MON.ITOR.US

Բովանդակություն

Ներածություն

Mon.itor.us-ը web կայքերի մոնիթորինգի համար նախատեսված գործիք է, որն աշխատում է օրական 24 ժամ և միշտ տեղյակ պահում իր օգտագործողներին վերջիններիս կայքերի կապված պրոբլեմների մասին: Դա արվում է ինչպես mail-երի, այնպես էլ messenger-ների և sms հաղորդագրությունների միջոցով: Բացի անմիջական արձագանքներից , Mon.itor.us-ը տրամադրում է նաև անալիտիկ ինֆորմացիա կայքերի մասին: Նմանատիպ անալիտիկ (ինչպես նաև այլ տիպի) ինֆորմացիա օգտագործողներին դյուրին ձևով տրամադրելու համար նախագծվել է Google Calendar-ի ինտեգրացում Mon.itor.us- ի վրա: Վերջինս հնարավորություն է տալիս ունենալ սեփական Google օրացույց Mon.itor.us կայքում:

- Google Index (ցույց է տալիս այլ կայքերից կայքի վրա եղած հղումների քանակը ըստ Google-ի հաշվարկների) ,
- MSN Index (ցույց է տալիս այլ կայքերից կայքի վրա եղած հղումների քանակը ըստ MSN -ի հաշվարկների),
- Age (ցույց է տալիս այլ կայքի տարիքը, ստեղծման տարեթիվը ըստ http://web.archive.org –ի, վերջինս ունի գերհզոր սերվերներ և ժամանակ առ ժամանակ արխիվացնում է բոլոր կայքերը),
- Yahoo linkdomain (ցույց է տալիս այլ կայքերից տվյալ domain-ի վրա եղած հղումների քանակը ըստ Google-ի հաշվարկների),
- Yahoo link (ցույց է տալիս այլ կայքերից տվյալ domain-ի և subdomain-ների վրա եղած հղումների քանակը ըստ Google-ի հաշվարկների)
- External Links (ցույց է տալիս այլ կայքերի վրա կայքում պարունակվող հղումների քանակը)
- Internal Links (ցույց է տալիս կայքի domain-ին պատկանող էջերի վրա կայքում պարունակվող հղումների քանակը)

- Mon.itor.us- ի տվյալների բազայի հիման վրա ստացվում է հետևյալ ինֆորմացիան՝
- կայքի միջին հասանելությունը շաբաթվա ընթացքում (արտահայտված տոկոսով)
- կայքի արձագանքի միջին տևողությունը շաբաթվա ընթացքում (արտահայտված վայրկյաններով)
- կայքի անհասանելի լինելու տևողությունը շաբաթվա ընթացքում (արտահայտված վայրկյաններով)։

Google օրացույցը օգտակար է ինչպես կայքերի սեփականատերերի, ադմինիստրատիվ աշխատողների, այնպես էլ սովորական օգտագործողների համար, քանի որ այն տալիս է հիշեցման հնարավորություն ,ինչպես փոստի վրա ,այնպես էլ հեռախոսի masseng.-ների տեսքով։

Աշխատանքն իր մեջ ներառում է ինչպես javascripti , HTML, Css, AJAX, PHP ծրագրավորման լեզուների կիրառում

Աշխատանք իր մեջ ներառում է նաև խնդրի տնտեսագիտական շահավետության, նրա հետ աշխատող մարդկանց կենսական անվտանգության պայմանների հաշվարկ և օգտագործողի կենսական պաշտպանության եղանակներ տարբեր վնասակարության իրավիճակներից :

Գլուխ 1

1.1 Խնդրի պահանջները

Դիպլոմային աշխատանքի հիմնական պահանջն է Google Calendar-ի ինտեգրացիան Mon.itor.us կայքի վրա : Mon.itor.us -ը ինչպես արդեն գիտեք (նախորդ էջում) նախատեսված է տարբեր տեսկի կայքեր մոնիտորինգ անելու համար: Ունեինք MonitorUs կայք և Google API և պետք է ստեղծեինք Google –ի օրացույցի ինտեգրացիան MonitorUs –ին:

Google Calendar –ը իր մեջ ընդգրկում է այնպիսի ցուցանիշներ, ինչպիսիք են՝

- Events –ների ուղարկում (հիշեցման կարքով)
- Masseng.-ների ուղարկում
- Email-ի վրա նամակների ուղարկում
- Հրավրիատոմսերի ուղարկում
- Համատեղ օրացույցի օգտագործում
- Ժամանակի տնտեսում
- ԵՎ առհասարակ պրոֆեսոնալ օրացույցի տրամադրում

Այսինքն բացի նրանից որ այն կատարելագործված օրացույց է այն հնարավորություն է տալիս օգտագործողին հիշեցնելու տվյալ օրվա սերվեռի վիճակի մասին :Ինտեգրացված օրացույցը հնարավորություն կտա քեզ տեսնել քո Google օրացույցի գործողությունները առանց login լինելու:Այն նախատեսված է Mon.itor.us օգտագործողներին(user-ներին)հիշեցնելո իրենց անելիքը ,իր գործողությունների(events-ների)միջոցով:Օրինակ եթե դու պետք է ամսվա որևէ օրը ամպայաման ուշադրություն դարձնես քո սերվերի ծանրաբեռնվածությանը, ինչ-որ մի մոդուլով ամպայման տվյալ օրը մոնիտորինգ անել քո կայքը և նմանատիպ այլ հիշեցումներ :

Գլուխ 2

Php

Php –ի լեզվի կիրառությունը

Հիմնականում ծրագրերը նախատեսված են որոշակի խնդիրներ լուծելու համար:Տվյալների փոխանցումը HTML-ում ՝ ինչպես գիտենք կատարվում է ֆորմի միջոցով:Օրինակ

<form action ='' /cgi/delete.cgi'' metod='' post''>

<input type =''text'' name =''user-name''>

<input type=''submit'' name=''go'' value=''Go!''>

Եթե այս գրածը պահպանենք HTML-ի ֆորմի մեջ ,ապա կստամնանք կոճակ որը սեխմելու ընթացքում հնարավորություն կտա տվյալները սերվերին փոխանցել: Php ի սկզբանէ նախատեսված էր ինտերնետի հետ աշխատելու համար,ավելի հստակ սերվերի հետ աշղատելու համար :Որպեսզի ավելի հեշտ հասկանանք phр լեզվի ինքնությունը բերենք օրինակ՝

<?

Echo ''<html><body><h1>'';

Echo ''hello.$my_mame'';

Echo ''</h1></body></html>'';

?>

Մինչ այս ծրագրի աշխատացնելը անհրաժեշտ է ճիշտ տեղադրել այն սերվերի վրա: Աշխատացնելուց հետո էկրանին կարտատպվի հետևյալը՝

Heloo, Anna

Բրաուզերի localhostum գրելով հետևյալը՝http://localhost/hirst.php?my_name=Anna:Եվ եթե այն նայենք html-ի տեսքով ապա կտեսնենք հետևյալը՝

<html><body><h1>

Hello,Anna

</h1></body></html>

Ինչպես տեսանք php տեղադրեց մեզ հարկավոր անվանումը և արդեն html-ում ցուցց տվեց մեզ հարկավոր արժեքը: Հիմա կարող ենք նայել կոդին և փորցել հասկանալ թե ինչ է այն տեղ գրված :php –ին գրվում է հատուկ <? ?> թագի մեջ :echo օպերատորը իրականացնում է ինֆորմացիայի արտատպումը:Նրան կարելի է համարել ամենակարևոր օպերատորը ,քնի որ առանց նրա հնարավոր չէր իրականացնել ինֆորմացիայի արտատպում:Նկատեք նաև ,որ մեր օրինակում մենք մեր . $my_mame փոփոխականը չենք հայտարարաել :Աֆտոմատ հայտարարել է php-ին:

```
<?
Echo  "<html><body><h1>";
Echo  "hello.$my_mame";
Echo  "<br>$ var";
Echo  "</h1></body></html>";
?>
```

Ինչպես կարող ենք հայտարարել php –ում հայտարարել փոփոխական:Պարզապես գրում ենք այսպես՝

$ x=100

$ x սա իրենից ներկայացնում է փոփոխական ,իսկ 100 արժեքը

$ x =4

$ y= $ x

Առաջին արտահահայտությունում 4-ը համարվում է հաստատուն ,իսկ երկրորդ օրինակում քանի որ $ x -ին առաջին օրինակում վերագրվել է 4 արժեքը:Եվ սրանից հետևություն ,որ $ y նույնպես հավասար է 4-ի:Սա իր հերթին

հնարավորություն է տալիս օգտագործել հետևյալ գործողությունները $ x =$ y=5 կամ $ y =($ x= 5):

Javascript

Javascript կիրառությունը

Javascript: -ը համարվում է ծրագրավորման սկրիպտային լեզու :Որը իր մեջ ներառում է types , values, variables : Javascript –ըի օգնությամբ ինտերնետը դարձել է օգտագործողի համար ավելի հասանելի ու ավելի արագագործ:Այն օգտագործվում է վեբ-էջերի ստեղծման համար: Javascript-ը ստեղծվում է HTML լեզվի մարմնում ,բայց համարվելով դինամիկ լեզու:

```
<html>
  <head><title>simple page</title></head>
  <body>
    <script type="text/javascript">
     document.write('Hello World!');
    </script>
    <noscript>
      <p>Your browser either does not support JavaScript, or you
have JavaScript turned off.</p>
    </noscript>
  </body>
</html>
```

Javascript օգտագործվում է բազմաթիվ վեբ-էջերում և աշխատում է բազմաթիվ բրաուզերներներում ,ինչպիսիք են Internet Explorer, Mozilla, Firefox, Netscape и Opera. Javascript-ը ստեղծվել է HTML էջերը ավելի կատարելագործելու համար: Javascript-ը համարվում է ծրագրավորման սկրիպտային լեզու :Որպես կանոն այն տեղադրվում է HTML –ի մարմնում:Ամենոք կարող է օգտագործել Javascript-ը առանց լիցենդիոն իրավունքի:javascript –ը պետք չէ շփոթել java-ի հետ,այն միանգամայն այլ լեզու: java

–ան ավելի դժվար և ավելի ուժեղ լեզու է համարվում:Այն կարելի է նաև օգտագործել ինֆորմացիայի փնտրման և պահպանման համար:

Javascript-ում փոփոխականը կարող է ունենալ կարճ անվանում `օրինակ x:Ինչպես հանրահաշվում այնպես Javascript-ում կարող եք կատարել հանրահաշվական գործողություններ`

```
y =x-5
d=y+5
<script type="text/javascript">
document.write ( "<h1> վերնագիր </ h1>");
document.write ( "<p> առաջին</ p>");
документ. писать ( "<p> երկրորդ </ p>");
</ скрипт>
```

AJAX

AJAX կիրարությունը

AJAX –ը հանդիսանում է web ծրագրավորման տեխնոլոգիա, որն օգտագործվում է ինտերակտիվ(փոխներգործող) web ծրագրերի(Aplication) ստեղծման համար: AJAX –ի նպատակն է server –ի հետ տվյալների փոքր փաթեթնփր փոխանակելու միջոցով web էջերը դարձնել ավելի արագ արձագանքող, այնպես որ կարիք չի լինում սկզբնական web էջը reload անել ամեն անգամ, երբ օգտագործողը(user) փոփոխություն է պահանջում: Սա հնարավորություն է ընձեռում մեծացնել web էջերի արագագործությունը, ֆունկցիոնալությունը և հարմարավետությունը(օգտագործելիությունը):

AJAX –ն ասինխրոն է նրանով, որ եթե server –ից պահանջվում են լրացուցիչ տվյալներ, դրանք ստացվում ցուցադրվում են առանց էջը վերաբեռնելու(reload) և տվյալների ստացվելու ընթացքում բացված էջում կարող է ոչինչ չփոխվել: JavaScript –ն այն սկրիպտային լեզուն է, որտեղ սովորաբար կանչվում են AJAX ֆունկցիաները: Server –ից տվյալները ստացվում են XMLHttpRequest օբյեկտի միջոցով, որը հասանելի է ժամանակակից browser -ներում օգտագործվող սկրիպտային լեզուներում (հիմնականում JavaScript): Նախնտրելի է, որ server –ից փոխանցվող տվյաների ֆորմատը լինի XML:

AJAX –ը միջպլատֆորմային տեխնիկա է, որը հասանելի է տարբեր ՕՀ –ներում, համակարգիչներում և web browser –ներում, քանի որ այն հիմնված է բաց ստանդարտների վրա ինչպիսիք են JavaScript –ը և DOM(Document Object Model) –ը:

AJAX –ն օգտագործում է.

- XHTML(կամ HTML) և CSS ինֆորմացիան ցուցադրելու և ձևավորելու համար
- DOM –ը client-side սկրիպտային լեզուներով, մասնավորապես ECMAScript –ի իմպլեմենտացիաներով, ինչպիսիք են JavaScript –ն ու Jscript –ը, ինֆորմացիան դինամիկորեն ցուցադրելու և ներկայացվող ինֆորմացիայի հետ փոխներգործելու համար
- XMLHttpRequest օբյեկտը server –ի հետ տվյալներ փոխանակելու համար:
- XML –ը, որպես server –ի և կլիենտի միջև փոխանակվող տվյալների ֆորմատ, չնայած ընդունելի են նաև HTML ֆոռմատը, հասարակ տեքստը և JSON –ը: Այս ֆայլերը կարող են գեներացվել դինամիկորեն server –ի կողմից:

AJAX –ի հիմնական էլեմենտը XMLHttpRequest օբյեկտն է, որը հնարավորություն է տալիս browser –ներին կատարել տվյալների դինամիկ և ասինխրոն հարցում առանց էջը անբեռնելու(unload) և վերաբեռնելու(relouad): Ընդունված է, որ AJAX –ը կարող է վերացնել էջի վերաբեռնման(reload) անհրաժեշտությունը:

AJAX -ով ծրագրավորման հիմնական նպատակը HTML/HTTP –ի վրա հիմնված web կայքերում էջի վերաբեռնման(reload) անհրաժեշտությունից խուսափելն է: AJAX –ն անհրաժեշտ նախապայման է ստեղծում web էջերում բարդ, զգայուն, դինամիկ, տվյալների վրա կենտրոնացված UI –ներ ստեղծելու համար: Web էջերն, ի տարբերություն տեղային(native) application -ների, թույլ կապնված են, ինչը նշանակում է, որ նրանցում ցուցադրվող ինֆորմացիան խստորեն կապված չէ տվյալների ակունքի(data source) հետ և պետք է դասավորվեն համապատասխան կարգով HTML էջի ֆորմատում կլիենտ մեքենայի user agent(web browser) –ին հանձնվելուց առաջ: Այս պատճառով web էջերն անհրաժեշտ է լինում վերաբեռնել(reload) ամեն անգամ, երբ օգտագործողը ցանկանում է նայել տվյալների տարբեր հավաքածուներ: Առանց էջը վերաբեռնելու(reload) տվյալների հարցում կատարելու և դրանք

ստանալու համար XMLHttpRequest օբյեկտն օգտագործելով ՝ հնարավոր է դառնում շրջանցել այս անհրաժեշտությունը և ստիպել թույլ կապնված(loosely coupled) էջերին իրենց դրսեվորել որպես ուժեղ կապնված (tightly coupled) application, բայց հեռվում գտնվող server –ին փոխանցվող տվյալների պատճառով փոփոխականների ավելի մեծ ուշացման ժամանակով:

AJAX –ի առավելությունները:

օգտագործման մեծ հնարավորություններ: Լոկալ browser –ում գեներացնելով HTML էջերը և նվազեցնելով JavaScript կանչերն ու ակտուալ տվյալները AJAX էջերը կարող են բեռնվել(load) ավելի արագ, քանի որ կանչվող ինֆորմացիայի չափն ավելի փոքր է և յուրաքանչյուր փոփոխության ժամանակ էջն ամբողջությամբ վերանկարելու կարիք չկա: Այս տեխնոլոգիայի օրինակ է արդյունքների մեծ հավաքածուն(large result set), որտեղ առկա են տվյալների մի քանի էջեր: AJAX –ի օգնությամբ, էջի HTML –ը կարող է գեներացվել լոկալ(browser -ում), այլ ոչ թե բեռնվել փաստաթղթի առաջին էջի հետ: Որպես ավելացում պարունակության պահանջով բեռնավորմանը(load on demand) որոշ web application –ներ բեռնում են հիմնական event-handler –ները, իսկ ֆունկցիաները բեռնվում են հետո ՝ աշխատանքի ընթացքում: Այս տեխնիկան զգալիորեն կրճատում է web application –ներում ռեսուրսների մեծ ծախսերը:

AJAX մոտեցման ավելի քիչ ցպեցիֆիկ առավելություններից է այն, որ AJAX –ը խրախուսում է web –ի միջոցով ինֆորմացիա ստանալու տարբեր ասպեկտների համար օգտագործվող մեթոդների ու ֆորմատների մաքուր տարանջատումը: Չնայած AJAX –ը կարող է լեզուների և տեխնոլոգիաների անկանոն խառնուրդ թվալ և ծրագարավորողները կարող են որդեգրել և հարմարեցնել այն ամենը ինչ ցանկանան, նրանք, հիմնականում ուղղորդվելով հենց ծրագրավորման պրոցեսով, պետք է տարբերություն դնեն հետևյալի միջև.

1. ստացվող չմշակված տվյալները , որոնք որպես կանոն ներառվում են XML –ում կամ ստացվում են server-ում Տվյալների Բազայից:
2. web էջի ֆորմատը կամ կառուցվածքը, որը ստացվում էHTML –ի կամ XHTML –ի միջոցով, որտեղ հետագայում հնարավոր են DOM –ով դինամիկ մանիպուլացիաներ:

3. web էջի տեսքի էլեմենտները: Տեսքի հետ կապված ամեն ինչ սկսած ֆոնտերից մինչև նկարների հասցեները ստացվում են CSS-ի միջոցով:

4. web էջի ֆունկցիոնալությունը, որն ապահովվում է

- ✓ կլիենտ browser –ի JavaScript -ի(DHTML)
- ✓ ստանդարտ HTTP և XMLHttp կամ կլիենտ – սերվեր կապ
- ✓ server-side սկրիպտներ և / կամ ծրագրավորողի նախընտրած ցանկացած լեզվով ծրագրեր, որոնք հարմարեցված են կլիենտի կողմից եկած հարցումները ստանալուն և դրանց պատասխանելուն:

AJAX –ի թերությունները:

Browser ինտեգրացիա:_ Դինամիկ ստեղծված web էջերը չեն գրանցվում browser –ի պատմության գործիքում(history engine) և օգտագործողի browser –ի back(ետ) ֆունկցիայի կանչը կարող է նախորդ արդյունքը չվերադարձնել: Այս խնդրի համար տարբեր լուծումներ են մտածված: Դրանցից է անտեսանելի Iframe –ների օգտագործումը browser –ի back(ետ) կոճակի կողմից օգտագործվող պատմությունում պարունակվող փոփոխություններն արթնացնելու համար: Օրինակ կառնելի է փոփոխությունները պահել անտեսանելի Iframe –ում և անհրաժեշտության դեպքում դրանք ուղարկել հետ ՝ web էջի տեսանելի էլեմենտին:

Մյուս խնդիրը կայանում է նրանում, որ դինամիկ web էջերում կատարվող փոփոխությունները օգտագործողի համար բարդացնում են application –ի որևէ վիճակի նշումը(bookmark): այս խնդրի լուծումներից է URL –ի ֆրագմենտ իդենտիֆիկատորի օգտագործումը(օրինակ URL –ից հետո '#' –ի ավելացումը) application –ի վիճակին հետևելու և օգտագործողին, տրված վիճակին վերադառնալու հնարավորություն ընձեռնելու համար: Սա հնարավոր է, քանի որ շատ browser –ներ JavaScript –ին URL –ի ֆրագմենտ իդենտիֆիկատորը դինամիկ փոփոխելու հնարավորություն են տալիս:

Պատասխանի ժամանակի սահմանափակում: AJAX ծրագրավորման ժամանակ օգտագործողի հարցման և server –ի պատասխանի միջև եղած ժամանակային ինտերվալին պետք է վերաբերվել զգուշությամբ: Տվյալների ճկուն վերաբերումը և XMLHttpRequest օբյեկտի անհրաժեշտ սպասումները կարող են web application –ում երբեմն դանդաղեցումներ առաջ բերել, որոնք օգտագործողի կողմից կարող են վատ ընդունվել: Բացի այդ էջի պարունակության փոփոխության ընթացքում աչքի համար տեսանելի են տեսքի միջանկյալ անկանոն վիճակները: Այս պրոբլեմից

խուսափելու համար կարելի է օգտագործել հատուկ էլեմենտներ էջի ետին պլանում ընթացող գործողությունների մասին օգտագործողին տեղյակ պահելու համար:

Կախված է JavaScript –ից:_ AXAX –ը հիմնված է JavaScript –վրա, որը տարբեր browser –ների կողմից իմպլիմենտացվում է տարբեր կերպ: Այս պատճառով JavaScript օգտագործող կայքերը պետք է տեստավորվեն տարբեր browser –ներով նրանց հետ համատեղելիությունը ստուգելու համար:

ինչպես է AJAX –ն աշխատում:

AJAX –ի աշխատանքը կապված է օգտագործողի գործողությունների արդյունքում առաջացող դեպքերի(event) հետ: DOM –ը այդ գործողությունները էջի էլեմենտների հետ կապելու հնարավորություն է տալիս: Server –ից տվյալներ ստանալու համար XMLHttpRequest օբյեկտը տրամադրում է 2 մեթոդ.

- ✓ Open: կապ է հաստատում server –ի հետ
- ✓ Send: Server –ին հարցում է ուղարկում
- ✓ Server –ից ուղարկվող տվյալները կարելի է ստանալ XMLHttpRequest օբյեկտի հետևյալ ատրիբուտների միջոցով.
- ✓ responseXml: XML ֆիլի համար
- ✓ responseText : տեքստի համար:
- ✓ Հարցումից հետո AJAX –ը սպասում է մինչև հնարավոր լինի ստանալ և մշակել server –ի պատասխանը: Հարցման կատարման ընթացքի մասին կարելի է տեղեկանալ XMLHttpRequest օբյեկտի readyState ատրիբուտի միջոցով, որի հնարավոր արժեքներն են.
- ✓ 0 - դեռ չի ինիցալիզացվել
- ✓ 1 - կապը հաստատված է
- ✓ 2 - հարցումն ընդունված է
- ✓ 3 - պատասխանը մշակվում է
- ✓ 4 - գործողություն ավարտված է

Եթե <EM>-ի համար ոչ մե գույն նախատեսցված չէ ապա նա կժառանգի <H1>-ի գույնը:Այսինքն նա նույնպես կարտատպվի կապույտ գույնով :Հաջորդ օրինակում կտեսնեք թէ ինչ հարմար ձևով է կապում իրար Աղյուսակի տեսքէ նկարների հետ՝

```
<STYLE TYPE="text/css">
      UL {
        background: red;
        margin: A B C D;
        padding: E F G H;
      }
      LI {
        color: white;
        background: blue;
        margin: a b c d;
        padding: e f g h;
      }
    </STYLE>
    ..
    <UL>
      <LI>1й элемент списка
      <LI>2й элемент списка
    </UL>
```

Գույնի որոշումը /color/

Այս հատկությունը որոշում է Css-ում տեքստի գույնը:Կա մի քանի եղանակներ Css-ում տեքստը գունավորելու համար՝

```
EM { color: red }
EM { color: rgb(255,0,0) }
```

Background-ը ապահովում է նկարի ֆոնը: background-color-ի նշանակությանը և օգտագործման օրինակ՝

```
H1 { background-color: #F00 }
```

```
BODY { background: url(banner.jpeg) right top }
BODY { background: url(banner.jpeg) top center }
BODY { background: url(banner.jpeg) center }
BODY { background: url(banner.jpeg) bottom }
```

Css-ը կարող է ազդել ոչ միայն մի աղյուսկի տեսքի վրա այլ միանգամից միքանի աղյուսակների:

HTML

HTML փաստաթղթի կառուցվածքը

HTML ստանդարտը միանշանակ որոշում է Web փաստաթղթի հիմնական կառուցվածքը: Web էջը, որպես կանոն, սկսվում է <html> - ով և վերջանում </html> - ով: Փաստաթղթի բաղադրիչ մասերն են վերնագիրը և գլխավոր մասը (մարմինը): Որպես այլընտրանք կարող է առաջարկվել փաստաթուղթը կառուցել շրջանակների (Frames) միջոցով: HTML փաստաթղթի նշանակումը կատարվում է <html> և </html> նշագրմամբ: Մնացած բոլոր HTML հրամաններն ու Web էջի պարունակությունը պատկերող ամեն ինչ պետք է տեղաբաշխված լինի տվյալ վերծանիչների միջև (մի քանի բացառություններ կան, որոնք նկարագրված են ստորև): Այսպիսով, HTML փաստաթուղթը բաժանվում է երկու տիրույթի՝ վերնագիր (head) և մարմին (body):

Վերնագրի մեջ որոշվում են կարևորագույն դրույթները, օրինակ՝ փաստաթղթի անունը, URL-հասցեն, ինչպես նաև օգտագործվող տառատեսակը: HTML փաստաթղթի գլխավոր մասը ներառում է բրոուզերով էկրանին բերվող Web էջի պարունակությունը:

Ձևը՝ դա մի գործիք է, որի օգնությամբ HTML փաստաթուղթը կարող է որոշ տեկեկություններ ուղարկել արտաքին աշխարհի նախապես որոշված որևէ մի կետ, որտեղ այդ տեղեկություններն ինչ-որ ձևով կմշակվեն: Տվյալները մշակող ծրա•րերը հաճախ անվանվում են CGI - սկրիպտներ (CGI - Common
Gateways Interface, այսինքն՝ անցախցերի ընդունված ինտերֆեյս): CGI - սկրիպտներ գրելու համար պահանջվում է համապատասխան ծրագրավորման լեզվի (օրինակ՝ PHP/FI) և UNIX օպերացիոն համակարգի լավ իմացություն:

Ձևերը հաղորդում են տեղեկությունները մշակող ծրագրերին [փոփոխականի անուն]=[փոփոխականի արժեք] զույգերի տեսքով: Փոփոխականների անունները պետք է առաջադրվեն միայն լատիներեն տառերով: Ձևը բացվում է <FORM> տեգով և ավարտվում </FORM>-ով: HTML փաստաթուղթը կարող է պարունակել մի քանի ձևեր, բայց դրանք չեն կարող իրար մեջ գտնվել: HTML տեքստը կարող է

տեղաբաշխվել ձևի մեջ՝ առանց սահմանափակման: <FORM> տե•ը կարող է ունենալ երեք ատրիբուտ, որոնցից մեկը պարտադիր է: Ահա դրանք. ACTION

Պարտադիր ատրիբուտ է: Այն որոշում է, թե որտեղ է գտնվում ձևի մշակիչը:

METHOD

Որոշում է, թե ի՞նչ ձևով (այլ կերպ ասած՝ հիպերտեքստի հաղորդման արձանագրության ո՞ր մեթոդի օգնությամբ) տվյալները ձևից պետք է հաղորդվեն մշակիչին: Թույլատրելի արժեքներն են.

METHOD=POST և METHOD=GET:

Եթե ատրիբուտի արժեքը սահմանված չէ, լռելյայն ընդունվում է METHOD=GET.

ENCTYPE

Սահմանում է տվյալների կոդավորման կարը՝ մշակիչին հաղորդելու համար: Եթե ատրիբուտի արժեքը սահմանված չէ, լռելյայն ընդունվում է

ENCTYPE=application/x-www-form-urlencoded:

HTMLը հնարավորություն է տալիս աշխատելու չհամարակալված, համարակալված ցուցակների և ցուցակսահմանումների հետ:

. Չհամարակալված ցուցակներ

Չհամարակալված ցուցակ պատրաստելու համար.

1. Սկսեք <UL> տեգից,
2. Հավաքեք <LI> տեգը և ցուցակի տարրը (</LI> փակող տեգը պետք չէ),
3. Հավաքեք </UL> փակող տեգը:

Ստորև բերված է երկու տողից կազմված պարզագույն մի օրինակ.

```
<UL>
<LI> խնձոր
<LI> տանձ
 <LI> բալ
</UL>
```

Արդյունքը կունենա հետևյալ տեսքը.

- խնձոր
- տանձ
- բալ

Ցուցակի տարրերը կարող են պարունակել մի քանի պարբերություններ: Դրանք նշելու համար օգտագործվում է <P> տեգը:

. Համարակալված ցուցակներ

Համարակալված ցուցակներ կազմելու համար, ի տարբերություն չհամարակալված ցուցակների, պետք է օգտագործել <OL> տեգը <UL>ի փոխարեն:

Ցուցակի տարրերը նշվում են նույն <LI> տեգի միջոցով: Հետևյալ HTML կոդը

```
<OL>
<LI> խնձոր
<LI> տանձ
<LI> բալ
</OL>
```

կտա այսպիսի արդյունք.

1.խնձոր
2.տանձ
3.բալ

Ցուցակսահմանումներ

Ցուցակսահմանումը սովորաբար բաղկացած է տերմինից (դրա DT հապավումից) և տերմինի սահմանումից (հապավումը՝ DD): Web դիտարկիչները սովորաբար սկսում են սահմանումները նոր տողից:

Ահավասիկ մի օրինակ՝

```
<DL>
<DT> Հաճախորդ
<DD> Ծրագրային միջոցների և սարքավորումների համադրություն, որն օգտագործողի կողմից մասնակցում է սերվերի հետ տեղեկատվության փոխանակմանը:
<DT>Հատկացված կապուղի
<DD> Մշտական միացում, որը թույլ է տալիս հաղորդել տեղեկատվության հոսքը մի համակարգչից մյուսին:
</DL>
```

որի արդյունքը հետևյալն է.

Հաճախորդ

Ծրագրային միջոցների և սարքավորումների համադրություն, որն օգտագործողի կողմից մասնակցում է սերվերի հետ տեղեկատվության փոխանակմանը:

Հատկացված կապուղի

Մշտական միացում, որը թույլ է տալիս հաղորդել տեղեկատվության հոսքը մի համակարգչից մյուսին:

<DT> և <DD> տարրերը կարող են պարունակել մի քանի պարբերություն, ցուցակներ կամ այլ տեղեկատվություն:

. Ներդրված ցուցակներ

Ցուցակները կարող են լինել ներդրված (գործնականում հավանական է, որ հարկ լինի սահմանափակվել ներդրման երեք մակարդակներով):

Դուք կարող եք նաև օգտագործել մի քանի պարբերություններ, որոնցից յուրաքանչյուրը պարունակի ներդրված ցուցակ (ցուցակի մի տարրի մեջ):

Ներդրված ցուցակի օրինակ.

```
<UL>
  <LI> Հայաստանի հյուսիսային մարզերից են.
    <UL>
    <LI> Շիրակը
    <LI> Լոռին
    <LI> Տավուշը
    </UL>
  <LI> Հարավային մարզը՝
    <UL>
    <LI>Սյունիքը
    </UL>
  </UL>
```

Այս ներդրված ցուցակն արտապատկերվում է հետևյալ կերպ.

Հայաստանի հյուսիսային մարզերից են.

Շիրակը

Լոռին

Տավուշը

Հարավային մարզը՝

Սյունիքը

Աղյուսակներ

Մինչ այժմ մենք գործ ունեինք այնպիսի փաստաթղթերի հետ, որոնց մեջ կար տեքստի միայն մեկ հոսք: ործնականում երբեմն հարկ է լինում տեքստը տեղաբաշխել մի քանի սյունակներով: Դրանում աղյուսակը կարող է օգնել:

Բացի այդ, մի բջջից կազմված աղյուսակում տեքստը կարող է շատ ավելի տպավորիչ լինել, գրավելով ընթերցողի ուշադրությունը:

Աղյուսակների ներկայացման համար օգտագործում են մի քանի տեսակի տեգեր: Դրանք են.

TABLE, որոնք իրենց մեջ են ներառում ողջ աղյուսակի նկարագրությունը;

CAPTION ոչ պարտադիր տեգերը, որոնք բնորոշում են աղյուսակի գլխագիրը (անունը);

TR տեգերը, որոնք բնորոշում են աղյուսակի տողերը;

TH տեգերը, որոնք բնորոշում են տողերի և սյունակների վերնագրերը;

TD տեգերը, որոնք բնորոշում են աղյուսակի տվյալները, այսինքն՝ աղյուսակային բջիջների պարունակությունը:

Դիտարկենք մի օրինակ.

```
<HTML>
<HEAD>
<TITLE>Օրինակ</TITLE>
</HEAD>
<BODY>
<H1>Պարզագույն աղյուսակ</H1>
<TABLE BORDER=1> <!--Սա աղյուսակի սկիզբն է-->
<CAPTION>       <!--Սա աղյուսակի գլխագիրն է-->
Աղյուսակը կարող է ունենալ գլխագիր
</CAPTION>
```

```
<TR>          <!--Սա առաջին տողի սկիզբն է-->
<TD>          <!--Սա առաջին բջջի սկիզբն է-->
Առաջին տող, առաջին սյունակ
</TD>         <!--Սա առաջին բջջի վերջն է-->
<TD>          <!--Սա երկրորդ բջջի սկիզբն է-->
Առաջին տող, երկրորդ սյունակ
</TD>         <!--Սա երկրորդ բջջի վերջն է-->
</TR>         <!--Սա առաջին տողի վերջն է-->
<TR>          <!--Սա երկրորդ տողի սկիզբն է-->
<TD>          <!--Սա առաջին բջջի սկիզբն է-->
Երկրորդ տող, առաջին սյունակ
</TD>         <!--Սա առաջին բջջի վերջն է-->
<TD>          <!--Սա երկրորդ բջջի սկիզբն է-->
Երկրորդ տող, երկրորդ սյունակ
</TD>        <!--Սա երկրորդ բջջի վերջն է-->
</TR>         <!--Սա երկրորդ տողի վերջն է-->
</TABLE>      <!--Սա աղյուսակի վերջն է-->
</BODY>
</HTML>
```

Աղյուսակն սկսվում է <TABLE> տեգով և վերջանում </TABLE> տեգով:
<TABLE> տեգը կարող է ունենալ մի քանի ատրիբուտներ.

ALIGN
Սահմանում է աղյուսակի դիրքը փաստաթղթի դաշտերի նկատմամբ: Թույլատրելի արժեքներն են.
ALIGN=LEFT (հավասարեցում ձախից),
ALIGN=CENTER (հավասարեցում կենտրոնով),
ALIGN=RIGHT (հավասարեցում աջից):

WIDTH
Աղյուսակի լայնքը: Այն կարելի է առաջադրել պիկսելներով (օրինակ՝ WIDTH=400) կամ էջի լայնքի տոկոսներով (օրինակ՝ WIDTH=80%):
BORDER Առաջադրում է աղյուսակի արտաքին շրջանակի և բջիջների լայնքը

պիկսելներով (օրինակ՝ BORDER=3): Եթե ատրիբուտը չի նշված, աղյուսակը կերևա առանց շրջանակի:

CELLSPACING

Սահմանում է աղյուսակի բջիջների միջև եղած հեռավորությունը պիկսելներով (օրինակ՝ CELLSPACING=2):

CELLPADDING

Սահմանում է բջջի շրջանակի և տեքստի միջև եղած հեռավորությունը պիկսելներով (օրինակ՝ CELLPADDING=10):

լխագրի <CAPTION> տեգը կարող է ունենալ ALIGN ատրիբուտը, որի հնարավոր արժեքներն են.

<CAPTION ALIGN=TOP> (վերնագիրը տեղադրվում է աղյուսակի վերևում) և

<CAPTION ALIGN=BOTTOM> (վերնագիրը տեղադրվում է աղյուսակի տակ):

Աղյուսակի յուրաքանչյուր տող սկսվում է <TR> տեգով և ավարտվում </TR> տեգով: <TR> տեգը կարող է ունենալ հետևյալ ատրիբուտները.

ALIGN

Սահմանում է տեքստի հավասարեցում տողի բջիջներում: Հնարավոր արժեքներն են.

ALIGN=LEFT (հավասարեցում ձախից)

ALIGN=CENTER (հավասարեցում կենտրոնով)

ALIGN=RIGHT (հավասարեցում աջից):

VALIGN

Սահմանում է տողի բջիջներում տեքստի ուղղաձիգ հավասարեցումը: Թույլատրելի արժեքներն են.

VALIGN=TOP (հավասարեցում վերին եզրով),

VALIGN=MIDDLE (հավասարեցում կենտրոնով),

VALIGN=BOTTOM (հավասարեցում ստորին եզրով):

Աղյուսակի յուրաքանչյուր բջիջ սկսվում է <TD> տեգով և ավարտվում </TD> տեգով: <TD> տեգը կարող է ունենալ հետևյալ ատրիբուտները.

NOWRAP

Այս ատրիբուտի ներկայությունը նշանակում է, որ բջջի պարունակությունը պետք

է ցույց տրվի մեկ տողով:

COLSPAN

Սահմանում է բջջի «թռիչքը» հորիզոնականով: Օրինակ՝ COLSPAN=2 բջիջը տարածվում է երկու սյունակների վրա:

ROWSPAN

Սահմանում է բջջի «թռիչքը» ուղղաձիգով: Օրինակ՝ ROWSPAN=3 նշանակում է, որ բջիջը գրավում է երեք տող:

ALIGN

Սահմանում է տեքստի հավասարեցումը բջջի մեջ: Թույլատրելի արժեքներն են. ALIGN=LEFT (հավասարեցում ձախով), ALIGN=CENTER (հավասարեցում կենտրոնով), ALIGN=RIGHT (հավասարեցում աջով):

VALIGN

Սահմանում է տեքստի ուղղաձիգ հավասարեցումը բջջի մեջ: Թույլատրելի արժեքներն են.

VALIGN=TOP (հավասարեցում վերին եզրով), VALIGN=MIDDLE (հավասարեցում միջնամասով), VALIGN=BOTTOM (հավասարեցում ստորին եզրով).

WIDTH

Սահմանում է բջջի լայնքը պիկսելներով (օրինակ՝ WIDTH=200):

HEIGHT

Սահմանում է բջջի բարձրությունը պիկսելներով (օրինակ՝ HEIGHT=40):

Եթե աղյուսակի բջիջը դատարկ է, դրա շուրջը շրջանակ չի նկարվում: Եթե բջիջը դատարկ է, իսկ շրջանակը պետք է, ապա բջիջը կարող է ներմուծել   հաջորդականությամբ (non-breaking space - չկտրվող բացատ): Բջիջն էլի դատարկ կլինի, իսկ շրջանակը դրա շուրջ կլինի:

Հետաքրքիր է նշել, որ աղյուսակի ցանկացած բջիջ կարող է իր մեջ պարունակել մի այլ աղյուսակ:

<u>Պատկերներ</u>

Դիտարկիչների մեծ մասը կարող են արտապատկերել տեքստում ներդրված, .XBM կամ .GIF ձևաչափով պատկերներ:

Յուրաքանչյուր պատկեր ժամանակ է պահանջում մշակման համար և դանդաղեցնում է փաստաթղթի բեռնումը, ուստի չարժե փաստաթղթում ներառել մեծ

քանակությամբ պատկերներ կամ մեծ չափի պատկեր:

Պատկեր ներմուծելու համար օգտագործեք այսպիսի գրառում.

<IMG SRC=image_URL>

որտեղ image_URLը նկարով ֆայլի URLն է: IMG SRC URLի շարահյուսությունը նույնական է HREF-ի մեջ օգտագործվողին: GIF ֆայլերը պետք է ունենան .gif ընդլայնում, X Bitmap ֆայլերը՝ .xbm:

Լռելյայն տեքստն արտաբերվում է պատկերի ստորին մասին կից:

Ավելացրեք ALIGN=TOP, եթե դուք ուզում եք հավասարեցնել տեքստը պատկերի վերին եզրով: Այդ դեպքում պատկերի ներդրման լրիվ տեգը կլինի.

<IMG ALIGN=top SRC=image_URL>

ALIGN=MIDDLE արտաբերում է տեքստը կենտրոնով:

Պարզագույն ձև

Ձևից մշակիչին տվյալների հաղորդման •ործընթացն սկսելու համար անհրաժեշտ է ինչ-որ ղեկավարման մարմին: Այդպիսի մարմին ստեղծելը շատ հեշտ է.

<INPUT TYPE=submit>

Ձևի մեջ հանդիպելով այսպիսի տողի, բրոուզերն էկրանին կնկարի Submit գրությամբ մի սեղմակ, որի վրա սեղմելիս ձևի մեջ եղած բոլոր տվյալները կհաղորդվեն <FORM> տեգի մեջ նշված մշակչին: Սեղմակի վրայի գրությունն առաջադրվում է VALUE ատրիբուտի միջոցով: Օրինակ՝

<INPUT TYPE=submit VALUE="Top!">

Հիմա մենք կարող ենք գրել պարզագույն մի ձև, որը ոչ մի տվյալ չի հավաքի, այլ պարզապես մեզ կվերադարձնի այս բաժնի սկզբին:

```
<HTML>
<HEAD>
<TITLE>Օրինակ</TITLE>
</HEAD>
<BODY>
<H1>Պարզա•ույն ձև</H1>
<FORM ACTION="exemple.htm"> <!--Սա ձևի սկիզբն է-->
<INPUT TYPE=submit VALUE="Top">
```

</FORM> <!--Սա ձևի վերջն է-->

</BODY>

</HTML>

Կան <INPUT> տարրի այլ տեսակներ ևս: Յուրաքանչյուր <INPUT> տարր պետք է ներառի NAME=[անուն] ատրիբուտը, որը որոշում է տարրի անունը և, համապատասխանաբար, փոփոխականի անունը, որը պետք է հաղորդվի մշակչին: Անունը պետք է առաջադրվի միայն լատիներեն տառերով:

<INPUT> տարրի հիմնական տեսակներն են.

TYPE=text

Որոշում է տաքստի տողի մուտքի պատուհանը: Կարող է պարունակել լրացուցիչ ատրիբուտներ. SIZE=[թիվ] (պատուհանի լայնքը նիշերով) և MAXLENGTH=[թիվ] (տողի առավելա•ույն թույլատրելի երկարությունը նիշերով):

Օրինակ.

<INPUT TYPE=text SIZE=20 NAME=user VALUE="Պողոս">

Տեքստի մուտքի համար որոշում է 20 նիշ լայնությամբ պատուհան: Լռելյայն պատուհանում գտնվում է «Պողոս» տեքստը, որնօգտագործողը կարող է խմբագրել: Խմբագրված (կամ չխմբագրված) տեքստը հաղորդվում է մշակչին user փոփոխականի մեջ:

TYPE=password

Որոշում է պայմանաբառի մուտքի պատուհանը: Լրիվ համանման է նախորդ տեսակին, միայն նիշերի փոխարեն ցույց է տալիս աստղանիշեր:

Օրինակ՝

<INPUT TYPE=password NAME=pw SIZE=20 MAXLENGTH=10>

Որոշում է 20 նիշ լայնությամբ պատուհան՝ պայմանաբառի մուտքի համար: Պայմանաբառի առավելա•ույն թույլատրելի երկարությունը՝ 10 նիշ: Ներմուծված պայմանաբառը հաղորդվում է մշակչին pw փոփոխականի մեջ:

TYPE=checkbox

Որոշում է մի քառակուսի, որի մեջ կարելի է նշում կատարել: Կարող է պարունակել checked լրացուցիչ ատրիբուտը (ցույց է տալիս, որ քառակուսին նշված է):

<INPUT> տարրից բացի ձևերը կարող են պարունակել <SELECT> աշխատակարգը և <TEXTAREA> դաշտեր՝ տեքստի ներմուծման համար:

n տարրերից բաղկացած <SELECT> աշխատակար•ը կունենա այսպիսի տեսք՝

<SELECT NAME="[անուն]">

<OPTION VALUE="[1 արժեք]">[տեքստ 1]

<OPTION VALUE="[2 արժեք]">[տեքստ 2]

...

<OPTION VALUE="[n արժեք]">[տեքստ n]

</SELECT>

Ինչպես տեսնում եք, աշխատակար•ն սկսվում է <SELECT> և ավարտվում </SELECT> տե•ով: <SELECT> տե•ը պարունակում է NAME պարտադիր ատրիբուտը, որը որոշում է աշխատակարգով առաջադրվող փոփոխականի անունը:

<SELECT> տեգը կարող է պարունակել նաև MULTIPLE ատրիբուտը, որի ներկայությունը ցույց է տալիս, որ աշխատակարգից կարելի է ընտրել մի քանի տարրեր: Բրոուզերների մեծ մասը ցույց է տալիս <SELECT MULTIPLE> աշխատակարգը մի պատուհանի տեսքով, որի մեջ •տնվում են աշխատակարգի տարրերը (պատուհանի բարձրությունը տողերով կարելի է առաջադրել SIZE=[թիվ] ատրիբուտի միջոցով): Շատ դեպքերում <SELECT> աշխատակարգը ցուցադրվում է ընկնող աշխատակարգի տեսքով: <OPTION> նշումը որոշում է աշխատակարգի տարրը: VALUE պարտադիր ատրիբուտը սահմանում է այն տարրը, որը պետք է հաղորդվի մշակչին, եթե ընտրված է աշխատակարգի այդ տարրը: <OPTION> նշումը կարող է ներառել checked ատրիբուտը, որը ցույց է տալիս, թե տվյալ տարրը նշված է լռելյայն:

Օրինակ.

<SELECT NAME="selection">

<OPTION VALUE="option1" checked>Տարբերակ 1

<OPTION VALUE="option2"> Տարբերակ 2

<OPTION VALUE="option3"> Տարբերակ 3

</SELECT>

Այսպիսի հատվածը տալիս է երեք տարրերով աշխատակարգ. Տարբերակ 1, Տարբերակ 2 և Տարբերակ 3: Լռելյայն ընտրված է Տարբերակ 1-ը: Մշակչին կհաղորդվի selection փոփոխականը, որի արժեքը կարող է լինել option1 (լռելյայն), option2 կամ option3:

Այս բոլորից հետո <TEXTAREA> տարրը կարող է բոլորովին պարզ երևալ: Օրինակ.

<TEXTAREA NAME=address ROWS=5 COLS=50>

Իսկ այստեղ Ձեր հասցեն է ...

</TEXTAREA>

Բոլոր ատրիբուտները պարտադիր են: NAME ատրիբուտը որոշում է այն անունը, որի տակ պատուհանի պարունակությունը պետք է փոխանցվի մշակչին (օրինակում՝ address): ROWS ատրիբուտը որոշում է պատուհանի բարձրությունը տողերով (օրինակում՝ 5): COLS ատրիբուտը սահմանում է պատուհանի լայնքը նշաններով (օրինակում՝ 50):

<TEXTAREA> և </TEXTAREA> տե•երի միջև տեղաբաշխված տեքստը ներկայացնում է պատուհանի լռելյայն պարունակությունը: Օգտագործողը կարող է խմբա•րել այն կամ պարզապես ջնջել:

Կարևոր է իմանալ, որ ոչ լատիներեն տառերը տեքստում կարող են վերափոխվել համապատասխան նշանային օբյեկտների:

Օրինակ՝

```
<HTML>
<HEAD>
<TITLE>Օրինակ</TITLE>
</HEAD>
<BODY>
<H1>Փոքր-ինչ ավելի բարդ ձև</H1>
<FORM ACTION="http:// www.shirak.am" METHOD=post>
<H2>Մի քիչ պատմեք Ձեր մասին</H2>
<P>Իսկական տվյալներ ցույց տալը բոլորովին էլ պարտադիր չէ: Ցուցադրման համար միան•ամայն բավարար են և հորինվածները </P>
<P>Անուն <INPUT TYPE=text SIZE=40 NAME=fn><BR>
Ազ•անուն <INPUT TYPE=text SIZE=40 NAME=ln><BR>
Սեռը <INPUT TYPE=radio NAME=gender VALUE="male" checked>Արական
<INPUT TYPE=radio NAME=gender VALUE="female">ի•ական<BR>
Տարիքը <INPUT TYPE=text SIZE=5 NAME=age>տարեկան<BR>
<INPUT TYPE=submit VALUE="Submit"></P>
</FORM>
</BODY>
</HTML>
```

HTML-ով ստեղծել ֆռեյմային web էջ' այն ըստ

տողերի բաժանելով 50%, 50%-ոց շրջանների, իսկ 50%-ոց

շրջանն էլ իր հերթին ըստ սյուների բաժանել 55%. 45% շրջանների և բաժանված շրջաններից 1-ինում տեղադրել C:\Program Files\MicrosoftOffice\OFFICE11\SAMPLES թղթապանակում գտնվող EMPID5.bmp անվանումը ունեցող նկարը,
իսկ մնացած 2-ում արտապատկերել Թիվ 12, Թիվ 13
առաջադրանքներում ստեղծվող web էջերը:Ֆռեյմային web էջ ստեղծելու համար անհրաժեշտ է իմանալ հետևյալ տեգերը և ատրիբուտները. <frameset> .... </frameset> - այս տեգի միջոցով ստեծվում ֆռեյմային էջ
<frame> - այս տեգի միջոցով ֆռեյմային web էջը բաժանվում է շրջանների, որը իր հերթին ընդունում src ատրիբուտը, որի միջոցով հայտարարվում է այն ֆայլի գտնվելու ուղղին կամ հասցեն, որը պետք է արտացոլվի նշված շրջանում
cols -այս հատկության միջոցով ֆռեյմային web էջը ըստ սյուների բաժանվում է շրջանների
rows -այս հատկության միջոցով ֆռեյմային web էջը ըստ տողերի բաժանվում է շրջանների :

Տեխնիկա-տնտեսագիտական հաշվարկ

1. Կապիտալ ներդրումների հաշվարկը նախագծվող տարբերակում
2. Կապիտալ ներդրումների հաշվարկը բազային տարբերակում
3. Շահագործման ծախսերի հաշվարկը նախագծվող տարբերակում
4. Շահագործման ծախսերի հաշվարկը բազային տարբերակում
5. Հետգնման ժամկետի հաշվարկը
6. Տարեկան էֆեկտի հաշվարկը
7. էֆեկտիվության գործակցի հաշվարկը

N.	Տեխնիկա-տնտեսական ցուցանիշ	Բազային	Նախագծվող

1.	Հիմնական աշխատավարձ (դրամ)	-	110.000
2.	Լրացուցիչ աշխատավարձ (% հիմնական աշխատավարձից)	-	10
3.	Պետ. ապահովագրական ծախսեր (% հիմնական + լրացուցիչ աշխատավարձից)	-	30
4.	Ծրագրի տեղադրման և օգտագործման համար անհրաժեշտ մակերես ($մ^2$)	0.1	0.1
5.	1 $մ^2$ - ի տարածքի արժեքը	150.000	150.000
6.	Ծրագիր մշակման համար անհրաժեշտ ժամանակ (ամիս)	-	3
7.	Հումքի, նյութերի և կիսաֆաբրիկատների ծախսերը (դրամ)	-	0
8.	Պատրաստի իրերի ծախսերը (դրամ)	-	4000
9.	Համակարգչի 1 ժամվա աշխատանքի արժեքը (դրամ)	-	200
10.	Ծրագրի կարգավորման (տեստավորման) համար անհրաժեշտ ժամանակը (ժամ)	-	80
11.	Արտաարտադրական ծախսեր (% գործարանային ինքնարժեքից)	-	3
12.	Վրադիր ծախսեր (% հիմնական + լրացուցիչ աշխատավարձից)	-	210
13.	Շահույթ (% լրիվ ինքնարժեքից)	-	30
14.	Շահագործող անձնակազմի 1 ամսվա հիմնական աշխատավարձ (դրամ)	100.000	100.000
15.	Տվյալ համակարգչի շահագործման համար զբաղեցված տարածքը ($մ^2$)	5	5
16.	Համակարգչի հզորությունը (շահագործման) (կՎտ)	0.4	0.4
17.	Ծրագրի մեծածախ գինը (դրամ)	500.000	-
18.	Շենքի շահագործման ժամկետը (տարի)	50	50
19.	1 կՎտ/ժ էլեկտրաէներգիայի արժեքը (դրամ)	25	25
20.	Համակարգչի մեծածախ գինը (դրամ)	600.000	600.000
21.	Համակարգչի շահագործման ժամկետը (տարի)	5	5
22.	Համակարգչի ընթացիկ վերանորոգ. քանակը 1 տարում	2	2
23.	1 վերանորոգման արժեքը (դրամ)	6.000	6.000
24.	Տարվա աշխատանքային օրերի քանակը	252	252
25.	Աշխատանքային օրվա տևողությունը (ժամ)	8	8
26.	Տարվա ընթացքում համակարգչի աշխատած ժամերը	1512	1000
27.	Տրանսպորտային ծախսեր (% մեծածախ գնից)	1.2	1.2
28.	Տեղադրման և մոնտաժման ծախսեր (% մեծածախ գնից)	1.3	1.3

.1. Ներածություն

Ներկայումս էլեկտրոնային հաշվիչ մեքենաները (համակարգիչները) լայն կիրառություն ունեն ոչ միայն գիտության մեջ, այլ նաև ժողովրդական տնտեսության տարբեր ճյուղերում: Ժողովրդական տնտեսության բազմաթիվ խնդիրների լուծումը ավելի արագ, հեշտ և էֆեկտիվ է կատարվում, եթե դրանք լուծվում են համակարգչի օգնությամբ՝ դրա համար մշակված ծրագրի կամ ծրագրային փաթեթի միջոցով: Ծրագիրը կարելի է օգտագործել այնքան անգամ որքան անհրաժեշտ է: Այն պետք է լինի տնտեսապես շահավետ և հարմար օգտագործման տեսանկյունից:

Տվյալ բաժնի նպատակն է հանդիսանում որոշել նախագծվող ծրագրի տեխնիկատնտեսական շահավետությունը:

Արտադրության օպտիմալ, էֆեկտիվ կազմակերպման, որակյալ արտադրանքի թողարկման, թողարկվող արտադրանքի ինքնարժեքի իջեցման և այլ ցուցանիշների բարելավման համար անհրաժեշտ է կատարել տնտեսագիտական հաշվարկներ: Դրանից հետո միայն կարելի է կատարել տեխնիկատնտեսական հիմնավորում նախագծվող տարբերակի համար:

Ծրագրի էֆեկտիվության որոշումը կատարվում է բազային և նախագծվող տարբերակների բերված ծախսերի համեմատման ճանապարհով: Դրա համար պետք է վերցնել նույն նպատակներին ծառայող արդեն գոյություն ունեցող ծրագրի (բազային տարբերակ) ցուցանիշները և համեմատել դրանք մեր կողմից նախագծվող ծրագրի ենթադրվող ցուցանիշների հետ, համեմատությունը ցույց կտա, թե ինչքանով է նոր տարբերակի նախագծումը շահավետ:

.2. Կապիտալ ներդրումների հաշվարկը նախագծվող տարբերակում

Տեսական մաս

Կապիտալ ներդրումները դա միանվագ ծախսեր են, որոնք կատարում է նախագծող կազմակերպությունը նոր սարք նախագծելու և թողարկելու համար, և իրացնողը՝ սարքը գնելու և տեղադրելու համար: Մեր դեպքի համար և նախագծող կազմակերպությունը, և իրացնողը մենք ենք, քանի որ

նախագծվում է նոր ծրագիր, և այդ ծրագիրը գրելը և բոլոր ծախսերը կատարելը արվում է մեր կողմից: Ցանկացած օբյեկտի վրա կատարվելիք ներդրումները հաշվելու համար կազմվում է նախահաշիվ (սմետա), որտեղ համաձայն նորմատիվային տվյալների հաշվարկվում է օբյեկտի արժեքը:

Կապիտալ ներդրումները հաշվարկվում են հետևյալ բանաձևով.

$$K_{նախ}= S_{մեծ} + K_{շենք} + K_{տրանս} + K_{մոնտ} + K_{լրաց}$$

որտեղ՝

- $K_{նախ}$ -ը նախագծվող տարբերակի կապիտալ ներդրումներն են
- $K_{շենք}$-ը շենքի շահագործման արժեքն է, որը որոշվում է հետեւյալ կերպ՝

$$K_{շենք} = S_{շենք}*C$$

- $K_{տեղ}$ -ը տեղափոխման վրա կատարված ծախսերն են, որոնք որոշվում են հետեւյալ կերպ՝

$$K_{տեղ} = 1.2\% \ S_{մեծ}$$

- $K_{մոնտ}$ -ը տեղադրման եւ մոնտաժման վրա կատարված ծախսերն են, որոնք մեր տարբերակում հավասար են 0-ի $K_{մոնտ} = 0$
- $K_{լրաց}$-ը լրացուցիչ ծախսերն են, բայց քանի որ մենք ծրագրի մշակման ընթացքում ոչ մի լրացուցիչ սարք չենք օգտագործում, հետեւաբար այս բաղադրիչը հավասար է 0 -ի:
- $S_{մեծ}$-ը ծրագրի մեծածախ գինն է, համարվում է այն գինը, որով արտադրողը իրացնում է իր արտադրանքը:
- Այն որոշվում է հետեւյալ կերպ՝

$$S_{մեծ} = S_{լր} + Շ$$

որտեղ՝

- **Շ** -ն շահույթն է, որը կազմում է Sլր -ի (լրիվ ինքնարժեքի) 30% -ը:
- $S_{լր}$-ը այն բոլոր ծախսերն են, արտահայտված դրամական ձևով, որը կատարում է ձեռնարկությունը տվյալ արտադրանքը թողարկելու եւ իրացնելու վրա:

Լրիվ ինքնարժեքը հաշվում են հետեւյալ բանաձեւով՝

$$S_{լր} = S_{գ} + U_{ա.ծ}$$

որտեղ՝

- $U_{ա.ծ}$ -ն արտաարտադրական ծախսերն են (օր. գովազդ, փաթեթավորում եւ այլն): Սա.ծ -ն կազմում է Sգ -ի (գործարանային ինքնարժեքի) 6% - ը ըստ տրված ցուցանիշների:
- $S_գ$ -ն այն բոլոր ծախսերն են, արտահայտված դրամական ձեւով, որոնք կատարում է ձեռնարկությունը տվյալ արտադրանքը թողարկելու համար:

Գործարանային ինքնարժեքը հաշվում են հետեւյալ բանաձեւով՝

$$S_գ = U_{հ.ն.կ} + U_{պ.հ} + U_{ա.ֆ} + U_{կարգ} + U_{վր.ծ}$$

որտեղ՝

- $U_{հ.ն.կ}$ -ը ծրագրի վրա ծախսվող հումքի, նյութի եւ կիսաֆաբրիկատի վրա կատարված ծախսերն են:
- Սհ.ն.կ = 0, քանի որ մենք ոչ մի հումք, նյութ, կիսաֆաբրիկատ չենք օգտագործում մեր ծրագիրը գրելու համար:
- Սպ.հ -ը պատրաստի իրերի ծախսերն են՝

Պատրաստի իրերի մեջ մտնում են սկավառակներ, թուղթ, մատիտ, պատրաստի ծրագրեր եւ այլն:

- $U_{վր.ծ}$ -ը վրադիր ծախսերն են, որոնք գործարանային եւ արտադրամասային ծախսերի գումարն են (օրինակ՝ ադմինիստրացիայի աշխատավարձը) եւ կազմում է հիմնական եւ լրացուցիչ աշխատավարձերի 210% -ը (մեկ ամսվա համար), այսինքն՝

$$U_{վր.ծ} = 210\% \ (A_{հիմ} + A_{լր}) * m$$

որտեղ՝

- m -ը ծրագիրը գրելու վրա ծախսված ամիսների քանակն է:
- Սա.ֆ -ը բանվորների աշխատավարձի ֆոնդն է: Այդ ֆոնդը հիմնական աշխատավարձի (Aհիմ), լրացուցիչ աշխատավարձի (Aլր) և պետ. ապահովագրական ծախսերի գումարն է ($A_{պ.ա.ծ}$), այսինքն

$$U_{ա.ֆ} = (A_{հիմ} + A_{լր} + A_{պ.ա.ծ}) * m$$

որտեղ՝

- $A_{հիմ}$ -ը հիմնական աշխատավարձն է:
- $A_{լր}$ -ը լրացուցիչ աշխատավարձն է և կազմում է հիմնական աշխատավարձի 10% -ը՝

$$A_{լր} = 10\% A_{հիմ}$$

- $A_{պ.ա.ծ}$ -ը պետ. ապահովագրական հատկացումներն են, որոնք գնում են պետ. բյուջե և կազմում են հիմնական և լրացուցիչ աշխատավարձների գումարի 30% -ը՝

$$A_{պ.ա.ծ} = 30\% (A_{հիմ} + A_{լր})$$

- $U_{կարգ}$ -ը տվյալ ծրագրի կարգավորման վրա կատարված ծախսերն են, այսինքն՝

$$U_{կարգ} = T_{կարգ} * C$$

որտեղ՝

- $T_{կարգ}$ -ը կարգավորման համար անհրաժեշտ ժամանակն է:
- C -ն համակարգչի մեկ ժամվա աշխատանքի արժեքն է:

Հաշվարկային մաս

Ունենալով հիմնական աշխատավարձը՝ $A_{հիմ}$ = 110.000 դրամ, որոշենք լրացուցիչ աշխատավարձը.

$A_{լր} = 10\% A_{հիմ} = 0.1 * 110.000 = 11.000$ (դրամ)

$A_{պ.ա.ծ} = 30\% (A_{հիմ} + A_{լր}) = 0.3 * (110.000 + 11.000) = 36.300$ (դրամ)

$U_{ա.ֆ.} = (A_{հիմ} + A_{լր} + A_{պ.ա.ծ}) * m = (110.000 + 11.000 + 36.300) * 3 =$

471900 (դրամ)

$U_{վր.ծ} = 210\% (A_{հիմ} + A_{լր}) * m = 2.1 * (110.000 + 11.000) * 3 = 762300$ (դրամ)

$U_{կարգ} = T_{կարգ} * C = 80 * 200 = 16.000$ (դրամ)(test)

$S_գ = U_{պ.հ.} + U_{ա.ֆ} + U_{կարգ} + U_{վր.ծ} = 4000 + 471900 + 16.000 + 762300 =$

$= 1.290.200$ (դրամ)

$S_{ա.ծ} = 3\% S_գ = 0.03 * 1.290.200 = 38706$(դրամ)

$S_{լր} = S_գ + S_{ա.ծ} = 1.290.200 + 38706 = 1.328.906$ (դրամ)

$Շ = 30\% S_{լր} = 0.3 * 1.328.906 = 39867$ (դրամ)

$S_{մեծ} = S_{լր} + Շ = 1.328.906 + 39867 = 1.368.773$ (դրամ)

$K_{շենք} = S_{շենք} * C = 150.000 * 0.1 = 15000$ (դրամ)

$K_{տեղ} = 1.3\% S_{մեծ} = 0.013 * 1.368.773 = 17.794$(դրամ)

$K_{մոնտ} = 1.2\% S_{մեծ} = 0.012 * 1.368.773 = 16.425$ (դրամ)

$K_{լրաց} = 0$

$K_{նախ} = S_{մեծ} + K_{շենք} + K_{տեղ} + K_{մոնտ} + K_{լրաց} = 1.368.773 + 15000 + 17.794 + 16.425 + 0 = 1.417.392$ (դրամ)

Այսպիսով ստացանք, որ նախագծվող տարբերակի համար կապիտալ ներդրումները կազմում են 1.417.392 դրամ:

.3. Կապիտալ ներդրումների հաշվարկը բազային տարբերակում

Տեսական մաս

Բազային տարբերակը արդեն գոյություն ունեցող ծրագիրն է, որի համար արդեն գոյություն ունի մեծածախ գին: Կապիտալ ներդրումների հաշվարկը բազային տարբերակում կատարվում է նույն ձեւով, ինչ նախագծվող տարբերակում, ուստի՝

$$K_{բազ} = S_{մեծ} + K_{շենք} + K_{տեղ} + K_{մոնտ} + K_{լրաց}$$

որտեղ՝

- $S_{մեծ}$ - ծրագրի մեծածախ գինն է:

- $K_{շենք}$ -ը շենքի շահագործման ծախսն է:
- $K_{տեղ}$ -ը տեղափոխման վրա կատարված ծախսերն են, որը հավասար է ըստ ցուցանիշների 1.2% ծրագրի մեծածախ գնից:
- $K_{մոնտ}$ -ը մոնտաժման վրա կատարված ծախսերն են, որը հավասար է 0
- $K_{լրաց}$ - ը լրացուցիչ ծախսերն են, $K_{լրաց} = 0$:

Հաշվարկային մաս

$S_{մեծ}$ = 500.000 դրամ

$K_{տեղ}$ = 1.3 % $S_{ծ.մեծ}$ = 0.013 * 500.000 =6500 (դրամ)

$K_{մոնտ}$ = 1.2% $S_{ծ.մեծ}$ = 0.012 * 500.000 =6000(դրամ)

$K_{շենք} = S_{շենք}*C$ = 150.000 * 0.1 = 0.1 = 15000 (դրամ)

$K_{բազ.}$ = 500.000 + 15000+ 6.000 + 6500 = **527500** (դրամ)

Այսպիսով ստացանք, որ բազային տարբերակի համար կապիտալ ներդրումները կազմում են **527500**դրամ:

.4. Շահագործման ծախսերի հաշվարկը նախագծվող տարբերակում

Տեսական մաս

Շահագործման ծախսերի տակ հասկանում ենք այն բոլոր ծախսերը, որոնք կատարվում են տվյալ արտադրանքը (ծրագիրը) շահագործելու համար: Շահագործման ծախսերը հաշվարկվում են մեկ տարվա կտրվածքով:

Շահագործման ծախսերը հիմնականում բաղկացած են հետևյալ տնտեսական տարրերից՝

$$U_{շահ} = U_{շենք} + U_{սարք} + U_{էէ} + U_{ա.ֆ} + U_{ը.վ} + U_{ը.տ} + U_{լ.}$$

որտեղ՝

- $U_{շենք}$ - ը շենքի ամորտիզացիան է:

  Հիմնական ֆոնդերի մաշվածությունը արտահայտված դրամական ձևով կոչվում է ամորտիզացիա: Այն փոխանցվում է թողարկվող արտադրանքի ինքնարժեքի վրա: Շենքի ամորտիզացիայի հաշվարկը կատարվում է հետևյալ բանաձևով՝

$$U_{շենք} = (f_{նախ} - L) / T_{շ.ժ.}$$

որտեղ՝

- $f_{նախ}$-ը հիմնական ֆոնդերի նախնական արժեքն է: Մեր տարբերակում շենքը դա այն սենյակն է, որտեղ շահագործվում է մեր ծրագիրը: Այն որոշելու համար պետք է 1մ2 - ու արժեքը բազմապատկել ծրագիրը շահագործելու համար անհրաժեշտ մակերեսով:
- L-ը հիմնական ֆոնդերի լուծարման արժեքն է՝ մեր տարբերակի համար հավասար է 0 -ի:
- $T_շ$ -ը շենքի շահագործման ժամկետն է:
- Uսարք - ը համակարգչի ամորտիզացիան է:

$$U_{սարք} = (f_{նախ} - L) / T_{շ.ժ}$$

որտեղ՝

- fնախ -ը հավասար է համակարգչի գնին:
- $T_{շ.ժ}$ -ը համակարգչի շահագործման ժամկետն է:
- $U_{էէ}$ - ն տարվա ընթացքում էլեկտրաէներգիայի վրա ծախսված գումարն է:

$$U_{էէ} = W * n * N * C$$

որտեղ՝

- W - ն օգտագործվող համակարգչի հզորությունն է

- n - ը 1 օրվա աշխատանքային ժամերի քանակն է
- N - ը 1 տարվա ընթացքում աշխատած օրերի քանակն է
- C - ն 1 կՎտ/ժ էլեկտրաէներգիայի արժեքն է
- $U_{ա.ֆ}$ -ը աշխատավարձային ֆոնդն է

$$U_{ա.ֆ} = (A_{հիմ} + A_{լր} + A_{պ.ա.ծ}) * m$$

որտեղ՝

- $A_{հիմ}$ -ը շահագործող անձնակազմի հիմնական աշխատավարձն է
- $A_{լր}$ -ը շահագործող անձնակազմի լրացուցիչ աշխատավարձն է
- $A_{պ.ա.ծ}$-ը պետ. ապահովագրական հատկացումներն են (ծախսերը):
- m -ը ամիսների քանակն է և քանի որ շահագործման ծախսերը հաշվում ենք մեկ տարվա կտրվածքով, ուստի՝ m = 12:

- $U_{ը.վ}$ -ը ընթացիկ վերանորոգման վրա կատարած ծախսերն են,

$$U_{ը.վ} = N * C$$

որտեղ՝

- N -ը վերանորոգումների քանակն է:
- C -ն 1 վերանորոգման գինն է

- $U_{ը.տ}$ -ն ընթացիկ տնտեսական ծախսերն են, որոնք իրենցից ներկայացնում են տարեկան հիմնական և լրացուցիչ աշխատավարձների գումարի 90% - ին՝

$$U_{ը.տ} = 90\% (A_{հ} + A_{լր}) * m$$

- $U_{լր}$-ը լրացուցիչ սարքերի շահագործման ծախսերն են, որը այս դեպքում հավասար է 0-ի, քանի որ լրացուցիչ սարքեր չեն օգտագործվում:

$U'_{նախ}$ -ը շահագործման ժամկետի ընթացքում շահագործման ծախսերն են և որոշվում է հետևյալ բանաձևով՝

$$U'_{նախ} = U_ա * t_ա / (N * n_ա)$$

որտեղ՝

- $U_ա$ -ը շահագործման ծախսերն են տարեկան կտրվածքով:
- $t_ա$ - ը ծրագրի համակարգչի վրա աշխատած ժամերի քանակը
- N - ը տարվա աշխատանքային օրերի քանակը
- $n_ա$ - ը օրվա աշխատանքային ժամերի քանակը

Հաշվարկային մաս

$U_{շահ} = U_{շենք} + U_{սարք} + U_{էէ} + U_{ա.ֆ} + U_{ը.վ} + U_{ը.տ} + U_լ$

$U_{ա.ֆ} = (A_{հիմ} + A_{լր} + A_{պ.ա.ծ}) * 12 = (100.000 + 10.000 + 11.000) * 12 =$

$= 1.716.000$ (դրամ)

$U_{ը.տ} = 90\% (A_{հիմ} + A_{լր}) * m = 0.9 * (100.000 + 10.000) * 12 = 1.320.000$ (դրամ)

$U_{էէ} = 0.4 * 8 * 252 * 25 = 20160$ (դրամ)

$U_{ը.վ} = 2 * 6.000 = 12.000$ (դրամ)

$U_{նախ} = S_{շենք} * C = 150.000 * 4 = 600.000$ (դրամ)

$T_{շ.ժ.} = 40$ տարի

$U_{շենք.ա} = 600.000 / 40 = 15.000$ (դրամ)

$U_{սարք.ա} = f_{նախ} / T_{շ.ժ.}$

$f_{նախ}$ = 600.000 դրամ

$U_{սարք. ա}$ = 600.000 / 5 = 120.000 (դրամ)

$U_{շահ}$ = 15.000 + 120.000 + 20160+ 1.716.000 + 12.000 + 1.320.000+600.000 =

= 3.803.160 (դրամ)

Այսպիսով ստացանք, որ նախագծվող տարբերակի համար շահագործման ծախսերը տարեկան կտրվածքով կազմում են 3.803.160 դրամ:

$$U'_{նախ} = U_{ն} * t_{ն} / (N * n_{ն}) = 3.803.160 * 1000 / 2016 = 1.885.993$$

Այսպիսով ստացանք, որ նախագծվող տարբերակի համար շահագործման ծախսերը շահագործման ժամկետի ընթացքում կազմում են 1.885.993դրամ:

12.5. Շահագործման ծախսերի հաշվարկը բազային տարբերակում

Տեսական մաս

Այս դեպքում հաշվարկները իրականացվում են լրիվ նույն սկզբունքով, ինչպես որ հաշվարկվել են շահագործման ծախսերը նախագծվող տարբերակում: Քանի որ օգտագործվում է նույն համակարգիչը , հետևաբա նախագծային և բախային տարբերակներում շահագործման ծախսերը նւոյնն են :

$U'_{\text{բազ}}$ -ը շահագործման ժամկետի ընթացքում շահագործման ծախսերն են եւ որոշվում են հետևյալ բանաձևով՝

$$U'_{\text{բազ}} = U_{\text{բ}} * t_{\text{բ}} / (N * n)$$

որտեղ՝

- $U_{\text{բ}}$ -ը շահագործման ծախսերն են տարեկան կտրվածքով:
- $t_{\text{բ}}$ - ը ծրագրի համակարգչի վրա աշխատած ժամերի քանակը
- N - ը տարվա աշխատանքային օրերի քանակը
- n - ը օրվա աշխատանքային ժամերի քանակը

Հաշվարկային մաս

$U_{\text{շահ}}$ = 3.803.160 (դրամ)

Այսպիսով ստացանք, որ բազային տարբերակի համար շահագործման ծախսերը տարեկան կտրվածքով կազմում են 3.803.160 դրամ:

$U'_{\text{բազ}} = U_{\text{բ}} * t_{\text{բ}} / (N * n_{\text{բ}})$ = 3.803.160 *** 1512 /**2016=2.852.370 (դրամ)

Այսպիսով ստացանք, որ բազային տարբերակի համար շահագործման ծախսերը շահագործման ժամկետի ընթացքում կազմում են 2.852.370 դրամ:

12.6. Հետգնման ժամկետի հաշվարկը

Հետգնման ժամկետը, դա այն ժամանակն է, որի ընթացքում կհետգնվի ներդրված գումարը: Հետգնման ժամկետի որոշման համար անհրաժեշտ է իմանալ, թե տարվա ընթացքում քանի ժամ է աշխատել մեքենան նախագծվող և բազային տարբերակներում ($n_ն$, $n_բ$): Հետգնման ժամկետի բանաձևը հետևյալն է՝

$$T_{հ.ժ.} = (K_{նախ} - K_{բազ}) / (U'_{բազ} - U'_{նախ})$$

որտեղ՝

- $U'_{նախ}$ -ը շահագործման ժամկետի ընթացքում շահագործման ծախսերն են նախագծվող տարբերակում:

$$U'_{նախ} = 1.885.993 դրամ$$

- $U'_{բազ}$ -ը շահագործման ժամկետի ընթացքում շահագործման ծախսերն են բազային տարբերակում:

$$U'_{բազ} = 2.852.370 դրամ$$

- $K_{նախ}$ - ը կապիտալ ներդրումներն են նախագծվող տարբերակում:

$$K_{նախ} = 1.417.392 դրամ$$

- $K_{բազ}$ - ը կապիտալ ներդրումներն են բազային տարբերակում:

$K_{բազ}$ = **527.500**դրամ

$T_{հ.ժ.}$ = 889.892/966.377=0.9տարի

Այսպիսով ստացանք, որ հետգնման ժամկետը կազմեց 0.9 տարի:

12.7. Տարեկան էֆեկտի հաշվարկը

Տարեկան էֆեկտը իրենից ներկայացնում է բերված ծախսերի տարբերությունը: Բերված ծախսերը դա ($U' + E_ն k$) -ն է: Որտեղ $E_ն$ -ը էֆեկտիվության գործակիցն է (սովորաբար ընդունվում է, որ $E_ն$ = 0.3 - ի), U՝-ը շահագործման ծախսերն են, K-ն կապիտալ ներդրումները: Տարեկան էֆեկտի հաշվարկման բանաձևը կունենա հետևյալ տեսքը՝

$N_{տ.էֆ} = (U'_{բազ} + E_ն * K_{բազ}) - (U'_{նախ} + E_ն * K_{նախ})$ =(2.852.370 +0.3* **527.500**)-(1.885.993+0.3*1.417.392)= 3.010.620-2.311.211=699.181 (դրամ)

Այսպիսով ստացանք, որ տարեկան էֆեկտը կազմում է **1175245.7** դրամ:

12.8. Էֆեկտիվության գործակցի հաշվարկը

Էֆեկտիվության գործակիցը հետգնման ժամկետի հակադարձ մեծությունն է, ուստի այն հավասար է՝

$$E_{հաշ} = 1/ T_{հ.ժ.} = 1 / 0.9 = 1.1$$

Այսպիսով ստացանք, որ տարեկան էֆեկտը կազմում է **1.1**:

.9. Եզրակացություն

Այսպիսով, այս հաշվարկները ցույց տվեցին, որ նախագծվող տարբերակը էֆեկտիվ է բազայինից, քանի որ հաշվարկային էֆեկտիվության գործակիցը ($E_{հաշ}$)մեծ է նորմատիվային էֆեկտիվության գործակցից ($E_{ն}$ = 0.3) ($E_{հաշ} > E_{n}$) և հետգնման ժամկետը ($T_{հ.ժ.}$) փոքր է նորմատիվային հետգնման ժամկետից ($T_{ն.հ.ժ.}$ = 3.33 տարի) ($T_{հ.ժ.} < T_{ն.հ.ժ.}$):

Աշխատանքային պայմանների բնութագրերը և աշխատանքի պաշտպանության հիմնական պահանջները

Դիպլոմային աշխատանքը կատարվում է աշխատասենյակային պայմաններում (լաբարատոր պայմաններում) : Որտեղ անհրաժեշտ է հաշվի առնել այն հնարավոր վտանգավոր և վնասակար գործոնները, որոնք կարող են բացասական ազդեցություն ունենալ մարդկանց առողջական վիճակի վրա :

Աշխատասենյակում հնարավոր վտանգավոր և վնասակար գործոնները են՝

- ճառագայթում
- անբավարար լուսավորվածություն
- աղմուկ
- էլեկտրական հոսանք
- անբավարար միկրոկլիմայական պայմաններ
- հրդեհ

Ճառագայթում : Աշխատասենյակում ճառագայթման հիմնական աղբյուրն են՝

- համակարգիչները
- մոնիտորները ,

որոնք հանդիսանում են ելտրամագնջսական և ռենտգենյան ճառագայթումների աղբյուր :

Ռենտգենյան ճառագայթներն իոնիզացնող ճառագայթներ են, որոնք ազդելով մառդկանց բջիջների վրա իոնացնում են դրանք, ինչի արդյունքում բջիջը կամ մահանում է, կամ պահպանվում է ինֆորմացիայի որոշ մասի կորուստով : Այսպիսի բջիջները կարող են ուռուցքի առաջացման պատճառ հանդիսանալ :

Էլեկտրամագնիսական ճառագայթներն իոնիզացնող չեն : Նրանց երկարատև ազդեցությունը կարող է բերել նյարդային համակարգի խանգարման, դիմադրողականության իջեցման, սիրտ-անոթային համակարգի խանգարումների, ինչպես նաև, հղիության ընթացքում հանգեցնել պտղի մոտ անոմալիաների :

Աշխատասենյակում ճառագայթումներից պաշտպանվելու համար հարկավոր է ՝

- օգտագործել պլազմային մոնիտորներ կամ պաշտպանիչ էկրաններ

- պահպանել մոնիտորից ամենաքիչը 0.3մ հեռավորություն և այլն :

Անբավարար լուսավորվածություն : Աշխատասենյակում անբավարար լուսավորվածության պատճառ կարող են հանդիսանալ սխալ բնական և արհեստական լուսավորվածությունները : Բնական լուսավորվածության անբավարարության աղբյուր կարող են լինել լուսամուտների տեղը, քանակը, չափերը, աղտոտվածությունը լույսի ներթափանցմանը խոչնդոտող առարկաների առկայությունը : Արհեստական լուսավորվածության անբավարարության աղբյուր կարող են հանդիսանալ լուսամփոփների սակավությունը, մաշվածությունը, սխալ տեղաբաշխումը և այլն :

Անբավարար լուսավորվածության պայմաններում աշխատողի մոտ գերլարվում են տեսողական օրգանները, խանգարվում է նյարդային համակարգը, ինչպես նաև մեծանում է դժբախտ պատահարների հավանականությունը, քանի որ հնարավոր է պատահաբար դիպչել սարքավորումների վտանգավոր մասերին :

Աշխատասենյակում անբավարար լուսավորվածության դեմ պայքարելու համար հարկավոր է ՝

- ապահովել լուսամուտների ճիշտ նախագծումը համապատասխան նորմաներով
- հեռացնել լույսի ներթափանցմանը խոչնդոտող առարկաները
- ապահովել բնական լուսավորվածության նորմատիվային գործակիցը՝

  e = 1.6 %

  լուսամփոփները տեղադրել այնպես, որ ապահովվի արհեստական

  լուսավորվածություն նորման

  կոմբինացված լուսավորվածության դեպքում՝ 750 լյուքս,

  ընդհանուր լուսավորվածության դեպքում՝ 400 լյուքս:

Աղմուկ : Աշխատասենյակում աղմուկի հիմնական աղբյուր են հանդիսանում՝

- համակարգիչները
- տպիչ սարքերը
- օդափ և ոխիչները
- այլ տեխնիկական սարքեր

Աղմուկի տևական ազդեցության հետևանքով մարդկանց մոտ առաջանում են նյարդային համակարգի խանգարումներ, սիրտ-անոթային համակարգի խանգարումներ, հոգնածություն, քնի խանգարում, ուշադրության խախտում, կարող է բարձրանալ կամ իջնել արյան ճնշումը, ընկնում են աշխատողի արտադրողականությունը և արտադրանքի որակը, ինչպես նաև, մեծանում է ոչ ճիշտ գործողությունների հավանականությունը, ինչի արդյունքում հնարավոր են դժբախտ պատահարներ :

Աշխատասենյակում աղմուկից պաշտպանվելու համար հարկավոր է ՝

- օգտագործել հնարավորինս անաղքուկ աշխատող սարքավորումներ
- տեղադրել ձայնամեկուսիչ պատուհաններ և այլն :

Էլեկտրական հոսանք : Էլեկտրական սարքավորումների հետ աշխատելիս միշտ առկա է հոսանքահարման վտանգը: Աշխատասենյակում հոսանքահարման պատճառ կարող են հանդիսանալ՝

- վատ մեկուսացված էլեկտրական սարքավորումները
- վատ մեկուսացված հաղորդալարերը և այլն:

Հոսանքը վտանգավոր է նրանով, որ հաղորդիչին չհպված մարդը չի զգում հոսանքի առկայությունը, ուստի չի կարող կանխատեսել սպառնացող վտանգը: Հոսանքահարումը մարդու օրգանիզմի վրա կարող է ունենալ ջերմային, էլեկտրոլիտիկ կամ քիմիական, կենսաէլեկտրական կամ մեխանիկական ազդեցություններ : Ջերմային ազդեցության դեպքում առաջանում են այրվածքներ, էլեկտրոլիտիկի դեպքում ՝ արյան և մօուս օրգանական հեղուկների տարալուծում, կենսաէլեկտրականի դեպքում ՝ հօուսվածքների խզում, ոսկորների կոտրվածքներ և այլն :

Հոսանքահարումից խուսափելու համար հակավոր է ՝

- զգուշանալ լարման տակ գտնվող մասերից, երբ համակարգիչը միացված է
- մեկուսացնել էլեկտրական սարքավորումները
- մեկուսացնել հաղորդալարերը և այլն:

Անբավարար միկրոկլիմայական պայմաններ : Աշխատասենյակում անբավարար միկրոկլիմայական պայմանների պատճառ կարող են հանդիսանալ `

- համակարգիչները
- լուսամփոփները
- տպիչ սարքերը
- այլ տեխնիկական սարքեր

Աշխատասենյակում անբավարար միկրոկլիմայական պայմանները բացասաբար են ազդում աշխատողների աշխատունակության և արտադրողականության, ինչպես նաև, հաշվողական տեխնիկայի հուսալի աշխատանքի վրա : Տեխնիկայի աշխատանքի ընթացքում անջատվում է լրացուցիչ ջերմաքանակ, որի հետևանքով խախտվում է մարդու օրգանիզմի ջերմային հավասարակշռությունը :

Միկրոկլիմայի պարամետրերի թույլատրելի միջակայքերն են՝

- օդի ջերմաստիճանը

  ամռանը ՝ 18 - 23 աստիճան,

  ձմռանը՝ 17 - 20 աստիճան,

- օդի հարաբերական խոնավությունը՝ 50 – 60% :

Աշխատասենյակում նորմալ միկրոկլիմայական պայմաններ ապահովելու համար հարկավոր է տեղադրել՝

- ջեռուցիչներ
- օդափոխիչներ՝ արհեստական օդափոխություն կատարելու համար :

Հրդեհ : Համակարգչային աշխատասենյակում հրդեհի առաջացման աղբյուր կարող են հանդիսանալ՝

- համակարգիչները
- էլեկտրոնային սխեմաները
- հոսանքի սնուցման սարքերը
- օդափոխիչները
- էլեկտրական կայծերը և այլն :

Հրդեհը կարող է հասցնել մարդկային զոհեր, վնասվածքներ, ինչպես նաև, նյութական մեծ վնասներ :

Հրդեհից պաշտպանվելու համար հարկավոր է `

- աշխատասենյակում ունենալ հրդեհամարման առաջնային միջոցներ, կրակմարիչներ, չոր ավազ, փրփուր և այլն
- միջանցքներում տեղադրել հրդեհային ծորակներ,որոնք տեղադրվում են հատակից 1.35 մ բարձրության վրա, հասանելի և անվտանգ տեղերում ,
- կիրառել ազդանշանող սարքեր:

Մարդածին ազդեծությունները արտաքին միջավայրի և մարդու առողջության վրա

Օրեկան մարդը ընդերքից հանում է մոտ 200մլդ ածուխ :Օգտգործում է ավելի քան 9մլդ վառելիք և տալիս դաշտերին ավելի քան 3մլդ պարարտանույթ:

Այսօր հիմնականի կարևոր խնդիր է հանդիսանում, արտաքին միջավայրի աղտոտումը :

Արտաքին միջավայրի վրա բացասական ազդեցություն է ունենում տրանսպորտային երթևեկությունը .ատոմային զենքի փորցակումը,դեպի մթնոլորտը կատարվող արտանետումները և այլն:

Համաշղարային մակարդակով ավտոմեքենաները տարեկան արտադրում են

մլն տ

ածխածնի օքսիդ -260

ազոտի օքսիդ-20

Պինդ վառելիքի այրման ժամանակ այրման խցիկից արտանետվում է H2O,CO2,HCI,CO,NO,CI ինչպես նաև ինչպես նաև AI2O3-ի մասնիկներ ,միջին 0.1 մկմ չափով /երբեմն նույնիսկ՝ 10 մկմ:

Համաձայն GOST 170201 -76-ի դեպի մթնոլորտ կատարվող արտանետումները դասակարգվում են ՝

- Արտանետումների մեջ պարունակվող վնասակար նյոթեր ագրեգատային վիճակներով ՝SO2,CO,NOx ածխաջրերի նայլն գազանման հեղուկներ ,թթուներ, օրգանական միացություններ ,աղերի և հեղուկ մետաղների լուծույթներ ,կապար և նրա միացություներ ,օրգանական և անօրգանական փոշի ,մուր ,խեժային նյութեր և այլն:
- Ըստ մասսայական արտանետման ,առանձնանում են արտանետումների հետևյալ խմբերը

Առավել վտանգավոր է դարձել արտաքին միջավայրի աղտոտումը մարդածին թունավուր նյութերով :Շատ երկրներում հազարավոր լճեր թթվային անձրևների

ազդեծությունից համարվում են կեսաբանորեն մեռած : Ոչնչացաման եզրին են կանգնած 25-30 հազար տեսակի խոտաբույսեր և այսպես շարունակ:

Այսպիսով արտաքին միջավայրի ինտեսիվ դեգրադացիան սպառնալիք է հանդիսանում մարդու կյանքի գոյության համար :Հատկապես քաղաքի բնակչության վրա ,որտեղ շատանում է այն մարդկանց քանակը ,որոնք հիվանդ են խրոնիկական, բրոնխիտ ,ասմմա,ալերգիա ,քաղցկեղ և այլ անբուժելի հիվանդություններով :

Հենց այս պատճառով ես հետաքրքրվեցի այս խնդրով ,քանի որ մարդու կյանքի գոյության ,սննդի որակի ,բերքատվության,կլիմայի ինչպես նաև Երկրի օզոնային շերտի վրա,որի վաթարացումը բերում է կենդանական և բուսական կյանքի ոչնչացմանը:Բացի դրանւց ,մթնոլորտ թափանցող նյութերը ,գազերը ,օքսիդները նույնպես բացասական ազդեցություն են թողնում մարդու օրգանիզմի վրա:

CO…..Անգույն առանց հոտի գազ է : Ազդում է զգայարանների ,սրտի անոթների աշխատանքի վրա `առաջացնելով շնչահեղություն :Սկզբնական նշաններն են ` գլխացավի առաջացում ,գլղապտույտ:

SO2……Անգույն գազ է սուր հոտով :Նույնիսկ շատ թույլ խտության դեպքում բերանի խոռոչում առաջացնում է ոջ հաճելի համ :

CO2…….. Անգույն առանց հոտի գազ է: Ազդում է զգայարանների ,սրտի անոթների աշխատանքի վրա:

Ազոտի օքսիդներ`NO,NO3,NO2 մթնոլորտ է ներթափանցում ,որը իրենից ներկայացնում է անգույն առանց հոտի թունավոր գազ է ,ազդում են շնչառական օրգանների վրա,հատկապես վտանգավոր են ազոտի օքսիդները քաղաքում:Օդի թունավորված ազոտի օքսիդով սկսում է ազդել մարդու օրգանիզմի վրա առաջացնելով թույլ հազ .փսխում,գլխացավ:

Ալդեհիդները`ազդում են մարդու արցունքագեղձերի ,շնչառական օրգանների վչա,նաև առաջացնում է գլխացավ թուլություն:

Ածխաջրածիններ`նույնիսկ թույլ խտության դեպքում առաջացնում է գլխացավ ,գլխապտույտ ,հազ և այլն: Տվայլների վերլուծությունը ցույց է տալիս ,որ առավել մեծ քանակությամբ թունաքիմիկատներ են պարունակում ՆԱՇ-ով գազատարների արտանետումները՝ CO,NOx,CnHm միացությունների հաշվին:

Դիզելային ՆԱԾ-ները արտանետում են մեծ քնքկությամբ մուր ,որը ի սկզբանե,թունավոր չէ,սակայն մրի մասնիկները իրենց մակերևույթի վրա պարունակում են թունավոր,այդ թվում նաև կոնցերոգեն տարրեր:Մուրը երկար ժամանակ կարող ՝ մնալ օդում կախված ՝դրանով իսկ մեծացնելով մ,զրդու վրա թունավոր նյութերի ազդեծությունը:

Էկոլոգիայի թունավոր հետևանք է առողջության զանգվածային դեգրադացիան :

Վերջին տասնամակյակների ընթացքում երկրի արտաքին միջավայրը ռադիոալիքների ազդեծության ներքո ենթարկվել է փոփոխման այնպես,որ դա կարծես այն Երկիր մոլորակը չէ ,որը ենթարկվել է էվոլյուցիայի ,այլ մեկ ուրիշ, դաժան Երկիր մոլորակ:էկոլոգիայի մասսայական աղետը դարձել է աղետ Էնդոէկոլոգիայի ,այսինքն սպառնալիք ներքին օրգանիզմի վրա :

Արտաքին միջավայրի աղտոտումը հանգեցրել է ներքին միջավայրի ,ներքին օրգանիզմի աղտոտմանը :Կարծես քիչ էր այն հիվանդութըուննները որոնք ունեն մարդիկ ,այսօր սկսեցին ի հայտ գալ շատ անբուժելի հիվանդություններ :Բացի այդ Էկոլոգիայի խախտումը ազդում է նաև մարդու հագեկան ներաշխարի վրա:

Վերջին 4 տարիների ընթացքում ծննդաբերությունը ընկել է 30 տոկոսով ,իսկ մահը աճել է 10 տոկոսով :7 տարեկան երեխաներից առողջ են ընդամենը 23 տոկոսը ,իսկ 17 տարեկան երեխաներից՝ ընդամենը 14 տոկոսը:Մեր տղամարդիկ ապրում են 7-10 տարի ավելի քան զարգացած երկրներում Շատ շրջաններում ալերգիկ հիվանդությամբ տառապում է երեխաների կեսից ավելին:

Էկոլոգիական թունանյութերից ագրեսիան իրենից ներկայացնում է հատուկ պատալոգիայի տեսակ :Ներթափանցելով մարդու օրգանիզմը:Էկոլոգիական թունավոր նյութերը չեն բնակվում մարդու այրման մեջ,այլ բնակություն են հաստատում միջբջջային հյուսվածքներում :

Որպեսզի կարողանաս պահպանել առողջությունդ և կարողանաս ապրել այս ծանր պայմաններում ,հարկավոր է ժամանակ առ ժամանակ մաքրել օրգանիզմը ՝ նրանում հավաքված թունանյութերիo մակարդակը հասցնելով անվտանգ մակարդակի:

Շատ հայտնի միջոց է օրգանիզմի ազատումը թույներից խոտաբույսերի միջոցով ՝օրինակ սև հաղարջ և այլն:

Բայց մինչ մեր օրգանիզմի մասին մտածելը անհրաժեշտ է մտածել արտաքին միջավայրի անվտանգության պահպանման մասին:

Եզրակացություն

Այսպիսով կարող ենք ասել որ այս դիպլոմային աշխատանքը հաջողված է:Ունեինք MonitorUs կայք և Google API և ստեղծեցինք Google –ի օրացույցի ինտեգրացիան MonitorUs –ին:Մեր արտադրանքը նախատեսված է ոչմիայն MonitorUs –ի այլև ցանկացած կայքերի համար :Այն արդյունավետ է նաև տնտեսագիտական հաշվարկներից ելնելով :Այն արդյունավետ է օգտագործողների օգտագործման համար է և ունի զարգացման ընաթացք:Հետագայում եթե Google տրամադրի API մենք հնարավորություն կունենանք ինտեգրացված օրացույցում ներմուծել գործողություններ MonitorUs-ից:

Գրականության ցանկ

1. Д. Н. Колинсенченко “ Самоучитель PHP 5”