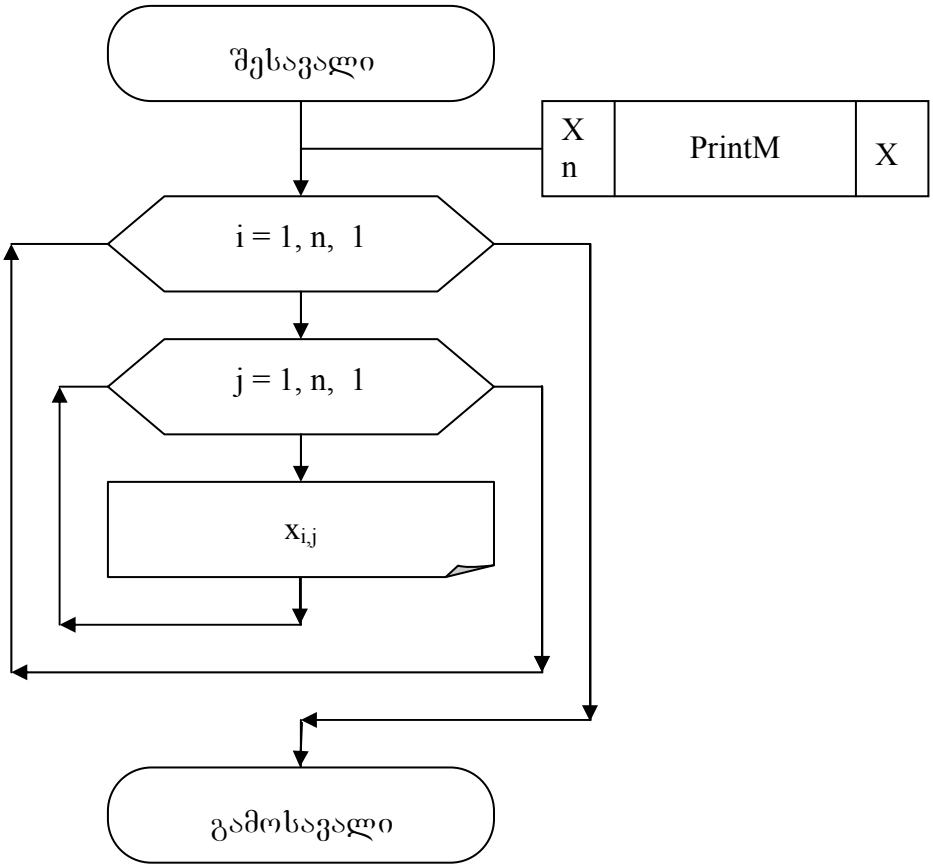
B მატრიცა

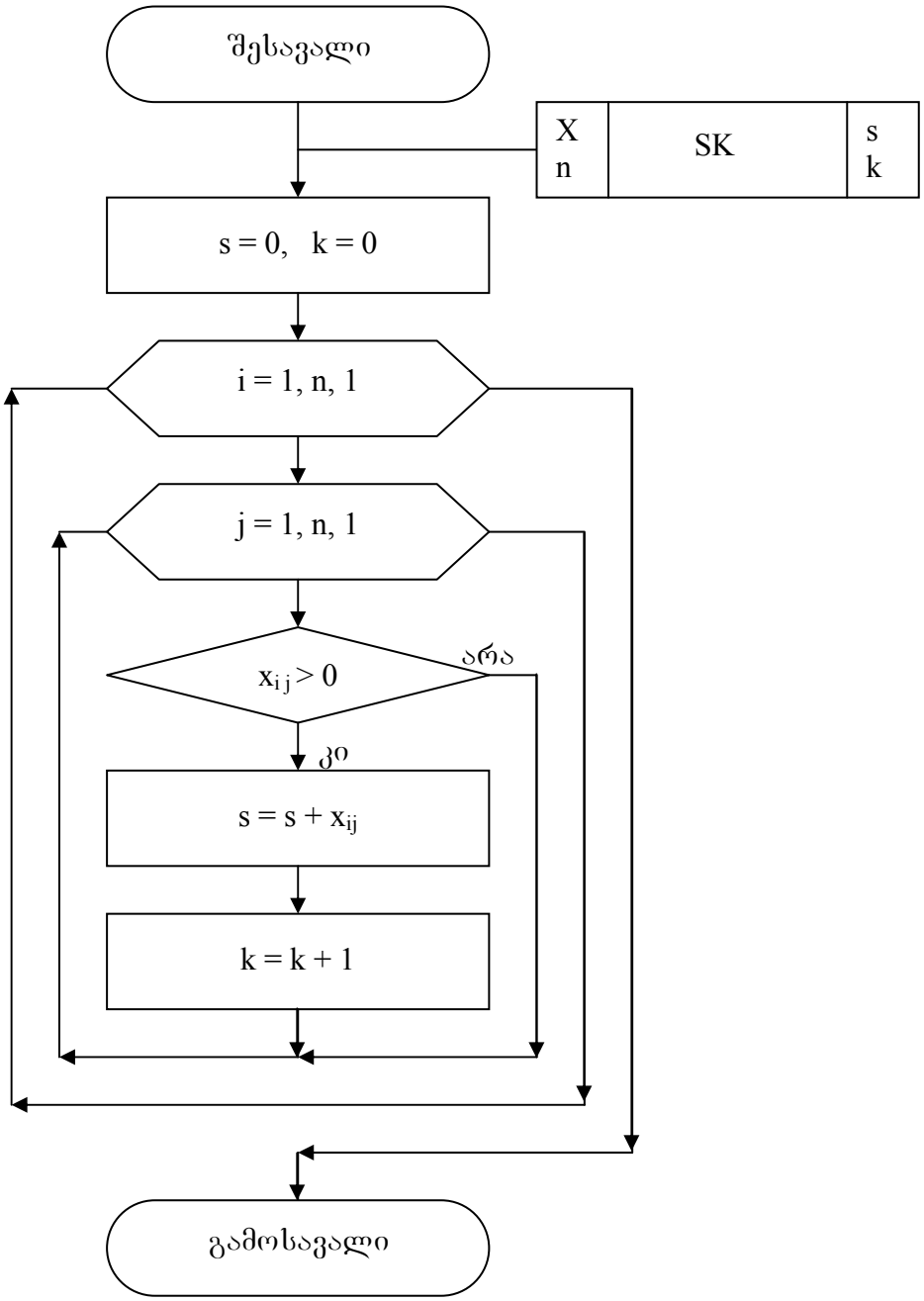
2	5	-7	-11
-1	0	0	4
-4	-2	-3	2
-9	-6	-5	6

3. ალგორითმის ბლოკ-სქემა




შესავალი
X
n
InputM
X
i = 1, n, 1
j = 1, n, 1
$x_{i,j}$
გამოსავალი


შესავალი
X
n
PrintM
X
i = 1, n, 1
j = 1, n, 1
$x_{i,j}$
გამოსავალი


შესავალი
X
n
SK
s
k
s = 0, k = 0
i = 1, n, 1
j = 1, n, 1
x i j > 0
არა
კი
s = s + x ij
k = k + 1
გამოსავალი

```
REM  Jg. <ჯგუფის ნომერი> Kapanadze 7-34
DECLARE  SUB  InputM(x(),n)
DECLARE  SUB  PrintM(x(),n)
DECLARE  SUB  SK(x(),n,s,k)
OPEN  "kap7-34.dat"  FOR  OUTPUT  AS  #1
DIM  a(5, 5), b(4, 4)
CLS
INPUT  "p"; p :  PRINT  #1,  "p ="; p
INPUT  "t"; t :  PRINT  #1,  "t ="; t
CLS:  PRINT  "A":  CALL  InputM(a(), 5)
CLS:  PRINT  "B":  CALL  InputM(b(), 4)
PRINT  #1,  "A":  CALL  PrintM(a(), 5)
PRINT  #1,  "B":  CALL  PrintM(b(), 4)
CALL  SK(a(), 5, sa, ka)
CALL  SK(b(), 4, sb, kb)
Z  =  p ^ ka * t ^ kb / EXP(sa + sb)
PRINT  "Z ="; Z
PRINT  #1,  "Z ="; Z
CLOSE  #1
END
SUB  InputM(x(),n)
FOR  i  =  1  TO  n
        FOR  j  =  1  TO  n
                LOCATE  i + 3, j * 8
                INPUT  x(i, j)
        NEXT  j
NEXT  i
END  SUB
SUB  PrintM(x(),n)
FOR  i  =  1  TO  n
        FOR  j  =  1  TO  n
                PRINT  #1,  x(i, j);
```

```
        NEXT  j
        PRINT  #1,
NEXT  i
END  SUB
SUB  SK(x(), n, s, k)
s = 0:  k = 0
FOR  i = 1  TO  n
        FOR  j = 1  TO  n
                IF  x(i, j) > 0  THEN
                        s = s + x(i, j)
                        k = k + 1
                END  IF
        NEXT  j
NEXT  i
END  SUB
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

```
p = 2
t = 3
A
 5      -7      -8       3       1
 2       3       -1      -4      -6
-9      -3       2       0       6
 0       4       -4       2       5
-5      -3       1       -3      -8
B
 2       5       -7      -11
-1       0       0       4
-4      -2       -3       2
-9      -6       -5       6
Z = 4.778908E-18
```

დაპროგრამება ქვეპროგრამის გამოყენებით

ლაბორატორიული სამუშაო № 7-32

1. ამოცანის პირობა

მოცემულია $X = (x_1, x_2, ..., x_{12})$ და $Y = (y_1, y_2, ..., y_{14})$ ვექტორები. გამოთვალეთ

$$z = \frac{a-b}{c+d},$$

სადაც

$$a = \min\{x_4, x_6, x_8, x_{10}, x_{12}\}$$
$$b = \min\{x_1, x_3, x_5, x_7, x_9\}$$
$$c = \min\{y_1, y_4, y_7, y_{10}, y_{13}\}$$
$$d = \min\{y_3, y_4, y_5, \ldots, y_{14}\}$$

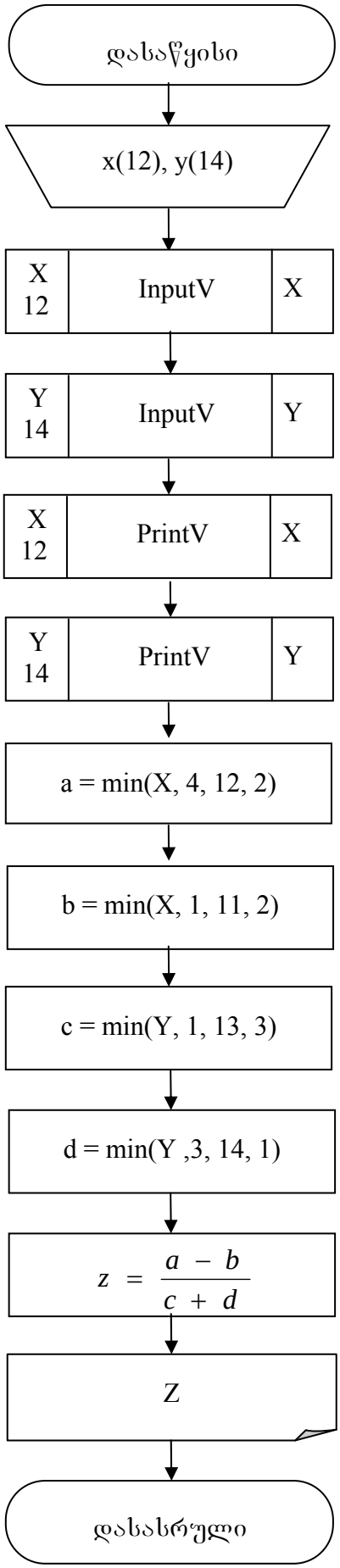
მინიმუმის გამოთვლის პროგრამა გააფორმეთ ფუნქციის სახით.

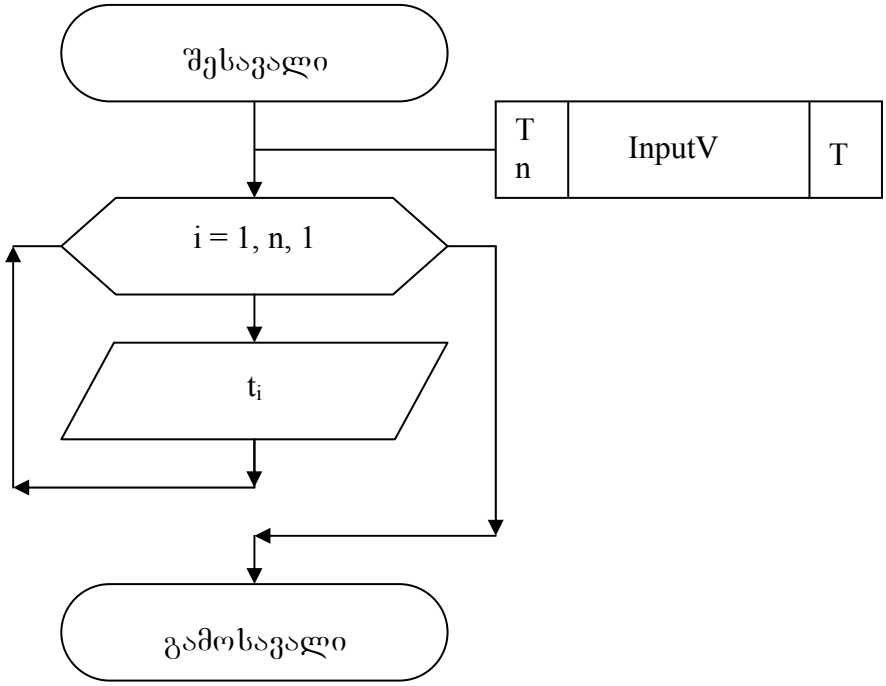
2. საწყისი მონაცემები

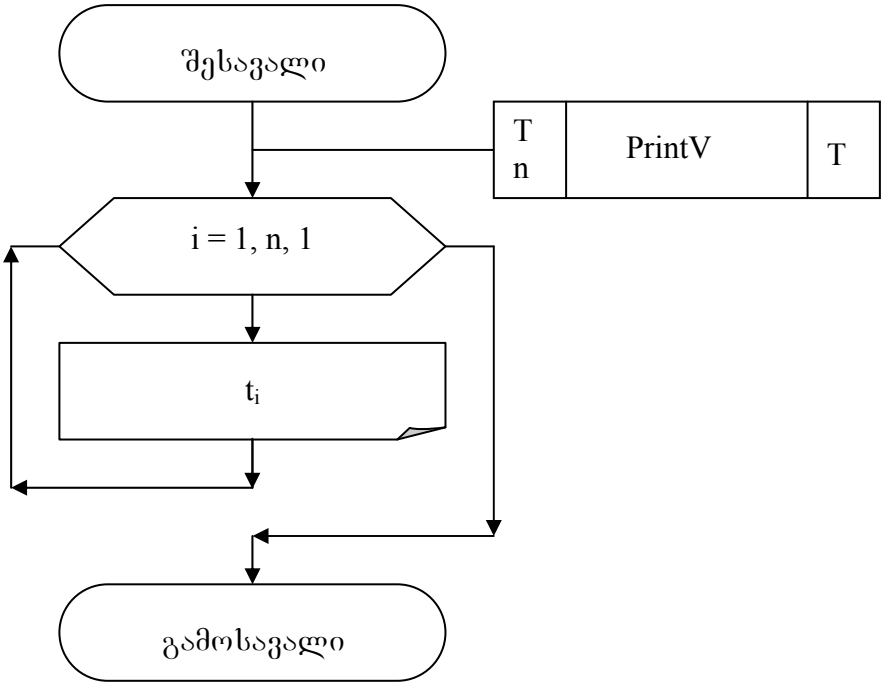
X = (1, -6, -3, 1, -1, 5, -4, 4, 2, 2, -7, 3)

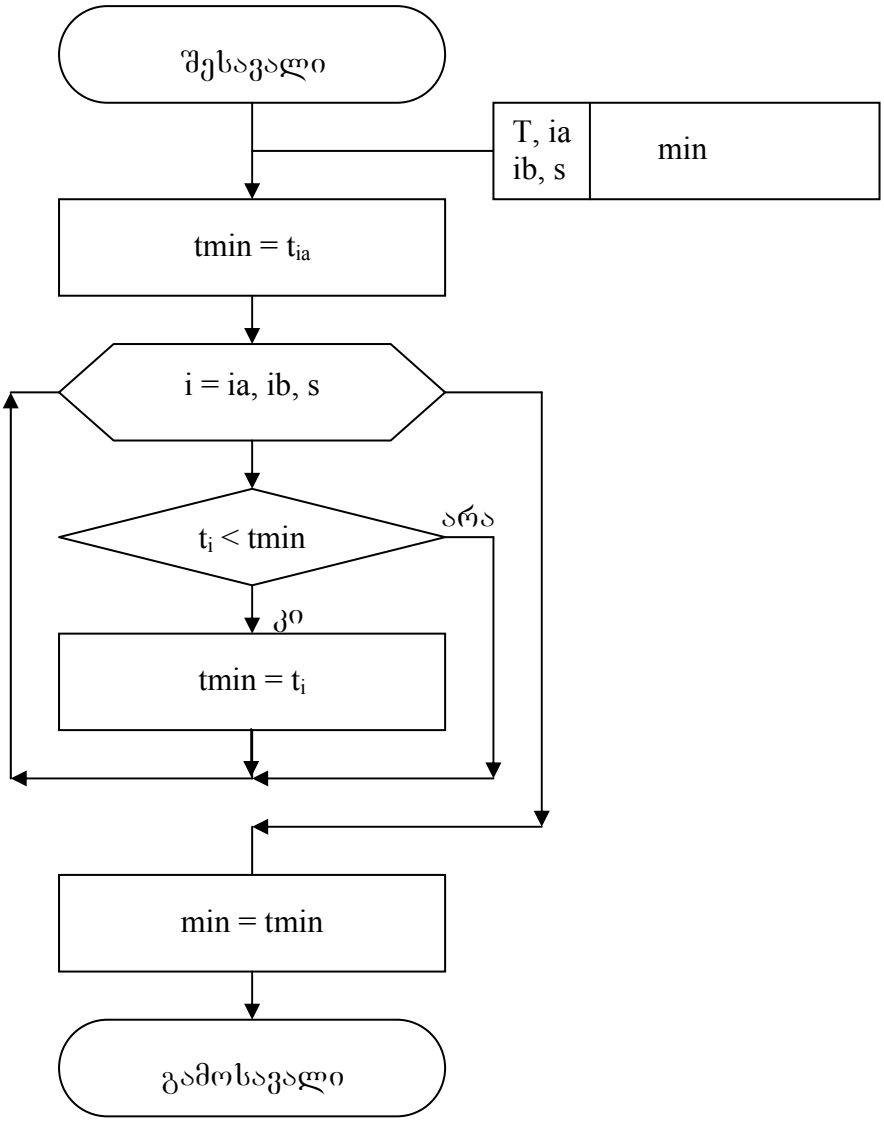
Y = (4, -5, -3, 2, 1, 5, 6, 3, 2, 2, 0, -1, 1, -2)

3. ალგორითმის ბლოკ - სქემა




შესავალი
T
n
InputV
T
i = 1, n, 1
t_i
გამოსავალი


შესავალი
T
n
PrintV
T
i = 1, n, 1
t_i
გამოსავალი


შესავალი
T, ia
ib, s
min
tmin = t_ia
i = ia, ib, s
t_i < tmin
არა
კი
tmin = t_i
min = tmin
გამოსავალი

```
REM  Jg.<ჯგუფის ნომერი>  Kapanadze  7-32
DECLARE  SUB  InputV(x(),12)
DECLARE  SUB  PrintV(x(),12)
DECLARE  SUB  InputV(y(),14)
DECLARE  SUB  PrintV(y(),14)
DECLARE  FUNCTION  min(t(),ia,ib,s)
OPEN  “kap7-32.dat”  FOR  OUTPUT  AS  #1
DIM  X(12), Y(14)
CLS
PRINT  “X”:  CALL  InputV(X(), 12)
PRINT  “Y”:  CALL  InputV(Y(), 14)
PRINT  #1,  “X”:  CALL  PrintV(X(), 12)
PRINT  #1,  “Y”:  CALL  PrintV(Y(), 14)
A  =  min(x(), 4, 12, 2)
B  =  min(x(), 1, 11, 2)
C  =  min(y(), 1, 13, 3)
D  =  min(y(), 3, 14, 1)
Z  =  (a - b) / (c + d)
PRINT  “z =”; z
PRINT  #1,  “z =”; z
CLOSE  #1
END
SUB  InputV(t(), n)
FOR  i  = 1  TO  n
        INPUT ;  t(i)
NEXT  i
PRINT
END  SUB
SUB  PrintV(t(), n)
FOR  i  =  1  TO  n
        PRINT  #1,  t(i);
NEXT  i
```

```
PRINT   #1,
END   SUB
FUNCTION   min(t(), ia, ib, s)
tmin  =  t(ia)
FOR   i  =  ia   TO   ib   STEP   s
        IF   t(i) < tmin   THEN   tmin  =  t(i)
NEXT   i
min  =  tmin
END   FUNCTION
```

5. კომპიუტერზე ამოცანის ამოხსნის შედეგი

Z = -3

ლიტერატურა

1. კ.კამკამიძე, ფ.პაატაშვილი. დაპროგრამება ბეისიკზე. “განათლება”, თბილისი, 1988.
2. Г.Моррил. Бейсик для ПК ИБМ, М.: Финансы и статистика, 1987.
3. Э.Каспер. Освоим QBasic (учебный курс). Радио и связь, 1999.
4. მეთოდური მითითებები ლაბორატორიული სამუშაოების შესასრულებლად “რიცხვითი მეთოდებისა და პროგრამირების” კურსში (ალგორითმიზაციისა და პროგრამირების საფუძვლები), სპი, თბილისი, 1989.
5. თ.მაჭარაძე, ზ.წვერაიძე. ინფორმატიკის საფუძვლები. “ტექნიკური უნივერსიტეტი”, თბილისი, 2003.