11 ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС

HTML ТІЛІНДЕ СКРИПТЕРДІ ПАЙДАЛАНУ

1. Программаға скриптер енгізу

Кейбір JavaScript тілін сүйемелдемейтін браузерлердің бұрынғы нұсқалары оларды орындамайды. Ондай браузерлер скриптерді түсінбейтін болғандықтан, JavaScript тілінің операторларын комментарий тәгтерінің `<!-- ... -->` ішіне орналастыру қалыптасқан. Тіл интерпретаторының дұрыс жұмыс істеуі үшін скриптерді жабу тәгінің `-->` алдына // белгісін қою керек. Сонымен, HTML-құжат ішіне JavaScript сценарийін орналастыру үшін төмендегідей жолдарды жазу керек:

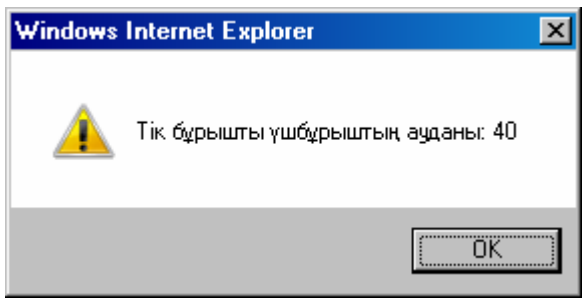
```
<script language=JavaScript>  // Скрипт басы
<!--                          // Комментарий белгісі
...
JavaScript  кодтары
...
//-->                         // Комментарий соңы
</script>                     // Скрипт соңы
<noscript>
...
JavaScript тілін сүйемелдей алмайтын браузерлерге арналған мәліметтер
...
</noscript>
```

Скриптердегі қолданылатын тілді анықтайтын параметрлердің бірі – <script> тәгінің language атрибуты болып табылады. JavaScript тілі үшін параметрдің мәні – "JavaScript". Егер басқа, мысалы, VBScript сценарийлер тілі қолданылуы керек болса, онда параметрдің мәні – "VBScript" сөзі болуы керек. Браузер алдын ала (үнсіз) келісім бойынша JavaScript тілін таңдайтын болғандықтан, language параметрін жазбаса да болады. Браузер JavaScript тілін қолдамайтын болса немесе оны сүйемелдеу алып тасталынған жағдайда, <noscript> тәгінде жазылған сөз тіркесі экранға шығарылады. Бұл тәг <script> ... </script> тәгтерінен кейін жазылады. Келесі мысалдарда браузер JavaScript тілін толық сүйемелдейді деп есептейміз.

Бір құжатта бірнеше <**script**> тәгтері болуы мүмкін. Олар орналасу реттілігі бойынша JavaScript интерпретаторы арқылы өңделеді. Келесі мысалда HTML-құжаттың <BODY> бөлігінде (құжат тұлғасында) JavaScript тілінің операторлары орналасқан.

1-мысал. Катеттері берілген тік бұрышты үшбұрыштың ауданын анықтайтын программа жазу керек болсын делік.

```
<HTML>
<HEAD> <title> Бірінші мысал </title>
</HEAD>
<BODY>
<P> Скриптер енгізілген парақ </P>
<script>
<!-- //
var a=8; h=10
alert ("Тік бұрышты үшбұрыштың ауданы: " + a*h/2)
//-->
</script>
<P> Скриптері бар парақтың соңы </P>
</BODY>
</HTML>
```



11.1 сурет. Үшбұрыштың ауданын табу

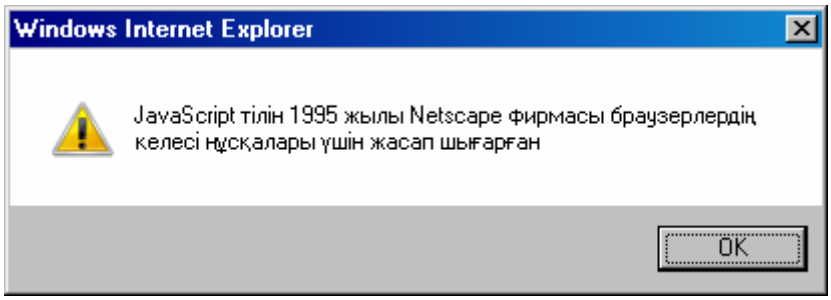
Скриптерде екі айнымалы сипатталып инициалданады, содан соң өрнектің мәні құжатқа жазылады. Мұнда JavaScript тілінің экранға мәлімет шығаратын **alert** функциясы қолданылған, ол ішінде ОК батырмасы бар ақпараттық шағын терезе (11.1-сурет) ашады.

Терезедегі мәлімет оқылып болған соң, батырма басылса, терезе экраннан алынып тасталады да, браузер келесі командаларды орындауға көшеді.

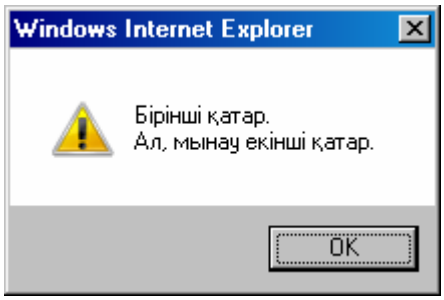
Егер alert функциясының аргументі ретіндегі мәтін көлемді болса, оны «+» (біріктіру операциясы) таңбасы арқылы бірнеше жолға жазуға болады, мысалы:

```
alert("JavaScript тілін 1995 жылы Netscape фирмасы " +
      "браузерлердің келесі нұсқалары үшін жасап шығарған");
```

Мұндай функция нәтижесі келесі 11.2-суреттегідей болады:



11.2 сурет. Терезеге көлемді мәлімет шығару



11.3 сурет. Мәліметтерді екі жолға шығару

Alert() функциясында, қажет болғанда мәліметті «\n» символы арқылы екі-үш жолға бөліп шығару мүмкіндігі де қарастырылған (11.3-сурет):

```
alert("Бірінші қатар.\nАл, мынау екінші қатар.");
```

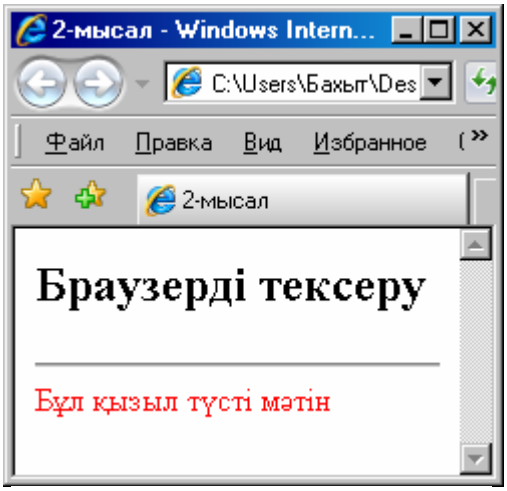
Alert() функциясы жиі қолданылады, тұтынушыға мәлімет берумен қатар ол программадағы қателерді іздеп түзету кезінде де кең пайдаланылады.

Келесі скрипт Web-параққа мәтін шығарып, HTML тәгтерін де пайдалана отырып, құжатты толық бейнелейді. Мұнда мәтін қызыл түсте шығарылады.

2-мысал:

```
<SCRIPT LANGUAGE="javascript">
document.write ("<FONT COLOR='RED'> Бұл қызыл түсті мәтін </FONT>")
</SCRIPT>
```

Мысал нәтижесі:



11.4 сурет. 2-мысал нәтижесі

Мұнда **document** сөзі шығарылатын мәтінді (HTML құжатын) жариялайды. Ол құжатқа бір сөз немесе сан жазылуы (write) тиіс. Жазылатын тізбек жақша ішінде функция тәрізді көрсетіледі. DOCUMENT *объект* болып саналады. Одан нүктемен бөлініп жазылған WRITE (жазу) сөзі *объект тәсілі* деп аталады. Бұл сценарий жұмысы: "Бір объект (бұрыннан белгілі) құрып, оған мәлімет жазу керек" дегенді білдіреді.

Жақша ішіндегі мәтін тырнақшаға алынып жазылғанына назар аудару керек. HTML тілінде олар қажет етілмейтін, ал мұнда олар міндетті болып табылады. Жақша ішіндегі мәтін – қарапайым HTML кодтары. RED сөзі жалқы

тырнақшаға (апостроф) алынған, егер қостырнақша қойылса, онда сөз тіркесі толық аяқталған болар еді. Мұнда олай емес екендігі білініп тұр, яғни қостырнақша ішінде басқа бір тіркесті пайдалану үшін оны жалқы тырнақшаға алу керек.

2. Меншіктеу тәсілдері

Айнымалылар алдын ала сипатталуы тиіс, ол үшін **var** түйінді сөзі қолданылады:

```
var x;                    // x айнымалысын сипаттау
var y=5;                  // y айнымалысын мән бере отырып сипаттау
var mes="Скриптер тілі"; // Сөз тіркесін мән меншіктей отырып сипаттау
```

Айнымалылар идентификаторлармен белгіленеді. *Идентификатор — міндетті түрде әріптен басталатын латын әріптері мен цифрлар тізбегі.*

Астын сызу белгісі «_» әріп болып саналады. *Бас әріп пен кіші әріп әр түрлі символ болып табылады*. Мысалы, *Counter* және *counter* идентификаторлары екі түрлі айнымалыны көрсетіп тұр.

Айнымалының типі сценарийде сақталған ақпараттың типіне байланысты болып, сол мәліметтің типі өзгерсе, айнымалының типі де өзгереді.

Меншіктеу операторынан кейін айнымалының мәні өзгереді. Меншіктеу операторы программаның кез келген жерінде тұра береді және ол айнымалының тек мәнін ғана емес, типін де өзгерте алады. Меншіктеу операторы мынадай түрде беріледі:

```
a=b
```

мұндағы a – біз бір мән бергіміз келетін айнымалы, b – айнымалының жаңа мәнін анықтайтын өрнек.

Сценарийде мынадай айнымалылар жазылған болсын делік:

```
var n=3725
var x=2.75
var p=true
var s="Программа орындалды"
```

n және x айнымалыларының типі number, p айнымалысының типі – логикалық, s айнымалысының типі string. JavaScript тілінде барлық стандартты және тұтынушының өзі анықтайтын функциялар үшін function типі анықталған. JavaScript объектілері үшін object типі бар. Object типті айнымалылар объектілерді сақтайтындықтан, оларды жай объектілер деп атай береді.

Сценарийлердегі <HEAD> бөлігінде және <BODY> бөлігінде де сипатталған айнымалылар екеуінде де бірдей жұмыс істей береді, бұларды осы құжаттағы кез келген сценарий пайдалана алады. Мұндай айнымалылар *глобальді* (ауқымды) деп аталады, ал функция ішінде анықталған айнымалылар *локальді* , яғни жергілікті болып саналады.

Меншіктеу операторының оң жағында орналасатын өрнек операндтардан (мәндер мен айнымалылардан) және операция таңбаларынан (+, -, *, /) тұрады. Мысалы, a*b формуласында a және b операндтар, * таңбасы – көбейту операциясы.

Есептелген мәннің типіне қарай өрнек арифметикалық, логикалық және тіркестік типтердің біріне жатқызылады. 1.1 кестеде көрсетілген арифметикалық операциялардың орындалу нәтижесінде өрнектер құрылады.

11.1 кесте. Арифметикалық операциялар

Операция	Аталуы
+	Қосу
-	Азайту
*	Көбейту
/	Бөлу
%	Бүтін сандарды бөлгеннен кейінгі қалдық
++	Операнд мәнін бірге өсіру
--	Операнд мәнін бірге кеміту

Өрнектегі операторлар арифметикалық операциялардың басымдылықтарына (приоритеттеріне) қарай солдан оңға қарай есептеледі. Керекті жағдайда өрнекке жақша енгізу арқылы операциялар реттілігін өзгертуге болады. JavaScript тілінде теңдіктің оң және сол жағында орналасқан операндтарға арифметикалық амалдар қолданып, нәтижені сол жақтағы операндқа меншіктейтін операторлар анықталған. Операциялардың мұндай қысқарған түрлері 1.2 кестеде көрсетілген.

11.2 кесте. Меншіктеу операторының қысқаша жазылу жолдары

Оператор	Соған сәйкес меншіктеу операторы
X += Y	**X = X+Y**
X –= Y	**X = X-Y**
X *= Y	**X = X*Y**
X /= Y	**X = X/Y**
X %= Y	**X = X%Y**

3. Логикалық операциялар

Шарт ретінде логикалық өрнектер де жазыла береді, ондайда келесі логикалық операциялар қолданылады:

Белгіленуі	**Сипаттамасы**	**Мысалы**
= =	Тең	x+1==8
!=	Тең емес	str !=”yes”
>	Үлкен	x*y>5
>=	Үлкен немесе тең – кіші емес	d>=0
<	Кіші	num>10
<=	Кіші немесе тең – артық емес	bonus<=5
&&	Логикалық ЖӘНЕ	1 < x && x < 10
||	Логикалық НЕМЕСЕ	x== 1 || x == 10
!	Логикалық ТЕРІСТЕУ	! (1 < x && x < 10)

Логикалық қатынас операциялары кез келген типтегі операндқа қолданыла береді. Операция нәтижесі – салыстыру мәндері дұрыс болса, логикалық `true`, ал оған кері жағдайда – `false` мәні болып табылады.

! (логикалық ЕМЕС) операциясы логикалық типтегі операндтарға қолданылады, егер операнд мәні `true` болса, онда !a өрнегінің мәні – `false`, ал егер операнд мәні `false` болса, онда !a = `true` болады. && (логикалық ЖӘНЕ) пен | | (логикалық НЕМЕСЕ) операцияларының қолданылу ережесі 1.3 кестеде көрсетілген.

11.3 кесте. Логикалық операциялардың орындалу ережелері

A	**B**	**A&&B**	**A||B**
`true`	`true`	`true`	`true`
`true`	`false`	`false`	`true`
`false`	`true`	`false`	`true`
`false`	`false`	`false`	`False`

A&&B өрнегінің мәні – екі операнд мәндері де ақиқат болғанда ғана ақиқат, қалған жағдайларда жалған болады. A||B өрнегінің мәні – кем дегенде, бір операнд ақиқат болса, ақиқат, ал басқа жағдайларда – жалған болып саналады.

Символдар тізбегінен тұратын тіркестік мәндер үшін *символдарды конкатенациялау (біріктіру)* операциясы анықталған. Операция «плюс» белгісімен белгіленеді. Бұл операцияның нәтижесі екі операндтар тізбегінен құралатын жаңа сөз тіркесі болады, мысалы, төмендегі меншіктеу операторының орындалуы нәтижесінде

```
st = "озат "+"студент"
```

st айнымалысы `"озат студент"` мәнін меншіктейді. Тағы бір мысал қарастырайық. Мынадай екі оператор берілген болсын:

```
stl = "озат "; st2 = "студент"
```

Төмендегі операторды орындау нәтижесінде

```
stl += st2
```

`st1` айнымалысы "`озат студент`" мәнін меншіктейді.

Операцияның басымдылығы өрнектегі амалдардың орындалу реттілігін анықтайды. 1.4 кестеде орындалу басымдылығы кему реті бойынша орналасқан операциялар тізімі қарастырылған.

11.4 кесте. Операциялардың басымдылық кестесі

Операция аты	Белгіленуі
Инкремент	`++`
Декремент	`--`
Терістеу	`!`
Унарлы минус	`-`
Көбейту	`*`
Бөлу, қалдық табу	`/, %`
Қосу	`+`
Азайту	`-`
Салыстыру	`<, >, <=, >=`
Теңдік	`==`
Теңсіздік	`! =`
Логикалық ЖӘНЕ	`&&`
Логикалық НЕМЕСЕ	`||`
Меншіктеу	`=, +=, -=, *=, /=, %=, ! =`

Мынадай командалардың ++x және x++ (--x және x--) айырмашылығы бұлар басқа командалар құрамына кіріп тұрғанда ғана есепке алынады. Алғашқысында (олар айнымалы алдында) операция айнымалыны пайдаланғанға дейін орындалады, ал соңғысында – пайдаланғаннан кейін орындалады.

x = 5; Меншіктеулерінен кейін:
y = ++ x; x және y-тің мәні 6-ға тең.
x = 5; Ал мына меншіктеулерден соң:
y = x ++; x-тің мәні 5-ке, ал y-тің мәні 6-ға тең болады.

Келесі мысалда:

```
var x = 1;
var y;
y = (x += 2) + 1;
```

y айнымалысының мәні 4, ал x айнымалысының мәні – 3. Мынадай тізбекті түрдегі меншіктеулерді де пайдалануға болады :

```
x = y = z = t = өрнек;
```

Мұнда бірнеше айнымалының бәріне бір ғана мән меншіктеледі.

«**++**» және «**--**» операциялары тек айнымалыларға тіркеледі, оларды өрнектерге қосып жазуға болмайды.

4. Функцияларды сипаттау және пайдалану

Программа құрғанда ондағы логикалық тәуелсіз бөліктерді (*программа ішіндегі подпрограммалар*) жеке-жеке бөліп қарауға болады. Программаны мұндай бөліктерге бөліп қарастыру оның жұмысын түсінуді жеңілдетеді. Программаны ішкі подпрограммаларға бөлу оны түзету ісін жеңілдетеді. Оның үстіне бір жазылған подпрограмманы әр түрлі программаларда қолдануға болады. Көптеген программалау тілдерінде подпрограммалар функциялар, модульдер және т.б. көмегі арқылы жүзеге асырылады.

Функция – JavaScript тілінің негізгі элементі. Функция мынадай түрде сипатталады

```
function F(v) {S}
```

мұндағы `F` – функцияны шақыруға болатын ат тағайындайтын функция идентификаторы; `v` – үтір арқылы бөлініп жазылатын функция параметрлерінің тізімі; `S` – нәтиже алу үшін орындалатын іс-әрекеттерден тұратын функция тұлғасы, яғни операторлар тізбегі. Міндет-

ті түрде жазылмайтын `return` операторы функцияның программаға қайтарылатын мәнін анықтайды.

Бір функцияның сипаттамасы басқа бір функция сипаттамасы ішіне жазылмайды. Функция параметрлері оның ішкі операторларында, яғни тұлғасында қарапайым айнымалылар рөлін атқарады, бірақ бұл параметрлерге бастапқы мәндер функцияны шақырғанда ғана беріледі. Егер сипаттама мынадай түрде болса:

```
function F(vl,v2,...,vn) {S}
```

онда функцияны шақыру мынадай болуы керек:

```
F(el,e2,...,en)
```

мұндағы `el,e2,...,en` – параметрлердің *нақты (нақтық)* мәндерін беретін өрнектер. Функция сипаттамасында көрсетілген `vl,v2,...,vn` параметрлері функцияны шақыру кезінде `el,e2,...,en` сияқты нақты параметрлерді бергеннен кейін мән қабылдайтын болғандықтан, *формальді* параметрлер деп аталады. Егер функцияда параметрлер болмаса, функция сипаттамасы мынадай түрде болады:

```
function F() {S}
```

Операторда функция атынан соң, жақшаның болуы міндетті, яғни функцияны шақыру мына түрде болуы керек:

```
F()
```

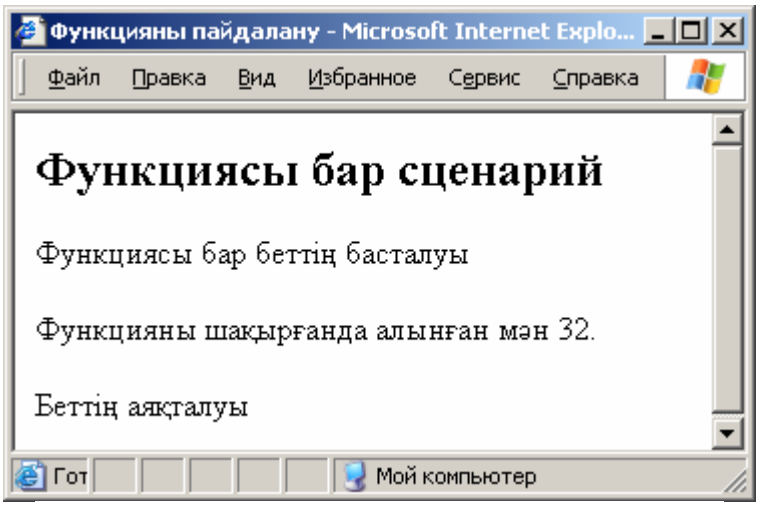
Көбінесе барлық анықтаулар мен функциялар құжаттың <HEAD> бөлігінде беріледі. Бұл құжатты браузерге жүктегенде, оны интерпретациялауды және барлық функциялардың компьютер жадында сақталуын қамтамасыз етеді.

Келесі мысалда катеттері берілген тікбұрышты үшбұрыш ауданын есептеудегі `care` функциясы құжаттың тақырып бөлігінде сипатталған.

Функциясы бар сценарий мысалы. Қабырғасы мен биіктігі берілген үшбұрыштың ауданын анықтайтын сценарий жазу қажет. Үшбұрыштың ауданын есептеу үшін HTML-құжаттың <HEAD> бөлігінде сипатталған care функциясын қолданамыз.

3-мысал. Функциясы бар сценарий жазу

```
<HTML> <HEAD> <H2> Функциясы бар сценарий</H2>
                <title> Функцияны пайдалану </title>
<script language="JavaScript">
<!-- //
  function care(a,h) {return a*h/2}
//-->
</script>
</HEAD>
<BODY>
<P> Функциясы бар
  беттің басталуы </P>
<script>
<!--
  var a1=4; h1=16
  var s=care(a1,h1)
document.write("Функцияны
шақырғанда алынған мән ",
s,".");
//-->
</script>
<P>Беттің аяқталуы </P>
</BODY> </HTML>
```



11.5 сурет. Функциясы бар сценарий нәтижесі

Функция тұлғасы функцияның қайтарылатын мәнін анықтайтын тек `return` операторынан тұрады. `s=care(a1,h1)` меншіктеу операторы орындалғанда құжат тұлғасын-

дағы функцияны шақыру іске асырылады. `A` және `h` формальді параметрлеріне `a1` и `s` нақты параметрлерінің мәні меншіктеледі де, функция тұлғасы есептеледі. Табылған мән `write` тәсілі арқылы экранға шығарылады.

Жаттығулар

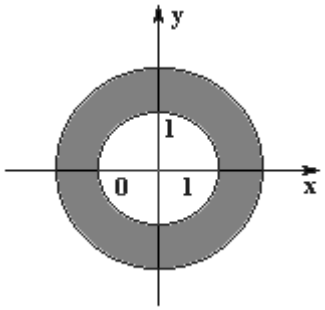
1. Төмендегі тұжырымдар нәтижесінде ақиқат болатын өрнек жазыңыздар:

- `m` бүтін айнымалысы `k` бүтін айнымалысына қалдықсыз бөлінеді;
- `A`, `B` және `C` нақты айнымалылардың мәні кемімейтін реттілік құрайды;
- үш `x,y,z` айнымалыларын жұптасқан түрде қарастырғанда, олардың ішіндегі ең үлкені `x` айнымалысы болып шыққан;
- `A,B,C` логикалық айнымалылары мәндерінің бірде-біреуі ақиқат емес;
- A, B, C логикалық айнымалылары мәндерінің, кем дегенде, біреуі ақиқат.

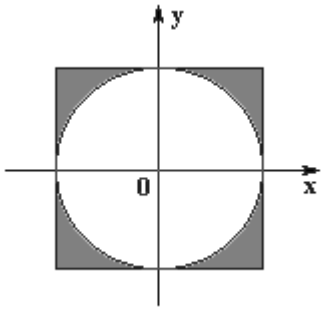
2. Берілген `x` және `y` мәндері жазықтықтағы нүкте координаталары болып табылады, төмендегі өрнектердің әрқайсысы `true` мәнін қабылдайтын аймақты штрих сызықпен бояңыздар:

- `(X*Y>0);`
- `Y+X<5 && X*X+Y*Y>i;`
- `Y<X*X+2 || Y>6.`

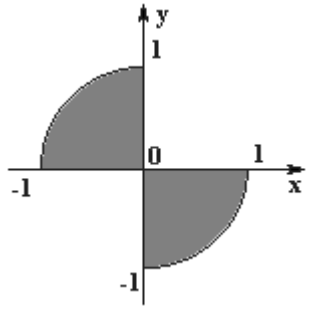
3. Жазықтықтағы координаталары **`x`** және **`y`** болып келген нүкте үшін, оның штрихталған аймаққа түсуі ақиқат болып келетін формула жазыңыздар (11.6 – 11.9 сурет).



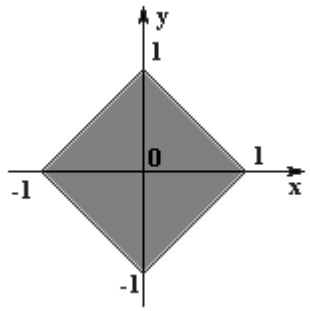
11.6 сурет. Нүкте екі шеңбермен қоршалған аймаққа түседі



11.7 сурет. Нүкте квадрат пен шеңбер аралығындағы аймаққа түседі



11.8 сурет. Нүкте көрсетілген екі секторға түседі



11.9 сурет. Нүкте ромб ішіне түседі