PHP 5 </>

ავტორი: ანდრო გაფრინდაშვილი (აკა)

დასაწყისისათვის გაეცანით!

სახელმძღვანელოში გადმოცემულია პროგრამირების ენა **php**, რომელიც დღეისათვის ვებ დაპროგრამების სამყაროში წამყვან ადგილს იკავებს თავის ფუნქციონალურობით, საიმედოობითა და მოხერხებულობით.

აქ წარმოდგენილი სასწავლო კურსის წარმატებით გავლის შემდეგ, თქვენ უპრობლემოდ შეგეძლებათ პროფესიონალური დონის, დაცული ვებგვერდების აწყობა. ასევე **php** პროგრამირების ენის წარმატებით შესწავლა, საწინდარია მომავალში უფრო რთული პროგრამირების ენების იოლად ათვისებისა, ისეთების მაგალითად როგორიცაა: **C#**, **C** და შემდგომ კიდევ უფრო რთული **C++**

მაქსიმალურად შევეცდები თავიდან აგარიდოთ არასასარგებლო და ხშირად გამოუყენებადი ინფორმაცია. ასევე თავს არ შეგაწყენთ აკადემიური ენით საუბრით, ვინაიდან და რადგანაც, დამწყებ **php** -
ს შემსწავლელებს ძალზედ გაუჭირდებათ მსგავსი ტიპის ინფორმაციის ათვისება. ჩვენ დაწვრილებით, დაბეჯითებით, ნაბიჯ-
ნაბიჯ და მარტივი ურთიერთობების ენით შევისწავლით მეტად რთულ **php** პროგრამირების ენას. სიტყვა რთული ნუ შეგაშინებთ,როგორც ამბობენ მთავარია მონდომება და რთული არაფერია.

თუკი სწავლების პროცესში პრობლემა შეგექმნათ, მაშინ კიდევერთხელ გადაავლეთ თვალი თეორიულ მასალას, გაითავისეთ იგი და მხოლოდ ამის შემდეგ შეუდექით პრაქტიკულ მეცადინეობას. აქვე დაიხსომეთ ერთი უმარტივესი და მეტად გონივრული გამონათქვამი: „არ არსებობს პრობლემა, არსებობს უბრალოდ ამოცანა“.

სამწუხაროდ წიგნის ელექტრონული ვერსია არაა შემოწმებული ქართული ენის სპეციალისტების მიერ, ასე რომ სასვენი ნიშნების შეცდომით გამოყენებისაგან და სინტაქსური დარღვევებისაგან სახელმძღვანელო დაზღვეული არ გახლავთ.

არავითარი შეცდომა არაა დაშვებული პროგრამირების მხრივ, კერძოდ ხშირია ელექტრონულ სახელმძღვანელოებში მაგალითებისა და ლისტინგების არასწორად ჩაწერის შემთხვევები, ხშირად ორმაგი პროგრამული ბრჭყალების ნაცვლად გამოყენებულია აპ

ოსტროფები, ან გამოტოვებულია მძიმეები და სხვა, ამ სახელმძღვანელოში კი მსგავსი ტიპის შეცდომებს ნამდვილად ვერ აღმოაჩენთ. მაშასე ვუსურვოთ ერთმანეთს წარმატებები. „წინპროგრამირების სამყაროს დასაპყრობად!!“

სხელმძღვანელოს ავტორი: ანდრო გაფრინდაშვილი -
(აკა) საქართველოს ტექნიკური უნივერსიტეტის, ინფორმატიკისა და მართვის სისტემების ფაკულტეტის, კომპიუტერული ტექნოლოგიების მესამე კურსი.
ელექტრონული ფოსტა: aka.net23@mail.ru
ვებგვერდის მისამართი: http://aka.net23.net
ავტორის გვერდი ფეისბუქზე: http://facebook.com/windowsaka

რა არის საჭირო **php** -ს ასათვისებლად ?

* პირველ რიგში საჭიროა ფლობდეთ ვებ დაპროგრამების ძალზედ მარტივ **html** (**HyperText Markup Language**) ენას, მხოლოდ ამის შემდეგ შეძლებთ აითვისოთ შედარებით უფრო რთული, **php** (**Personal Home Page**) დაპროგრამების ენა. უფრო დასაფასებელია თუკი **html** -თან ერთად, თქვენ ასევე ფლობთ პროგრამირების ენას **css** (**Cascading Style Sheet**) თუმცაღა ამ უკანასკნელის არცოდნა, ხელს არ შეგიშლით **php** -ს ათვისებაში.
* **php** ენა **html** -გან განსხვავებით სერვერული ტიპისაა, ანუ **php** სკრიპტს (პროგრამას, სცენარს) ამუშავებს სერვერი და რეზულტატი **html** ენის სახით მიეწოდება ბრაუზერს, რომელიც შემდეგ გამოისახება ეკრანზე მომხმარებლისათვის გასაგებ ენაზე. სწორედ ამიტომაა, რომ საიტის პროგრამული კოდის ნახვის დროს თქვენ ბრაუზერი გაწვდით არა **php**, არამედ **html** ანუ ჰიპერტექსტის პროგრამულ კოდს.
* თუკი სერვერთან არ მუშაობთ, მაშინ თქვენს ლოკალურ კომპიუტერზე უნდა დააყენოთ ვებსერვერი. ჩვენ ვისარგებლებთ დღეისათვის ყველაზე პოპულარული **apache web server** -ით. ლოკალურ კომპიუტერზე **http** (**HyperText Transform Protocol**) -ის დაყენების შემდგომ, ჩვენს შემთხვევაში კი **apache** -ის ინსტალაციის დასრულების შემდეგ, თქვენ ასევე უნდა დააყენოთ **php** და მიაბათ ი გი **apache** -

ს. ამ პროცედურების წარმატებით ჩატარების შემდგომ, თქვენ უპრობლემოდშეძლებთ **php** სკრიპტის გახსნას თქვენს ლოკალურ კომპიუტერზე.
* საჭიროა ლოგიკური აზროვნების უნარი, კომპიუტერთან დიდხანს ჯდომამ არ უნდ ა გადაგღალოთ, ასევე უნდა იყოთ მაქსიმალურად დაკვირვებულები, რათა არ გამოგრ ჩეთ პატარა წერტილი ან მძიმეც კი, რომელმაც შეიძლება ფატალური შედეგები გამოი წვიოს პროგრამის წერის დროს, პროგრამულ კოდში.
* ასევე საჭიროა გქონდეთ ინტერნეტში მუშაობის გამოცდილება, ინფორმაციის სწრაფ ად მოძიების უნარი, ძალზედ სასიხარულო ფაქტია თუკი ერკვევით კომპიუტერულ ქ სელებში ოდნავ მაინც, ხოლო თუკი გაქვთ გამოცდილება სხვა პროგრამირების ენებთა ნ მუშაობისა, მაშინ **php** ენის ათვისება თქვენთვის უმარტივესი გახდება.
* საჭიროა დაიცვათ სტანდარტები. დღეს დღეობით უამრავი ბრაუზერი არსებობს, რო მლის საშუალებითაც ხდება ვებგვერდების დათვალიერება, თითოეული ბრაუზერი ს ხვადასხვა პროგრამულ ბირთვზეა დაწერილი, ამიტომ აუცილებელია გაითვალისწინ ოთ თითოეული მათგანის მოთხოვნები, რათა სხვადასვხა ბრაუზერებმა თქვენი ვებგვ ერდი სხვადასხვაგვარად არ გამოსახონ მომხმარებელთა მონიტორებზე.
* აუცილებლად წაიკითხეთ თეორიული მასალა, პრაქტიკული მეცადინეობების დრო ს შეეცადეთ უშეცდომოდ დაწეროთ პროგრამები, ხოლო თუკი თქვენ პროგრამებს სახ ელმძღვანელოს ფურცლებიდან დააკოპირებთ, მაშინ შეეცადეთ თავიდან ბოლომდე მ ის გაშიფვრას, გაამახვილეთ ყურადღება თითოეულ დეტალზე.
პროგრამის გაშვების დროს შეცდომის დაფიქსირების შემთხვევაში დაუბრუნდით პრ ოგრამულ კოდს თავიდან, მოიძიეთ მასში შეცდომა, გაასწორეთ იგი და დაიმახსოვრე თ შეცდომის მიზეზი, რათა სხვა დროს, მსგავსი ტიპის შეცდომა ხელთავიდან აღარ დ აუშვათ.

ზოგიერთი რამ, რაც უნდა იცოდეთ!

* **http** (**hypertext transfer protocol**) ჰიპერტექსტის გადაცემის პროტოკოლი, სწორედ ეს პროტოკოლი მუშაობს ვე ბგვერდებთან.
* **www** (**world wide web**) მსოფლიოს გაფართოებული ობობას ქსელი, ანუ გლობალური სივრცე, ინტერნე ტი.
* **internet** - ინტერნეტი, ანუ საერთაშორისო ქსელი **international net** (სიტყვა **net** - ქსელს ნიშნავს).
* **ftp** (**file transfer protocol**) - ფაილების სერვერზე გადაგზავნის პროტოკოლი (იყენებს ოცდამეერთე პორტს).
* **server** - სერვერი, ჩვეულებრივ კომპიუტერთან შედარებით მძლავრი მანქანა, სადაც ინახება გ არკვეული სახის ინფორმაცია და რომელიც მართავს ქსელს.

* ip (internet
protocol) ინტერნეტ პროტოკოლი, რომლის საშუალებითაც ცნობენ ქსელში ჩართული კომპიუტერები ერთმანეთს.
* dns (domain name
system) დომენური სახელების სისტემა, დომენი სახელია, რომელიც ip მისამართზეა მ იბმული.
* hosting -
ჰოსტინგი, ინტერნეტში ვებგვერდების განთავსება, ანუ ლოკალური საიტის გლობალ ურ სივრცეში გადატანა.
* localhost -
ლოკალური ჰოსტი (ჰოსტი ანუ კომპიუტერი, მომხმარებლი, რომელიც ლოკალურ ქს ელშია ჩართული).
* error 404 - შეცდომა
404. გამოდის იმ შემთხვევაში, თუ მაგალითად მოთხოვნილი მისამართი ვერ მოიძებნ ა, ანუ არ არსებობს.
* error 403 - შეცდომა
403, რომელიც გამოდის იმ შემთხვევაში, როცა მოთხოვნილი მისამართი მოიძებნა, მაგ რამ აკრძალულია მასზე შეღწევა.
* error - ნიშნავს შეცდომას. php -
ზე პროგრამის წერის დროს თუკი დაფიქსირდა შეცდომა, იგი გეცნობებათ error - ით.
* index - დირექტორიის, ანუ კატალოგის საწყისი ფაილია.
(index.php ანდაც index.html არის ასევე default.html ანდაც default.php საწყისი ფაილები).
* url -
რესურსების უნივერსალური ლოკატორი, გამოსახავს დოკუმენტის ან სხვა ობიექტე ბის მისამართს.
* დირექტორია -
დირექტორია, ანუ იგივე ფოლდერი, საქაღალდე და როგორც ყველაზე ხშირად ეძახი ან - პაპკა.
* download - სიტყვა load ტვირთს ნიშნავს,
down კი ქვემოთ, ძირს, მაშასე, download ჩამოტვირთვას ნიშნავს.
* upload -
სიტყვა up ნიშნავს ზემოთ, მაღლა, ხოლო load როგორც უკვე განვიხილეთ ტვირთს, ან უ upload ნიშნავს ატვირთვას.
* კეში -
კეში, ანუ საიდუმლო მეხსიერება, რომელიც არ ჩანს. თქვენი ბრაუზერი სხვადასვა სა იტებს იმახსოვრებს თავის კეშ-
მეხსიერებაში, შედეგად კი, უკვე ერთხელ ჩამოტვირთული ვებგვერდი, მეორედ უფრ ო სწრაფად იტვირთება, რაც კეშ-მეხსიერების დამსახურებაა.
* დაპროგრამების ენა -
პროგრამის შესადგენი ალგორითმის გამოსახვა სპეციალურ კოდირებულ ენაზე, რომ

ელსაც ბრძანებები გადაჰყავს მანქანურ ენაზე. მანქანური ენის დროს პროცესორში ბრძ
ანებები იწერება ორობით კოდში 0 -ით და 1 -ით (ასავე 0 -ის და 1-
ების მიმდევრობით გადაცემით ხდება ქსელში პაკეტების გაცვლა, სადაც სიმბოლო 0 შ
ეესაბამება დენის მაღალ ძაბვას ხოლო 1 შედარებით დაბალ ძაბვას,
რომელსაც გამოიმუშავებს ქსელის ადაპტერი. მიღებული პაკატები შემდგომ გადაეგზ
ავნება კაბელს შემდგომი მარშრუტიზაციისათვის. ასეთი ტიპის კოდირება გამოიყენე
ბა ქსელის Ethernet ტექნოლოგიაში,
nrz ან მანჩესტერის კოდის გამოყენებისას. ხოლო უკაბელო შეერთებისას პაკეტები გა
დაიცემა რადიოარხით, სატელეფონო სიგნალებით ან თანამგზავრული ქსელით, wi-
fi ტექნოლოგიით და სხვ).

* network -
ქსელი, რომელიც უზრუნველყოფს კომპიუტერებს შორის ინფორმაციის გაცვლას (რო
გორც ზემოთ ავღნიშნეთ, ქსელში ინფორმაცია გადაიცემა პაკეტების სახით. კერძოდ,
რაიმე ფაილის გადაცემისას, ფაილი იყოფა პაკეტებად, გადადის იგი ქსელისათვის გას
აგებ ფორმატში და გაივლის კაბელს, შემდეგ მივა დანიშნულების ადგილამდე, მიმღე
ბი კომპიუტერი ააწყობს მიღებულ პაკეტებს და გადაიყვანს მას ქსელისათვის გასაგებ
ი ენიდან -ოპერაციული სისტემისათვის გასაგებ ენაზე. ამ ყოველივეს კოდირება-
დეკოდირების პროცესი ჰქვია და მას ქსელი უზრუნველყოფს თავისი საქსელო მოწყო
ბილობებით).

* პორტები -
პორტი შეიძლება აღვიქვათ როგორც ვირტუალური კარი, სადაც შედის და გამოდის ი
ნფორმაცია, ანუ ხორციელდება ტრაფიკი. პორტების რაოდენობა 1 -დან 65536 -
მდე გრძელდება. 80-ე პორტს იკავებს ვებსერვერი, 21 -ს ftp პროტოკოლი, 25 -
ს smtp პროტოკოლი და სხვა.

* პროტოკოლები - პროტოკოლი ეს წინასწარ გაწერილი სამუშაო ნაბიჯთა კრებულია,
ანუ პროტოკოლში უკვე წინასწარაა გაწერილი, მისი დანიშნულება.
არსებობს tcp, udp, ip, http, ftp, smtp და სხვა პროტოკოლები. პროტოკოლი tcp
უზრუნველყოფს ქსელში პაკეტების დანიშნულების ადგილამდე მიტანას, იგივე
მოვალეობა აკისრია udp პროტოკოლს, ოღონდ მისი გამოყენებისას პაკეტების
შემოწმება არ ხდება, ანუ udp პროტოკოლი არ იძლევა გარანტიას, რომ პაკეტებს
ბოლომდე მიიტანს და მათ დაუკარგავად გადასცემს დანიშნულების ჰოსტს.
პროტოკოლი http კი მუშაობს ვებგვერდებთან. smtp პროტოკოლს იყენებს ელ-ფოსტა.
ftp პროტოკოლი გამოიყენება ფაილების სერვერზე გადასაგზავნად.

* software - პროგრამული უზრუნველყოფა, ანუ როგორც ხშირად მას ეძახიან
„სოფტი“, რომელიც საჭიროა კომპიუტერის გასამართად.

* hardware - აპარატურული უზრუნველყოფა, ტექნიკური მხარე და როგორც მას
ხშირად ეძახიან, იგივე „ჰარდი“.

* admin - ადმინისტრატორი, ანუ სუპერმომხმარებელი, რომელიც სარგებლობს
სისტემის ყველა ფუნქციით და განსაზღვრავს მართვის პოლიტიკას.

* user - მომხმარებელი და როგორც მას ხშირად ეძახიან „იუზერი“.

1. apache სერვერის და php -ს დაყენება.

როგორც უკვე ავღნიშნეთ, იმისათვის, რომ თქვენ შეძლოთ **php** სკრიპტის გახსნა, მოგი წევთ დააყენოთ ვებსერვერი **apache** და შემდგომ **php**. წინააღმდეგ შემთხვევაში ბრაუზ ერი (ბრაუზერი პროგრამაა, რომლის საშუალებითაც ხდება საიტების დათვალიერება) თქვენს **php** სკრიპტს არასწორად გახსნის, ან მისი გახსნის ნაცვალდ, მის გადმოწერას შემოგთავაზებთ.

მაშასე, იმისათვის, რომ **php** ფაილი ბრაუზერმა გახსნას ისე, როგორც მაგალითად ჩვე ულებრივი **html** დოკუმენტი, საჭიროა **windows** ოპერაციულ სისტემაზე (სერვერზე მუ შაობისას ასევე ხშირად გამოიყენება ოპერაციული სისტემა **linux**, მაგრამ ჩვენ განვიხი ლავთ სერვერის დაყენებას **Microsoft windows** -
ის მაგალითზე) მოვიქცეთ შემდეგნაირად:
1. გადმოწერეთ **apache http server**
2.0 მისამართიდან: **http://httpd.apache.org/download.cgi** (ეს გახლავთ აპაჩის ოფიციალ ური ვებგვერდი, თუმცაღა სხვა საიტებიდანაც შეძლებთ აპაჩის გადმოტვირთვას). მაშ ასე, შევდივართ ზემოთ აღნიშნულ მისამართზე და ვიწერთ ფაილს სახელად **apache_2 .0.55-win32-x86-**
no_ssl.exe რომელიც განკუთვნილია ოპერაციული სისტემა ვინდოუსის **32** ბიტიანი ვე რსიისათვის (თუკი თქვენ სარგებლობთ **64** ბიტიანი სისტემით, მაშინ აირჩიეთ **64** ბიტიანი ოპერაციული სისტემისათვის შესაფერისი საინსტალაციო
ფაილი). ფაილის გადმოწერის შემდგომ ვიწყებთ აპაჩის ინსტალაციას:
2. ინსტალაციის პროცესი მიმდინარეობს ჩვეულებისამებრ ასე, რომ თქვენ პრობლემა არ შეგექმნებათ, მთავარია მიჰყვეთ ბრძანებებს, დანარჩენს კი აპაჩის საინსტალაციო ფ აილი თავად შეასრულებს.
საინსტალაციო **exe** ფაილის გაშვების შემდეგ მიჰყევით მოცემულ ინსტრუქციას:
* დაეთანხმეთ ლიცენზირებულ შეთანხმებას და დააჭირეთ ღილაკს **next (apache http server** წარმოადგენს უფასო პროგრამულ უზრუნველყოფას, ასე რომ თქვე არავითარი თანხის გადახდა არ დაგჭირდებათ ლიცენზიის მოსაპოვებლად).
* შემდგომ მოგიწევთ შეავსოთ ტექსტური ველები.
პირველ ველში უნდა მიუთითოთ თქვენი საიტის დომენის სახელი, თუკი დომენი არ გაქვთ ჩაწერეთ **localhost** ან სხვა თქვენთვის სასურველი მისამართი. მეორე ველში შეი ტანეთ საიტის სრული მისამართი ან კვლავ გამოიყენეთ მისამართი **localhost** და ბოლ ოს მიუთითეთ მეილი, თუკი მეილი არ გაქვთ მიუთითეთ ნებისმიერი.
* აუცილებლად აირჩიეთ რეჟიმი **for All users, on Port 80, as a Service – Recommended**. რათა აპაჩი მეოთხმოცე პორტზე
დაყენდეს, და პრობლემები არ შეგექმნათ მომავალში.
* აირჩიეთ რეჟიმი **Typical** და დააჭირეთ ღილაკს **next**.

ჩვენ აპაჩს დავაინსტალირებთ დუმილით გათვალისწინებულ საქაღალდეში, რომელი ც განთავსდება თქვენი სისტემური დისკის **Program Files** კატალოგში.
ამის შემდეგ, ამოწმებთ დაყენებულია თუ არა თქვენს კომპიუტერზე აპაჩი. ამისათვის შედიხართ მისამართზე **http://localhost** და თუკი, გამოდის შეტყობინება **It's work** ეს იმას ნიშნავს, რომ ყველაფერი რიგზეა, ხოლო თუკი ბრაუზერმა არავითარი შე ტყობინება არ გამოიტანა ეს იმას ნიშნავს, რომ აპაჩი არ მუშაობს, ამის მიზეზი შეიძლე ბა გახლდეთ შემდეგი ფაქტორები:
* ფაირვოლმა (firewall) დაბლოკა მეოთხმოცე პორტი, რომელიც აუცილებელია აპაჩის ფუნქციონირებისათვის. გამოსავალი გახლავთ ფაირვოლის ზოგიერთი ფუნქციის შე ზღუდვა.
* მეოთხმოცე პორტი მიუწვდომელია, ანუ დაკავებულია სხვა პროგრამის მიერ. გამოს ავალი გახლავთ იმ პროცესის ამოგდება სისტემიდან, რომელიც იყენებს მეოთხმოცე პ ორტს. ანდაც საჭიროა თავად პროგრამის კონფიგურაცია, რათა მან აღარ გამოიყენოს მეოთხმოცე პორტი (**skype**, **teamviewer** - განსაკუთრებით!)
* საჭიროა აპაჩისათვის ავტოსტარტის მინიჭება, ანუ აპაჩი უნდა ირთვებოდეს ავტომა ტურად, სისტემის ჩატვირთვისთანავე, როგორც სერვერი.
* აპაჩის შეფერხების მიზეზი შეიძლება იყოს ოპერაციული სისტემის გაუმართაობა ან სხვა. ოპერაციულ სისტემა **windows 7** -
ში, ხშირად შექმნილა პრობლემა აპაჩის გაშვებასთან დაკავშირებით. მახსოვს პრაქტიკ აში მქონდა შემთხვევა, როდესაც აპაჩ ვებსერვერის შეფერხების მიზეზი, სრულიად მო ულოდნელი, **C:\Windows\System32\drivers\etc** საქაღალდეში განთავსებული **host** ფაი ლი იყო.
2. მომდევნო ეტაპზე დავაყენოთ **php**. ამისათვის შედიხართ მისამართზე: **http://www.php.net/downloads.php** შემდგომ იწერთ არქივს სახელწოდებით **php-5.1.2-Win32.zip** და ახორციელებთ მის გაშლას. ჩვენ დასაწყისისათვის სისტემურ დისკზე **Program Files** საქაღალდეში შევქმნათ ახალი საქაღალდე სახელწოდებით **php** და მასში გავშალ ოთ (გამოვაარქივოთ), გადმოწერილი არქივი **php-5.1.2-Win32.zip**.

მორჩა **php** დაყენებულია, ეხლა საჭიროა იგი მივაბათ აპაჩს და მოვახდინოთ აპაჩ ვებს ერვერის კონფიგურაცია.

2. apache -ის და php -ს კონფიგურაცია.

მას შემდეგ, რაც დავაყენებთ **apache http server** –ს და **php** -
ს, ვიწყებთ მათ კონფიგურაციას. ჩვენ აპაჩი დავაყენეთ **C:\Program Files\Apache Group** საქაღალდეში, შესაბამისად თქვენი ვებგვერდის ელემენტები, **html** და **php** ფაილები უნდა გადაიტანოთ საქაღალდეში **C:\Program Files\Apache Group\Apache2\htdocs**

თუკი გსურთ საიტის ძირეული ფოლდერი შეცვალოთ, ანუ თქვენი საიტი განათავსო თ სხვა საქაღალდეში, ამისთვის მოგიხდებათ აპაჩის კონფიგურირება. მაგალითად D დისკზე შევქმნათ საქაღალდე სახელწოდებით saiti რომელშიც განთავსდება შემდგომ თქვენი ვებგვერდი.
მაშასე შექმენით D დისკზე საქაღალდე სახელწოდებით saiti შემდეგ კი თავად saiti საქა ღალდეში, შექმენით ახალი საქაღალდეები სახელწოდებებით www და cgi-
bin ამის შემდეგ, გახსენით აპაჩის საქაღალდე, სახელწოდებით C:\Program Files\Apache Group\
Apache2\logs და დააკოპირეთ ფაილები access.log და error.log (access და error) ჩვენს მი ერ შექმნილ საქაღალდეში D:\saiti შემდეგ გახსენით საქაღალდე C:\Program Files\Apache
Group\Apache2\conf მასში მოძებნეთ ფაილი httpd.conf (httpd) დაშეიტანეთ მასში შემდ ეგი სახის ცვლილებები:
* მოძებნეთ httpd.conf -ში ჩანაწერი:
ServerRoot "C:/Program Files/Apache Group/Apache2" -
ეს გზა მიუთითებს აპაჩ სერვერის მდებარეობის ადგილს, მას ჩვენ შეუცვლელად დავ ტოვებთ, ვინაიდან, აპაჩი ჩვენ სწორედ ამ მისამართზე გვაქვს დაინსტალირებული.
* აპაჩი იყენებს მე-80 პორტს, ამიტომაც ჩვენ Listen 80 ჩანაწერს არ შევცვლით.
* მიუთითოთ გზა საწყის დოკუმენტამდე. მოძებნეთ ჩანაწერი:
DocumentRoot "C:\Program Files\Apache Group\Apache2\htdocs"
და შეცვალეთ იგი ჩანაწერით:
DocumentRoot "D:/saiti/www"
ყურადღება გაამახვილეთ სლეშ ნიშანზე, კერძოდ:
ოპერაციულ სისტემა windows -ში გამოიყენება ესეთი სლეში: „\“
აპაჩის კონფიგურაციისას კი ჩვენ გამოვიყენებთ ესეთს: „/“
* მიუთითოთ საწყისი დირექტირია. მოძებნეთ ჩანაწერი:
<Directory "C:/Program Files/Apache Group/Apache2/htdocs">
და შეცვალეთ იგი ჩანაწერით:
<Directory "D:/saiti">
Options Indexes Includes
AllowOverride All
Order allow,deny
Allow from all
</Directory>
* მიუთითეთ დირექტორიის საწყისი ფაილები:
DirectoryIndex index.php index.html index.html.var
* ჩართეთ SSI:
AddType text/html .shtml
AddHandler server-parsed .shtml .html .htm
* მიუთითეთ გზა log ფაილებამდე, ამისათვის მოძებნეთ ჩანაწერი:
ErrorLog C:/Program Files/Apache Group/Apache2/logs /error.log
CustomLog C:/Program Files/Apache Group/Apache2/logs /access.log common

და შეცვალეთ იგი ჩანაწერით:
ErrorLog D:/saiti/error.log
CustomLog D:/saiti/access.log common
ეხლა დაარესტარტეთ აპაჩი და მორჩა, აპაჩის კონფიგურაცია დასრულდა. შემდეგ შექ მენით რაიმე მარტივი **index.html** ფაილი და შეინახეთ იგი **D:\saiti\www** საქაღალდეში. შემდგომ გახსენით მისამართი **http://localhost** თუკი თქვენს მიერ შექმნილი **index.html** ფაილი გამოდის ამ მისამართზე, ეს იმას ნიშნავს, რომ ყველაფერი რიგზეა.
თუკი რეზულტატი ვერ მიიღეთ და აპაჩი აღარ ირთვება, ეს იმას ნიშნავს, რომ თქვენ რ აღაც შეგეშალათ კონფიგურაციის დროს, საჭიროა დაბრუნდეთ თავში და გაასწოროთ შეცდომა.
აპაჩის კონფიგურაციის შემდგომ, საჭიროა **php** -
ს მიბმა მასზე, რათა შეძლოთ **php** ფაილის გახსნა. ამისათვის კვლავ გახსენით ნაცნობი **httpd.conf** ფაილი და შეიტანეთ მასში ცვლილება, კერძოდ, განყოფილება **AddType** -
ში ჩაამატეთ ჩანაწერი:
AddType application/x-httpd-php phtml php
<Directory "C:/Program Files/php">
Allow from all
Options ExecCGI
</Directory>
ScriptAlias "/php_dir/" "C:/Program Files/php/"
Action application/x-httpd-php "/php_dir/php-cgi.exe"
კვლავ დაარესტარტეთ აპაჩი. შემოწმებისათვის შექმენით პატარა **index.php** ფაილი, მა გალითად ასეთი შინაარსის:
<?php
print "hello php!";
?>
შეინახეთ ეს ფაილი **D:\saiti\www** საქაღალდეში და გახსენით მისამართი **http://localho st** თუკი გამოდის შეტყობინება „**hello php**“ ეს იმას ნიშნავს, რომ ყველაფერი რიგზეა, ხოლო თუკი მის ნაცვლად მთლიანი ტ ექსტი გამოვიდა, ანდაც არ გაიხსნა ფაილი, მაშინ მოგიწევთ მოძებნოთ კონფიგურაცი ის დროს დაშვებული შეცდომა და შეასწოროთ იგი. ჩამოთვლილი პროცედურების წა რმატებით დასრულების შემდეგ, თქვენ თქვენს ლოკალურ კომპიუტერზე დააყენებთ ვებსერვერს.. მომავალში კი თქვენს **php** ფაილებს შეინახავთ **D:\saiti\www** დირექტორი აში (ფოლდერში).

3. ჩვენ ვიწყებთ **php** -ს სწავლას!

დასაწყისისათვის გავეცნოთ **php** -ს სტრუქტურას.
php სკრიპტი შეიძლება საერთოდ არ შეიცავდეს **html** ტეგებს, ანდაც შესაძლებელია იგ ი ჩასმული იყოს **html** -ში, ნებისმიერ შემთხვევაში,

php კოდის შემცველი ფაილი, უნდა შეინახოთ **php** ფორმატში, მაგალითად **faili.php** სდ აც **faili** დოკუმენტის სახელია, ხოლო **.php** გაფართოება.

php პროგრამის დასაწერად არაა აუცილებელი რაიმე განსაკუთრებული პროგრამულ ი უზრუნველყოფა, უბრალო ტექსტური რედაქტორის **notepad** -
ის საშუალებით თქვენ უპრობლემოდ შეძლებთ **php** -ზე პროგრამების წერას.
html -ში, **php** სკრიპტის ჩასმის რამოდენიმე ხერხი არსებობს:

1. **<script language="phpscript">** კოდი **</script>**
2. **<?** კოდი **?>**
3. **<?php** კოდი **?>**

უნდა აღინიშნოს, რომ მესამე ვარიანტი უფრო მისაღებია ვიდრე დანარჩენი ორი. პრო გრამირების დროს **php** -
ს თითოეული დასრულებული ხაზი ერთმანეთისაგან უნდა გამოიყოს წერტილ-მძიმით.
თუკი **php** -
ზე პროგრამირების დროს თქვენ დაუშვებთ შეცდომას, მაშინ ეკრანზე გამოვა შეცდომ ის მანიშნებელი სიტყვა **error** და იმ ხაზის ნომერი, რომელზედაც აღმოჩენილი იქნა შ ეცდომა. მაგალითად:

Parse error: syntax error ... on line
4 რაც ნიშნავს, რომ პროგრამის მეოთხე ხაზზე დაშვებულია შეცდომა, ამიტომ თქვენ უ ნდა გადაათვალიეროთ მეოთხე ხაზი და გაასწოროთ შეცდომა. თუკი პროგრამაში შეც დომა ბევრჯერაა დაშვებული, გამოდის მხოლოდ წინამდებარე შეცდომით დაწერილი ხაზის ნომერი, ამ შეცდომის აღმოფხვრის შემდეგ მომდევნო ხაზის ნომერი და ასე გაგ რძელდება მანამ, სანამ არ გასწორდება ყველა შეცდომა.
Php ძალიან ნაკლებად თუ გაპატიებთ რაიმე ტიპის შეცდომებს, ამიტომ აუცილებელი ა პროგრამის წერის დროს არ აურდაურიოთ ერთმანეთში სხვადასხვა ელემენტები, არ გამოგრჩეთ წერტილ-მძიმე და პროგრამული ფრჩხილები.
უნდა აღინიშნოს, რომ **php** დოკუმენტის პირდაპირ გახსნა არ გამოიტანს სასურველ რ ეზულტატს, მას აუცილებლად უნდა მივაკითხოთ **http** პროტოკოლით, ჩვენს შემთხვე ვაში ბრაუზერმა უნდა მიაკითხოს **http://localhost** მისამარს. მხოლოდ **http** პროტოკო ლით გადაცემული მოთხოვნის შემდეგ გამოიტანს ბრაუზერი **php** -
ის რეზულტატს ეკრანზე.

4. ჩვენი პირველი პროგრამა.

გახსენით ტექსტური რედაქტორი ბლოკნოტი, **notepad** (**start menu** >> **all programs** >> **accessories** >> **notepad**) და მასში აკრიბეთ შემდეგი ტიპის ტექსტი:

```
<?php
print "es chemi pirveli programaa php-ze";
?>
```

შეინახეთ ფაილი როგორც index.php (შევთანხმდეთ კიდევ ერთხელ იმაზე, რომ php ფ
აილებს შეინახავთ D:\saiti\www საქაღალდეში) და შემდგომ მიაკითხეთ თქვენს შექმნი
ლ ფაილს http://localhost მისამართზე. თუკი ეკრანზე აისახება წარწერა „es chemi
pirveli programaa php-
ze“ ეს იმას ნიშნავს, რომ ყველაფერი რიგზეა, ხოლო თუკი რეზულტატი ვერ მივიღეთ,
მაშინ შესაძლოა ფაილის გაფართოება .php არასწორადაა მითითებული, ან აპაჩის კონ
ფიგურაციის დროს დაშვებულიაშეცდომა.
თუკი დაფიქსირდა error, მაშინ თავიდან გადახედეთ პროგრამას და გაასწორეთ შეცდ
ომით ჩაწერილი ხაზი.

დავბრუნდეთ დასაწყისში და განვიხილოთ ჩვენი პირველი მაგალითი. სდაც გამოყენ
ებულია ტექსტური მონაცემი "es chemi pirveli programaa php-
ze" და ოპერატორი print. მაშასე გავშიფროთ ზემოთ მოყვანილი პროგრამა:

print ჩანაწერის საშუალებით შესაძლებელია მონაცემების (ტექსტური, გრაფიკული თ
უ სხვა სახის) ეკრანზე გამოსახვა. ასევე იყენებენ ჩანაწერს echo რომელსაც ზუსტად იგ
ივე ფუნქცია აკისრია რაც print -ს, მაგალითად:

```
<?php
echo "es chemi pirveli programaa php-ze";
?>
```

შედეგად მივიღებთ იგივე რეზულტატს რაც პირველი პროგრამის გაშვებისას დაფიქს
ირდა. ეხლა კი მოდით დავწეროთ ქართული შრიფტით.

გაითვალისწინეთ, რომ notepad -
ში ქართული შრიფტით წერის დროს, სტანდარტული ANSI კოდირების ნაცვლად უნდ
ა გამოიყენოთ კოდირების რეჟიმი იუნიკოდი ან კოდირება UTF-
8, წინააღმდეგ შემთხვევაში ქართული შრიფტი გამოისახება კითხვის ნიშნებით, მაგა
ლითად:

```
<?php
echo "ეს ჩემი პირველი პროგრამაა php -ზე";
?>
```

ამ ტიპის ჩანაწერის შენახვის დროს თქვენ ირჩევთ UTF-
8 რეჟიმს, რათა ქართული შრიფტი გამოისახოს კორექტულად. იუნიკოდის არჩევის შე
მდეგ ზემოთ მოცემული პროგრამის გაშვებისას ეკრანზე გამოვა ჩანაწერი: „ეს ჩემი
პირველი პროგრამაა php -ზე“.

ბოლოს კი განვიხილოთ მაგალითი, სადაც html ტეგები და php პროგრამა ერთადაა გა
მოყენებული:

```
<html>
<head>
<title>
HTML და PHP
</title>
</head>
```

```
<body>
<?php
print "html და php მშვენივრად ეწყობიან ერთმანეთს!";
?>
</body>
</html>
```

როგორც ხედავთ html და php თავისუფლად შეიძლება განთავსდეს ერთ ფურცელზე.

შემდგომ გაკვეთილში უფრო დაწვრილებით განვიხილოთ ეს თემა.

5. html და php ერთად.

აუცილებელია იცოდეთ, რომ **html** არაჩვეულებრივად „მეგობრობს“ **php** -თან. **html** -ს ყველა ტეგი მშვენივრად თავსდება **php** -ში, მხოლოდ შევთანხმდეთ, რომ html ტეგებში ორმაგი ბრჭყალის ნაცვლად გამოიყენებთ ერთმაგს, ეს მეტად აუცილებელია, რათა **php** „გაუგებრობაში არ ჩავარდეს“, ვინაიდან და რადგანაც **php** სხვა დანიშნულებისათვისაც იყენებს ორმაგ ბრჭყალს, ამიტომ ადვილია ალბათობა იმისა, რომ „**php** -ს საკუთარი ბრჭყალი, **html** -ს ბრჭყალშიაერიოს“.

მაგალითად: **html** -ში სურათის ჩასმა:

```
<img src="file.png" alt="surati" title="surati html-shi">
```

php -ში კი ეს ყოველივე ასე გამოიყურება:

```
print "<img src='file.png' alt='surati' title='surati php-shi'>";
```

სხვა ტეგები კი, რომელთაც ბრჭყალები არ სჭირდებათ ჩვეულებრივად ჩაიწერება, მაგალითად:

```
print "<br>";
print"<h1>tetx</h1>";
print "<hr>";
```

აქვე შევთანხმდეთ, რომ **php** -ში ტექსტურ მონაცემებს ჩავსვავთ ორმაგ ბრჭყალებში, ხოლო ციფრებსა და რიცხვებს ჩავწერთ ბრჭყალების გარეშე, მაგალითად:

```
print "ტექსტური მონაცემი";
print 30;
```

ეხლა განვიხილოთ მაგალითი, რომლის საშუალებითაც თქვენ შეძლებთ დაინახოთ, თუ რა მარტივად შეიძლება **html** ტეგების გამოყენება **php** -ში, კერძოდ, მოდით ამოვბეჭდოთ ცხრილი.

<u>მაგალითი 1:</u>

```
<html><head><title>ცხრილი</title>
</head><body><?php
```

```
print "<table border=2 width=500 height=200 align='center'>
<tr>
<td align='center' bgcolor='#FFFF99'><h1>hello</h1></td>
<td align='center' bgcolor='#CCFF99'><h1>hello</h1></td>
</tr>
</table>";
print "<br><br>";
print "<HR WIDTH='80%' COLOR='#800080' SIZE=7>";
?></body></html>
```

ჩვენ ავღნიშნეთ, რომ **php** -
ში ყოველი დასრულებული ხაზის შემდეგ დაისმის წერტილ-
მძიმე, ზემოთ მოცემულ პროგრამაში, ცხრილის ამობეჭდვისას, ახალ ხაზზე გადასვ
ლა მრავალჯერ მოხდა, თუმცაღა წერტილ-
მძიმე არ გამოვიყენეთ, რადგანაც ცხრილის ამობეჭდვა არ დაგვისრულებია, ანუ **print** ჩანაწერში ვათავსებთ არა ერთ არამედ რამოდენიმე ტეგს ერთდროულად, ყველა ტეგი ს ჩაწერის შემდეგ კი **print** ჩანაწერი დაიხურება ორმაგი ბრჭყალით დადასრულდება წ ერტილ-მძიმით.

6. php -ს ზოგიერთი მოთხოვნა.

ამ პარაგრაფში განვიხილოთ ორი მაგალითი, მაშასე: პირველი მაგალითი:

```
<?php
print 5+2;
?>
```

და მეორე მაგალითი:

```
<?php
print "5+2";
?>
```

ალბათ შეამჩნევდით განსხვავებას, პირველი პროგრამა ეკრანზე გამოიტანდა ციფრს **7**, ხოლო მეორე დაბეჭდავდა **5+2** -
ს. ეს განპირობებულია იმით, რომ პირველ მაგალითში ჩვენ არ გამოვიყენეთ ბრჭყალე ბი, ანუ **php** -
მ ციფრები **5** და **2** აღიქვა როგორც ჩვეულებრივი ციფრი და შეკრიბა ისინი, რადგანაც მათ შორის შეკრების ნიშანია დასმული, შედეგად ეკრანზე გამოვიდა ციფრი **7**. მეორე მაგალითში ჩვენ გამოვიყენეთ ბრჭყალები, შედეგად **php** -მ ციფრები **5**,
2 დანიშანი **+** აღიქვა როგორც ტექსტი და დაბეჭდა ისინი ეკრანზე **print** ჩანაწერის საშუ ალებით.

მაშასე, კიდევ ერთხელ შევთანხმდეთ, რომ თუკი თქვენ მოგესურვებათ, მონაცემების, როგორც ტექსტის, ისე ამობეჭდვა, მაშინ თქვენ გამიყენებთ ორმაგ ბრჭყალებს. ასევე შევთანხმდეთ იმაზედაც, რომ ბრჭყალების და მსგავსი სიმბოლოების გამოყენების დრ ოს იქნებით მაქსიმალურად დაკვირვებულები, ჯერ ლოგიკურად ჩამოაყალიბებთ, იმა ს თუ რა გსურთ და ამის შემდეგ კი დაიწყებთ პროგრამის წერას. ეხლა კი დროა გადავ იდეთ მომდევნო თემაზე.

7. კომენტარები php -ში.

კომენტარის არსი ალბათ **html** -
დანაც კარგად გახსოვთ. ისევე როგორც სხვა პროგრამირების ენებს, ასევე **php** -
საც გააჩნია კომენტარების ჩაწერის საშუალებები.
კომენტარი პროგრამის ნებისმიერ ნაწილში შეიძლება განთავსდეს, თუმცაღა მისასალ მებელია, თუკი თქვენ, თქვენს საკუთარ კომენტარებს განათავსებთ ისეთ ადგილზე, ს ადაც შედარებით უფრო მოსახერხებელია მათი ჩაწერა, რათა ხელი არ შეუშალოთ „პრ ოგრამის მოწესრიგებულად წერას“, ანუ არ შევქმნათ ქაოსი პროგრამულ კოდში.
მაშასე php -ში არსებობს კომეტარის ჩაწერის სამი ვარიანტი:

```
// ეს კომენტარია ოღონდ მხოლოდ ერთ ხაზზე
# ესეც კომენტარია და ესეც მხოლოდ ერთ ხაზზე
/* ეს კი კომენტარია არა ერთ, არამედ რამოდენიმე ხაზზე, კერძოდ მასში შეიძლება გა
ნთავსდეს უაზრმაზარი ტექსტი */

<?php
print 50; // დაიბეჭდება ციფრი 50 (ეს კომენტარია და არ ჩანს!)
?>
```

8. ცვლადები.

ცვლადი არის რაიმე მნიშვნელობის მქონე ელემენტი. ცვლადს წინასწარ უნდა მივანი ჭოთ მნიშვნელობა. ცვლადის მნიშვნელობა შეიძლება იყოს, ციფრები, ტექსტი, ობიექ ტები, მასივი და ასე შემდეგ.
Php -
ში ცვლადები გამოისახება **$** სიმბოლოთი, შემდეგ ეტაპზე კი მას უნდა მიენიჭოს სახე ლი. ცვლადის სახელი არ შეიძლება იყოს ციფრები, კერძოდ **$12** არასწორია, ხოლო ანბ ანის სიმბოლოების გამოყენების დროს ციფრების გამოყენება დასაშვებია, მაგალითად

, ჩანაწერი $cvladi12 აბსოლიტურად ჭეშმარიტია. ცვლადის სახელში შეიძლება ასევე დეფისის გამოყენებაც, მაგალითად: $cvladi_1, $cvladi-2, $cvl-adi_3.
აუცილებლად უნდა მიექცეს ყურადღება დიდი და პატარა ასოებით წერას, კერძოდ $cvladi და $CVLADI ერთმანეთისგან აბსოლიტურად განსხვავებული ცვლადებია.
მაგალითი 2:

```
<?php
$cvladi = "გამარჯობა php"; //მივანიჭეთ ცვლადს მნიშვნელობა
print $cvladi; // ამოვბეჭდეთ ცვლადის მნიშვნელობა ეკრანზე
?>
```

შედეგად ეკრანზე დაიბეჭდება „გამარჯობა php“, ანუ მოვახდინეთ ცვლადის მნიშვნელობის ეკრანზე გამოტანა print ჩანაწერით.
ეხლა გამოვიყენოთ სიმბოლო წერტილის პრინციპი, რომელიც უზრუნველყოფს ორი ტექსტური, ან თუნდაც რიცხვითი მონაცემის შეერთებას.
მაგალითი 3:

```
<?php
$cvladi_1 = "ჩვენ ვაგრძელებთ ";
$cvladi_2 = "ცვლადების შესწავლას";
print "$cvladi_1" . "$cvladi_2"; //მოხდა ორი ტექსტის შეერთება
?>
```

შედეგად ეკრანზე დაიბეჭდება ფრაზა „ჩვენ ვაგრძელებთ ცვლადების შესწავლას“. ზოგადად ცვლადები ბრჭყალებში არ ჩაისმის, გამომდინარე იქედან, რომ php - მ ცვლადი უნდა აღიქვას, როგორც ცვლადი და არა როგორც ტექსტური მონაცემი. ზემოთ მოყვანილ მაგალითში კი ჩვენ ცვლადები ჩავსვით ბრჭყალებში, რაც მოცემულ სიტუაციაში აბსოლიტურად გამართლებულია, ვინაიდან და რადგანაც ჩვენ ცვლადებს შორის გამოვიყენეთ სიმბოლო წერტილი, რომლის საშუალებითაც მოხდა ორი ცვლადის მნიშვნელობის ერთმანეთზე მიბმა. ეხლა კი განვიხილოთ მომდევნო მაგალითი, სადაც ხდება ცვლადების მნიშვნელობების შეკრება.
მაგალითი 4:

```
<?php
$cvladi_1 = 10;
$cvladi_2 = 20;
print $cvladi_1 + $cvladi_2; //დაიბეჭდება რიცხვი 30
?>
```

ამ შემთხვევაში, თუკი ცვლადებს ჩავსვავდით ბრჭყალებში, ეკრანზე დაიბეჭდებოდა ტექსტი **10+20**, ხოლო რადგანაც ჩვენ გვსურდა, რომ მოგვეხდინა რიცხვების შეკრება, ა ნუ **php** -
თვის „მიგვეცა საშუალება“ რომ ცვლადი აღექვა როგორც ცვლადი და არა როგორც ტე ქსტური მონაცემი, ამიტომაც არ გამოვიყენეთ ბრჭყალები.
მაგალითი 5:

```
<?php
$cvladi_1 = 10;
$cvladi_2 = $cvladi_1;
print $cvladi_2; //დაიბეჭდება რიცხვი 10, რადგანც ერთი ცვლადი მეორეს გაუტოლეთ.
?>
```

9. მონაცემთა ტიპები.

ქვემოთ მოცემულია მონაცემთა ტიპების ცხრილი.

integer	მთელი რიცხვი: 5; -5; 10;
double - float	მცურავმძიმიანი, ათწილადი რიცხვი: 3.14; -3.14;
string	ტექსტური ტიპის მონაცემი: "text"
array	მასივი: array
object	ობიექტი: object

gettype ჩანაწერს გამოაქვს ეკრანზე ცვლადის ტიპი.
settype ჩანაწერი ცვლის ცვლადის ტიპის მნიშვნელობას.
მაგალითი 6:

```
<?php
$cvladi = 10;
print gettype ($cvladi); // გამოვიყვანეთ ცვლადის ტიპი ეკრანზე
?>
```

მაგალითი 7:

```
<?php
$cvladi_1 = 3.14; // მოვახდინოთ ცვლადის ტიპის შეცვლა.
```

```
$cvladi_2 = (double) $cvladi_1;
print gettype ($cvladi_2);
print " -- $cvladi_2<br>";
$cvladi_2 = (integer) $cvladi_1; // მოვახდინეთ double -
ს გადაყვანა integer ტიპში
print gettype ($cvladi_2);
print " -- $cvladi_2<br> ";
?>
```

10. ოპერატორები.

არითმეტიკული ოპერაციების ცხრილი:

+	შეკრება	>	მეტია
-	გამოკლება	<	ნაკლებია
*	გამრავლება	=>	მეტია ან ტოლია
/	გაყოფა	<=	ნაკლებია ან ტოლია
%	ნაშთიანი გაყოფა	++	ინკრემენტი
=	უდრის, ტოლია	--	დეკრემენტი
==	გატოლება	===	გატოლება
!=	არ არის ტოლი	.	ათწილადის მძიმე

ლოგიკური ოპერატორების ცხრილი:

if	ლოგიკური ოპერატორი if
elseif	ლოგიკური ოპერატორი elseif
else	ლოგიკური ოპერატორი else
and & &&	ლოგიკური „და“, ლოგიკური“სწრაფი და“
xor or | ||	ლოგიკური „ან“, ლოგიკური“სწრაფი ან“
not !	ლოგიკური „არა“ , (უარყოფა)

ზემოთ ჩამოთვლილ არითმეტიკულ ოპერაციებს და ლოგიკურ გამოსახულებებს სა თითაოდ განვიხილავთ მომდევნო გაკვეთილებში. ლოგიკური ჩანაწერების კარგად გა საგებად შემოგთავაზებთ უამრავ მაგალითებს, მანამდე კი დაუბრუნდეთ არითმეტიკ

ულ ოპერაციებს. არითმეტიკულ ოპერაციებში ალბათ გაუგებარი არაფერი არაა გარდა ინკრემენტისა და დეკრემენტისა.

++ ინკრემენტი

-- დეკრემენტი

ინკრემენტი მონაცემს ზრდის ერთით, ხოლო დეკრემენტი კი ამცირებს, მაგალითად **1 0** -ის ინკრემენტი არის **11**, ხოლო დეკრემენტი **9**.

რაც შეეხება **integer**, **double**, **float** და **string** ტიპებს, აქ რთული და ამოუცნობი არაფერია **integer** წარმოადგენს მთელ რიცხვს, როგორც ნატურალურს ასევე უარყოფით რიცხვებს.

double და **float** წარმოადგენს ათწილად რიცხვებს, ანუ როგორც მას ხშირად ეძახიან მც ურავმძიმიან რიცხვს, როგორც დადებითს ასევე უარყოფითს.

string ტიპი კი წარმოადგენს ჩვეულებრივ ტექსტურ ჩანაწერს ანუ ტექსტს.

php 4 -ში შემოიტანეს სიახლე, კერძოდ, **===** ოპერატორი ამოწმებს არამარტო ცვლადების ტოლობას, არამედ მათი მონაცემთა ტიპების თანხვედრასაც.

11. არითმეტიკული ოპერაციები.

განვიხილოთ რიცხვების შეკრება გამოკლება, გამრავლება, გაყოფა, პროცენტის გამოთვლა და ნაშთის დადგენა, ანუ ყველა ტიპის სტანდარტული არითმეტიკული მოქმედებები. ამისათვის ქვემოთ მოცემულია მაგალითები, რომელთა საშუალებითაც ნათლად დაინახავთ, თუ როგორ მარტივად ხდება **php** -ში მათემატიკური ოპერაციების შესრულება:

მაგალითი **8:**

```
<?php
$ricxvi1 = 10;
$ricxvi2 = 5;
print $ricxvi1 + $ricxvi2; //მივიღებთ 15-ს, რადგანაც 10+5=15
print "<br>";
print $ricxvi2 - $ricxvi1; //მივიღებთ -5-ს, რადგანაც 5-10=-5
print "<br>";
print $ricxvi1 * $ricxvi2 / 10; //მივიღებთ 5-ს, რადგანაც 10*5/10=5
?>
```

მაგალითი **9:**

```
<?php
$ricxvi1 = 200;
$ricxvi2 = 50;
```

```
print "$ricxvi1 -ის $ricxvi2 პროცენტი არის " . $ricxvi1 * $ricxvi2 / 100;
?>
```

მაგალითი 10:

```
<?php
$ricxvi1 = 50;
$ricxvi2 = 5;
$ricxvi3 = 4;
print "$ricxvi1 -ის $ricxvi2 -ზე გაყოფისას, ნაშთში დაგვჩება " . $ricxvi1 % $ricxvi2;
print "<br>";
print "$ricxvi1 -ის $ricxvi3 -ზე გაყოფისას კი, ნაშთში დაგვჩება " . $ricxvi1 % $ricxvi3;
print "<br><hr>";
print "როგორც ხედავთ php -ს თურმე კარგად ცოდნია არითმეტიკა.";
print "<br>";
print "ჩვენი გაკვეთილი კი ამით დამთავრდა!";
?>
```

ბოლო, მეათე მაგალითში ჩვენ განვიხილეთ ნაშთიანი გაყოფის მაგალითი, კერძოდ, 50 გავყავით ჯერ 5 -ზე და შემდეგ 4 -ზე, შედეგად 5 -ზე გაყოფისას ნაშთში დაგვრჩა 0, რადგანაც 50 უნაშთოდ იყოფა 5-ზე, ხოლო 4 -ზე გაყოფისას ნაშთში დაგვრჩა 2.

როგორც ხედავთ უბრალო მათემატიკური გამოთვლების ჩატარება პრობლემას არ წარ მოადგენს, ასევე სიძნელეს არ წარმოადგენს შედარებით უფრო რთული მათემატიკურ ი გაანგარიშებების ჩატარება, კერძოდ, სამკუთხედის ფართობის გამოთვლა, მოქმედებ ები კვადრატულ ფესვებზე, ლოგარითმების გამოყვანა, ახარისხება და სხვა.ზემოთ აღნიშნულ შედარებით რთულ მათემატიკურ გამოთვლებ ს ჩვენ აუცილებლად განვიხილავთ მომდევნოეტაპებზე, ეხლა კი გადავიდეთ შემდეგ გაკვეთილზე, სადაც თქვენ გაეცნობით ლოგიკურ ოპერატორებს.

12. ლოგიკური ოპერატორები.

ალბათ გსმენიათ გამოთქმა „ხედაავ რა ჭკვიანია კომპიუტერი?!“ დასაწყისისათვის უნ და ითქვას, რომ ჭკვიანი არის პროგრამისტი ვინც უწერს კომპიუტერს პროგრამის სახ ით, თუ რა, სად და როგორ უნდა შეასრულოს მან. ნებისმიერი „რთული“ პროგრამა შე იცავს ლოგიკურ ოპერატორებს, რომლების საშუალებითაც ხდება ინფორმაციის ნაკა დის მართვა. ლოგიკური ოპერატორების საშუალებით თქვენ კომპიუტერს უბრძანებ თ, გააკეთოს „ან“ ერთი, „ან“ მეორესაქმე ანდაც,

„თუ“ ერთი დავალება არ სრულდება „მაშინ“ გაამოიტანოს შესაბამისი შეტყობინება, „ან“ გადავიდეს მომდევნო დავალების შესრულებაზე და სხვა.

ლოგიკური ოპერატორები ჩვენ ცხრილის სახით გვქონდა მოცემული მეათე გაკვეთი ლში, მანამ სანამ, უშუალოდ დაიწყებდეთ პროგრამებთან მუშაობას, კიდევ ერთხელ შ ეიხსენეთ ლოგიკური ოპერატორების ცხრილი.
<u>მაგალითი 11:</u>

```
<?php
$ricxvi1 = 2;
$ricxvi2 = 5;
if ($ricxvi2 > $ricxvi1) // აქ ორწერტილის დაწერა შეცდომაა, პირობა გრძელდება!
{
print "ჭეშმარიტებაა, რომ $ricxvi1 ნაკლებია $ricxvi2 -ზე";
}
else // პირობა კვლავ გრძელდება, ასე რომ ორწერტილს აქაც არ დაისმება!
{
print "შეცდომაა, რომ $ricxvi1 ნაკლებია $ricxvi2 -ზე";
}
?>
```

რა მოხდა პროგრამაში? თქვენ პროგრამაში გამოაცხადეთ ორი ცვლადი, რომელთა მნი შვნელობები იყო 2 და 5. ოპერატორ if -
ში ჩაწერეთ, რომ თუ პირველი ცვლადის მნიშვნელობა 2 ნაკლები იქნებოდა მეორე ცვ ლადის მნიშვნელობაზე 5, მაშინ პროგრამას უნდა გამოეტანა ფრაზა „ჭეშმარიტებაა, რ ომ 2 ნაკლებია 5-
ზე“ ხოლო თუკი ეს პირობა არ შესრულდებოდა, მაშინ გააქტიურდებოდა ოპერატორი else რომელიც გამოიტანდა შეტყობინებას „შეცდომაა, რომ 2ნაკლებია 5 -
ზე“, მაგრამ else -ს გააქტიურება აღარ გახდა საჭირო, რადგანაც if -
ში მითითებული პირობა შესრულდა,
2 მართლაც ნაკლებია 5 ზე, და სწორედ ამიტომაც ეკრანზე დაიბეჭდა ფრაზა „ჭეშმარი ტებაა, რომ 2 ნაკლებია 5 -ზე“. თუკი if -ში თქვენ ჩაწერთ ესეთ პირობას if (ricxvi2 < ricxvi1) მაშინ if -
პირობაში არსებული ჩანაწერი აღიქმება როგორც ტყუილი (FOLSE) რადგანაც 5 ნაკლებ ი არაა 2 -
ზე და ეკრანზე გამოვა else ოპერატორის პირობა, როგორც მართალი (TRUE)რადგანაც ჩ ვენი მაგალითიდან გამომდინარე, თუ ერთი ჩანაწერი არაა მართალი, მაშინ მეორე აუ ცილებლად მართალი უნდა იყოს.
if და else ლოგიკური ოპერატორების გარდა ნაკადის სამართავად გამოიყენება ასევე elseif ოპერატორი, რომელიც პროგრამაში else -

გან განსხვავებით ბევრჯერ შეიძლება გამოიყენოთ. ასევე შესაძლებელია თითოეულ ლოგიკურ ოპერატორს ცალ-ცალკე განუსაზღვროთ ლოგიკური პირობა.

სტყვა **if** ქართულად „თუ“-ს ნიშნავს, **else** ჩანაწერი კი **if** -ის მომდევნო ეტაპზეა, ასე ვთქვათ მის საწინააღმდეგო პირობას ასრულებს. უფრო ზუსტად რომ გაიგოთ, რისთვის და რატომ გამოიყენება ეს ოპერატორები, ამისათვის მოვიყვანოთ საიტზე ავტორიზაციის გავლის მაგალითი.

```
if (password = 123)
{საიტზე შესვლა ნებადართულია}
else
{საიტზე შესვლა შეუძლებელია}
```

დასაწყისისათვის გაცნობებთ, რომ ზემოთ მოყვანილი მაგალითი **php** პროგრამას არ წარმოადგენს, აქ უბრალოდ ნაჩვენებია ლოგიკური ოპერატორების პრინციპი, ეხლა გავაანალიზოთ მოცემული ჩანაწერი. ესეიგი, თუკი თქვენი პაროლი არის **123** მაშინ **if** ჩანაწერის თანახმად თქვენ მოგეცემათ საიტზე შესვლის უფლება, ხოლო სხვა დანარჩენ შემთხვევაში, ანუ ტყუილი პაროლის აკრებისას გააქტიურდება **if** -ის საწინააღმდეგო ჩანაწერი **else**, რომელიცარავითარ შემთხვევაში არ მოგცემთ საიტზე შესვლის უფლებას.

ეხლა კი დროა განვიხილოთ მომდევნო მაგალითი.

მაგალითი **12:**

```
<?php
$ricxvi1 = 2;
$ricxvi2 = 5;
if ($ricxvi1 + $ricxvi2 == 8) //აქ გამოიყენება == და არა = რადგანაც ჯამს უტოლებთ 8-ს
{
print "ჭეშმარიტებაა, რომ $ricxvi1 -ს დაუმატოთ $ricxvi2 უდრის 8 -ს ";
}
else
{
print "შეცდომაა, რომ $ricxvi1 -ს დაუმატოთ $ricxvi2 უდრის 8 -ს";
}
?>
```

თუკი აქ წარმოდგენილი უმარტივესი მაგალითებით და ნამდვილად დაწვრილებით გადმოცემული თეორიული ნაწილის გაცნობის შემდეგ მაინც გაგიჭირდათ ლოგიკური ოპერატორების არისის გაგება, გთხოვთ კიდევ ერთხელ გადაათვალიეროთ თეორიული და პრაქტიკული ნაწილი. მოცემულ ლოგიკურ ოპერატორებზე ჩვენ კიდევ გვექნება მაგალითები მომდევნო თავებში.

მომავალში ასევე გავეცნობით ლოგიკურ „და“-ს ლოგიკურ „არა“-ს ლოგიკურ „ან“ ჩანაწერს და სხვა.

13. მეთოდი GET და მეთოდი POST.

ალბათ გახსოვთ ფორმებთან მუშაობა **html** პროგრამირების დროს, შესაბამისად კარგა
დ უნდა გახსოვდეთ **get** და **post** ჩანაწერებიც. ყოველი შემთხვევისათვის შეგახსენებთ,
რომ ზემოთ ხსენებული ჩანაწერები გამოიყენება ფორმის გადასაცემად, გადასაგზავნა
დ. რაღათქმაუნდა მათ შორის არის განსხვავებაც, კერძოდ: მეთოდი **get** გადასცემს ფო
რმას, მაგალითად ერთი ვებ-
ფურცლიდან მეორე ვებ ფურცელზე, ან თავად იმავე ფურცელზე და ამსათან ერთადბ
რაუზერის **URL** მისამართის არეში ფაილის მისამართს მიებმება ფორმის ზოგიერთი ე
ლემენტის მნიშვნელობები. მაგალითად თუ დოკუმენტის მისამართია: **http://domainn
ame.com/pages/index.php** და თუკი მასზე **get** ჩანაწერით მარტივ ფორმას გადაცემთ, სა
დაც გამოყენებული იქნება მხოლოდ ერთი ტექსტური ველი, რომელშიც მაგალითად
ჩავწერთ სიტყვას **ok** -
შედეგად ბრაუზერის ნავიგაციის ველში მივიღებთ: **http://domainname.com/pages/ind
ex.php?text=ok** მისამართს,სადაც ჩანაწერი **text=ok** ფორმის ტექსტურ ველში ჩაწერილ
ინფორმაციას ასახავს.

მეთოდი **post** გადასცემს ფორმას, ოღონდ ბრაუზერის ნავიგაციის ველში არ ჩაამატებს
არავითარ ფორმის ელემენტების მნიშვნელობებს, ანუ დოკუმენტის **URL** მისამართს შ
ეუცვლელად დატოვებს. ესეიგი, ჩვენს მიერ მოყვანილ მაგალითში, მისამართი **http://
domainname.com/pages/index.php** ფორმის გადაცემის შემდეგ შეუცვლელი დარჩება, თ
უკი **post** მეთოდს გამოვიყენებთ.
$_GET['textbox'] და **$_POST['textbox']** ჩანაწერებით ხდება **php** დოკუმენტში ფორმის აღ
ქმა, ანუ უფრო გასაგებად რომ ვთქვათ, თუკი თქვენ რაიმე ფორმას აგზავნით **get** ჩანაწ
ერით, მაშინ იმისათვის, რომ **php** –
მ თქვენს მიერ გაგზავნილი ფორმა დაამუშაოს, გამოიყენებთ ჩანაწერს **$_GET['textbox']**
სადაც ერთმაგ ბრჭყალში ჩაწერილი **textbox** სიტყვა იმ ტექსტური ველის, ან ფორმის ს
ხვა ელემენტის, ან მთლიანი ფორმის სახელია, რომელიც **php** -
მ უნდა დაამუშაოს. იგივე ვარიანტია**post** მეთოდის გამოყენებისას, ოღონდ ამ შემთხვ
ევაში გამოიყენებთ ჩანაწერს **$_POST['textbox']** ასევე უნდა იცოდეთ, რომ **html** დოკუმე
ნტს არ შეუძლია ფორმის მიღება და მისი დამუშავება, რადგანაც იგი პროგრამებს არ ა
მუშავებს, არამედ უბრალოდ გამოიყენება ინფორმაციის გადასაცემად. დაპროგრამები
ს შედარებით რთულ და ფუნქციონალურ ენებს, ისეთებს როგორებიცაა **php**,
javascript თავისუფლად შეუძლიათ მიიღონ და დაამუშაონ ნებისმიერი სახის ფორმა.
იმისათვის, რომ უკეთ გაიგოთ ფორმებთან მუშაობის არსი, გთავაზობთ შექმნათ **php**
ფაილი სახელწოდებით **forms.php** და მასში განათავსოთ პატარა პროგრამა, რომელიც
ქვემოთ, მაგალით **13** -შია წარმოდგენილი.
<u>მაგალითი **13**:</u>

```
<html><head><title>ფორმები</title></head><body>
<form action="forms.php" method="GET">
<input type="text" name="textbox" />
<input type="submit" value"გაგზავნა" />
</form>
<?php
print $_GET['textbox'];
?>
</body></html>
```

შედეგად ფორმის გაგზავნისას ეკრანზე დაიბეჭდება ის სიმბოლოები, რაც თქვე აკრიბ ეთ ტექსტურ ველში, ხოლო თუკი ტექსტური ველი ცარიელია, ეკრანზე ფორმის მნიშვ ნელობა არ გამოვა. ეხლა კი დავუბრუნდეთ მაგალითს და გავარჩიოთ იგი, ჩვენ **forms.php** ფაილიდან გავაგზავნთ ფორმა იმავე **forms.php** ფაილზე, ანუ დოკუმენტმა ფორმა გადასცა თავის თავს. გამოვიყენეთ **$_GET['textbox']** ჩანაწერი, ანუ პროგრამაში მიუთი თეთ, რომ ეკრანზე გამოსულიყო **textbox**სახელის მქონე ფორმის ელემენტის მნიშვნელ ობა (**textbox** სახელი კი ჩვენ ფორმის წერისას, მივანიჭეთ ტექსტურ ველს), პროგრამამა ც არ დააყოვნა და პასუხიც მოგვაწოდა.

<u>მაგალითი 14:</u>

```
<html><head><title>ფორმები</title></head><body>
<form action="forms.php" method="GET">
<input type="text" name="textbox" />
<input type="submit" value"გაგზავნა" />
</form>
<?php
$forma = $_GET['textbox'];
print $forma;
?>
</body></html>
```

ეს მაგალითიც თითქმის იგივეა, რაც მაგალითი **13**, განსხვავება მხოლოდ ისაა, რომ ე რთმა ცვლადმა მიიღო მეორე ცვლადის მნიშვნელობა, ანუ **$forma** გაუტოლდა **$_GET['t extbox']** ცვლადს.

<u>მაგალითი 15:</u>

```
<html><head><title>აბა თუ გამოიცნობ!</title></head>
<body><h1 align="center">
<form action="forms.php" method="POST">
<input type="text" name="textbox" />
```

```
<input type="submit" value"გაგზავნა" />
</form>
<?php
$forma = $_POST['textbox'];
print "რამდენია 5+5 ?<br>";
if ($forma ==10)
{
print "$forma - პასუხი სწორია";
}
else {
print "$forma - პასუხი არასწორია";
}
?>
</h1></body></html>
```

14. პატარა გამოთვლითი პროგრამა.

მაშასე, მოდით შევქმნათ პატარა გამოთვლითი პროგრამა, ამისათვის მოვიყვანოთ ესეთი პროგრამული კოდი:

```
<html><head>
<title>ჩემი პროგრამა</title>
</head><body><h3 align="center">
პროგრამა გამოარკვევს უდრის თუ არა, მეტია თუ არა, ან ნაკლებია თუ არა თქვენს მიე
რ პირველ და მეორე ტექსტურ ველებში შეტანილი რიცხვების ჯამი მესამე ველში შეტ
ანილ რიცხვზე.<br><hr><br>
<form action="forms.php" method="post">
1 ველი <input type="text" name="t1" />
2 ველი <input type="text" name="t2" />
3 ველი <input type="text" name="t3" />
<input type="submit" value="გამოანგარიშება" />
<input type="reset" value="გასუფთავება!" />
</form>
</h3>
<h1 align="center">
<?php
$a = $_POST ['t1'];
$b = $_POST ['t2'];
$c = $_POST ['t3'];
if ($a == "" || $b == "" || $c == "") // გამოვიყენეთ „სწრაფი ან“ (||) ოპერატორი
```

```
{
print "გთხოვთ შეავსოთ სამივე ტექსტური ველი";
}
elseif ($a + $b > $c)
{
print $a + $b." მეტია $c -ზე";
}
elseif ($a + $b < $c)
{
print $a + $b." ნაკლებია $c -ზე";
}
elseif ($a + $b == $c)
{
print $a + $b." უდრის $c -ს";
}
?></h1>
</body></html>
```

ჩვენი გამოთვლითი პროგრამაც უკვე მზადაა მუშაობისათვის.

15. include -ს გამოყენება.

ჩანაწერი include -
ს საშუალებით შესაძლებელია რამოდენიმე php დოკუმენტის ერთ დოკუმენტში გამოყ ვანა, ასევთქვათ სხვადასხვა php დოკუმენტები ერთ ფანჯარაში გამოისახება, ისე როგ ორც ეს ფრეიმის ჩამატების დროს ხდება.

მაგალითად გვაქვს ორი php დოკუმენტი სახელწოდებებით: 1.php და 2.php ხოლო ჩვე ნ გვსურის ეს ორივე დოკუმენტი გამოვიყვანოთ ერთად index.php ფაილში, ამისათვის index.php დოკუმენტში უნდა მიუთითოთ შემდეგი:

```
<?php
include "1.php";
include "2.php";
?>
```

შედეგად ზემოთ აღნიშნული ორივე დოკუმენტის შინაარსი ბრაუზერის ფანჯარაში ე რთად გამოისახება.

16. კონსტანტები.

კონსტანტის, ანუ იგივე მუდმივას გამოცხადება **php** -
ში ხდება ჩანაწერით **define()** კონსტანტა შეიძლება ცვლადათაც კი აღვიქვათ, ანუ ის შე
იცავს მნიშვნელობას, იმ განსხვავებით, რომ ცვლადის მნიშვნელობის შეცვლა შესაძლ
ებელია, ხოლო კონსტანტის მნიშვნელობის შეცვლა შეუძლებელი, ანუ რა მნიშვნელო
ბაც მიენიჭება კონსტანტას გამოცხადებისას, იგივე დარჩება ბოლომდე, სწორედ ამიტ
ომაცაა იგი მუდმივა.
<u>მაგალითი 16:</u>

```
<?php
define ('mudmiva','500');
print mudmiva; // შედეგად ეკრანზე დაიბეჭდება 500
?>
```

როგორც ხედავთ მაგალითში კონსტანტის ამობეჭდვისას კონსტანტის სახელი **mudmi
va** ბრჭყალებში არაა ჩასმული, ვინაიდან და რადგანაც ჩვენ **php** -
ს უნდა მივცეთ საშუალება, რომ კონსტანტა აღიქვას, როგორც კონსტანტა და არა როგ
ორც ტექსტური მონაცემი (სტრინგ მონაცემი - **string)**

როგორც მაგალითიდან ჩანს, კონსტანტის სახელი და მისი მნიშვნელობა ერთმაგ ბრჭყ
ალში ჩაისმის, ხოლო კონსტანტის გამოძახება ხდება მისი სახელით, რომელიც მას გამ
ოცხადებისას უნდა მიენიჭოს. ასევე აუცილებელია გაითვალისწინოთ დიდი და პატა
რა სიმბოლოების პრინციპი, ანუ ჩანაწერი **mudmiva** და **MUDMIVA** ერთმანეთისგან სრუ
ლიად განსხვავდებიან, ხოლო თუკი გვსურს, რომ მათში განსხვავება არ იყოს, ანუ უგ
ულვებელვყოთ დიდი და პატარასიმბოლოების პრინციპი, მაშინ კონსტანტის აღწერი
სას უნდა მიუთითოთ ჩანაწერი **true** (**folse** -
ს შემთხვევაში კი დიდი და პატარა სიმბოლოების პრინციპი უგულვებელყოფილი არა
ა. ანუ კონსტანტის აღწერისას ჩანაწერი **folse** დუმილით გათვალისწინებულია.)
<u>მაგალითი 17:</u>

```
<?php
define ('mudmiva','500', true);
print mudmiva; // პირველ შემთხვევაში დაიბეჭდება 500
print "<br>";
print MUDMIVA; // მეორე შემთხვევაშიც 500, რადგან გამოყენებულია ჩანაწერი true
?>
```

რისთვისაა საჭირო კონსტანტები? რაღათქმაუნდა ისინი ძალიან საჭირონი არიან, კე
რძოდ, კონსტანტის საშუალებით შესაძლებელია ვებგვერდის ბმულებით
დაქსელვა, ამისათვის კონსტანტას გაუტოლებთ დოკუმენტის **url** მისამართს და ყოვე

ლ ნაბიჯზე აღარ დაგვჭირდება დიდი ბმულების ხელთავიდან წერა, ხოლო მათ ნაცვლად გამოვიყენებთ კონსტანტებს.

მაგალითი 18:

```
<?php
define('linki','/images/png/logo/events/');
print linki; // დაიბეჭდება სურათის url მისამართი
print "<br>";
print '<img src="'.linki.'pic.png">'; // გზა სურათამდე, დაიბეჭდება სურათი
?>
```

გარდა ჩანაწერისა **define**(), კონსტანტის გამოცხადებისას ასევე იყენებენ ჩანაწერს **defined**() მათ შორის კი თავისთავად არის განსხვავებაც.

define() -

გან განსხვავებით ჩანაწერი **defined**() იღებს მხოლოდ ერთ არგუმენტს და თუკი კონსტანტა დაყენებულია იგი იღებს მნიშვნელობას **true** ხოლო წინააღმდეგ შემთხვევაში მნიშვნელობას **folse**.

php ში არსებობს უკვე წინასწარ განსაზღვრული კონსტანტებიც:

__LINE__ მიმდინარე ხაზის ნომერი ფაილში

__FILE__ გზა მოცემულ დოკუმენტამდე

__FUNCTION__ ფუნქციის სახელი

__CLASS__ კლასის სახელი

__METHOD__ კლასის მეთოდის სახელი

მაგალითი 19:

```
<?php
print 'ფაილამდე არსებული გზა '. __FILE__ ;
print "<br>";
print 'პროგრამული ხაზის ნომერი '.__LINE__;
?>
```

შედეგად ეკრანზე ამოიბეჭდება სრული გზა თქვენს **php** დოკუმენტამდე და პროგრამული ხაზის ნომერი.

მაშასე თქვენ უკვე იცით კონსტანტების გამოყენება, ასევე ფლობთ **php** -ს საწყის ეტაპს, ეხლა კი დროა გადავიდეთ შემდგომ ეტაპზე.

17. ციკლები.

ჩვენ ყოველ წელს ვხვდებით შემოდგომას, ზამთარს, ზაფხულსა თუ გაზაფხულს, მაშასადამე ჩვენ მომსწრენი ვართ წელიწადის დროთა ციკლირებისა. წელიწადის დროთა

ბრუნვადობა უსასრულოდ გრძელდება, ანუ ამ შემთხვევაში საქმე გვაქვს უსასრულო ციკლირებასთან, თუმცაღა ციკლი შეიძლება შეწყდეს, ანუ არ იყოს უსასრულო. როგო რც წესი ციკლირება უნდა ხდებოდეს მოწესრიგებულ რეჟიმში, ანუ წელიწადის დრო თა ციკლირებისას ზამთრისშემდეგ არ უნდა მოდიოდეს შემოდგომა, ზაფხულის შემ დეგ გაზაფხული და სხვა.

ზემოთ მოყვანილ წელიწადის დროთა ციკლირების მაგალითი შეიძლება თამამად გა უთანაბროთ პროგრამულ ციკლებს, ანუ ისევე როგორც ბუნებაში, პროგრამაშიც ხდება რაღაც პროცესის განმეორება გარკვეული წესრიგით, ანუ ხდება ციკლის შესრულება.

Php -
ში გამოიყენებენ სამი სახის ციკლს (შინაარსობრივად ისინი ერთიდაიმავე დატვირთვ ის გახლავთ, განსხვავდებიან მხოლოდ ჩაწერიოლობით) პირველი: **while** ციკლი, მეორ ე **do … while** ციკლი და მესამე **for** ციკლი.

while - ციკლი

განვიხილოთ კონკრეტული მაგალითი რათა უფრო ადვილი გახდეს თქვენთვის **whil e** ციკლის გამოყენების არსი:

მაგალითი **20:**

```
<?php
$cikli = 0;
while ($cikli <= 9)
{
print $cikli;
print "<br>";
$cikli++;
}
?>
```

შედეგად ეკანზე ამოიბეჭდება ციფრები **0** -დან **9** -
მდე. ზემოთ მოყვანილ მაგალითში ჩანაწერი **$cikli++** ციკლის მნიშვნელობას ზრდის ე რთით. ჩვენს მაგალითში ცვლადის მნიშვნელობა უდრის **0** -ს,
while ჩანაწერში კი მითითებულია, რომ ციკლი არ უნდა გასცდეს, არ უნდა აღემატებო დეს **9** -ს სწორედ ამიტომაც ციკლი იწყება **0** -
დან, ყოველ ახალ ეტაპზე იზრდება ერთით და მთავრდება მითითებულ ციფრთან **9**.

do … while ციკლი

მომდევნო ეტაპზე განვიხილოთ **do … while** ციკლი. **while** ციკლსა და **do …**
while ციკლს შორის განსხვავება ის გახლავთ, რომ **do …**
while ციკლის გამოყენებისას, ციკლის მოქმედების არე მიეთითება **while** ჩანაწერში, ხო ლო ჩანაწერში **do** თავად ციკლის მოქმედება.

მაგალითი **21:**

```
<?php
$cikli = 1;
do
{
print "$cikli<br>\n";
$cikli++;
}
while ($cikli > 200 && $cikli < 400);
?>
```

მომდევნო ეტაპზე ჩვენ განვიხილავთ **for** ციკლს, უნდა აღინიშნოს, რომ **for** ციკლის გ ამოყენება უფრო მოსახერხებელია ვიდრე დანარჩენი ორის.

for ციკლი

მოვიყვანოთ მაგალითი სადაც გამოყენებულია ციკლის ოპერატორი **for**

მაგალითი **22:**

```
<?php
for ($cikli=1; $cikli<=15; $cikli++)
{
print "ნომერი $cikli<br><hr>";
}
?>
```

ეხლა განვიხილოთ მაგალითი, სადაც გამოყენებულია ერთმანეთში ჩადგმული ციკლ ები. პროგრამას ეკრანზე გამოჰყავს გამრავლების ტაბულა.

მაგალითი **23:**

```
<?php
print "<table border=1\n>";
for ($y=1; $y<=9; $y++)
{
print "<tr>\n";
for ($x=1; $x<=9; $x++)
{
print "<td align='center' width=50 height=50>";
print ($x * $y);
print "</td>\n";
}
print "</tr>\n";
}
print "</table>";
?>
```

როგორც ხედავთ მაგალითში გამოყენებულია ორი **for** ციკლი, ისინი ერთმანეთში არიან ჩადგმულნი და თითოეული მათგანი თავის დავალებას ასრულებს ერთმანეთისგან დამოუკიდებლად.

შემდეგ ეტაპზე განვიხილოთ ციკლში **break** და **continue** ჩანაწერების გამოყენების მაგალითი.

break და continue ჩანაწერები ციკლში.

break ჩანაწერის საშუალებით შესაძლებელია ციკლის შეწყვეტა ნებისმიერ მონაკვეთში.

continue ჩანაწერის საშუალებით შესაძლებელია ციკლის შეჩერება და კვლავ განახლება, ანუ ციკლიდან რომელიმე ელემენტის ამოგდება.

განვიხილოთ მაგალითები თითოეულ მათგანზე, რათა უფრო გაგვიადვილდეს მათი შემდგომი გამოყენება:

მაგალითი **24:**

```
<?php
for ($cikli = -5; $cikli <= 10; $cikli++)
{
if ($cikli == 0)
break; // როგორც კი $cikli გაუტოლდება ნულს ციკლი შეწყდება
$cvladi = 100 / $cikli;
print "100 -ის $cikli -ზე გაყოფისას მივიღებთ $cvladi -ს<br>";
}
?>
```

მაგალითი **25:**

```
<?php
for ($cikli = -5; $cikli <= 10; $cikli++)
{
if ($cikli == 0)
continue; // როგორც კი $cikli გაუტოლდება ნულს ციკლი გადაახტება მას
$cvladi = 100 / $cikli;
print "100 -ის $cikli -ზე გაყოფისას მივიღებთ $cvladi -ს<br>";
}
?>
```

ოცდამეხუთე მაგალითში ჩვენ გამოვიყენეთ **continue** ჩანაწერი, კერძოდ, პირობაში მიუთითეთ, რომ როდესაც **$cikli** -
ის მნიშვნელობა გახდებოდა ნული ამ დროს ციკლი იმოქმედებდა **continue** ბრძანების

შესაბამისად, სწორედ ამიტომაც პროგრამის შესრულებისას, **100** -
ის ნულზე გაყოფა არ მოხდა, ანუ ციკლი **-1** -დან პირდაპირ **1**-ზე გადახტა.

ოცდამეხუთე მაგალითში გამოყენებული **continue** ჩანაწერი არითმეტიკული ჭეშმარი ტების დასტურიცაა, კერძოდ იმისა, რომ ნულზე გაყოფა არ შეიძლება, ანუ ჩვენს შემთ ხვევაში **100** რომ ნულზე გაგვეყო არ ვიქნებოდით მართალნი მათემატიკური თვალსაწ იერიდან გამომდინარე და ამ დარღვევას თავად **php** -იც გვამცნობდა.
ციკლები პროგრამირების პროცესში უდიდეს როლს თამაშობენ, მათი საშუალებით დაპროგრამების ენა უფრო მოსახერხებელი და ფუნქციონალური ხდება.
მომდევნო ეტაპზე ჩვენ განვიხილავთ მასივიბეს, რომელთა შემოღებაც უდიდესი წინგადადგმული ნაბიჯია პროგრამირების სფეროში.

18. მასივები.

მასივი ერთი სახელის მქონე ელემენტთა კრებულია, რომელთაც შეგვიძლია მივმართ ოთ ინდექსით ან სტრიქონების მეშვეობით. არსებობს სამი სახის მასივი: მარტივი მასი ვი, რთული -
მრავალგანზომილებიანი მასივი და ასოცირებული მასივი. ასოცირებული მასივი ისე თი სახის მასივია, რომლის გამოყენებაც დაპროგრამების სხვა ენებშიდაცაა შესაძლებე ლი. რთული, ანუ მრავალგანზომილებიანი მასივი კი ეწოდება ისეთი სახის მასივებს, რომელთაელემენტები თავად მასივებს წარმოადგენს.

მასივის ელემენტები აუცილებლად უნდა გამოიყოს ერთმანეთისაგან მძიმით, გამონაკ ლისს წარმოადგენს მხოლოდ ბოლო ელემენტი.

მასივის წევრების აღნუსხვა ხორციელდება ნატურალური რიცხვებით, თუმცაღა ათვ ლა იწყება არ **1** -დან, არამედ **0** -
დან, ანუ მასივის პირველი ელემენტის რიგითი ნომერი გახლავთ **0**.

მარტივი მასივი.

დასაწყისისათვის აუცილებელია გავეცნოთ მასივის ელემენტთათვის მნიშვნელობები ს მინიჭების წესს, კერძოდ, მასივის ელემენტთათვის მნიშვნელობის მინიჭება ხდება ო რი ხერხით:
1. **$masivi = array (1, 2, 3, 4);** -
სადაც **$masivi** მასივის სახელია ხოლო ჩანაწერი **array** მიუთითებს იმაზე, რომ შემდგო მ იწყება მასივი, ფრჩხილებში ჩასმული ციფრები კი მასივის ელემენტებს წარმოადგენ ს.

2. `$masivi = array ( "pirveli" => 1,`
`"meore" => 2,`
`"mesame" => 3 );` - სადაც მასივის ელემენტებს ცალ-ცალკე აქვთ მინიჭებული მნიშვნელობები. ოპერატორი `=>` ს საშუალებით ხდება მასივის ელემენტ თათვის მნიშვნელობის მინიჭება.

ეხლა განვიხილოთ კონკრეტული მაგალითი, სადაც წარმოდგენილია უმარტივესი მა სივი:

მაგალითი 26:

```
<?php
$mas = array (
"სახელი" => "ნინო",
"გვარი" => "ქამუშაძე",
"ასაკი" => 25
);
print $mas ['სახელი'];
?>
```

შედეგად ეკრანზე ამოიბეჭდება სახელი „ნინო“, მაშასადამე ჩვენ მივმართეთ მასივის პ ირველ ელემენტს და მოვითხოვეთ მისი მნიშვნელობის ამობეჭდვა ჩანაწერით `print $mas ['სახელი']` შედეგად ეკრანზე ამოიბეჭდა ჩვენს მიერ მოთხოვნილი ელემენტის მნიშვნე ლობა, კერძოდ ტექსტური ჩანაწერი „ნინო“.

მომდევნო ეტაპზე განვიხილოთ შემდეგი მაგალითი.

მაგალითი 27:

```
<?php
$mas = array ( "ნატალია", "ქეთი", "თამუნია" );
print $mas ['0'];
?>
```

ზემოთ, ოცდამეშვიდე მაგალითში მოყვანილი პროგრამის გაშვებისას ეკრანზე ამოიბე ჭდება სახელი „ნატალია“, ვინაიდან და რადგანაც ჩვენ მივმართეთ მასივის ელემენტს , რომლის რიგითი ნომერი გახლავთ **0**, ანუ მასივის პირველ ელემენტს. მასასადამე კი დევ ერთხელ გავიხსენოთ ზემოთ აღნიშნული მითითება, სადაც ჩვენ ვთქვით, რომ მა სივის ელემენტების აღნუსხვა იწყება **0** -დან და არა **1**-დან.

რთული, ასოცირებული მასივი

როგორც უკვე ავღნიშნეთ, რთული მასივები ისეთი სახის ს მასივებს წარმოაგენენ, რო

მელთა ელემენტები თავად მასივები არიან, ანუ ერთ მასივში ჩადგმულია რამოდენიმ
ე სხვა მასივი.
მაგალითი 28:

```
<?php
$mas = array (
array (
"სახელი" => "ნიკა",
"გვარი" => "გოგიჩაიშვილი",
"ასაკი" => 30
),
array (
"სახელი" => "სანდრო",
"გვარი" => "სამხარაძე",
"ასაკი" => 20
),
array (
"სახელი" => "ვახტანგ",
"გვარი" => "ტარასაშვილი",
"ასაკი" => 40
)
);
print $mas ['1'] ['სახელი'];
?>
```

შედეგად ეკრანზე მივიღებთ წარწერას „სანდრო“, მაშასადამე ჩვენ მივმართეთ მასივ
ის იმ ელემენტს (იმ ქვემასივს) რომლის რიგითი ნომერი გახლავთ 1 და ეკრანზე გამოვ
იტანეთ ამ მასივის ელემენტ „სახელი“ -
ის მნიშვნელობა, რომელიც ჩვენს შემთხვევაში წარმოადგენს ტექსტურ მონაცემს „სან
დრო“ პროგრამამაც არ დააყოვნა და შესაბამისი მონაცემი ამობეჭდა ეკრანზე.
შემდგომ ეტაპზე განვიხილოთ მასივების გამოყვანის მაგალითები და წესები,
ამისათვის გამოვიყენებთ ფუნქციას foreach(). სწორედ ამ ფუნქციის საშუალებით
ხორციელდება მასივების გამოყვანა.

მასივების გამოყვანა

მასივის გამოყვანა გახლავთ მასივის ყველა ელემენტის ეკრანზე ამობეჭდვა, თავისთა
ვად მასივის ელემენტების ერთად გამოსახვა მარტივ და რთულ მასივებში სხვადასხვა
ნაირად ხდება:
მაგალითი 29:

```
<?php
$masivi = array (
```

```
"სახელი" => "დავით",
"გვარი" => "სამხარაძე",
"პროფესია" => "პროგრამისტი"
);
foreach ($masivi as $key => $value)
{
print $key = $value;
print "<br>";
}
?>
```

პროგრამის გაშვების შემდეგ ეკრანზე გამოისახება მასივის სამივე ელემენტის მნიშვნ ელობები. მაშასადამე ფუნქცია **foreach()** -
ის საშუალებით ჩვენ მოვახდინეთ მასივის ელემენტთა გამოყვანა. **$key** და **$value** ჩვენ ს მიერ შემოტანილი ცვლადებია, სადაც პირველი ცვლადი **$key** აღნიშნავს მასივის ელ ემენტებს, ხოლო **$value** მასივის ელემენტების მნიშვნელობებს. შემდგომ მაგალითში გ ანვიხილოთ რთული მასივის გამოყვანის წესი.

<u>მაგალითი 30:</u>

```
<?php
$masivi = array (
array (
"სახელი" => "ნიკა",
"გვარი" => "გოგიჩაიშვილი",
"ასაკი" => 30
),
array (
"სახელი" => "სანდრო",
"გვარი" => "სამხარაძე",
"ასაკი" => 20
),
array (
"სახელი" => "ვახტანგ",
"გვარი" => "ტარასაშვილი",
"ასაკი" => 40
)
);
foreach ($masivi as $value)
{
foreach ($value as $key => $final_value)
{
print "$key = $final_value<br>";
```

```
}
print "<hr>";
}
?>
```

როგორც ხედავთ რთული მასივის გამოსაყვანად საჭიროა მისი დაყვანა მარტივ მასივ ამდე.

მასივების გაერთიანება.

მასივების გაერთიანებას ემსახურება ჩანაწერი **array_merge()**
მასივების გაერთიანება სხვადასხვა მომენტებში ძალზედ ხელსაყრელი და მოსახერხე ბელია. პროგრამის წერის დროს შესაძლებელია ნებისმიერი რაოდენობის მასივების ე რთ, უფრო დიდ მასივად გაერთიანება.
მაგალითი 31:

```
<?php
$mas_1 = array ("a", "b", "c");
$mas_2 = array (1, 2, 3, 4, 5);
$mas_3 = array_merge ($mas_1, $mas_2);
foreach ($mas_3 as $key => $value)
{
print $key = $value;
print "<br>";
}
?>
```

მასივში ახალი ელემენტის ჩამატება

მასივში ახალი ელემენტის ჩამატებას უზრუნველყოფს ჩანაწერი **array_push()**
მასივში ახალი ელემენტის ჩამატება შესაძლებელია ნებისმიერი რაოდენობით.
მაგალითი 32:

```
<?php
$mas_1 = array ("a", "b", "c");
$mas_2 = array_push ($mas_1, 0, 1, 2);
foreach ($mas_1 as $key => $value)
{
print $key = $value;
}
?>
```

მასივების დახარისხება

მასივების დახარისხება, ანუ როგორც ხშირად ეძახიან -
სორტირება ხორციელდება რამოდენიმე გზით.

sort() - გამოიყენება მარტივი მასივის სორტირებისათვის
assort() - გამოიყენება ასოცირებული მასივის სორტირებისათვის
ksort() - მასივის სორტირება მისი ელემენტების მიხედვით (ველების სახელებით)
მაგალითი 33:

```
<?php
$masivi = array ("f", "g", "a", "d");
sort ($masivi);
foreach ($masivi as $value)
{
print "$value<br>";
}
?>
```

შედეგად სიმბოლოები დახარისხდება ანბანის მიხედვით. ეხლა განვიხილოთ მომდე ვნო მაგალითი.
მაგალითი 34:

```
<?php
$masivi = array (
5 => "microsoft.com",
3 => "apple.com",
2 => "twiter.com",
4 => "youtube.com",
1 => "facebook.com"
);
ksort ($masivi);
foreach ($masivi as $key => $value)
{
print "$key = $value";
print "<br>";
}
?>
```

შედეგად მასივის ელემენტთა მნიშვნელობების სორტირება მოხდება ველების სახელ ების მიხედვით. გარდა მასივების სორტირებისა, შეერთებისა და მასივში ახალი ელემ ენტების ჩამატებისა, შეგვიძლია მაივიდან ქვემასივის მიღება და მასივიდან პირველი ელემენტის ამოგდება:
array_slice() - მასივიდან ქვემასივის მიღება.
array_shift() - მასივიდან ელემენტის ამოგდება.

19. ფუნქციები.

პროგრამირების დროს ფუნქციების გამოყენება ყველაზე ხშირად უწევთ პროგრამისტ ებს, ასე რომ ფუნქციების არცოდნა არ ეპატიება არა მოყვარულ პროგრამისტს, არამედ დამწყებსაც. პროფესიონალზე რომ აღარაფერი ვთქვათ.

მოდით განვმარტოთ ფუნქციის არსი, კერძოდ, ფუნქცია არის რაღაც სამუშაოს შესრუ ლებისათვის გაწერილი მოქმედებათა კრებული, უფრო ზუსტად, რომ ვთქვათ, ფუნქც იას უნდა მიეწოდოს მასალა,
„ნედლეული“ და ფუნქციამ უნდა გადაამუშაოს იგი, შედეგად კი მივირებთ უკვე შესრ ულებულ დავალებას.
ფუნქციის გამოცხადება **php** -
ში ხდება ჩანაწერით **function()** ხოლო შემდგომ ეტაპზე მას უნდა მიენიჭოს სახელი და არგუმენტი, ანუ სწორეს ის „ნედლეული“, რომელიც უნდა გადაამუშაოს ფუნქციამ.
<u>მაგალითი **35:**</u>

```
<?php
function funqcia ($txt)
{
print "<h1>$txt</h1>";
}
funqcia ("ტექსტი");
funqcia ("ტექსტი");
?>
```

მოდით განვიხილოთ **35** -
ე მაგალითში წარმოდგენილი ფუნქცია. მაშასე ჩვენ გამოვაცხადეთ ფუნქცია, შემდეგ ე ტაპზე მას მივანიჭეთ სახელი **funqcia** ასევე შევიტანეთ მასში არგუმენტი **$txt** ცვლადის სახით და მიუთითეთ ის პირობა, რაც უნდა შეგვისრულოს ფუნქციამ. ჩვენს შემთხვევ აში ფუნქციას ევალება ტექსტი დაწეროს დიდი შრიფტით და ყოველი ახალი ტექსტი დაიწყოს ახალი ხაზიდან. ამის შემდეგ მივმართავთ ფუნქციას სახელით და ვაწვდით მასშესაბამისი არგუმენტის მნიშვნელობას, ანუ როგორც უკვე ავღნიშნეთ „ნედლეულ ს“, ჩვენს შემთხვევაში კი „ნედლეული“ გახლავთ ფრაზა „ტექსტი“ შედეგად ფუნქცია იღებს ამ მონაცემს, ამუშავებს მას პირობის მიხედვით და შედეგი ეკრანზე გამოაქვს.

ყურადღება მიაქციეთ ჩანაწერებს, კერძოდ, ჩვენ გამოყენებული გვაქვს:
funqcia ("ტექსტი"); და არა **print funqcia ("ტექსტი");** ანუ ამ შემტხვევაში **print** ოპერატორის გამოყენება ზედმეტია, ვინაიდან დ ა რადგანაც ჩვენ ფუნქციის პირობაში უკვე მითითებული გვაქვს **print "<h1>$txt</h1>";** შესაბამისად ფუნქციას როცა მიმართავთ ის ავტომატურად ამობეჭდა ვს ტექსტს, რადგანაც ეს მის პირობაშია მითითებული და კვლავ ხელმეორედ **print** ოპე რატორის მიწერა აღარაა საჭირო.

მაგალითი 36:

```
<?php
phpinfo ();
?>
```

ამ ფუნქციის საშუალებით მიიღებთ ინფორმაციას თქვენს **php** -ზე.
მაგალითი 37:

```
<?php
print time ();
?>
```

ამ ფუნქციის საშუალებით კი თქვენ ეკრანზე ამობეჭდავთ მიმდინარე დროს (გაითვა ლისწინეთ, რომ კომპიუტერი დროს ითვლის მიკროწამებში).
მომდევნო ეტაპზე განვიხილოთ ფუნქცია, რომლის საშუალებითაც მოვახდენთ ორი რ იცხვის შეკრებას.
მაგალითი 38:

```
<?php
function funqcia ($pirveli, $meore)
{
$shedegi = $pirveli + $meore;
print $shedegi;
}
funqcia (2, 3);
?>
```

პროგრამის შესრულების შემდეგ ეკრანზე დაიბეჭდება ციფრი **5**, რადგანაც **2+3=5.**
მაგალითი 39:

```
<?php
function funqcia ($pirveli, $meore)
{
$shedegi = $pirveli + $meore;
return $shedegi;
}
print funqcia (2, 3);
?>
```

ეს მაგალითი თითქმის იგივეა რაც წინა, განსხვავება ისაა, რომ აქ გამოყენებულია ჩან აწერი **return** რომელიც უზრუნველყოფს რეზულტატის მიღებას, ხოლო ბოლოს, ფუნქ ციის მიმართვისას გამოყენებულია ჩანაწერი **print**, მისი მითითება აუცილებელია, რა

დგანაც იგი არაა მითითებული ფუნქციის პირობაში და მის გარეშე რეზულტატის ეკრანზე გამოტანა ვერ მოხერხდება.

ფუნქციის წერის დროს უნდა გავითვალისწინოთ „ცვლადის ხილვადობის პრინციპი“, კერძოდ ფუნქციაში გამოცხადებული ცვლადებზე პროგრამის სხვა, (ფუნქციის გარეთ არსებული) არიდან მიმართვა არ ხდება. თუკი გვსურს, რომ ფუნქციაში გამოცხადებულ ცვლადებს მივმართოთ პროგრამის ნებისმიერი წერტილიდან, საჭიროა მათ მივცეთ გლობალური ცვლადების სახე, კერძოდ მათი გამოცხადებისას უნდა გამოვიყენოთ ჩანაწერი **global** (მაგალითად:**$cvladi = 10; global $cvladi** - შედეგად ცვლადს მიეცა გლობალური სახე).

20. ობიექტები, კლასები.

პროგრამირებაში ობიექტების ცნების შემოტანით დაპროგრამების ენები გაცილებით მრავალფუნქციონალური გახდა. ობიექტები თავის თავზე იღებენ ყველა იმ ნიუანსს და „შავ სამუშაოს“ რასაც პროგრამის წერის დროს ვერ ასცდები. ობიექტების გამოყენებამდე საჭიროა გამოცხადდეს კლასი, თვით კლასი კი შეიძლება აღვიქვათ ერთ დიდ ჯგუფად, სადაც განისაზღვრება ობიექტთა თვისებები. კლასში თავისუფლად შეიძლება გამოვიყენოთ ფუნქციებიც.

მაგალითი 40:

```
<?php
class klasi
{
var $cvladi;
}
$obieqti1 = new klasi ();
$obieqti2 = new klasi ();
$obieqti1 -> cvladi = "სახელი";
$obieqti2 -> cvladi = " გვარი";
print $obieqti1 -> cvladi;
print $obieqti2 -> cvladi;
?>
```

პროგრამის გაშვების შემდგომ ბრაუზერის ფანჯარაში გამოისახება სტრიქონი „სახელი გვარი“. ეხლა მოდით განვიხილოთ ეს პროგრამა: დასაწყისისათვის ჩვენ გამოვაცხადეთ კლასი, რომლის სახელი გახლავთ **klasi** შემდგომ მასში გამოვაცხადეთ ცვლადი **var $cvladi** (აუცილებელია ცვლადს წინ მიეთითოს ჩანაწერი **var**) და შემოვიტანეთ პროგრამაში ორი ობიექტი **$obieqti1** და **$obieqti2** შემდგომ კი სწორედ ამ ობიექტების საშუა

ლებით მოხდა ცვლადზე მნიშვნელობისმინიჭება, ჩვენს შემთხვევაში კი კლასში გამო
ცხადებულმა ცვლადმა მიიღო ორი მნიშვნელობა:
1. $obieqti1 -> cvladi = "სახელი"; და **2. $obieqti2 -> cvladi = " გვარი";**
კარგად დააკვირდით ოპერატორს -
> სწორედ მისი საშუალებით ხდება ობიექტიდან თვისებაზე გადასვლა.

შემდგომ ეტაპზე განვიხილოთ ისეთი კლასი, სადაც გამოვიყენებთ ფუნქციას. ფუნქცი
ის გამოცხადება ალბათ გახსოვთ, თუ არა და მაშინ კიდევ ერთხელ გადაავლეთ თვალ
ი თეორიულ და პრაქტიკულ მასალას ფუნქციებზე. მაშასე, ვქმნით კლასს, სადაც შემდ
გომ გამოვაცხადებთ ფუნქციას, ფუნქციის გამოცხადების შემდეგ კი შევეცდებით ფუნ
ქციისათვის არგუმენტის მინიჭებას ობიექტის საშუალებით. ქვემოთ მოყვანილ მაგალ
ითში ფუნქციასთანერთად კლასის შიგნით მოთავსებულია ცვლადიც, ყურადღება გაა
მახვილეთ იმ ფაქტზე, რომ ობიექტიდან თვისებაზე გადასვლა ხდება ფუნქციაზე გასვ
ლით და არა ცვლადით, როგორც ეს წინა მაგალითში იყო.
ეხლა კი გადავიდეთ უშუალოდ მაგალითზე:
<u>მაგალითი **41:**</u>

```
<?php
class klasi
{
var $cvladi;
function funqcia ($cvladi)
{
print "<h1><u>$cvladi</u></h1>";
}
}
$obieqti1 = new klasi ();
$obieqti1 -> funqcia ("მაშასე ვაგრძელებთ კლასების შესწავლას!");
?>
```

პროგრამის გაშვების შემდეგ ბრაუზერის ფანჯარაში გამოისახება დიდი ზომის და ქვ
ემოდან ხაზგასმული ტექსტი, კერძოდ ტექსტის დაფორმატება მოხდა უშუალოდ ფუნ
ქციაში, ხოლო არგუმენტის მნიშვნელობის მინიჭება ობიექტის საშუალებით, შემდგო
მ ეტეპზე თუ გვინდა გამოვაცხადებთ არა ერთ, არამედ უამრავ ობიექტს და არგუმენტ
ს მივანიჭებთ არა ერთ, არამედ რამოდენიმე მნიშვნელობას.
<u>მაგალითი **42:**</u>

```
<?php
class klasi
{
var $saxeli = "ნინი";
function funqcia ()
```

```
{
print "გამარჯობა! მე მქვია $this->saxeli";
}
}
$obieqti1 = new klasi ();
$obieqti1 -> funqcia ();
?>
```

მოცემულ მაგალითში გამოყენებულია ჩანაწერი **$this->saxeli** ცვლადი **$this** ჩანაწერის საშუალებით ჩვენ ფუნქციას მივანიჭედ არგუმენტად კ ლასში გამოცხადებული ცვლადის მნიშვნელობა, შედეგად კი ეკრანზე ამოიბეჭდება ს ტრიქონი „გამარჯობა! მე მქვია ნინი“

ეხლა გავეცნოთ კონსტრუქციებს, რა არის კონსტრუქცია?

php კონსტრუქციას უწოდებენ ისეთ კლასებს, რომლის დროსაც კლასის სახელი და ამ კლასში გამოცხადებული ფუნქციის სახელი ერთნაირია. წინა მაგალითებს თუ დააკვი რდებით შენიშნავთ, რომ კლასის სახელად არჩეული გვაქვს სახელი **klasi** ხოლო ამ კლ ასში გამოცხადებულ ფუნქციას მივანიჭეთ სახელი **funqcia** ეხლა კი მოდით განვიხილ ოთ ისეთი მაგალითი, სადაც კლასის სახელი და ფუნქციისსახელი ერთიდაიგივე იქნე ბა - ანუ მოვიყვანოთ კონსტრუქციის მაგალითი:

<u>მაგალითი **43:**</u>

```
<?php
class konstruqcia
{
var $saxeli;
function konstruqcia ($saxeli2 = "ქეთინო")
{
$this->saxeli = $saxeli2;
}
function funqcia ()
{
print "გამარჯობა! ჩემი სახელია $this->saxeli<br>";
}
}
$obieqti1 = new konstruqcia ();
$obieqti2 = new konstruqcia ("თამუნია");
$obieqti3 = new konstruqcia ("ნიკუშა");
$obieqti1 -> funqcia ();
$obieqti2 -> funqcia ();
$obieqti3 -> funqcia ();
?>
```

მომდევნო ეტეპზე განვიხილოთ მემკვიდრეობით გადაცემის მაგალითი. მემკვიდრეობით გადაცემა, ანუ მშობელი კლასის თვისებების შვილობილ კლასზე გადაცემა ხორციელდება extends ჩანაწერით:
მაგალითი 44:

```
<?php
class konstruqcia
{
var $saxeli = "ნიკუშა";
function konstruqcia ($saxeli2)
{
$this->saxeli = $saxeli2;
}
function funqcia ()
{
print "გამარჯობა $this->saxeli<br>";
}
}
class klasi extends konstruqcia
{
}
$obj1 = new klasi ("ნიკუშა");
$obj1 -> funqcia ();
?>
```

კლასების შესწავლა ამ მაგალითით დასრულებულია, ეხლა დროა გადავიდეთ მომდევნო თემაზე.

21. არითმეტიკული ფუნქციები.

კვლავ დაუბრუნდეთ ფუნქციებს და განვიხილოთ ზოგიერთი უკვე წინასწარ გამზადებული ფუნქციის მაგალითი, კერძოდ, მოდით განვიხილოთ რთული არითმეტიკული ფუნქციები.
არითმეტიკული ფუნქციები გამოიყენება რთული მათემატიკური ფორმულების გამოსაყვანად. ქვემოთ ჩამოთვლილია სხვადასხვა არითმეტიკული ფუნქციები:

sin(x)	სინუსი	sqrt(x)	კვადრატული ფესვი
cos(x)	კოსინუსი	pow((x),y)	ახარისხება
tan(x)	ტანგესი	max(x,y)	მაქსიმალური

abs(x)	მოდული	min(x,y)	მინიმალური
exp(x)	ექსპონენტა	floor(x)	დამრგვალება ნაკლებობით
log(x)	ლოგარითმი	ceiling(x)	დამრგვალება მეტობით
log10(x)	ათაბითი ლოგარითმი	round(x,y)	ფორმატირება
pi	რიცხვი პი, 3.14	sign(x)	არგუმენტის ნიშანი

ეხლა განვიხილოთ რამოდენიმე არითმეტიკული ფუნქცია პრაქტიკულ ასპექტში.
მაგალითი 45:

```
<?php
print exp(12); // ექსპონენტა 12 -ის
print "<br>";
print pow((2),5); // ორი მეხუთე ხარისხში
print "<br>";
print sqrt(9) + sqrt(4); // კვადრატული ფესვი 9 -დან პლიუს ფესვი 4 -დან
print "<br>";
print exp(pow((2),3)); // ექსპონენტა ორის მესამე ხარისხისა
print "<br>";
print max(2, 4, 5, 12, 6); // მაქსიმუმი ჩამოთვლილთა შორის
print "<br>";
print pow((2),5) + sqrt(81); // 2 -ის მეხუთე ხარისხს პლიუს ფესვი 81 -დან
print "<br>";
print abs(-55); // მოდული -55 ის.
?>
```

ჩვენ განვიხილეთ მარტივი მაგალითები, რათა თქვენთვის ადვილი ყოფილიყო არი თმეტიკული ფუნქციების გააზრება. პროგრამირების უფრო რთუ ეტეპზე შესაძლებე ლია შედარებით უფრო რთული მათემატიკური ფორმულების გამოყვანა, ისეთების მ აგალითად როგორიცაა კვადრატული განტოლება, სისტემების ამოხსნა და სხვა.
ზემოთ ჩამოთვლილ რთულ მათემატიკურ ოპერაციებს ჩვენ აღარ განვიხილავთ, გამ ომდინარე იქედან, რომ უშუალოდ ვებ-
პროგრამირებასთან გვაქვს საქმე, ანუ შემდგომ ეტაპზე დიდი ყურადღება დაეთმობა ფაილების სერვერზე გადასაგზავნ სკრიპტებს, რეგისტრაციისა და ავტორიზაციის სკ რიპტებს, სერვერის და მომხმარებლების შესახებ ინფორმაციის მიღებას და სხვა. ხოლ ო თუკი თქვენ გსურთ გამოთვალოთ შედარებითრთული მათემატიკური ფორმულებ ი, ვფიქრობ რომ არ უნდა გაგიჭირდეთ, რადგანაც უკვე იცნობთ არითმეტიკულ ფუნქ ციებს და შესაბამისად იცით მათი გამოყენებაც.

22. ინფორმაცია სერვერზე და მომხმარებელზე.

სერვერის შესახებ ინფორმაციის მისაღებად, ასევე მომხმერებელთა იდენტიფიკაციი სათვის გთავზობთ გაეცნოთ შემდეგ ფუნქციებს, რომელიც უკვე წინასწარაა განსაზღვ რული:

$_SERVER['DOCUMENT_ROOT'] გზა დოკუმენტამდე
$_SERVER['GATEWAY_INTERFACE'] გეითვეი ინტერფეისი
$_SERVER['HTTP_ACCEPT'] აპლიკაციის ტიპი (მატარებლის ტიპი)
$_SERVER['HTTP_ACCEPT_CHARSET'] არჩეული კოდირების რეჟიმი
$_SERVER['HTTP_ACCEPT_LANGUAGE'] მითითებული ენა
$_SERVER['HTTP_CONNECTION'] ინფორმაცია http შეერთების შესახებ
$_SERVER['HTTP_HOST'] ჰოსტის მისამართი
$_SERVER['HTTP_USER_AGENT'] ინფორმაცია მომხმარებელზე
$_SERVER['REMOTE_ADDR'] სერვერის ინტერნეტ პროტოკოლის მისამართი. ip
$_SERVER['REMOTE_PORT'] პორტი
$_SERVER['REQUEST_METHOD'] მოთხოვნის მეთოდი (get, post)
$_SERVER['REQUEST_URI'] მოთხოვნილი url მისამართი
$_SERVER['SERVER_ADMIN'] სერვერის ადმინისტრატორი
$_SERVER['SERVER_NAME'] სერვერის სახელი
$_SERVER['SERVER_PORT'] პორტი (სტანდარტული მეოთხმოცე პორტი)
$_SERVER['SERVER_PROTOCOL'] პროტოკოლი (http პროტოკოლი)
$_SERVER['SERVER_SOFTWARE'] პროგრამული უზრუნველყოფა (მაგ apache)

იმისათვის რომ ინფორმაცია ამოიბეჭდოს ეკრანზე საჭიროა ზემოთ ჩამოთვლილ ფუნ ქციებს წინ ჩაუწეროთ print ან echo ჩანაწერი.
მაგალითი 46:

```
<?php
print $_SERVER['DOCUMENT_ROOT'];
print "<br>";
print $_SERVER['HTTP_USER_AGENT'];
print "<br>";
print $_SERVER['SERVER_NAME'];
print "<br>";
print $_SERVER['SERVER_PORT'];
print "<br>";
print $_SERVER['SERVER_PROTOCOL'];
print "<br>";
print $_SERVER['SERVER_SOFTWARE'];
```

```
?>
```

შედეგად მივიღებთ ინფორმაციას ჩვენს კომპიუტერზე (ან დაშორებულ სერვერზე).

23. სათაურის ტეგები php -ში.

ალბათ ყველას გახსოვთ html -დან meta ტეგები. meta ტეგების გამოყენება php -ში ხდება header () ფუნქციით.

```
header ('content-type: text/html; charset=utf-8'); // კოდირება
header ('description: dokumentis shesaxeb informacia');
header ('keywords: sadziebo sityvebi, sadziebo sistemisatvis');
header ('pragma: no-chache'); // კეშირების აკრძალვა
header ('chache-control: no-chache'); //კეშირების აკრძალვა
header ('location: http://google.com/');
```

ყურადღება გაამახვილეთ ჩანაწერზე: header ('location: http://google.com/'); კერძოდ, ამ ჩანაწერის საშუალებით ხორციელდება კლიენტის გადართვა სხვა ვებ-ფურცელზე (ჩვენს შემთხვევაში google.com -ზე) ეს ჩანაწერი თითქმის იგივეა, რაც:
<meta http-equiv="refresh" content="პარამეტრი"/ > -რომელიც html -ში გამოიყენება.

24. დრო php -ში.

მიმდინარე დროის ეკრანზე გამოსატანად გამოვიყენებთ ფუნქციას date () რომელსაც გააჩნია შემდეგი პარამეტრები:

```
a ფორმატირების დროს გამოდის ინგლისური am, pm (A – AM , PM)
d დღე, შემდეგი ფორმატირებით: 01, 02 ... 31
F თვე, შემდეგი ფორმატირებით: Aprel, Jun
g საათის მაჩვენებელი, შემდეგი ფრომატირებით: 1, 5, 7 ... 11 (12 ციფრიანი)
G საათის მაჩვენებელი, შემდეგი ფორმატით: 1, 2, 5 ... 23, 00 (24 ციფრიანი)
h საათის მაჩვენებელი, შემდეგი ფორმატით: 01, 02 ... 11, 12 (12 ციფრიანი)
H საათის მაჩვენებელი, შემდეგი ფორმატით: 01, 02 ... 23, 00 (24 ციფრიანი)
i წუთების მაჩვენებელი, შემდეგი ფორმატით: 01, 02, ... 60
I კვირის დღე (კვირის დღის მაჩვენებელი)
L ნაკიანი წელი 1, დანარჩენ შემთხვევაში 0
m თვე, შემდეგი ფორმატით: 01, 02, 03 ... 12
M თვე, შემდეგი ფორმატით: jun, nov
```

n თვე, შემდეგი ფორმატით: 1, 2, 3, 4 ... 12
t დღეების რაოდენობა თვეში.
s წამების მაჩვენებელი, შემდეგი ფორმატით: 01, 02, 03 ... 60
S წამების მაჩვენებელი, შემდეგი ფორმატით: 1, 2, 3 ... 60
T არჩეული სასაათო სარტყელი.
U წამების რაოდენობა საუკუნის დასაწყისიდან.
w კვირის დღეები, შემდეგი ფორმატით: 1, 2, 3 ... 7
y წელი, შემდეგი ფორმატით: 2007, 2008 ... 2012
Y წელი, შემდეგი ფორმატით: 07, 08 ... 12
z დღეების რაოდენობა წელიწადში.

ეხლა მოვიყვანოთ კონკრეტული მაგალითი იმისა, თუ როგორ უნდა ამოვბეჭდოთ ეკრანზე მიმდინარე დრო:

მაგალითი 47:

```
<?php
print (date("H სთ : i წთ : s წმ -- d : F : Y"));
?>
```

შედეგად ეკრანზე ამოიბეჭდება მიმდინარე დრო, თუკი თქვენ ყოველ წამში განაახლებთ ვებ-ფურცელს, დაინახავთ თუ როგორ იცვლება დროც. ეკრანზე ამობეჭდილი დრო სინქრონულია თქვენს კომპიუტერზე დაყენებული დროისა, ხოლო თუ სერვერთან მუშაობთ, მაშინ სერვერისა.

25. ფორმებთან მუშაობა.

ფორმებთან მუშაობის პროცესში ჩვენ განვიხილავთ უმარტივესი ფორმების დამუშავებისათვის საჭირო პროგრამებს, შემდგომ უფრო რთულ ეტაპზე განვიხილავთ ვებგვერდზე სტუმართა წიგნის შექმნისათვის საჭირო სკრიპტს, ვინაიდან და რადგანაც ჩვენ mysql -ზე მონაცემთა ბაზების აგების მაგალითებს ამ სახელმძღვანელოში არ განვიხილავთ, შესაბამისად გამოვიყენებთ ისეთ სკრიპტებს რომელთაც მონაცემების შენახვისათვის ან წაკითხვისათვის არ დაჭირდებათ mysql მონაცემთა ბაზების გამოყენება, ხოლო მის ნაცვლად გამოვიყენებთ უბრალო ტექსტურ ფაილს, სადაც შევინახავთ ინფორმაციას, ანუ ავაგებთ მასში პატარა მონაცემთა ბაზას.

მაგალითი 48:

დასაწყისისათვის განვიხილოთ უმარტივესი ფორმის დამუშავების პროგრამა, ამისათვის დაგვჭირდება შევქმნათ ორი ფაილი: index.html და program.php

index.html ფაილში ვწერთ შემდეგი სახის მარტივ ფორმას:

```
<form action="program.php" method="get">
<input type="text" name="txt">
<input type="text" name="txt2">
<input type="submit" value="enter">
</form>
```

program.php ფაილში კი ვწერთ შემდეგ პროგრამას:

```
<?php
print $txt;
print "<br>";
print $txt2;
?>
```

მივმართოთ ჩვენს index.html ფაილს (რაღათქმაუნდა http პროტოკოლით და არა პირდაპირ) შემდეგ ტექსტურ ველში ჩავწეროთ ნებისმიერი ფრაზა და გავაგზავნოთ იგი. შედეგად program.php ფაილი დაამუშავებს ფორმას და თქვენს გაგზავნილ მონაცემს ა მობეჭდავს ეკრანზე. კიდევ ერთხელ გავამახვილოთ ყურადღება იმაზე, რომ ტექსტუ რ ველზე მიმართვა ხდება მისი სახელით. ($txt ჩვენს შემთხვევაში.)

26. ტექსტებთან სამუშაო ფუნქციები.

განვიხილოთ რამოდენიმე საინტერესო ფუნქცია, რომელიც ნამდვილად გამოგადგებათ პროგრამირების პროცესში, კერძოდ კი ტექსტებთან მუშაობისას.

empty () - გამოიყენება ცარიელი ტექსტური ველის აღსანიშნავად, ანუ თუ ფორმაში არსებული ტექსტური ველი ცარილია, შეგვიძლია ლოგიკურ ჩანაწერებში მიუთითოთ პროგრამას, თუ როგორ მოიქცეს იგი, როცა ტექატური ველი ცარიელია.

preg_match () - ამ ფუნქციის საშუალებით შეგვიძლია ავკრძალოთ ზოგიერთი სიმბოლოს, ან სიმბოლოების გამოყენება, ან მოვითხოვოთ კონკრეტული სიმბოლო. მაგალითად ტექატური ველისათვის, სადაც უნდა შევიყვანოთ მეილი, მოვითხოვოთ სიმბოლო @ -ის ჩაწერა, რათა მომხმარებელმა ტყუილი ელ ფოსტა არ შეიყვანოს ტექსტურ ველში.

preg_replace () - ამ ფუნქციის საშუალებით შეგვიძლია ავკრძალოთ ტექსტში პრობელების გამოყენება, ეს ფუნქცია განსაკუთრებით ხელსაყრელია საიტზე ავტორიზაციის დროს, რათა მომხმარებელმა პაროლის შეყვანის დროს არ გამოიყენოს პრობელი.

str_replace () - ამ ფუნქციის საშუალებით შეგვიძლია კონკრეტული სიმბოლოების აკრძალვა ტექსტის წერისას, ეს ფუნქცია განსაკუთრებით ხელსაყრელია მაშინ

როდესაც გვსურს ჩათებში და ფორუმებზე ავკრძალოთ უცენზურო სიტყვები და რეკლამები, რომლებიც შეიცავენ სხვა ვებგვერდის ბმულებს.
strtolower () - ამ ფუნქციის საშუალებით შესაძლებელია რეგისტრის უგულვებელყოფა, ანუ შეგვიძლია პატარა და დიდ სიმბოლოებს შორის მოვხსნათ განსხვავება და პროგრამამ პატარა და დიდი სიმბოლოები ორივე ერთნაირად აღიქვას, მაგალითად ინგლისური ან რუსული სიმბოლოებით წერისას. ეს ფუნქცია განსაკუთრებით ხელსაყრელია საძიებო სისტემისთვის, რათა სისტემამ დიდ და პატარა სიმბოლოებში განსხვავება ვერ იგრძნოს და სწრაფად მოიძიოს ინფორმაცია.
ეხლა განვიხილოთ თითოეული ამ ფუნქციის გამოყენების კონკრეტული მაგალითები, რათა შემდგომ უფრო გაგვიადვილდეს მათთან მუშაობა.
მაგალითი 49:

ამ მაგალითის განსახილველად საჭიროა შევქმნათ ორი ფაილი, შემდეგი სახელწოდებებით index.html და program.php

ფაილი index.html შეიცავს შემდეგი ტიპის მარტივ ფორმას:
```
<form action="program.php" method="get">
<input type="text" name="txt">
<input type="submit" value="enter">
</form>
```

ფაილი program.php შეიცავს შემდეგი ტიპის პროგრამულ კოდს:
```
<?php
if(empty($txt))
{
print "ტექსტური ველი არაა შევსილი!";
}
else
{
print "$txt";
}
?>
```

თუკი ტექსტური ველი ცარიელია შესაბამისად ამოქმედდება ლოგიკური ოპერატორი if რომელიც გამოიტანს ეკრანზე შესაბამის ტექსტურ მონაცემს, ხოლო სხვა დანარჩენ შემთხვევაში, ანუ როცა ტექსტური ველი შევსილია, პროგრამა დაასრულებს ნაკადის მართვას else ლოგიკური ოპერატორით და მონიტორზე გამოისახება ის სიმბოლო, ან სიმბოლოები რაც თქვენ შეიყვანეთ ტექსტურ ველში.
ეხლა განვიხილოთ მომდევნო მაგალითი, რომლის საშუალებითაც შევძლებთ რეგისტრის უგულვებელყოფას.
მაგალითი 50:

ამ მაგალითის განსახილველად საჭიროა ისევ შევქმნათ ორი ფაილი, შემდეგი სახელწოდებებით index.html და program.php

ფაილი index.html შეიცავს შემდეგი ტიპის მარტივ ფორმას:

```
<form action="program.php" method="get">
<input type="text" name="txt">
<input type="submit" value="enter">
</form>
```

ფაილი program.php შეიცავს შემდეგი ტიპის პროგრამულ კოდს:

```
<?php
$txt = preg_replace ("/\s+/" , "" , $txt);
print $txt;
?>
```

თუკი თქვენ ტექსტური ველის შევსებისას გამოიყენებთ ერთ ან რამოდენიმე პრობელს, იგი უგულვებელყოფილი იქნება, შედეგად ეკრანზე ამოიბეჭდება ტექსტურ ველში შეყვანილი სიმბოლოები პრობელის გარეშე.

მაგალითი 51:

ამ მაგალითის განსახილველად საჭიროა კვლავინდებურად შევქმნათ ორი ფაილი, შემდეგი სახელწოდებებით index.html და program.php

ფაილი index.html შეიცავს შემდეგი ტიპის მარტივ ფორმას:

```
<form action="program.php" method="get">
<input type="text" name="txt">
<input type="submit" value="enter">
</form>
```

ფაილი program.php შეიცავს შემდეგი ტიპის პროგრამულ კოდს:

```
<?php
$txt = strtolower ($txt);
print $txt;
?>
```

შედეგად ტექსტურ ველში შეტანილი სიმბოლოები, დიდიც და პატარაც აღიქმება პატარა სიმბოლოებად, ანუსიმბოლოები უგრძნობი გახდნენ რეგისტრის მიმართ.

მაგალითი 52:

ამ მაგალითის განსახილველად საჭიროა კვლავინდებურად შევქმნათ ორი ფაილი, შემდეგი სახელწოდებებით index.html და program.php

ფაილი index.html შეიცავს შემდეგი ტიპის მარტივ ფორმას:

```
<form action="program.php" method="get">
<input type="text" name="txt">
<input type="submit" value="enter">
</form>
```

ფაილი program.php შეიცავს შემდეგი ტიპის პროგრამულ კოდს:

```
<?php
if(preg_match("/[@][.]/"))
{
print "ელ-ფოსტის მისამართი არასწორია";
}
else
{
print "ელ-ფოსტის მისამართი სწორია";
}
?>
```

შედეგად პროგრამა შეამოწმებს მეილის მისამართი სწორად იქნა თუ არა შეყვანილი ტექსტურ ველში. თუკი მეილის მისამართი შეიცავს ნიშანს @ -ს და წერტილს, მაშინ პროგრამა იხელმძღვანელებს if ოპერატორში არსებული პირობით, წინააღმდეგ შემთხვევაში აქტიურდება ოპერატორი else.

ზემოთ მოყვანილ მაგალითში შეიძლება მიუთითოთ სხვა ჩანაწერებიც, მაგალითად ჩანაწერი [0-9] მიუთითებს, რომ აუცილებელია ციფრების შეტანა ტექსტურ ველში, ჩანაწერი [a-z} მიუთითებს, რომ საჭიროა ანბანის სიმბოლოების შეყვანა, ანუ აკრძალულია სხვა სიმბოლოების გამოყენება, მაგალითად მძიმის, წერტილის, ფრჩხილების, დახრილი ხაზის და სხვა.

მომდევნო მაგალითში მოდით განვიხილოთ საინტერსო რამ, კერძოდ მოდით ავკრძალოთ სიტყვა „php“ -ს გამოყენება ტექსტში, ანუ თუკი მომხარებელი ტექსტურ ველში შეიტანს სიტყვას „php“, მაშინ სწრაფადვე მოხდეს მისი აკრძალვა.

<u>მაგალითი 53:</u>

ამ მაგალითის განსახილველად საჭიროა კვლავინდებურად შევქმნათ ორი ფაილი, შემდეგი სახელწოდებებით index.html და program.php

ფაილი index.html შეიცავს შემდეგი ტიპის მარტივ ფორმას:

```
<form action="program.php" method="get">
<input type="text" name="txt">
```

```
<input type="submit" value="enter">
</form>
```

ფაილი program.php შეიცავს შემდეგი ტიპის პროგრამულ კოდს:

```
<?php
if($txt = str_replace('php' , '' , $txt))
{
print "ამ სიტყვის გამოყენება აკრძალულია";
}
elseif($txt = str_replace('PHP' , '' , $txt))
{
print "ამ სიტყვის გამოყენება აკრძალულია";
}
else
{
print $txt;
}
?>
```

მაშასე თქვენ უკვე გაეცანით რამოდენიმე საინტერესო ფუნქციას და მათი გამოყენების წესებს.

27. ფაილებთან სამუშაო ფუნქციები.

განვიხილოთ რამოდენიმე საინტერესო ფუნქცია, რომლებიც გამოგვადგება ფაილებთან მუშაობისას. კერძოდ, ამ ფუნქციების საშუალებით შესაძლებელია ფაილების წაშლა, შექმნა, ფაილებში ინფორმაციის ჩაწერა და სხვა.

count ("faili.php") – ფაილის გამოთვლა, გზა ფაილამდე მითითებულია ბრჭყალებში.
basename ("faili.php") – გზა მითითებულ ფაილამდე.
copy ("faili.php") – ფაილის კოპირება, ფაილის სახელი მითითებულია ბრჭყალებში.
stat ("faili.php") – ინფორმაცია ფაილის შესახებ.
unlink ("faili.php" , პარამეტრი) – ფაილის წაშლა, ამოშლა ფოლდერიდან.
fgets ("faili.php" , 5) – ფაილიდან სტრიქონების ამოკითხვა, ჩვენს შემთხვევაში პროგრამა ამოიკითხავს 5 სტრიქონს.
fgetss ("faili.php" , 5) – ფაილიდან სტრიქონების ამოკითხვა, html ტეგების გარეშე და php კოდის გარეშე. ჩვენს შემთხვევაში პროგრამა ამოიკითხავს 5 სტრიქონს.
fgetc ("faili.php") – სიმბოლოთა დათვლა ფაილში არსებულ ტექსტში.
file ("file.php") – ფაილის გამოცხადება, ფაილის შინაარსი აღიქმება მასივად.
fileatime ("faili.php") – ფაილზე ბოლო მიმართვის დრო.

file_exists ("faili.php") – ფაილის არსებობის შემოწმება.
filectime ("faili.php") – ფაილის ბოლო რედაქტირების დრო.
filesize ("faili.php") – ფაილის ზომა.
filetype ("faili.php") – ფაილის ტიპი, გაფართოება.
fpassthru ("faili.php") – ფაილის შინაარსის ეკრანზე გამოტანა.
fputs ("faili.php" , 4 , 2) – ფაილში მონაცემების ჩაწერა, ჩვენს შემთხვევაში ჩაწერისას სტრიქონების რაოდენობა უდრის 4 -ს ზომა კი 2 -ს.
fread ("faili.php" , 5) – ფაილიდან ინფორმაციის ამოკითხვა, ჩვენს შემთხვევაში ამოსაკითხი სიმბოლოთა რაოდენობა უდრის 5 -ს.
fseek ("faili.php" , "პარამეტრი") – ძიება ფაილში.
fwite ("faili.php" , 5) – ფაილში ინფორმაციის ჩაწერა, ჩვენს შემთხვევაში ჩასაწერი სიმბოლოების რაოდენობა უდრის 5 -ს.
rename ("old.php" , "new.php") – ფაილის სახელის გადარქმევა. ჩვენს შემთხვევაში old.php არის ძველი სახელი, ხოლო new.php სახელის გადარქმევისას მინიჭებული ახალი სახელი.
tempname ("folderi" , "პარამეტრი") – ფაილისთვის უნიკალური სახელის მინიჭება.
fclose ("faili.php") – ფაილის დახურვა.
fopen ("faili.php") – ფაილის გახსნა, რომელსაც გააჩნია შემდეგი პარამეტრები:

* r ფაილი იხსნება წაკითხვისთვის
* r+ ფაილი იხსნება წაკითხვა- ჩაწერისთვის
* w ფაილი იხსნება ჩაწერისთვის
* w+ ფაილი იხსნება ჩაწერა-წაკითხვისთვის
* a ფაილი იხსნება ჩაწერისათვის
* a+ ფაილი იხსნება ჩაწერა-წაკითხვისთვის

ეხლა განვიხილოთ ფუნქციები, რომელთა საშუალებით შესაძლებელია დირექტორიებთან, და როგორც მას ხშირად ეძახიან, პაპკებთან მუშაობა.
mkdir ("folderi" , 0700) – ახალი დირექტორიის შექმნა.
ჩვენს შემთხვევაში შექმნილი დირექტორიის სახელი იქნება folder
ჩანაწერი 0700 მიუთითებს დირექტორიაში შეღწევის უფლებაზე.
rmdir ("folderi") – დირექტორიის ამოშლა, დირექტორიის სახელი მითითებულია ბრჭყალებში.
opendir ("folderi") – დირექტორიის გახსნა.
closedir ("folderi") – დირექტორიის დახურვა.
chdir ("folderi") – ცვლის მიმდინარე დირექტორიას მითითებულით.
getcwd ("folderi") – სრული გზა დირექტორიამდე.
direname ("folderi") – დირექტორიის სახელი.
თუკი opendir () ფუნქციის საშუალებით ვერ მოხერხდა მითითებული დირექტორიის გახსნა, მაშინ გამოდის შეცდომის აღმნიშვნელი ტექსტი, თუკი გსურთ, რომ ეს ტექსტი არ გამოვიდეს ფუნქციას წინ უდა დაუმატოთ ნიშანი @

28. შევქმნათ ფაილის სერვერზე ატვირთვის პროგრამა.

მაშასე მოდით შევქმნათ ფაილის სერვერზე გადაცემის პროგრამა, კერძოდ სურათის სერვერზე გადაგზავნის პროგრამა.

პროგრამის შესაქმნელად დაგვჭირდება სამი ფაილი და ერთი ფოლდერი. მაშასე, ერთ საქაღალდეში ვქმნით ახალი ფოლდერს ერთ html და ერთ php ფაილს. ფოლდერი სახელი გახლავთ: upload ხოლო ფაილების index.html და program.php

index.html ფაილი შეიცავს შემდეგი ტიპის მარტივ ფორმას:

```
<form action="program.php" method="post" enctype="multipart/form-data">
<input type="file" name="uploadfile">
<input type="submit" value="ატვირთვა">
</form>
```

იმისათვის, რომ ფაილი გადაეცეს სერვერს, საჭიროა ფორმა შეიცავდეს ჩანაწერს: enctype="multipart/form-data"

program.php ფაილში კი ვწერთ შემდეგ პროგრამას:

```
<?php
$uploaddir = './upload/';
if ($uploadfile = $uploaddir.basename($_FILES['uploadfile']['name']))
{
$_FILES['uploadfile']['name'] = (date("h_i_s_d_m_y")).".jpg";
}
$uploadfile = $uploaddir.basename($_FILES['uploadfile']['name']);
if (copy($_FILES['uploadfile']['tmp_name'], $uploadfile))
{
echo "ფაილი წარმატებით აიტვირთა სერვერზე!";
}
else
{
echo "ფაილი ვერ აიტვირთა სერვერზე!";
exit;
}
echo "<br><h2>";
echo "ფაილის სახელი - ".$_FILES['uploadfile']['name']."<br>";
echo "ფაილის ტიპი - ".$_FILES['uploadfile']['type']."<br>";
echo "ფაილის ზომა - ".$_FILES['uploadfile']['size']."<br>";
echo "დროებითი გზა - ".$_FILES['uploadfile']['tmp_name']."<br>";
$url_file = $_FILES['uploadfile']['name'];
$full_url = "http://localhost/www/aka/";
```

```
print "<a href='$full_url$url_file'>$full_url$url_file</a>";
echo "</h2>";
?>
```

იმისათვის, რომ ერთნაირი სახელების მქონე ფაილები არ დაემთხვეს ერთმანეთს და არ მოხდეს მათი ერთმანეთზე გადაწერა ჩვენ გამოვიყენეთ ჩანაწერი:
$_FILES['uploadfile']['name'] = (date("h_i_s_d_m_y")) რომლის თანახმადაც ყველა ახალ ატვირთულ ფაილს მიენიჭება უნიკალური სახელი, ანუ
ფუნქცია (date("h_i_s_d_m_y")) რომელიც დროს გამომთვლელია, მიანიჭებს ფაილებს იმ სახელს რა დროსაც ისინი აიტვირთნენ სერვერზე. გამომდინარე იქედან, რომ ფაილების ატვირთვის დრო ერთმანეთ არ დაემთხვევა, შესაბამისად გამოირიცხება ერთნაირი სახელების მქონე ფაილების თანხვედრა.
ასევე ყურადღება გაამახვილეთ იმ ფაქტზე, რომ ეს პროგრამა განკუთვნილია სურათების ასატვირთათ, კერძოდ jpg ფორმატის სურათებისათვის, ხოლო თუკი გვუსრს სხვა ტიპის ფაილების ატვირთვა, მაშინ ჩანაწერი:
$_FILES['uploadfile']['name'] = (date("h_i_s_d_m_y")).".jpg";
უნდა შეიცვალოს სხვა ჩანაწერით, მაგალითად pdf ფაილების ასატვირთად ჩანაწერი მიიღებს შემდეგ სახეს:
$_FILES['uploadfile']['name'] = (date("h_i_s_d_m_y")).".pdf";

29. კომენტარების დამატება ვებგვერდზე.

ამ პროგრამის დასაწერად დაგვჭირდება ერთ საქარალდეში შევქმნათ სამი ფაილი სახელწოდებებით: index.html | add.php | database.txt
database.txt -ში შეინახება დამატებული კომენტარები, ასევთქვათ ეს ფაილი წარმოადგენს მინი მონაცემთა ბაზას.

index.html ფაილში ვწერთ შემდეგ ფორმას:

```
<form name="form1" method="post" action="add.php">
<input type="text" name="radiobutton" />
<input type="submit" value="submit" />
</form>
```

add.php ფაილში კი ვწერთ შემდეგ პროგრამას:

```
<?php
$fp=@fopen("database.txt","a");
fputs($fp,"$radiobutton \r\n\n");
@fclose($fp);
$file=@file("database.txt");
```

```
$q=count($file);
foreach ($file as $key => $val)
{
print $key=$val."<br>";
}
?>
```

30. შევქმნათ საიტის მთვლელი.

ამ პროგრამის დასაწერად დაგვჭირდება ერთ საქარალდეში შევქმნათ ორი ფაილები სახელწოდებებით: count.php | database.txt
database.txt -ში შეინახება მონაცემები, ასევთქვათ ეს ფაილი წარმოადგენს მინი მონაცემთა ბაზას.

count.php ფაილი შეიცავს შემდეგ პროგრამულ კოდს:

```
<?php
$file = fopen("counter.txt", "r");
$c = fgets ($file, 150);
fclose($file);
$c++;
$file = fopen("counter.txt", "w");
fputs ($file, $c);
fclose($file);
echo "<p align='center'>ეს გვერდი ნანახია $c -ჯერ</p>";
?>
```

31. შევქმნათ სტუმართა წიგნაკი ვებგვერდისათვის.

მოცემულ გაკვეთილში ჩვენ განვიხილავთ პროგრამას, რომლის საშუალებითაც შესაძლებელია ვებგვერდზე სტუმართა წიგნის დამატება, სადაც სტუმრებს შეუძლიათ დატოვონ თავიანთი კომენტარები და შთაბეჭდილებები თუნდაც ვებგვერდის შესახებ.
დაგვჭირდება ერთ საქაღალდეში შევქმნათ ოთხი ფაილი შემდეგი სახელებით:
add.php | database.txt | form.php | guest.php
database.txt ფაილში შეინახება ინფორმაცია, ანუ ეს ფაილი წარმოადგენს მინი მონაცემთა ბაზას.

add.php ფაილში იწერება შემდეგი სახის პროგრამა:

```
<?php
if ($name!=="" && $mail!=="" && $mess!=="")
{
$z=1;
}
else
{
$z=0;
};
if ($z==1)
{
$time=date("d.m.Y წელი, H:i წუთი");
$ob_stat="$name#$mail#$mess#$time";
$ob_stat=strip_tags($ob_stat);
$file=file("database.txt");
$counter=count($file);
$fp=@fopen("database.txt","a"); fputs($fp,"$ob_stat \r\n"); @fclose($fp);
header("Location:guest.php?str");
}
else
{
header("Location:form.php?error=1");
};
?>
```

form.php ფაილში ვწერთ შემდეგი სახის ფორმას:

```
<?php
if (!empty($error))
{
echo "<font color=\"red\"><strong> გთხოვთ შეავსოთ ყველა ტექსტური
ველი! </strong></font><br><br>";
};
echo "<form name=\"form1\" method=\"post\" action=\"add.php\">
<table width=\"600\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\">
<tr>
<td width=\"25%\" valign=\"top\" scope=\"col\">
<p><strong> თქვენი სახელი: </strong></p>
<p><strong> თქვენი გვარი: </strong></p><br>
<p><strong> კომენტარი: </strong></p></td>
<td width=\"89%\" scope=\"col\"><p>
<input name=\"name\" type=\"text\" size=\"35\" maxlength=\"25\">
```

```
</p>
<p>
<input name=\"mail\" type=\"text\" size=\"35\" maxlength=\"30\">
</p>
<p>
<textarea name=\"mess\" cols=\"40\" rows=\"5\"></textarea>
</p>
<p>
<input type=\"reset\" name=\"Submit\" value=\" ველების გასუფთავება! \">
<input type=\"submit\" name=\"Submit\" value=\" კომენტარის დამატება \">
</p></td>
</tr>
</table>
</form>
";
?>
```

guest.php ფაილი კი შემდეგი შინაარსის გახლავთ:

```
<?php
echo "<a href='form.php'> <strong>
<< ახალი კომენტარის დამატება!</strong></a><br><br>";
$xfile1 = @file("database.txt");
$xfile=array_reverse($xfile1);
$a=count($xfile);
$p=10;
$kol_st=ceil($a/$p);
$k=0+($p*$str);
$j=$p+($p*$str);
if($j>$a)
{
$j=$a;
};
for ($i=$k;$i<$j;$i++)
{
$udata = explode("#",$xfile[$i]);
echo "
<p> <b>სახელი:</b> $udata[0] <br>
<b>გვარი:</b> $udata[1] </p>
<p> <b>კომენტარი:</b> $udata[2] </p>
<p> <small><b>დამატებულია:</b> $udata[3] </small></p>
<br>";
```

```
};
echo "<p align=center><br>";
if ($str>0)
{
$id1=($str-1);print"<a href=guest.php?str=$id1><strong> << წინა გვერდი </strong></a>";
};
print" | ";
if($str<$kol_st-1)
{
$str=($str+1);print"<a href=guest.php?str=$str><strong> მომდევნო გვერდი >>
</strong></a>";
};
print"<br><br>გვერდების რაოდენობა: <strong>$kol_st</strong>";
print"<br>კომენტარების რაოდენობა: <strong>$a</strong>";
echo"</p>";
?>
```

ეხლა მივმართოთ ჩვენს form.php ფაილს და დავტესტოთ ჩვენი პროგრამა.

პროგრამა თქვენს მიერ შევსებული ფორმის ელემენტთა მნიშვნელობებს შეინახავს database.txt ფაილში და ყოველი ახალი ჩანაწერი მიებმება ძველ ჩანაწერს.

database.txt ფაილის ნაცვლად შეგვიძლია გამოვიყენოთ სხვა გაფართოების მქონე ფაილიც, მაგალითად database.html ან database.php ამას არავითარი მნიშვნელობა არ აქვს. თავისთავად სხვა გაფართოების ფაილის გამოყენებისას ზემოთ მოყვანილ პროგრამებშიც უნდა მოხდეს ცვლილების დაფიქსირება და იმ ფაილის სახელის ზუსტად მითითება, რომელსაც ჩვენ ვიყენებთ როგორც მონაცემთა ბაზას.

პროგრამაში ასევე გამოყენებულია ახალ გვერდზე ავტომატურად გადასვლის პრინციპი, კერძოდ, როდესაც კომენტარების რაოდენობა გადააჭარბებს ათს პროგრამა ავტომატურად გამოიტანს მომდევნო გვერდს, მასასადამე, ერთ გვერდზე მხოლოდ ათი კომენტარის დატოვების საშუალება გვეძლევა.

თავისთავად უფრო მოსახერხებელია თუ მსგავსი ტიპის პროგრამები მიებმება მონაცემთა ბაზას.

ჩვენს შემთხვევაში ტექსტური ფაილი წარმოადგენს მონაცემთა ბაზას, ანუ საჭიროა ამ ფაილის ჯერ გახსნა, მერე მასში ინფორმაციის შეტანა და ბოლოს მისი დახურვა, ზოგიერთ შემთხვევაში კი ფაილის პირდაპირ გახსნა ვერ ხორციელდება სხვადასხვა შეზღუდვების გამო, აქედან გამომდინარე უფრო ხელსაყრელია გამოვიყენოთ MySQL მონაცემთა ბაზა, სადაც შევინახავთ საჭირო ინფორმაციას.

ამით ჩვენი გაკვეთილები დასრულებულია, რაც შეეხება სესიებს, ავტორიზაციის სკრიპტებს და სხვა, ჩვენ ამ ყოველივეს განვიხილავთ MySQL მონაცემთა ბაზების შესწავლის შემდგომ.
მაშასე თქვენ უკვე დაეუფლეთ PHP პროგრამირების ენას, მომავალში კი გისურვებთ უამრავ წარმატებებს, მე კი აქ გემშვიდობებით!