13 ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС. ЦИКЛ КОМАНДАЛАРЫ

Белгілі бір әрекеттердің қайталанып орындалуын цикл операторларының көмегімен ұйымдастыруға болатындығы бізге белгілі. JavaScript программалау тілінде циклді ұйымдастырудың өзіндік ерекшеліктері бар.

1. While циклі және оны пайдалану

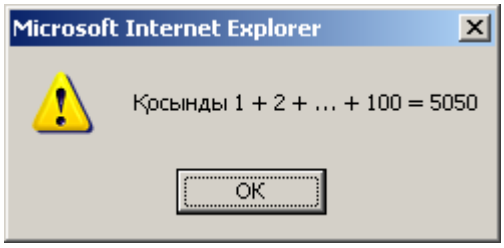
Циклдің орындалуы: алдымен шарт тексеріледі. Егер ол ақиқат болса, командалар (цикл тұлғасы) орындалады. Келесі жолы да осы әрекеттер қайталанады, яғни шарт тексеріледі, егер ол ақиқат болса, цикл орындалады, т.с.с. Кезекті тексеру кезінде шарт жалған болған кезде, цикл жұмысы аяқталады. Циклде шарт алдын ала тексерілетін болғандықтан, ол бір де бір рет орындалмауы да мүмкін.

Жалпы жазылуы	Мысалы

`while(шарт)`
`  команда;`
	`var i = 1;`
	`var sum = 0;`
	`while(i <= 100)`
	`{`
	`  sum += i;`
	`  i ++;`
	`}`
	`alert("Қосынды 1 + 2 + ... + 100=" + sum);`

Программаның толық мәтіні:

```
<HTML>
 <HEAD>
  <TITLE> while командасы </TITLE>
 </HEAD>
 <BODY bgcolor = yellow text = red>
 <H2>While командасын пайдалану </H2>
    <HR>
    <SCRIPT language=JavaScript>
    <!--
       var i = 1;
        var sum = 0;
        while(i <= 100)
         { sum += i; i ++;}
        alert("Қосынды 1 + 2 +...+ 100 = " + sum);
    //-->
    </SCRIPT>
   </BODY>
</HTML>
```



13.1-сурет. Мысалды орындау нәтижесі

Тапсырмалар

1. Жоғарыдағы мысалға қарай отырып, келесі мысалдарды орындау кезінде **alert** терезесіне қандай мәлімет шығатынын ауызша айтуға (кейіннен оны тексеру қажет) тырысыңыздар. Егер қателер болса, оның қалай туындағанын түсіндіріңіздер.

1-мысал	2-мысал

`var x = 5;`
`var s=0;`
`while(x) {s += x; x--;}`
`alert(s);`
	`var x = 5;`
	`var s = 0;`
	`while(--x) s += x;`
	`alert(s);`

3-мысал	4-мысал

`var x = 5;`
`var s = 0;`
`while(x--) s += x;`
`alert(s);`
	`var x = 5;`
	`var s = 0;`
	`while(s) s += x;`
	`alert(s);`

5-мысал	6-мысал

`var x = 5;`
`var s = 0;`
`while(!x) s += x;`
`alert(s);`
	`var x = 5;`
	`var s = 0;`
	`while (--x) s += x; s ++;`
	`alert(s);`

7-мысал	8-мысал

`var x = 5;`
`var s = 0;`
`while(--x && s < 10) s+= x;`
`alert(s);`
	`var x = 5;`
	`var s = 0;`
	`while(--x || s < 10) s += x;`
	`alert(s);`

9-мысал	10-мысал

`var x = 5;`
`var s = 0;`
`while(--x || !s) s += x;`
`alert(s);`
	`var x = 5;`
	`var s = 0;`
	`while(--x && s) s += x;`
	`alert(s);`

2. Төмендегі тізбектің бесінші мүшесін тауып, оны `alert` терезесіне шығарыңыздар:
$a_1 = 2$; $a_n = a^2_{n-1} + 1$.

3. Төмендегі тізбектің қосындысын тауып, оны `alert` терезесіне шығарыңыздар:
summa = 1+1/2+1/3+. . . + 1/10

4. Төмендегі тізбектің қосындысын 0,0001 дәлдігімен тауып, оны `alert` терезесіне шығарыңыздар:
summa = $1+1/2+1/4+. . . + 1/2^n + \ldots$

5. 1-ден 200-ге дейінгі тақ және жұп сандардың қосындысын `alert` терезесіне шығарыңыздар:
summa1 = 1+3+5+. . . + (2n-1) + . . . + 199; summa2 = 2+4+6+. . . + 2n + . . . + 200.

2. For циклі

Төменде for циклының жалпы жазылу түрі мен алдыңғы мысалдың осы команда арқылы орындалуы көрсетілген.

Жалпы жазылу түрі:

for (цикл басы; шарты; қадамы)
команда;

Мысалдар

```
var i;
var sum = 0;
for(i=1; i<=100; i ++)
  sum += i;
alert ("Қосынды 1 + 2 +  ... + 100 = " + sum);
```

Бұл мысалдың да нәтижесі алдыңғы 13.1-суреттегі мысалдағыдай болады. Цикл жұмысы келесі түрде атқарылады: циклдегі алғашқы теңдік жүзеге асырылып (мысалдағы **`i=1;`** командасы), содан соң келесі әрекеттер орындалады:

- шартты тексеру (мысалдағы **`i<=100`**);
- цикл тұлғасын орындау (мысалдағы **`sum+=i`**);
- қадамды көрсету командасын орындау, (мысалдағы **`i++`**).

Егер шарт бірден жалған болса, **`while`** командасындағы сияқты цикл тұлғасы бір де бір рет орындалмауы да мүмкін. Мұндайда қадам беру командасы да орындалмайды. Ал циклдің алғашқы командасы әрқашанда кем дегенде бір рет орындалады.

Цикл басындағы **for** жолының үш командасының – басы, шарты, қадамы кез келгені жазылмай кетуі де мүмкін, бірақ олардың арасында тұратын нүктелі үтір міндетті түрде сақталады. Егер шарт көрсетілмесе, оның мәні ақиқат (true) болып саналады. Мұндайда, цикл шексіз түрге айналып кетеді:

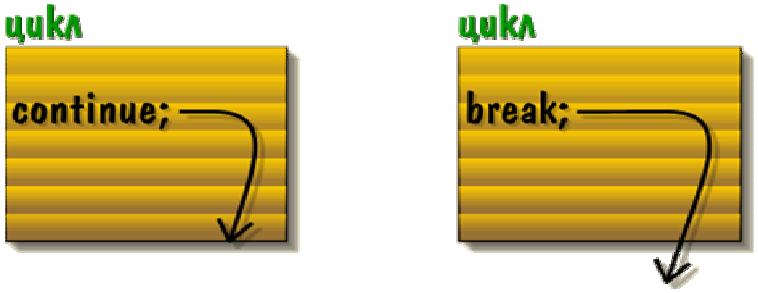
```
for (;;) команда
```

Бұл цикл шексіз орындала береді, яғни аяқталмайды. Оны тек цикл ішіндегі **break** командасы көмегімен ғана аяқтауға болады.

3. Break және continue командалары

Бұл командалар цикл командаларының орындалу ретін өзгерту үшін қолданылады.

Continue командасы циклдің онан кейінгі тұрған барлық командаларын аттап өтіп, цикл параметрінің келесі мәніне көшіреді. Ал **break** командасы жалпы цикл орындалуын аяқтап, одан кейінгі келесі командаларға көшіреді.

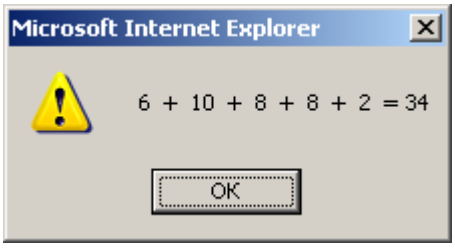


13.2-сурет. **Continue** және **break** командаларының орындалуы

1-мысал (continue). [1, 20] аралығынан кездейсоқ түрде алынған 5 жұп санның қосындысын табу керек.

```
var   len = 5;   // Сандардың қанша екендігі.
var   a = 1;     // Аралықтың сол жақ шекарасы.
var   b = 20;    // Аралықтың оң жақ шекарасы.
var   sum = 0;   // Қосқыш.
var  counter = 0;  // сандардың санауышы.
var  number;       // Кездейсоқ сан.
var  str = " ";    // Экранға шығаруға арналған қатар.
while (counter < len)
{  number = Math.floor(a + (b-a+1)*Math.random());
   if (number % 2) continue;
   sum += number;  str += number;
   if (counter < len-1)  str += " + ";
   else  str += " = ";
   counter++;
}
str += sum;  alert(str);
```

Осы мысалдың орындалу нәтижесінің бірі келесі суреттегідей болады. Программаны әр орындаған кезде 1 мен 20 сандарының арасынан жұп сандар кездейсоқ алынатын болғандықтан, нәтиже әрбір программаны қайталап орындаған кезде өзгеріп отырады.



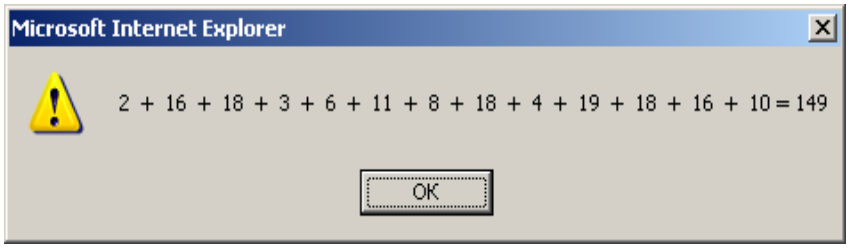
13.3. сурет. 1-мысалдың орындалу нәтижесі

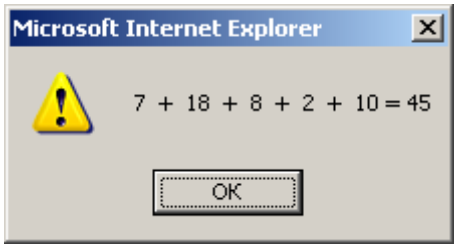
Math.random() стандартты функциясы [0,1] аралығынан кез келген кездейсоқ сан береді. Math.floor(num) стандартты функциясы аргументке тең немесе одан кіші бүтін сан мәнін береді.

2 мысал *(break).* Керекті сандар [1,20] аралығынан кездейсоқ түрде алынып отырады. Осы сандардың кезекті келесісі 10-ға тең болғанша, олардың қосындысын табу керек.

```
var a = 1;          // аралықтың сол жақ шекарасы.
var b = 20;         // аралықтың оң жақ шекарасы.
var special = 10; // кездейсоқ санның өзекті мәні
var sum = 0;        // қосындылауыш.
var number;         // кездейсоқ сан.
var str = "";       // экранға мәні шығарылатын айнымалы.
for ( ; ; )         // шексіз цикл.
{  number = Math.floor(a + (b-a+1)*Math.random( ));
   sum += number;   str += number ;
   if (number == special)  break;
   str  +=  "  +  ";
}
str += " = " + sum;   alert(str);
```

Бұл мысалда да сандар кездейсоқ алынатын болғандықтан нәтижесі әр түрлі бола береді (13.4-сурет).



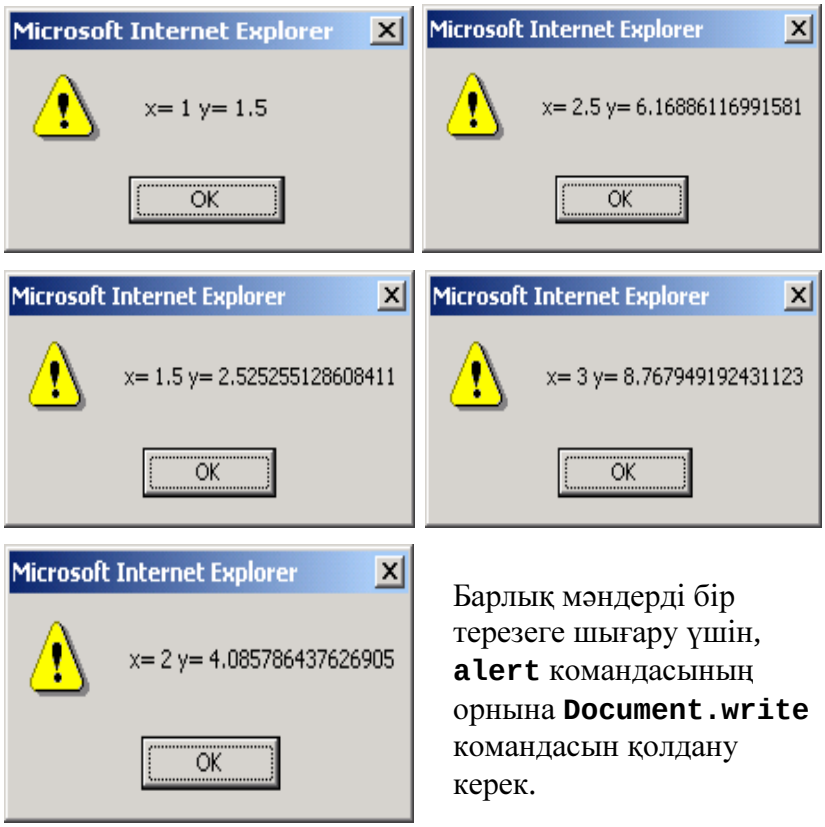


13.4-сурет. 2-мысалдың орындалу нәтижелері

3 мысал. $y = x^2 - \sqrt{x} + 1.5$ функциясы берілген. Мұндағы x аргументі 0-ден 3-ке дейін, қадамы 0,5 болып өзгергенде, y функциясының мәндерін табу керек.

```
var  x0 = 0;      // x-тің бастапқы мәні
var  xk = 3;      // x-тің соңғы мәні
var  dx = 0.5;    // x-тің өзгеру қадамы
var  x=x0;        // x-ке алғашқы мәнді меншіктеу
var y;            // y айнымалысын сипаттау
while (x =< xk)
{  y = x*x - Math.sqrt(x)+1.5;
   alert("x= " + x + " y= " + y);
   x+=dx ;
}
```

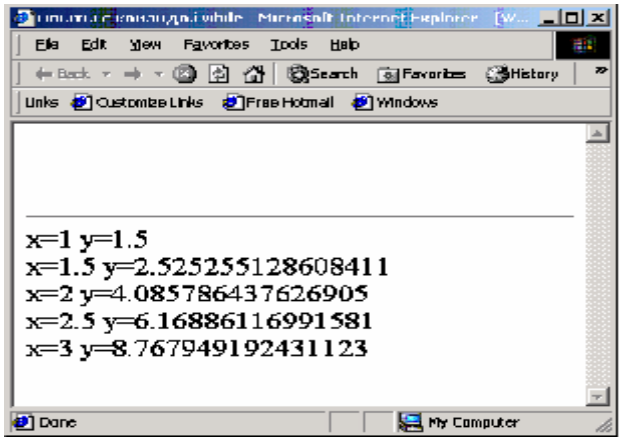
Бұл программа орындалғанда x-пен y-тің әрбір мәндері жеке терезелерге шығады (13.5-сурет).



Барлық мәндерді бір терезеге шығару үшін, **`alert`** командасының орнына **`Document.write`** командасын қолдану керек.

13.5-сурет. 3-мысалдың орындалу нәтижесі

Осы мысалдың **`Document.write`** командасының көмегімен орындалуының нәтижесі келесі суретте көрсетілген:



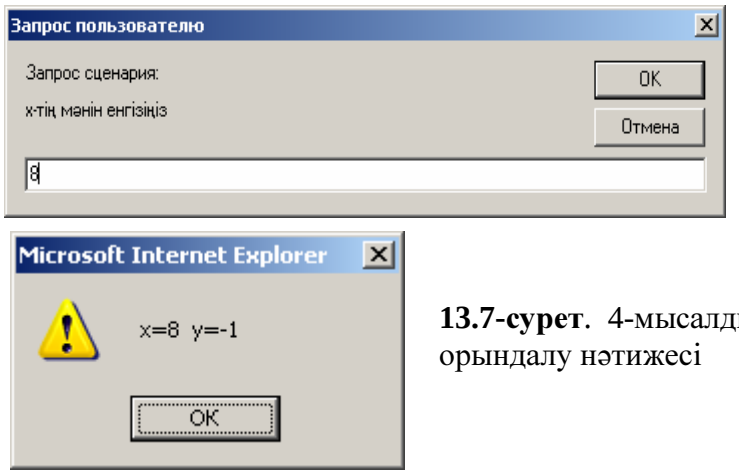
13.6-сурет. 3-мысалдың **`Document.write`** командасының көмегімен орындалу нәтижесі

4-мысал. $y = \begin{cases} x^2 + x + 1, & егер\ x \le 0 \\ x - \sqrt{x+1}, & егер\quad x > 0 \end{cases}$ функциясы берілген.

x-тің мәнін енгізе отырып, *y* функциясының мәндерін анықтау керек.

```
var x = prompt("x-тің мәнін енгізіңіз", "1");
y = (x <= 0) ? x*x+x+1:x-Math.sqrt(x+1);
alert ("x="+x+"  y=" + y);
```

Программаны орындаған кезде, *x*-тің мәнін енгізуге мүмкіндік беретін терезе ашылады (13.7-сурет).



13.7-сурет. 4-мысалдың орындалу нәтижесі

Енді осы функцияның мәнін, x-тің мәні 0-ден 3-ке дейін, 0,5 қадаммен өзгерген кезде анықтау керек болсын.

```
var  x0 = 0;     // x-тің бастапқы мәні
var  xk = 3;     // x-тің соңғы мәні
var  dx = 0.5;   // x-тің өзгеру қадамы
var  x = x0;     // x-ке алғашқы мәнді меншіктеу
var y;           // y-ті сипаттау
while (x =< xk)
 {  y = x*x - Math.sqrt(x) +1.5;
    alert(" x= " + x + " y= " + y);
    x+=dx ;
 }
```

4. Циклдерді кері бағытта программалау

Келесі мысалда set жиымы (массиві) элементтері қосындысын табу жолы көрсетілген (элементтерді нөмірлеу 0-ден басталады; **set.length** – жиым қасиеті, ол оның ұзындығын білдіреді):

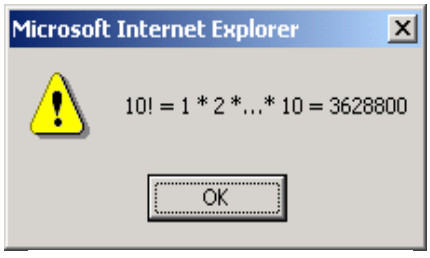
```
var sum = 0;
for(var i=0; i < set.length; i++)
    sum += set[i];

Былай да жазуға болады:
var sum = 0;
for(var i = set.length; --i >= 0; )
    sum += set[i];
```

Екінші тәсіл дұрысырақ, өйткені санды нөлмен салыстыру басқа санмен салыстыруға қарағанда, жылдамырақ орындалады. Басқаша айтсақ, **--i>=0** шарты **i<set.length** шартына қарағанда тиімдірек болады. Жалпы ереже: санауышы бар циклде санауыш мәні мүмкіндігінше нөлге дейін төмендегені дұрыс болып табылады.

Келесі мысалда 10!= 1*2*. . .*10 мәні есептеледі.

```
var proizvedenie = 1;
for(var i=10; i; --i)
    proizvedenie *= i;
alert("10!=1 * 2 *...* 10 =
```



13.8-сурет. 10! есептеу

```
"+proizvedenie);
```

Келесі нұсқа:

```
var sum = 0;
for(var i = set.length; (i--) >= 0; )
   sum += set[i];
```

қате болып табылады, өйткені ондағы **i** мәнін бірге кеміту шартты тексеруден кейін жазылған, ол оның алдында тұруы тиіс.

5. Цикл айнымалысын сипаттау

Цикл айнымалысын JavaScript тілінде келесі түрде жазуға болады:

```
for(var i = 100; i; i--) ...
```

немесе

```
var i;
for(i = 100; i; i--) ...
```

Браузер үшін бұл екеуі де бірдей болады. Келесі цикл құрылымында:

```
for(басы; шарты; өсуі)
    команда;
```

басы командасы цикл басында бір-ақ рет орындалады. Сондықтан **i** айнымалысы 100 рет **var** сипаттауышы арқылы тексерілмейді.

Келесі ұсыныстар беріледі:

- і цикл санауышын функция немесе скрипт басында басқа айнымалылар сипатталаған кезде сипаттап кету керек;
- егер бір санауыш бірнеше циклде пайдаланылатын болса, оны жалпы сипаттау бөлімінде көрсету қажет;
- егер функция құрамында цикл біреу ғана болса, онда **for (var i = …, …; …)** – түрінде жазған дұрыс, өйткені **i** айнымалысы бір-ақ рет кездеседі.

Егер бір **digit** айнымалысы цикл ішінде пайдаланылып, оның сыртында кездеспесе, оны циклдің ішінде локальді айнымалы ретінде сипаттаған дұрыс.

Төмендегі **while** командасының жазылуы:

```
while (...)
{ var digit = ...
  ... }
```

мынадай сипаттаумен бірдей болып саналады:

```
var digit;
...
while (...)
{... digit = ...}
```

Бақылау сұрақтары:

1. While операторының көмегімен цикл қалай ұйымдастырылады?
2. Циклдің орындалу реттілігіне түсініктеме беріңіз.
3. For операторының көмегімен цикл қалай ұйымдастырылады?
4. Break және continue командалары қай кезде қолданылады? Олардың айырмашылықтар қандай?
5. [0,1] аралығынан кездейсоқ санды беру үшін қандай функция қолданылады?
6. Циклді кері бағытта программалау дегенді қалай түсінуге болады?
7. Цикл айнымалысын қалай сипаттауға болады? Оның қандай нұсқалары бар?
8. Функцияны кестелеуде нәтижені жеке терезелерге шығаруды қалай ұйымдастыруға болады?

Тапсырмалар:

1. 1-ден 20-ға дейінгі кездейсоқ тақ сандардың қосындысын есептеу программасын құрыңыздар.

2. y=5x+1 функциясының мәнін, x-тің мәні 1 мен 5 аралығында 0,5 қадаммен өзгергенде анықтауды жеке терезеге және бір терезеге шығару программасын құрыңыздар.

3. $$y=\begin{cases} x^4+x^2+1, & егер\ x\le 0 \\ \sqrt{x+1}, & егер\quad x>0 \end{cases}$$

функциясының мәнін *x*-тің мәні -2 мен 2 аралығында 0,1 қадаммен өзгергенде анықтауды жеке терезеге және бір терезеге шығаруды ұйымдастырыңыздар.

4. 5-тен 15-ке дейінгі кездейсоқ сандардың келесісі 9-ға тең болғанда, олардың қосындысын есептеуді тоқтатып, нәтиже шығаратын программа құрыңыздар.
5. 1-ден 5-ке дейінгі кездейсоқ жұп сандардың көбейтіндісін табу программасын құрыңыздар.
6. Жоғарыдағы мысалдарға қарай отырып, келесі мысалдарды орындау кезінде `alert` терезесіне қандай мәлімет шығатынын ауызша айтуға (кейіннен оны тексеру қажет) тырысыңыздар. Егер қателер болса, олардың қалай пайда болғанын түсіндіріңіздер.

11-мысал
```
var s = 0;
for(var i=1; i<10;i++)
     s += i;
alert(s);
```

12-мысал
```
var s = 0;
for(var i=1; i<10; i+=2)
     s += i;
alert(s);
```

13-мысал
```
var s = 0;
for(var i = 10; i; i --)
    s += i;
alert(s);
```

14-мысал
```
var s = 0;
for(var i = 10; -- i;)
    s += i;
alert(s);
```

15-мысал
```
var s = 0;
for(var i = 10; i --)
    s += i;
alert(s);
```

16-мысал
```
var s = 0;
for(var i = 10;;)
(if(--i) break;  s+- i;}
alert(s);
```

17-мысал
```
var s = 0;
var i = 10;
for(;;)
{if(!(--i) break; s += i;}
alert(s);
```

18-мысал
```
var s = 0;
var i = 10;
for(;;)
{if(s>6) break; s+= --i; }
alert(s);
```

19-мысал
```
var s = 0;
for(var i = 10;--i; s+= i);


alert(s);
```

20-мысал
```
var s = 0;
for(var i = 10; -- i;)
{if(i > 5) continue;
 s += i;}
alert(s);
```

21-мысал
```
var s = 0;
var i =10;
while(-- i)
{if(i > 5) continue; s +=
i;}
alert(s);
```

22-мысал
```
var s = 0;
var i = 10;
while(-- i)
{if(i < 5) break; s += i;}

alert(s);
```

Төмендегі мысалдардағы сан деп отырғанымыз натурал сандар болып есептеледі.

7. Келесі есептерді `for` және `while` циклдері арқылы орындайтын программалар жазып шығыңыздар:

а) 15! = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 *10 *11 *12 *13 *14 * 15;

ә) берілген сан цифрларының қосындысын анықтау керек;

б) берілген сан цифрларының көбейтіндісін табу керек;

в) берілген сан цифрларының ең үлкенін табу керек;

г) берілген сан цифрларының ең үлкенін табу керек;

д) берілген сан цифрларының ішінде неше рет «3» цифры кездеседі;

е) берілген сан цифрларының ішінде неше рет «3» және неше рет «5» цифры кездеседі;

ж) берілген сан цифрларын кері бағытта жазып шығыңыздар;

з) берілген сан цифрлары арасына «+» белгісін жазу керек;

и) берілген сан цифрларының әрқайсысын екі еселеңдер.

8. `For` циклін пайдалана отырып, келесі есептерді шығаратын программа құрыңыздар:

а) [1, 100] аралығынан алынған 10 кездейсоқ санның қосындысын анықтау керек;

ә) [1, 100] аралығынан алынған 10 кездейсоқ жұп санның қосындысын анықтау керек;

б) $a_1 = 5$; $a_2 = 10$; ... болып келетін арифметикалық прогрессияның 10 мүшесін және солардың қосындысын анықтау қажет;

в) $a_1 = 5$; $a_2 = 10$; ... болып келетін геометриялық прогрессияның 10 мүшесін және солардың қосындысын анықтау қажет;

г) 8 бірліктен тұратын екілік санды (11111111_2) ондық санау жүйесіне айналдыратын программа жасау керек;

д) 5 бірліктен тұратын оналтылық санды (11111_{16}) ондық санау жүйесіне айналдыратын программа құру қажет;

е) 8 екіліктен тұратын үштік санды (22222222_3) ондық санау жүйесіне айналдыратын программа құрыңыздар;

ж) і-ші орында тұрған цифры i^2 санының соңғы цифры болып келетін 10 таңбалы санды анықтаңыздар;

з) і-ші орында тұрған цифры $a_j = 11 * i + 13$ тізбегінің соңғы цифры болып келетін 10 таңбалы санды табыңыздар;

и) **`math.random()`**функциясының 0 және 1 сандарын беретінін немесе бермейтінін тексеріп шығу керек;

к) **`math.random()`**функциясын 1000 рет пайдаланғанда, (0,4; 0,6) аралығында орналасатын сандардың нешеу екенін анықтау қажет.

6. JavaScript тұрақтылары мен функциялары

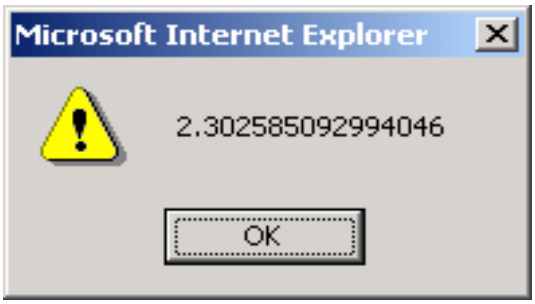
Барлық стандартты математикалық функциялар мен жиі қолданылатын тұрақтылар **`Math`** математикалық объектісінің тәсілдері мен қасиеттері түрінде беріледі.

`Math` объектісі JavaScript тілінің ішкі объектісі болып табылады. Мысалы, PI тұрақтысын пайдалану үшін, оны **`Math.PI`** түрінде жазу керек. JavaScript тұрақтыларының дәлдігі үтірден кейінгі 15 цифрды қамтиды. **`Math`** функциялары тәсіл ретінде, мысалы, синус функциясы – **`Math.sin(argument)`** түрінде жазылады, мұндағы ***argument*** функция аргументі болып табылады.

6.1. Стандартты тұрақтылар

1. Натуралдық логарифм негізі – **E** тұрақтысы (2,718281828) түрінде жазылады.

Мысалы:
```
alert(Math.E)
```
2. 10 санының натуралдық логарифмі – **LN10** тұрақтысы болып табылады.
Мысалы (13.9-сурет):
```
alert(Math.LN10)
```



13.9-сурет. **LN10** тұрақтысы мәнін экранда бейнелеу

3. Екі санының натуралдық логарифмі – **LN2**.
Мысалы:
```
alert(Math.LN2)
```
4. π саны – **PI** болып жазылады.
Мысалы:
```
alert(Math.PI)
```
5. 1/2 санының квадраттық түбірі – **SQRT1_2** .
Мысалы:
```
alert(Math.SQRT1_2)
```
6. 2 санының квадраттық түбірі – **SQRT2** .
Мысалы:
```
alert(Math.SQRT2)
```

6.2. Стандартты функциялар

1. Аргументтің абсолюттік мәні – **abs(arg)**, мысал:

13.10-сурет. **Abs(-25)** функциясы мәнін экранда бейнелеу

```
var x = -25;
var y
y = Math.abs(x);
alert(y);
```
2. **sin(arg), cos(arg), tan(arg)** тригонометриялық функциялары:
мысал:
```
var x = Math.PI/4;
var y
y = Math.sin(x);
alert(y);
```
3. Кері тригонметриялық функциялар **asin(arg), acos(arg)**, **atan(arg)** :
```
var x = 0.5;
var y
y = Math.asin(x);
alert(y);
```
4. Экспонента және натуралдық логарифм: **exp(arg), log(arg)**
```
var x = 1;
var y
y = Math.exp(x);
alert(y);
```
5. Аргументтен үлкен немесе соған тең сан (13.10-сурет): **ceil(arg)**

13.11-сурет. **Ceil(-2.69)** функциясы мәнін экранда бейнелеу

```
var x = -2.69;
var y
y = Math.ceil(x);
alert(y);
```
6. Аргументтен кіші немесе соған тең сан:
```
floor(arg)
var x = -2.69;
```

```
var y
y = Math.floor(x);
alert(y);
```
7. Аргументті жақын бүтін санға дейін дөңгелектеу: **round(x)**
```
var x = -2.69;
var y = Math.round(x);
alert(y);
```
8. Экспонента мен натуралдық логарифм: **exp(arg), log(arg)**
```
var x = 1;
var y
y = Math.exp(x);
alert(y);
```
9. Аргументтің квадраттық түбірі : **sqrt(arg)**
```
var x = 3;
var y
y = Math.sqrt(x);
alert(y);
```
10. Екі аргументтің үлкенін немесе кішісін анықтау **max(arg1, arg2), min(arg1, arg2):**
```
var x = 3;
var y = Math.min(x,10);
alert(y);
```
11. Санның дәрежесін табу **pow(arg1, arg2)** : **arg1**-дің **arg2** дәрежесі
```
var x = 2;
var y = Math.pow(x,10);
alert(y);
```
12. [0,1] аралығынан кездейсоқ сан алу**: random()**
```
alert(Math.random());
```

6.3. Тұтынушы функциялары

Программалауда бір командалар тобын бірнеше рет қайталауға тура келетін жағдайлар жиі кездеседі. Егер қайталау «бір орында» орындалатын болса — (while, for) циклдер қолданылады. Ал егер кодтарды программаның әр жерінде қайталау қажет болса – оны тұтынушы функциясы ретінде жазу керек болады.

Функцияларды сипаттау және пайдалану. Мысалы, егер берілген бүтін санның цифрларының қанша екенін анықтау керек болса, оны бір рет анықтап алып, одан кейін математикадағы сияқты F(num) деп жазып, ол функцияның ішкі әрекетін қарастырмай, кодтарын жазбай, нәтижесін есептеуге болады. Мысалы, тұтынушы үш сан енгізіп солардың жалпы цифрлары санын анықтауы қажет болсын.

Осы мысалдың программасы:

```
// Бүтін оң num санындағы цифрлар санын анықтау
// Енгізу: num (бүтін оң сан).
// Шығару: num санындағы цифрлар саны.
function F(num)
{
 var len = num ? 0 : 1;
 while(num)
 {
     num = (num - num % 10) / 10;
     len++;
 }
 return len;
}
```

```
var sum = 0;
var num1, num2, num3;
num1 = prompt("Бірінші санды енгізіңіз", "");
sum += F(numl) ;
num2 = prompt("Екінші санды енгізіңіз", "");
sum += F(num2) ;
num3 = prompt("Үшінші санды енгізіңіз", "");
sum += F(num3) ;
alert("Енгізілген цифрлардың жалпы саны: " + sum);
```

Браузер бұл скриптіні былай орындайды. Мұндағы мына тізбек:

```
function F(num)
{
  ...
}
```

бірден орындалмайды, бірақ браузер «F атты функция сипатталды» деген мәліметті есте сақтайды. Өйткені **function** түйінді сөзі және оның блогы {...} командалар тізбегі емес, тек декларация (хабарлама, сипаттама) ғана болып саналады.

Функциядан кейін орналасқан командалар әдеттегідей орындалады. Соның ішінде F функциясы кездессе, браузер функция сипаттамасына оралып, оның формальді аргументі num орнына num1, num2, num3 тәрізді нақты параметрлерді қойып орындап шығады. Негізінде параметрлерді ауыстыру кезінде браузер F функциясын орындамай тұрып, мынадай меншіктеулер жасайды:

num = num1; — содан соң **F(numl)** орындау;
num = num2; — содан соң **F(num2)** орындау;
num = num3; — содан соң **F(num3)** орындау.

Жалпы функцияны сипаттау былай атқарылады:

function Функция_аты(үтірмен бөлінген формальді аргументтер тізімі)
{
...
функция тұлғасы
...
return (мәні);
}

return командасы программада қолданылатын функция мәндерін анықтайды, ол бірнешеу болуы да мүмкін, тіпті болмауы да мүмкін. Ол болмаса, функция ешқандай мән бермейді, мұндайда функцияны өрнектерге енгізуге болмайды. Мысалы, мәні жоқ функцияны меншіктеу командасында пайдалануға болмайды.

Функцияны шақыру (пайдалану) былай орындалады:

Функция_аты (үтірмен бөлінген нақты аргументтер тізімі)

Нақты аргументтер ретінде тұрақты, айнымалы, өрнек немесе сан қолданылады. F атауын функцияға дұрыс берілген атау деп айта алмаймыз, мұнда математикадағы сияқты функция бір әріппен белгіленген. Программалауда бір әріппен емес, бір сөзбен белгілеу қалыптасқан, ол функция әрекетін білдіретін сөз болуы тиіс. Жоғарыда келтірілген мысалда DlinaChisla немесе LenOfNumber атаулары F атауына қарағанда түсініктірек болар еді. Функция аты шектелмейді, бірақ ол бір сөзден ғана тұруы тиіс (бос орын болмауы керек). Атау латын әріптері мен цифрлардан тұрады, астын сызу таңбасын қолдануға болады. Орыс, қазақ әріптерін пайдалануға болмайды. Атаудың алғашқы символы әріп немесе астын сызу таңбасы болуы керек. Әріптер регистрі бірдей болып қабылданбайды. Мысалы, «LenOfNumber» және «Lenofnumber» атаулары бірдей емес болып саналмайды.

Бақылау сұрақтары:

1. JavaScript тілінің функциялары қалай жазылады?
2. Тұтынушы функциясы қай кезде қолданылады?
3. 10 санының квадрат түбірін жазыңыз.
4. x^5 өрнегі қалай жазылады?
5. x пен y-тің үлкенін анықтау қалай жазылады?
6. 5 санының натурал логарифмі қалай жазылады?
7. *x* пен *y*-тің кішісін анықтау қалай жазылады?

Тапсырмалар:

1. 4 сан енгізіп олардың үлкенін анықтауды екі санның үлкенін анықтау функциясын қолдана отырып, программа құрыңыз.
2. Стандартты санды дәрежелеу функциясын қолданбай, кез келген енгізілген санның 5 дәрежесін анықтау программасын құру керек.
3. Үш таңбалы сан цифрларының қосындысын анықтау программасын құрыңыз.
4. sin*x* пен cos*x*/2 функциясының мәндерінің үлкенін анықтау программасын құрыңыз.
5. Кез келген енгізілген санның квадрат түбірін, квадратын, абсолют шамасын шығаратын программа құрыңыз.
6. Төмендегі программаны теріп, оның қалай жұмыс істейтінін түсіндіріңдер:

```
<HTML>
<HEAD>
<TITLE> Функция </TITLE>
</HEAD>
<BODY link=blue alink=red vlink=purple>
<H1> Функция жұмысын тексеру </H1> <hr>
<p id="p1"> Бұл мысал осы мәтіннің қаріптері көлемін өзгертуді
жүзеге асырады. </p>
<br> <b> <i> Қаріп көлемі: </i> </b>
<input type="Button" value="Үлкейту" onClick="modification(1);">
<input type="Button" value="Кішірейту"
                                    onClick="modification(-1);">
<input type="Button" value="Алғашқы қалып"
                                    onClick="modification(0);">
<script language="JavaScript">
function modification(action)
{
  if(action==1){
  document.all["p1"].style.fontSize="150%";}
  if(action==-1){
  document.all["p1"].style.fontSize="60%";}
  if(action==0){
  document.all["p1"].style.fontSize="100%";}
}
</script>
</BODY>
</HTML>
```