ლექცია № 9. ციკლი. **while ; do** …  **while**;

ოპერატორი **while** (მანამ) ახდენს ციკლის ორგანიზებას და მას შემდეგი სტრუქტურა (სინტაქსისი და სემანტიკა) გააჩნია:

**while (პირობა)**

**{**

**კოდი**

**}**

ციკლის ეს ოპერატორი მუშაობს შემდეგნაირად:

1. იწყება პირობის შემოწმება და თუ კი პირობა ჭეშმარიტია, მაშინ გადასვლა ხდება მეორე პუნქტზე, წინააღმდეგ შემთხვევაში პუნქტ 3-ზე

2. სრულდება ციკლის ტანი, ანუ ის კოდი, რომელიც მოთავსებულია ფიგურულ ფრჩხილებში და შემდეგ ხდება გადასვლა პირველ პუნქტზე.

3. ციკლის მუშაობის დასრულება.

ამდენად მისაქცევია ყურადღება იმ გარემოებაზე, რომ თუ კი თავიდანვე არ სრულდება პირობა - ანუ ბულეანის ტიპის ცვლადი - გამოსახულება მცდარია - false, ციკლის ტანი არცერთხელ არ შესრულდება.

არსებობს მეორე უკიდურესობაც, როცა პირობა მუდამ სრულდება და მაშინ ხდება ჩაციკვლა და ეს რომ არ მოხდეს ისევე როგორც ციკლის **for** ოპერატორისთვი, გამოიყენება ოპერატორები **break** და **continue**; განვიხილოთ ისინი მოგვიანებით. ეხლა კი ავღნიშნოთ თუ რით განსხვავდება ოპერატორი **while** ციკლის სხვა ოპერატორისგან **for** .

* **while** ციკლის ტანის შესრულება შეიძლება დავუკავშიროთ ნებისმიერ პირობას, მაშინ როცა ციკლ **for** -ში მონიტორინგს ექვემდებარება მთვლელის მთელი მნიშვნელობა. ამ შემთხვევაში კი პირობა შეიძლება იყოს სხვადასხვა - ვთაქვათ დროსთან ტოლობა ან ვებ-გვერდის ფერის მნიშვნელობასთან თანხვედრა.

ლისტინგი 1 **while** ციკლის გამოყენება

<html>

<head>

<title> ციკლი while</title>

<script language="javascript">

<!-- ვინიღბებით!

mass=new Array(10)

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

mass[i]=i+1; document.write ("გამოგვყავს მასივის ელემენტები mass[",i,"]=",mass[i],"<BR>");

}

total=0; n=0;

while (total<10)

{

total+=mass[n];

n++;

}

document.write ('მასივის პირველი ',n--,' ელემენტის ჯამი ნაკლებია ან ტოლია 10-ზე

total=',total)

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

ლისტინგი 2 ახარისხება

<html>

<head>

<title> ციკლი while ახარისხება</title>

<script language="javascript">

<!-- ვინიღბებით!

var z,х,y;

x=parseInt(prompt("შეიყვანე ფუძე","x"));

y=parseInt(prompt("შეიყვანე ხარისხის მაჩვენებელი","მთელი რიცხვი y"));

i=1; z=1;

while(i <=y){

z =z\*x;

i++

}

document.write(x," აყვანილი ",y," ხარისხად ტოლია ",z,"-ის");

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

ლისტინგი 3 ფაქტორიალი

<html>

<head>

<title> ციკლი while ფაქტორიალი</title>

<script language="javascript">

<!-- ვინიღბებით!

var n,х;

n=parseInt(prompt("შეიყვანე ნატურალური რიცხვი","ნატურალური რიცხვი n"));

z=1;

if(n>1){

i=2;

while(i <=n){

z =z\*i;

i++;

}

}

document.write(n," -ის ფაქტორიალი n!=",z);

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

მეთოდი **Math.random()** იძლევა შემთხვევითი რიცხვების გენერირების საშუალებას 0-დან ერთამდე, თანაც 0 შეიძლება მოგვცეს ამ მეთოდმა, მაგრამ 1 -არა. ამ მეთოდით მიღებულ შემთხვევით რიცხვებს ფსვდოშემთხვევითი რიცხვები ეწოდება. რამდენად ახლოა ფსვდოშემთხვევითი რიცხვები ნამდვილად შემთხვევით რიცხვებთან? ამ კითხვაზე პასუხის გასაცემად მიმართავენ ალბათობისა და მათემატიკური სტატისტიკის თეორიას, რომლის თანახმად თუ კი ავიღებთ უსასრულო რაოდენობას შემთვევითი რიცხვებისა, მაშინ მისი საშუალო მნიშვნელობა უნდა უტოლდებოდეს 0,5 -ს. ასეთ შემთხვევაში შემთხვევითი რიცხვების გენერატორი სანდოა. შევაფასოთ **Javascript** -ის **Math.random()** მეთოდის სანდოობა და გამოვთვალოთ 10 000 შემთხვევითი რიცხვი, ვიპივოთ მათი საშუალო მნიშვნელობა სამი ათნიშნა რიცხვის სიზუსტით

ლისტინგი 4 ფსევდოშემთხვევითი რიცხვების გენერატორის სანდოობის შემოწმება

<HTML>

<HEAD>

<TITLE>შემთხვევითი რიცხვების გენერატორი</title>

<STYLE>

H2, P {font-family: LitNusx}

</style>

</head>

<BODY>

<H2> generatoris Semowmeba </h2>

<P> ramdenad SemTxveviTia SemTxveviTi ricxvebis generatoriT

miRebuli ricxvebi? gamovTvaloT 100 000 aseTi ricxvis saSualo

mniSvneloba.

<HR>

<SCRIPT LANGUAGE="JavaScript">

total=0; i = 0;

while(i < 100000)

{

num=Math.random();

total += num;

i++;

}

average=total/100000;

average=Math.round(average\*1000)/1000;

document.write ("<H2> SemTxveviTi ricxvebis saSualo ariTmetikuli tolia : " +average + "</h2>");

</script>

</body>

</html>

**ოპერატორი** **do …. while**

ოპერატორ **do …. while** –ს (გააკეთე .. მანამ) აქვს შემდეგი კონსტრუქცია

**do {**

**კოდი**

**}**

**while(პირობა)**

ციკლის ეს ოპერატორი მოქმედებს შემდეგნაირად:

1. შესრულდება ფიგურულ ფჩხილებში მოთავსებული კოდი და გადადის მეორე პუნქტზე.

2.მოწმდება პირობა, თუ ის ჭეშმარიტია გადადი პუნქტ 1-ზე, თუ მცდარი - პუნქტ 3-ზე.

3. ციკლის დასრულება.

მოვიყვანოთ ადრე განხილული მაგალითები while ციკლის ოპერატორის გამოყენებით, do…while ციკლის ოპერატორის საშუალებით;

ლისტინგი 5\_1 **do …. while** ციკლის გამოყენება

<html>

<head>

<title> ციკლი do ... while</title>

<script language="javascript">

<!-- ვინიღბებით!

mass=new Array(10)

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

mass[i]=i+1; document.write ("გამოგვყავს მასივის ელემენტები mass[",i,"]=",mass[i],"<BR>");

}

total=0; n=0;

do

{

total+=mass[n];

n++;

}

while (total<10)

document.write ('მასივის პირველი ',n--,' ელემენტის ჯამი ნაკლებია ან ტოლია 10-ზე

total=',total)

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

აღსანიშნავია, რომ **do …. while** ციკლის ოპერატორისთვის, პირველ ჯერზე აუცილებლად მოხდება ციკლის ტანის კოდის (ფიგურულ ფრჩხილებში ჩაწერილი კოდის) შესრულება, და მხოლოდ ამის შემდეგ მოხდება პირობის შემოწმება. ანუ ციკლი ერთხელ მაინც იმუშავებს.

ლისტინდი 6\_2 ახარისხება

<html>

<head>

<title> ციკლი do ... while ახარისხება</title>

<script language="javascript">

<!-- ვინიღბებით!

var z,х,y;

x=parseInt(prompt("შეიყვანე ფუძე","x"));

y=parseInt(prompt("შეიყვანე ხარისხის მაჩვენებელი","მთელი რიცხვი y"));

i=1; z=1;

do

{

z =z\*x;

i++

}

while(i <=y)

document.write(x," აყვანილი ",y," ხარისხად ტოლია ",z,"-ის");

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

ლისტინგი 7\_3 ფაქტორიალი

<html>

<head>

<title> ციკლი do ...while ფაქტორიალი</title>

<script language="javascript">

<!-- ვინიღბებით!

var n,х;

n=parseInt(prompt("შეიყვანე დადებითი მთელი რიცხვი","მთელი რიცხვი n"));

z=1;

if(n>1){

i=2;

do

{

z =z\*i;

i++;

}

while(i <=n)

}

document.write(n," -ის ფაქტორიალი n!=",z);

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

ოპერატორი **break** -ის გამოყენება, როცა საჭიროა ავიცილოთ თავიდან ჩაციკვლა და გამოვიდეთ ციკლიდან. ქვემოთ მოყვანილ მაგალითში მოიცემა ორი შემთხვევითი რიცხვი 0-დან 20-მდე და შეყვანის ფანჯარაში უნდა ჩაიწეროს ამ ორი მთელი რიცხვის ნამრავლის მნიშვნელობა. თუ მომხმარებლის მიერ შეყვანილი ნამრავლი სწორია, მაშინ ხდება ციკლიდან გამოსვლა, წინააღმდეგ შემთხვევაში ციკლი გრძელდება და შედეგად გვაქვს უსასრულო ციკლი.

ლისტინგი 8 ციკლის შეწყვეტა ოპერატორი **break** -ით

<html>

<head>

<title> ციკლი while და break</title>

<script language="javascript">

<!-- ვინიღბებით!

var c=1;

while (c){

num1=Math.round(Math.random()\*19+1);

num2=Math.round(Math.random()\*19+1);

result=prompt(("შეიყვანეთ ნამრავლი"+num1+" X "+num2)," ");

if(result&&parseInt(result)){

if(result==num1\*num2)break

}

}

document.write ('ნამრავლი ',result)

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

</body>

</html>

ოპერატორი **continue** -ს გამოყენება, როცა საჭიროა ავიცილოთ თავიდან ციკვლში არასასურველი ქმედება, და გავდივართ ციკლის ბოლოს. ეს არ ნიშნავს, რომ უსათუოდ დავტოვეთ ციკლი, ანუ გამოვედით იქიდან. მხოლოდ ციკლის ერთი იტერაცია დასრულდება, ციკლი კი გაგრძელდება შემდეგი იტერაციიდან.

ლისტინგი 9 continue

<html>

<head>

<title> ციკლი while და continue</title>

<script language="javascript">

<!-- ვინიღბებით!

function culc(){

var start=Math.round(document.forms[0].elements[0].value);

var end=Math.round(document.forms[0].elements[1].value);

if(start&&end&&(start<end)){

output.innerHTML=" ";

for(n=start;n<=end;n++){

if(n==0) continue

sVal=1/n+";<br>";

output.innerHTML+=sVal;

}

}

}

// ვიხსნით ნიღაბს. -->

</script>

</head>

<body>

<form>

რიგის პირველი წევრი:<input type='text' VALUE='-5'><br>

რიგის ბოლო წევრი:<input type='text' VALUE='5'><br>

<input type='button' VALUE='გამოვიყვანოთ შებრუნებულ რიცხვთა რიგი'

ONCLICK='culc()'><br>

</form>

<p name='output' id='output'></p>

</body>

</html>

ორგანზომილებიანი მასივები

ლისტინგი 10 მასივების გადამრავლება

<HTML>

<HEAD>

<TITLE> მატრიცების გადამრავლება </title>

</head>

<BODY>

<H2> მატრიცების გამრავლება </h2>

<P> შეგვყავს A მატრიცა </p>

<SCRIPT LANGUAGE="JavaScript">

<!-- ვინიღბებით

var a=new Array();

for(i=0;i<=1;i++){

a[i]=new Array();

for(j=0;j<=2;j++){

a[i][j]=prompt("შეიყვანეთ A მასივის a(i,j)ელემენტი");

document.write( "a(" + i+ "," +j+ ")="+a[i][j]+"; ");

}

document.write("<br>");

};

document.write("<P> შეგვყავს"+" B"+" მატრიცა </p>");

var b=new Array();

for(i=0;i<=2;i++){

b[i]=new Array();

for(j=0;j<=3;j++){

b[i][j]=prompt("შეიყვანეთ B მასივის b(i,j)ელემენტი");

document.write( "b(" + i+ "," +j+ ")="+b[i][j]+"; ");

}

document.write("<br>");

};

document.write("<P>"+"ვითვლით A და B მასივების ნამრავლს C მასივში"+ "</p>");

var c=new Array();

for(i=0;i<=1;i++)

{

c[i]=new Array();

for(j=0;j<=3;j++){

s=0;

for (k=0;k<=2;k++)

s=s+a[i][k]\*b[k][j];

c[i][j]=s;

document.write( "c(" + i+ "," +j+ ")="+c[i][j]+"; ");

}

document.write("<br>");

};

function table (a) { //Вывод матрицы в таблицу HTML

var rows=a.length, cols;

document.writeln ('<table cellpadding="4" cellspacing="0" border="0" ');

for (i=0; i<rows; i++) {

document.writeln ('<tr>');

cols=a[i].length;

for (j=0; j<cols; j++) {

document.writeln ('<td>'+a[i][j]+'</td>');

}

document.writeln ('</tr>');

}

document.writeln ('</table>');

}

document.write("<P>"+" A მატრიცა"+ "</p>");

table (a); //Построение таблицы

document.write("<P>"+" B მატრიცა"+ "</p>");

table (b);

document.write("<P>"+" C მატრიცა"+ "</p>");

table (c);

//ვიხსნით ნიღაბს-->

</script>

<br>

<P> მატრიცების გადამრავლება დასრულებულია</p>

</body>

</html>