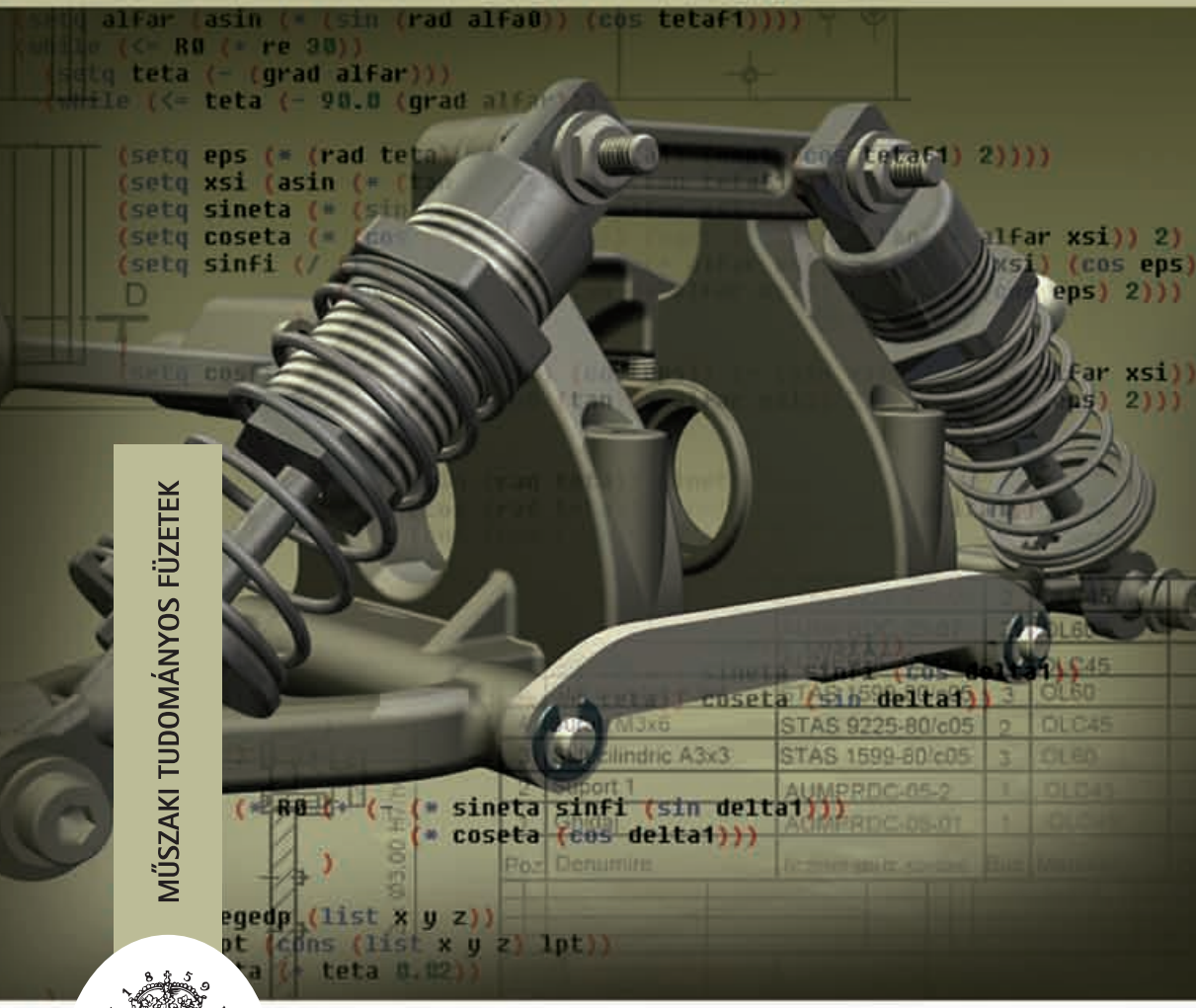
TOLVALY-ROȘCA FERENC

A SZÁMÍTÓGÉPES TERVEZÉS ALAPJAI

AUTOLISP ÉS AUTODESK INVENTOR ALAPISMERETEK



MŰSZAKI TUDOMÁNYOS FÜZETEK

ERDÉLYI MÚZEUM-EGYESÜLET

TOLVALY-ROŞCA FERENC

A SZÁMÍTÓGÉPES TERVEZÉS ALAPJAI
AUTOLISP ÉS AUTODESK INVENTOR ALAPISMERETEK

MŰSZAKI TUDOMÁNYOS FÜZETEK

7.

TOLVALY-ROŞCA FERENC

A SZÁMÍTÓGÉPES TERVEZÉS ALAPJAI

AUTOLISP ÉS AUTODESK INVENTOR ALAPISMERETEK

ERDÉLYI MÚZEUM-EGYESÜLET
Kolozsvár
2009

A könyv megjelenését támogatta:

a Szülőföld Alap

Lektor: Bitay Enikő

Kiadja: Az Erdélyi Múzeum-Egyesület
Felelős kiadó: Sipos Gábor
Sorozatszerkesztő: Bitay Enikő
Olvasószerkesztő: PONTLAB Kft.
Borítóterv: Könczey Elemér
A borító Autodesk grafikát is tartalmaz
Tördelés: PONTLAB Kft.

Nyomdai munkálatok:
Europrint Kft, Nagyvárad
Tel./Fax: +40-259-472631

Descrierea CIP a Bibliotecii Naţionale a României

TOLVALY-ROŞCA, FERENC
A számítógépes tervezés alapjai : AutoLisp és Autodesk Inventor alapismeretek / Tolvaly-Roşca Ferenc. - Cluj-Napoca : Societatea Muzeului Ardelean, 2009
Bibliogr.
ISBN 978-973-8231-81-8

004.42 CAD

Tartalom

Előszó

Ami 7–8 évvel ezelőtt valószínűtlennek tűnt, ma valóság: az ember már óvodás kortól nem csak játékra, hanem tanulásra használja a személyi számítógépet, felnőttként pedig mindennapos használati eszközként munkára vagy szórakozásra is. A személyi számítógépek ilyen méretű elterjedése, szinte a tudomány minden területére alkalmazhatóvá tette, napjainkban pedig kijelenthetjük, egyes ellenvélemények ellenére is, hogy nélkülözhetetlen a modern kutatás minden területén, a mérnöki tudományok terén pedig alapvető munkaeszköz.

A számítógépes tervezés (Computer Aided Design – a továbbiakban CAD), vagy más néven Számítógéppel Segített Tervezés, egy nagyon széles fogalomterület.

A könyv, amelyet az olvasó a kezében tart, a Sapientia Magyar Tudományegyetem mechatronika szakán oktatott Számítógépes Tervezés tantárgy előadásanyagát igyekszik írott formába önteni. Az itt bemutatott, illetve oktatott anyag csupán egy kis része a CAD-nek és a gépészetben alkalmazott számítógépes tervezésnek is.

Az előadás célja alapszinten megismertetni az Autodesk AutoCAD szoftvere alatt alkalmazható, felhasználói programozói felületek – jelen esetben az AutoLisp – által nyújtott lehetőségeket, valamint a parametrikus modellező programok közül, az Autodesk Inventor 2009-et.

Meg kell jegyeznünk, hogy az említett programok mellett, a szoftverek magas szintű alkalmazása is folyik az egyetemi oktatás minden szintjén, de nem kötelező tananyag formájában.

Igen sajátságos helyzetet jelent az a tény, hogy a romániai oktatásban használt licenszek kizárólag angol nyelvű szoftverre szólnak, a Sapientia Egyetemen az oktatás pedig csak magyar nyelven zajlik. Emiatt az anyag az angol nyelvű utasítások, parancsok mellett azoknak a magyar megfelelőjét is bemutatja, a parametrikus modellezéshez kapcsolódó fogalmakat pedig magyarul közli. Ebből kifolyólag helyenként kissé szokatlan az angol–magyar fogalmak látszólagos keveredése, de ez elkerülhetetlen az angol nyelvű szoftverek magyar nyelven történő oktatásánál.

A jelen könyv feladatokat nem tartalmaz, hanem gyakorlati példákon keresztül igyekszik a legrövidebb úton a legtöbb elméleti ismeretanyagot átadni az olvasóknak. Az Inventorban csupán a testmodellezés alapvető parancsait és síkrajzból csupán a nézetek készítésének alapvető lépéseit ismertetjük. A síkrajz teljes ismertetése, a szerelési modellezés, bemutatók készítése, az Inventor Studio, végeselem- és dinamikai analízis, az elkövetkező kötetek anyaga.

Az AutoLISP elsajátítása lehetetlen az AutoCAD alapos ismerete nélkül. Az első részben bemutatott AutoLISP ismeretek, számos hivatkozást tartalmaznak AutoCAD szakkifejezésekre. Ezért a szerző, e könyv forgatása előtt, az AutoCAD bármelyik változatának alapos elsajátítását ajánlja először.

A jelen könyvet a szerző elsősorban az oktatás támogatásának céljából állította össze, de oly módón, hogy azt bizalommal forgathatják, azok a legkülönbözőbb szakosítású mérnök szakemberek is, akik az ismereteiket szeretnék bővíteni a számítógépes tervezés ezen a két sajátságos területén.

Marosvásárhely, 2009. *A szerző*

1. AutoLISP

1.1. Az AutoCAD programozhatósága

Az AutoCAD egyik legrégebbi és legelterjedtebb számítógépes tervezői (CAD) program. Legelső megjelent változatától kezdve, a program fejlesztői igyekeztek lehetőséget adni a felhasználónak az alapprogram utasításainak bővítésére, a felhasználó által készített újabb utasítások használatának lehetségesé tételére. A CAD programoknál használt programozói felület általánosan **API** vagyis **Advanced Programming Interface** néven ismert.

Ezek közül a legelső, amelyet AutoCAD alatt használni lehetett az AutoLISP, ekkor még a program lehetőségei korlátozottak voltak, ennek felhasználói felülete a DOS operációs rendszer alatt nem volt, vagy csak elenyésző mértékben volt változtatható. Az új AutoCAD programok változatainak azonban már szinte minden aspektusa változtatható:

- ActiveX;
- Visual Basic;
- AutoLISP és VisualLisp;
- ObjectARX;
- Menük módosítása;
- DIESEL – szöveg feldolgozás, például állapotsor.

Az **Active X:** OLE 2.0-t használva, script állományok, makrók vagy saját alkalmazások készíthetők partner programoknak, magas szintű programnyelvek segítségével, mint Visual Basic 4.0 és C++. A következő három fogalom, alapfogalmak az Active X terminológiában:

- Azon AutoCAD objektumokat, amelyek elérhetők és módosíthatók az AutoCAD programban, Active X elemeknek nevezzük (vonalak, layerek, szövegstílus, block stb.);
- Az eljárások olyan függvények, amelyek hatnak egy Active X elemre;
- A tulajdonságok olyan függvények, amelyek egy Active X elem állapotáról adnak információkat,

Az **ObjectARX** a haladó felhasználóknak nyújt inkább programozási lehetőségeket:

- AutoCAD Runtime Extension;
- DLL (Dynamic Link Library) ;
- C és C++ nyelven új parancsokat lehet létrehozni;
- Közvetlenül az AutoCAD objektumokat kezeli.

Az AutoLISP alaposabb tanulmányozása előtt lássuk, néhány fontos tulajdonságát:

- AutoCAD rendszer hatékonyságát növelheti;
- Rutin feladatokat automatizálhat;
- Módosítani lehet rendszerváltozókat;
- Módosítani lehet rajzokat, rajzelemeket;
- AutoCAD legrégebbi programozói környezete;
- Sok AutoCAD parancs maga is AutoLISP függvény.

1.2. AutoLISP alapfogalmak

Az AutoLISP a LISP programozási nyelv AutoCAD-re alkalmazott változata. A LISP egyik legrégebbi magas szintű programozási nyelv (1960), amelynek több változata is létezik: Common LISP, Franz LISP, Mu LISP és X LISP.

Az AutoCAD programozására használt AutoLISP-nek nagyon sok vonzó tulajdonsága van mind a kezdők, mind a haladók szempontjából:

- Szintaxisa rendkívüli egyszerű és pontos. Könnyen megtanulható mind a Lisp interpretor kiértékelési alapja, mind a programnyelv szintaxisa.
- Interpretor típusú programozási nyelv, ami azt jelenti, hogy a megírt program interaktív módon az AutoCAD parancssorából kipróbálható;
- Az AutoLISP egyenesen az AutoCAD programra hat, egyéb külső – például kompilátor – segítsége nélkül;
- Segíti a programozót egyszerű függvények megírására, és ezek összefűzésével bonyolult**,** de könnyen érthető programok elkészítésében.
- Az AutoLISP kódprogram könnyen áttekinthető.
- Megengedi ciklusok, teszt és logikai függvények használatát.

Az AutoLISP az AutoCAD standard tartozéka, minden egyes változatában megtalálható, és minden új munkaszesszió indításakor automatikusan betöltődik. A programok a legegyszerűbb ASCII szövegszerkesztőben (például Notepad) megírhatók, de a bonyolult, legmodernebb programozási felületekre jellemző Visual Lisp-ben is, ami a 2000-es AutoCAD változattól a program standard tartozéka. Lévén egy interpretor típusú programozási nyelv, az AutoLISP kifejezések futtathatók az AutoCAD parancssorából.

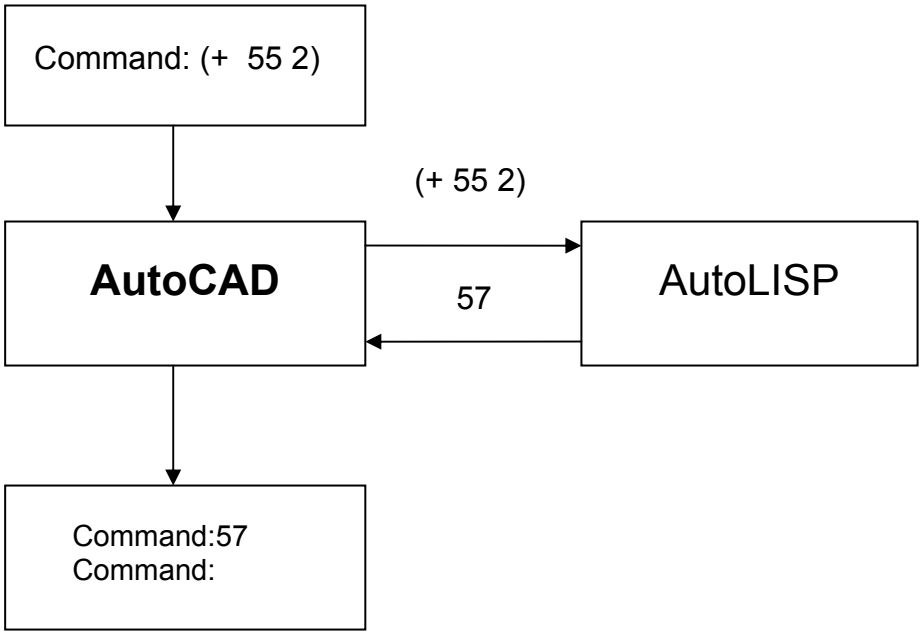
Ha az AutoCAD egy nyitott kerek zárójelet olvas be a parancssorból, akkor rögtön az utána következő karaktersorokat átadja kiértékelés végett az AutoLISP interpretornak. Ugyanaz történik felkiáltójel „!" használata esetén is.

Az AutoLISP utasítássor egy kerek zárójel-pár (zárójel-párok) közé írt karakterek sorozata. A zárójelben található karakterek sorozata az ún. szimbolikus kifejezésekből, vagy röviden s-kifejezésekből áll. A s-kifejezések az AutoLISP progra-

mok alap utasításai, alapvetően zárójelek közé zárt listák formájában használjuk őket. Amikor az AutoLISP s-kifejezésekre bukkan, azokat felméri és kiértékeli, a kiértékelés eredményét pedig visszafordítja az AutoCAD parancssorába.

A programozás során a leggyakrabban elkövetett hiba a zárójelek nem megfelelő számú párosítása: minden kinyitott zárójelet ugyanolyan számú bezárt zárójel kell, hogy kövessen, egyébként az interpretor hibaüzenetet ír ki: "extra right paren on input".

Az AutoLISP interpretor működését az **1.1. ábra** szemlélteti.

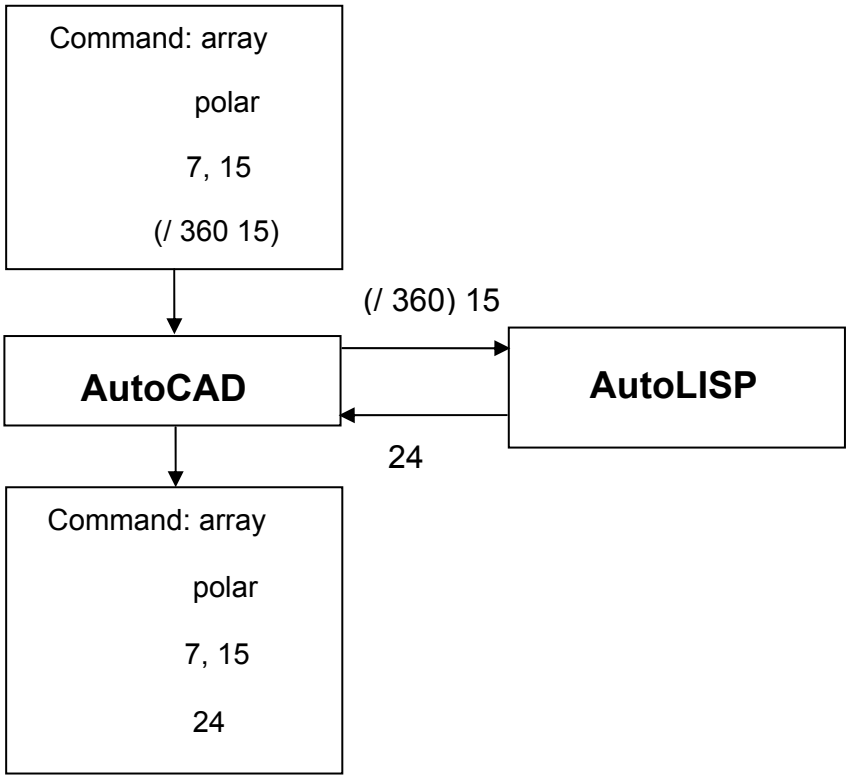


1.1. ábra
AutoLISP interpretor működése

Az **1.1. ábrán** megfigyelhető, hogy az AutoCAD parancssorba írt nyitott zárójel átadja az utána következő + 55 2 karaktereket az AutoLISP kiértékelőnek, kiértékelés után pedig, az AutoCAD parancssorban megjelenik a kiértékelés eredménye, az 57.

Az **1.2. ábrán** látható egy hasonló folyamat abban az esetben, ha a zárójel egy már futó AutoCAD parancsban jelenik meg. Ebben az esetben az array utasítás, egy opció – Rectangular/Polar – követ, polar opciót választva és beírva, a következő kérdés a körsokszorozás középpontját kéri. Ez megadva a 7, 15 koordinátájú pontokkal, az AutoCAD parancssorban szöveg kérdezi az elemek számát. Ha nem

tudjuk, csupán azt, hogy 15 fokonként vannak az elemek, használhatjuk az AutoLISP által nyújtott számítási lehetőséget: nyitott zárójellel meghívjuk az AutoLISP interpretort és beírjuk (/ 360 15). Zárójelet zárva, visszafordul a parancssorban a 360:15 = 24, és az AutoCAD parancssorban az elemek számát a 24 fogja jelenteni.



1.2. ábra
AutoLISP kiértékelési ciklus

A felkiáltójel használata szimbólumok és változok lekérdezésére alkalmas:

```
Command:!pi
3.14159
Command:
```

Atomok és listák

Két alapvető AutoLISP objektum létezik: **atom** és **lista**. Az atomok egyszerű-, a listák pedig komplex elemek. Ezek az AutoLISP **alapelemei.**

Alapelemek:

- Listák
  - Atomok sorozata zárójelek között
- Atomok
  - Definíció:
    - „minden ami nem lista”
    - a nyelv eszközeivel nem bontható tovább
  - Konstans atomok
  - Szimbólikus atomok
- Speciális atomok: nil, t

AZ ATOM ÉRTÉKE MAGA AZ ATOM.

Konstans atomok:
- Számok: egész (integer), valós (real) szám;
  1234, 56.78, 123.56e-3
- Karaktersorok (string), kettős aposztróf között;
  ”abcdefghijk...”
- Kiválasztási halmaz (selection set);
- Objektum név (entity name);
- Állomány-leíró (file descriptor).

Szimbolikus atomok:
- Szimbolikus atomok = Változók;
- A változó neve betűvel kezdődik, és bármilyen jel követheti, kivétel az elválasztó jelek;
  Példa: point1, point2, teljes hossz
- A változó egy értéket jelöl, tárol.

Speciális atomok:
- Értékét nem lehet/szabad felüldefiniálni;
- t – igaz értéket jelöli;
- nil – hamis érték és üres lista ();
  (atom amely lista is !!!)
- pi – 3.141592;
- pause – ”\\”.

A lista kerek zárójelek között elhelyezett, szóközökkel elválasztott atomok listája:

- A listák egymásba ágyazhatók;
- Egymásba ágyazás szintjeinek nincs korlátja;
- Üres lista: () vagy nil.

Példa:

```
(1 2 3 5)
(11 (2 3) (4 (5 6)))
(1 ”string” 1455.6001)
```

Ha a listákat hibásan írjuk be, hibaüzenetet térit vissza az AutoLISP.
Hibás listalehetőségek:

```
(ez (nincs befejezve)
(ez mar)(ket lista)
(tul (sok (a)) bezaro) zarojel)
)hibasan kezdodik a lista)
hianyzik a kezdo zarojel
(ez egy list (es egy)) atom
```

Lista típusok:

- Adatlista: (123 345.56 ”qwerty” 78e-6);
- Pontlista:
  2D pontlista: (0.0 1.0);
  3D: (12.3 45.6 0.0);
- Függvénylista:
  (+ 3 4);
- (setq qq 1.2).

Egyszerűen eldönthetjük, hogy egy elem atom vagy lista: ha egy elem nincs zárójelben, akkor atom, egyébként mindig lista.

Kiértékelés:

A kiertékelés egy folyamat, amely során egy AutoLISP kifejezés feldolgozásra kerül. AutoLISP egy interpretált nyelv. Az Interpretor „agyközpontja” az **evaluator** vagy magyarul a kiértékelő, értelmező. A felhasználó interaktív módon párbeszédet folytat az értelmezőprogrammal.

A kiértékelési ciklus folyamata:

1. Az értelmező program beolvas egy s-kifejezést
2. A kifejezést kiértékeli

- meghatározza a kifejezés értékét
- mintha függvény lenne
- lehetnek mellékhatások

3. A kifejezés értékét kiírja, majd kezdi az 1. ponttól

A három lépésnek megfelel egy-egy LISP függvény: READ, EVAL, PRINT. Ezért nevezik még ezt a végtelen ciklust: READ–EVAL–PRINT ciklusnak.

Ugyanakkor a beolvasás előtt még kiírja a prompt-ot (parancssort):

```
Command:
```

Atomok kiértékelése

Az atomok egyszerű elemek, legtöbbször a saját értékükre vannak kiértékelve. Ezek közül kivétel a szimbólum, amelyet a legutoljára hozzárendelt értékre értékel ki az Interpretor. Például ha az xcb szimbólumhoz legutoljára a 2.25553 valós számot kötöttük, akkor az xcb szimbólum kiértékelése a 2.2553.

Listák kiértékelése

MINDEN KIÉRTÉKELENDŐ LISTA ELSŐ ELEME EGY FÜGGVÉNY KELL LEGYEN!

Az AutoLISP **első lépésben** minden lista első elemét elemzi, ha ez nem **egy függvény**, hibaüzenetet küld. A lista függvény után következő elemei a függvény **argumentumai.**

A **második lépésben** az AutoLISP a függvényt értékeli ki, mint utasítássorozatok egymásutániságát. Például a + függvény kiértékelését szövegben így írhatnánk le: **„**Keresd meg a lista argumentumaiként szereplő többi elem értékeit. Mikor megvannak, add őket össze és fordítsd vissza az összeadás eredményét, mint a lista kiértékelt végeredményét.”

Harmadik lépésként kiértékeli az első argumentumot, ideiglenesen tárolva annak az értékét. Megkeresi, hogy létezik-e még argumentum a listában, ha igen kiértékeli azt is, és újra keresi, hogy létezik-e még argumentum, mindig kiértékelve azt és ideiglenesen tárolva azokat az értékeket.

Negyedik lépésben, ha már nincs kiértékelendő argumentum, az AutoLISP interpretor befejezi a függvény utasítássorozatát, például egy összeadással, a + függvény esetében, és visszafordítja annak az értékét.

Nézzük példaként egy bonyolultabb lista kiértékelési ciklusát:

```
(+ 2 (+ 2 3))
```

Első ránézésre azt mondanánk, hogy valószínűleg belülről fog kifele értékelni az interpretor. Tévedés!

Az AutoLISP Interpretor kivétel nélkül mindig balról jobbra végzi el az értékelést!

Első lépésként egy lista értékelésénél **mindig egy függvényt** keres: (+ ... A + függvény utasítássorozatát követve keresi az argumentumokat, sorjában kiértékelve azokat. Az első argumentum az a 2-es atom. Ezt kiértékeli és ideiglenesen tárolja. Keresi a harmadik elemet, amely egy újabb argumentum, a mi esetünkben egy lista. Mit tesz egy listával? Keresi az első elem formájában a függvényt, amelynek az utasítássorozatát az előzővel azonos módon négy lépésben végrehajt, végeredményként visszafordítva az 5-öt és ideiglenesen tárolva azt, mint a lista második argumentumát. Nem találván újabb argumentumokat, végrehajtja a kiértékelési ciklus negyedik lépését, visszafordítva az argumentumok összegét, a 7-et. Egy kiértékelhető lista formája, mindig az alábbi kell legyen:

(fuggveny argumentum1 argumentum2 ... argumentumn)

Köztelező módon mindig egy függvénnyel kezdődik, ha nem, akkor az AutoLISP hibaüzenetet ad, tudatva, hogy az első elem nem függvény, megállítva a kiértékelést.

Nil kiértékelése

A nil az AutoLISP egyik kivételes eleme. Amíg az elemek vagy atomok, vagy listák, a nil lehet mindkettő együtt, atom és lista is. Listaként kifejezhető úgy, mint () – üres lista. Logikai kiértékelésben pedig hamisat jelent.

A kiértékelés megállítása

A **quote** függvény egyetlen argumentumot enged meg, ezt kiértékeletlenül fordítja vissza. Használata: olyan esetekben, amikor egy atom vagy lista kiértékelését meg akarjuk állítani.
Példa:

```
Command:(quote (1.1 2.2 3.3))
(1.1 2.2 3.3)).
```

Természetesen észrevehető, hogy a quote függvény nélkül az AutoLISP hibát üzenne, mivel a második listában nem létezik egy függvény sem.

A quote helyett a „‘” jel is használható:

```
Command:'(1.1 2.2 3.3)
(1.1 2.2 3.3)
```

A hozzárendelés

A hozzárendelés vagy hozzákötés az a specifikus AutoLISP folyamat, amely során egy szimbolikus atomhoz – változóhoz – egy értéket kötünk hozzá. Ez az érték-hozzárendelés ideiglenes, adott pillanatban ugyanahhoz a változóhoz egy más érték köthető hozzá. A változó típusát így nem szükséges meghatározni.

A hozzárendelés a **setq** függvénnyel végezhető el.

Használata:

(setq szimb1 kifej1 szimb2 kifej2 .... szimbn kifejn)

Ahol a szimb1 ... szimbn változók, a kifej1... kifejn pedig atomok, vagy listák.
Példa:

```
Command:(setq szimbolum 78)
78
Command:!szimbolum
78
Command:(setq szimbolum "karaktersor")
"karaktersor"
Command:!szimbolum
"karaktersor"
```

Megfigyelhető, hogy a **szimbolum** nevű változóhoz először hozzákötöttük a 78 egész számot. Visszakérdezve a felkiáltójellel, visszafordítja a 78-at. Továbbiakban ugyanazon változóhoz hozzákötjük a "karaktersor" karaktersor típusú adatot, a továbbiakban visszakérdezve a felkiáltójellel a "karaktersor"-t fordítja vissza.

Számadatok típusai

Két alapvető számadattípus létezik az AutoLISP-ben: **egész számok** és **valós számok**. A számadatok típusát egyszerűen le is kérdezhetjük a **type** függvény segítségével:

```
Command:(type 1.0)
REAL
```

A valós számok úgy ismerhetők fel, hogy tizedespontot tartalmaznak. Az AutoCAD-el ellentétben AutoLISP-en a –1 és +1 közötti valós számok mindig kell tartalmazzák a tizedespont előtt a 0-t is: –0.025, +0.369.

```
Command:(type 5)
INT
```

Az egész számok nem tartalmaznak tizedespontot. Értékük -2 147 483 648 és +2 147 483 647 között lehet. Belső rendszerében az AutoLISP 32 biten használja az egész számokat, az AutoCAD-el való kapcsolata során csupán 16 biten. Emiatt az AutoCAD-be nem vihetők át egész számok csak –32 767 és + 32 767 között. Ha az intervallumon kívüli számokat akarunk használni, vagy valós formában használjuk, vagy a számítások során nyert értékeket az átvitel előtt átalakítjuk valósakká a **float** függvénnyel. Ha egy lista argumentumai csak egész számok, akkor a visszafordított kiértékelés is egész szám lesz. Ha legalább az egyik argumentum valós szám, akkor a visszafordított érték valós lesz.

A kijelzések pontossága és a belső pontosság

AutoLISP-ben a kifejezés implicit pontossága 5 tizedes. A belső pontossága ellenben legkevesebb 14 tizedes pontosságú. Ha egy valós szám kijelzése nagyobb pontossággal szükséges, mint az implicit 5, akkor azt előbb át kell alakítani karaktersorrá, majd a megfelelő pontossággal azt kiíratni.

Az **rtos** függvény három argumentumot enged meg, amely egy valós szám, egy AutoCAD által felismert egység és a tizedes számok számát jelentik. Neve az angol **real to string** szóösszetételből származik. Szintaxisa:

(rtos <szam> [mod] <pontossag>)

A **mod**, a **pontossag**, valamint az AutoCAD UNITMODE rendszerváltozójának és DIMZIN méretváltozójának beállításától függ. A mod és a pontossag egész számok, amelyek a választott hosszmértékegységet és a pontosságot határozzák meg.

Az értelmezhető mód beállítások a következők:

Mód értéke	Karaktersor formátuma
1	Tudományos
2	Tizedes
3	Mérnöki (láb és tizedes hüvelyk)
4	Építészeti (láb és a hüvelyk törtrésze)
5	Tört

Példa:

```
Command: (rtos 10 2 10)
"10.0000000000"
```

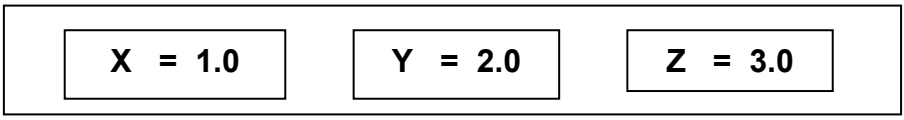
A gyakorlatban szinte kizárólag a mód 2-es beállítását használjuk, amint azt a fenti példa is mutatja.

1.3. AutoCAD pontok AutoLISP-ben

1.3.1. Az AutoCAD pontok tárolási formája

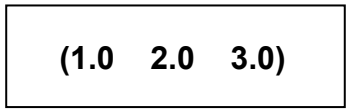
Az AutoCAD program a saját adatbázisában Descartes-koordináta rendszert használ a két- és háromdimenziós pontok leírására. Az AutoLISP az AutoCAD pontot egy valós számokból álló listával írja le.

Egy háromdimenziós AutoCAD pont három egymástól független számból áll, amelyek az X, Y és Z tengelyre vetített távolságok. Az AutoCAD-ben nagyon sokféleképpen adható meg egy pont koordinátája: abszolút-, relatív-, vagy poláris-koordinátákban, vagy akár object snap formájában is. Az AutoCAD adatbázisában azonban csupán három szám formájában van tárolva. Az 1, 2, 3 koordinátájú pontot az **1.3. ábra** szemlélteti.



1.3. ábra
Az 1, 2, 3 koordinátájú pont

Ez a pont AutoLISP-ben három valós számból álló lista, ahol az első elem az X koordináta, a második az Y és a harmadik a Z (**1.4. ábra**).



1.4. ábra
Az 1, 2, 3 koordinátájú pont AutoLISP lista formájában

Lista készítéséhez a **list** vagy **quote** függvényt használhatjuk. Példaként lássuk ezek használatát az előbb szemléltetett pont esetében:

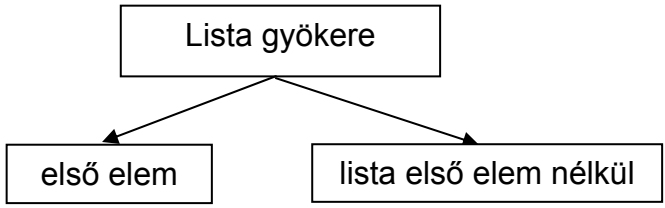
```
Command:(quote (1.0 2.0 3.0))
(1.0 2.0 3.0)
```

A list függvény estében a lista argumentumaiból egy lista lesz a kiértékelés eredménye.

```
Command:(list 1.0 2.0 3.0)
(1.0 2.0 3.0).
```

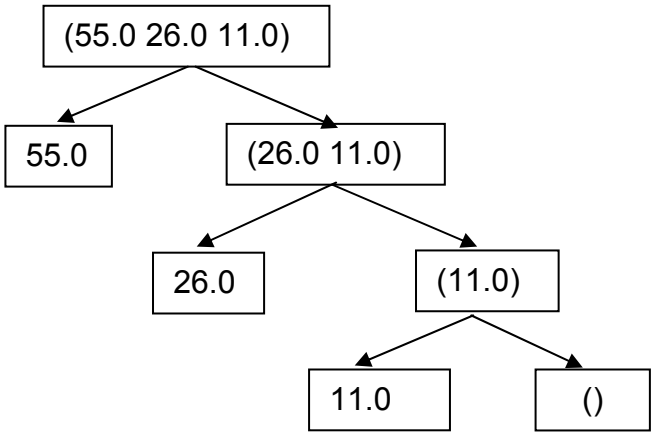
1.3.2. A listák elemeihez való hozzáférés

Egy AutoLISP lista a számítógép memóriájában bináris faként van jelen. Ennek a bináris fának bármely ágazata egy csomópont. A bináris fa csúcsa a gyökér-csomópont. Innen kezdve az AutoLISP lista struktúrája csomópontonként szétszedhető. Alapvetően minden csomópont két ágra bomlik: a lista első eleme és a lista az első elem nélkül (**1.5. ábra**).



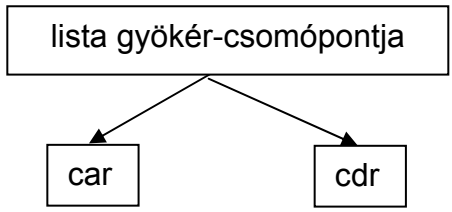
1.5. ábra
A gyökér-csomópont és bináris fa első elagazása

Példaként az **1.6. ábrán** nézzük meg az (55.0 26.0 11.0) lista teljes leágazását a bináris fában.



1.6. ábra
Példa egy lista lebontására

Az AutoLISP a bináris ágazat mindkét elemét elérheti a **car** és a **cdr** függvények segítségével (**1.7. ábra**).



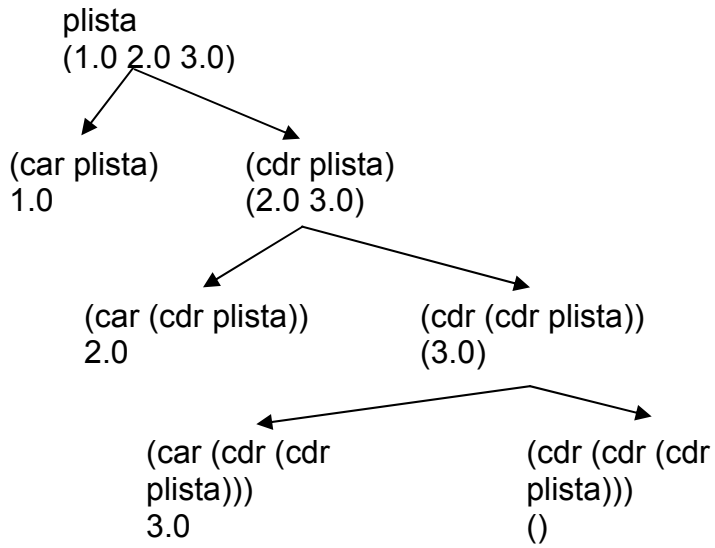
1.7. ábra
A car és cdr függvények használata egy lista bináris ágainak elérésére

A **car** függvény egy lista első elemét fordítja vissza kiértékelésként:

```
Command:(car (list 2.0 1.0 3.0))
2.0
```

A **cdr** függvény egy listából visszafordítja listát az első elem nélkül.

```
Command:(cdr (list 2.0 1.0 3.0))
(1.0 3.0).
```



1.8. ábra
Egy AutoCAD pont koordinátáihoz való hozzáférés

Mindkét függvény az argumentumban szereplő listát vagy a változót, amihez egy listát kötöttünk, nem bontja le véglegesen, csupán visszatéríti a már említett részeket.

Egy AutoCAD pont X, Y, Z koordinátáihoz való hozzáférést az **1.8. ábra** szemlélteti.

1.4. AutoLISP és az AutoCAD utasítások

Mint már említettük az AutoLISP egyik óriási előnye, hogy interpretor típusú programozási nyelv. Interaktív módon meghívható a kiértékelési modul bármely AutoCAD utasítás közben, vagy beírva az AutoLISP utasítást, az rögtön látható az AutoCAD programban. Nagyon fontos, hogy az AutoLISP programon belül lehetőség van bármely AutoCAD utasítást is használni! A **command** függvény, amelynek bármennyi argumentuma lehet, „tudatja" az AutoCAD-el, hogy az argumentumaiban levő kifejezések AutoCAD utasítások. Ekképpen a command függvény argumentumai karaktersorok és változok lesznek. Például:

```
Command:(command "line" "1, 1" "5, 5" "")
Command:nil
```

A kiértékelés eredménye egy egyenes lesz az 1, 1 és 5, 5 koordinátájú pontok között. A nil visszafordítása azt jelenti, hogy az AutoLISP egy AutoCAD utasítással fejezte be, és ennek a visszafordított kiértékelése nem lehet más, mint nil. A legutolsó argumentum, a "" egy ENTER-nek felel meg.

Vagy:

```
Command:(setq pont1 (list 1.0 1.0 1.0)
              pont2 (list 5.0 5.0 5.0))
Command:(command "line" pont1 pont2 "")
Command:nil
```

A command függvény használata rendkívül egyszerű: bármely AutoCAD utasítás beprogramozható egy AutoLISP programsorba, ha először kipróbáljuk magát az utasítást és a parancsban levő, illetve beírandó szöveget karaktersorként, vagy az értékeket számadatként a command függvény argumentumaiként írjuk be.

A **pause** szimbólum lehetőséget ad a felhasználónak egy AutoCAD kérés futtatására. Használata kizárólag a command függvényen belül történhet csak.

1.5. Interaktív adatbevitel AutoLISP alatt

Az AutoLISP-nek léteznek olyan függvényei, amelyek lehetővé teszik a kiértékelés megállítását és megvárják egy vagy több felhasználói adatbevitelét (egész szám, valós szám, pont stb.), majd utána folytatódik a kiértékelés. Ezek a függvények a szóképzés szempontjából előtagként a **get-** szótaggal kezdődnek és utótagként a bekért adat típusával: **getpoint, getreal, getdist, getstring** stb. Ebben a fejezetben a **getpoint** és a **getdist** függvények használatát mutatjuk be, a többi függvényt a *„Gyakran használt AutoLISP függvények”* című fejezetben (**1.17. fejezet**) mutatjuk be.

A **getpoint** függvény lehetővé teszi a kiértékelés megállítása mellett egy AutoCAD pont valamilyen megengedett formában való kijelölését, bevitelét. Példaként:

```
Command:(getpoint)
1,1,2
(1.0 2.0 3.0)
```

A **getdist** függvény, két AutoCAD pont által meghatározott távolság bevitelét teszi lehetővé. Opcionálisan megenged egy karaktersort is argumentumként. Példa:

```
Command:(setq tav1 (getdist))
1,1
Second point:2,2
1.141421
Command:!tav1
1.41421
```

opcionális karaktersorral pedig:

```
Command:(getpoint "Kerem az elso pontot:")
Kerem az elso pontot:1,1
(1.0 1.0 0.0)
```

1.6. Saját AutoLISP függvények létrehozása

A létező belső AutoLISP függvények mellett lehetőség van saját, a felhasználó által definiált függvények elkészítésére, amelyben a már létező belső vagy előzőleg létrehozott saját függvények is használhatók. A **defun** függvény szintaxisa a következő:

(defun <szimb> (<arg lista> / <helyi valtozok> <kifejezesek…>

A **defun** függvény egy **szimb** nevű függvényt definiál (nem szabad közvetlenül idézőjelbe tenni). A függvény nevét az argumentumok listája követi (amely kitöltetlen is lehet), majd ezt opcionálisan egy törtjel után a függvény egy vagy több lokális szimbólum követheti. A törtjelet, az első lokális szimbólumtól és az utolsó argumentumtól legalább egy-egy szóközzel el kell különíteni. Ha a függvénynek nem adunk meg sem argumentumot, sem lokális szimbólumot, akkor a függvény nevét egy üres zárójelpárnak kell követnie.

Az alábbi példák az argumentumlisták használatának lehetséges helyes és egy nem megengedett változatát szemléltetik:

```
(defun fug (x y)...)          két argumentuma
(defun fug (/ a b)...)        két lokális szimbóluma
(defun fug (x / ideigl)...)egy argumentum és egy lokális szimbólum
(defun fug ()...)             sem argumentum, sem lokális szimbólum
```

Azonos névvel nem definiálhat több argumentumot egy függvényhez, azonban egy lokális változó neve azonos lehet egy másik lokális változó vagy egy argumentum nevével, mint például:

```
(defun fubar ( a a  / b ). . . )        nem megengedett
(defun fubar ( a b / a a b ). . . )     hibátlan
```

Az argumentumok és lokális szimbólumok listáját a függvény végrehajtása során kiértékelésre váró egy vagy több kifejezés követi. Ha az argumentumok/szimbólumok listában ugyanaz a tétel többször is szerepel, akkor az AutoLISP minden név első előfordulását használja fel, a továbbiakat pedig figyelmen kívül hagyja. A **defun** függvény az általa definiált függvény nevével tér vissza. Az így definiált függvény hívásakor az AutoLISP kiértékeli az argumentumait, és hozzárendeli az argumentumokat a szimbólumokhoz. A függvényen belüli lokális szimbólumok használata a külső szinteken hozzájuk tartozó értékeket nem befolyásolja. A függvény az utolsóként kiértékelt kifejezés eredményével tér vissza. A függvény minden előző kifejezésének csak mellékhatása van. Maga a defun függvény a definiált függvény nevével tér vissza.

Az alábbi példa új függvényt határoz meg a defun függvény segítségével, és bemutatja az új függvények által visszaadott értékeket:

```
(defun minusz20 (x)
    (- 20 x)
)                   eredménye MINUSZ 20
minusz20 10)        eredménye -10
minusz20 72.5)      eredménye 52.5.
```

Soha ne használjuk beépített függvények vagy szimbólumok nevét **szimb** elnevezésként, mivel ebben az esetben a beépített függvény többé nem lesz elérhető. A beépített, valamint az előzőleg definiált függvények listáját az **atoms-family** függvény segítségével kaphatjuk meg.

A defun egy speciális esete az, amikor a létrehozott függvényt használni lehet, mint bármilyen AutoCAD utasítást, zárójel használata nélkül. Az új saját parancsfüggvény úgy hozható létre, hogy a definiálás során a függvény neve elé egy **c:** jelet teszünk. Ebben az esetben a létrehozandó függvény nem használhat argumentumokat. Példa:

```
(defun c:bre ()
      (setq bp (getpoint „Valassza ki a torespontot !”))
      (command “Break” bp „@”)
)
Command:bre
Valassza ki a torespontot!
Command:nil
```

Megfigyelhető, hogy a parancsfüggvényt úgy kell meghívni, mint bármely AutoCAD utasítást! A parancssorba egyszerűen irjuk be a függvény nevét és adjunk egy Entert!

Helyi és globális változók

Az AutoLISP programok használata során gyakori eset, amikor ugyanazt a függvényt egy AutoCAD szesszió alatt többször is használni kívánjuk. Ilyenkor a setqval megkötött változók, ha nem voltak a saját függvény definiálásakor helyi változóként meghatározva, akkor mindig az előző programfuttatás során hozzákötött utolsó értékkel indulnak. Legtöbb esetben ez súlyos hibákhoz vezet. A **defun** szintaxisában található “**helyivaltozok**” éppen ezt a célt szolgálják: a saját függvény létrehozásakor az itt feltüntetett változók minden futtatás után visszanyerik azelőtti értéküket. Az összes többi változó, amely kiértékelésre kerül, az utolsó hozzákötött adattal lép ki a saját függvény futtatása után.

A globális szimbólumokhoz bármelyik függvény hozzáférhet, illetve azokat módosíthatja. A globális szimbólumok továbbá bármely kifejezésben felhasználhatók. A lokális szimbólumok és az argumentumok csak az őket definiáló függvény (és az ezen függvényből hívott függvények) kiértékelése során bírnak jelentéssel.

A függvények argumentumai lokális argumentum módjára viselkednek: a függvény megváltoztathatja értéküket, de a függvényből történő kilépést követően ezek a változtatások elvesznek.

Példa:

```
(setq helyivaltozo 10)
(defun fuggveny (/ helyivaltozo)
      (setq helyivaltozo 3)
      (setq valtozo (+ helyivaltozo 3))
)
Command:6
Command:!valtozo
Command:6
Command:helyivaltozo
Command:10
```

1.7. Listakezelő függvények

Ebben az fejezetben néhány alapvető listakezelő függvény használatát ismerhetjük meg.

A **length** függvény egy lista elemeinek számát fordítja vissza egész szám formájában. Szintaxisa:

(**length <lista>).**

Példa:

```
Command:(setq lista1 (list 1 jj lp "piros" 0.1)
Command:(length lista1)
6
```

Az **nth** függvény egy lista n-ik elemét téríti vissza kiértékelésként. Szintaxisa:

(nth <arg> <lista>)

ahól az **arg** egy egész szám **0 és n-1 között**, ha a lista hossza **n**. Példa:

```
(setq lista2 (list "ttt" 78 56 8.2))
(nth 0 lista2)
"ttt"
(nth 2 lista2)
56.
```

Az **apply** függvény átad egy argumentumlistát egy függvénynek. Szintaxisa:

(apply <fuggveny> <lista>).

Példa:

```
(apply  '+  '(1 2 3))
9.
```

A **mapcar** függvény a **fuggveny** kiértékelésének eredményével tér vissza, oly módon, hogy a listák elemeit a lista*1*-től a lista*n*-ig egyenként a függvény argumentumaiként kezeli.

(mapcar <fuggveny> <lista1>... <listan>)

A listák számának egyeznie kell a **fuggveny** által igényelt argumentumok számával. Példa:

```
(setq a 1 b 2 c 30)
(mapcar '1+ (list a b c))
(2 3 31).
```

1.8. AutoLISP program-állományok

Eddig azt láthattuk, hogy az AutoCAD parancssorból hogyan lehet AutoLISP utasításokat futtatni. Komplex kifejezéseket tartalmazó függvények esetén ez a módszer nem használható, ilyenkor egy szövegszerkesztő használata ajánlott, a létrejött állományt pedig .lsp kiterjesztéssel mentjük el. A szövegszerkesztőben megírt függvényt lehet utólag szerkeszteni, javítani.

Az **.lsp** kiterjesztésű állományokat be lehet tölteni az AutoCAD program alá az **appload** utasítással. Miután betöltöttük az állományt, a benne található függvények futtathatók a már ismert módon. A gyakran használt programok betölthetők egy speciális állományba az acad.lsp nevű állományba. Ez minden AutoCAD szesszió során automatikusan betöltődik és a benne található függvények bármikor indíthatók.

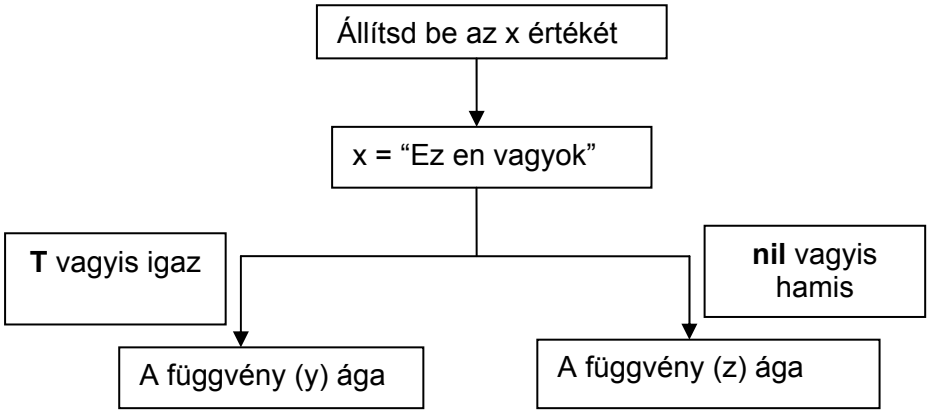
Az AutoLISP programok szerkeszthetők, futtathatók és kereshetők a hibák, az AutoCAD-ben megtalálható **VisualLISP** szerkesztőben. Ebben a szerkesztőben magas nyelvű programozásra jellemző programozó-felületen soronként futtathatók a bonyolult függvények, megállítható a kiértékelés, vizuálisan ellenőrizhető az egyes változó pillanatnyi értéke, szintaxisellenőrzés és hibakeresés végezhető.

Megjegyzés: az AutoLISP programokon belül megengedett a kommentárok használata.

A „;” jelet használva bármilyen magyarázatot a programba lehet írni. Ha egy sorba „;” jelet írunk (abban a sorban) az utána következő jeleket, karaktereket az AutoLISP nem veszi figyelembe, és így nem kerülnek kiértékelésre.

1.9. Logikai és egyenlőségi tesztek

A feltételes kifejezések ellenőrzik egy AutoLISP program futását. Az **1.9. ábra** egy AutoLISP program elágazást szemléltet.



1.9. ábra
Egy AutLISP programelágazás

A fenti diagrammban, ha az x változó értéke az “Ez en vagyok” karaktersor, akkor a program az y ágon, ellenkező esetben a z ágon fog továbbmenni.

Az **if és cond** függvények feltételes tesztfüggvények, amelyek az AutoLISP programelágazások kezelésére szolgálnak.

Az if függvény működését az alábbiakkal szemléltethetjük:

```
(if
(visszafordítok T)
      (akkor tedd ezt)
      (ha nem akkor tedd ezt)
)
```

Az if függvény szintaxisa:

(if <feltetel-kif> [akkor-kif] [egyebkent-kif])

Ez a függvény feltételesen értékeli ki a kifejezéseket. Ha a **feltetel-kif** értéke nem **nil**, akkor az **akkor-kif** kifejezés kerül kiértékelésre, ha igen, akkor **az egyebkent-kif** kifejezés. Az utolsó argumentum (egyebkent-kif) el is hagyható. Az if függ-

vény a kiválasztott kifejezés értékével tér vissza; amennyiben az egyebkent-kif hiányzik és a feltetel-kif értéke nil, az if függvény nil értékkel tér vissza. Példaként:

```
(if (= 77 3) "IGEN!!" "nem.")     eredménye "nem."
(if (= 6 (+ 4 2)) "IGEN!:")       eredménye "IGEN"
(if (= 1 (+ 9 4)) "IGEN:!")       eredménye nil.
```

Mivel az if függvénynek csupán három argumentuma lehet, amiből az egyik éppen a tesztkifejezés, a második a T állapotra kiértékelt argumentum, a harmadik pedig a nil logikai állapotra kiértékelt argumentum, több argumentum kiértékelése lehetetlen. Erre használjuk a **progn** függvényt, amely úgymond egy argumentum alá gyűjti az egyes logikai elágazás alá kerülő, akármilyen nagyszámú kifejezést. Például:

```
(if (> 3 2)
      (progn
            (setq wer 12)
            (command „line" pt1 pt2)
            (command „zoom" „e" „")
      )
      (setq wer 789)
)
```

A **cond** függvény az AutoLISP programnyelv elsődleges feltételes függvénye. Szintaxisa:

(cond (<feltetel1> <eredmeny1>. . .) (<feltetel2> <eredmeny2>...) . . .)

Ez a függvény argumentumként tetszőleges számú listát fogad el. Minden lista első elemét kiértékeli (a megadott sorrendben), egészen addig, amíg valamelyik értéke nem különbözik a nil értéktől. Ekkor kiértékeli a listában a teljesülő feltételt követő kifejezéseket, és az allista utolsó kifejezésének értékével tér vissza. Amennyiben az allistában csak egyetlen kifejezés szerepel (azaz az **eredmeny** hiányzik), a függvény a feltétel kiértékelésének eredményével tér vissza. A **cond** függvény esetszétválasztásos (case típusú) függvényként használható. Általános gyakorlat a T használata utolsó (alapértelmezés szerinti) **feltetel** kifejezésként.

Nézzünk egy példát. Legyen adott az s-szimbólumban egy, a felhasználótól válaszként kapott karakterlánc. Az alábbi függvény leellenőrzi a felhasználó által adott választ, és 1-gyel tér vissza, ha a karakterlánc I vagy i, illetve 0-val, ha a válasz N vagy n; egyéb esetekben a függvény eredménye nil.
Például:

```
(cond ((= s "I") 1)
      ((= s "i") 1)
      ((= s "N") 0)
      ((= s "n") 0)
      (T nil)
)
```

1.10. Ciklusszervezés AutoLISP-ben

A ciklusok szervezése bármely programozási nyelv egyik elengedhetetlen eszköze. A ciklusok két részből állnak: a ciklusfejből és ciklusmagból. A ciklusfej tartalmazza a függvény nevét és a ciklus működésének, illetve befejezésének feltételét. A ciklusmag azon kifejezések összessége, amelyekre a ciklusban ismételten visszatérünk. A ciklusban módosítjuk a kifejezésben szereplő változók értékeit, és ugyanazokat a műveleteket hajtjuk végre ezekkel a módosított értékekkel. Ciklusszervező függvényekkel előtesztelő ciklusokat készíthetünk. Ez azt jelenti, ha a ciklusfüggvény kerül kiértékelésre, az első lépés egy feltételvizsgálat lesz, hogy teljesülnek-e a ciklusmagban lévő kifejezések kiértékelésének feltételei. Ha igen, akkor megtörténik a ciklusmagban található kifejezések kiértékelése. Ha pedig nem, akkor a ciklusmagot kihagyja és folytatja a következő kifejezés kiértékelésével.

A **repeat, while** és a **foreach** az AutoLISP ciklusfüggvényei. A három függvény között lényeges különbség van tartalmilag és formailag is.

A legegyszerűbb a **repeat** függvény:

(repeat <hanyszor> <kifejezesek…>).

A függvény annyiszor ismétli a kifejezéssorokat, amennyi a **hanyszor** argumentum (amely egy egész szám vagy egy egész számra értékelendő kifejezés) értéke.

Egy lista minden elemének kiértékelését egy kifejezésre, a **foreach** függvény segítségével tehetjük meg.

(foreach <nev> <lista> <kifejezes…..>)

Ez a függvény a listán végighaladva, annak minden elemét hozzárendeli a **nev**hez, és a listában lévő minden elem esetére mindegyik kifejezést kiértékeli. A függvényhívásban tetszőleges számú kifejezés adható meg. A **foreach** függvény az utolsó kifejezés kiértékelésének eredményével tér vissza. Például:

```
(foreach lovak '("EZ" "EGY" "PELDA") (print lovak))
"EZ"
"EGY"
"PELDA" "PELDA"
```

A **foreach** függvény csak az utoljára kiértékelt kifejezés eredményével tér vissza, ezért jelenik meg a kiírás után még egyszer a “PELDA” karaktersor.

A harmadik ciklusfüggvény a **while**, a legkomplexebb és egyben egyik leghasználtabb is:

(while <feltetel> < kifejezesek. . .>)

Ez a függvény kiértékeli a **feltetel** kifejezést, és ha értéke nem nil, akkor a többi **kifejezesek**-et is kiértékeli, majd újból kiértékeli a feltetel kifejezést. Ezt addig folytatja, amíg a feltetel értéke nem nil. A while függvény az utolsó kifejezés legutolsó értékével tér vissza. Például:

```
(setq x 1)
(while (<= x 5)
      (sajatfuggv x)
      (setq x (1+ x))
)
```

Ebben az esetben a ciklus öt alkalommal hívná meg a felhasználó által definiált **sajatfuggv** függvényt, a felt 1-től 5-ig terjedő értékeivel. Ezt követően az utolsó kiértékelt kifejezés értékével, 6-tal térne vissza.

1.11. Karaktersorok kezelése és kiírása

AutoLISP alatt bármilyen idézőjelbe “” tett jelek karaktersort jelentenek. Ezek akár *, ^, % vagy $ jelek is lehetnek. A karaktersorok kiértékelése saját maguk. Példa:

```
Command:!”This is a string”
 ”This is a string”
```

Egy karaktersorban, a **** jel egy kontroll karakter. Az alábbi táblázat az AutoLISP-ben felismert kontrollkarakterek listáját tartalmazza:

Kód	Jelentés
****	A \ karakter
*****	A ***** karakter
\e	Az **Esc** karakter
\n	A **Newline** karakter
\r	A **Return** karakter
\t	A **Tab** karakter
\nnn	Az nnn oktál kodú karakter

A **strcat** függvény, karaktersorok összevonására szolgál. Szintaxisa:

(strcat <arg1> <arg2>..... <argn>)

Az argumentumok száma bármennyi lehet, és mindegyikük karaktersor. Példa:

```
Command:(stracat "Ez" "egy" "osszevont" "karakter"
"sor")
"Ezegyosszevontkaraktersor".
```

Az AutoLISP-ben hat kiíró függvény létezik. Öt belőlük karaktersorokat ír ki, egy pedig karaktereket. Három bármilyen kifejezést kiír. Öt függvény pedig vagy képernyőre ír ki, vagy egy állományba.

A **prin1** függvény megjeleníti a kifejezés értékét a képernyőn, és ugyanezzel az értékkel tér vissza.

(prin1 <kifejezes <fajlleiro>>)

A **kifejezés** argumentum tetszőleges kifejezés lehet, nem szükséges, hogy karaktersor legyen. Ha a **fajlleiro** is szerepel az argumentumok között (és ez egy írásra megnyitott állományra vonatkozik), akkor a függvény a kifejezés értékét az állományba írja, pontosan úgy, ahogyan az a képernyőn jelent volna meg. A függvény csak az adott kifejezést jeleníti meg, szóköz vagy újsor karaktereket nem illeszt hozzá.

A **princ** függvény megegyezik a prin1 függvénnyel, de a kifejezésben lévő vezérlőkaraktereket kibővítés nélkül jeleníti meg.

(princ <kifejezes <fajlleiro>>)

A prin1 arra szolgál, hogy a **load** függvénnyel kompatibilis módon jelenítse meg a kifejezéseket, míg a **princ** olyan függvények számára szolgáltat olvasható kifejezéseket, mint például a **read-line**.

A **print** függvény megegyezik a prin1 függvénnyel, azzal a különbséggel, hogy a **kifejezes** elé egy újsor karaktert illeszt, a kifejezés mögé pedig egy szóközt.

(print <kifejezés <fajlleiro>>)

Ha egy egyszerű üzenetet akarunk megjeleníteni a képernyőn, akkor a **prompt** függvényt használjuk.

(prompt <uzenet>)

Ez a függvény az **uzenet**-et megjeleníti a képernyő promptterületén, és nil értékkel tér vissza. Az **uzenet** karakterlánc típusú.

write-char

A **write-char** függvény egy karaktert ír ki a képernyőre, vagy a **fajlleiro**-val azonosított nyitott állományba. A szám a kiírni kívánt karakter decimális ASCII kódja; a függvény ezzel az értékkel tér vissza.

(write-char szam <fajlleiro>)

Például:

```
(write-char 67)        eredménye 67 és megjeleníti a
                       képernyőn a C betűt
```

Feltételezve, hogy az f egy megnyitott állomány azonosítója:

```
(write-char 67 f)      eredménye 67, és a C betűt ebbe
                       az állományba írja ki
```

A **write-line** függvény egy karaktersort ír ki a képernyőre, vagy a **fajlleiro**-val azonosított nyitott állományba. A függvény a szokott módon idézőjelek közé tett karaktersorral tér vissza, de a állományba az idézőjeleket nem illeszti be.

(write-line karaktersor <fajlleiro>)

1.12. Az AutoCAD entitásaihoz való hozzáférés

Az AutoCAD minden rajzeleme egy adott rajz-szesszióban, egy egységes **entitásnév**-vel rendelkezik. Ez egy egységes entitásazonosító és csak az adott szesszió alatt érvényes, ez alatt nem változik, ellenben ha újra kinyitjuk ugyanazt a rajzot, akkor igen. Az entitásazonosító egy karakterekből és számokból álló sor, még entitásnévnek is hívjuk.

Az **entlast** függvény kiértékelése az utolsó létrehozott AutoCAD entitás neve (azonosítója). Példa:

```
Command:(entlast)
<Entity name: 600000028>
```

Az **entnext** függvény visszafordítja a legelső létrehozott AutoCAD entitás nevét. Példa:

```
Command:(entnext)
<Entity name: 600000022>.
```

Ez a függvény opcionálisan megenged egy argumentumot, így használni lehet nem csak az első entitás nevének visszafordítására, hanem például a másodiknak is:

(entnext (entnext))

Entsel segítségével interaktív módon kiválaszthatjuk az entitást, amelynek a nevét fordítja vissza **és a pontot az entitáson, amelyre rákattintottunk**. Példa:

```
Command:(entsel)
Select object: (<Entity name: 7ef95ed0> (32.7847
16.8466 0.0)
```

Rendkívüli figyelemmel kell használni, mivel gyakori hiba a visszafordított pont figyelmen kívüli hagyása. Ha az entitás nevét akarjuk így elérni, akkor mindig használjuk előtte a car függvényt is, egyébként hibázni fogunk. Példa:

```
Command:(car (entsel))
Select object:
<Entity name: 7ef95ed0>.
```

1.13. Entitás nevének használata AutoCAD parancsokban

Ha egy AutoCAD entitás nevét egy változóhoz kötjük, a továbbiakban az AutoLISP programban, ezt az adott entitással kapcsolatos műveletekre használhatjuk.

```
(command ”line” ’(0.0 0.0) ’(10.0 10.0) ””)
Command:nil
Command:(setq ename (entlast))
<Entity name: 7ef98ea0>
Command:(command “erase” ename “”)
Command:nil
```

A fenti programsorokban először egy vonalat rajzolunk, majd utolsó elemként ennek nevét az ename változóhoz kötjük. Végül az AutoCAD erase paranccsal töröljük az ename entitást, vagyis a vonalat. Vagy:

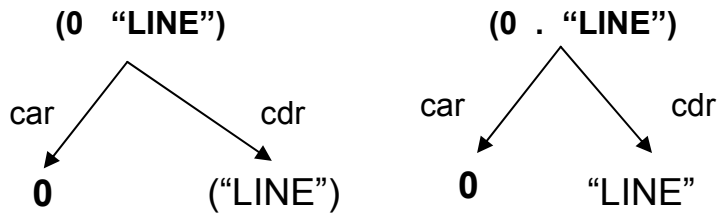
```
Command:circle
3P/2P/TTR <center point>:5,5
Diameter/<Radius>:10
Command:(setq kor1 (entlast))
```

```
<Entity name: 7ed95a70>
Command:(command ”move”  kor1 ”” ”1,0” ””)
```

1.14. Hozzáférés az entitások tulajdonságaihoz

Az entitások nevét használni lehet teljes paraméterlistájuk meg találásához és kikérdezéséhez, **asszociációs listák** formájában. Ez az eljárás és ennek a könnyedsége az AutoLISP egyik legnagyobb előnye. Mielőtt megtekintenénk, hogy miképp kell eljárni a fent említett cél érdekében, tisztázni kell egy újabb fogalmat: a **pontozott párok** fogalmát. Ezek felépítését és tulajdonságait azért kell ismernünk, mivel az egyes rajzelemek tulajdonságainak kikérdezésekor ilyen típusú listákat fogunk kapni, ezek az entitás asszociációs listáinak az építőelemei.

A **pontozott pár**, vagy **pontos lista** egy különleges struktúrájú lista. Ha megfigyeljük az **1.10. ábrát** akkor láthatjuk, hogy ez egy olyan lista, amelynek a bináris ágán a cdr függvénnyel elért adat a lista második és egyben utolsó elemét tartalmazza, a várt lista – első elem nélkül – helyett.



1.10. ábra
Szokásos és asszociációs lista

A pontos lista építése a **cons** függvény segítségével történik. Szintaxisa:

(cons <arg1> <arg2>)

Az arg1 a listához hozzáadandó elem, az arg2 pedig az elem, amelyhez hozzáadjuk az új elemet. Ha az elem, amelyhez hozzáadjuk az új elemet egy atom, akkor a visszafordított új adat egy pontozott pár lesz. Példa:

```
Command:(cons 0 ”LINE”)
(0 . ”LINE”)
```

Ha az elem, amelyhez hozzáadjuk az új elemet egy lista, akkor a **cons** egy listát fordít vissza, amelynek az első eleme az új hozzáadott elem:

```
Command:(setq lista (cons (list 1 2 3) lista))
((1 2 3))
Command:(setq lista (cons (list 4 5 6) lista))
((4 5 6 ) ( 1 2 3 ))
```

Az **entget** függvény egy entitás asszociációs listáját fordítja vissza:

(entget <arg>)

Az **arg**umentum egy entitásnév, a visszafordított adat pedig egy pontozott párokból álló lista:

((-1 . <Entity name: 7ef95ed0>) (0 . "LINE") (330 . <Entity name: 7ef95cf8>) (5 . "92") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbLine") (10 21.8778 7.81191 0.0) (11 33.3876 17.1751 0.0) (210 0.0 0.0 1.0))

A fenti lista egy egyszerű, két pont között húzódó, vonal rekord típusú listája. Többek között megfigyelhetjük, hogy tartalmazza a vonal nevét, az elemtípust és sok más adatot. A listában található adatok az entitások úgynevezett DXF állomány csoportkódjai szerint vannak értelmezve, és entitástípusként különbözőek lehetnek.

Például a fenti listában:

(-1 . <Entity name: 7ef95ed0>) – rajzelem neve
(0 . "LINE") – az entitás típusa vonal
(8 . "0") – a réteg vagy fólia, amelyben található
(10 21.8778 7.81191 0.0) – a vonal kezdőpontja
(11 33.3876 17.1751 0.0) – a vonal végpontja.

Amennyiben nem ismerjük az entitásban szereplő kódokat, ezt legegyszerűbben próbálkozással azonosíthatjuk. Jól ismert tulajdonságokkal ellátott entitások listáját megfigyelve, tulajdonságait változtatva és újravizsgálva a listáját, viszonylag hamar és pontosan megtudhatjuk a keresett tulajdonságok, karakterisztikák kódjait, ami általában elemfüggő is: például a vonal és a kör 10-es kódú allistája, vonal esetében annak kezdetét, kör esetén pedig a kör középpontját jelenti.

Az egyszerű vonal rekord mezőinek száma nem konstans, ha egy újabb vonalét mezőjét elemezzük, amelynek más színe van, akkor láthatjuk, hogy az hosszabb, megjelenik benne a (62 . 4) is például. Levonhatjuk következtetésnek, hogy az elemek rekordlistája nem azonos számú mezőből áll. A kérdés akkor, hogy hogyan férünk hozzá a különböző adatokhoz, ha listák nem azonos hosszúak, és esetleg a listaelemek nem mindig ugyanazon a helyen vannak?

A pontozott párok első eleme kulcsadat, amelyet az **assoc** AutoLISP függvénnyel lehet megtalálni az asszociációs listákon belül. A pontozott párok második

eleme a keresett adat, ezeket ki tudjuk olvasni és ezeken keresztül meg tudjuk a rajzelemeket is változatni. Szintaxisa:

(assoc <kod> <asszociacios lista>)

A **kod**, az allisták első eleme is változhat entitástípusként. Mint már említettük, a kör esetén a 10-es kódszámú allista a kör középpontjának a koordinátáit tartalmazza. Példa:

```
Command:circle
3P/2P/TTR <center point>:0,0
Diameter/<Radisu>:10
Command:(setq koros1 (entlast))
<Entity name: 7ef95ee8>
Command:(setq korkozep (assoc 10 koros1))
(10 0.0  0.0)
```

Vagy ha a kör középpontjának kell egyenesen a koordinátája:

```
Command:(setq korkozep (cdr (assoc 10 koros1)))
(0.0  0.0).
```

A DXF csoport-kódok a kezelése bonyolultnak tűnik, de ha a programozás előtt egyszerűen minden entitás típus esetében azonosítjuk őket, egy-egy bonyolultabb elemen (létezik a szín, layer stb.) keresztül, akkor nagyon egyszerűvé válik majd minden.

1.15. Rajzelemek módosítása az asszociációs listák segítségével

Az AutoCAD razjelemeket meg lehet változatni az asszociációs listájának megváltoztatásával egy újabb AutoLISP függvény használatával. Egy lista vagy a bonyolultabb esetben, egy asszociációs listatípus megváltoztatása lehetséges a **subst** függvény segítségével.

(subst <ujelem> <regielem> <lista>)

Ez a függvény kikeresi a listában szereplő régi tételeket (**regielem**), és eredményül a lista egy olyan másolatával tér vissza, amelyben a régi tételek minden előfordulását lecseréli az új tételre (**ujelem**). Amennyiben a **subst** függvény a listában nem talál **regielem**-et, a változatlan formájú listával tér vissza.

Például:

```
(setq lista1 '(a b (c d) b))
(subst 'XX 'b minta)              eredménye ( A XX ( C D ) XX )
(subst 'XX 'z minta)              eredménye ( A B ( C D ) B )
(subst 'XX '(c d) minta)          eredménye ( A B XX B )
(subst '(XX rr) '(c d) minta)     eredménye ( A B ( XX RR ) B )
(subst '(XX rr) 'z minta)         eredménye ( A B ( C D ) B )
```

Megjegyzés: az AutoLISP a kiértékelésben az s-kifejezéseket (a karaktersorok kivételével) nagybetüvel téríti vissza.

A subst függvényt az assoc függvénnyel együtt használva, egy asszociációs lista adott kódjához tartozó értékeit egyszerűen lecserélheti. Példa:

```
Command: circle
3P/2P/TTR <center point>:0,0
Diameter/<Radisu>:10
Command: (setq koros1 (entget (entlast)))
((-1 . <Entity name: 7ef95f10>) (0 . "CIRCLE") (330 .
<Entity name: 7ef95cf8>) (5 . "9A") (100 .
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 .
"AcDbCircle") (10 0.0 0.0 0.0) (40 . 10.0) (210 0.0 0.0
1.0))
Command:(setq old_radius (assoc 40 koros1)))
(40 . 10.0)
Command: (setq new_center (cons 40 1))
(40 . 1)
Command: (setq koros1 (subst new_center old_center ko-
ros1))
((-1 . <Entity name: 7ef95f10>) (0 . "CIRCLE") (330 .
<Entity name: 7ef95cf8>) (5 . "9A") (100 .
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 .
"AcDbCircle") (10 0.0 0.0 0.0) (40 . 1.0) (210 0.0 0.0
1.0)).
```

Látható, hogy a 40-es kódszámú pontozott pár tartalmával megváltozott a **koros1** asszociációs lista tartalma is. De nem történt semmi a rajzon. Megállapítható, hogy a **subst** függvény nem destruktív (a listát érintetlenül hagyja)! Ha a rajzon is változatni akarunk, akkor egy másik AutoLISP függvényt is kell használnunk: az **entmod**-ot.

(entmod <asszocios lista>)

A fenti példát folytatva:

```
Command:(entmod koros1)
((-1 . <Entity name: 7ef95f10>) (0 . "CIRCLE") (330 .
<Entity name: 7ef95cf8>)(5 . "9A") (100 . "AcDbEntity")
(67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbCircle")
(10 0.0 0.0 0.0) (40 . 10.0) (210 0.0 0.0 1.0))
```

A végső utasítás eredménye egy visszafordított asszociációs lista, a rajzon pedig az előzőleg 10-es sugarú kör 1-es sugarú körre való zsugorodása. A módszer alkalmazható bármilyen AutoCAD rajzelemre.

1.16. Kiválasztott elemsorozatok kezelése AutoLISP-el

Az AutCAD csoportosíthatja az entitásokat elemcsoportokban, az AutoLISP pedig feldolgozhatja ezeket, saját függvényeket használva. Az adatbázisban a kiválasztott elemsorozat nem egy lista, hanem egy különleges adattípus: PICSET! Mindig kössük egy változóhoz őket!

Az **ssget** függvény egy kiválasztott elemsorozattal tér vissza.

(ssget <mod> <pont1 <pont2>> <pontlista> <szurolista>)

Az opcionális mód argumentum egy karakterlánc, amely az alkalmazott rajzelem kiválasztási módszerét határozza meg. A **mod** lehet: **„W"** és **„C"** - window, vagyis ablak, illetve keresztablak kijelölés a pont1 és pont2 között. Amennyiben mod argumentum nélkül csak egyetlen pontot ad meg, ez a pontkijelöléssel történő rajzelem kiválasztásának felel meg. Az **„L"** és **„P"** mod az utolsó, illetve az előző kijelölést jelenti.

Egy további opció az **„X"** mod, amely a teljes adatbázist választja ki. A szurolista argumentum bármelyik moddal együtt használható. Ide az asszociációs listák pontozott párjai írhatók be, az AutoLISP ilyenkor csupán azokat az elemeket jelöli ki, amelyeknek az asszociációs listájában jelen van a szűrőlista minden eleme. Példa:

(ssget "X" '((0 . "LINE") (8 . "CURENT") (62 . 2)))

A fenti programsor az aktuális AutoCAD minden rajzeleméből csak azokat válassza ki, amelyek vonalak, az aktuális rétegen vannak és sárga színűek (AutoCAD színek száma).

Abban az esetben, ha az összes argumentumot elhagyja, az ssget függvény az AutoCAD általánosan használt **Select objects:** módszerével kéri a felhasználót, hogy interaktív módon állítson össze egy kiválasztott elemsorozatot. A kiválasztott elemsorozat egyaránt tartalmazhat papír- és modelltérbeli rajzelemeket, de amikor

a kiválasztott elemsorozatot felhasználja egy műveletben, akkor az aktuálisan nem használt tér elemeit az AutoCAD automatikusan kiszűri. Ez az összes AutoCAD parancsra igaz. Az ssget függvénnyel kiválasztott elemsorozat csak fő rajzelemeket tartalmazhat (Attribútumokat vagy Polyline csomópontjait nem).

Az **ssadd** függvény új, elemek nélküli kiválasztott elemsorozatot hoz létre, ha argumentumok nélkül kerül hívásra.

(ssadd <elemnev <elemsor >)

Ha az argumentum egyetlen elemnév, akkor az ssadd függvény ezt, az egy kiválasztott elemet tartalmazó elemsorozatot készíti el. Abban az esetben, ha egy elemnévvel és egy **elemsor** kiválasztott elemsorozattal hívja meg, akkor a függvény a rajzelemet hozzáadja a kiválasztott elemsorozathoz. Az ssadd függvény mindig az új vagy módosított kiválasztott elemsorozattal tér vissza. Amennyiben egy rajzelemet hozzáad egy kiválasztott elemsorozathoz, ez a rajzelem ténylegesen is hozzáadódik a létező sorozathoz, és a függvény az így kapott elemsor sorozattal tér vissza. Amennyiben a nevezett rajzelem már szerepel a sorozatban, a rendszer az ssadd műveletről nem vesz tudomást, és hibajelzést sem ad.

Az **ssdel** függvény kitörli az elemnév nevű elemet az elemsor nevű kiválasztott elemsorozatból, és az **elemsor** nevével tér vissza.

(ssdel <elemnev> <elemsor>)

Meg kell jegyezni, hogy a függvény nem egyszerűen egy, a megadott elem nélküli új elemsorozattal tér vissza, hanem a rajzelem ténylegesen is kitörlődik az adott kiválasztott elemsorozatból. Ha a rajzelem nem szerepel az elemsorozatban, akkor a függvény nil értékkel tér vissza.

Ha egy kiválasztott elemsor elemeinek számát akarjuk megtudni, használjuk az **sslegth** függvényt.

(sslength <elemsor>)

A kiértékelés az elemsor kiválasztott elemsorozatban lévő rajzelemek egész értékben kifejezett számával tér vissza. Amennyiben a szám a 32767 értéket meghaladja, valós típusú lesz. A kiválasztott elemsorozatban soha nem szerepel kétszer ugyanaz az elem.

Az **ssmeb** függvény megvizsgálja, hogy az elemnév nevű rajzelem szerepel-e az elemsor kiválasztott elemsorozatban.

(ssmemb <elemnev> <elemsor>)

Ha igen, akkor az **ssmemb** a rajzelem nevével (**elemnev**) tér vissza, ha nem, akkor pedig nil értékkel.

Az AutoCAD elemeihez való hozzáférés AutoLISP programmal elképzelhetetlen lenne egy olyan függvény nélkül, amely egy elemsor valahányadik eleméhez nem férhetne hozzá, mint egy jól meghatározott entitáshoz. Ezt az **ssname** függvény teszi lehetővé.

(ssname <elemsor> <index>)

Ez a függvény az elemsor nevű kiválasztott elemsorozat **index**-edik elemének **nevével** tér vissza. Amennyiben az index értéke negatív, vagy nagyobb, mint a kiválasztott elemsorozat legutolsó elemének indexe, a függvény nil értékkel tér vissza. **Az elemsorozat első elemének indexe 0!** Az ssget függvénnyel képzett kiválasztott elemsorozatokban csak a fő rajzelemek nevei találhatók. Alárendelt rajzelemekhez (Block-ok attribútumaihoz és Polyline-ok csomópontjaihoz) ezzel a módszerrel nem férhetünk hozzá. Ezek elérésére az **entnext** függvény nyújt lehetőséget. Például:

```
(setq esor (ssget)) - létrehoz egy esor nevű kiválasz-
                      tott elemsorozatot
(setq ent1 (ssname ssget 0))- az elemsor első elemének
                              a nevét téríti vissza
(setq ent4 (ssname sset 3)) - az ent4 az elemsor negye-
                              dik elemének a nevét
                              téríti vissza
```

A 32767-nél magasabb indexű rajzelemek eléréséhez az index argumentumot valós számként kell megadni.

1.17. Gyakran használt AutoLISP függvények

Ez a fejezet az eddig nem tárgyalt, de gyakran előforduló AutoLISP függvényeket mutatja be. Természetesen, ezeknek a függvényeknek nem kisebb a jelentősége mint az előzőknek. Néhányukat, mint például a matematikai függvényeket is, állandóan használjuk, azonban megértésükhöz elégséges egy vázlatos ismertetés. A jelen fejezetben az egyszerűség kedvéért a „Command:” kiírását elhagyjuk.

+ (összeadás)

(+ <szam1> <szam2> . . .)

A függvény a listában szereplő összes **szám összegével** tér vissza. Egyaránt használható valós, illetve egész számok esetében. Amennyiben az összes szám egész típusú, akkor az eredmény is egész lesz; ha bármelyik szám valós, akkor az

egész számok valós típusúvá lépnek elő, és a végeredmény valós típusú lesz. Például:

```
(+ 1 2) eredménye 3
```

– (kivonás)

(– <szam1> <szam2> . . .)

Ez az aritmetikai függvény az **első számból kivonja a második számot**, és a kettő különbségével tér vissza. Amennyiben kettőnél több szám szerepel a listában, akkor a függvény a másodiktól az utolsóig terjedő számok összegét vonja ki az elsőből, és ezzel a végeredménnyel tér vissza. Amennyiben a listában csak egy szám szerepel, akkor az eredmény ennek nullából levont értéke lesz. A függvény a kiértékelés fent ismertetett szabályaival valós és egész számokkal egyaránt használható. Példa:

```
(– 7 0 3 0) eredménye 4
```

* (szorzás)

(* <szam1> <szam2>. . .)

Ez a függvény a listában szereplő összes **szám szorzatával tér vissza**. A kiértékelés szokásos szabályainak figyelembevételével valós és egész számokkal egyaránt használható. Abban az esetben, ha csak egy számot ad meg, a függvény ezt 1-gyel szorozza meg, és ennek eredményével tér vissza. Példa:

```
(* 4 3)     eredménye 12
```

/ (osztás)

(/ <szam1> <szam2> . . .)

Ez a függvény **elosztja az első számot a másodikkal**, és a hányadosukkal tér vissza. Amennyiben argumentumként kettőnél több számot ad meg, akkor a függvény az első számot a másodiktól az utolsóig terjedő számok szorzatával osztja el, és a végső hányadossal tér vissza. A kiértékelés szokásos szabályainak figyelembevételével valós és egész számokkal egyaránt használható. Abban az esetben, ha csak egy számot ad meg, a függvény ezt 1-gyel osztja el, és ennek eredményével tér vissza. Példa:

```
(/ 300 2)        eredménye 150
(/ 2 5)          eredménye 0
(/ 2 5.0)        eredménye 0.4
```

= (egyenlő)

(= <atom1> <atom2>...)

Ez **az egyenlő relációs** függvény. Ha a listában megadott összes atom numerikus értéke megegyezik, akkor eredményül T értékkel, egyébként nil értékkel tér vissza. Ez a függvény számokra és karaktersorokra érvényes. Példa:

```
(= 2.5  2.5)          eredménye T
(= 2  389)            eredménye nil
(= "one" "one")       eredménye T
(= " one" "one")      eredménye nil
```

/= (nem egyenlő)

(/= <atom1> <atom2>...)

Ez a **nem egyenlő relációs** függvény. Ha az első atom numerikus értéke nem egyenlő a második atom numerikus értékével, akkor T, ha egyenlő, akkor pedig nil értékkel tér vissza. A függvény kettőnél több argumentum esetére határozatlan. Például:

```
(/ = 101  2  0)       eredménye T
(/ = "Janos" "Janos") eredménye nil
```

< (kissebb mint)

(< <atom1> <atom2>...)

Ez a **kisebb mint** relációs függvény. Amennyiben az első atom numerikus értéke kisebb a második numerikus értékénél, akkor T, egyébként nil értékkel tér vissza. Amennyiben a listában kettőnél több atom szerepel, akkor a függvény csak abban az esetben tér vissza T értékkel, ha minden atom kisebb a tőle jobbra lévőnél. Példa:

```
( < 1 20)         eredménye T
(< "b" "c")       eredménye T
(< 327 53,2)      eredménye nil
( < 2 3 88)       eredménye T
( < 2 3 4 4)      eredménye nil
```

< = (kissebb vagy egyenlő)

(<= <atom1> <atom2>...)

Ez a **kisebb vagy egyenlő** relációs függvény. Ha az első atom numerikus értéke kisebb vagy egyenlő a második numerikus értékénél, akkor a függvény T, egyébként nil értékkel tér vissza. Amennyiben a listában kettőnél több atom szerepel, a függvény csak abban az esetben tér vissza T értékkel, ha minden atom kisebb vagy egyenlő a tőle jobbra lévőnél. Példa:

```
(<= 1 20)        eredménye T
(<= "b" "a")     eredménye T
```

> (nagyobb mint)

(> <atom1> <atom2>...)

Ez a **nagyobb mint** relációs függvény. Ha az első atom numerikus értéke nagyobb a második numerikus értékénél, akkor a függvény T, egyébként nil értékkel tér vissza. Amennyiben a listában kettőnél több atom szerepel, a függvény csak abban az esetben tér vissza T értékkel, ha minden atom nagyobb a tőle jobbra lévőnél. Például:

```
(> 12 17)             eredménye nil
(> "d" "b")           eredménye T
(> 7777,51 1792)      eredménye T
(> 77 4 2)            eredménye T
(> 77 4 4)            eredménye nil
```

>= (nagyobb vagy egyenlő)

(>= <atom1> <atom2>...)

Ez a **nagyobb vagy egyenlő** relációs függvény. Ha az első atom numerikus értéke nagyobb vagy egyenlő a második numerikus értékénél, akkor a függvény T, egyébként nil értékkel tér vissza. Amennyiben a listában kettőnél több atom szerepel, a függvény csak abban az esetben tér vissza T értékkel, ha minden atom nagyobb vagy egyenlő a tőle jobbra lévőnél. Példa:

```
(>= 120 120)     eredménye T
(>= "c" "c")     eredménye T
(>= 1,51 1722)   eredménye nil
(>= 70 4 4)      eredménye T
(>= 27 4 11)     eredménye nil
```

~

(~ szam)

Ez a függvény a szám értékén végzett bitenkénti NOT művelet eredményével, vagyis a szám **egyes komplemensével** tér vissza. A számnak egész típusúnak kell lennie. Példa:

```
(~ 3)            eredménye – 4
(~ 100)          eredménye -101
(~ - 4)          eredménye 3
```

1+ (plusz egy)

(1+ <szam>)

Ez a függvény a szám **1-gyel megnövelt értékével** tér vissza. A szám egész vagy valós típusú egyaránt lehet. Példa:

```
(1+ 56)          eredménye 57
(1+ -11,2)       eredménye –10. 2
```

1– (minusz egy)

(1– <szam>)

Ez a függvény a szám **1-gyel csökkentett értékével** tér vissza. A szám egész vagy valós típusú egyaránt lehet. Példa:

```
(1- 3)           eredménye 2
(1- -37,5)       eredménye -38.5
```

abs (abszolút)

(abs <szam>)

Ez a függvény a szám **abszolút értékével** tér vissza. A szám egész vagy valós típusú egyaránt lehet. Például:

```
(abs 1011)       eredménye 1011
(abs -30)        eredménye 30
(abs -29,45)     eredménye 29.45
```

ads

(ads)

Az **AutoCAD Fejlesztői Rendszer (ADS) aktuálisan betöltött alkalmazásainak listájával tér vissza**. A listában minden alkalmazás és annak elérési útvonala idézőjelbe tett karaktersorként szerepel. Példa:

```
(ads)      eredménye lehet("files / progs / PROG1 " "
PROG2 " )
```

alert

(alert <karaktersor>)

Egy **figyelmeztető ablakot** jelenít meg a karaktersor argumentumban átadott hibaüzenettel vagy figyelmeztetéssel. A figyelmeztető ablak egyetlen OK nyomógombot tartalmazó párbeszédablak. Példa:

```
(alert "Valassz nagyobb szamot.")
```

Ha a kiírandó karaktersorban újsor karaktert használunk, akkor az több sorban is megjeleníthető.

and (logikai és)

(and <kifejezes...>)

Ez a függvény a kifejezések listáján elvégzett **logikai AND művelet** eredményével tér vissza. Amennyiben bármelyik kifejezés értéke nil, a függvény abbahagyja a kiértékelést, és visszatér nil értékkel; egyébként T értékkel tér vissza:

```
(setq a 103) (setq b nil) (setq c "karakterek")
(and 1,4 a c )        eredménye T
(and 1,4 a b c )      eredménye nil
```

angle

(angle <pont1> <pont2>)

Ez a függvény az UCS **pont1 pontját az UCS pont2 pontjával összekötő egyenes szögével** (az OX tengellyel bezárt szöge) tér vissza. A szög az aktuális szerkesztési sík X tengelyétől értendő, radiánban kifejezve, az óramutató járásával

ellentétes irányban. Amennyiben a függvény bemenő adatként 3D pontokat kap, ezeket merőlegesen levetíti az aktuális szerkesztési síkra.

angtof

(angtof <karaktersor> [mod])

Lebegőpontos értékké konvertálja a karaktersort, amely egy szöget ábrázol a **mod** által megadott kijelzési formátumban. Az angtof függvény az eredményt radiánban fejezi ki.

A mod argumentum határozza meg, hogy a karaktersor formátuma milyen mértékegységnek felel meg. Értéke az AutoCAD AUNITS rendszerváltozója által megengedett, alább látható értékek egyike kell legyen. A mód argumentum elhagyása esetén az angtof függvény az AUNITS rendszerváltozó aktuális értékét veszi alapul.

Szögmértékegységek értékei:

Mód (**mod**)	Karaktersor alakja
0	Fok
1	Fok/perc/másodperc
2	Új fok (Grad)
3	Radián
4	Geodéziai mértékegység

A karaktersornak az angtof függvény számára a megadott mód szerint értelmezhetőnek kell lennie. Formája azonos lehet azzal, amelyet az angtos függvény szolgáltatna, vagy amelyet az AutoCAD a billentyűzetről elfogad. **Az angtof és az angtos függvények egymás ellentettei.** Az angtof függvény egy radiánban kifejezett valós számértékkel, máskülönben pedig nil értékkel tér vissza.

angtos

(angtos <szog> [mod] <pontossag>)

Ez a függvény a valós számként, radiánokban megadott **szöget átírja egy karaktersorba**, a mod, a pontossag, az AutoCAD UNITMODE rendszerváltozó és a DIMZIN méretváltozó beállításának megfelelően. A mod és pontosság argumentumok egész számok, amelyek a szögmértékegységeket és a pontosságot határozzák meg. Az értelmezhető **mod** értékek megegyeznek az előző függvény táblázatában bemutatott értékekkel.

A pontosság argumentum egész szám, amely a kívánt tizedes jegyek számát adja meg. A mod és a pontosság megfelel az AutoCAD AUNITS, illetve AUPREC rendszerváltozójának. Abban az esetben, ha elhagyja az argumentumokat, az angtos függvény az AUNITS és AUPREC rendszerváltozók aktuális beállítását veszi alapul.

append

(append <kifejezes...>)

Ez a függvény **tetszőleges számú listát** és csakis listát **egyetlen listává fűz össze**. Példa:

```
(append '(a b) '(c d))            eredménye (A B C D)
(append '((a) (b)) '((c) (d)))    eredménye ((A) (B) (C)
(D))
```

ascii

(ascii <karaktersor>)

A függvény a karaktersor első karakterének az **ASCII kódjával** (egy egész számmal) tér vissza. Példa:

```
(ascii "A")      eredménye 65
(ascii "a")      eredménye 97
(ascii "BAL")    eredménye 66
```

atan

(atan <szam1> <szam2>)

A szam1/szam2 radiánokban kifejezett **arctangensével** tér vissza. Amennyiben a szám2 nem szerepel a listában, az atan függvény a szam1 radiánokban kifejezett **arctangensével** tér vissza. A szám1 lehet negatív is, az eredményül kapott szög értéke a -n/2 és +n/2 közötti intervallumban helyezkedik el. Példa:

```
(atan 0.5)            eredménye 0.463648
(atan 2.0 3.0)        eredménye 0.588003
```

atof

(atof <karaktersor>)

Ez a függvény **egy karaktersort valós számmá konvertál**. Példa:

```
(atof "25.1")    eredménye 25.1
(atof "78")      eredménye 78.0
```

atoi

(atoi <karaktersor>)

Ez a függvény **egy karaktersort egész számmá konvertál**. Példa:

```
(atoi "9527")    eredménye 9527
(atoi "32")      eredménye 32
```

atom

(atom <tetel>)

Ha a **tetel** lista, ez a függvény nil értékkel tér vissza, egyéb esetekben T értékkel. Ami nem lista, az mind atomnak tekinthető. Példa:

```
(atom 'a)        eredménye T
(atom a)         eredménye nil
(atom '(a b c))  eredménye nil
```

atoms-family

(atoms-family [formatum] <szimblista>)

Ez a függvény a beépített és **adott alkalommal definiált egyéb szimbólumok listájával tér vissza**. A **formatum** argumentum egy egész szám, melynek értéke 0 vagy 1 lehet. Ha a formátum értéke 0, akkor az aktuálisan definiált szimbólumok listájával tér vissza; ha értéke 1, akkor a szimbólumok karaktersorok formájában jelennek meg a listában. Példa:

```
(atoms-family 0) az aktuálisan definiált szimbólumok
                 listáját adja eredményül
```

boundp

(boundp <atom>)

Ez a függvény **T értékkel tér vissza, amennyiben az atomhoz érték tartozik** (figyelmen kívül hagyva az értelmezési tartományát). Ha az atomhoz nincs érték rendelve (vagy ez az érték nil), akkor a boundp függvény nil értékkel tér vissza. Abban az esetben, ha az atom egy definiálatlan szimbólum, akkor automatikusan létrejön, és nil értéket kap.

Példa:

```
(setq a 8) (setq b nil)
(boundp 'a) eredménye T
(boundp 'b) eredménye nil
```

caar, cadar, cadr, caddr, cdar, cddr

(caar <lista>)

(cadar <lista>)

(cadr <lista>)

(caddar <lista>)

(cdar <lista>)

(cddr <lista>)

Az AutoLISP megengedi **a car és cdr függvények összefűzését** maximum négy szint mélységig. A fenti függvények 3D AutoCAD pontok koordinátáinak az egyenes elérését segítik, amint azt az alábbi példák is szemléltetik.:

```
(setq mmm '((a b) c d)))
```

Ebben az esetben:

```
(caar mmm) = (car (car mmm))          eredménye A
(cdar mmm) = (cdr (car mmm)) eredménye (B)
(cadar mmm) = (car (cdr (car mmm)))   eredménye B
(cadr mmm) = (car (cdr mmm))          eredménye ()
(cddr mmm) = (cdr (cdr mmm))          eredménye (D)
(caddr mmm) = (car (cdr (cdr mmm)))   eredménye D
```

Az AutoLISP programnyelvben a cadr függvényt gyakran használjuk arra, hogy segítségével megkapjuk egy 2D vagy 3D pont Y koordinátáját (két vagy háromelemű lista második elemét). Hasonlóképpen, a caddr függvény egy 3D pont Z koordinátájának meghatározására használható. Nézzük például az alábbiakat:

```
(setq pt2 '(1.2 1.0))
(setq pt3 '(1.2 1.0 3.0))
(car pt2)         eredménye 1.2
(cadr pt2)        eredménye 1.0
(caddr pt2)       eredménye nil
(car pt3)         eredménye 1.2
(cadr pt3)        eredménye 1.0
(caddr pt3)       eredménye 3.0
```

chr

(chr <egesz>)

Ez a függvény egy ASCII kódot jelentő **egész számnak megadja a megfelelő egyelemű karaktersorát**. Például:

```
(chr 65)  eredménye "A"
(chr 66)  eredménye "B"
(chr 97)  eredménye "a"
```

close

(close <fajlleiro>)

Ez a függvény lezár egy flet, és nil értékkel tér vissza. A **fajlleiro** az állomány azonosítója, melyet az open függvény adott eredményül. A **close** függvény hívását követően a **fajlleiro** változatlan marad, de érvényét veszti. Példa:

```
(close x) lezárja a hozzá tartozó állományt; eredménye
nil.
```

cos (cosinus)

(cos <szog>)

Ez a függvény a radiánban kifejezett szög koszinuszával tér vissza. Példa:

```
(cos 0.0)           eredménye 1.0
(cos pi)            eredménye -1.0
```

cvunit

(cvunit <mennyiseg> [ebbol] [ebbe])

Ez a függvény **egy mennyiséget vagy egy pontot (a lista elemeit) az egyik mértékegységből egy másikba számít át**. Sikeres végrehajtás esetén a függvény az így kapott mennyiséggel vagy ponttal tér vissza. Ha azonban valamelyik mértékegység neve ismeretlen (nem található az acad.unt állományban) vagy a két mértékegység eltérő dimenziójú (például grammok átszámítása évekre), akkor a függvény nil értékkel tér vissza.

A mennyiség az átszámítani kívánt számértéket jelöli. Ez két vagy három számot tartalmazó lista (2D, illetve 3D pont) is lehet. Az **ebbol** argumentum azt a mér-

tékegységet jelöli, amelyből a mennyiséget át kívánja számítani. Az **ebbe** argumentum pedig az a mértékegység, amelyikbe az átváltás történik. Az ebből és az ebbe argumentum az acad.unt állományban található bármilyen típusú mértékegység lehet.
Példa:

```
(cvunit 1 "minute" "second")     eredménye 60.0
(cvunit 1 "gallon" "furlong")    eredménye nil
(cvunit 1.0 "inch" "cm")         eredménye 2.54
(cvunit '(1.0 2.5) "ft" "in")    eredménye (12.0 30.0)
```

distance

(distance <pont1> <pont2>)

Ez a függvény a pont1 és pont2 AutoCAD pontok közötti térbeli (3D) távolság értékével tér vissza. A következő példák ezt szemléltetik:

```
(distance '(1.0 2.5 3.0)  '(7.7 2.5 3.0)) eredménye 6.7
(distance  '(1.0  2.0  0.5)   '(3.0  4.0  0.5))  eredménye
2.82843
```

Abban az esetben, ha a megadott pontok egyike, vagy mindkettő 2D pont, akkor a distance a 3D pontok Z koordinátáját nem veszi figyelembe, és a pontok aktuális szerkesztési síkra vetített távolságát számítja ki.

distof

(distof <karaktersor> [mod])

A distof függvény **valós értékké konvertálja a karaktersort**, amely egy valós (lebegőpontos) értéket ábrázol a **mod** által megadott kijelzési formátumban.

A **mod** argumentum határozza meg, hogy a karaktersor formátuma milyen mértékegységnek felel meg. Értékének az AutoCAD LUNITS rendszerváltozója által megengedett, az alábbi táblázatban látható értékek egyikének kell lennie. A mod argumentum elhagyása esetén a distof függvény a LUNITS rendszerváltozó aktuális értékét veszi alapul.

Hosszmértékegységek értékei

Mód (**mod**)	Karaktersor alakja
1	Tudományos
2	Tizedes
3	Mérnöki (láb és tizedes hüvelyk)
4	Építészeti (láb és a hüvelyk törtrésze)
5	Tört

A karaktersor argumentumnak a distof függvény számára a megadott mód szerint értelmezhető karaktersornak kell lennie. Formája azonos lehet azzal, amelyet az rtos függvény szolgáltatna, vagy amelyet az AutoCAD a billentyűzetről elfogad. **A distof és az rtos függvények egymás ellentettei:** amennyiben a distof függvénynek egy, az rtos függvény által előállított karaktersort ad át, akkor a distof bizonyosan érvényes eredménnyel tér vissza, és megfordítva (feltéve, hogy a mód értékei azonosak).

entdel

(entdel <elemnev>)

A függvény az elemnev **argumentum által meghatározott rajzelemet kitörli**, ha az szerepel a rajzban, illetve visszaállítja, ha azt az adott szerkesztési alkalommal előzőleg kitörölték. A kitörölt rajzelemeket az AutoCAD-ből való kilépéskor véglegesen eltávolítja a rajzból, így az entdel a törölt rajzelemeket csak addig képes visszaállítani, amíg az adott szerkesztési szesszió véget nem ér.

Az entdel függvény csak fő rajzelemek esetében használható; az Attribútumok és a Polyline-ok csomópontjai nem törölhetők az őket tartalmazó rajzelemektől függetlenül (erre a célra a command függvény segítségével az ATTEDIT, illetve a VLEDIT parancsokat használhatja). Példa:

```
(setq elem (entnext))
(entdel e1)             törli az elem rajzelemet
(entdel e1)             visszaállitja az elem rajzelemet
```

Block-definíción belül nem törölhet ki rajzelemeket. Az entmake függvénnyel azonban átírhat egy teljes Block-definíciót (kihagyva belőle a törölni kívánt rajzelemet).

entmake

(entmake [elemlista])

Ez a függvény **új rajzelemet hoz létre a rajzban**. Amennyiben az elem létrehozása sikerrel járt, akkor annak definíciós adatlistájával tér vissza. Amikor valamilyen okból nem sikerült az elemet létrehozni (például hibás adatmegadás miatt), a visszatérési érték nil.

Az elemlista argumentumban egy rajzelem definíciós adatlistáját kell megadni az entget függvénnyel kapható formátum szerint. Ennek tartalmaznia kell a rajzelem megadásához szükséges összes információt. Bármely szükséges definíciós adat elhagyása az entmake függvény nil értékkel történő visszatéréséhez, és a rajzelem elvetéséhez vezet. Az opcionális adatok, mint a fólia elhagyása esetén az entmake az alapértelmezés szerinti értékeket használja. Az új rajzelem létrehozá-

sának egyik módja, hogy egy rajzelemnek az entget függvénnyel megszerzett definíciós adatait módosítja, és az entmake függvénnyel új rajzelemként a rajzhoz csatolja.

Mielőtt létrehozná az új rajzelemet, az entmake függvény ellenőrzi, hogy a megadott layer-név, vonaltípus-név és szín érvényes-e. Amennyiben új layer adott meg, az entmake automatikusan létrehozza az új layert. Amennyiben a rajzelem típusa szükségessé teszi, az entmake a Blockok, Dimension Style, Text Style neveit is ellenőrzi.

Az elemlista első vagy második tételének a rajzelem típusának (például Circle, Line, Polyline stb.) kell lennie. Második csak akkor lehet, ha a rajzelem neve előzi meg. Ez az a formátum, amelyet az entget függvény szolgáltat. Ilyen esetekben az új rajzelem létrehozásakor a rajzelem nevét figyelmen kívül hagyja, valamint, ha az **elemlista** sorszámot tartalmaz, akkor azt is.

entupd

(entupd <elemnev>)

Amennyiben az entmod függvénnyel egy Polyline csomópontját vagy egy Block attribútumát módosítja, az összetett rajzelem rajza a képernyőn nem aktualizálódik. Amikor például egy 100 csomóponttal rendelkező Polyline-t kell módosítani, a csomópontok megváltoztatásával egyidejűleg, az újraszámolás és az újrarajzolás elfogadhatatlanul lelassítaná a program működését. A módosított Polyline vagy Block képernyőn történő aktualizálására az entupd függvényt használhatja. Ez a függvény a Polyline vagy a Block bármely alárendelt rajzelemének nevével meghívható. Nem szükséges, hogy ez a fejrész rajzelem neve legyen, az entupd függvény meg fogja azt találni. Bár az entupd függvény főleg a Polyline-ok és az attribútummal rendelkező Blockok esetében használatos, valójában bármely rajzelem aktualizálására alkalmas. A függvény a rajzelemet valamennyi alárendelt rajzelemével egyetemben minden esetben regenerálja a képernyőn.

eq

(eq <kifejezes1> <kifejezes2>)

A függvény megvizsgálja, hogy a **kifejezés1 és a kifejezés2 azonos-e egymással**, azaz, ugyanazon objektumokhoz lettek-e hozzárendelve (például a setq függvénnyel). Amennyiben a két kifejezés azonos, az eq függvény T értékkel, egyébként pedig nil értékkel tér vissza. Ez a függvény tipikusan annak megállapítására használható, hogy két lista tényleges tartalma azonos-e egymással. Példa:

```
(setq lista1 '(a b c))
(setq lista2 '(a b c))
(setq lista3 lista2)
```

ekkor:

```
(eq lista1 lista3)     eredménye nil
(eq lista2 lista3)     eredménye T
```

equal

(equal <kifejezes1> <kifejezes2> [elteres])

Ez a függvény megvizsgálja, hogy a **kifejezes1 és a kifejezes2 egyenlő-e egymással**, azaz, kiértékelésük eredménye megegyezik-e. Példa:

```
(setq f1 '(a b c))
(setq f2 ' (a b c))
( setq f3 f2)

Ekkor

(equal f1 f3) eredménye T
(equal f3 f2) eredménye T
```

Azt a két listát, amelyet az equal függvény egyenlőnek talált, nem biztos, hogy az eq függvény is azonosnak fogja találni, viszont két egymással egyenlő (equal) atom egyben azonos (eq) is. Azok a listák és atomok viszont, amelyek az eq függvény szempontjából azonosak, az equal függvény szempontjából is mindig megegyeznek.

Opcionálisan egy numerikus **elteres** argumentum is megadható, amely azt a maximális eltérést határozza meg, amely határon belül a kifejezés1 és a kifejezés2 még egyenlőnek tekinthető. Példa:

```
(setq a 1.123456)
(setq b 1.123457)
(equal a b)           eredménye nil
(equa1 a b 0.000001 ) eredménye T
```

error

(*error* <karaktersor>)

Ez a felhasználó által **definiálható hibakezelő függvény**. Ha értéke nem nil, akkor minden AutoLISP hiba bekövetkezésekor végrehajtásra kerül. Egyetlen argumentumaként az AutoLISP a hiba leírását tartalmazó karaktersort adja át számára. Példa:

```
(defun *error* (uzenet)
      (princ "hiba :" )
      (princ uzenet)
      (terpri)
)
```

Az így definiált függvény pontosan úgy működik, mint az AutoLISP szabvány hibakezelője; kiírja az **error:** feliratot és a hiba leírását. A **terpr**i függvény egy újsor karaktert ír ki a képernyőre, és nil értékkel tér vissza.

eval

(eval <kifejezes>)

A **kifejezés kiértékelésének eredményével tér vissza**, ahol a kifejezés helyén tetszőleges AutoLISP kifejezés szerepelhet.

exit

(exit)

Az exit függvény **kilépést hajt végre az aktuális alkalmazásból**. Az exit függvény hívásának eredményeképpen a quit/exit megszakítás hibaüzenet, majd az AutoCAD Command: promptja jelenik meg.

exp

(exp <szam>)

Ez a függvény az e konstansnak a szám kitevőre emelt hatványával tér vissza (természetes logaritmus inverze). Az eredmény egy valós szám lesz. Példa:

```
(exp 1.0)  eredménye 2.71828
(exp 2.2)  eredménye 9.02501
```

expt

(expt <alap> <kitevo>)

Ez a függvény az **alap számnak a kitevore emelt hatványával tér vissza**. Amennyiben mindkét szám egész típusú, az eredmény is egész lesz; egyébként valós szám. Példa:

```
(expt 2 3)        eredménye 8
(expt 3.0 2.0)    eredménye 9.0
```

fix

(fix <szam>)

Ez a függvény **a szám egész típusúvá konvertált értékével tér vissza**. A szám egyaránt lehet egész vagy valós típusú. A valós számokat a egész típusú szám értékére csonkolja, eltávolítva a tört részt. Példa:

```
(fix 7)           eredménye 7
(fix 31.99)       eredménye 31
```

float

(float <szam>)

Ez a függvény **a szám valós típusúvá konvertált értékével tér vissza**. A **szám** lehet egész vagy valós típusú szám. Példa:

```
(float 55)        eredménye 55.0
(float 3.75)      eredménye 3. 75
```

gcd

(gcd <szam1> <szam2>)

Ez a függvény **a szam1 és szam2 legnagyobb közös osztójával tér vissza.** A szam1 és szam2 értékeknek egész típusúaknak kell lenniük. Példa:

```
(gcd 100 50)      eredménye 50
(gcd 12 20)       eredménye 4
```

getangle

(getangle <pont> [prompt])

A függvény arra vár, hogy a felhasználó megadjon egy szöget, majd ezután **a szög radiánban kifejezett értékével tér vissza**. A getangle a szöget az ANGBASE rendszerváltozó által meghatározott kiinduló iránytól méri az óramutató járásával ellentétes irányba növekvő értékkel. Az eredményül kapott szög radiánokban van kifejezve, és az aktuális szerkesztési síkban (az aktuális FKR XY síkja az aktuális kiemelési magasságban) értendő.

A **prompt** egy opcionális karaktersor, amelyet az AutoCAD promptként jelenít meg, a **pont** pedig az aktuális FKR egy opcionális 2D bázispontja. A szöget az AutoCAD aktuális szögmértékegységében kifejezve, a billentyűzeten begépelve is megadhatja. Jóllehet a szögek aktuális kifejezési formátuma fok, grad vagy bármely más mértékegység lehet, a függvény mindig radiánban kifejezett szögértékkel tér vissza.

getcorner

(getcorner <pont> [prompt])

A getcorner függvény, a getpoint függvényhez hasonlóan, az aktuális UCS egy pontjával tér vissza. A getcorner azonban argumentumként egy pont alappont megadását várja, és ebből a pontból a szálkereszt mozgatásával egy téglalapot rajzol ki a képernyőn. A **prompt** argumentum egy opcionális karaktersor, amely promptként fog megjelenni. Az alappont az aktuális UCS-ben van kifejezve. Ha a felhasználó 3D pontot ad meg, akkor annak Z koordinátájáról a függvény nem vesz tudomást; Z koordinátaként az aktuális kiemelési síkot fogja használni.

getfiled

(getfiled <cim> <alapertelmezes> <kiterjesztes> <flagek>)

A getfiled függvény **egy párbeszédablakot jelenít meg**, amely az elérhető és a megadott kiterjesztéssel rendelkező állományok felsorolását tartalmazza. Ennek segítségével különböző meghajtókat és könyvtárakat „átfésülve" kiválaszthat egy meglévő állományt, vagy megadhatja egy új állomány nevét. Ez a függvény egy állománynév megadását kéri a felhasználótól az AutoCAD állomány kiválasztásra szolgáló szabvány párbeszédablakának felhasználásával. A **cim** argumentum az egész párbeszédablak címét, az **alapertelmezes** az alapértelmezés szerinti állománynevet (amely üres karaktersor [" "] is lehet), a **kiterjesztes** pedig a állománynév alapértelmezés szerinti kiterjesztését (üres karaktersor [" "] esetén a kiterjesztés alapértelmezés szerint * lesz) adja meg.

getint

(getint [prompt])

Ez a függvény **arra vár, hogy a felhasználó egy egész értéket adjon meg**, és ezzel az értékkel tér vissza. Az értéknek a –32768 és +32767 közötti tartományba kell esnie. A **prompt** argumentum egy opcionális karaktersor, amely promptként fog megjelenni.

Például:

```
(setq szám (getint))
(setq szám (getint "Kerem a sugarat: "))
```

getkword

(getkword [prompt])

A getkword függvény **egy kulcsszó megadását várja a felhasználótól.** Az érvényes kulcsszavak listáját, a getkword hívása előtt, az **initget** függvény segítségével kell megadni. A **prompt** argumentum egy opcionális karaktersor, amely promptként fog megjelenni.

A getkword függvény a felhasználó válaszának megfelelő kulcsszó karaktersor értékével tér vissza. Amennyiben a felhasználó nem a kulcsszavak egyikével válaszol, az AutoCAD megismétli a kérést. Üres válasz (0) esetén (ha az megengedett) a getkword függvény nil értékkel tér vissza. Ugyancsak nil értékkel tér vissza a függvény, ha hívását nem előzte meg olyan initget hívás, amely egy vagy több kulcsszót állított be. Példa:

A következő példában az initget függvény olyan hívása szerepel, amely kulcsszavak (Igen és Nem) listáját állítja fel, valamint letíltja az üres választ (bitek értéke 1) a soron következő getkword függvény számára:

```
(initget 1 "Igen Nem")
(setq x (getkword "Biztos? (Igen vagy Nem)"))
```

A fenti programrészlet megkérdezi a felhasználót, és válaszától függően az x szimbólumot Igenre vagy Nemre állítja be. Ha a válasz egyik kulcsszónak sem felel meg vagy a felhasználó üres választ ad, akkor az AutoCAD a prompt argumentumban megadott karaktersort megjelenítve megismétli a kérdést, kiírva, hogy “Invalid option keyword”.

getorient

(getorient <pont> [prompt])

Ez a függvény a getangle függvényhez hasonló, azzal a különbséggel, hogy a getorient visszatérési szögértéke független az AutoCAD ANGBASE és ANGDIR rendszerváltozóitól. A getorient függvény a szögeket mindig úgy méri, hogy a nulla radián jobbra (Keletre) mutat, és a szögek az óramutató járásával ellentétes irányban növekszenek. A getangle függvényhez hasonlóan, a getorient a szögek visszatérési értékét radiánban fejezi ki, az aktuális szerkesztési síkban értelmezve.
Példa:

```
Command:(setq pt1 (getpoint "Valasz pontot: "))
(4.55028 5.84722 0.0)
Command: (getorient pt1 "Valasz pontot: ")
5.61582
```

getreal

(getreal [prompt])

Ez a függvény **egy valós számérték megadását várja a felhasználótól**, és ezzel az értékkel tér vissza. A **prompt** argumentum egy opcionális karaktersor, amely promptként fog megjelenni. Példa:

```
(setq xxx (getreal))
(setq xxx (getreal "Kerem az utolso erteket:"))
```

getstring

(getstring [cr] [prompt])

A getstring függvény **egy karaktersor megadását várja a felhasználótól**, és eredményként ezzel a karaktersorral tér vissza. Amennyiben a karaktersor hossza meghaladja a 132 karaktert, a függvény csak az első 132 karakterrel fog visszatérni. Amennyiben a megadott karaktersor fordtott törtjelet (\) tartalmaz, ezt a függvény két fordtott törtjellé (\\) konvertálja. Erre azért van szükség, hogy a visszatérési érték más függvények által használható állományneveket (elérési útvonallal együtt) is tartalmazzon. Amennyiben az argumentumok között a **cr** is szerepel, és értéke nem nil, akkor a megadott karaktersor szóköz karaktereket is tartalmazhat (így csak a *** billentyűvel lehet lezárni). Egyéb esetekben az input karaktersor egy szóközzel vagy a 0 billentyűvel zárható le. A **prompt** argumentum egy opcionális karaktersor, amely promptként fog megjelenni. Példa:

```
(setq s (getstring "Kerem irja be a nevet "))
```

getvar

(getvar <valtozonev>)

Ez a függvény **egy AutoCAD rendszerváltozó értékét olvassa ki**. A rendszerváltozó nevét idézőjelek közé kell zárni. Példa:

Tételezzük fel például, hogy az AutoCAD mindig, amikor lehetséges a dialóguskazettákat használja:

```
(getvar "CMDDIA")     eredménye 1
```

grdraw

(grdraw <ponttol> <pontig> <szin> [kiemeles])

A grdraw függvény **két pont közé egy vektort rajzol** az aktuális nézetablakban. A ponttól és a pontig argumentumok 2D vagy 3D pontok (két vagy három valós számból álló listák), amelyek a vektor végpontjait határozzák meg az aktuális UCS rendszerben. Az AutoCAD a vektornak a képernyőn kívülre eső részét lehagyja. A függvény a vektort az egész típusú **szin** argumentum által megadott színnel rajzolja meg; jelöli az XOR tintát, amely egy olyan „szín", mely az átrajzolt objektumok színét a kiegészítő színükre fordítja át, saját magát pedig kioltja.

grtext

(grtext [<mezo> <szoveg> [kiemeles]])

A grtext függvény az AutoCAD grafikus képernyőjének szöveges részébe ír. Ha a **mezo** argumentum értéke nulla és a legmagasabb számú képernyő menümező számánál eggyel kisebb érték közé esik, akkor a **szoveg** karaktersor argumentumot az így megadott képernyőmenü-mezőben jeleníti meg. Ha a **karaktersor** túl hosszú ahhoz, hogy a menümezőben elférjen, akkor a függvény csonkolja, ha pedig rövidebb, akkor szóközökkel egészíti ki. Ha a megadott mezőszám érvénytelen, akkor a függvény nil értékkel tér vissza.

Amennyiben az opcionális, egész típusú kiemelés argumentum adott és pozitív, a grtext kiemelten jeleníti meg a szöveget a megjelölt mezőben. A menümezőkbe először a **kiemeles** argumentum nélkül kell beírni a szöveget, és csak azután kiemelni. Egy mező kiemelése automatikusan normál módba kapcsolja bármely korábban kiemelt másik mező megjelenítését. Nullaértékű kiemelés kikapcsolja az adott menütétel kiemelését. Negatív értékű kiemelés argumentumot a függvény nem vesz figyelembe.

Ez a függvény csak megjeleníti a megadott szöveget a képernyő menüterületén, de nem változtatja meg a mögötte rejlő menütételt. Példa:

```
(setq modelen (menucmd "M=$(linelen)"))
```

initget

(initget [bitek] [karaktersor])

Ez a függvény a következő entsel, nentsel, nentselp vagy getxxx függvények (kivéve a getstring, getenv és getvar függvényeket) számára **különböző opciókat állít be**. Az initget függvény mindig nil értékkel tér vissza.

Az opcionális **bitek** argumentum egy egész szám (bit-kódolású) lehet a következő értékekkel:

Bit	Jelentés
1	Nem engedélyez üres inputot
2	Nem engedélyez nulla értéket
4	Nem engedélyez negatív értéket
8	Még bekapcsolt LIMCHECK mellett sem ellenőrzi a rajzhatárokat
16	(Jelenleg használaton kívül)
32	A „gumivonalat" vagy az ablakot szaggatott vonallal rajzolja
64	Nem engedélyezi Z koordináta inputját (csak a getdist függvényre vonatkozik)
128	Tetszőleges billentyűzet inputot is visszaad

Vigyázat: Az AutoLISP következő verziói további initget szabályzóbiteket is használhatnak, ezért ajánlott csak a táblázatban látható bitek használata.

A speciális szabályzóértékeket csak azok a getxxx függvények veszik figyelembe, amelyeknél azoknak értelmük van, amint a következő táblázatban látható:

Függvény	Kulcs-szó	1	2	4	8	32	64	128	256	512	1024
getint	x	x	x	x				x			
getreal	x	x	x	x				x			
getdist	x	x	x	x		x	x	x	x		x
getangle	x	x	x			x		x	x		x
getorient	x	x	x			x		x	x		x
getpoint	x	x			x	x		x	x	x	x
getcorner	x	x			x	x		x	x	x	x
getkword	x	x						x			
entsel	x										
nentsel	x										
nentselp	x										

A következő felsorolás részletesebben ismerteti az egyes szabályzóbiteket:

- **Érték = 1** (bit 0) Megakadályozza, hogy a felhasználó a kérésre csupán az <ENTER> lenyomásával válaszoljon.
- **Érték = 2** (bit 1) Megakadályozza, hogy a felhasználó a kérésre a nulla érték megadásával válaszoljon.
- **Érték = 4** (bit 2) Megakadályozza, hogy a felhasználó a kérésre negatív érték megadásával válaszoljon.
- **Érték = 8** (bit 3) Megengedi, hogy a felhasználó az aktuális rajzhatárokon kívül adjon meg egy pontot. Ez a feltétel akkor is érvényes a következő fel-

használói input függvényre, ha a LIMCHECK AutoCAD rendszerváltozó be van kapcsolva.

- **Érték = 16** (bit 4) Jelenleg nincs használatban.
- **Érték = 32** (bit 5) Azon függvények esetében, amelyek megengedik, hogy a felhasználó a grafikus képernyőn jelöljön ki egy pontot, e bit beállítása következtében a rajzszerkesztő (editor) a „gumivonalat" vagy ablakot folytonos helyett szaggatott vonallal rajzolja meg (egyes képernyőmeghajtók szaggatott vonalak helyett eltérő színt használnak). Abban az esetben, ha a POPUPS rendszerváltozó értéke nulla, akkor az AutoCAD figyelmen kívül hagyja ezt a bitet.
- **Érték=64** (bit 6) Megtiltja Z koordináták átadását a getdist függvénynek, ezáltal az alkalmazás biztosíthatja, hogy ez a függvény csak 2D (síkbeli) távolságokat határozzon meg.
- **Érték=128** (bit 7) Tetszőleges inputot engedélyez, mintha az egy kulcsszó volna, figyelembe véve előbb a többi szabályzóbitet és a megadott kulcsszavakat. Ez a bit elsőbbséget élvez a 0. bithez képest; amennyiben a 7. bit be van állítva, és a felhasználó az <ENTER> billentyűt nyomja le, a függvény üres karaktersorral tér vissza.

Ha az initget függvény olyan szabályzóbitet állít be, amelynek nincs értelme az ezután meghívott felhasználói input függvény esetében, akkor az a bit egyszerűen hatástalan lesz. A bitek tetszőleges kombinációban adhatók össze, és egy 0 és 255 közé eső számértéket eredményeznek. Ha a *bitek* argumentum nincs megadva, akkor az nulla (nincsenek feltételek) értékkel egyenértékű. Amennyiben a felhasználói adatbevitel nem felel meg egy vagy több megadott feltételnek (például nulla érték, amikor az nem megengedett), abban az esetben az AutoCAD üzenetet küld, és kéri az adatbevitel megismétlését.

Kulcsszavak meghatározása

1. Az opcionális karaktersor argumentum azon kulcsszavak (opciók) listáját határozza meg, amelyeket a következő entsel, nentsel, nentselp vagy getxxx függvény meg fog vizsgálni, ha a felhasználó nem a kért típusú adatot (a getpoint esetében például nem egy pontot) adja meg. Amennyiben a felhasználói input megegyezik a listában található kulcsszavak egyikével, az illető adatbeviteli függvény azzal a kulcsszóval, mint karaktersorral fog visszatérni. A felhasználói program megvizsgálhatja, hogy a kulcsszavak egyikét kapta-e eredményül, és mindegyik kulcsszó esetén más műveletet hajthat végre. Ha a felhasználó által megadott adat nem a kért típusú, és egyik kulcsszóval sem egyezik meg, akkor az AutoCAD az adatbevitel megismétlését kéri a felhasználótól.
2. A **karaktersor** argumentum értelmezése a következő szabályok szerint történik:

- A kulcsszavakat egy vagy több szóköz választja el egymástól;
  Például, a "Bal Jobb Fent" karaktersor három kulcsszót határoz meg.

- Az érvényes kulcsszavak csak betűkből, számokból és kötőjelekből (-) állhatnak.

Mindenik kulcsszó meghatározásakor előírhatja, hogy az AutoCAD annak rövidítését is elfogadja.

Az initget függvény által beállított szabályzóbitek és kulcsszavak csak a következő entsel, nentsel, nentselp vagy getxxx függvényhívásra vonatkoznak, azután automatikusan érvényüket vesztik. Ennek következtében nincs szükség újabb függvényhívásra a speciális feltételek törléséhez.

itoa

(itoa <egesz>)

Ez a függvény **az egész számot karaktersorrá alakítja át**, és ezzel a karaktersorral tér vissza. Példa:

```
(itoa 32223) eredménye ”32223”
(itoa -217) eredménye ”-217”
```

lambda

(lambda <argumentumok> <kifejezes...>)

A lambda **egy névtelen függvényt definiál**. Ezt a függvényt olyan esetben szokás használni, amikor új függvény definiálása nem indokolt. A programozó szándékát is világosabbá teszi, ha a függvény definíciója a felhasználás helyén látható. A lambda függvény az utolsó kifejezés értékével tér vissza, és gyakran használatos az apply és/vagy mapcar függvényekkel együtt, amikor egy függvényt egy listára alkalmazunk. Példa:

```
(apply  '(lambda (x y z)
      (* x (- y z))
      '(2 18 6)
)
```

vagy:

```
(setq szamlalo 0)
(mapcar  '(lambda (x)
      (setq szamlalo (1+ szamlal))
      (* x 5))
      '(1 5 11 10.2)
)
```

last

(last <lista>)

A last függvény **a lista utolsó elemével tér vissza**. A lista nem lehet nil. Példa:

```
(last '(a ”rrr” “sd” d e)) eredménye E
(last '(a b c (d e)))      eredménye (D E)
```

listp

(listp <tetel>)

Ez a függvény **T értékkel tér vissza, ha a tetel lista**, egyéb esetben pedig nil értékkel.

load

(load <fajlnev> [hibaeset])

Ez a függvény **egy AutoLISP kifejezésekből álló állományt tölt be**, és kiértékeli az abban szereplő kifejezéseket. A fajlnev egy karaktersor, amely a kiterjesztés nélküli állománynevet tartalmazza (a függvény a .lsp kiterjesztést feltételezi). A állománynév könyvtár előtagot is tartalmazhat, mint például a "/funkc/teszt1" esetében. DOS operációs rendszerek esetében megengedett a lemezmeghajtó betűszimbólumának használata, valamint fordított törtjel (\) használata is a törtjel (/) helyett.

Abban az esetben, ha a **fajlnev** karaktersorba nem foglalja bele a könyvtár előtagot, a load függvény a megadott állomány kikereséséhez a **findfile** függvénynél ismertetett módon végigjárja az AutoCAD könyvtár útvonalát. A load betölti az állományt, bárhol találta azt meg. Több azonos nevű állomány esetén, mindig a telepített AutoCAD könyvtár gyökeréhez legközelebbit tölti be.

Amennyiben a betöltés művelete sikeres volt, a load függvény az állomány utolsó kifejezésének értékével tér vissza, amely gyakran az állománybanban utolsóként definiált függvény neve. Amennyiben a load művelet sikertelen, normális esetben egy AutoLISP hibát okoz. Ha azonban a hibaeset argumentumot is megadtuk, akkor a hibajelzés helyett ennek az argumentumnak az értéke tér vissza. Ez lehetővé teszi, hogy a load függvényt hívó AutoLISP alkalmazás hiba esetén alternatív műveleteket hajtson végre.

Természetesen biztosítani kell, hogy a hibaeset argumentum különbözzön az állomány utolsó kifejezésétől, máskülönben a load által visszaadott érték nem lesz egyértelmű. Ha a hibaeset argumentum érvényes függvény, akkor kiértékelésre kerül. Ezért ennek, a legtöbb esetben, karaktersornak vagy atomnak kell lennie.

log (logaritmus naturalis)

(log <szam>)

Ez az aritmetikai függvény **a szám természetes alapú logaritmusával tér vissza**, valós érték formájában. Példa:

```
(log 5)    eredménye 1.609438
(log 1.9)  eredménye 0.641854
```

logand

(logand <szam1> <szam2> ...)

A logand függvény **a számok listáján elvégzett bitenkénti logikai AND műveletek végeredményével tér vissza**. A számoknak egész típusúaknak kell lenniük, és az eredmény is egész típusú lesz. Példa:

```
(logand 7 15 3)  eredménye 3
(logand 2 3 15)  eredménye 2
(logand 8 3 4)   eredménye 0
```

logior

(logior <egesz> ...)

A függvény a számok listáján elvégzett **bitenkénti logikai inkluzív OR műveletek végeredményével tér vissza**.

A számoknak egész típusúaknak kell lenniük, és az eredmény is egész típusú lesz.

Példa:

```
(logior 1 2 4)   eredménye 7
(logior 9 3)     eredménye 11
```

max

(max <szam1> <szam2> ...)

Ez a függvény az **adott számok közül a legnagyobb értékével tér vissza.** Mindegyik szám lehet valós vagy egész típusú. Ha az összes szám egész, akkor az eredmény is az lesz, ha viszont bármelyik szám valós, akkor az eredmény is valós szám lesz. Példa:

```
(max 122.07 -144)     eredménye 122,07
(max 8 -19 5 2)       eredménye 8.0
(max 2.1 42 8)        eredménye 42.0
```

member

(member <kifejezes> <lista>)

Ez a függvény **a kifejezés előfordulását vizsgálja a listában**, és a lista fennmaradó elemeinek listájával tér vissza, amelynek első eleme a kifejezés első előfordulása. Amennyiben a kifejezés nem fordul elő a listában, akkor a member függvény nil értékkel tér vissza. Példa:

```
(member 'd '(a b c d e))    eredménye (D E)
(member 'mmm '(a b c d e)) eredménye nil
```

min

(min <szam1> <szam2> ...)

A **min** függvény az adott számok közül **a legkisebb értékével** tér vissza. Mindegyik szám lehet valós vagy egész típusú. Ha az összes szám egész, akkor az eredmény is az lesz, ha viszont bármelyik szám valós, akkor eredmény valós szám lesz. Példa:

```
(min 7 -80 3 -10.0)   eredménye -80.0
(min 7 2 78 1)        eredménye 1
```

minusp

(minusp <tetel>)

Ez a függvény T értékkel tér vissza, amennyiben a **tetel** egész vagy valós típusú, és kiértékelése negatív eredményt ad, egyébként a függvény értéke nil lesz. A tétel más típusú értékeire a függvény nem definiált. Példa:

```
(minusp -1)      eredménye T
(minusp -14.23)  eredménye T
(minusp 81.2)    eredménye nil
```

nentsel

(nentsel [prompt])

Ezzel a függvénnyel **a beillesztett** (egy Block elemeihez) **rajzelemek definíciós adataihoz lehet hozzáférni.**

A nentsel függvény kéri a felhasználót, hogy válasszon ki egy objektumot. További szolgáltatásként a Command: promptnál, a nentsel függvény az előző initget hívás által meghatározott kulcsszavakat is elfogadja.

Ha megadja az opcionális prompt argumentumot, akkor annak karaktersornak kell lennie. Ha elhagyja, akkor a szabvány Select objects: prompt jelenik meg.

Amennyiben a választott objektum nem összetett rajzelem (nem Polyline vagy Block), a nentsel az entsel függvénnyel azonos értékkel tér vissza. Ha azonban a kiválasztott rajzelem Polyline, akkor a nentsel listája az alárendelt rajzelem (Vertext) nevét és a kiválasztás pontját tartalmazza. Ez hasonlít az entsel visszatérési listájához, azzal a különbséggel, hogy a Polyline fejrésze helyén a kiválasztott csomópont neve szerepel. A nentsel függvény mindig a Polyline kiválasztott szakaszának kezdőpontjával tér vissza.

not

(not <tetel>)

Ez a függvény **T értékkel tér vissza, amennyiben a tetel értéke nil**, egyéb esetekben nil értékkel. Rendszerint a null függvény használatos a listák, a not pedig minden egyéb adattípus vizsgálatára, néhány más típusú vezérlő függvénnyel együtt. Példa:

```
(setq a 12)
(setq b "karaktersor")
(setq c nil)
```

Ebben az esetben:

```
(not a)    eredménye nil
(not b)    eredménye nil
(not c)    eredménye T
```

null

(null <tetel>)

Ez a függvény **T értékkel tér vissza, amennyiben a tetel-hez rendelt érték nil**, egyéb esetekben nil értékkel.

numberp

(numberp <tetel>)

A függvény **T értékkel tér vissza, ha a tetel valós vagy egész típusú szám**, egyébként nil értékkel.

open

(open <fajlnev> [mod])

Ez a függvény megnyit egy állományt az AutoLISP I/O függvényei általi hozzáférésre. A függvény az állományleíróval tér vissza, amelyet az I/O függvények fognak felhasználni, így ezt a setq függvénnyel hozzá kell rendelni egy szimbólumhoz. Példa:

```
(setq a (open "fajl.kit" "r"))
```

A fajlnév argumentum egy olyan karaktersor, amely a megnyitni kívánt állomány nevét és kiterjesztését adja meg. A mod argumentum az írás/olvasás címkéjét tartalmazza, amely csak egyetlen kisbetűből álló karaktersor lehet. A módok érvényes betűjeleit a következő táblázatban foglaltuk össze.

Az open függvény mod opciói:

- "r" - Olvasásra nyitja meg az állományt. Amennyiben az állománynév nem található, a függvény nil értékkel tér vissza.
- "w" - Írásra nyitja meg az állományt. Amennyiben az állománynév nem létezik, a függvény egy új állományt hoz létre, és azt nyitja meg.
- "a" - Hozzáfűzésre nyitja meg az állományt. Amennyiben a állománynév nem létezik, a függvény egy új állományt hoz létre, és azt nyitja meg.

or (vagy)

(or <kifejezes...>)

Ez a függvény a kifejezések listáján elvégzett **logikai OR művelet** végeredményével tér vissza. Az or függvény a kifejezéseket balról jobbra haladva értékeli ki, eközben egy nem nil kifejezést keres. Ha talált egyet, az or függvény abbahagyja a további kiértékelést, és T értékkel tér vissza. Amennyiben mindegyik kifejezés nil értékű, az or függvény nil értékkel tér vissza. Példa:

```
(or nil 45 '())  eredményeT
(or nil ())      eredménye nil
```

pi

Ez egy állandó, a matematikai pi nagyon pontos megközeltése. Értéke megközelítőleg 3.1415926.

polar

(polar <pont> <szog> <tavolsag>)

A polar függvény az UCS **pont** pontjától a **szog** szögre és **tavolsag** távolságra lévő 3D pontjának értékét adja eredményül. A szög az X tengelytől mérve, radiánokban értendő, és az óramutató járásával ellentétes irányban növekszik. Bár a pont lehet 3D pont, a szöget a függvény mindig az aktuális szerkesztési síkra vonatkoztatja.

quit

(quit)

A quit **függvény kilépést hajt végre az aktuális alkalmazásból**. A quit függvény hívásának eredményeképpen a quit/exit megszakítás hibaüzenet, majd az AutoCAD **Command:** promptja jelenik meg.

quote

(quote <kifejezes>)

A **kifejezés kiértékeletlen formájával tér vissza**. Ez a következőképpen is írható:

```
(quote a)        eredménye  A
(quote cat)      eredménye  CAT
(quote (a b))    eredménye  (A B)
'a               eredménye  A
'cat             eredménye  CAT
'(a b)           eredménye  (A B)
```

read

(read <karaktersor>)

Ez a függvény a **karaktersor-ból vett első listával vagy atommal tér vissza.** A karaktersor argumentum nem tartalmazhat szóközöket, kivéve, ha azok egy lis-

tán vagy karaktersoron belül fordulnak elő. A read függvény az argumentumát a megfelelő adattípussá konvertálja. Példa:

```
(read "hello")              eredménye atom  HELLO
(read "he te")              eredménye karaktersor  HE TE
(read "\"Kis Ibolya\"")     eredménye karaktersor "Kis
                            Ibolya"
(read "(a b c)")            eredménye lista  (A B C)
(read "(a b c) (d)")        eredménye lista  (A B C)
(read "7.300")              eredménye valós szám  7.3
```

read-char

(read-char [fajlleiro])

A read-char függvény **egyetlen karaktert olvas be a billentyűzet input pufferéből**, vagy a fajlleiro argumentummal meghatározott, előzőleg megnyitott állományból. A függvény a beolvasott karakter decimális (egész típusú) ASCII kódjával tér vissza.

Ha nem ad meg fajlleiro argumentumot, és a billentyű pufferben sincs karakter, akkor a read-char függvény addig vár, amíg a klaviatúra egyik billentyűjét le nem nyomja (<ENTER> billentyűvel lezárva).

read-line

(read-line [fajlleiro])

A függvény **egy karaktersort olvas be a billentyűzetről**, vagy a **fajlleiro** argumentummal meghatározott nyitott állományból. Amennyiben a read-line függvény a "állomány vége" jellel találkozik, nil értékkel tér vissza, egyébként pedig a beolvasott karaktersorral.

regapp

(regapp <alkalmazas>)

Ez a függvény **egy, az aktuális AutoCAD rajzhoz tartozó alkalmazás nevét jegyzi be** (regisztrálja). Az alkalmazások által definiált bővített elemadatok csoportosításának, tárolásának, lekérdezésének és módosításának alapja az alkalmazásnév. Egy alkalmazás a bővített elemadatok szervezéséhez annyi alkalmazásnevet tarthat nyilván, amennyi csak szükséges.

Amennyiben az alkalmazás neve azonos egy már bejegyzett névvel, a függvény nil értékkel tér vissza, ellenkező esetben az alkalmazás nevével.

Amennyiben a regisztrálás sikerrel jár, az alkalmazás bekerül az APPID szimbólumtáblába. Ez a tábla a rajzban bővített elemadatokat használó alkalmazások listáját tartalmazza. Ez lehetővé teszi, hogy egy alkalmazás a saját bővített elemadatait megkülönböztesse más alkalmazásokétól. A bővített elemadatokat az AutoLISP nyelven vagy AutoCAD Fejlesztői Rendszerben (ADS) készült alkalmazások egyaránt kiegészíthetik vagy módosíthatják.

Az alkalmazás argumentuma egy maximum 31 karakter hosszú karaktersor, amelyre, a szimbólumokra (úgy mint a táblanevekre) érvényes elnevezési szabályok vonatkoznak. Az alkalmazásnév tartalmazhat betűket, számokat, valamint speciális $ (dollár), - (kötőjel) és _ (aláhúzás) karaktereket, de nem tartalmazhat szóközt. A névben szereplő betűket az AutoCAD nagybetűkké alakítja át. Példa:

```
(regapp "ADESK_415")
(regapp "USER1-v2.5-124")
```

rem

(rem <szam1> <szam2> ...)

Ez a függvény a **szam1 értéket elosztja a szam2 értékkel, és az osztás (szam1 mod szam2) maradék értékével tér vissza**. A rem függvény a kiértékelés szokásos szabályai szerint egész vagy valós számokkal is használható. Példa:

```
(rem 43 12)      eredménye 7
(rem 13.0 570)   eredménye 11.0
(rem 60 2)       eredménye 0
```

reverse

(reverse <lista>)

Ez a függvény **a lista elemeinek sorrendjét megfordítja**, és a megfordított sorrendű listával tér vissza. Példa:

```
(reverse '((a) b c)) eredménye (c b (a))
```

rtos

(rtos <szam> [mod] <pontossag>)

Ez a függvény a szám (valós) **argumentumban szereplő számjegyek karaktersorával tér vissza,** melynek formátuma a mod, a pontosság, valamint az AutoCAD UNITMODE rendszerváltozójának és DIMZIN méretváltozójának beállításától függ. A mód és a pontosság egész számok, amelyek a választott hossz-

mértékegységet és a pontosságot határozzák meg. Az értelmezhető mód beállítások a következők:

Hosszmértékegységek értékei:

Mód értéke (mod)	Karaktersor formátuma
1	Tudományos
2	Tizedes
3	Mérnöki (láb és tizedes hüvelyk)
4	Építészeti (láb és a hüvelyk törtrésze)
5	Tört

A **mod** és a **pontossag** argumentumok megfelelnek az AutoCAD LUNITS illetve LUPREC rendszerváltozóinak. Amennyiben az argumentumokat elhagyja, az rtos függvény a LUNITS és LUPREC rendszerváltozók aktuális értékét fogja alapul venni.

A UNITMODE rendszerváltozónak csak akkor van hatása a visszatérő karaktersorra, ha mérnöki, építészeti vagy tört mértékegységet választ (a mód értéke 3, 4 vagy 5).

set

(set <szimbolum> <kifejezes>)

Ez a függvény **a szimbolumhoz a kifejezes értékét rendeli hozzá** (ahol a szimbólum egy aposztrófált szimbólumnév), és ezzel az értékkel tér vissza. Példa:

```
(set 'a 2.12)          eredménye 2.12 és beállítja az a
                       szimbólumértékét
(set (quote b) 'a)     eredménye a és beállítja a b
                       szimbólum értékét.
```

Ha a set függvényt egy aposztróf nélküli szimbólumnévre alkalmazza, akkor közvetett módon új értéket adhat egy másik szimbólumnak. A fenti példa alapján:

```
(set b 640)            eredménye 640 és a 640 értéket az
                       a szimbólumhoz rendeli hozzá (mi
                       vel a b szimbólum ezt tartalmaz
                       za).
```

setq

(setq <szimbolum1> <kifejezes1> [<szimbolum2> <kifejezes2>] ...)

Ez a függvény a szimbólum1 szimbólumhoz a kifejezés1, a szimbólum2 szimbólumhoz a kifejezés2 értéket rendeli hozzá, és így tovább. Ez az AutoLISP alapvető értékadó függvénye. Egyetlen setq függvényhívással több szimbólumnak is adhat értéket, de a függvény csak az utolsó kifejezés értékével tér vissza.

A **set** és setq függvények által közvetlenül a szimbólumokhoz legfeljebb 132 karakter hosszúságú karaktersorok rendelhetők. Azonban a strcat függvénnyel több karaktersort összefűzve, az eredmény hozzárendelhető egy szimbólumhoz.

A setq megegyezik a set függvénnyel, azzal a különbséggel, hogy a szimbólumnév ez esetben nincs aposztrofálva. Más szóval, a set kiértékeli az első argumentumot, a setq pedig nem. A következő példa a két függvény hasonlóságát mutatja:

setq = set quota

`(setq a 5.0)`. egyenértékű kifejezése `( set ( quote a) 5.0)`

A set és a setq függvények globális szimbólumokat hoznak létre, illetve módosítanak, kivéve, ha defun függvényen belül, a függvény argumentumának vagy lokálisnak deklarált szimbólumának értékadására használja őket.

A set és setq függvények arra is alkalmasak, hogy az AutoLISP beépített szimbólumainak és függvényeinek új értéket adjon elvetve, vagy hozzáférhetetlenné téve az eredeti jelentésüket. Ezért ne használjuk a beépített szimbólumok vagy függvények nevét saját szimbólumaiknak neveként!

Ha nem bizonyos a kívánt szimbólumnév eredetisége, akkor ez leellenőrizhető ha beírjuk AutoCAD parancssorába:

`Command:(atoms-family 0 '("enszimb"))...` ahol az **`enszimb`** `a vizsgálandó szimbólum név.`

Ha az visszafordított érték nil, akkor ez a szimbólum még nincs definiálva.

setvar

(setvar <valtozonev> [ertek])

Ez a függvény a változónév nevű **AutoCAD rendszerváltozót az adott értékre állítja be**, és ezzel az értékkel tér is vissza. A rendszerváltozó nevét idézőjelek közé kell tenni. Példa:

```
(setvar "FILLETRAD" 0.10)
```

A felső sor eredménye 0.1 az AutoCAD lekerekítési sugarát 0.1 egységre állítja be.

Egész típusú értékkel rendelkező rendszerváltozók esetében a megadott értéknek –32768 és +32767 közé kell esnie.

Bizonyos AutoCAD parancsok, még mielőtt bármilyen promptot kiadnának, a rendszerváltozóktól vesznek át értékeket. Ha a setvar függvény segítségével a folyamatban lévő parancs használata során ad új értéket ezen rendszerváltozóknak, akkor az újonnan beállított érték csak, a következő AutoCAD parancs használatakor fejti ki hatását.

sin

(sin <szog>)

Ez a függvény **a radiánokban kifejezett szög valós számmal kifejezett szinuszával** tér vissza. Figyelem: az argumentum mindig legyen radiánban, egyébként hibás eredményeket fogunk kapni! Példa:

```
(sin 1.0)  eredménye 0.841471
(sin 0.0)  eredménye 0.0
```

sqrt

(sqrt <szam>)

Ez a függvény a **szam négyzetgyökének valós számmal kifejezett értékével tér vissza**. Negatív **szam** esetén, “Bad argumentum” hibaüzenetet kapunk.
Például:

```
(sqrt 9)         eredménye 3.0
(sqrt 144.0)     eredménye 12.0
```

strcase

(strcase karaktersor [melyik])

A strcase függvény **a karaktersor argumentumában megadott karaktersor másolatával tér vissza**, amelyben a betűkaraktereket a **melyik** argumentumnak megfelelően nagy- vagy kisbetűkké alakítja át. Amennyiben a melyik argumentum nincs megadva vagy az értéke nil, a karaktersor minden betűjét nagybetűvé alakít-

ja. Ha a **melyik** argumentum is szerepel a függvényhívásban, és értéke nem nil, akkor az összes betű kisbetű lesz. Példa:

```
(strcase "Mint a")    eredménye "MINT A"
(strcase "Minta" T)   eredménye "minta"
(strcase "Minta" nil) eredménye "MINTA"
```

strlen

(strlen [karaktersor]...)

Ez a függvény **a karaktersor egész értékben kifejezett hosszával**, azaz a benne szereplő karakterek számával tér vissza. Amennyiben több karaktersor argumentumot ad meg, az argumentumokban szereplő összes karakter összege lesz a visszatérési érték. Ha elhagyja az argumentumokat, vagy üres karaktersort ad meg (amint azt az utolsó példa mutatja), akkor a függvény a 0 (nulla) egész számmal tér vissza.

```
(strlen "abcde")              eredménye  5
(strlen "abc")                eredménye  3
(strlen "abcde" "abc" "ab") eredménye  10
(strlen)                      eredménye  0
```

substr

(substr <karaktersor> <kezdo> [hossz])

Ez a függvény a **karaktersornak** azzal **a rész-karaktersorával tér vissza**, amely a **kezdo** karakterpozíciónál kezdődik, és hossz számú karaktert tartalmaz. Amennyiben a hossz argumentumot nem adja meg, a rész-karaktersor a **karaktesor** végéig fog tartani. A **kezdo** (és amennyiben megadja, a hossz) értékeinek pozitív egész számoknak kell lenniük.

Fontos megjegyezni, hogy a karaktersor első karaktere az 1-es számú karakter. Ez különbözik a listaelemekkel foglalkozó összes többi függvénytől (mint az nth, ssname, és így tovább), amelyek az első elemet a 0 sorszámmal kezelik.

Példa:

```
(substr "abcdefgh" 2)       eredménye "bcdefgh"
(substr "abcde" 2 2)        eredménye "bc"
(substr "abcdefgh" 3 3)     eredménye "cde"
```

tblnext

(tblnext <tablanev> [vissza])

Ezt a függvényt **egy teljes szimbólumtábla végigpásztázására** használhatja. Az első argumentuma egy karaktersor, amely a kívánt szimbólumtáblát azonosítja. A lehetséges táblanevek:

"LAYER", "LTYPE", "VIEW", "STYLE", "BLOCK", "UCS", "APPID", "DIMSTYLE" és "VPORT".

A karaktersornak nem szükséges nagybetűkből állnia.

Ha a tblnext függvényt egymás után többször használja, akkor az normális esetben mindig a megadott tábla következő elemével tér vissza. (A későbbiekben ismertetett tblsearch függvény segítségével beállítható a következő lekérdezni kívánt táblabejegyzés.) Ha azonban a **vissza** argumentumot is megadta, és kiértékelése nil-től különböző értéket ad, akkor a függvény visszatér a szimbólumtábla elejére, és annak első bejegyzését olvassa ki. Amikor a táblában nincs több bejegyzés, a függvény nil értékkel tér vissza. A függvény törölt táblabejegyzéssel soha nem tér vissza.

Amennyiben megtalálta a keresett bejegyzést, a függvény egy DXF típusú kódokat és értékeket tartalmazó kapcsolt értékpárokból álló listával tér vissza, amely hasonló az entget függvény által szolgáltatott listához. Példa:

```
(tblnext "layer" T)   lekérdezi az első réteget
```

eredményül az alábbiakkal térhet vissza:

```
(0 . "LAYER")         szimbólum típusa
(2 . "0")             szimbólum neve
(70 . 0)              flagek
(62 . 7)              színszám, ha ki van kapcsolva,
                      negatív
(6 . "CONTINUOUS")    vonaltípus neve
```

Az AutoCAD megjegyzi az egyes táblákból utoljára kiolvasott bejegyzést, és a tblnext függvénynek ugyanarra a táblára vonatkozó minden újabb hívásakor a következő bejegyzéssel tér vissza. Amikor egy tábla végigpásztázását elkezdi, egy nem nil értékű második argumentummal biztosítsa a tábla visszapörgetését, hogy a függvény az első bejegyzéssel térjen vissza.

A Block táblából kiolvasott bejegyzések egy 2-es csoportot, és a Block definíciójában szereplő első rajzelem elemnevét tartalmazzák (amennyiben létezik ilyen).

Azaz, ha adott egy HAZ nevű Block:

```
(tblnext "block")     lekérdezi a Block definícióját
```

Eredményül:

```
((0 . "BLOCK")              szimbólum típusa
(2 . "HAZ")                 szimbólum neve
(70 . 0)                    flagek
(10 9.0 2.0 0.0)            origó X, Y, Z koordinátái
(-2 . <Elem neve: 40000126>)     az első rajzelem
```

A 2-es csoportban szereplő elemnevet az entget és az entnext függvények elfogadják, de a többi rajzelem kezelő függvény nem. Nem illesztheti be például egy kiválasztott elemsorozatba az ssadd függvény segítségével. A 2-es csoportban található elemnevet átadva az entnext függvénynek, végigpásztázhatja a Blockdefinícióban szereplő rajzelemeket; a Block-definícióban lévő utolsó elem után az entnext függvény nil értékkel tér vissza.

Megjegyzés: Amennyiben egy Block nem tartalmaz rajzelemeket, a tblnext függvény eredményében szereplő –2-es csoport a Block Endblk elemének nevét tartalmazza.

tblsearch

(tblsearch <tablanev> <szimbolum> [kovetkezot])

Ez a függvény végigvizsgálja a **tablanev** argumentummal megadott táblát a szimbólum argumentummal (a tblnext függvényhez hasonlóan) megadott szimbólumnév kikeresése céljából. Mindkét argumentum betűit a függvény automatikusan nagybetűkké konvertálja. Amikor talált egy szimbólumot a megadott névvel, a függvény a tblnext függvénynél ismertetett formátumú listával tér vissza. Ha nem talált, akkor nil értékkel tér vissza. Példa:

```
(tblsearch "style" "standard")   lekérdez egy szövegstí
                                 -lust
((0 . "STYLE")                   szimbólum típusa
(2 . "STANDARD")                 szimbólum neve
(70 . 0)                         flagek
(40 . 0.0)              rögzített magasság
(41 . 1.0)              szélességi tényező
(50 . 0.0)              dőlésszög
(71 . 0)                generálási flag
(3 . "txt.shx")         elsődleges font állomány
(4 . ""))               big-font állomány
```

Normális esetben a tblnext függvény által lekérdezett táblabejegyzések sorrendjére a tblsearch függvény nincs kihatással. Ha azonban a tblsearch művelet sikeres volt, és a következő argumentum van jelen nil-től különböző értékkel, akkor a tblnext bejegyzésszámlálója úgy módosul, hogy a következő tblnext hívás a jelen tblsearch hívással lekérdezett bejegyzés utáni bejegyzéssel fog visszatérni.

terpri

(terpri)

A terpri függvény **egy újsor karaktert ír ki a képernyőre** (sort emel), és nil értékkel tér vissza. A terpri függvénnyel nem lehet állományba írni.

trans

(trans <pont> <kiindulo> <cel> [elmozdulas])

Ez a függvény **egy pontot (vagy egy elmozdulást) transzformál át az egyik koordináta-rendszerből egy másikba**. A **pont** argumentum három valós értékből álló lista, amely jelenthet egy 3D pontot, vagy egy 3D elmozdulást (vektort). A kiinduló argumentum annak a koordináta-rendszernek a kódja, amelyben a pont meg van adva, a **cel** pedig annak a koordináta-rendszernek a kódja, amelybe a pontot transzformálni kívánja. Az opcionális elmozdulás argumentum, amennyiben jelen van és értéke nem nil, azt jelzi, hogy a pont argumentumot nem pontként, hanem 3D elmozdulásként kell értelmezni. A **kiindulo** és a **cel** argumentum a következő értékek valamelyikét veheti fel:
Egy egész érték az alábbiakból.

Koordináta-rendszerek kódjai:

0 Világ (WCS)
1 Felhasználói (aktuális UCS)
2 Képernyő (DCS)

A trans függvény a megadott cél koordináta-rendszerben kifejezett 3D ponttal (vagy elmozdulással) tér vissza.

type

(type <tetel>)

Ez a függvény **a tetel típusával tér vissza**, ahol a típus az alább felsoroltak (atomok) egyike lehet. Azon tételek esetében, amelyek kiértékelése nil eredményt ad (például azon szimbólumok esetében, amelyekhez nincs érték rendelve), a függvény nil értékkel tér vissza.

Szimbólum típusok:

REAL	Valós számok;
FILE	Állományleírók;
STR	Karaktersorok;
INT	Egész számok;
SYM	Szimbólumok;
LIST	Listák (és felhasználói függvények);
SUBR	Belső függvények;
EXSUBR	Külső (ADS) függvények;
PICKSET	Kiválasztott elemsorozatok;
ENAME	Elemnevek;
PAGETB	Függvények laptáblája.

Példa:

```
(setq a 123 r 3.45 s "Helló:" x '(a b c))
(setq f (open "név" "r"))
```

Ebben az esetben:

```
(type 'a)        eredménye  SYM
(type a)         eredménye  INT
(type f)         eredménye  FILE
(type r)         eredménye  REAL
(type s)         eredménye  STR
(type x)         eredménye  LIST
(type +)         eredménye  SUBR
(type setq)      eredménye  SUBR
(type nil)       eredménye  nil
```

vports

(vports)

Ez a függvény az aktuális nézetablak-konfiguráció **nézetablak-leíróinak listájával tér vissza**. Mindegyik nézetablak-leíró a nézetablak azonosítószámát, valamint a nézetablak bal alsó és jobb felső sarokpontjának pozícióját tartalmazza.

Amennyiben az AutoCAD TILEMODE rendszerváltozója 1-re van állítva, a lista az AutoCAD **Viewport** parancsával létrehozott nézetablak-konfigurációt írja le. A nézetablakok sarkait 0.0 és 1.0 közötti értékekkel fejezi ki, ahol a (0.0 0.0) a képernyő grafikus területének bal alsó, az (1.0 1.0) pedig a jobb felső sarkát képviseli. Ha a TILEMODE értéke 0 (ki), akkor a visszatérő lista az **mview** paranccsal létrehozott nézetablak a rajzelemek leírását tartalmazza. A nézetablak rajzelemek sa-

rokpontjai Paper Space koordinátákban vannak kifejezve. Kikapcsolt (0) TILEMODE esetén az 1-es számú nézetablak mindig a papírtérben van.

Egyetlen nézetablakból álló konfiguráció és TILEMODE 1 értéke esetén a vports függvény az alábbi eredménnyel térhet vissza:

```
((1 (0.0 0.0) (1.0 1.0)))
```

Hasonlóképpen, ha adott a képernyő négy sarkában elhelyezkedő négy, egyenlő nagyságú nézetablakból álló konfiguráció, és a TILEMODE be van kapcsolva, akkor a vports függvény a következő eredménnyel térhet vissza:

```
((5 (0.5 0.0) (1.0 0.5)) (2 (0.5 0.5) (1.0 1.0)) (3
(0.0 0.5) (0.5 1.0)) (4 (0.0 0.0) (0.5 0.5)))
```

A listában mindig az aktuális nézetablak leírója szerepel az első helyen. A fenti példában az 5-ös számú az aktuális nézetablak.

write-char

(write-char <szam> [fajlleiro])

A write-char függvény **egy karaktert ír ki a képernyőre, vagy a fajlleiroval azonosított nyitott állományba**. A szám a kiírni kívánt karakter decimális ASCII kódja; a függvény ezzel az értékkel tér is vissza.

xload

(xload <alkalmazas> [hibaeset])

Ez a függvény egy **AutoCAD Fejlesztői Rendszerben (ADS) készült alkalmazást tölt be**. Amennyiben az alkalmazás betöltése sikeres volt, a függvény annak nevével tér vissza, egyébként hibaüzenetet küld. A függvény hibát jelez, ha egy, már betöltött alkalmazással próbálkozunk.

Az alkalmazás megadható a futtatható állomány nevét tartalmazó, idézőjelbe tett karaktersorral, vagy változóval. Az állomány betöltésekor az AutoLISP ellenőrzi, hogy az ADS alkalmazás érvényes-e, valamint kompatibilis-e az aktuális ADS rendszerrel és az éppen használt AutoLISP verzióval.

`(xload "/myown/app1")` ha sikeres, eredménye `"/myown/app1"`

Amennyiben az xload művelet sikertelen, normális esetben egy AutoLISP hibát okoz. Ha azonban a hibaeset argumentumot is megadta, akkor a hibajelzés helyett ennek az argumentumnak az értéke tér vissza. Az xload függvénynek ez a tulajdonsága megegyezik a load függvényével.

xunload

(xunload <alkalmazas> [hibaeset])

Ez a függvény **egy ADS alkalmazást töröl a memóriából**. Amennyiben az alkalmazás törlése sikeres volt, annak nevével tér vissza, egyébként hibaüzenetet küld. Az alkalmazásnévnek pontosan meg kell egyeznie azzal, amit az alkalmazás betöltésekor megadott. Az xload függvényhívásban használt elérési útvonal (könyvtárnév) elhagyható.

Például, a következő függvényhívás sikeresen törli az xload függvénnyel az előbb betöltött alkalmazást.

```
(xunload "myapp") ha sikeres, eredménye "myapp"
```

Amennyiben az xunload művelet sikertelen, ez esetben egy AutoLISP hibát okoz. Ha azonban a **hibaeset** argumentumot is megadta, akkor a hibajelzés helyett ennek az argumentumnak az értéke tér vissza. Az xunload függvénynek ez a tulajdonsága megegyezik a load függvényével.

zerop

(zerop <tetel>)

Ez a függvény **T értékkel tér vissza, ha a tétel valós vagy egész típusú,** és kiértékelésének eredménye nulla. Minden egyéb esetben nil értékkel tér vissza. Más típusú tételekre a függvény nem definiált. Példa:

```
(zerop 0)        eredménye T
(zerop 0.0)      eredménye T
(zerop 0.00001)  eredménye nil
```

1.18. Függvények ábrázolása AutoLISP segítségével

Az AutoLISP egyik lehetséges alkalmazása a 2D vagy 3D függvények ábrázolása. Ez nem csak a vizualizálás érdekében fontos, hanem a létrehozott rajzelemeket alkalmazhatjuk bonyolultabb rajzelemek létrehozására. Egyik ilyen lehetséges példa az evolvens generálása, majd ennek felhasználása felületek generálására.

A grafikai, és nem csak, megjelenítés egyik lehetséges megoldása a line használata. Ez azt jelenti, hogy egy görbét egyenes szakaszokkal kell megközelítenünk. A megközelítés, lényegében, bármilyen finomsággal megtörténhet. Nyilvánvaló, hogy ha egy görbét egyenes szakaszokkal akarunk megközelíteni minél rövidebbek az egyenes szakaszok annál pontosabb lesz a görbe. Gyakorlatilag ezt a finomítást az elképzelhető műszaki méretek alatt tehetjük meg(10^{-15} mm). Gondoljunk arra, hogy ha csak a kijelzési pontosságot használjuk ki (5 tizedes), és egy

milliméteres egységekben gondolkozunk, akkor képesek vagyunk akár 10^{-5} mm-es egyenes szakaszokkal is megközelíteni. Már az ekkora pontosság is kielégítő lenne műszaki szempontból, de ez messze felülmúlható, ha az AutoCAD belső pontosságát használjuk.

Ha a létrehozott új rajzelemek további felhasználhatóságát tekintjük, akkor a line művelet helyett célszerűbb lenne a polyline. Ezt a rajzelemet a további műveletekben könnyen használhatjuk akár bonyolult felületek generálására is.

A további kérdés az lenne, hogy akkor miként kellene ezt elképzelni egy AutoLISP programban. A továbbiakban ezt vizsgáljuk meg vázlatosan.

Adott egy másodfokú függvény: **$y = ax^2 + bx + c$.**

Írjunk egy AutoLISP programot, amely kiszámolja az **$ax^2 + bx + c = 0$** egyenlet gyökeit, ha valósak, ha pedig komplexek, kiírja azt. Továbbá ábrázoljuk a függvényt úgy, hogy a szélsőértéke látható legyen.

Első lépésben szükségünk van az a, b, c valós számokra. Ezeket egyenként a getreal függvénnyel és egy karaktersorral: “Kerem az ”a” együtthatót” (ugyanígy kérjük be a b és c együthatókat is). Ezeket az **adatok** nevű függvényben, foglaljuk egybe:

```
(defun adatok ()
      (getreal "Kerem az a egyutthatot:")
      ........
)
```

Az adatok tulajdonában készíthetünk, **második lépésben**, egy függvényt, amely kiszámolja az egyenlet gyökeit, ha ezek valósak.
Ehhez definiálunk egy újabb függvényt:

(defun **gyokok** ()
      (setq delta (... itt kiszámoljuk a $b^2 – 4ac$ -t)

Ezután eldöntjük, a **delta** függvényében, hogy milyen típusú gyökei vannak az egyenletnek, és ha valósak, akkor kiszámoljuk és kiírjuk a képernyőre őket.

```
      (cond
            ((> delta 0)
                  (... x1, x2 ... =)
                  (print x1)
                  (print x2))
            ((= delta 0)
```

```
                (x1 = x2 = ….)
                (print x1=x2=…))
          ((< delta 0)
                (print "Komplex gyokok!"))
    )
```

Majd kiszámoljuk a függvény szélsőértékét is:

```
(setq szelsoertek (- (/ b (* 2 a))))
)
```

A harmadik lépésben, készítünk egy AutoLISP függvényt, amely bizonyos határok között kirajzolja az ábrázolni kívánt függvényt. Először is meg kell állapítanunk, hogy a függvényt milyen határok között akarjuk ábrázolni. Ezeket a határokat úgy kell megválasztani, hogy a függvény (a mi esetünkben egy parabola) csúcsa, ahol a szélsőértéke is található, valahol középen legyen. Ezt úgy tudjuk megoldani, hogy eldöntjük, hogy mekkora legyen a függvény ábrázolási hossza az x tengelyen: xhossz. Ennek alapján kiszámoljuk az xbal és xjobb értékeit, a határokat, amelyek között meg akarjuk jeleníteni a függvényt úgy, hogy a szélsőérték az xbal és xjobb között legyen:

xbal = szelsoertek – (hossz/2), illetve **xjobb = szelsoertek + (hossz/2)**

Szükségünk van továbbra a függvény pontjainak a megállapítására, valamilyen lépéstávolságokra. Ezért bekérünk a getreal függvény segítségével egy **lepes** nevű változót.

A továbbiakban figyelembe véve, hogy a függvényünk ábrázolását polyline segítségével akarjuk megoldani, szükség lesz az egymás utáni pontok kiszámítására. Ezt egy **while ciklus** segítségével tudjuk a legkönnyebben megoldani. Ha egy ciklust készítünk, amely az xbal értéktől az xjobb értékig működik, mindig tesztelve, hogy az x koordináta kisebb vagy egyenlő legyen mint az xjobb, akkor már meg is kapjuk a ciklus tesztkifejezését. A ciklust tovább léptetni pedig úgy tudjuk, hogy az x koordináta értékét növeljük, úgy hogy egy **lepestav**-nyi értéket adunk hozzá (a ciklus minden lépésében).

Minden lépés során az éppen aktuális x értékre kiszámoljuk a függvény értéket egyszerűen az $\mathbf{y = ax^2 + bx + c}$ függvénybe helyettesítve az éppen aktuális x értéket. **Az így kapott y érték és az aktuális x érték éppen a függvényünk egy pontját képezi!** További feladatunk az lesz, hogy miként tartjuk meg a lépésenként kiszámított pontokat. Ehhez készítünk egy listát, amelyhez minden lépésben hozzáadunk egy x, y pontpárt. A listának tudjuk, hogy valós számokat kell tartalmazniuk, és mivel síkfüggvény a parabola, csak két valós számpárt.

Ahhoz, hogy a polyline rajzolásához listánkat használni tudjuk, a következő formájú kell legyen:

```
((x1 y1) (x2 y2) .....(xk yk) ..... (xn yn))
```

Megfigyelhető a lista felépítése: listák valós számpárokkal egy listán belül.

Ezt legkönnyebben a **cons** függvény segítségével oldhatjuk meg, amint majd azt látni fogjuk az alábbiakban. Ha a listát megépítettük, akkor még hátra van a kirajzolás megoldása. A már megépített listát, az AutoCAD polyline utasítást és a **foreach** AutoLISP függvényt használva, az alábbi programsorban leírtak alapján oldhatjuk meg a feladatot:

```
(defun rajzolas ()
      (setq xbal..........)
      (setq xjobb..........)
      (setq lepes (getreal "Kerem a lepest:"))
(setq x xbal)
      (while (< x xjobb)
            (setq y =……..)
            (setq pontlista (cons (list x y) lista))
            (setq x (+ x lepes))
      )
      (command "pline" (foreach pont pontlista (command
pont)))
      (setq pontlista (reverse pontlista))
)
```

Az utolsó programsorok eredménye: a pontlista nevű változóban tárolt pontok alapján egy polyline rajzolódik ki, ami éppen a kért függvény ábrája.

Az **adatok, gyokok** és **rajzolás** saját függvényeket egy újan definiált függvénybe foglaljuk össze, hogy ne keljen egyenként futtatni őket.

```
(defun fuggvenyabrazolas (/ xbal xjobb szelsoertek
pontlista)
      (adatok)
      (gyokok)
      (rajzolas)
)
```

Látható, hogy az összefoglaló függvénynek helyi változói is vannak.

Rendkívül fontos, hogy a függvény újrafuttatása során be nem kért adatokat visszaállítsuk az eredeti állapotaik értékére (a mi esetünkben nil). Ezt elmulasztva, a többszöri futtatásra, a **pontlista** változónk tartalmazná az előző futtatások pontjait is, és a kirajzolt függvény az összes előzőleg kirajzolt függvényt is újra rajzolná.

A fent ismertetett eljárás egy a lehetséges megoldásokból, hasonló módon ábrázolhatunk más síkfüggvényeket is.

Ha 3D függvények ábrázolását akarjuk megtenni, hasonló módon kell eljárnunk, csupán a polyline helyett a 3dpoly AutoCAD parancsot alkalmazzuk.

2. Autodesk Inventor 2009

2.1. Bevezetés

Az Autodesk Inventor egy parametrikus 3D modellező szoftvercsomag, amely kiváló számítógépes segítséget nyújt elsősorban a gépész tervezői munkában.

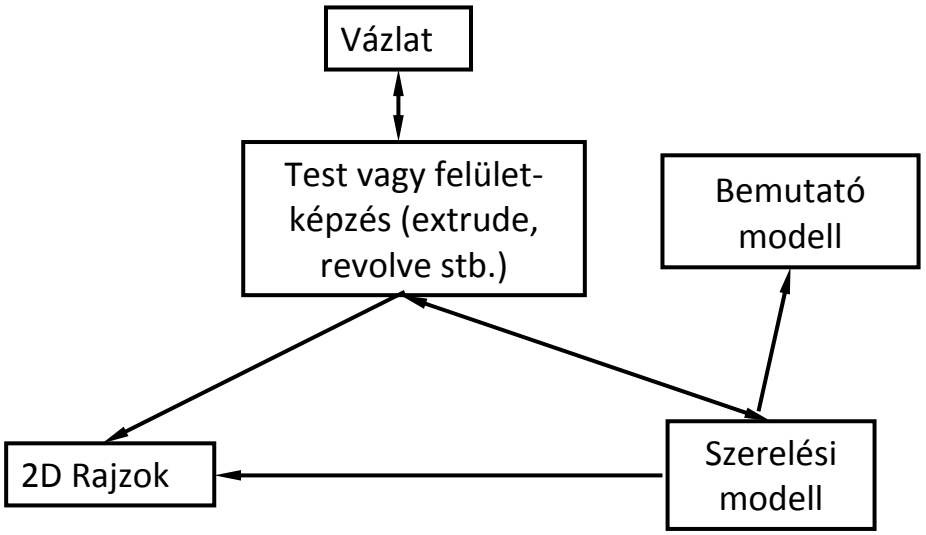
Az alkatrészek térbeli modelljének megrajzolása nagyon rövid idő alatt, szinte néhány egérkattintással megvalósítható. Ezek az alkatrészmodellek a továbbiakban alapját képezik síkbeli alkatrészrajzoknak, térbeli összeállításoknak, továbbá az alkatrészekből létrehozott térbeli modellekből készíthetünk szerelési sorrendet bemutató robbantott ábrákat és akár mozgás szimulációkat is. A térbeli alkatrész- és összeállítási-modellekből készített síkbeli rajzok létrehozásakor a nézeti és metszeti vetületek egyetlen egérkattintással létrehozhatók. Alkalmazhatunk részletes nézeteket, kitöréseket, megtört vetületeket is. Kiegészíthetjük a síkbeli rajzokat méretekkel, tűrésekkel, felületminőségi, valamint alak- és helyzettűrés jelekkel, rendkívüli egyszerű és gyors műveletekkel. A nagyszámú lehetőségeken felül a térbeli alkatrész- és összeállítás-modelleknél minden korábbi programnál látványosabb anyag és színmegjelenítést alkalmazhatunk, szinte fotórealisztikus modelleket hozva létre.

Az alkatrészek létrehozásakor készített vázlatok kényszerekkel és parametrikus méretekkel határozhatók meg. Ugyancsak parametrikus méretek szerint hozhatók létre a különböző sajátosságok is, így a későbbiekben bármikor tetszőleges mértékben változtathatjuk az alkatrészmodellek méreteit, amely azonnal érvényesül a síkbeli rajzokon és az összeállításokban is. A programnak ez a nagyfokú rugalmassága és nagyfokú termelékenysége tette kedveltté és egyik leghasználtabb szoftverré igen rövid idő alatt, az Inventort. Az Inventor-ban nagyfokú rugalmasság érhető el azáltal is, hogy a nem teljesen meghatározott vázlatokból létrehozott-, illetve a meghatározatlan méretű sajátosságokra előírhatunk adaptivitást. Ez azt jelenti, hogy az egymáshoz kapcsolódó alkatrészek az összeállításban veszik fel egymás méretét, vagyis az alkatrészek egyes méretei a csatlakozó alkatrészhez igazodnak. Az Autodesk Inventor program rövid idő alatt jelentősen fejlődött. Az egymást gyorsan követő újabb verziók, mindig jelentős változást hoztak a felhasználóbarát megoldásokban. A térbeli modellek gyors létrehozását biztosító eszközökön kívül, ma már nagyszámú szabványos gépelem áll rendelkezésünkre, de a hegesztett szerkezetek létrehozását elősegítő szabványos idomacélok elérhetősége is biztosított. A tervezői raktárnak nevezett elemtárakban számos ország szabványainak megfelelően választhatjuk ki az alkalmazni kívánt gépelemeket.

Fontos megemlíteni az Autodesk Inventornak azt a tulajdonságát, hogy az alkatrészek létrehozásakor, a felhasznált méretek automatikusan táblázatba íródnak, amely alapján a kész alkatrész bármilyen megváltoztatott mérettel fel-

használható különböző összeállításokban. A program maga is tartalmaz ilyen táblázatos méretekkel megadott elemeket, amelyeket iPart-nak neveztek el. Ezek közül talán a leghasznosabbak a hengeres és kúpos fogaskerekek, amelyek beillesztése után néhány jellemzőjük megváltoztatásával kész alkatrészek állnak rendelkezésünkre.

Nézzük, hogy mit is jelent az, hogy parametrikus modellező program.



2.1. ábra
Parametrikus modellezés elve a gépészeti tervezésben

A **2.1 ábra** az Inventorban lezajló modellező folyamatot mutatja be vázlatosan. A modellezés mindig egy vázlatból indul ki, amely geometriai és méreti kényszerekkel láthatunk el, ezek a kényszerek utólag mindig megváltoztathatók. A vázlatokat profilokból építjük fel. Egy profil összefüggő vagy különálló geometriai alakzatok csoportja. Egy profil lehet nyitott vagy zárt profil. Nyílt profillal készült vázlatokat jellemzően felületek kialakítására, zárt profilokat pedig testek képzésére alkalmazunk.

A különböző testképző műveletekkel 3D testeket vagy akár felületeket építhetünk a vázlatok alapján. Ezeket később összeszerelési modellekbe építhetjük egybe, szerelési kényszereket alkalmazva. A szerelt modellek animálhatók, mozgásszimulációra alkalmazhatók, bemutató modellek készíthetők belőlük, dinamikai és végeselem (Finit Element Analyzis) analízis végezhetők rajtuk. Az újabb és újabb verziójú Inventor-ba egyre nagyobb számmal beépített, szoftver-modulokat, elem-

zési lehetőségeket ki lehet bővíteni az Inventor API által nyújtott, szinte végtelen programozási lehetőségekkel.

Mind a testmodellek, mind az összeszerelési modellek alapján 2D rajzmodellek készíthetők. A kapcsolat a vázlat - testmodell - összeszerelési modell – 2D rajzmodell között olyan szoros, hogy balról jobbra haladva, bármely módosítás alkalmazása, megfelelő módosítást eredményez a felsőbb szintű modelleken is.

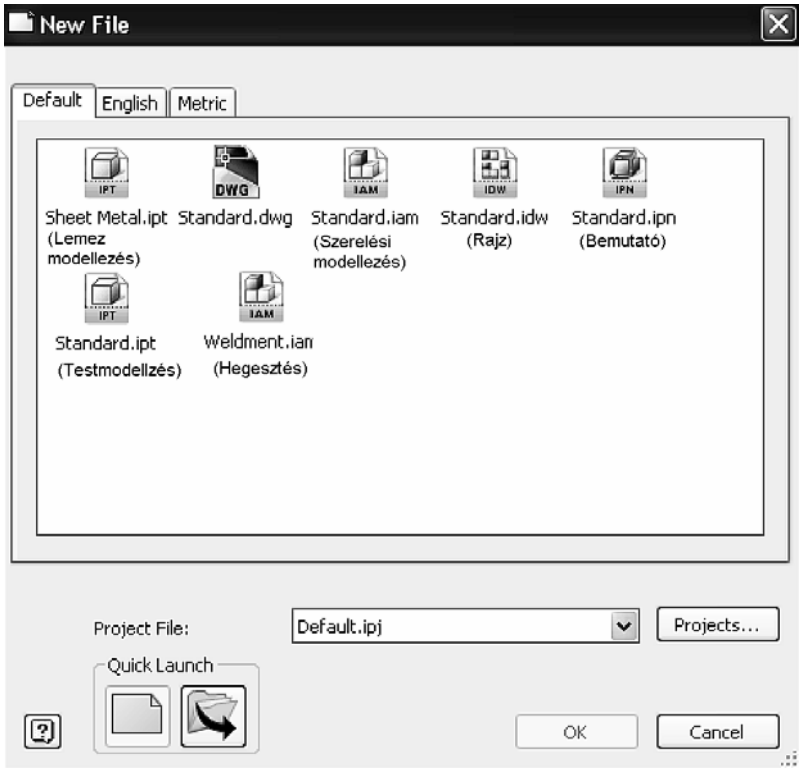
A **nem parametrikus 3D és rajzmodellekre jellemző**:

- Az objektum méretei meg vannak határozva a tervezésből, e nélkül nem hozhatók létre;
- Az objektum méretei nem változtathatók utólag, vagy csak igen körülményesen;
- Az objektumok nem „köthetők” össze mértani vagy méreti kényszerekkel.

A **parametrikus modellezésről** pedig:

- Az objektum méretei nincsenek meghatározva;
- Az objektum méretei **változtathatók** utólag;
- Az objektumok **„összköthetők” mértani vagy méreti kényszerekkel, ezek változtathatók.**

2.2. Meglévő rajz kinyitása. Új rajz kezdése



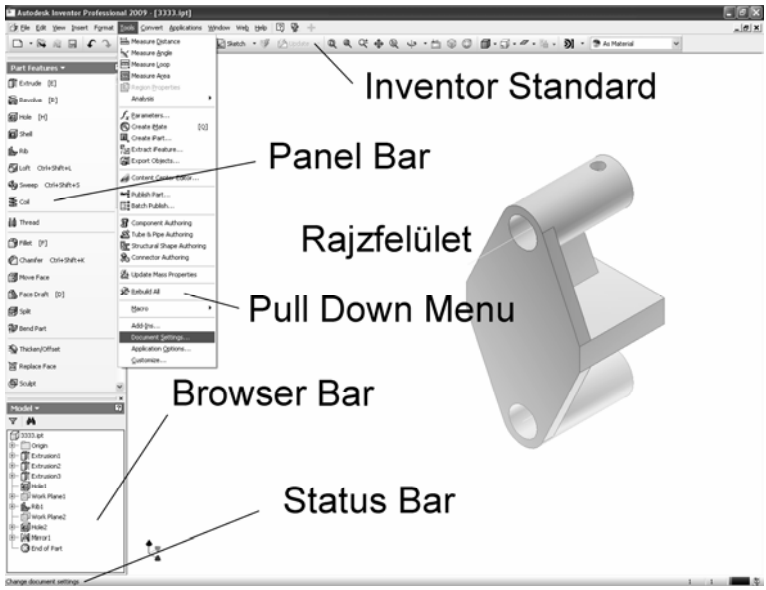
2.3. ábra
A kezdeti párbeszédablak

Az Autodesk Inventor 2009 indításakor megnyíló párbeszédablak a **2.3. ábrán** látható. A párbeszédablak jobb alsó felében a **Projects..**. ikon a **Projects** párbeszédablakot nyitja meg, ahol aktuális projektjeink állíthatók be.

Megjegyzés: Soha ne használjuk ugyanazt a testmodell (part) nevet két különböző test esetében, akkor sem, ha más-más almappákban is találhatók!

Mivel az egyes alkatrészek keresése a szerelési modell újraépítésénél, a part lapok neve után minden almappában megtörténik, az ilyen hiba, ha nem is a modellezés közben vagy annak az elején, de előbb-utóbb komoly gondokhoz vezet!

Az Open (Megnyitás) párbeszédablak bal oldalán az New (Új) ikonra kattintással jeleníthetjük meg azokat az ikonokat, amelyek lehetővé teszik a megfelelő típus kiválasztását. Indításkor az elkészítendő rajztól függően kell kiválasztani a megfelelő ikont.



2.4. ábra
A projekt ablak

A New lapon az alábbi típusokból választhatunk:

Standard.ipt

Új testmodell vagy új Alkatrész (Part) rajzolására használjuk az .ipt kiterjesztésű sablonokat.

Standard.iam

Új Összeállítás (Assembly) létrehozásához az .iam kiterjesztésű sablont kell választanunk.

Standard.idw

Új síkrajz létrehozásához az .idw kiterjesztésű sablont kell kiválasztanunk.

Standard.ipn

Új Összeállítás-bemutató (Presentation) készítésére az .ipn kiterjesztésű sablonok közül kell választanunk.

Sheet Metal.ipt

Új lemez alkatrész a Lemez (Sheet Metal).ipt sablon-állományok kiválasztásával hozható létre.

Weldment.iam

Új hegesztési szerelés.

A fentiekben ismertetett sablon-kezdőlapok mellett, a New ablak második és harmadik lapjáról is indíthatók colos és metrikus méretezésű (English, illetve

Metric) sablonok, több változatú szabvány alatt. Ugyanitt jelennek meg majd a felhasználó által létrehozott saját sablonok (template-ek) is, amelyeket a felhasználó saját beállításait tartalmazhatják.

2.3. A grafikus képernyő

Az **Open** (Megnyitás) párbeszédablak **Metric** (Metrikus) lapján válasszuk ki a **Standard(mm).ipt** sablon-állomány ikonját, amely alkatrész módban indítja el a programot. A megjelenő grafikus képernyő felosztása a többi sablon-állomány választásakor is azonos, csak a tartalom változik, hiszen más parancsok kellenek egy összeszerelési modell vagy egy bemutató létrehozásához. A grafikus képernyő fő részei a **2.4. ábrán** figyelhetők meg és ezek a következők:

- **Pull Down Menus** – Legördülő menük
- **Panel Bar** – Paneltár
- **Browser Bar** – Áttekintőtár
- **Status Bar** – Állapotsor
- **Rajzfelület**

Pull Down Menu – A Legördülő Menü

Ez a sor csoportosítva tartalmazza az általánosan használható parancsokat. Valamelyik címre kattintva, legördítve jelenik meg a választék. Néhány esetben jobbra mutató nyíl jelzi, hogy további almenü is megjeleníthető, illetve egyes parancsoknál három pont jelzi, hogy a további beállítások párbeszédablakban végezhetők el. Attól függően, hogy Part, Assembly, Presentation, Sheet metal, Welment vagy síkbeli rajz létrehozását választottuk, az egyes menükben megjelenő sorok változhatnak.

Ha valamelyik sor szürke színnel jelenik meg a menüben, akkor az a parancs nem használható az aktuális művelet végrehajtása közben. Amely parancsok billentyűzetről is kiadhatók, azoknál ez szerepel a megfelelő menüsor végén.

Inventor Standard – Központi eszköztár

Részben az általánosan használt Windows ikonokat, részben az általánosan használt Inventor ikonokat tartalmazza. Az utóbbiak többnyire a megjelenítési módok szabályozására szolgálnak. Az eszköztár jelentős része minden sablontípusnál azonos, vannak azonban változó részei is.

Panel Bar – Panel tár

A Panel Bar (Paneltár) tartalma attól függ, hogy milyen módban kezdünk el egy rajzot. Ha új Part (Alkatrész) létrehozásához .ipt kiterjesztésű sablont választunk, akkor a vázlat létrehozásához szükséges parancsok jelennek meg. A vázlat be-

fejezése után azonban már a Part Features (Alkatrész sajátosságok) jelenik meg.

Az ikonok mellett érdemes a tájékoztató szöveget is bekapcsolni. Csak később, nagyobb tapasztalat után, esetleg nem lesz szükség a feliratokra. Ekkor a jobb egérgombbal megjeleníthető helyi menüben kiválaszthatjuk az Expert sort, ezután a parancsok csak ikonok formájában jelennek meg.

A Paneltár (Panel Bar) nem csak önálló ikonokat tartalmazhat, hanem előfordul, hogy egy ikon továbbiak fedő ikonjaként működik, ezek mellett egy lefelé mutató nyílra kattintva tudjuk használni a további ikonokat is. Némely parancsnál csak két-három további ikon jeleníthető meg, de van olyan, amelynél jóval nagyobb a választék.

Szükség esetén a Panel Bar-ban megjelenő ikonok külön eszköztárban is megjeleníthetők, a View (Nézet) menü Toolbar (Eszköztár) sorához tartozó almenüből.

Browser Bar – Áttekintőtár

A Browser Bar tartalma a rajz típusától függően változik. Általánosságban úgy határozható meg, hogy információkat jelenít meg az aktuális rajz létrehozásának sorrendjéről, és ami nagyon fontos, az elemek kapcsolódásáról, tulajdonságairól. Egyúttal lehetővé teszi az egyes vázlatok, sajátosságok, alkatrészek nézetek stb. szerkesztését is, ugyanis a már létrehozott objektumok innen is elérhetőek. Az itt megjelenített információk szabályozhatók a felső részen lévő, szűrő ikonnal megjelenített legördülő menüben.

Browser Bar első eleme az Origin mappa, amely a + jelre kattintással kifejthető. Ekkor válnak láthatóvá azok az ikonok, amelyek az alapértelmezett munkasíkok (OYZ, OXZ, OXY), alapértelmezett munkatengelyek (X, Y, Z) és a középpont kijelölését teszik lehetővé. Ha a kurzort egy ikon fölé állítjuk, akkor ez a sík vagy tengely megjelenik a rajzszerkesztő ablakban. Az ikonra történő kattintáskor a hozzá tartozó elem aktívvá válik az ablakban. Alkatrészek készítésekor a sajátosságok a létrehozásuk sorrendjében jelennek meg az áttekintőtárban. Összeállításoknál az alkatrészek beillesztésének sorrendjében jelennek meg az elemnevek, de a szűrővel beállíthatjuk, hogy az egyes alkatrészek létrehozásának lépései is elérhetők legyenek.

Status Bar – Állapotsor

A képernyő alján lévő keskeny sor csak akkor jelenít meg információt, amikor egy parancs aktív. A bal oldali részen minden típusú rajznál üzenet jelenik meg, amely az aktuális művelet végrehajtását segíti. Új alkatrész létrehozásánál, a vázlat rajzelemeinek elhelyezésekor, ennek a sornak a jobb oldali részén koordináta értékek, illetve hosszúság, szög, sugár stb. jelenik meg a rajzelemtől függően.

2.4. Billentyűparancsok

Az Inventor parancsok kiadása általában az ikonokról vagy a legördülő, illetve a helyi menükből történik. Számos parancs azonban billentyűzetről is indítható egy karakter beírásával vagy két billentyű együttes megnyomásával. Ezeket a parancsokat az alábbi táblázatban ismertetjük.

Billentyű	Inventor parancs
F1	Az aktív parancs Help menüjét (súgóját) jeleníthetjük meg
F2+bal egérgomb	Ha a bal egérgombot lenyomva tartjuk az egérmozgatás közben, akkor eltolhatjuk a grafikus ablakot (Pan)
F3+bal egérgomb	Ha a bal egérgombot lenyomva tartjuk a felfelé vagy lefelé történő egérmozgatás közben, akkor a grafikus ablak tartalma nagyítható, illetve kicsinyíthető (Zoom)
F4+bal egérgomb	Ha a bal egérgombot lenyomva tartjuk az egérmozgatás közben, akkor elforgathatjuk az objektumokat (Rotate parancs)
F5	Az előző nézethez térhetünk vissza (Previous View)
F6	Az izometrikus nézethez térhetünk vissza (Isometric View)
B	Síkbeli rajzoknál egy tételszámot adunk a rajzhoz
C	Asembly módban megjeleníti a Place constraint párbeszédablakot
D	Síkbeli rajzhoz vagy az aktuális vázlathoz új méretet adhatunk hozzá (General Dimension)
DO	Koordinátaméretet méretet ad a rajzhoz
E	Megjeleníti a Kihúzás (Extrude) párbeszédablakot, a kiválasztott profil kihúzásához
F	Alaktűrés-keretet helyezhetünk el a síkbeli rajzon (Format)
FC	Geometriai vagy helyzettűrési ablakot ad a rajzhoz
H	Hole (Furat) sajátosságot hozhatunk létre
L	Egyenest vagy ívet rajzolhatunk (Line)
P	Az aktív összeállításban egy új részegységet helyezhetünk el
R	Megjeleníti a Megforgatás (Revolve) párbeszédablakot, amely segítségével megforgatott sajátosságot hozhatunk létre
S	Vázlatot hozhatunk létre egy felületen vagy síkon
Esc	Kiléphetünk az aktív parancsból
Delete	Törölhetjük a kiválasztott objektumokat
Backspace	Vázlatként rajzolt vonalszakaszok közül törölhetjük az utolsóként megrajzolt szakaszt
Alt + egér elmozdítás	Mate kényszert adhatunk az összeállításhoz
Ctrl + Shift	A kiválasztott elemekhez hozzáadhatunk vagy törölhetünk
Shift + jobb egérgomb	A Select (Kiválasztás) eszközmenüjét indíthatjuk el
Ctrl + Enter	Az előző szerkesztési állapotba tér vissza

Billentyű	Inventor parancs
Shift + Rotate Tool	Automata módon forgatja a modellt a grafikus ablakban. Kattintásra kilép.
Ctrl + Y	Visszalépés után az Előre (Redo) parancs érvényesíthető
Ctrl + Z	Visszaléphetünk az előző művelethez
Space	Ha elindítottuk a 3D forgatást, akkor átkapcsolhatunk Free Rotate és a Common View eszközök között

Ezeken a billentyűparancsokon kívül használhatók az általános Windows billentyűkombinációk is:

Billentyű	Windows parancs
Ctrl + C	Copy
Ctrl + N	New Document
Ctrl + O	Open
Ctrl + P	Print
Ctrl + S	Save
Ctrl + V	Paste
Ctrl + Y	Redo
Ctrl + Z	Undo

2.5. Alapbeállítások

2.5.1. Document Settings... (Dokumentum beállítások)

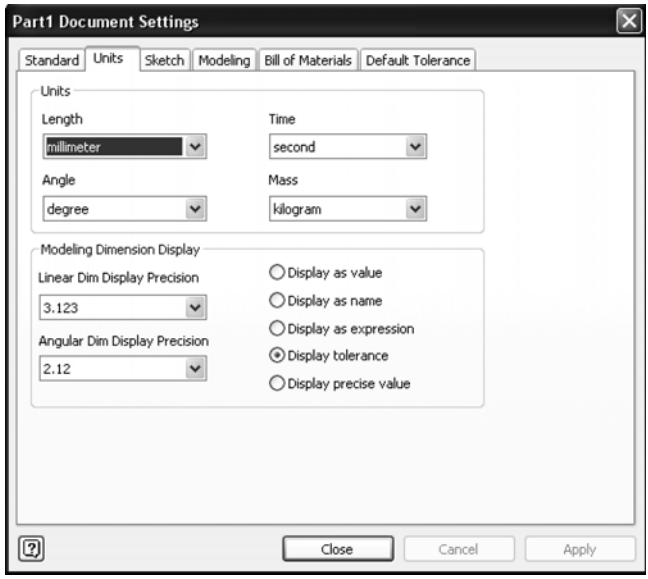
Az alábbi műveleteket nem kell minden rajzkezdéskor ellenőrizni, de az első rajznál célszerű megismerni az alkalmazás lehetőségeit. A Tools menüben kattintsunk a Document Settings.. sorra, amely párbeszédablakot jelenít meg (**2.5. ábra**). Ebben az első lapon a mértékegységek megadását, a modellméretek pontosságának megjelenését, a másodikon a 2D vázlatok raszter és háló beállítását, a harmadikon a 3D modellezéssel kapcsolatos térbeli rasztereket, a negyediken pedig az alkalmazható tűréseket állíthatjuk be.

A **Modelling Dimension display** a felhasznált és kiírt mértékegységek pontosságát, illetve azok kiírási formáját határozza meg. A hosszúság és a szög mértékegységén kívül az idő és a tömeg mértékegységeit is be lehet állítani, a későbbi számolási műveletek miatt. Ha a milliméter, fok, szekundum, kilogramm beállítás jelenik meg, akkor a metrikus mértékegységeknek ez megfelelő. A méretek megjelenítési formájának lehetséges változatai:

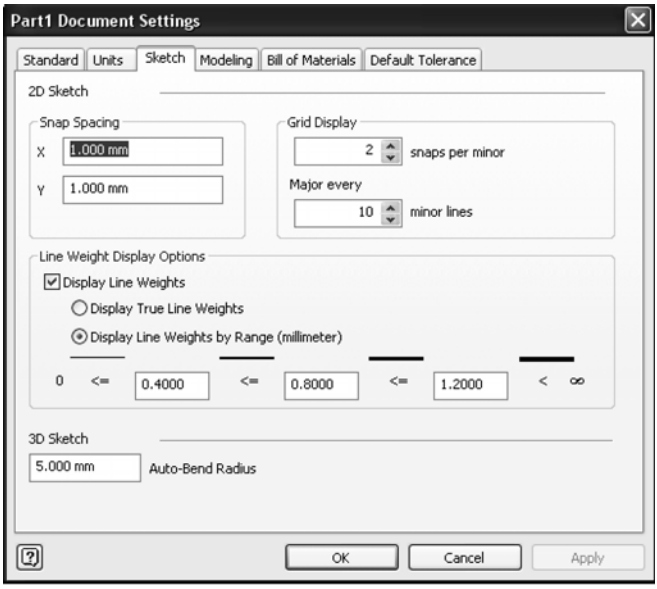
Display as value – A vázlaton elhelyezett parametrikus méretek számértékkel jelennek meg.

Display as name – A vázlaton elhelyezett parametrikus méreteknél a program által hozzárendelt azonosítók jelennek meg.

Display as expresion – A vázlaton elhelyezett parametrikus méretek az azonosítókból képzelt matematikai kifejezésként (képletként) jelennek meg.



2.5. ábra
A Part1 Document Settings párbeszédablaka



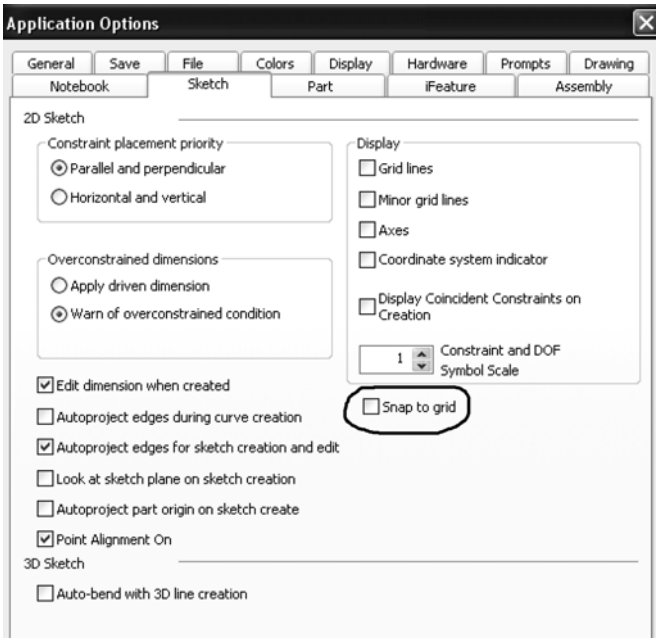
2.6. ábra
A Sketch lapon található beállítások

A **Sketch lapon** 2D vázlat (2D Sketch) részben a **Snap Spacing** beállításait ellenőrizhetjük (**2.6. ábra**), valamint a vázlatkészítéskor megjelenő háló rácstávolságát szabályozhatjuk, illetve az új változatban a vonalak vastagságának a megjelenítését állíthatjuk be.

Raszter – szálkereszt elmozdulásának lépésközét meghatározó szám. Ha például a raszter értéke 1, és a rácsra illesztés (Snap to grid) be van kapcsolva, akkor a szálkereszt (az egér mozgatásakor) 1 egységnyit mozdul el.

A **Snap Spacing** értékét a párbeszédablakban célszerű 1-re állítani, a **Grid Display-ét** pedig 2-re. Tapasztalatok alapján megfelelő a 2 raszterpontonként történő megjelenítés, illetve a 10 osztásonként megadott vastagvonal beállítása.

A **Grid** beállítása csak akkor érvényesül, ha a **Tools** menüben, az **Applications Options...** menüsorra kattintáskor megjelenő Options párbeszédablak Sketch lapján, be van kapcsolva a **Snap to Grid** sorkapcsolója (**2.7. ábra**). Ebben az esetben akkor is a raszterpontok kerülnek kijelölésre, ha a szálkereszttel nem pontosan állunk rá valamelyik osztásra.

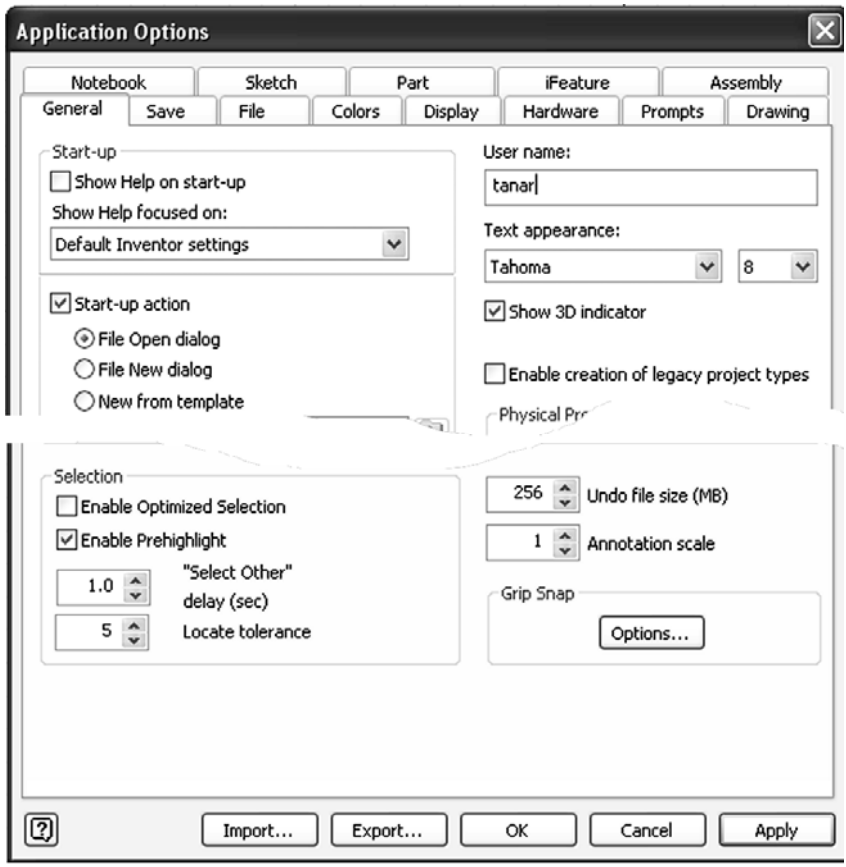


2.7. ábra.
A Snap to grid be- és kikapcsolása

A raszter ki- vagy bekapcsolására rajzolás közben is van lehetőség. A vázlat készítése közben – egy rajzoló parancsot befejezve – a jobb egérgombbal kattintva a rajztérre, a megjelenő helyi menüben ki- vagy bekapcsolhatjuk a rasztert. A raszter használatát tulajdonképpen az dönti el, hogy viszonylag pontos, vagy kevésbé pon-

tos vázlatot akarunk-e készíteni. Mivel a méreteket általában utólag pontosítani kell, az indokolja a raszter bekapcsolását, hogy a rajzelemek csatlakozása pontosabb lesz.

2.5.2. Application Options (Alkalmazás beállítások)



2.8. ábra.
A Genaral lap beállításai

További beállítások végezhetők el, illetve ellenőrizhetők le a Tools menü **Applications Options...** menüsorára kattintva, a megjelenő Application Options nevű párbeszédablakban. Ebben tizenhárom lapon szabályozhatjuk a program beállításait. Ezek közül most csak egy pár lapon található lehetőséget tekintünk át. Az első a General lapon található beállítási lehetőségek (**2.8. ábra**).

Undo file size (MB)	Rajz készítésekor a rajzolás lépéseit megőrzi a program egy ideiglenes állományban. Ennek segítségével a lépések visszavonhatók. Nagyméretű, bonyolult modellek készítésekor célszerű megnövelni ennek a méretet.
Start-Up	Beállításokat tartalmaz az alkalmazás elindításával kapcsolatosan: indul-e a Help (Segítő); az alkalmazás indításakor File Open (meglévő állomány nyitása), File New (új állomány inditása) vagy New from Template (Új állomány sablon alapján) opciók választása?
Selection	A kijelölések során alkalmazott opciók beállításai.
Physical properties	A modellek mechanikai tulajdonságait számoló, megjelenítő modul beállításai.
Show 3D indicator	Az origóban megjelenő, tengelyirányokat jelző, színes mutatók megjelenését szabályozhatjuk.
User	A projekten dolgozó felhasználó neve.

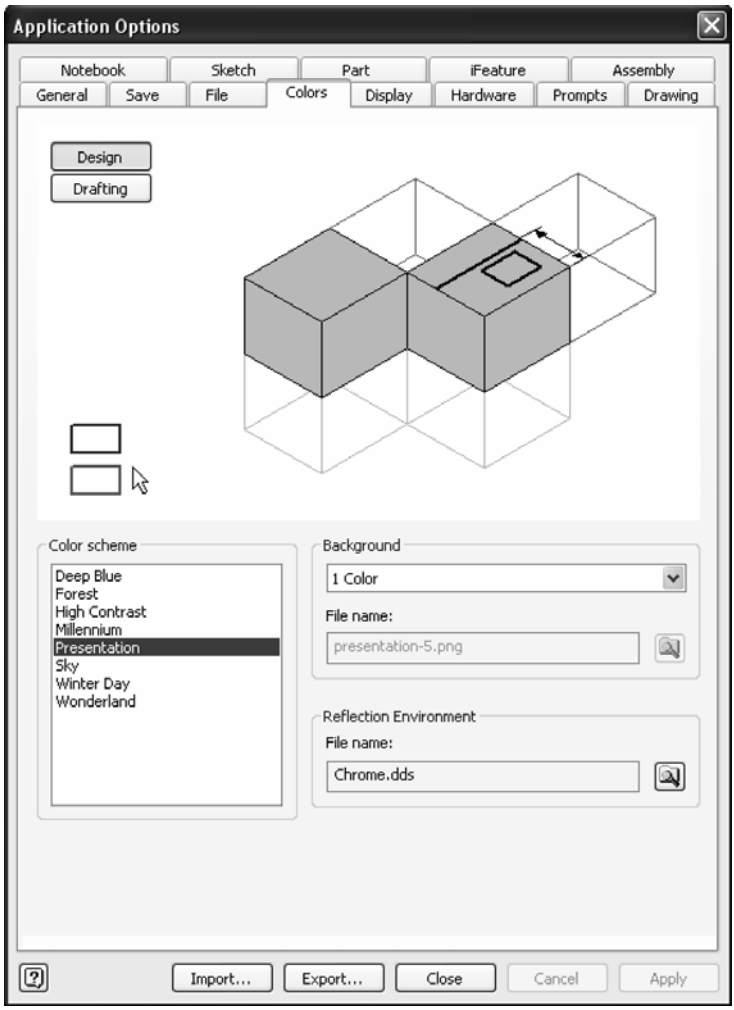
A Colors lap

A rajzfelület színbeállításait végezhetjük el ezen a lapon (**2.9. ábra**).

Design/Drafting	A Design kapcsolóval a vázlat- és modellkészítéskor megjelenő háttér és alkatrész színeket állíthatjuk, a Drafting pedig a 2D rajzok létrehozásának színösszeállítás választékát kínálja fel. A rajzlap színe a Format menü Standards... menüsorával megjeleníthető párbeszédablakban állítható be, 2D rajzmódban.
Color Scheme	A beépített színminták listájából választhatjuk ki a megfelelő összeállítást.
Background	A háttér gradiens típusú megjelenítését kapcsolja be illetve ki.
Show Reflections and Textures	Tükröződő szín- és megvilágítási stílusok beállítását teszi lehetővé. Ha nincs bekapcsolva, akkor a modellrajzokon nem jelenik meg például a csavarmenet sajátosság.

Sketch lap

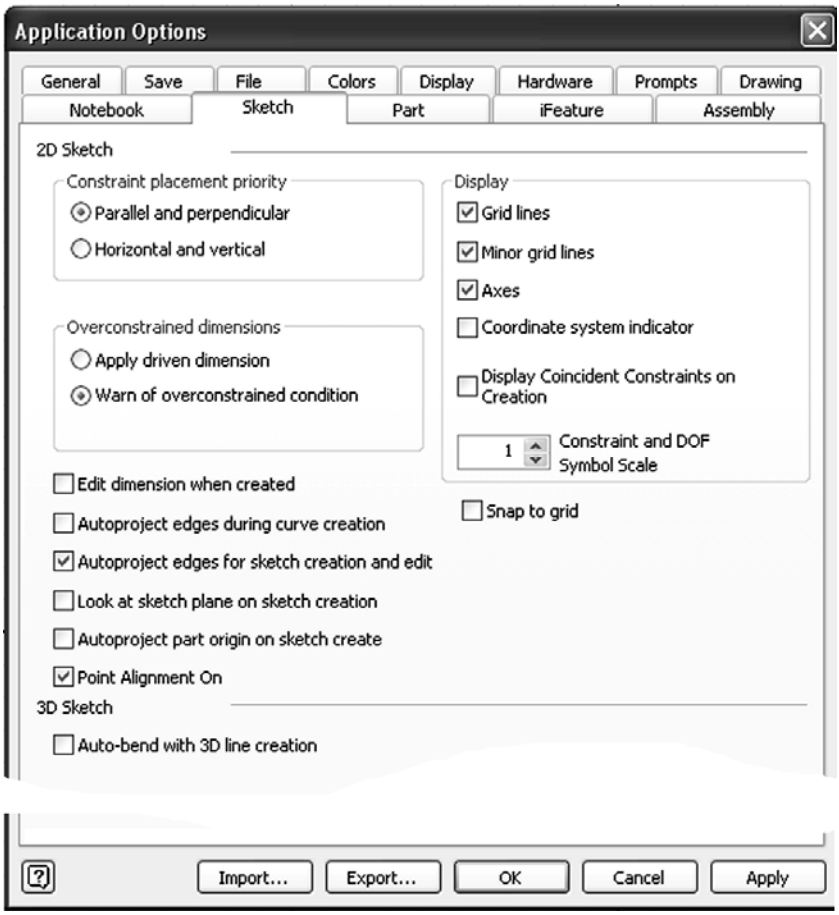
A 2D vázlatok készítésével kapcsolatos beállítások végezhetők el a lap felső részén (**2.10. ábra**).
Constraint Placement Priority – Kiválaszthatjuk, hogy az automatikus kényszer elhelyezésnél a Parallel and Perpendicular, vagy a Horizontal and Vertical kényszertípust akarjuk-e használni.



2.9. ábra
A Colors lap tartalma

Overconstrained Dimensions – Egy zárójelbe foglalt méretet alkalmaz túlhatározottság esetében. A méret a vázlattal együtt megváltozik, de nem változtathatja meg a vázlatot. A **Warn of Overconstrained Condition** választásakor üzenet jelenik meg.
Edit dimension when created – Bekapcsolt állapotban, új méret létrehozásakor megjelenik az Edit Dimension párbeszédablak, amelyben lehetőséget kapunk a méret pontosítására, a „találomra” megrajzolt méret javítására. Ez a beállítás

megtehető akkor is, ha egy méretre a jobb egérgombbal klikkelünk, és ott jelöljük ki a fenti opciót.

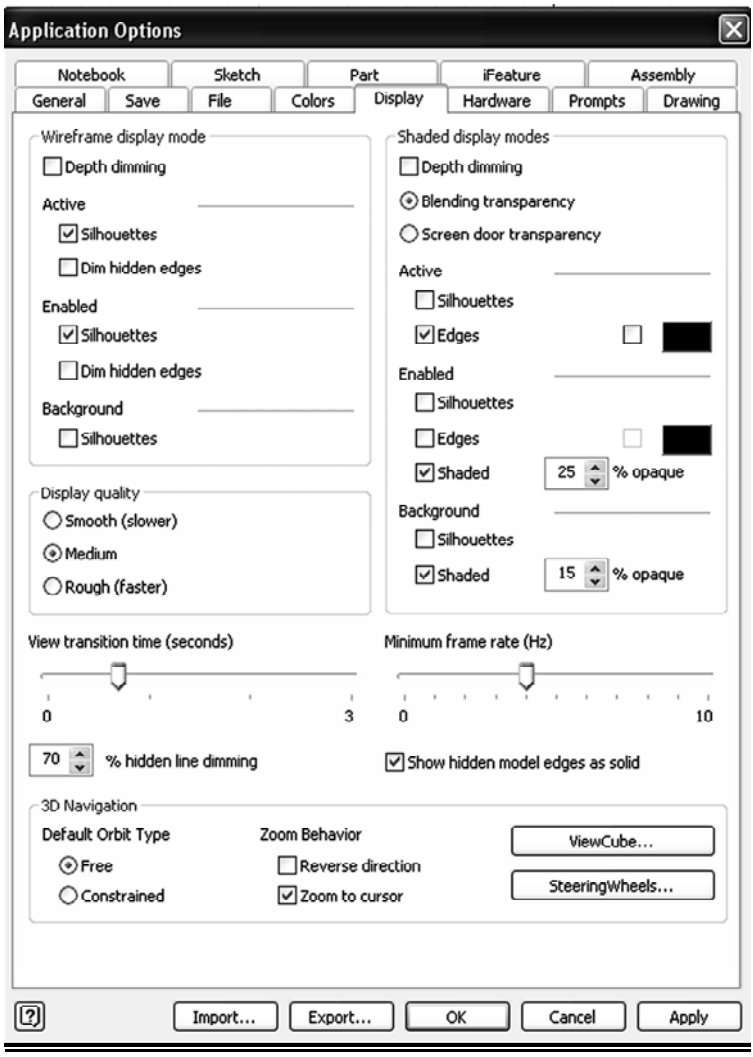


2.10. ábra
Vázlat beállítások

Automatic reference edges for new sketch – Bekapcsolva új vázlat létrehozásakor az Inventor automatikusan referencia felületként vetíti a kiválasztott felület éleit a vázlatba.
Display – **Grid Lines** és **Minor Grid Lines** (rács megjelenítése); **Axes** – Tengelyek megjelenítése; **Coordinate System Indicator** – Koordináta rendszer megjelenítése; **Display Coincident Constraints on Creation** – Egybeesési kényszerek megjelenítése létrehozásukkor.

Display lap

Ezen a lapon szabályozhatjuk az alkatrészek és összeállítások megjelenési módját. Két elkülönített rész szolgál a drótvázas, illetve az árnyalt megjelenítési mód beállításaira. Az itt elvégzett módosításokkal jelentősen befolyásolhatjuk a rajzok megjelenésének a minőségét (**2.11. ábra**).



2.11. ábra
A Display lap

Wireframe Display Mode – Drótvázas megjelenítési mód.

A **Depth Dimming** kapcsoló használatával a modell távolabbi részei elhalványíthatók. Az Active részben egy alkatrész vagy egy részegység megjelenítési módját szabályozhatjuk, ami majd az összeállításban érvényesül.

Be- vagy kikapcsolhatjuk a **Silhouettes** kapcsolót és meghatározhatjuk, hogy a **Hidden Edges** (Takart élek) elhalványítása érvényesüljön-e azoknál az éleknél, amelyeket más alakzatok eltakarnak. Az Enabled címszó azokra az alkatrészekre és részegységekre érvényesíti a beállításokat, amelyek a szerelésben engedélyezve vannak. Egy összeállításban engedélyezettnek tekinthetők mindazon alkatrészek, amelyek a Place Component paranccsal lettek beillesztve, és amelyeket nem tiltottunk le.

A **Background** részben azoknak az alkatrészeknek vagy részegységeknek a megjelenése szabályozható, amelyek az összeállításban letiltásra kerültek.

Display Quality – A megjelenítés minősége

A modell megjelenítésének felbontását adja meg. Általában a finomabb felbontás hosszabb várakozási időt eredményez a változtatások utáni ismételt megjelenítésnél. Amikor nagyon nagy és bonyolult modellekkel dolgozunk, valószínű, hogy le kell csökkentenünk a megjelenítés minőségét, hogy felgyorsítsuk a műveleteket. A **Rough** beállítás például ideiglenesen csökkenti a nagy alkatrészek részletességét, de gyorsabb frissítést eredményez, míg a Smooth beállítás kevesebb részletet vesz el ideiglenesen, de lassabb frissítést eredményez.

Shaded Display Modes – Árnyalt megjelenítési módok

Az alkatrész és összeállítási modellek árnyalt megjelenítési módjának beállításait végezhetjük el ebben a részben. Az egyes elemtípusokra vonatkozóan, a kiemelt élek színét is beállíthatjuk a szín téglalapra kattintással megjeleníthető párbeszédablakban. A felső részen a **Blending Transparency** is hasonló a drótváz módnál leírtakkal, de itt két másik választási lehetőség is szerepel. A Blending Transparency az egymás takarásában lévő részek színének átlagszámításával hoz létre összemosásos megjelenítést. A **Screen Door Transparency** használata akkor célszerű, ha a grafikus kártya rosszabb teljesítményű. Ennek a lehetőségnek a bekapcsolása felgyorsítja a műveleteket, de gyengébb minőségű megjelenítést eredményez.

View Transition Time (seconds) – Nézetváltási idő

Amikor a **Previous View**, az **Isometric View, Zoom-All, Look At** stb. parancsot használjuk, akkor a nézőpont a csúszósávon beállított idő alatt veszi fel az új helyzetet. A beállítható idő 1–3 másodperc között lehet. A nulla másodperc hosszúságú átállási idő az átmenetet hirtelenné teszi.

Minimum Frame Rate (Hz) – Minimális megjelenítési frekvencia

Nagyméretű rajzoknál a **Rotate**, a **Pan** és a **Zoom** alkalmazásakor előfordulhat, hogy az Inventor a beállított képfrissítési frekvenciát próbálja követni, de ehhez a nézet egyes részeinek leegyszerűsítésére vagy elhagyására lehet szükség. Minél kisebb értéket állítunk be, annál kevésbé egyszerűsíti le a program a megjelenítést, de az időtartam is hosszabb lesz.

% Hidden Line Dimming – Rejtett vonalak megjelenítési módja
A lap alján lévő százalékos beállítás lehetővé teszi, hogy beállító mezőben megadjuk, hogy az eltakart élek hány százalékkal halványabban jelenjenek meg, mint a látható élek.

Toolbars lap

A **Customize** párbeszédablaknak ez a lapja az eszköztárak módosításait teszi lehetővé, a jobb felső részen lévő nyomógombok alkalmazásával. Ezek a lehetőségek a következők:

- új eszköztárak hozzáadása;
- létező eszköztárak átnevezése;
- létező eszköztárak másolása;
- felhasználói eszköztárak törlése;
- átalakított eszköztárak visszaállítása;
- a kijelölt eszköztár megjelenítése.

2.6. Kényszerek

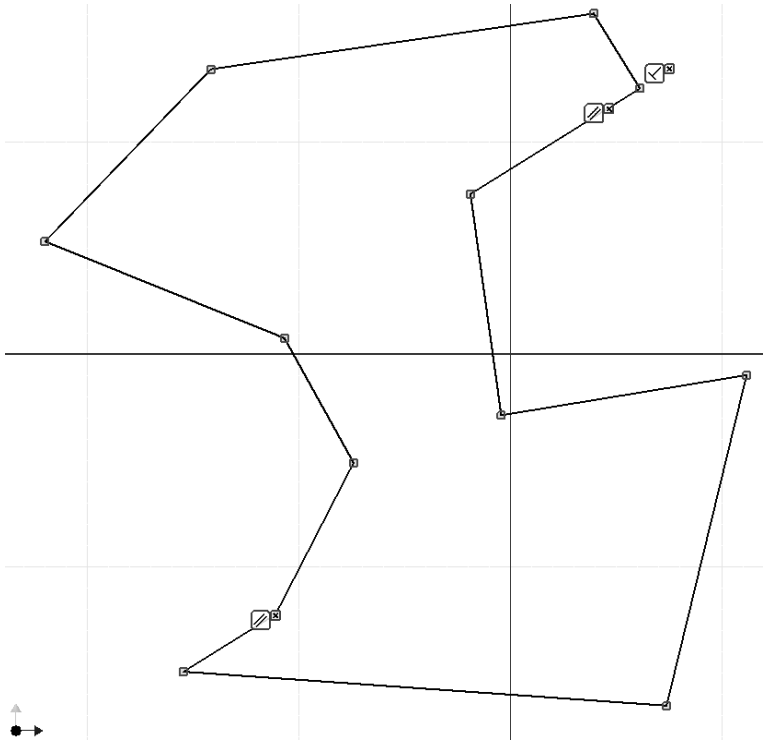
A vázlatokban alkalmazott kényszerek, a vázlat elemei – vonalak, ívek, spline-ok stb. – közötti kapcsolatokat, az elemek egymáshoz viszonyított helyét határozzák meg.

Kényszernek nevezzük az alkatrész-modellezésnél a vázlat alakjának és méretének változását szabályozó kapcsolatot. A vázlatban alkalmazott kényszerek vonalak és ívek közötti kapcsolatokat határoznak meg.

Amikor vázlatot készítünk, az Application Options párbeszédablak Sketch lapján a **Constraint Placement Priority** sorrendje részben beállított kényszerek automatikusan érvényre jutnak. Ha az alapértelmezett **Parallel and Perpendicular** (Párhuzamos és Merőleges) kényszert használjuk, akkor a vázlatként rajzolt vonalaknál azok is párhuzamosak, illetve merőlegesek lesznek, amelyek csak megközelítően párhuzamosak vagy merőlegesek. Az automatikusan érvényesülő kényszerek alkalmazása csökkenti a teljes meghatározottsághoz szükséges méretek számát, de további kényszereket is hozzárendelhetünk a vázlat rajzelemeihez. Természetesen a kényszerek eltávolítására is van lehetőség.

Nézzük a **2.12. ábrán** látható vázlatot. Ha az Options párbeszédablak Sketch lapján kikapcsoljuk a Snap to Grid kapcsolót, akkor az alábbi szándékosan pontatlan (sem vízszintes, sem függőleges, sem merőleges, sem párhuzamos vonalak nincsenek benne) vázlatot meg tudjuk rajzolni. A vázlat menüben kiválasztva a **Show Constraints** sort és rákattintva a rajz minden egyes elemére, megfigyelhető, hogy még ebben az esetben is nagyszámú kényszer látható már a vázlaton (Coincident, Paralel és Perpendicular).

Az Autodesk Inventor Professional 2009-es változatában már nem eszköztárszerűen jelennek meg a kényszerek, hanem a **2.12. ábrán** látható grafikus ikonok formájában. A már meglevő egybeesési kényszerek mellett nem találhatók az ikon elrejtésére használható **x** jelek, a merőleges és párhuzamos ikonok mellett ellenben igen. Ezekre rákattintva a kényszer rejtetté válik. Ha a kényszerre jobb egérgombbal rákattintunk, akkor a megjelenő menüben is megtalálható az elrejtést szolgáló **Hide** utasítás. A kényszer kétféleképpen törölhető: ugyaninnen a Delete sorral, vagy kijelölve a kényszert és megnyomva a **Delete** billentyűt. Az egybeesési kényszer esetén, ha ráhelyezzük az ikonra az egér mutatóját, akkor felette két négyszög jelenik meg. Ezekre kattintva, jobb egérgombbal, a Coincident kényszer elrejthető és törölhető is. Az ábrán látható, hogy a párhuzamos és merőleges kényszereken kívül, minden vonalhoz tartozik két úgynevezett **Coincident** (Ráeső) kényszer. Ezek a vonalszakasz csatlakozását jelzik az előtte és utána lévő szakaszhoz.



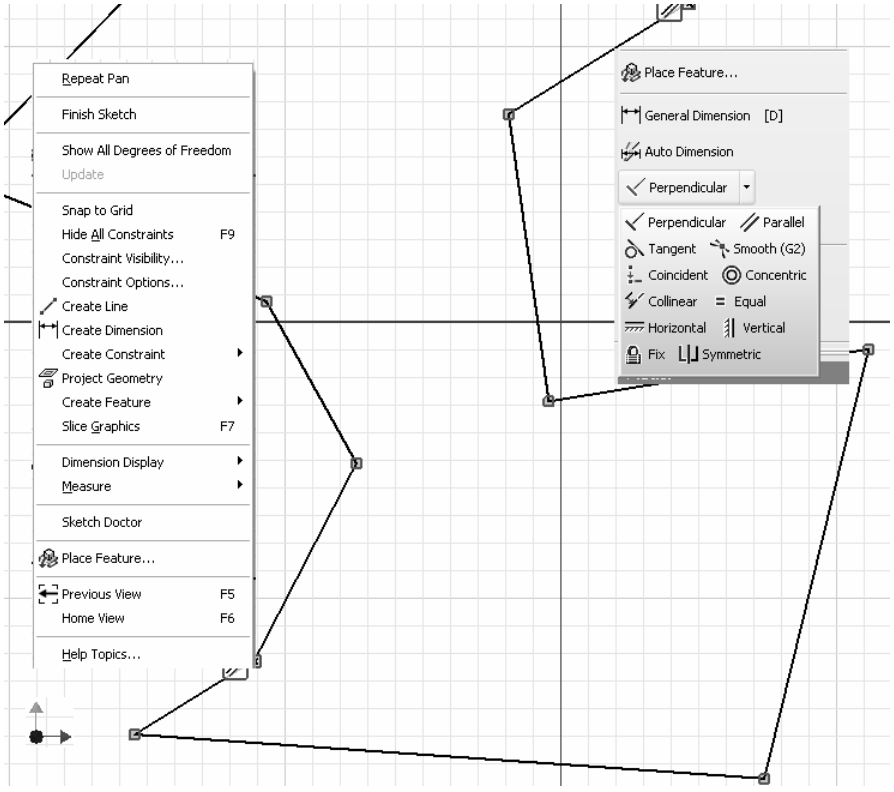
2.12. ábra
Szándékosan szabálytalanra húzott alakzat is tartalmaz kényszereket

A kényszerek ikonjai csoportosan is megjeleníthetők, illetve elrejthetők: a vázlatra jobb egérklikkel kattintva megjeleníthető a **2.13. ábra** bal oldalán látható me-

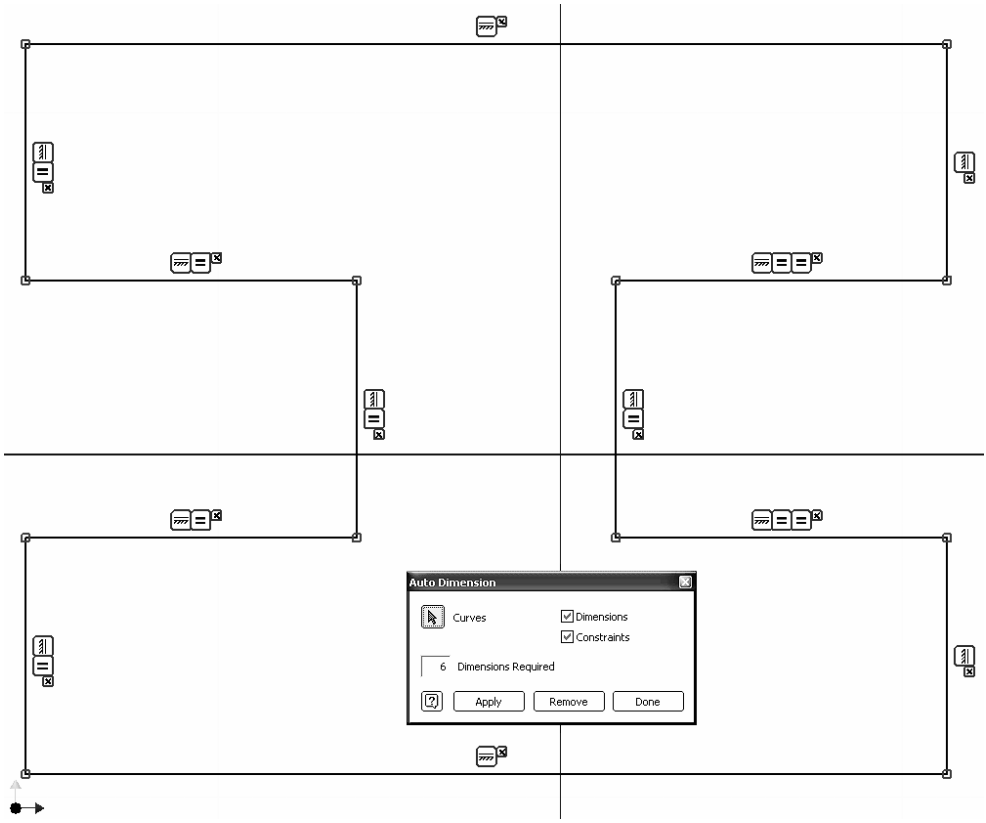
nü, amelyben a **Show All Constraints** és a **Hide All Constraints** parancsok is találhatók. Ezekre kattintva a vázlat minden elemének kényszerei egyszerre is megjeleníthetők vagy elrejthetők.

A továbbiakban kipróbáljuk, hogy milyen hatással vannak a kényszerek a vázlatra. Letöröljük az összes párhuzamos és merőleges kényszert. Ehhez a kényszer ikonjára kell kattintani jobb egérgombbal, majd a Delete billentyűvel végrehajtjuk a törlést. Ha ezeknek a kényszereknek az eltávolítása után kiadjuk az Automatikus méretezés – **Auto Dimension** – parancsot, akkor a párbeszédablakban üzenetet kapunk, hogy 22 méret szükséges a teljes meghatározottsághoz.

Próbáljuk ki, hogy mennyit változik a helyzet, ha a Horizontal, Parallel, Vertical és az Equal kényszereket hozzárendeljük a vonalakhoz. A művelet elvégzéséhez, a jobb egérgombbal megjelenített helyi menüben, a Create Constraint sort kell kiválasztani, majd a kényszer típusától függően egy vagy két rajzelemet jelöljünk ki. A kényszerek létrehozását a 2D Sketch Pannel ikonjaival is elvégezhetjük (**2.13. ábra** jobb felső része). A **2.14. ábrán** immár a kényszerekkel ellátott vázlat látható.



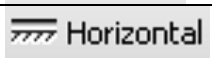
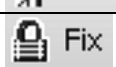
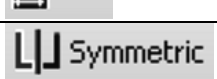
2.13. ábra
A síkvázlat kényszereinek létrehozási lehetőségei



2.14. ábra
A vázlat kényszerek hozzáadás után

Az Inventorban előírható kényszerek:

Tangent	Tangent (érintő)	Érintőleges kényszer
Parallel	Parallel (párhuzamos)	Párhuzamosság kényszere
Perpendicular	Perpendicular (merőleges)	Merőlegesség kényszere
Smooth (G2)	Smooth (simuló)	Spline görbe, egyéb vázlatelemekhez való simulását biztosító kényszer
Coincident	Coincident (egybeeső)	A kijelölt pont rajta lesz a megfelelő rajzelemen
Concentric	Concentric (koncentrikus)	A körök és körívek a középpontjai egybeesnek
= Equal	Equal (egyenlő)	A rajzelemek hossza azonos

	Collinear (Egyvonalú)	Kolineáris egyenesek
	Horizontal (vízszintes)	Vízszintességi kényszer
	Vertical (függőleges)	Függőlegességi kényszer
	Fix (rögzített)	Rögzített elemeket eredményez
	Symmetric (szimmetrikus)	Megadott tengelyre tükrözi a kijelölt egyes elemeket.

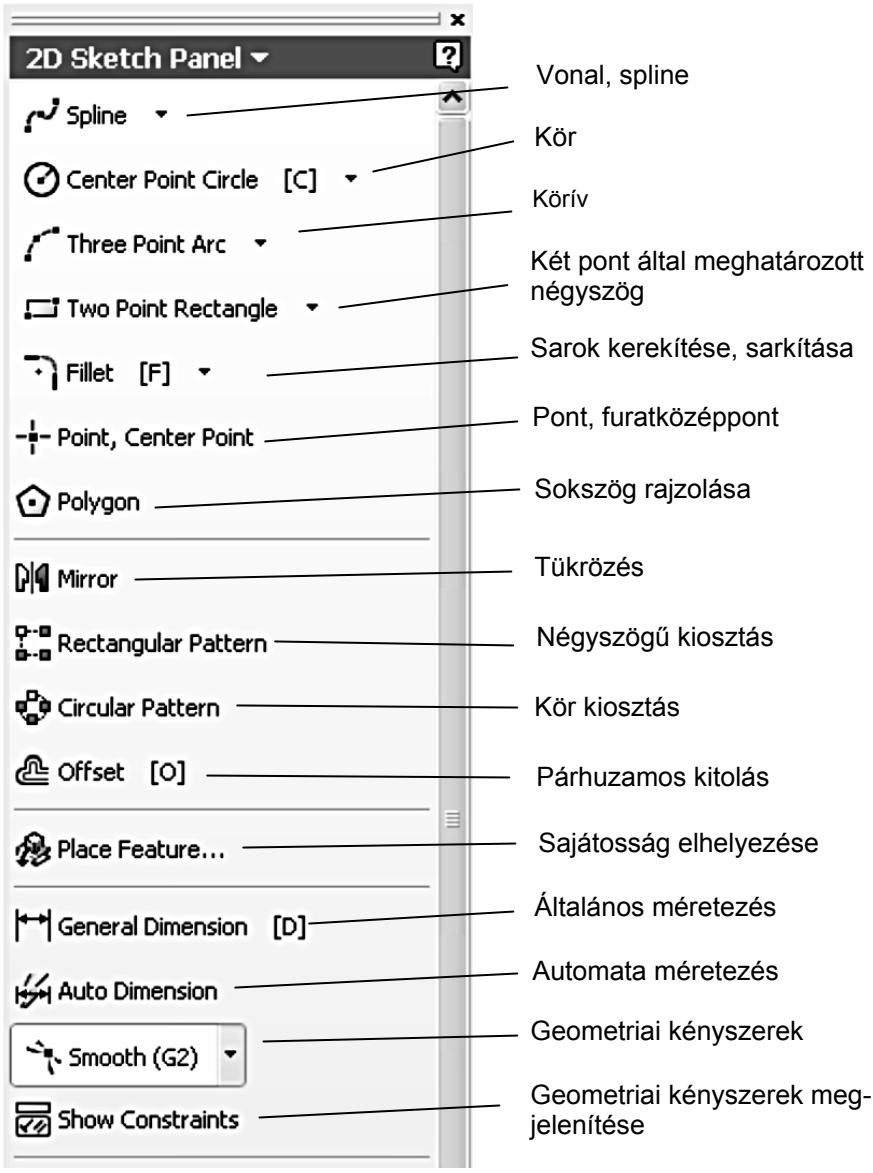
A vázlatkészítés megkönnyítésére már eddig is felhasználtuk azokat a lehetőségeket, amelyek a kényszerek bekapcsolásával rendelkezésünkre álltak. Akár a Vízszintes – Függőleges, akár a Párhuzamos – Merőleges párosítást kapcsoljuk be a Tools menü Application Options... menüsorára kattintáskor megjelenő **Options** párbeszédablak Sketch lapján, a **Constraint Placement Priority** részben, a vázlat készítésekor mindenképpen jelentkezik ezek előnyös hatása.

Ha az alapértelmezés szerinti beállítást hagyjuk érvényesülni, akkor a vonalak rajzolásakor mindig megjelennek a Perpendicular és a Parallel kényszerjelek, amelyek automatikusan hozzárendelésre kerülnek az adott rajzelemhez. A jelek abban is segítenek, hogy azonos hosszúságú rajzelemeket hozzunk létre, hiszen egy pontozott vonal jelzi, ha az aktuális vonal végpontja egy vonalba kerül egy korábban megrajzolt másik vonal végpontjával.

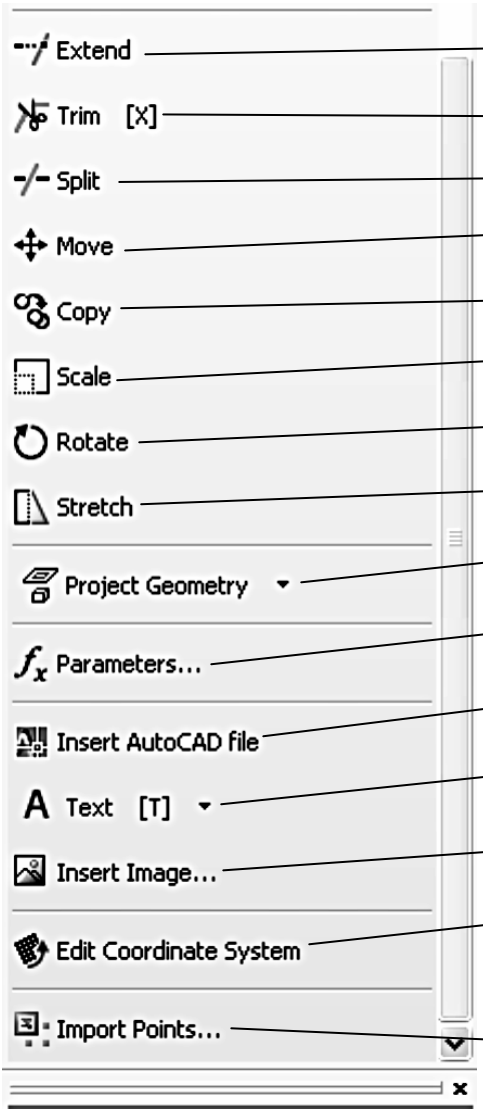
2.7. Vázlatkészítő és -szerkesztő parancsok. A Sketch Panel

A **Sketch Panel**-ben található vázlatkészítő parancsok a **2.15.a. és 2.15.b. ábrákon** láthatók. Az egyes menük mellett található kis legördülő nyilak jelzik, hogy legördítve azokat, újabb parancsokat találunk alatta (például a **Line**-t legördítve megtaláljuk a **Spline** parancsot is).

Az itt található parancsok rendkívül hasonlítanak az AutoCAD-ben használt parancsokhoz, csupán a műveletek sokkal rövidebbek és előre vannak szemléltetve a félreértés elkerülése véget. Emiatt a fenti műveletek közül csak néhányat mutatunk be.



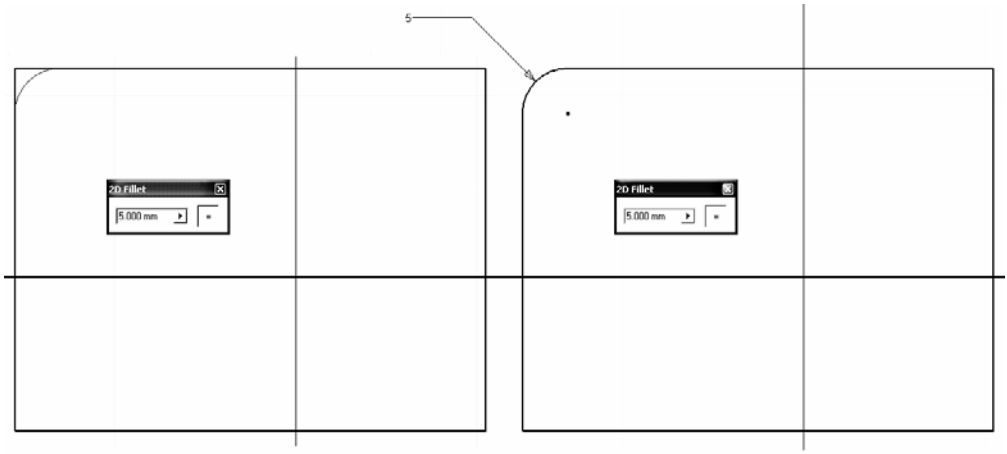
2.15.a. ábra
A Sketch Panel tartalma



2.15.b. ábra
A Sketch Panel tartalma

A Fillet parancs

A **2.16. ábra** egy négyszög sarkainak a lekerekítését szemlélteti 5 milliméteres sugárral. A sugár értékét a megjelenő ablakocskában írhatjuk be, itt mindig a legutoljára használt sugárérték jelenik meg.

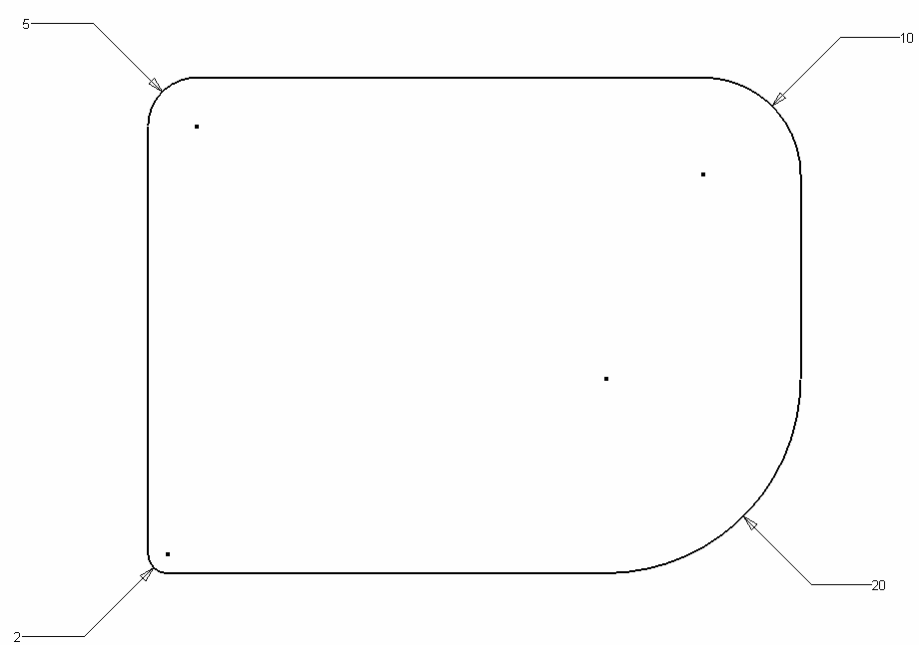


2.16. ábra
A Fillet használata

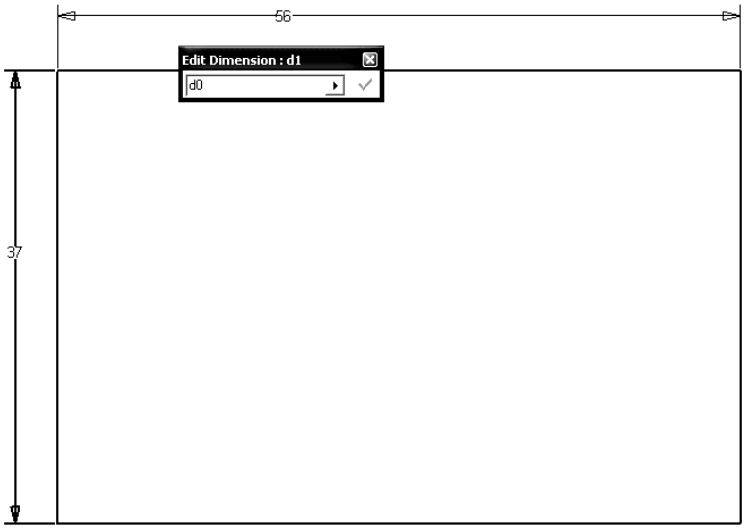
A bal oldali ábrán látható, hogy mielőtt rákattintanánk a második vonalra, csupán felé helyezve a kurzort, az piros színű lesz, és zöld színnel előre megjelenik a kerekítés geometriai formája is. Rákattintva a második vonalra is, a művelet befejeződik. **A parancs továbbra is aktív marad, mint minden Inventor parancs, amíg jobb egérgombbal a rajzfelületre nem kattintunk, és ott nem válasszukki a Done címszót.**

Ha egy sarok kerekítése után folytatni akarjuk a műveletet a többi sarkon is, egyszerűen változtathatjuk a sugár értékét és újabb rajelemeket kiválasztva, lekerekíthetjük a többi sarkot is, ha szükséges (**2.17. ábra**).

A **General Dimension** parancs lehetővé teszi a méretek egyenkénti feltevését, egyben lehetőséget nyújtva a méretek meghatározására az előző méretek függvényében. A **2.18. ábra** szemlélteti a méretek ilyen módon való használatát. Az ábrán láthatjuk, amint a második méretet feltéve a rajzunkon és az első (37 egységnyi) méretre kattintva, annak a jele töltődik be a második méret ablakába. Ez azt jelenti, hogy a d1-es méret mindig egyenlő lesz a d0-s mérettel, utólag bármikor változtatva az első méretet, a második mindig egyenlő lesz azzal. A méreteket megadhatjuk egy paramétertáblázatban is.



2.17. ábra
Sarkok kerekítése különböző sugárértékekkel



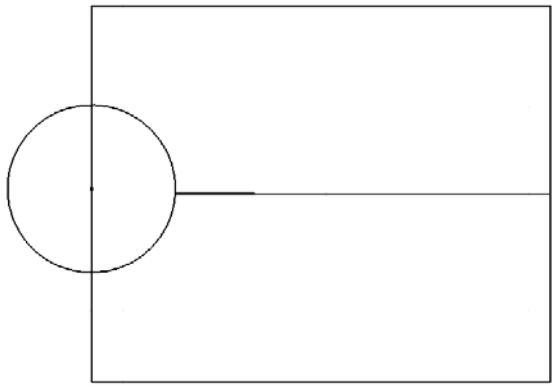
2.18. ábra
Méretezés egy már meglévő méret függvényében

2.19. ábra
Matematikai összefüggések használata a méretezés során

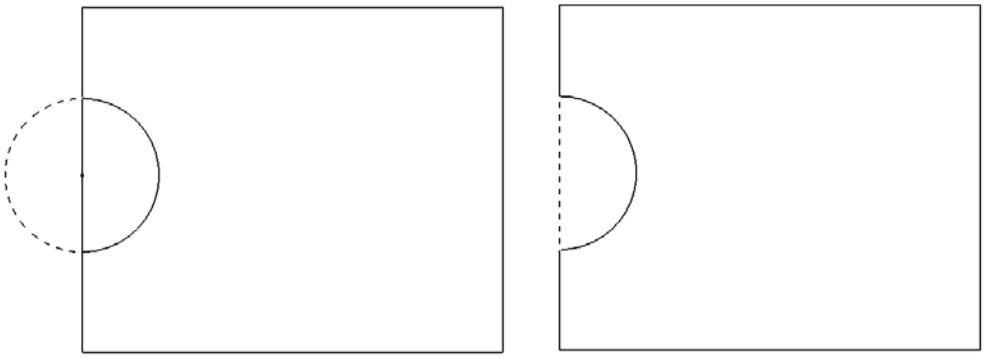
A méretek ugyanakkor kifejezhetők egyszerű matematikai összefüggésekkel is, mint a már meglévő méretek függvényeként. Ha például mindig azt akarjuk, hogy a rövidebb oldalhossz kétszerese legyen a hosszabb oldal mérete, akkor egyszerűen a d1 méret kazettájába beírjuk, hogy **2 * d0** (**2.19. ábra**). Természetesen ennél bonyolultabb kifejezések is írhatók.

Extend és Trim

Ha a megrajzolt elem túl rövid és hosszabbra van szükségünk, a meghosszabbítására lehetőségünk van **Extend** parancsot alkalmazásával.



2.20. ábra
Az Extend parancs használata



2.21. ábra
A Trim parancs használata

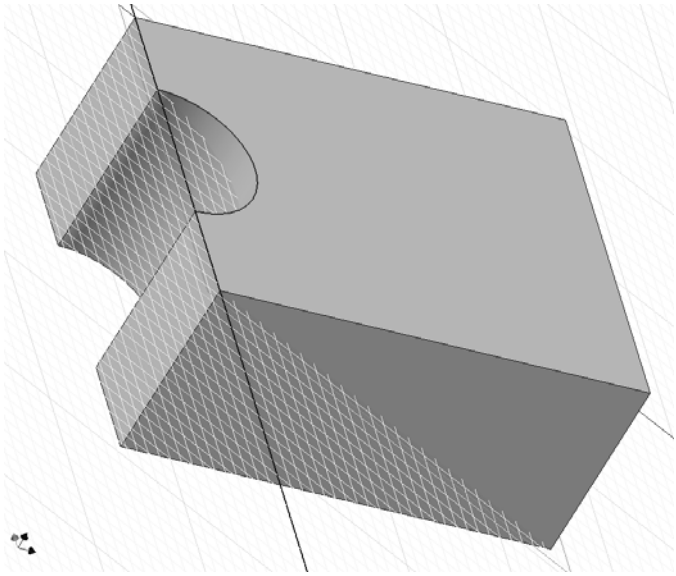
Az **Extend** parancsot elindítva, a hosszabbítandó elem felé mozgatva a kurzort, az elem piros színűvé válik, és a legközelebbi elemig való meghosszabbítása fekete színnel megjelenik. A parancs aktív marad továbbra is, újabb elemek kijelölését várva. Ha már továbbá nincs szükségünk rá, a művelet befejeződik a már ismertetett jobb klikk – Done módszerrel.

Gyakran előfordul, hogy a vázlatszerű rajzolás következtében a létrehozott elem egy részét törölni kell. Ez általában egy másik elemen túlnyúló rész, és **Trim** parancs alkalmazásával távolítható el. Ennek használatakor ráállva egy rajzelemre, szaggatott vonallal jelzi, hogy mekkora részt tudunk kivágni (**2.21. ábra**).

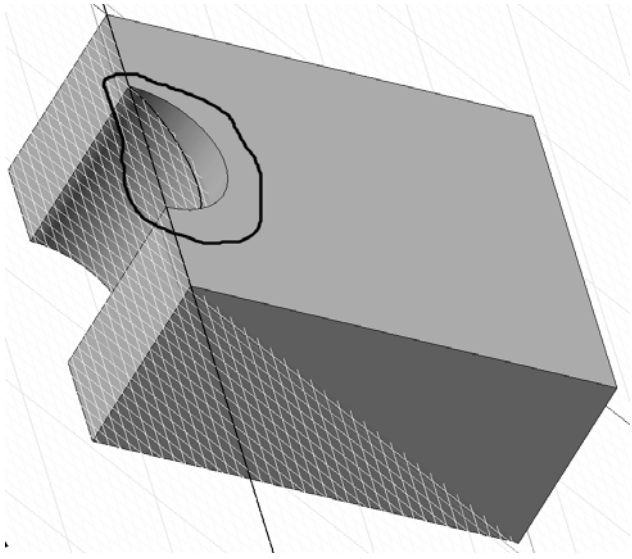
Project geometry

A parancs főleg 3D elemek éleinek vagy egyéb létező Inventor elemek levetítésére használható. A **2.22. ábrán** látható test kijelölt élét akarjuk egy új vázlatsíkra vetíteni.

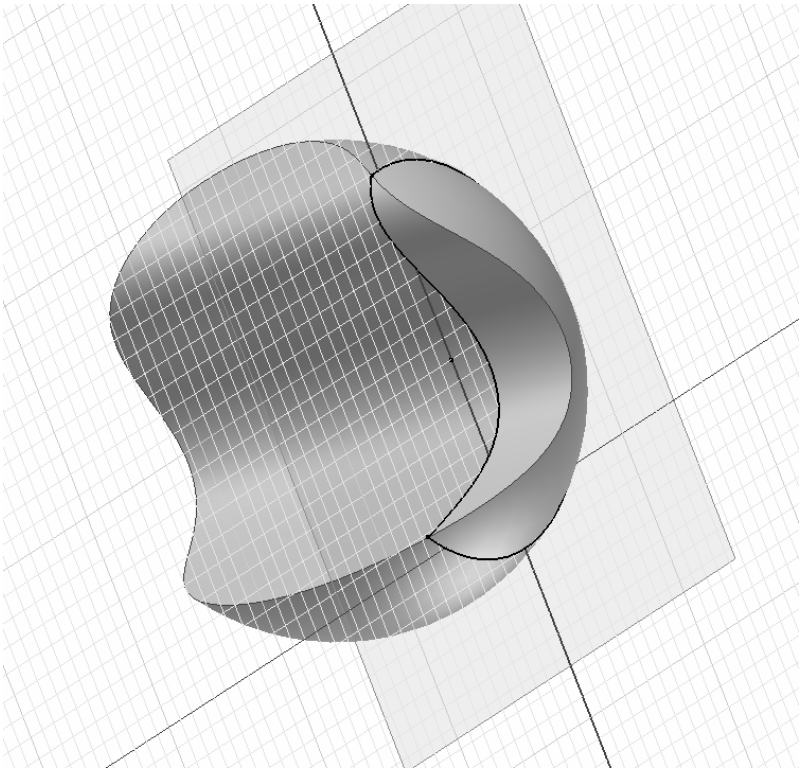
A **Project geometry** utasítás futtatása közben, ha a kurzort egy elem felé mozgatjuk, az a szokásos módon, piros színűre vált, ha pedig rákattintunk, akkor az aktuális 2D vázlatra vetítődik le az elem (**2.23. ábra**). A Project geometry parancs mögött található a **Project cut edges** parancs is, amely lehetőséget nyújt felületek és síkok metszésének meghatározására 3D testmodell részekkel. Példa látható ennek alkalmazására a **2.24. ábrán**, ahol egy gömbrészlet metszésvonalát határozzuk meg egy síkkal, amely nem megy át a gömb középpontján.



2.22. ábra
A Project geometry parancs indítása



2.23. ábra
A Project geometry utasítással létrehozott új rajzelem



2.24. ábra
A Project cut edges utasítás alkalmazása

A Project Cut Edges és Project geometry parancsok használatánál vigyáznunk kell arra, hogy a létrehozott geometriai elemek **referencia** elemek, linkelve vannak az alap geometriához. Ha a kapcsolatot meg akarjuk szüntetni közöttük, akkor jobb egérgomb kattintásra megjelenő menüből a **Break Link** parancsot kell futatnunk. Az így létrehozott elemek még nem változtathatóak, csak akkor, ha töröljük a kényszerek közül a vetítés során keletkezett vetítési kényszert.

2.8. Alaptest létrehozása profilból

A kényszerekkel és méretekkel meghatározott vázlatból – amit ezután profilnak nevezünk – alaksajátosságok hozzáadásával térbeli modellt tudunk létrehozni. Ezek az alaksajátosságok szerepüktől és alkalmazásuktól függően három fő csoportba sorolhatók:

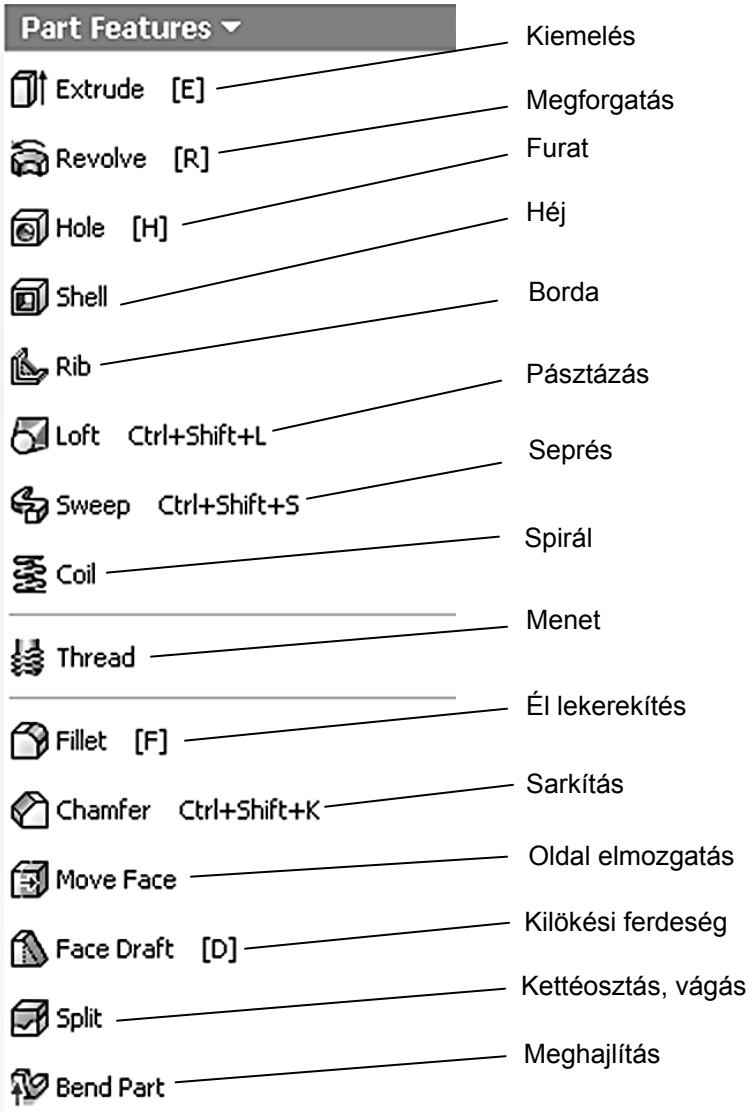
Sketched Features Vázlatra épülő alaksajátosságok	**Placed Features** Elhelyezett alaksajátosságok	**Work Features** Munka alaksajátosságok
A profilból kialakított alaptest létrehozására szolgálják	Az alkatrész részleteinek kialakítására használhatók.	Az elhelyezett alaksajátosságok helyzetének meghatározását teszik lehetővé
Extrud, loft, revolve, sweep, coil, rib, split	Hole, fillet, chamfer, face draft, shell, rectangular és circular pattern, mirror, thread	Work Plane, work axis, work point,

A parancsok a Part Featuress Panel-ből érhetők el, és a **2.25.a., 2.25.b. és 2.25.c. ábrákon** láthatók.

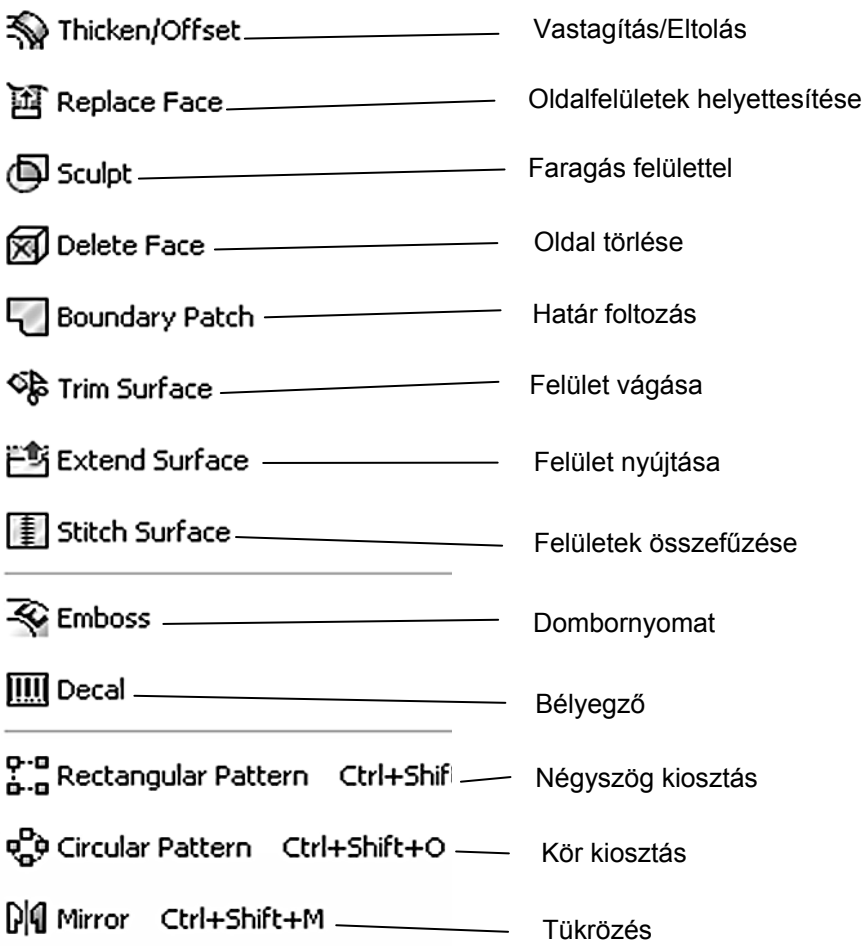
A szakszerű testmodellezés néhány elvét már a kezdetben figyelembe kell vennünk. Az alaptest létrehozását mindig úgy kell tekintsük, hogy az a test, amelyhez nem csak hozzáadni lehet sajátságokat, hanem nagyon sokszor, kivonni is, mely rendkívül előnyös lehet. Ezért ajánlatos minden alaptest létrehozása előtt, megfelelő időt áldozni arra, hogy elgondolkozzunk a modellezés további lépésein, úgymond előre meghatározzuk és „levetítjük", hogy a modellezés legvalószínűbb lépéseit. Egyszerűbb darabok esetén természetesen ez maga a modellezés lépéseinek a meghatározása lehet, de bonyolult darabok, szerelések esetén elégséges a fő irányvonalat meghatározni. Ajánlott mindig egy olyan alaptestnek az elkészítése, amely lehetővé teszi a jövőbeli modellezési lépéseinek során a szimmetria elemek használatát.

Rendkívüli fontos, hogy a modell elvi felépítése olyan legyen, hogy annak sajátosságait meg tudjuk változtatni (ha erre szüksége lesz) anélkül, hogy nagyon nehezen kiküszöbölhető modellezési konfliktusokat hozzunk létre.

Vigyáznunk kell, hogy alaposan ismerjük, és helyesen alkalmazzuk majd a vázlatokban megtett kényszereket, mert ezek nem megfelelő ismerete sokszor lehetetlené teszi az utólagos módosításokat.

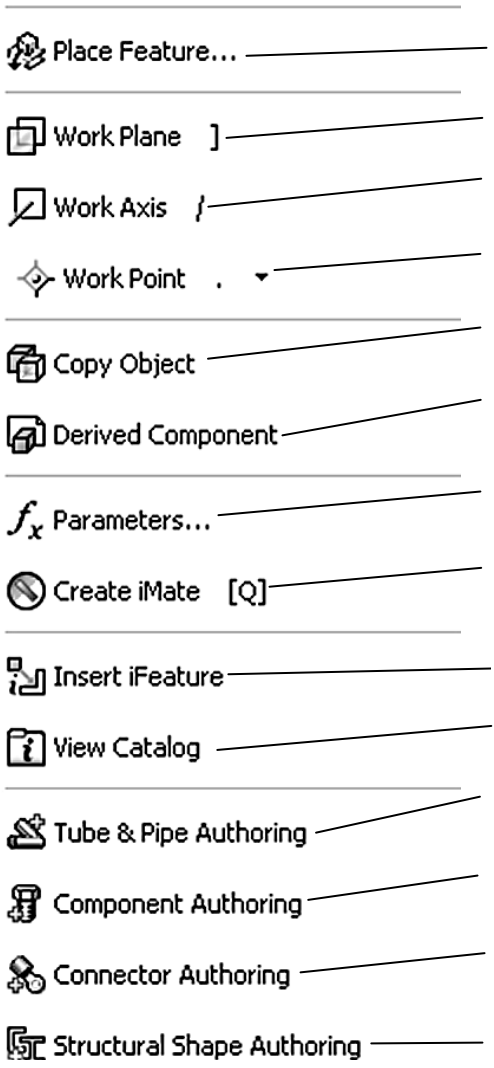


2.25.a. ábra
Az alaksajátosságok panelje (Part Featuress)



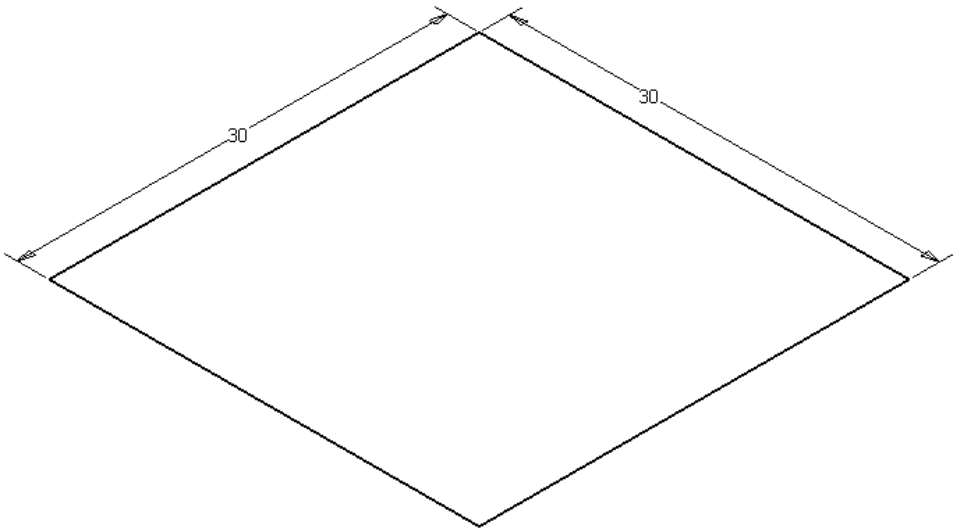
2.25.b. ábra
Az alaksajátosságok panelje (Part Featuress)

A 2009-es Inventor Professional-ig a felületek kezelése igen kezdetleges volt. Az új verzióban, amint az a **2.25.b. ábrán** is megfigyelhető, a program fejlesztői felületeket kezelő alaksajátosságokat készítettek. Ezek megteremtik a komplex felületeket kezelni képes munkaeszközöket, parancsokat, ezzel jelentősen kibővítve az Inventor felületi modellezési lehetőségeit.



2.25.c. ábra
Az alaksajátosságok Panel (Part Featuress)

Az alaksajátosságok bemutatásra tekintsük a **2.26. ábrán** látható 30–30 mm oldalhosszúságú négyszögprofilt.



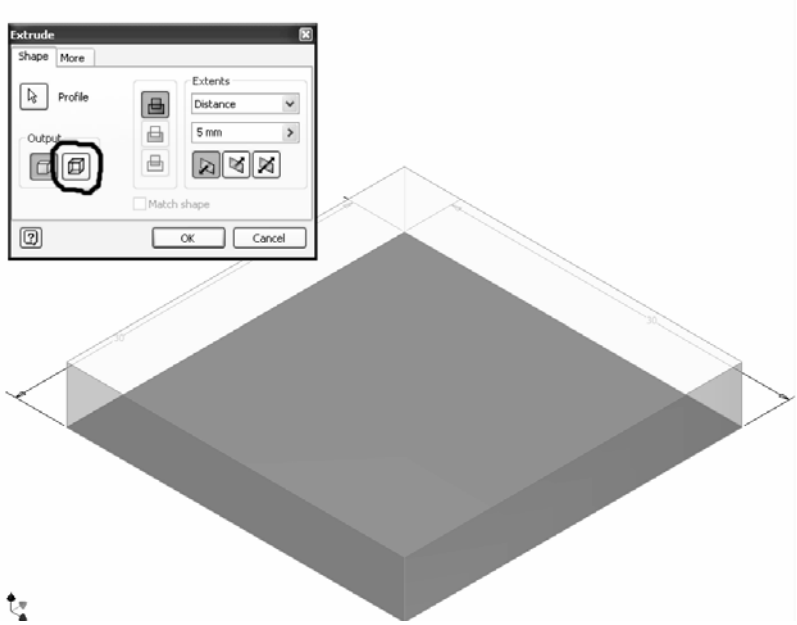
2.26. ábra
A kiindulási profil

2.8.1. Az Extrude (Kihúzás) sajátosság alkalmazása

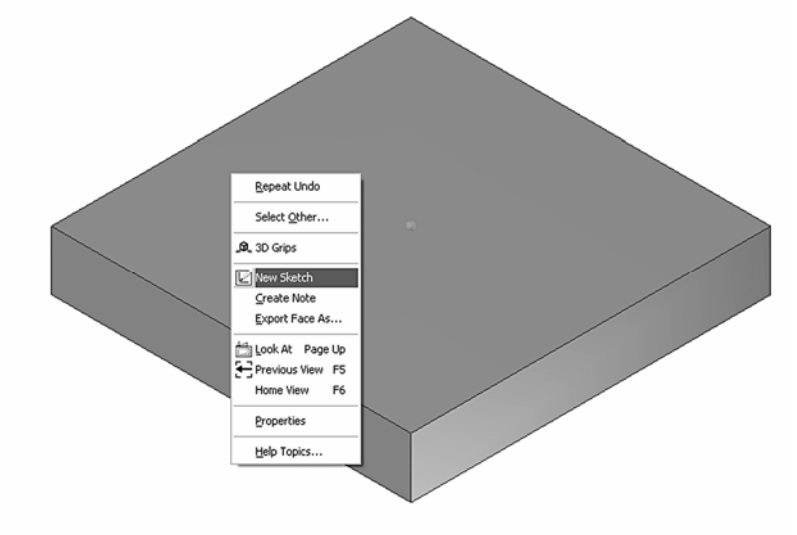
Az elkészült vázlatunkból, síkból történő kihúzással hozzuk létre a modell alapját képező szilárd testet. A kihúzást a vázlatsíktól felfelé mutató irány szerint, 0 fok szűkítési szög meghagyásával végezzük el.

A megjelenő modell alapbeállítás szerint az anyag szerinti színnel kitöltve jelenik meg (**2.27. ábra**).

A **2.27. ábrán** figyeljük meg a bejelölt ikont: minden vázlatra épülő alaksajátosságnak a párbeszédablakában megtalálható. Ez a felületek létrehozását teszi lehetővé. Ha profilból vagy profilokból ilyen ikon kijelölésével akarunk testet képezni, nem lehetséges, ha zárt is az alakzat csak a felületi burok jön létre. Ha a profilunk egy nyitott profil, akkor az alaksajátosság parancs indításakor, automata módón ez az ikon marad csak aktív. Ilyenkor a párbeszédablakban egy kis piros kereszt jelenik meg, amely segítségével szerkeszthető a nyílt profil, bezárható és ezután már lehetséges a testmodell képzése a vázlatból. Gyakori hiba, e kis jel figyelmen kívüli hagyása, ilyenkor a kezdő modellező nem érti, hogy miért nem lehetséges a testmodell létrehozása, vagy figyelmetlenség következtében akár felületet is generál, testmodell helyett, a felhasználó. Ne felejtsük el, hogy a létrehozott „véletlen” felület szerkeszthető, a profil zárása után a modell kiindulási profilját újra kijelölve, végül is 3D testet nyerhetünk.



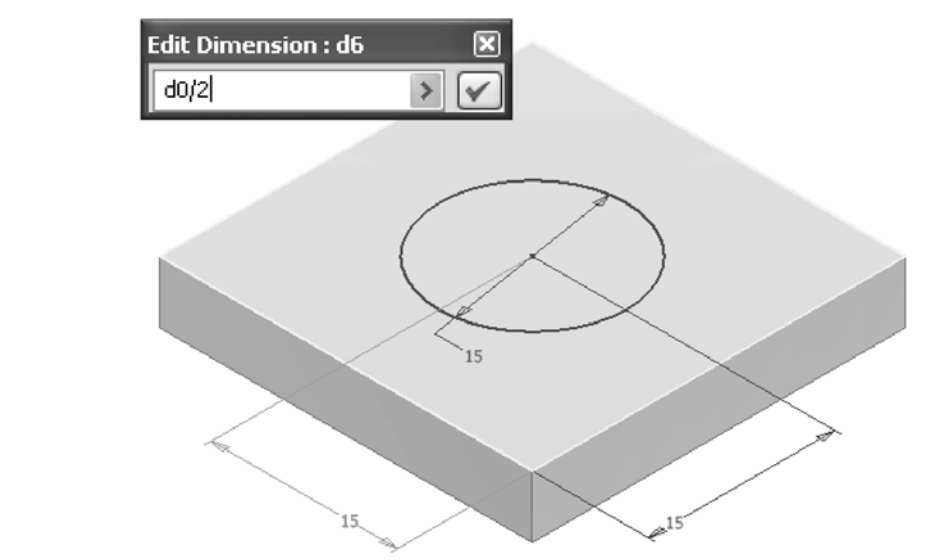
2.27. ábra
Az első Extrude művelet



2.28. ábra
Új vázlat létrehozása, előzetes vázlatsík kijelölésével

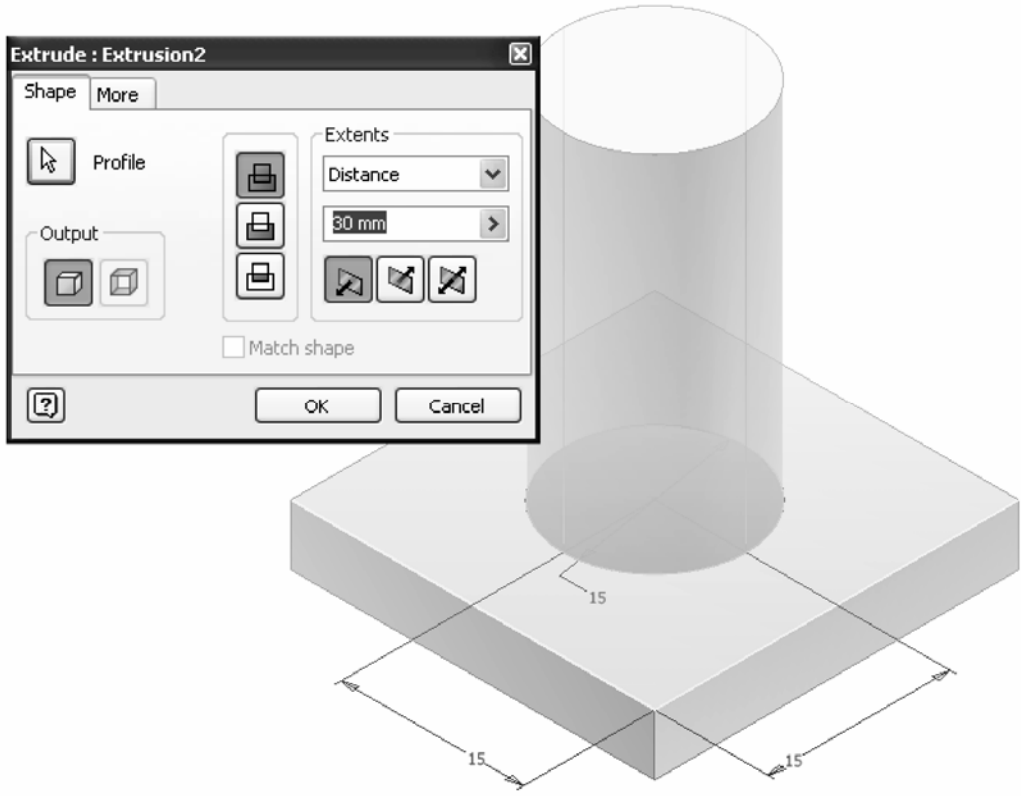
Az alaptest létrehozása után, további részekkel egészítjük ki a modellt, amelyhez szükséges egy újabb vázlat rajzolása. Ehhez ki kell adnunk a **New Sketch** parancsot, és ki kell jelölnünk a modell egyik lapját vázlatsík céljára. A New Sketch parancs a jobb egérgombbal is megjeleníthető helyi menüben is kiválasztható. Ha a felületet előre kijelöljük egy kattintással, akkor a menü kevesebb választást kínál fel, ahogy ez az is ábrán látható (**2.28. ábra**).

A parancs kiadásakor – a beállítástól függően – újra megjelenik a háló, valamint a 2D Sketch Panel ikonjai. Az új vázlatsíkra rajzoljunk így egy kört tetszőleges középpont kijelöléssel és tetszőleges mérettel. Ennek méreteit a General Dimension eszközzel adjuk meg. A General Dimension kiválasztása után, az átmérő méretét a körre kattintva adhatjuk meg, a hasáb oldalhosszának feleként. A középpont helyzetének meghatározásához a modell élére és a körre kell kattintanunk, hogy a két oldalhossz közepén legyen (**2.29. ábra**).



2.29. ábra
Az új, kör alakú profil létrehozása

Az alkatrész készítését újabb kihúzással folytatjuk. Először is nyomjuk meg a Return gombot (a Finish Skecth helyett), és ezzel ugyanúgy lezárjuk a vázlatot. Az Extrude ikonra való kattintáskor megjelenő párbeszédablakban (csak a Distance értékét kell megváltoztatnunk) írjunk be 20 mm értéket. Azonban nekünk kell a profilt kijelölnünk, mert választható lenne a négyszögnek a körön kívüli része, illetve maga a kör (**2.30. ábra**).

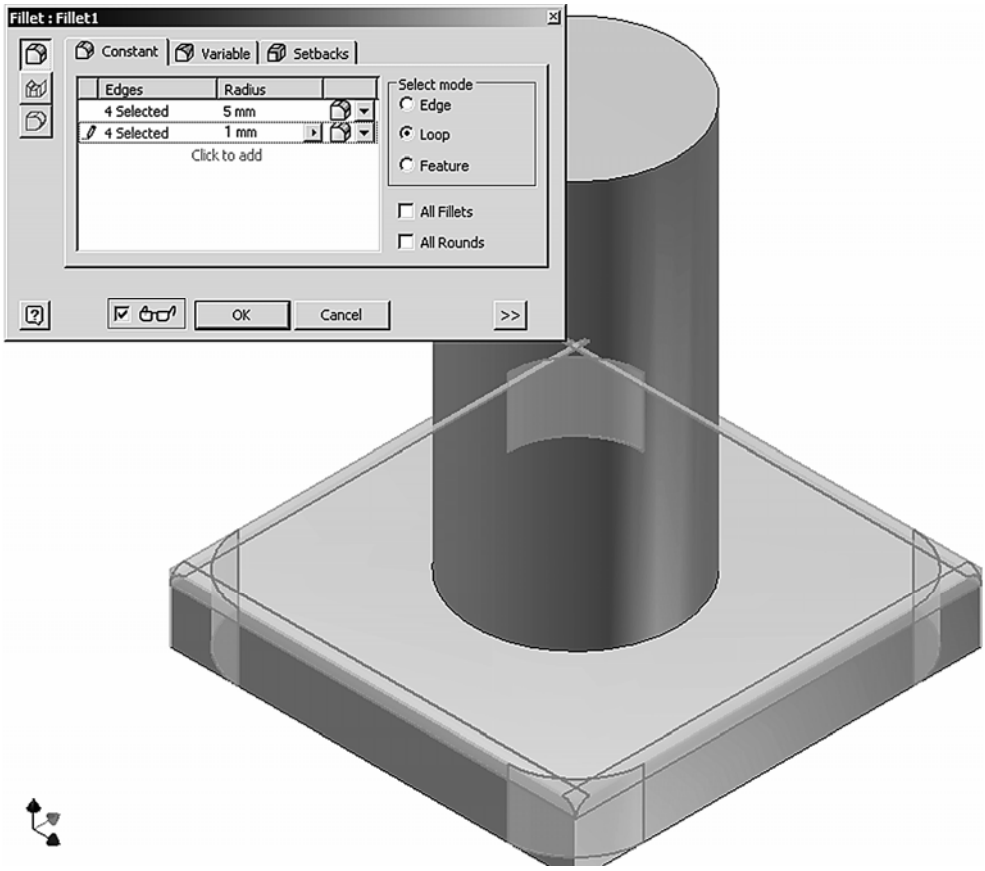


2.30. ábra
A körprofil kihúzása

2.8.2. A Fillet (Éllekerekítés) sajátosság alkalmazása

A hengeres rész hozzáadása után újabb sajátosságot alkalmazunk. Az elsőként kihúzott rész felső lapja és a hengeres rész közös élénél alkalmazunk egy 2 mm-es sugárral való lekerekítést (Loop kijelölve), valamint a négy függőleges él is le lesz kerekítve 5 mm-es sugárral. Az eltérő méretek ellenére mind az öt él egyetlen parancskiadással lekerekíthető.

A parancs kiadásakor az alapértelmezés szerinti 1 mm sugár jelenik meg a párbeszédablakban, amit változtatás nélkül alkalmazhatunk. Rákattintva a henger és a síklap közös élére, piros színnel megjelenik a kijelölésnek megfelelő lekerekítés két kis jelképpel (**2.31. ábra**).



2.31. ábra
A lekerekítés folyamata

Ezt követően a párbeszédablak Click here to Add sorára kattintva, lemásolásra kerül az első sorba beírt sugárméret, amelyet át kell írnunk 5 mm értekre. Ha a beírást az ENTER billentyű megnyomásával elfogadjuk, kijelölhetjük azokat az éleket, amelyekre ezt az újabb értéket érvényesíteni akarjuk.

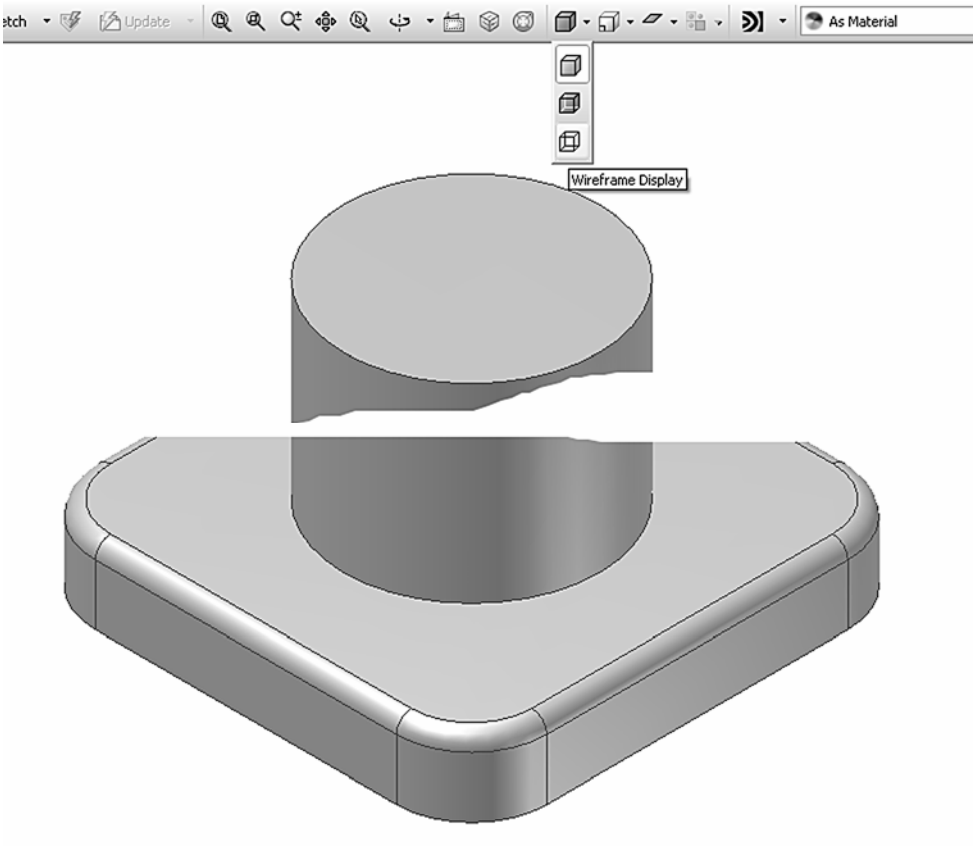
Az öt él kijelölése után az OK nyomógombra kattintáskor a megadott sugárral kerül végrehajtásra a lekerekítés művelete.

A lekerekítések végrehajtását követően újabb vázlatot kezdünk, amit a jobb egérgombbal megjeleníthető helyi menüből indítunk. Rákattintva New Sketch feliratra, ki kell jelölnünk egy felületet, amelyen vázlatot akarunk készíteni. Ha a négyzet alakú rész alsó élére kattintunk, akkor a hozzátartozó függőleges lapot kínálja fel a program. Ha nem fogadjuk el, akkor rövid idő múlva megjelenik

egy léptetőikon, amelynek a közepére kattintva az aktuális kijelölés fogadható el, a jobbra mutató nyíl segítségével pedig, kijelölhetjük a következő felületet. Most az utóbbira lesz szükségünk, ezért a használni kívánt felület kiemelésekor a léptetőikon közepére kattintunk.

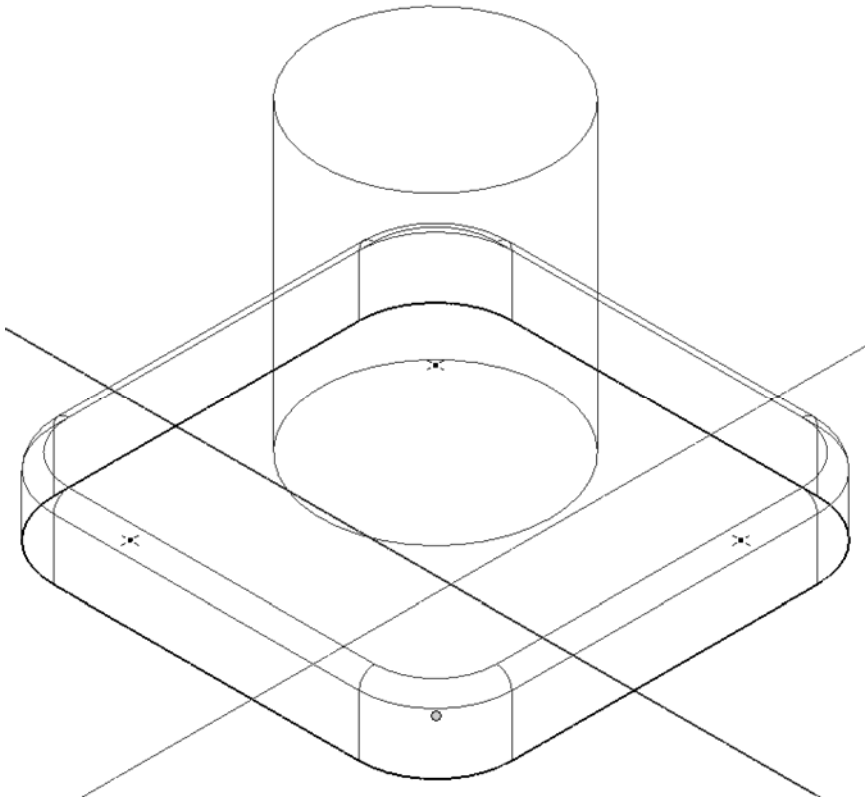
2.8.3. A Hole (Furat) sajátosság alkalmazása

A létrehozott vázlatsíkon most furatközpontokat akarunk létrehozni a Point, Center Point center paranccsal. Felhasználhatjuk erre a kerekítések középpontjait, de előbb drótvázas nézetre kell áttérnünk, ami a **2.32. ábrán** látható.



2.32. ábra
A Wireframe paranccsal drótvázas nézetet aktiválunk

Indítva a Point, Center Point ikonról a parancsot a kurzorral a kerekítések középpontja felé érve, azok zöld színűek lesznek, rákattintva pedig kis keresztek jelennek meg rajtuk (**3.33. ábra**).

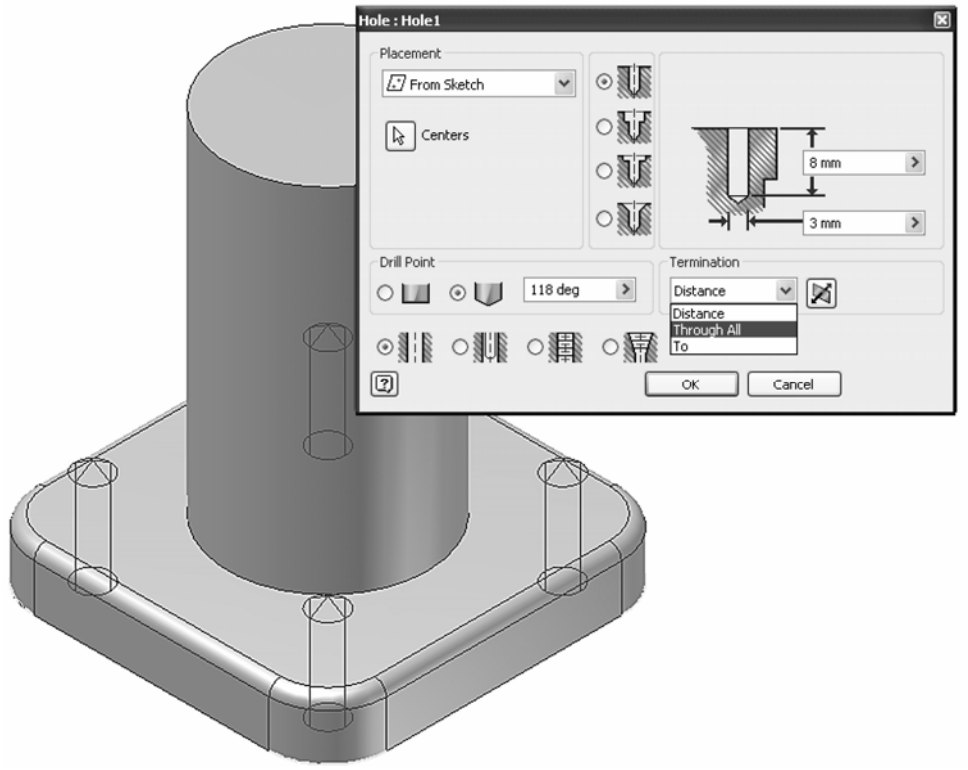


2.33. ábra
A furatközéppontok kijelölése

A furatok létrehozásának van egy rövidebb útja is. Ha a vázlaton nem hozunk létre furatközpontokat, befejezzük a vázlatot a Return gomb megnyomásával, és a megjelenő Feature panelből a Hole parancsot indítjuk. Ekkor a megnyílt ablak furatközéppontok kijelölését várja („Be van nyomva" Centers gomb). Most kijelölhetjük a kerekítési körívek középpontjait, ami a **2.34. ábrán** látható**.**

Ezt követően kiválasztjuk a Termination lapon, a Through All (Átmenő) opciót. Továbbá a párbeszédablak jobb oldali részén lévő szemléltető ábrán, a méretek beállítása következik. Ehhez a bemutató ábrán megjelenő méretre duplán kell kattintani, majd az inverz módú megjelenéskor megadhatjuk az új érteket. Állít-

suk ezt 5 mm értékre. Az OK nyomógombra kattintáskor megjelennek a végleges furatok és kilép a program a vázlatmódból.



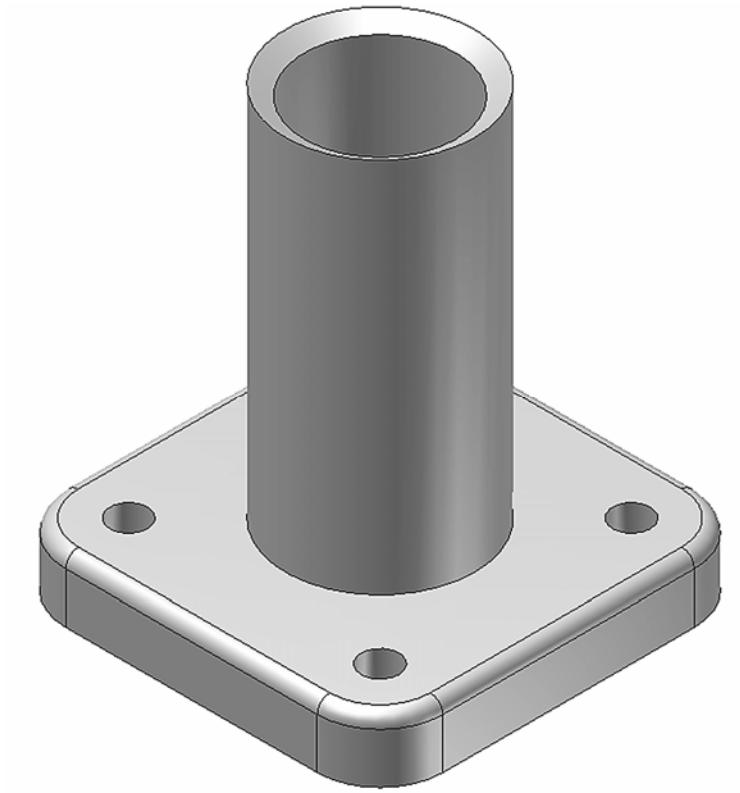
2.34. ábra
A Hole sajátosság párbeszédablaka

Még egy furatra lesz szükségünk, de ehhez ki kell jelöljük a középpontot. A képernyő felső részén lévő parancstárból indítjuk a Sketch parancsot és kijelöljük a modell hengeres részének a felső lapját.

A vázlat panel Point, Center Point ikont megnyomva, kijelöljük a felső lap középpontját, majd következhet újra a Hole parancs és a párbeszédablak a méretbeállításokkal. Az előző beállításhoz képest, csak az átmérő méretét írjuk át 12 milliméterre. Az utolsó lépés eredménye látható a **2.35. ábrán**, immár a következőkben létrehozott sarkításokkal és visszatérve az árnyalt megjelenítési módra.

2.8.4. A Chamfer (Élletörés) sajátosság alkalmazása

A Chamfer parancsot a Part Features panelben találhatjuk meg, amely – a korábbi sajátosság parancsokhoz hasonlóan – párbeszédablakot jelenít meg. Ebben a bal oldali részen lévő három ikon közül jelöljük ki a középsőt, amely távolság és szög megadásával hozza létre a sajátosságot. A másik kettővel egyenlő-, illetve különböző élhosszúságú letörést végezhetünk. Ennél a módnál elsőként a letörésben közös felületet kell kijelölnünk, ahogy azt az automatikusan bekapcsolódó Face (Lap) kapcsoló is jelzi.



2.35. ábra
A darab az utolsó, letörési művelet után

A hengeres rész gyűrű alakú felső lapjának kiválasztása után – amelynek következtében zöldre vált a felület – a belső élt jelöljük ki, a 3.35. ábra szerint. A Distance legyen 1.5 mm, az Angle (Szög) pedig 30o. Az OK nyomógombra kattintva végrehajtásra kerül a sajátosság elhelyezése.

2.9. Sajátosságok

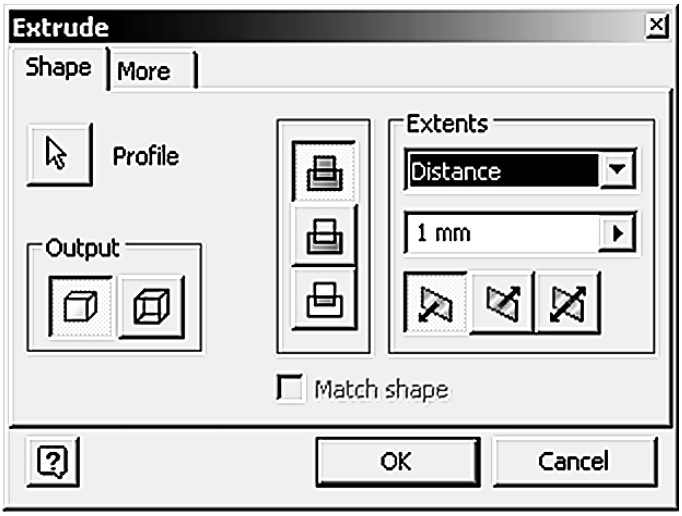
2.9.1. Az Extrude (Kihúzás) sajátosság

A Extrude (Kihúzás) parancs használatának feltétele, amint láttuk az előzőekben az, hogy legyen egy aktív vázlat, amelyen elvégezhető a művelet. A sajátosság párbeszédablaka a **2.36. ábrán** látható.

Shape (Alak) lap

A kihúzással kapcsolatos legfontosabb beállítások ezen a lapon végezhetők el.
Profil
A megjelenő párbeszédablak bal oldalán a kihúzni kívánt profil kiválasztására szolgáló ikonnal jelölhetjük ki azt a területet, amelyre érvényesíteni kívánjuk a kihúzást. A profil kiválasztására azonban csak akkor van szükségünk, ha több választási lehetőség is adódna.
Output
Ebben a részben két ikon közül választva állíthatjuk be, hogy a kihúzás eredményeként Solid vagy Surface (felület) jöjjön létre.

A párbeszédablak középső részén három ikon áll rendelkezésünkre annak meghatározására, hogy a kihúzás során létrehozott sajátosság miként viszonyuljon a korábban létrehozott modellrészekhez.



2.36. ábra
Az Extrude sajátosság párbeszédablaka

Egyesítés – a meglevő modellhez hozzáadódik az új rész

Kivonás – már létező modellből lesz kivonva az új rész

Közös rész – a meglévő és az új rész közös része jön létre

Amikor egy alkatrész létrehozása során az első sajátosságot hozzuk létre (bázis-sajátosság), akkor csak a legfelső ikon használható.

Extents (Terjedelem)
A párbeszédablak jobb oldali részén határozható meg a kihúzás mértéke, amely során meghatározhatjuk a kihúzás lezárásának módját, illetve megadhatjuk a kihúzás mértékét. Alapesetben a Distance (Távolság) adható meg a beíró területen, de választhatjuk a további lehetőségeket is, amelyek során a kihúzást egy adott síkig, síktól – síkig, vagy mindenen átmenően adhatjuk meg.
Distance (Távolság)
A kihúzás méretét az alatta lévő második sorban adhatjuk meg. Ez az alapértelmezett beállítás.
To Next (Következőig)
Kiválaszthatjuk azt a következő lehetséges lapot vagy síkot, mely lezárja az adott irányú kihúzást.
To (Addig)
Egy felületet vagy síkot kell kiválasztanunk, amely lezárja a kihúzást.
From To
A kihúzás lezárására meghatározhatunk egy kezdő- és záró felületet.
All (Mindenen keresztül)
Kihúzható vele a profil minden más sajátosságon és vázlaton keresztül a meghatározott irányba. Az Egyesítés művelet ilyenkor nem alkalmazható.

Az érték beírására szolgáló terület alatt a kihúzás irányát meghatározó ikonok találhatók. Valamelyikre rákattintva, ideiglenesen, zöld színnel megjelenik a művelet eredménye; a vázlatsíktól induló két irányon kívül választhatjuk azt a lehetőséget is, hogy a megadott távolság felével szimmetrikus kihúzás jöjjön létre.
Taper (Szög alatti kihúzás)
A párbeszédablak More (További opciók) lapján állítható a szűkülés szöge. A grafikus ablakban egy szimbólum jelzi az irányt és a rögzített éleket. Pozitív szűkülési szög esetén a kijelölt profil területe növekszik, a kihúzás során negatív szűkülési szög esetén a területe csökken.

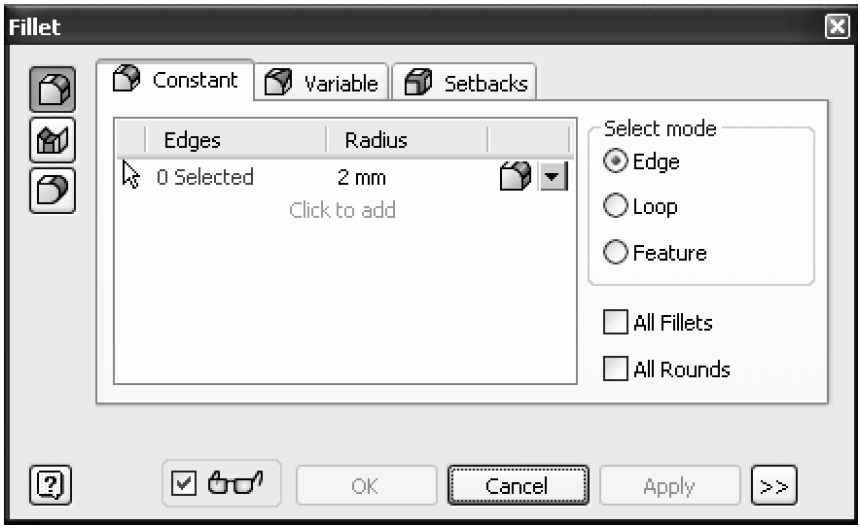
2.9.2. A Fillet (Lekerekítés) sajátosság

A példában a leggyakrabban használt állandó sugarú lekerekítés alkalmazására került sor, de ugyanakkor a párbeszédablakban további lehetőségeket is beállíthatunk.

A párbeszédablak megnyitásakor (**2.37. ábra**) a választható lapok ikonjából és feliratából látható, hogy létrehozhatunk állandó és változó sugarú, valamint különböző méretű lekerekítést egyetlen műveleten belül.

Constant (Állandó) **lap**

Itt a legáltalánosabban használt lekerekítések létrehozásához adhatunk meg méreteket. Ahogy a példában is látható volt, egyetlen parancskiadással több méretet is érvényesíthetünk az állandó sugarú lekerekítések létrehozásához.
Edges (Élek)



2.37. ábra
A Fillet párbeszédablak

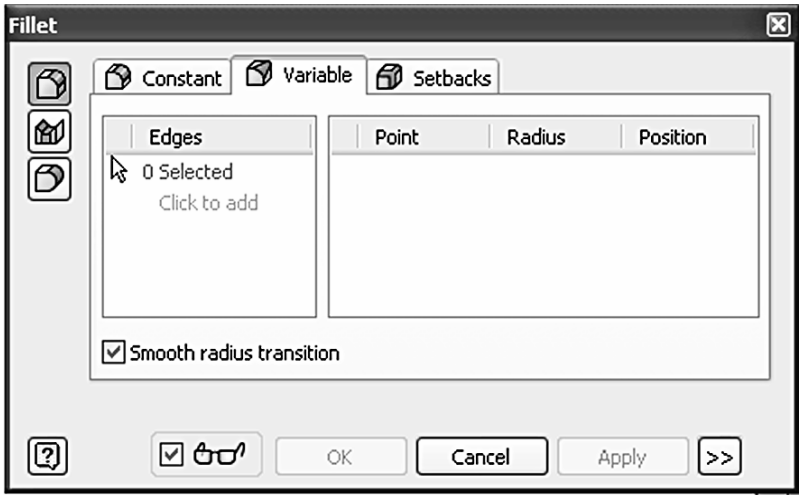
A lekerekítések helyének meghatározásához az alkatrész éleit kell kijelölni. Ha olyan él is kijelölésre került, amelyet mégsem akarunk lekerekíteni, akkor ezeket az éleket úgy tudjuk eltávolítani a kijelölési halmazból, hogy Ctrl billentyű lenyomva tartása mellett rákattintunk az adott vonalra.
Radius (Sugár)
A kiválasztott élek lekerekítési sugarát kell megadni ebben a részben.
Select mode (Kiválasztási mód)

Ebben a részben meghatározhatjuk, hogy egyetlen élre rákattintva mi kerüljön kijelölésre.

- **Edge** – egyenként jelölhetjük ki az éleket.
- **Loop** – egy lapon lévő zárt hurok éleit jelölhetjük ki egyszerre.
- **Feature** (Sajátosság) – egy létrehozott sajátosság éleit jelölhetjük ki egyszerre.
- **All Fillets** (Minden él) – egy alkatrész minden élét kijelölhetjük egyetlen kattintással.
- **All Rounds** (Minden sarok) – egy alkatrész minden élét és sarkát kijelölhetjük egy kattintással.

Variable (Változó) lap

Változó sugarú lekerekítések létrehozásához adhatunk meg méreteket (**2.38. ábra**).



2.38. ábra
Változó lekerekítés párbeszédablaka

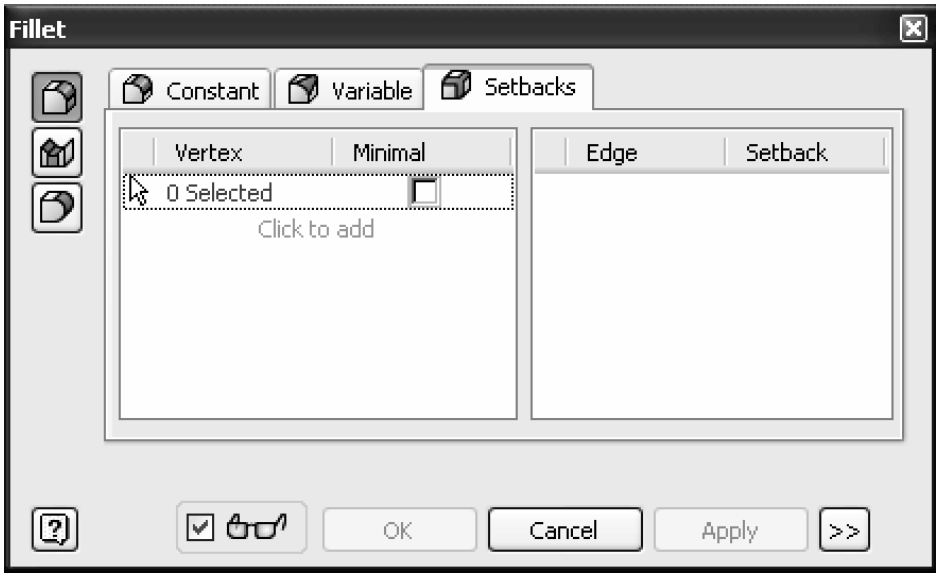
Edges (Élek)
A kiválasztott élek száma jelenik meg ebben a részben, illetve a Hozzáadás (Click to add) szóra kattintással további éleket jelölhetünk ki, a korábbiaktól eltérő kiinduló sugárral.

Pont (Point)
A változó sugárméret megadásához kijelölhetünk pontokat. A **Position** (Helyzet) mezőben megjelenő érték azt mutatja meg, hogy az elhelyezett pont milyen távolságra van az adott élen a kezdőponttól.

Sugár (Radius)
Itt adjuk meg, hogy a kijelölt pontban mekkora legyen a lekerekítés sugara.
Position (Elhelyezés)
A kijelölt pont helyzetét változtathatjuk meg. A 0 (kezdőpont) és 1 (végpont) között megadható érték az él hosszának arányában adja meg a pont helyét.
Smooth radius transition (Sima sugárátmenet)
Ha be van kapcsolva, akkor a pontok között fokozatos átmenettel jön létre a lekerekítés. Kikapcsolt állapotban a pontok között egyenletes átmenettel rendelkező lekerekítést készít a program.

Setbacks lap

Metsző élek lekerekítésekor folyamatos érintőjű átmenet hozható létre akkor is, ha minden élnek egyedi sugárértéket adunk meg (**2.39. ábra**).

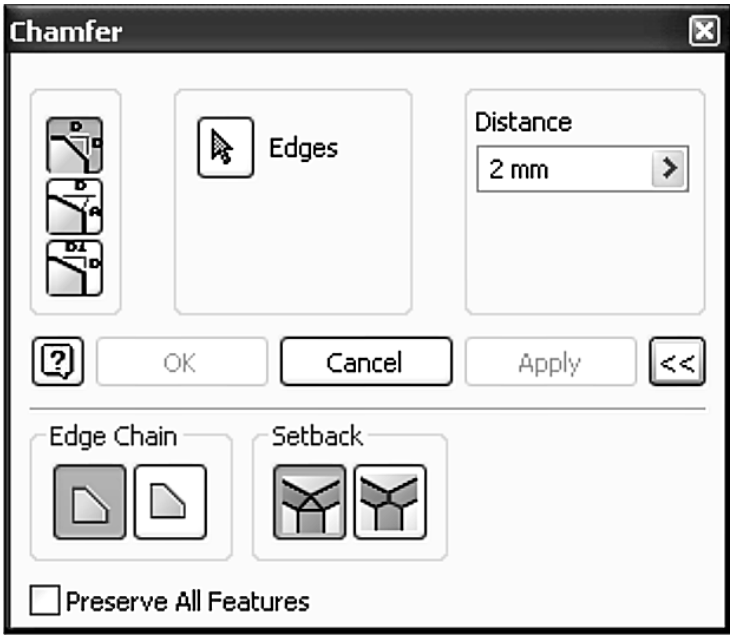


2.39. ábra
Setbacks lap

Vertex (Csúcspont)
A metsző élek csúcspontját jelölhetjük ki.
Edges (Élek)
Egy közös vonalban találkozó oldalak éleinek egyike kerül kijelölésre.
Setback (Egyedi sugár)
Itt kell megadnunk az egyedi sugarak hosszát.

2.9.3. A Chamfer (Élletörés) sajátosság

A párbeszédablakban megadott értékekkel, alkatrészek éleinek letörése végezhető el ezzel a sajátossággal (**2.40. ábra**).



2.40. ábra
Chamfer párbeszédablak

 Élletörés két azonos letörési hossz megadásával

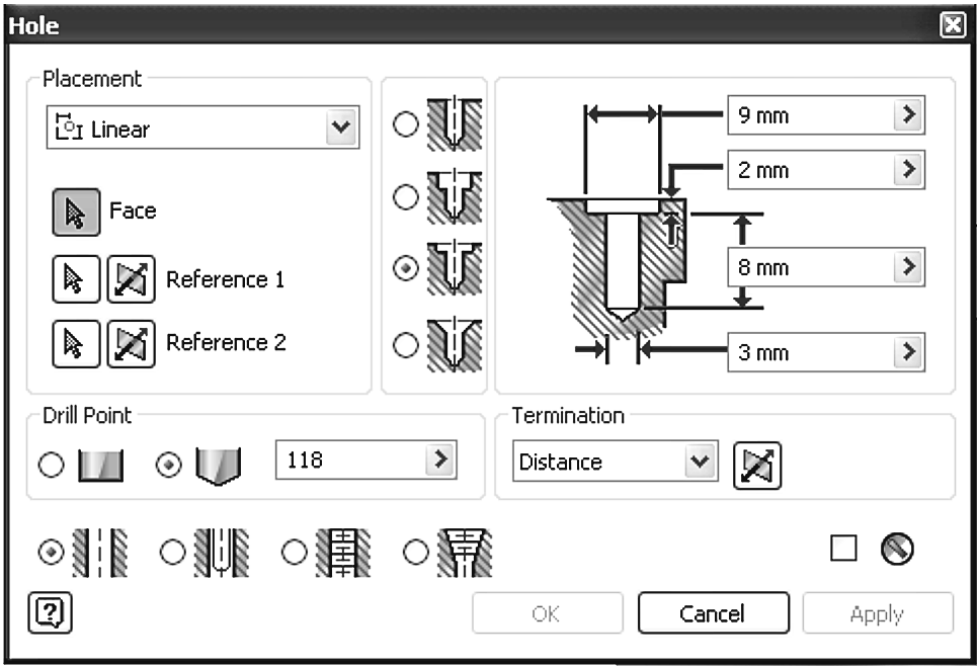
 Élletörés egy letörési hossz és egy szög megadásával

 Élletörés két nem azonos hossz megadásával

Ha megnyomjuk a párbeszédablakban a „<<” jelű ikont, a legördült menüben az **Edge Chain** és **Setback** (élláncok és egyedi élletörés) ikonokat találjuk. Az Edge Chaine alatt az egyik ikon az érintőfolytonos élek, a másik az egyedi élek kiválasztására szolgál. Ezeket a Fillet-nél már ismert módon kell kezelni.
Edges (Élek)
Az élek egyenkénti kijelölését teszi lehetővé.
Distance és Angle (táv és szög)
Az élletörés fajtájától függően távolság és szög megadását kell megadnunk.

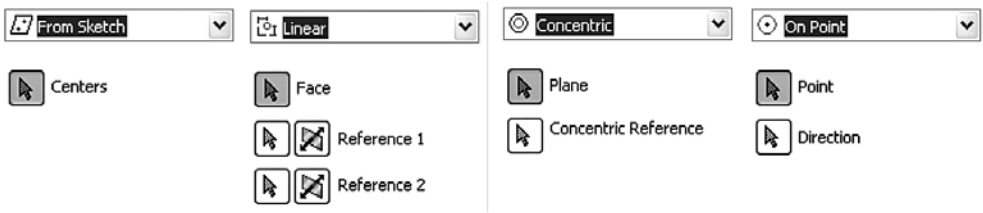
2.9.4. A Hole (Furat) sajátosság

Különböző furatsajátosságok létrehozásakor a parancs kiadása után megjelenő párbeszédablakban határozhatjuk meg, a méreteken kívül, a furatok kiviteli módjára vonatkozó utasításokat is. A furatsajátosság létrehozásakor szükség van egy vázolt furat középpontra (Point, Center Point), de kijelölhetők a furatok középpontjaként az alkatrészen található végpontok vagy középpontok, ha vázlatmódban dolgozunk (lásd az előző példát). A Hole párbeszédablak a **2.41. ábrán** látható.



2.41. ábra
Furatok párbeszédablak

Placement
A furatközpontok kijelölésének 4 alapvető módszerét állíthatjuk: **From Sketch** – vázalat alapján, **Linear** – egy oldal és két referenciairány alapján, **Concentric** – vázlat vagy sajátossággal koncentrikusan, **On Point** – egy oldalon tetszőlegesen meghatározott pontban (**4.42. ábra**).
Furatok típusa
Az Inventor csak 4 ikonnal jelöli a furatok négy alapvető típusát, a párbeszédablak középső részén. A furatok típusa lehet sima, részben sima, hengeres süllyesztésű és kúpos süllyesztésű.



2.42. ábra
A furatok elhelyezésének lehetséges módjai

Termination (Lezárás)
Itt állítható be a furat lezárása is, amelyet a legördülő listából választhatunk ki.

Distance (Táv)
A furat mélységét az ábrán megadott számértékkel határozhatjuk meg.

Through All (Átmenő)
A furat a teljes alkatrészen átmenően jön létre.

To (Addig)
Az itt megadott síklap kijelölésével határozhatjuk meg a furatot.

Drill Point
A furat végződésének típusa állítható be. Lehetséges estetek a Flat és az Angle. Az utóbbin kijelölhető a furatfenék dőlésszöge is.

A párbeszédablak alsó részében beállítható a furatok sima vagy 3 menetes változata.

Simple (Sima) – Egyszerű, fenekes vagy átmenő furatok készítése.

Clearance Hole – Könnyítési furatok.

Tapped Hole – Menetelt furatok. Rákattintva lenyílik a **Thread** párbeszédablakrészlet, amelyben a különböző menettípusokat és adatokat állíthatjuk be.

Tapper Tapped Hole – Kúpos menetelt furatok.

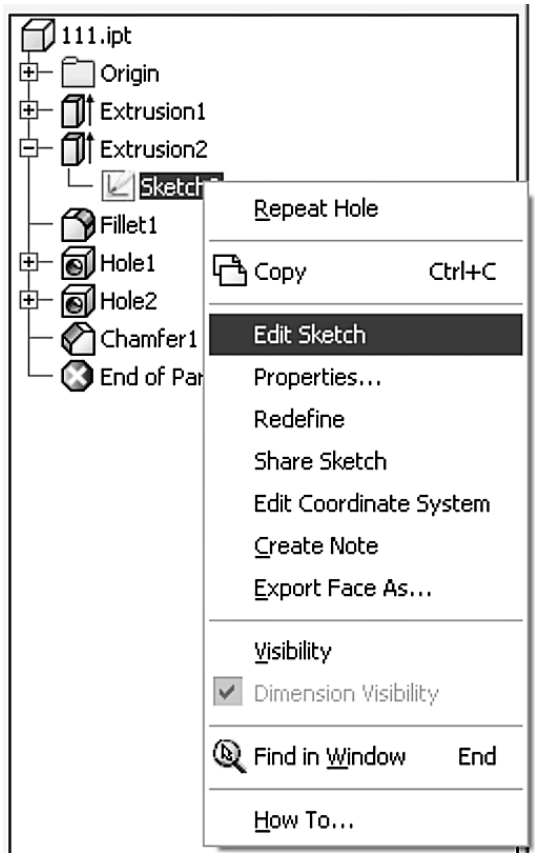
Full Depth – Átmenő furatok.

Taped – Fenekes furatok.

A párbeszédablak bal felső részében a furatok fő méretei állíthatók be, a furatot méretezetten ábrázoló képen. Az adatok bevitelére hossz- és szögértékeket írhatjuk be a megfelelő méretek egyszerű átírásával.

2.9.5. A Browser Bar (Áttekintőtár) használata

A **Browser Bar** fő szerepe, hogy az alkatrész készítése közben minden műveleti lépés **feljegyzésre** kerüljön. Az ide bekerülő feljegyzések egyik szerepe, hogy utólag is láthatjuk egy adott rajz elkészítésének felépítését. Ennél azonban sokkal fontosabb, hogy bármikor visszatérhetünk az egyes vázlatokhoz vagy a sajátosságok elhelyezéséhez, és módosíthatjuk ezeket. Nagyon sok esetben az egyes elemek kijelölése vagy azonosítása sokkal könnyebb az Áttekintőtárból, mint a sokszor igen bonyolult grafikai elemeket tartalmazó modellekből.



2.43. ábra
Vázlat szerkesztése a Browser Bár segítségével

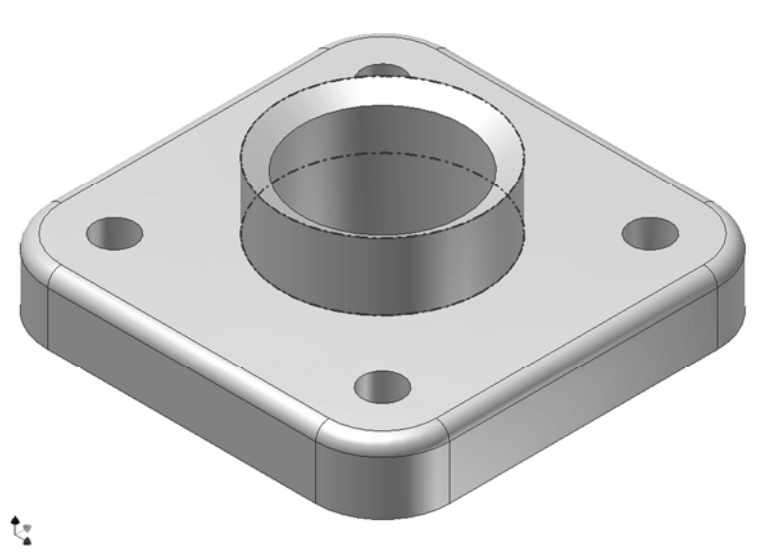
Ha az Áttekintőtárban ráállunk az egérkurzorral valamelyik sorra, azonnal megjelenik az alkatrészen a bejegyzéshez tartozó sajátosság. Egy sajátosság kiinduló vázlatsora a megrajzolt profil, és a hozzá tartozó méreteket, a Part sor pedig a művelettel létrehozott részt mutatja meg. Megfigyelhető a különböző elemek és műveletek kronológiai sorrendje is.

Ha módosítani akarjuk az alkatrészt, akkor a jobb egérgomb kattintásra megjelenő menüből adhatjuk ki az **Edit Sketch** (Vázlat szerkesztése) parancsot (**2.43. ábra**).

Probaképpen változtassuk meg az alaplapon levő második Extrusion2 méretét. Az eredetileg 30 mm magas kiemelés méretét változtassuk meg 5 milliméterre. A Browser Bar Extrusion2 sorára kattintva, a jobb egérgombbal, az **Edi**t Feature sort jelöljük ki. A megjelenő **Extrude: Extrusion2** párbeszédablakban, a

kiemelés méretét megváltoztatva, és az OK gombra kattintva végrehajtásra kerül a módosítás (**2.44. ábra**).

Megjegyzendő, hogy a Browser Bar-ban bármikor módosíthatók a már létező beírások, ha nem eredményeznek egyéb megoldhatatlan változásokat. Ezeket is elfogadhatjuk, ha a megjelenő hibautasításra az **Accept** gombot nyomjuk meg. Tudni kell, hogy ilyenkor a modelleken általában eltűnhetnek sajátosságok vagy kényszerek. Ezeket később feltétlenül javítanunk kell.



2.44. ábra
A módosított magasságú darab

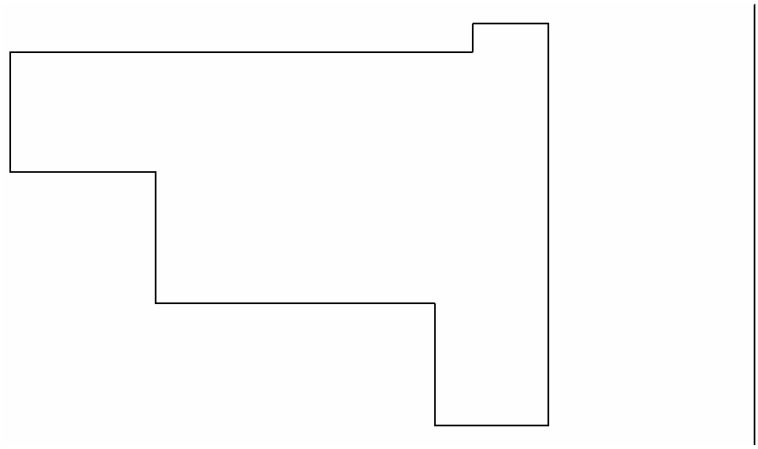
2.9.6. A Revolve (Megforgatás) sajátosság

A kihúzás alaksajátossághoz hasonlóan, a megforgatáshoz is vázlatot kell létrehoznunk, amelyet majd profiként alkalmazunk. Indítsunk egy új Metric Standard (mm).ipt-t.

A vázlat készítése előtt most nem kell a beállítással foglalkoznunk, hiszen az első példánál elvégzett beállítások továbbra is érvényesülnek. A vázlatot a Line parancs alkalmazásával kezdjük. Először a **2.45. ábrán** látható vonalkombinációt rajzoljuk meg, majd egy különálló vonalat is rajzoljunk, amelyet tengelyeként fogunk használni.

A vázlat végleges alakját parametrikus méretek hozzáadásával alakítjuk ki. Méretezésre alkalmazhatjuk az **Auto Dimension** (Automatikus méretezés) parancsot is, de mivel minden méretet módosítanunk kell, célszerűbb a **General Dimension** (Kézi méretezés) alkalmazása. Ezzel a paranccsal ugyanis, az

általunk meghatározott helyre kerülnek a méretek, míg az Auto Dimension által elhelyezett méreteknél viszonylag sokat kell számolni a megfelelő érték beírásához. Mindig javasoltabb a General Dimension utasítás használata, egyébként minden méret automatikus feltevését a még nem kívánt méretek letörlése is kell kövesse.



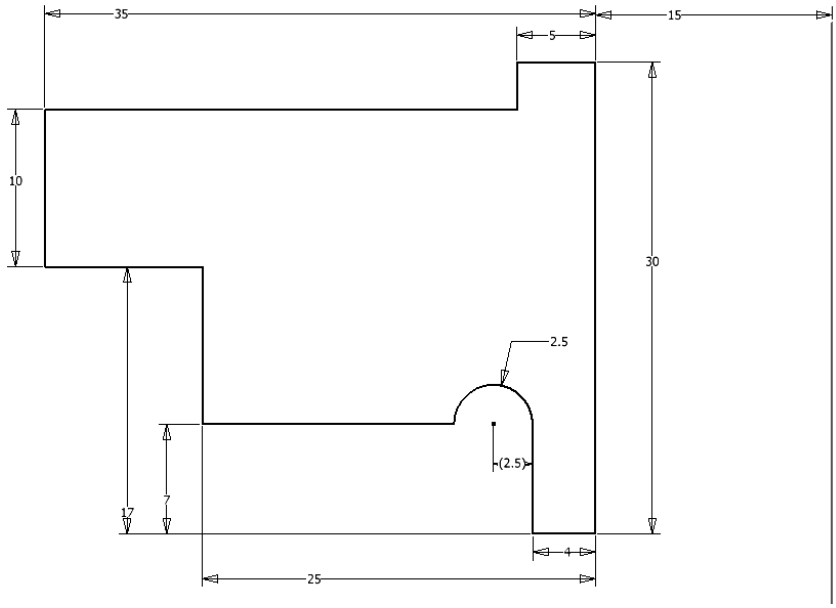
2.45. ábra
A kiindulási profil vázlata

A vázlat jobb alsó részén egy félkör alakú beszúrást hozunk létre, amelyhez egy kört kell rajzolnunk. A **Trim** (Metszés) parancs alkalmazásával a kör felét és a körön belüli vonalszakaszt kivágjuk, majd a General Dimension parancsot alkalmazva a **4.46. ábrán** látható méreteket tesszük fel a vázlatunkra.

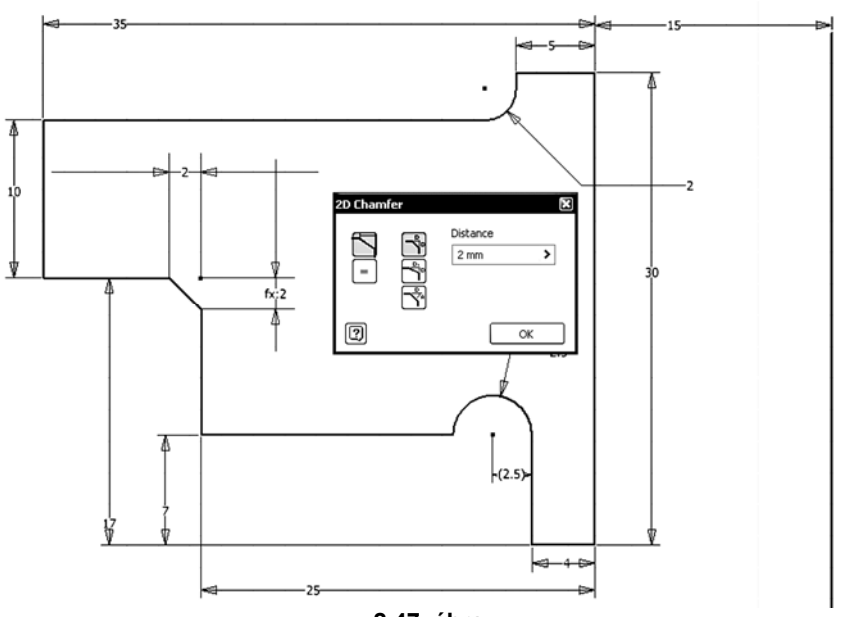
A vázlaton elhelyezünk még néhány sarkítást és lekerekítést is. Ezek feltevése a vázlatra a gyakorlatban nem igazán helyes, inkább a már megforgatott test éleit törjük és kerekítjük. A bemutató gyakorlat céljából azonban végezzük el a megfelelő műveleteket: készítsünk egy 2 mm-es lekerekítést és egy 2 mm-es sarkítást, amint az a **2.47. ábrán** látható.

A műveletek után, jobb egérkattintásra megjelenő menüből, válasszuk ki a Finish Sketch parancsot, vagy nyomjuk meg az Inventor Standard tárban található **Return** ikont**,** és máris megjelenik a Panel Bar-ban a Part Featuress (Sajátosságok).

Indítsuk el a Revolve (Megforgatás) utasítást, és a **2.48. ábrá**n látható módon jelöljük ki a megforgatási profilt. Megjegyzendő, hogy le kell mindig ellenőrizni a profil zárt jellegét, egyébként csak felületet fogunk eredménynek kapni. Ez a Revolve párbeszédablak bal alsó sarkában látható sárga kocka kijelölésével lehetséges. Nyílt profil esetén ez nem jelölődik ki automatikusan, ha pedig kézzel jelöljük ki, az ablak bal alsó sarkában, egy piros kereszt is megjelenik, **Examine Profile Problems** szöveg alatt.

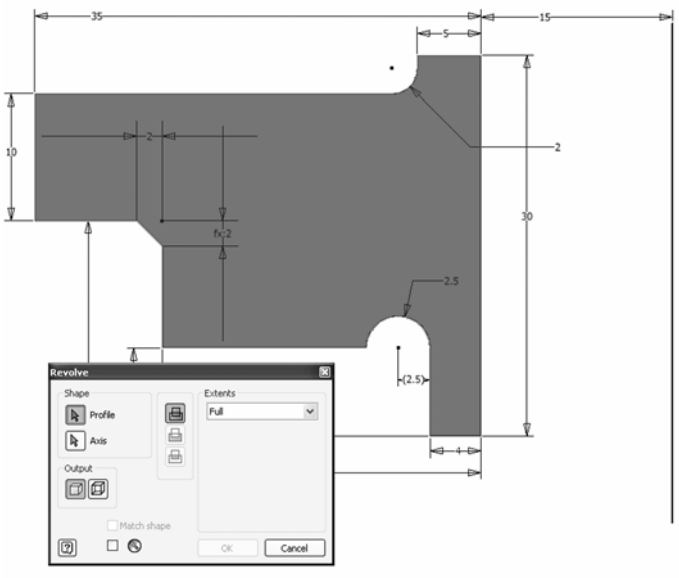


2.46. ábra
A méretezett vázlat

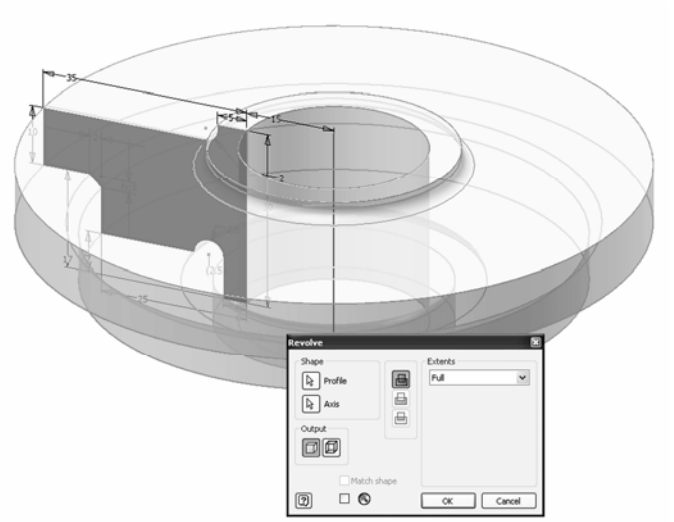


2.47. ábra
A vázlat, letörés és kerekítés után

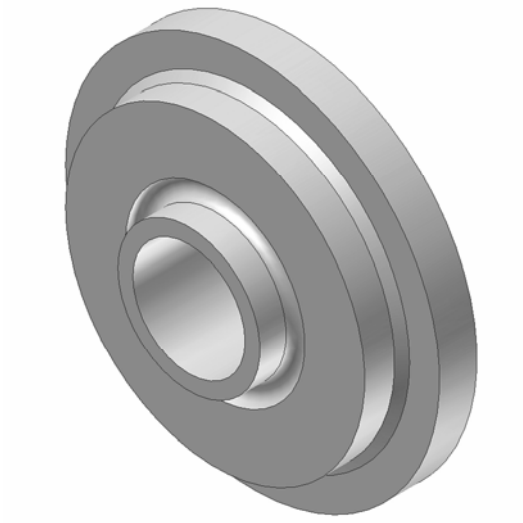
Ezután, csak ha megfelelően zárjuk a profilt (általában Coincident kényszerrel), akkor tudunk forgó testet létrehozni a profilból.



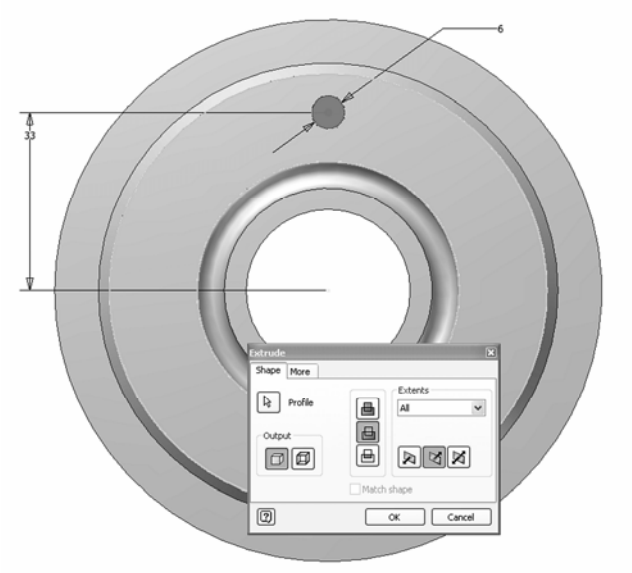
2.48. ábra
A kijelölt forgatási profil



2.49. ábra
A megforgatás előnézete



2.50. ábra
A megforgatással elkészült darab



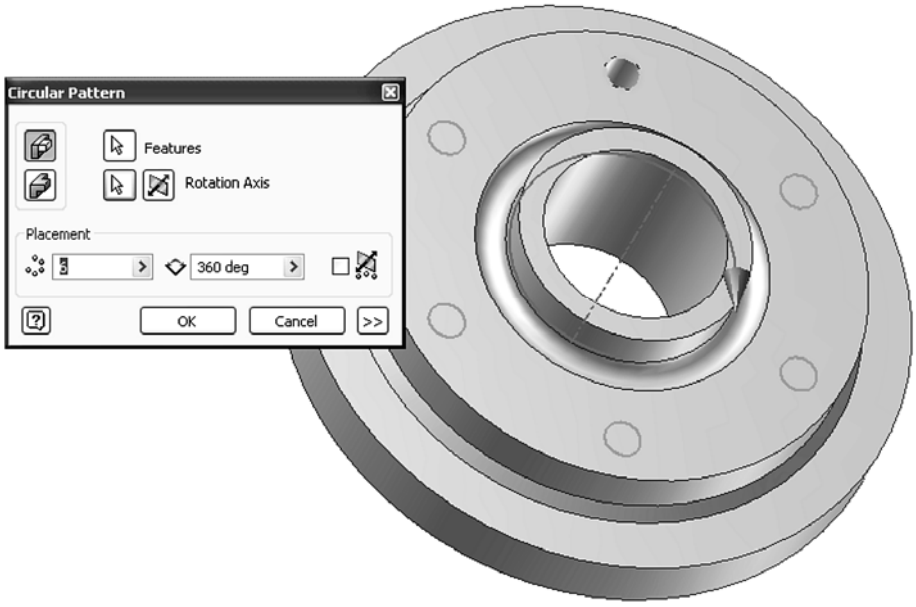
2.51. ábra
A furat készítése Extrude paranccsal

Nézzünk rá a **Look At** paranccsal (View menü vagy az Inventor standard Barból a Look At ikon) a darabunknak arra a felére, amelyiken hengeres rész 2 mm-es kerekítéssel illeszkedik a vájathoz (a továbbiakban a darab felső fele).

A ránézéssel (Look At) a szembefordított lapot jelöljük ki vázlatsíkként a New Sketch ikonnal, majd rajzoljunk egy tetszőleges méretű kört. A kör méreteit a **General Dimension** eszközzel pontosítjuk. Középpontja az alkatrész középpontjától 33 mm-re legyen, átmérője pedig 6 mm-re. A vázlatból az Extrude paranccsal hozzunk létre furatot, profilként kiválasztva a létrehozott kört. A párbeszédablakban kell megadnunk azt is, hogy a kihúzással létrehozott rész üregként jelenjen meg. Ehhez a **Cut** (Kivágás) ikonra kell kattintanunk**.** Az Extents részben az **All** (Mindent) beállítást szükséges használni (**2.51. ábra**).

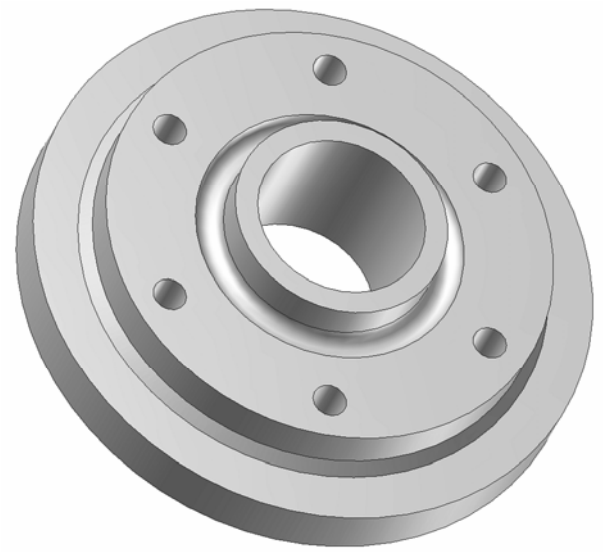
2.9.7. A Circular Pattern (Körkiosztás) sajátosság

Miután létrehoztunk a furatot, továbbiakat a Circular Pattern parancs alkalmazásával további hasonló furatokat helyezhetünk el az alkatrészen. A Circular Pattern parancs a Part Features eszköztárban található. A művelet végrehajtásához állítsunk be egy izzómetrikus nézetet, hogy az alkatrész felső fele teljesen látható legyen. Indítsuk a Circular Pattern parancsot a megfelelő ikonról.



2.52. ábra
A Circular Pattern előnézete

A megjelenő párbeszédablakban (Circular Pattern) először válasszuk ki a furatot (**2.52. ábra**), a **Feature** gomb megnyomása után. A furat kiválasztása lehetséges az Object Brower-ből is, Extrusion1 név alatt. Bonyolultabb darabok estében ez igen hatékony művelet. Ezután válasszuk ki a Rotation Axis gombot, megnyomva a forgástest bármelyik forgáselemét (kört vagy hengeres felületrészt).

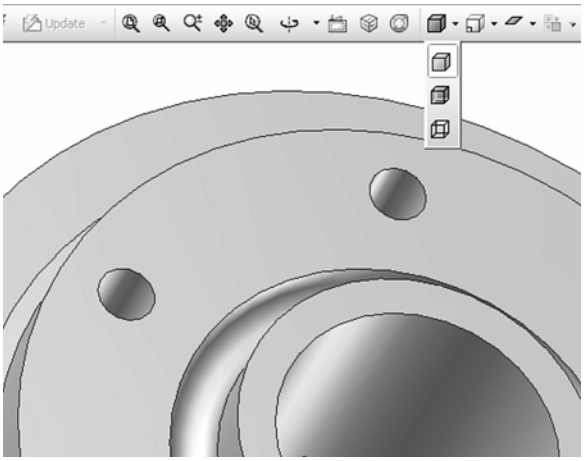


2.53. ábra
A darab végső állapotban

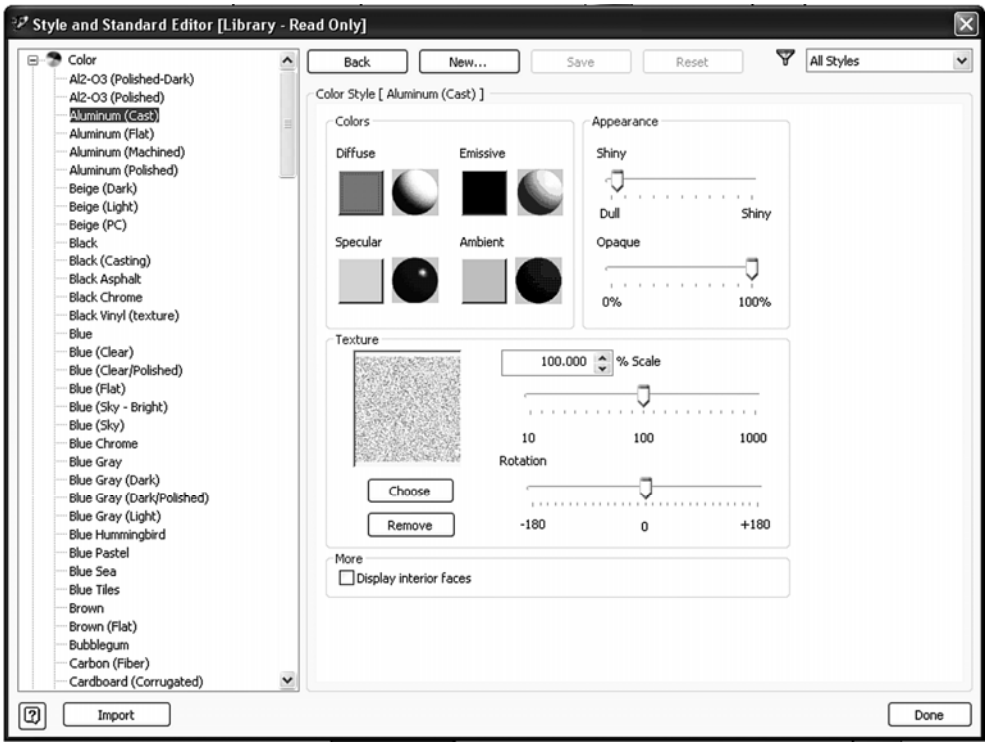
A párbeszédablakban a **Placement** rész bal oldalán a szükséges sokszorozási számot (esetünkben 6) válasszuk ki, és jelöljük ki, hogy milyen szög alatt akarjuk a sajátosságokat sokszorozni (itt a darab teljes felületén, tehát 360^{o} alatt). A **2.52. ábrán** megfigyelhető, hogy előnézetben már láthatók a megsokszorozott sajátosságok, majd az OK gombot megnyomva a darabunk elnyeri a végső, hat furattal ellátott alakját (**2.53. ábra**).

2.9.8. Az alkatrészek színe

Az Inventorban szabályozhatjuk a darabok grafikus megjelenést. A Központi Eszköztár legördülő menü ikonjainak segítségével (**2.54. ábra**), **Shaded Display** (Árnyalt), **Hidden Edge Display** (Takartvonalas) és **Wireframe Display** (Drótvázas) megjelenítést választhatjuk. Induláskor, alapesetben az Shaded Display megjelenés érvényesül, de szerkesztés közben sok esetben át kell kapcsolni a Drótváz megjelenítési (Wireframe Display) módra. Ezeken a megjelenési módokon kívül azonban változtathatjuk az alkatrészek színét is. A legördíthető lista bőséges választékot kínál a különböző anyagokkal jelzett színekből.



2.54. ábra
A megjelenítési módok legördülő menüje



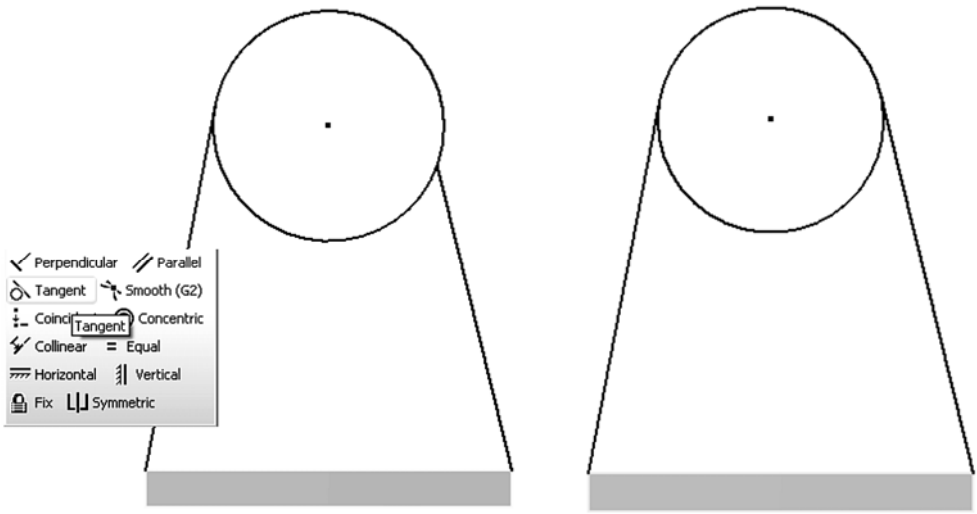
2.55. ábra
A darabok színét meghatározó párbeszédablak

A színek megjelenése is szabályozható. A **Format** menü **Style and Standard Editor...** menüsorára kattintva, és a megjelenő ablak bármelyik színét kiválasztva, a **2.55. ábrán** látható lap nyílik meg.

2.9.9. A Rib (Borda) sajátosság

Az újabb sajátosság megismeréséhez elkészítünk egy újabb alkatrészt. Először is egy új Standard (mm).ipt sablon-állományt indítunk.

Az első vázlatot a **Two point rectangle** nevű paranccsal hozzuk létre, majd a General Dimension eszközzel beállítjuk a 60 mm és a 80 mm méreteket. Ebből a vázlatból, az Extrude alkalmazásával, hozzuk létre az alkatrész alapját. A kihúzás mérete 8 mm. Az újabb vázlathoz a vázlatsíkot ennek a téglatestnek a rövidebb oldalánál lévő függőleges felületre való kattintással jelöljük ki. A második vázlatban először egy kört kell rajzolnunk, majd egy Line paranccsal létrehozzuk a további részeket. A vonal rajzolását a kör bal oldalán kezdjük, ahol megközelítöleg érintőlegesen jelöljük ki az első pontot. A vonalrajzolást, a kör jobb oldalán, hozzávetőlegesen érintőként fejezzük be (**2.56. ábra** bal oldali képe).



2.56. ábra
A Tangent kényszer alkalmazása

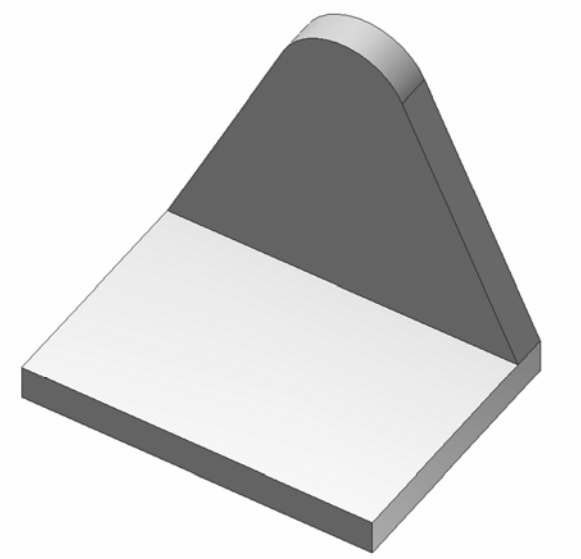
A két vonal és a kör csatlakozását a **Tangent** kényszer hozzáadásával pontosítjuk. Az ikonra kattintás után, kijelölve a jobb oldali ferde vonalat és a kört (**2.56. ábra** jobb oldali képe). Elsőként a körre kattintsunk, majd amikor a második kijelölést kéri a program, akkor kell a vonalat kiválasztani. A pontos érintés a

feltétele annak, hogy a **Trim** (Vágás) parancs alkalmazásával a kör fölösleges részét el tudjuk távolítani.

Most húzzunk egy vonalat a már kész testrészlet felső vonalának közepéről (a kurzort megközelítőleg a közepére téve, a középpont zölden fog megjelenni), a kör középpontjáig. Adjunk hozzá – a meghúzott vonal és az alsó alakzat felső vonalához – egy merőlegességi (Perpendicular) kényszert. Ez a szerkesztési mód biztosítani fogja, hogy a kör középpontja mindig az alsó alakzat középvonalán helyezkedjen el. Adjunk, a General Dimensiont használva, 30 mm-es méretet a kör átmérőjének és rögzítsük a középpontját az alsó alakzattól 50 mm-re, és végül, Trimmel vágjuk ki a kör belső részét.

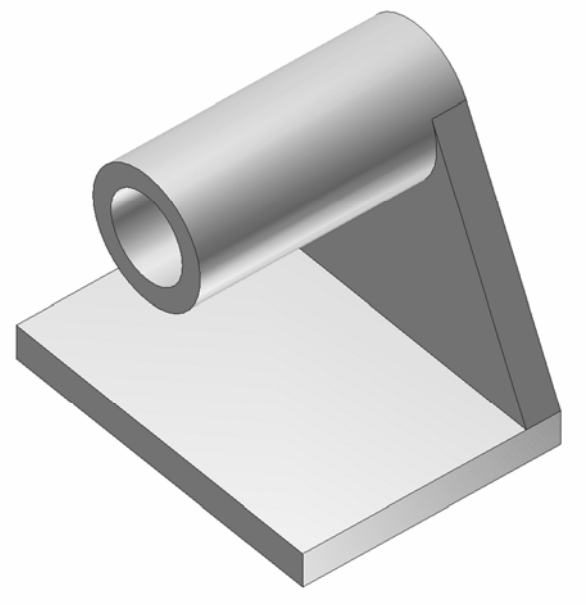
A kapott profilt emeljük ki Extrude-al 8 mm-re. Vigyázzunk a kiemelés irányára, ha az előnézettben nem megfelelő, akkor változtassuk meg az irányt a párbeszédablak jobb alsó felében található sík-nyíl (normálvektor) ikonnal! A kapott alakzat izommetrikus nézetben, a **2.57. ábrán** látható.

A modellezés következő lépésében az utoljára létrehozott alakzat belső felére készítsünk egy újabb vázlatot, és az első kör közepét, illetve szélét használva készítsünk egy újabb, az elsővel megegyező átmérőjű kört.

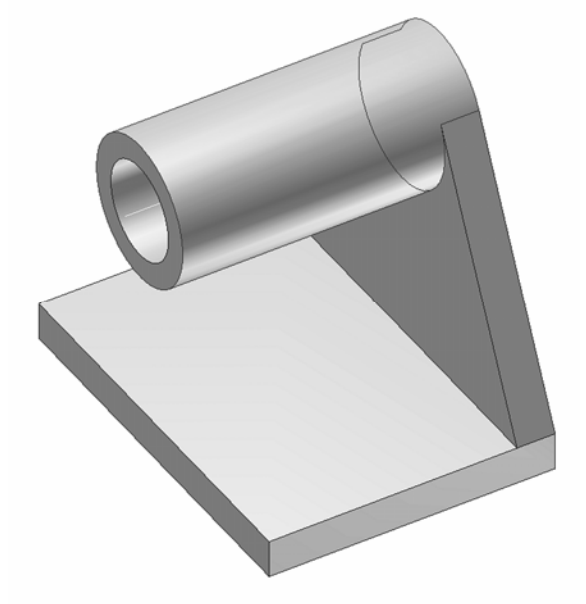


2.57. ábra
A darabunk a második kiemelés után

A létrehozott profilt emeljük ki 55 mm-el, készítsünk a külső részére egy újabb kört, középpontjának felhasználva a meglévő körív középpontját. Emeljük ki ezt a profilt is, ezúttal 55 mm-re. A hengeres részre pedig készítsünk, vázlat készítése nélkül, a Concentric opciót használva, a Hole sajátosságból, egy sima átmenőfuratot 20 mm-es átmérővel. A végeredmény a **2.58. ábrán** látható.



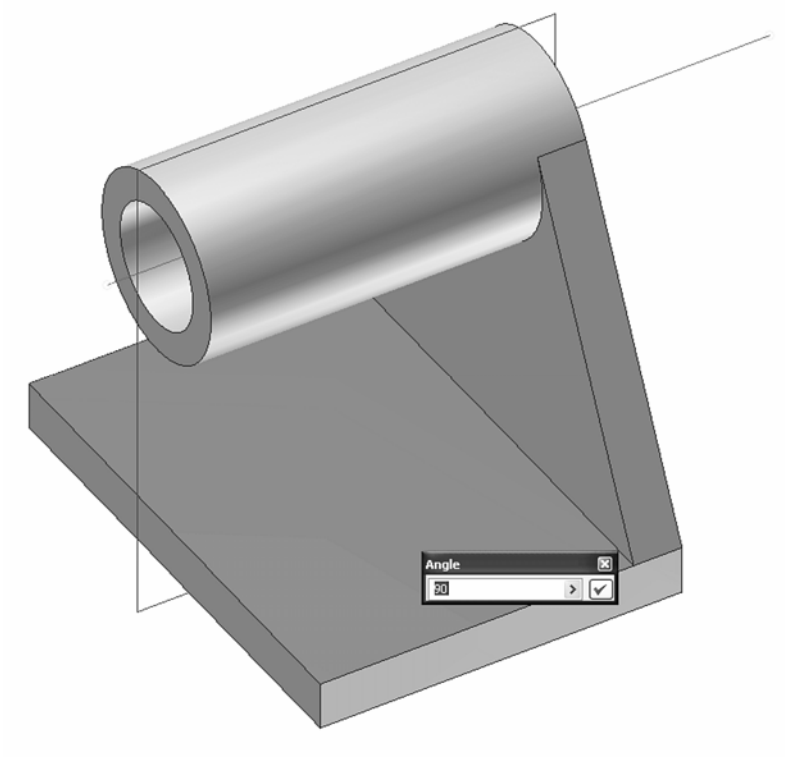
2.58. ábra
A testmodell három Extrude és egy Hole parancs után



2.59. ábra
Munkatengely létrehozása

A következőkben, a hengeres rész forgástengelyére egy munkatengelyt készítünk a Work Axis parancs kiadásával, a Part Featuress tárból. A kurzort a hengeres rész felé közelítve, megjelenik előnézetben a munkatengely, amint az a **2.59. ábrán** is látható. Rákattintva létrehozzuk a kívánt munkatengelyt, amely megjelenik az Browser Bar-ben is.

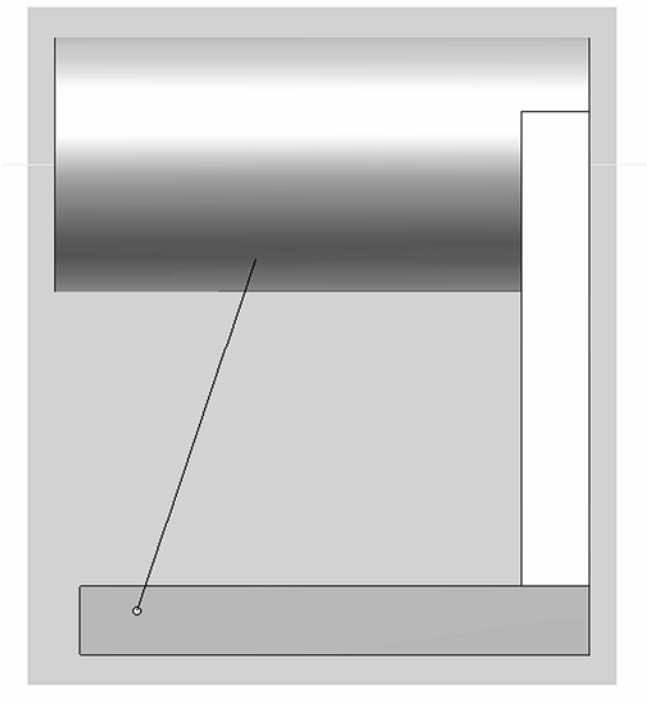
Most egy munkasíkot készítünk, amely átmegy ezen a tengelyen és merőleges a darab alsó, sík részére. Ehhez elindítunk egy Work Plane (Munka Sík) parancsot a Part Features-ból, kijelöljük az előzőleg létrehozott munkatengelyt, és kijelöljük az alatta lévő darab lapjának a felső részét. A **2.60. ábrán** látható kép jelenik meg. Itt, ha az **Angle** ablakba 0^o írunk, a munkatengelyen átmenő és az alsó síkkal párhuzamos munkasík jön létre. Ha a felajánlott 90^o hagyjuk meg (megnyomva a kipipált gombot), akkor egy olyan munkasík jön létre, amely átmegy a munkatengelyen és merőleges az alatta levő síkfelületre. Ez a munkasík lesz a következő vázlat síkja, a borda profilja.



2.60. ábra
Munkasík létrehozása

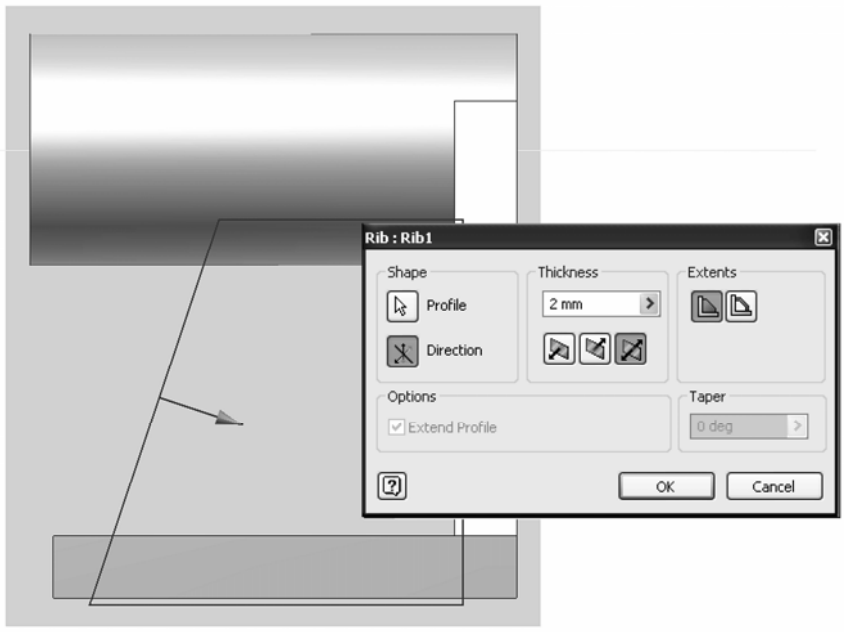
Jelöljük ki a létrehozott munkasíkot és nyomjuk meg az Inventor Standard menüből a Sketch ikont. Nézzünk rá a **Look At**-tel erre a vázlatra és húzzunk egy vonalat, amint az a **2.61. ábrán** látható, utána pedig, a jobb-egérgomb kattintással és Finish Sketch-el zárjuk a vázlatot. Ezután indíthatjuk a Rib parancsot a Part Features panelből.

A Rib parancs indításakor a **2.62. ábrán** látható párbeszédablak jelenik meg, amelyben a **Profil** gombot megnyomva kijelölhetjük a borda profilját. A mi esetünkben, mivel a legegyszerűbb profilról, egy vonalról van szó, elégséges a kurzort a vonaltól jobbra vagy balra elhelyezni, a **Thickness** ablakba beírni a kívánt bordavastagságot, kiválasztani az **Extents** részből a két lehetséges bordaforma egyikét és rákattintani a vázlatra.

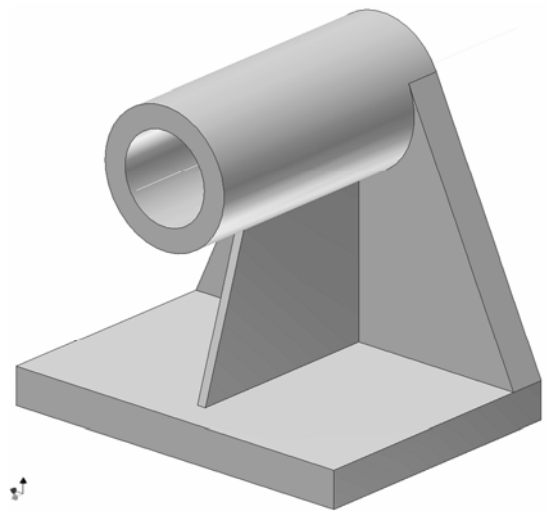


2.61. ábra
A borda vázlatának létrehozása

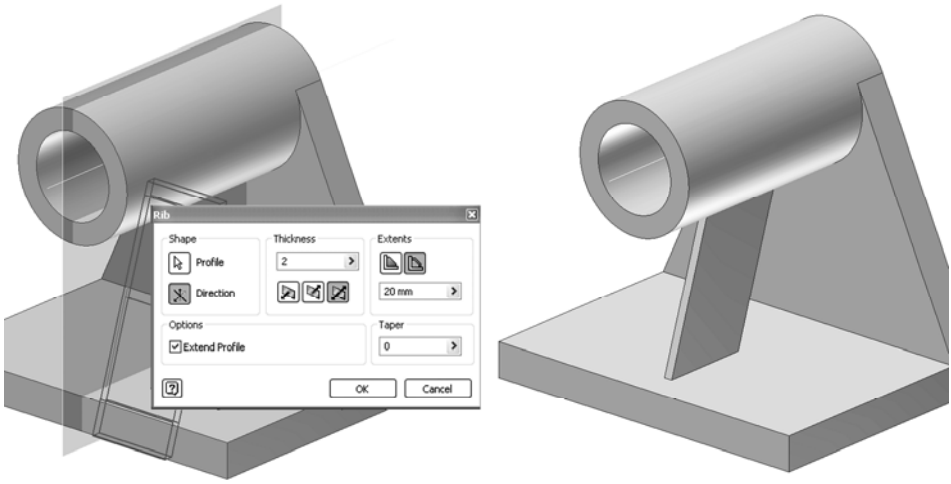
Az eredmény a **2.63. ábrán** látható. Ezen már elrejtettük az előzőkben látható munkasíkot, úgy, hogy az Browser Bar-ban a jobb egérgombbal rákattintottunk és kijelöltük – a már kipipált – **Vizibility** sort. A Visibility megnyomásával elrejthetünk vagy megjeleníthetünk bármilyen létrehozott elemet a testmodellünkön, vagy a szerelési modelleken az egyes alkatrészeket.



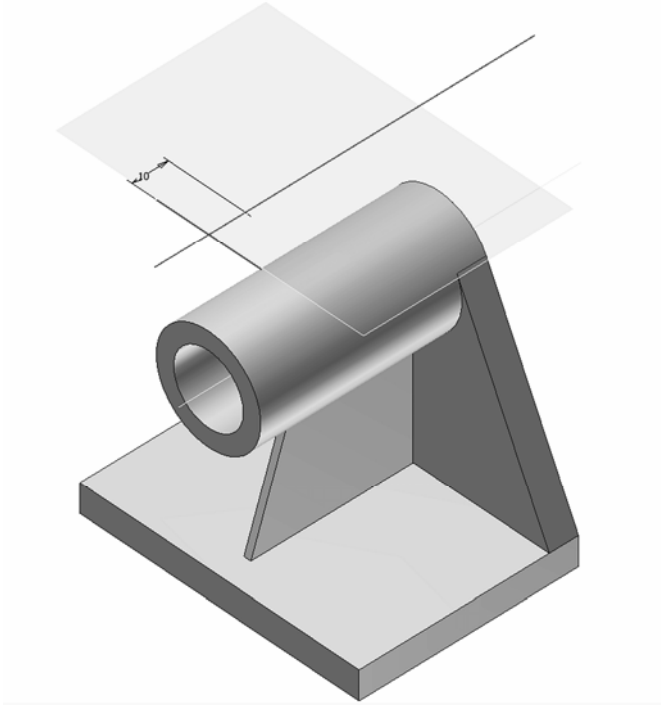
2.62. ábra
A megjelenő Rib párbeszédablak



2.63. ábra
A borda sajátosság első változata



2.64. ábra
A Rib második változata



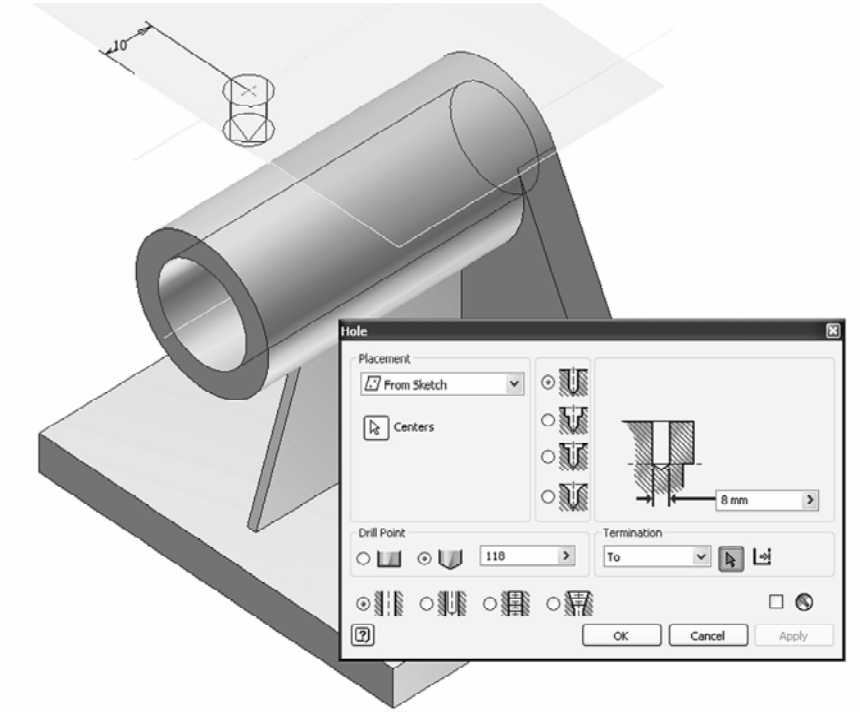
2.65. ábra
A furatközpont helyzete

Ha az **Extens** másik, jobboldali változatát akarjuk alkalmazni, akkor még egy vastagsági méret megadására lesz szükségünk, amint az a **2.64. ábrán** is látható.

Természetesen a borda sajátosság is, a többi sajátossághoz hasonlóan, bármikor szerkeszthető, módosítható, elsősorban és legkönnyebben az Browser Barban jobb egérklikkel és Edit Feature parancs kiadásával.

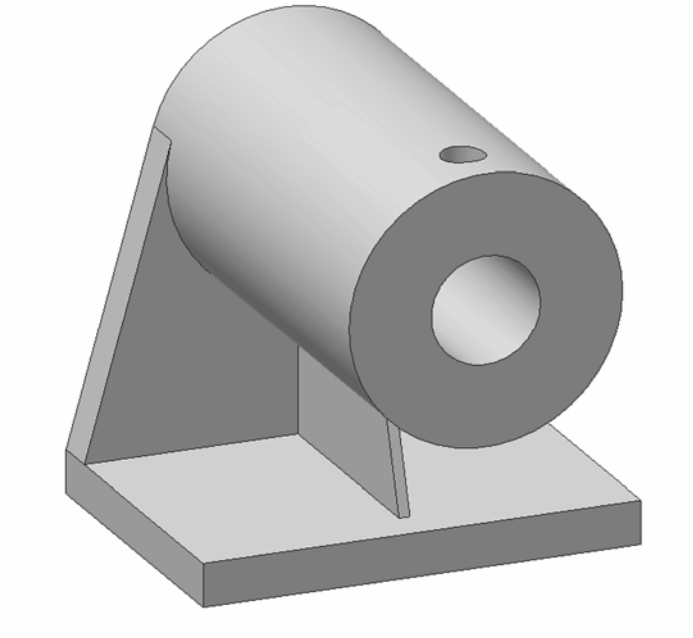
Készítsünk most egy furatot, amelynek forgástengelye merőleges az előzőkben létrehozott munkatengellyel, és a már létező furat faláig tart kívülről befele. Ehhez először is vegyünk fel egy munkasíkot, amely a darab alsó sík felületével párhuzamos és nem metszi az alkatrészt. Kattintsunk a Part Features-ből a Workplane ikonra, jelöljük ki a darab alsó hasáb részének a tetejét, húzzuk felfele az egérkurzort, és a megjelenő **Offset** ablakba írjunk 90 mm-t.

A létrejött munkasíkon, amelyet az Browser Bar-ben Work Plane2 név alatt találunk meg, indítsunk egy új vázlatot (New Sketch). A Project Geometry parancsot indítva, rákattintva a meglévő munkatengelyre, vetítsük azt le az új vázlatra. Vetítsük le hasonló módon a hengeres rész homlokfelületét (amely egy vonal lesz), és a Point, Center Point paranccsal tegyünk egy furatközpontot a munkatengely vetületére. Méretezzük ezt a homlokfelület vetületétől 10 mm-re, ahogy az a **2.65. ábrán** látható.



2.66. ábra
A belső forgásfelületig futó furat párbeszédablaka

Megnyomva a Return ikont, fejezzük be a vázlatot és indítsunk egy Hole parancsot. A furat átmérőjét állítsuk be 8 mm-re, a **Termination** ablakban jelöljük ki a **To** opciót és jelöljük ki a már létező hengeres rész furatának a falát, ami a **2.66. ábrán** látható. Az eredmény egy olyan furat lesz, amely csak a hengeres rész felső részét fúrja át. A Hole sajátosság ilyen formájú is marad, ha a hengeres rész átmérőjét 50 mm-re, a hosszát 80 mm-re változtatjuk (**2.67. ábra**).



2.67. ábra
A furat helyzete és formája változatlan marad

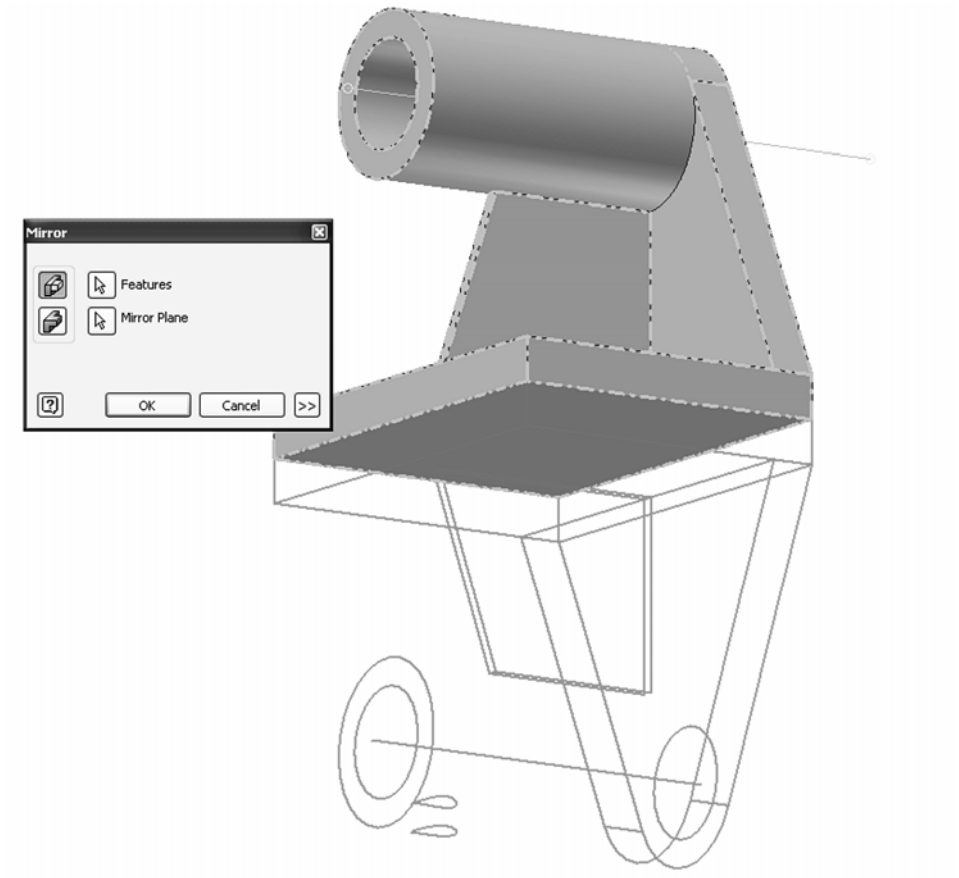
2.9.10. Mirror parancs – Sajátosságok tükrözése

A Part Features panel Mirror parancsát használva lehetőségünk van a sajátosságok tükrözésére, egyenként vagy csoportosan. A tükrözés síkjaként lehetőség van egy munkasíkot vagy a darabunk egyik oldalát tükrözési síknak használni.

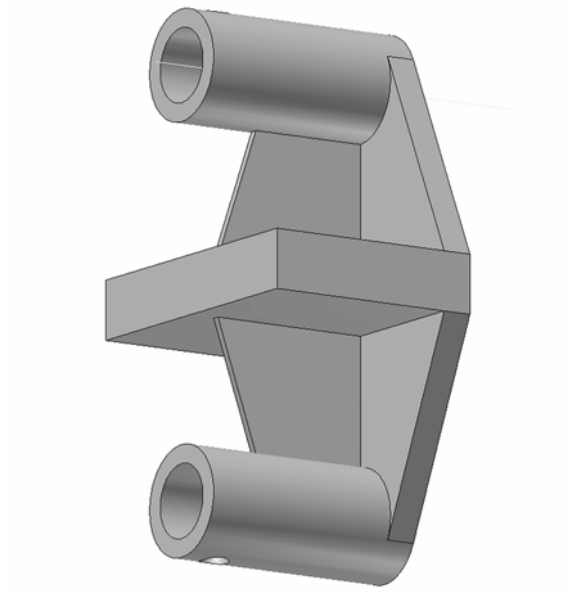
Tükrözzük az elkészült darabot a talp alsó síkja alapján. Futtassuk a Mirror parancsot és jelöljük ki a darab összes sajátosságát a testmodellen. Ha a darabunk a **2.68. ábrán** látható nézetben áll, akkor az utolsó furat kijelölését csak a darab elforgatása után lehetne elvégezni. Ekkor azonban könnyebb az Browser Bar használata. Itt is kijelölhető a furat, az Áttekintőtárban megtalálható Hole2 sajátosság kijelölése egyszerűbb lesz: nem igényel egy újabb darabnézet változtatást. Ugyanakkor az Áttekintőtárban könnyebben követhetők a tükrözésre már kijelölt

sajátosságok, amint az a **2.68. ábrán** is látható. A sajátosságok kijelölése után, meg kell jelölnünk, a **Mirror Plane** ikonra kattintva, a tükrözés síkját. Kijelölhetjük immár a darabunk legalsó sík felületét és meg is jelenik előnézetben a tükrözött darab, amint az ábrán is látható.

Tudnunk kell, hogy ha lemaradnak véletlenül sajátosságok, vagy később létrehozott sajátosságokat akarunk tükrözni, akkor jó ha az utólagos, könnyebb hozzáférés érdekében a használt szimmetria felületekre, oldalakra és síkokra, munkasíkokat építünk. Ezeknek újabbi kijelölése, tükrözésre való használata megmarad a modellezések későbbi részében is. Például, ha az eddigi lépések után, tovább akarnánk használni a testmodell alsó, sík felületét, ez nem lenne lehetséges, mivel gyakorlatilag az eltűnik az első tükrözés után.



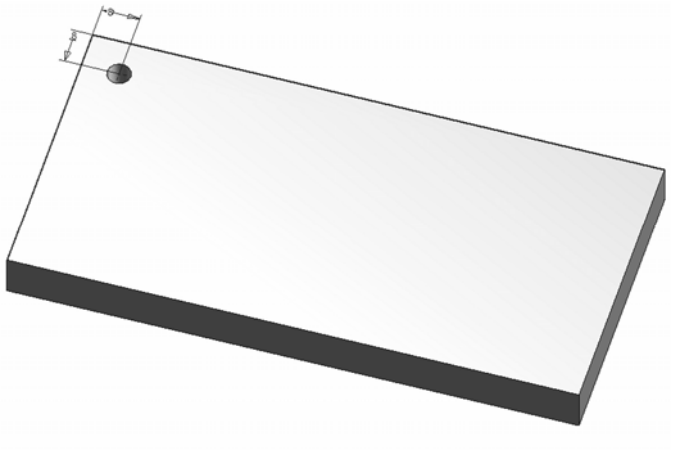
2.68. ábra
A tükrözés előnézete



2.69. ábra
Tükrözött alkatrész

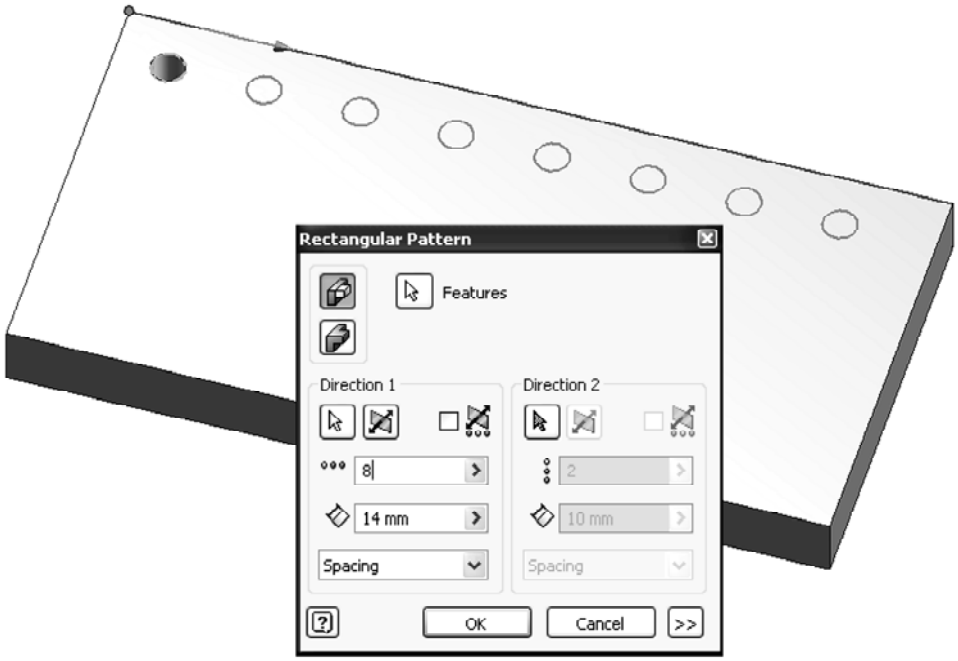
2.9.11. A Rectangular Pattern (Négyszög kiosztás) sajátosság

Készítsünk egy 80x80x10 mm hasábot, a bal felső sarkába pedig helyezzünk egy sima, 5 mm átmérőjű, átmenő furatot, a bal felső saroktól 8–8 mm-re (**2.70. ábra**).



2.70. ábra
A kiindulási furat

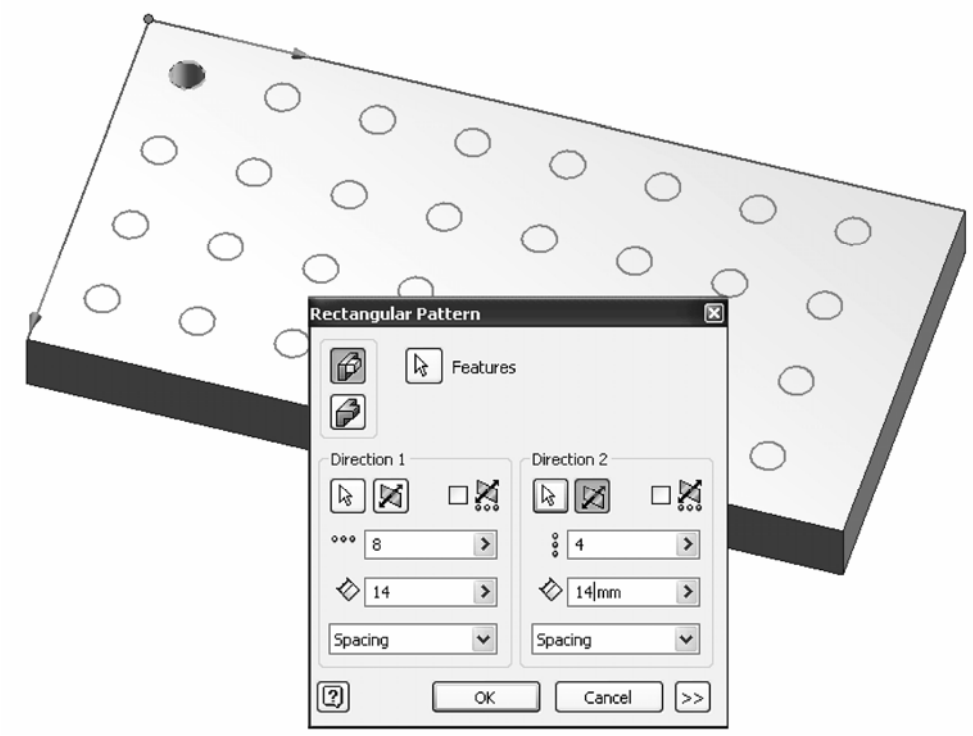
Indítsuk a Part Features panelből a **Rectangular Pattern** (Négyszög kiosztás) parancsot. A **2.71. ábrán** látható párbeszédablak jelenik meg.



2.71. ábra
A kiosztás egyik irányának meghatározása

A Features ikont megnyomva a sokszorozásra (kiosztásra) jelöljük ki a sajátosságokat. A **Direction 1** és **Direction 2** rész alatt két kiosztási iránynak a kijelölésére van lehetőség. Megjegyzendő, hogy csak az egyik megadása is elégséges, ha csak egy irányba akarjuk sokszorozni a sajátosságokat. Mindkét részben a nyíl ikonra kattintva kijelölhetünk egy vonalat vagy élet a modellünkön, amely meghatározza az igényelt kiosztás irányát. A nyíl ikon mellett lévő sík és egy duplanyíl ikon lehetővé teszi az irányítás megváltoztatását is. E két ikon alatt lévő részbe kell beírni a sokszorozás számát, illetve ha a legördülő menüben a **Spacing** opciót választjuk, akkor a sajátosságok közötti távolságot kell megadni (**2.72. ábra**). A kiosztás, a 2009 Inventor-ban, lehetséges **Distance** és **Curve Length** opciókkal is.

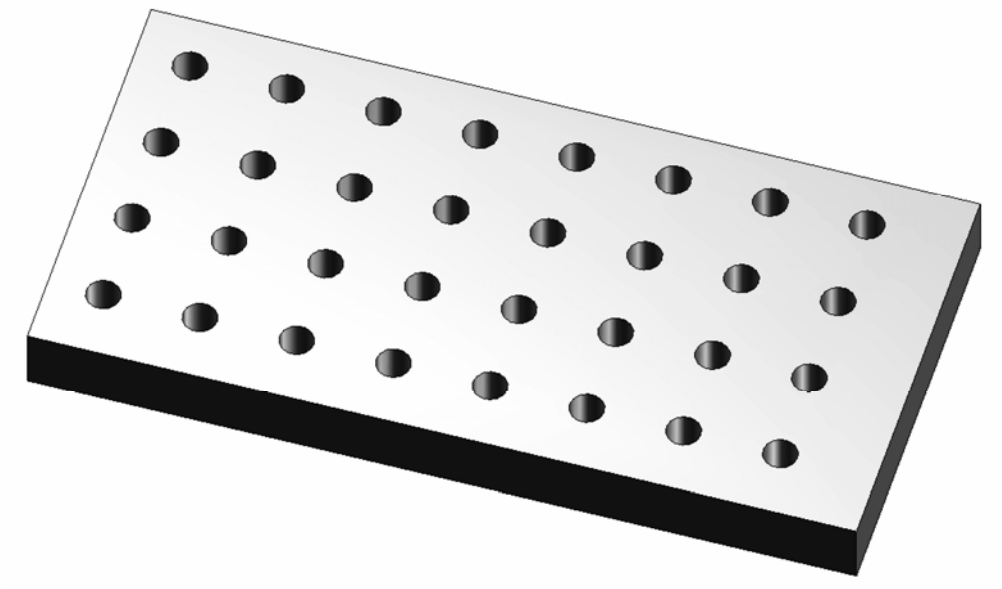
Ha kijelöljük a második kiosztási irányt is, akkor a **2.72. ábrán** látható értékeket írjuk be: első irány – 8 elem, 14 mm távolságra, az irányunk a testmodell bal felső éle lesz, az irányítást pedig úgy határozzuk meg, hogy az előnézetben látható furatok a testmodellre essenek. A második irányú kiosztás a test felső éle, irányítása ugyanúgy az összes furat testre való kerülését kell biztosítsa, 4 elem kiosztással, 14 mm távolságban.



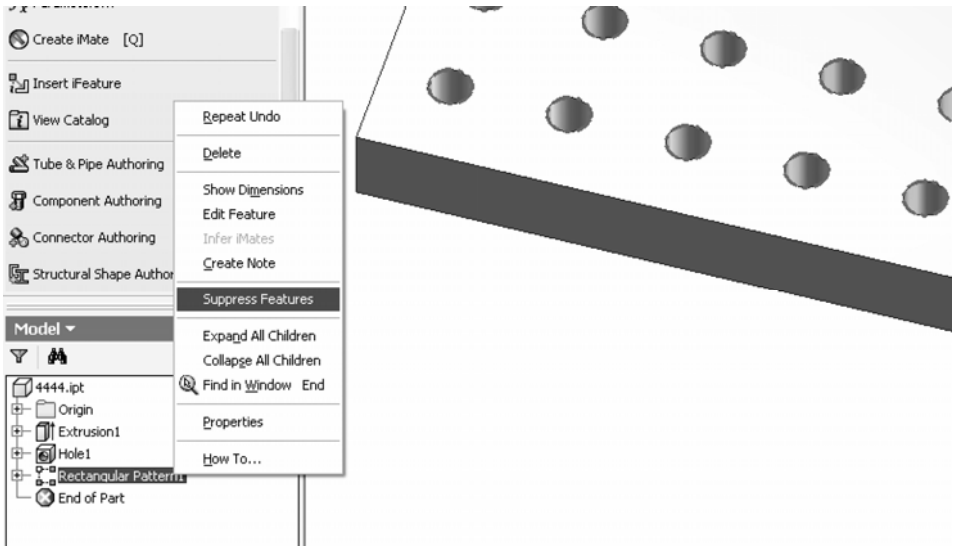
2.72. ábra
A kétirányú kiosztás előnézete

A parancs befejezése az OK gombbal, a **2.73. ábrán** látható testmodellt eredményezi.

Mint bármelyik sajátosság, ez is szerkeszthető: az Browser Bar-ban a Rectangular Pattern1 címre kattintva a jobb egérgombbal, a **2.74. ábrán** látható menü gördül le. Itt, ha az **Edit Feature**-ra kattintunk, visszajön a **2.72. ábrán** látható párbeszédablak és lehetőségünk van módosítani az előzőkben bevitt kiosztási adatokat. Ha a legördülő menüből a **Suppres Features** parancsra kattintunk, akkor az előbbi sajátosságunk eltűnik a modellről, ellenben benne marad az Áttekintőtárban, **áthúzva**.

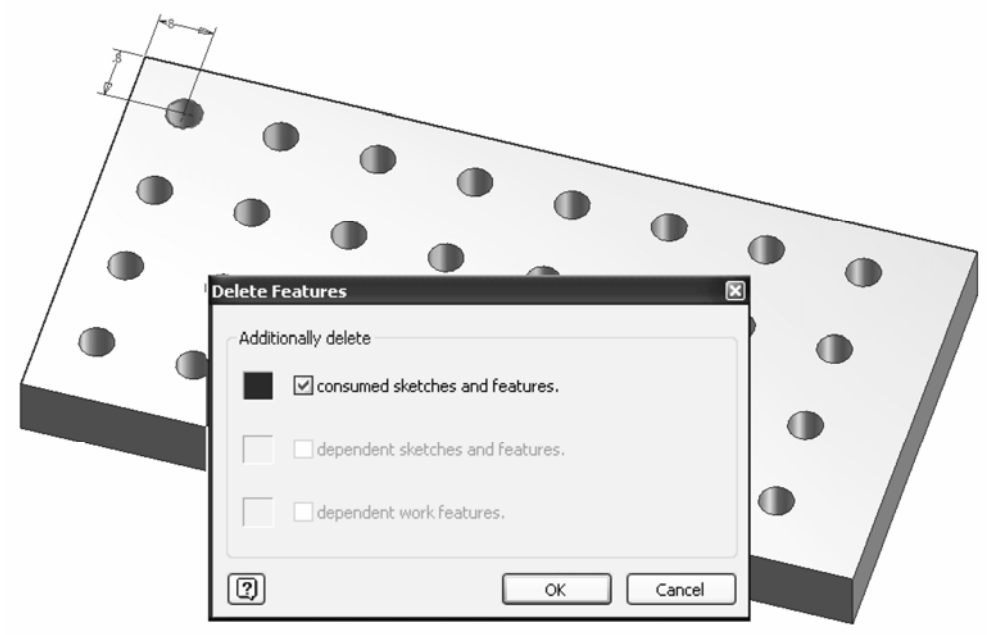


2.73. ábra
Furatok Rectangular Pattern parancs után



2.74. ábra
Az egyes sajátosságok legördülő menüje

Itt kell megjegyeznünk, hogy a modellezés során bármely sajátosságunk törölhető az előzőkben ismertetett legördülő menüből a **Delete** parancsot választva. A törlés során arra kell vigyáznunk, hogy a sajátosság után létrehozott és az azt felhasználó egyéb sajátosságok eltűnhetnek, ha a megfelelő opciókat nem tesszük meg a Delete parancs kiadása után (**2.75. ábra**). A megjelenő **Delete Features** párbeszédablakban lehetőség van az **Additionally delete** opcióknál az elhasznált vázlatokat és sajátosságokat is letörölni (**consumed sketches and features**). Vigyázat, ez automatikusan ki van jelölve, ha nem akarjuk ezeket törölni, akkor a kijelölést szüntessük meg! Lehetőségünk van a továbbiakban megtartani a letörlésre kerülő sajátossághoz tartozó vázlatokat és sajátosságokat (**dependent sketches and features**), és a munka-síkokat, tengelyeket és pontokat (**dependent work features**) is megtartani, ha ezeknek is az implicit kijelölését megszüntetjük.

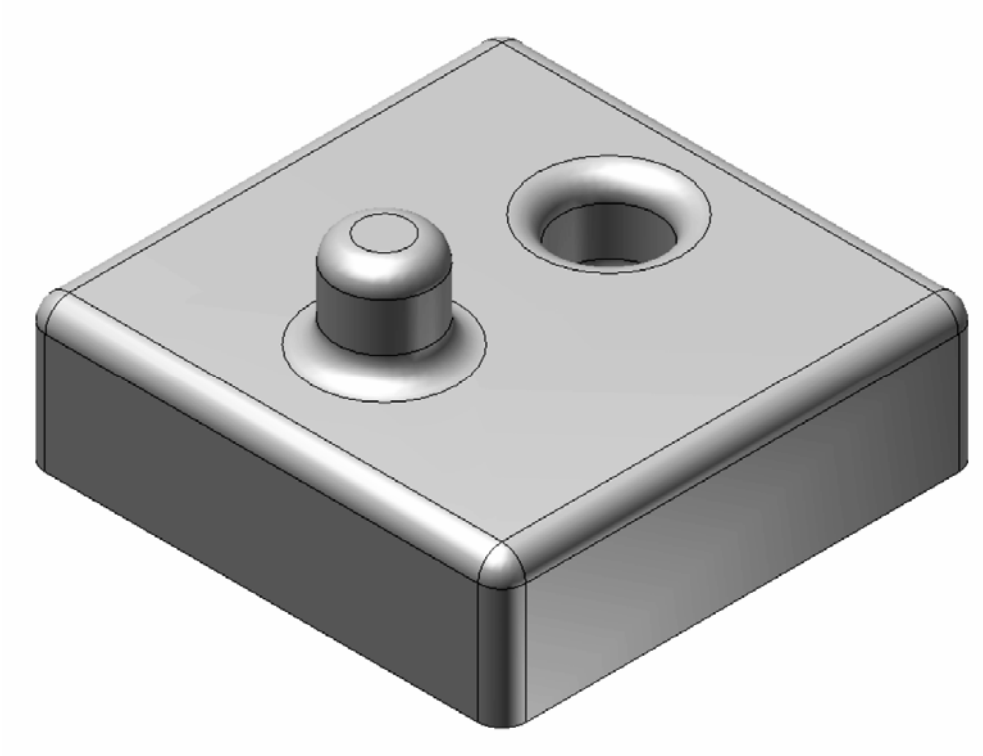


2.75. ábra
Sajátosság törlésekor megjelenő párbeszédablak

Az alapvető különbség a Delete és Supress Features között: a Delete véglegesen letörli a sajátosságot (és az egyéb hozzá tartozó elemeket), a Supress Features pedig „kikapcsolja” a sajátosságot, később lehetséges a „visszakapcsolása”.

2.9.12. A Shell (Héj) sajátosság

Ez a fajta sajátosság lehetővé teszi, hogy tömör testmodellekből vékonyfalú, üreges alkatrészeket hozzunk létre. Készítsünk egy 100x100 mm oldalhosszúságú téglalap vázlatot és emeljük ki 20 mm-re Extrude-al. A hasábra készítsünk egy 20 mm átmérőjű kört, és vonjuk ki a hasábból 20 mm mélyen. Egy újabb vázlatra a hasáb felső oldalára készítsünk egy másik kört, és ezt húzzuk ki ugyancsak 20 mm-re. Kerekítsünk le test minden élet 5 mm-es sugárral, az alsó élek kivételével. A modellünknek úgy kell kinéznie, mint a **2.76. ábrán** látható.

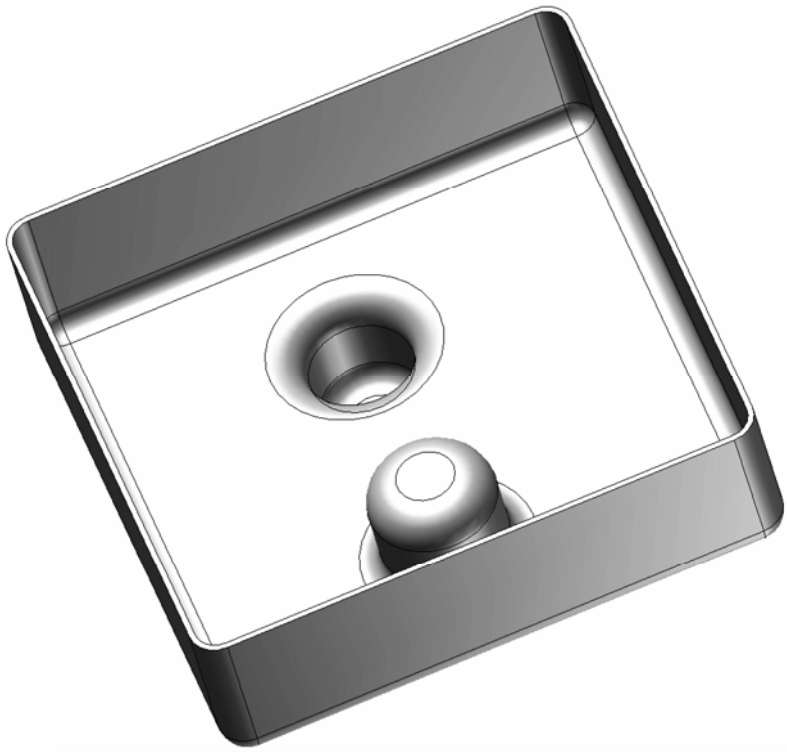


2.76. ábra
A hasáb egy kiemeléssel és egy kivonással

Az **Orbit** parancs segítségével (View menű, vagy Inventor Standard Panel) forgassuk egy olyan nézetbe, ahól teljesen látszik a test alsó sík felülete. Indítsuk a Shell parancsot Part Featuress panelből. A **2.77. ábrán** látható párbeszédablak jelenik meg.

2.77. ábra
A Shell párbeszédablak

A megjelenő párbeszédablak jobb felső részén található a **Remove Faces** (Lapok eltávolítása) ikon, amely arra szolgál, hogy kijelöljük a test azon lapjait, amelyeket ki akarunk vonni a parancs hatása alól. Ha nem jelölnénk ki lapokat, akkor üreges, de teljesen zárt test jönne létre. A most már felül lévő síklapot kell kijelölni. A **Thickness** (Vastagság), a létrejövő fal vastagsága, legyen 1 mm. A párbeszédablak bal oldalán lévő három ikon a héj létrehozási módjának kiválasztására szolgál. Sorrendben az **Inside** (Belül), **Outside** (Kívül), **Both** (Kétirányba) módok választhatók annak eldöntésére, hogy a megadott vastagság hogyan jelenjen meg a már létező felületekhez viszonyítva. A példában maradhat az alapértelmezés szerinti Inside beállítás. Az üreges test az OK nyomógombra kattintáskor jön létre és a **2.78. ábrán** látható.



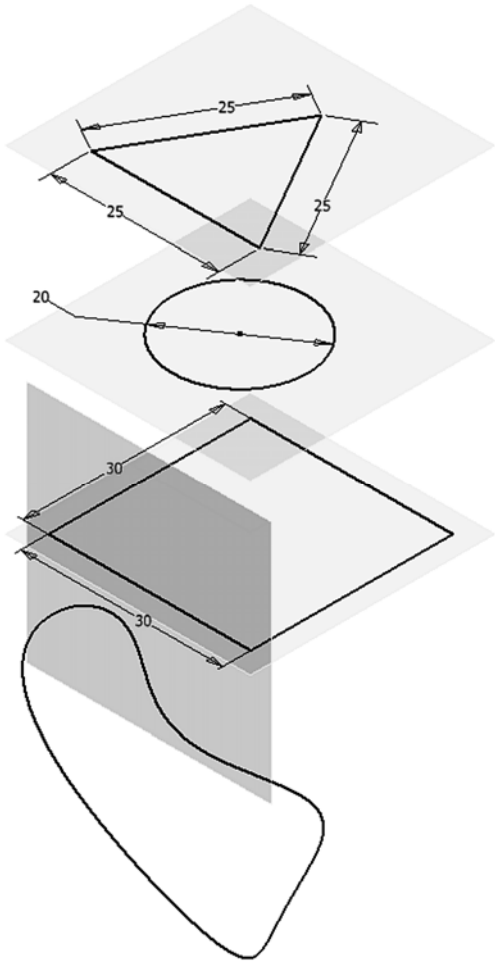
2.78. ábra
Az alkatrész a Shell parancs után

A Shell parancs után készített sajátosságok már nem üreges jellegűek, ezért vigyázzunk, hogy a művelet előtt jól nézzük meg, ha minden sajátosság rajta van a modellünkön.

A modellezések során általában az állandó falvastagság ajánlott, a későbbi megmunkálási torzulások és feszültségek elkerülése végett, de lehetőség van a párbeszédablak alján található „<<” ikonnal ellátott gombot megnyomva az egyedi falvastagságok beállítására is.

2.9.13. A Loft (Pásztázás) sajátosság

A sajátosságoknak ez a típusa, az eltérő profilok összekötésére szolgál. A profilok száma bármennyi lehet, ezáltal egészen különleges formájú alkatrészek is létrehozhatók. A bemutató példában 4 profilt alkalmaztunk, amely négy vázlat létrehozását igényli.

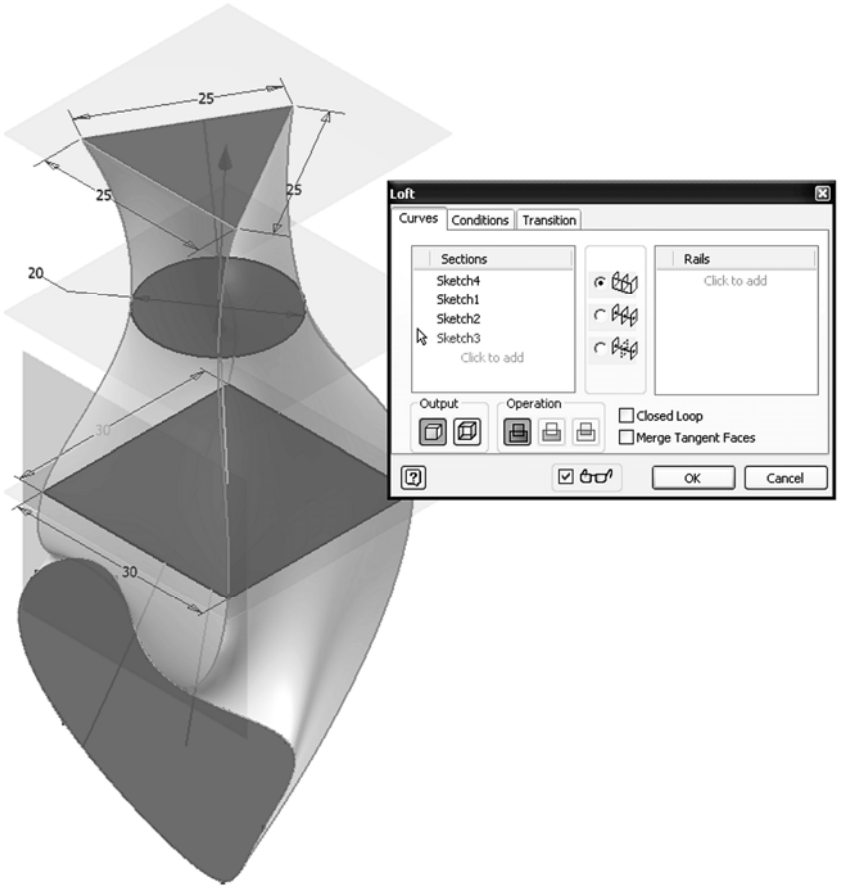


2.79. ábra
A Loft parancsot előkészítő négy vázlat

Nyissunk egy új **Standard(mm).ipt** lapot. Az első vázlatunk egy 30x30 mm-es négyszög lesz. Fejezzük be a vázlatot egy Return paranccsal. Work Plane paranccsal készítsünk egy munkasíkot rá, úgy, hogy jelöljük ki az Áttekintőtárból az első vázlatot. Készítsünk még két munkasíkot úgy, hogy kijelöljük (kék színű lesz) az első síkot, és elhúzzuk azt felfele, offset távolságnak 25 mm-t írva. A harmadikat a második elhúzásával, ugyancsak felfele 25 mm-re készítjük. A negyedik munkasíkot az első vázlatából az egyik oldalt kijelölve merőlegesen húzzuk az első síkra. A második síkra egy új vázlatot helyezünk (New Sketch), erre egy 30 mm átmérőjű kört rajzolunk. A harmadik síkra is készítsünk egy új vázlatot, erre pedig

egy 25x25x25 mm-es, egyenlő oldalú háromszöget helyezzünk el. A **2.79. ábrán** látható negyedik vázlatunkat a merőleges síkra helyezzük, és egy, az ábrán látható görbéhez hasonló, zárt **spline** görbét készítünk.

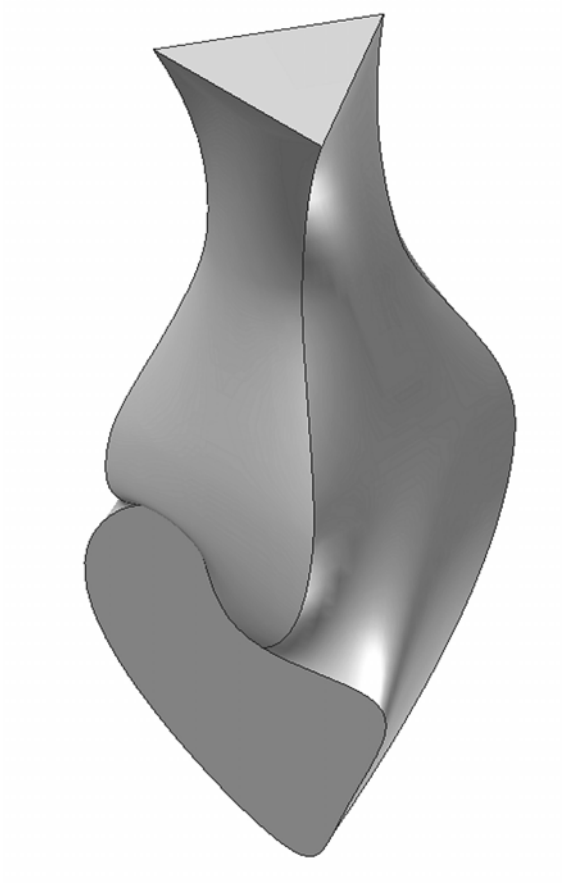
Indítsuk el a Loft parancsot a Part Features panelből.



2.80. ábra
A négy vázlat kijelölésének eredménye

A **Sections** ablakba kattintva ki kell jelölnünk először a spline görbe vázlatát, újra az ablakocskába kattintva az alsó vízszintes vázlatot, újrakattintva a második vízszintes, majd újabb kattintás után a felső vízszintes vázlatot. Minden új vázlat kijelölése után megjelenik a pasztázott testmodellünk pillanatnyi állapotában, végül a negyedik vázlat kijelölése után, a **2.80. ábrán** látható a modell előnézete, majd a **2.81. ábrán** az alkatrész végső állapotában.

Amint a fenti példában is láttuk, a Loft sajátosság tetszőleges számú vázlatot használhat, de legkevesebb kettőt. Ha több vázlatunk is létezik a szükségesnél, kihagyhatjuk bármelyiküket, minden vázlat kijelölésnél szükség van tudatni, egy kattintással a párbeszédablakban, hogy újabb vázlat kijelölése következik.



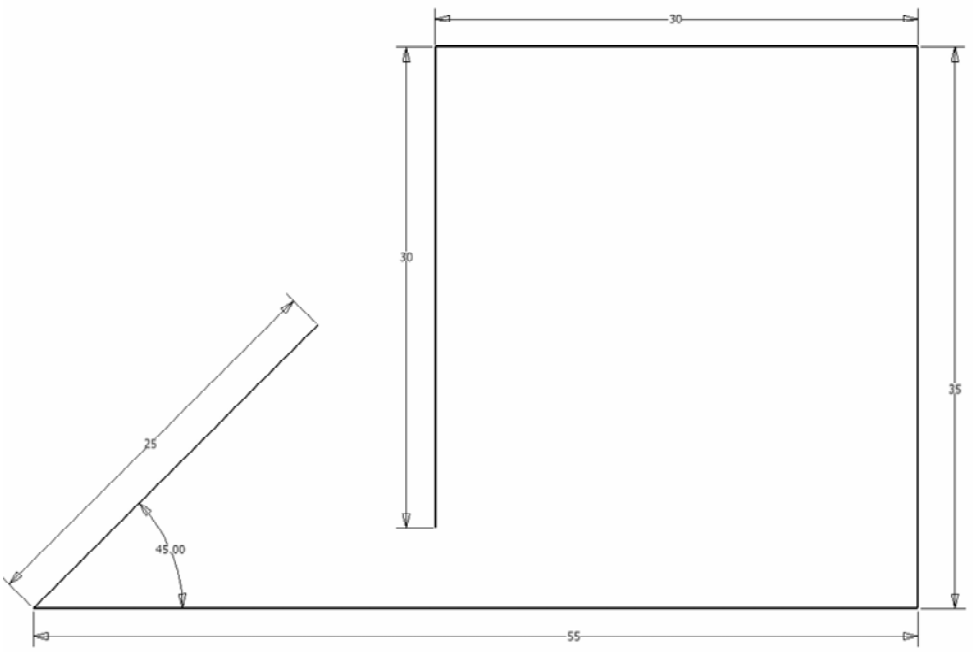
2.81. ábra
A Loft paranccsal elkészített testmodell

A 2009-es Inventor Professional Loft párbeszédablakában még olyan lapok találhatók, mint Conditions és Transitions, amely a pasztázás finomítását szolgálják. Ezek a részletek, a felületi modellezési sajátosságok egy későbbi kötet, a Parametrikus modellezés anyaga lesz.

2.9.14. A Sweep (Seprés) sajátosság

Ez a sajátosság egy megadott pályagörbe „végigseprését” teszi lehetségesé egy profillal, eredményként egy felületi vagy testmodellt képezve.

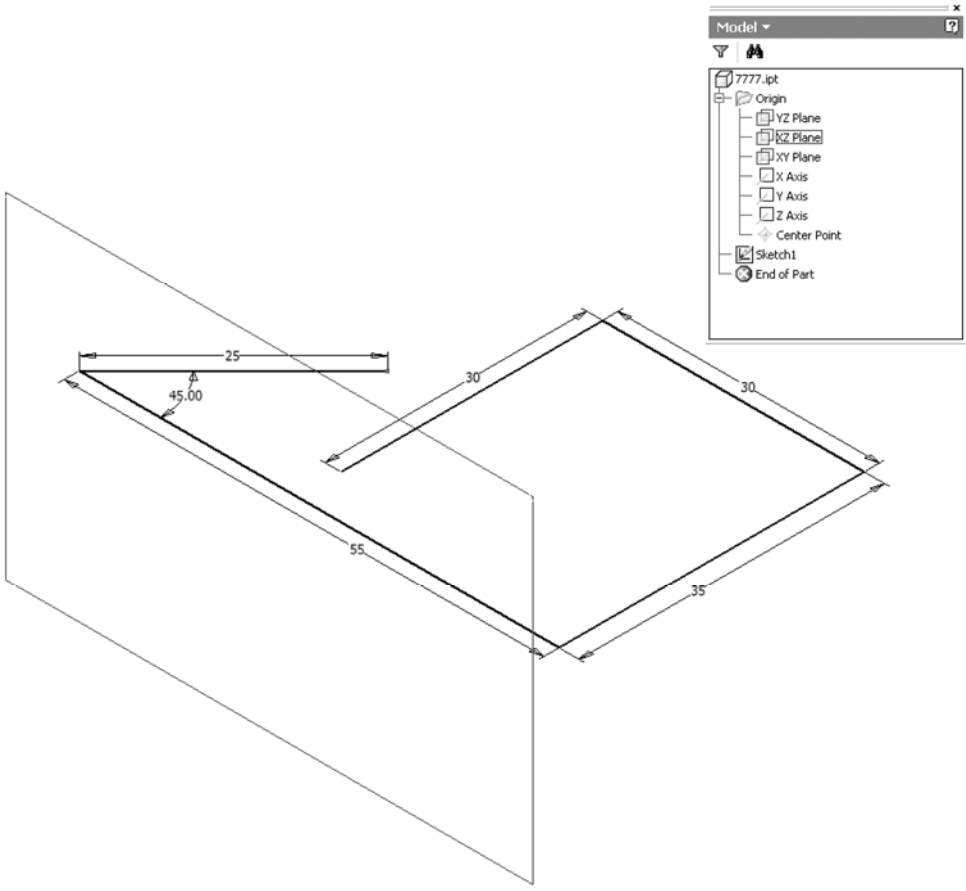
Példánkban két egymásra merőleges síkban fekvő vázlatra lesz szükségünk. Az elsőt egy új Standard(mm).ipt nyitásakor induló vázlatra készítjük; öt egyenesből áll, és a **2.82. ábrá**n látható méretekkel látjuk el, General Dimensiont használva. Vigyázzunk, a vázlatot a ferde egyenes végétől kezdve, folytonosan rajzoljuk, utána tegyük fel a méret kényszereket!



2.82. ábra
Az első, öt egyenesből álló vázlat

A következőkben térjünk át egy Home View paranccsal izommetrikus nézetre. Készítsünk egy munkasíkot úgy, hogy kijelöljük a 45° megdőlt egyenes szabad végét, és az XZ alapsíkot (**2.83. ábra**). A létrejött munkasíkon indítsunk egy új vázlatot, erre pedig rajzoljunk egy 10 mm magas és 5 mm széles négyszöget a Two Point Rectangle paranccsal úgy, hogy a ferde vonal szabad vége a négyszög belsejében legyen. Használjuk az Orbit parancsot (View menü vagy Inventor Standard Panel), hogy egy, a munkánkat megkönnyítő nézetre forgathassuk rajzunkat. Vetítsük le az egyenes végét (amely egy pont) erre a síkra, és ebből kiindulva húzzunk egy függőleges és egy vízszintes vonalat. Az egyenesek meghúzása előtt

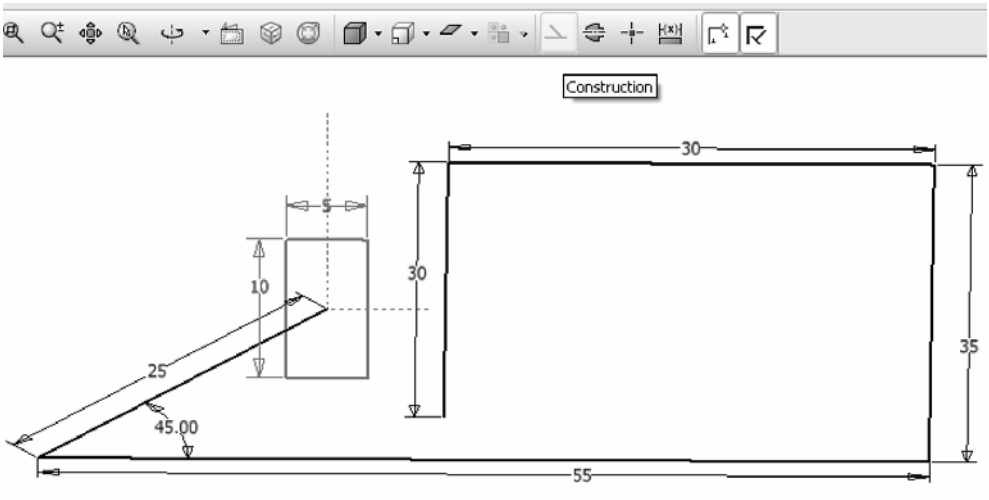
kattintsunk az Invetor Standard Panel Construction ikonjára (**2.84. ábra**). Ez lehetővé teszi, hogy az utána létrehozott geometriai elemeket csak különböző konstrukciókra használjuk, és ne legyen hatásuk, csak abban a vázlatban amelyben használjuk. A Construction-nal rajzolt elemek szaggatott vonallal jelennek meg a rajzokon.



2.83. ábra
A második vázlatsík meghatározása

Indítsunk Symmetric (szimmetrikus) kényszert a Sketch Panel Bar-ból. Két lépésben jelöljük ki, páronként, a négyzetnek az egymással szemben fekvő oldalait, utána pedig a közéjük eső egyeneseket, majd a rájuk merőleges oldalakat, majd közöttük levő egyenest. Vigyázzunk, hogy az egyes szimmetria kényszerek megtétele után, fejezzük is azt be, a jobbegérgomb – Done paranccsal, egyébként hibajelzést kapunk!

Ily módon, a második vázlat helyzetét szimmetrikusan rögzítettük az első vázlathoz (**2.84. ábra**). Most az Browser Bar-ban két felhasználatlan vázlatunk lesz. Mindkettőre szükségünk lesz egy Swep sajátosság létrehozásához. A második vázlat befejező lépéseként indítsunk egy Offset parancsot a vázlat Paneltárából, jelöljük ki a négyszöget, és húzzuk az egérrel a belseje felé. Tegyük 1 mm-re a két négyszög közötti távolságot (későbbiekben ez lesz a falvastagság). Return-al fejezzük be a vázlatot.



2.84. ábra
A két megrajzolt vázlat

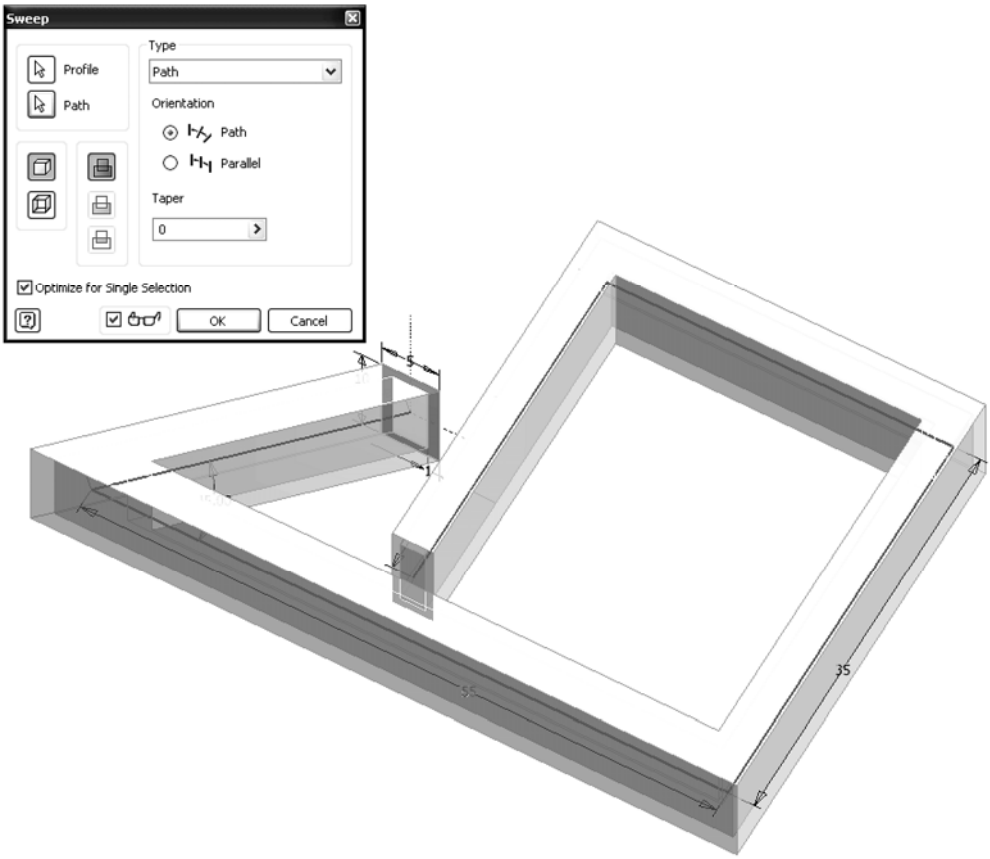
Indítsunk a Part Features-ből egy Sweep Parancsot. A **2.85. ábrán** látható párbeszédablak nyílik meg. A mi esetünkben, mivel két zárt profilunk van, nem jelölődik ki automatikusan a két négyszög közötti profil, így ki kell azt jelölnünk. Szükséges a továbbiakban kijelölni – a Path gombra kattintással (itt már automatikusan meg van nyomva) – a seprés útvonalát. Ez nálunk az első vázlat 5 egyenesből álló alakzata.

Orientation – Irány. A párbeszédablak bal felső részében, a **Path** és a **Paralel** opciókat ajánlja. Az első, implicit választás, az egymással nem párhuzamos profilseprést alkalmazza. A Paralel opció, olyan alakzatnak a létrehozást biztosítja, amelyben pontosan a seprési profil található meg, minden egymással párhuzamos metszetben. A **2.84. ábrán** látható profilok estében nem alkalmazható, mivel a második vonalrésznél a profil már érintőleges az útvonalhoz.

Taper – Az Extrude-hoz hasonló módon lehetőséget biztosít, adott szögel való seprésnek.

Amint a **2.85. ábrán** is látható, megjelenik előnézetben a seprés eredménye, és ha ez megfelel céljainknak, akkor az OK gombbal befejezhetjük a műveletet. Az

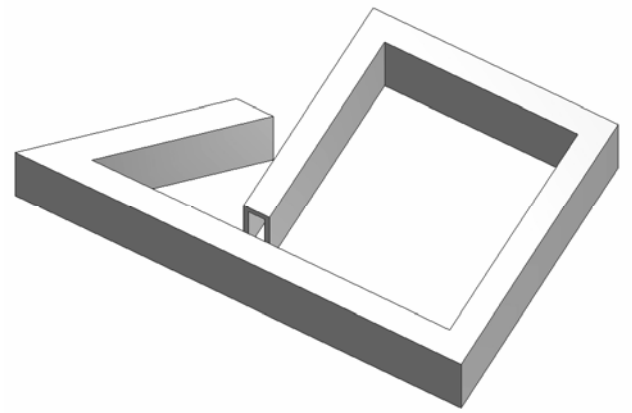
eredmény, az első vázlat vonalaival megegyező négyszög profilú csővezeték (**2.86. ábra**).



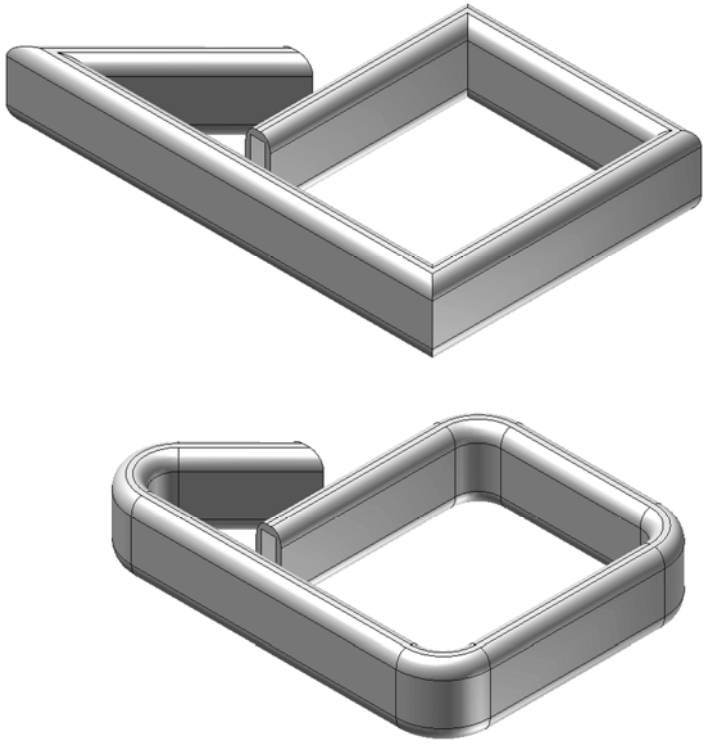
2.85. ábra
A seprés, előnézetben

A Sweep művelettel épített testmodell is szerkeszthető. Mind a profil, mind a seprés útvonala, vázlatszinten szerkeszthető. A **2.87. ábra** felső képén a profilmódosítás eredménye látható: dupla egérklikkelve a **Sketch2** vázlatra, szerkesztési állapotba kerül, majd egy 2 mm-es sarokkerekítést alkalmazva a négy sarokra, és befejezésként a Return gombot megnyomva, az ábra felső képén látható testet nyerjük.

Az **2.87. ábra** alsó képén további módosításokat láthatunk: a **Sketch1** vázlatban 5 mm-el kerekítettük le a seprési útvonalat, majd Return-el zártuk a szerkesztést.



2.86. ábra
A Sweep parancs végeredménye

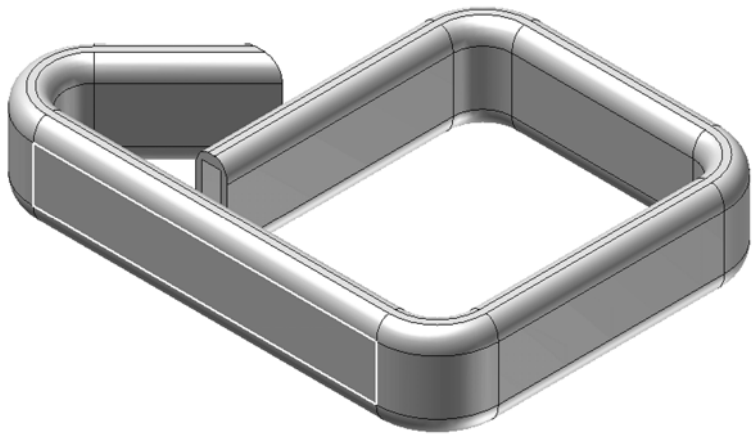


2.87. ábra
A Sweep paranccsal létrehozott test utólagos módosításai

2.9.15. A Split (Kettéosztás) sajátosság

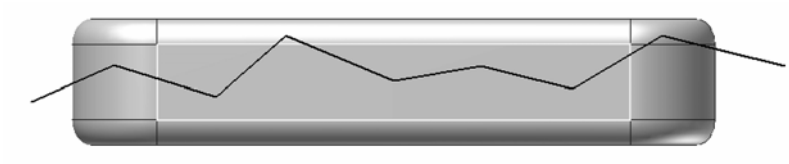
A Split parancs segítségével lehetőségünk van egy testet kettőbe vágni, és ha szükséges, annak egy részét eltávolítani. Ez a művelet igen hasznos bonyolult testmodellek létrehozására: bonyolult felületeket importálhatunk más alkalmazásokból, vagy készíthetünk az Inventor 2009 Professional-al, ezek segítségével pedig, elvághatjuk a sajátságokkal készült testmodelljeinket.

A Split parancs legegyszerűbb alkalmazása, egy egyszerű felület készítése az előző modellünkön. Ehhez, egy új vázlatot készítünk a **2.88. ábrán** látható felületre.



2.88. ábra
Új vázlat elhelyezése

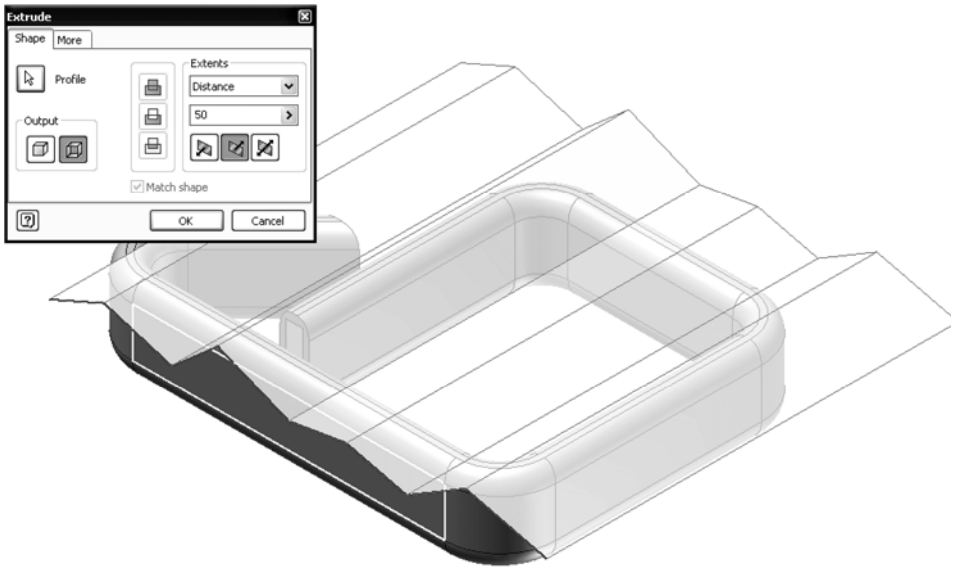
Erre a **2.89. ábrán** látható vonalakhoz hasonló profilt rajzolunk, vigyázva, hogy magasságban ne lógjon ki a testmodellből, vízszintesen pedig a végei legyenek a testen kívül.



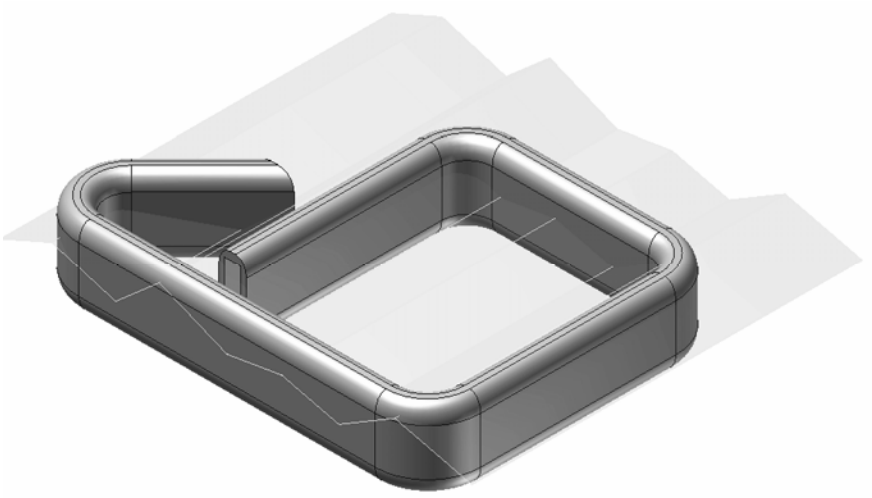
2.89. ábra
Az új vázlat profilja

Újabb lépésként egy Extrude paranccsal húzzuk ki a profilt: jelöljük ki a felületi kihúzásnak megfelelő ikont (**2.90. ábra**), a vonalakkal rajzolt profilt, válasszuk ki azt a kihúzási irányt, amellyel a kihúzás a test felé történik, majd állítsuk be a kihú-

zási távolságot 50 mm-re. Az Extrude eredménye izommetrikus nézetben a **2.91. ábrán** látható.



2.90. ábra
A felület kihúzásnak előnézete



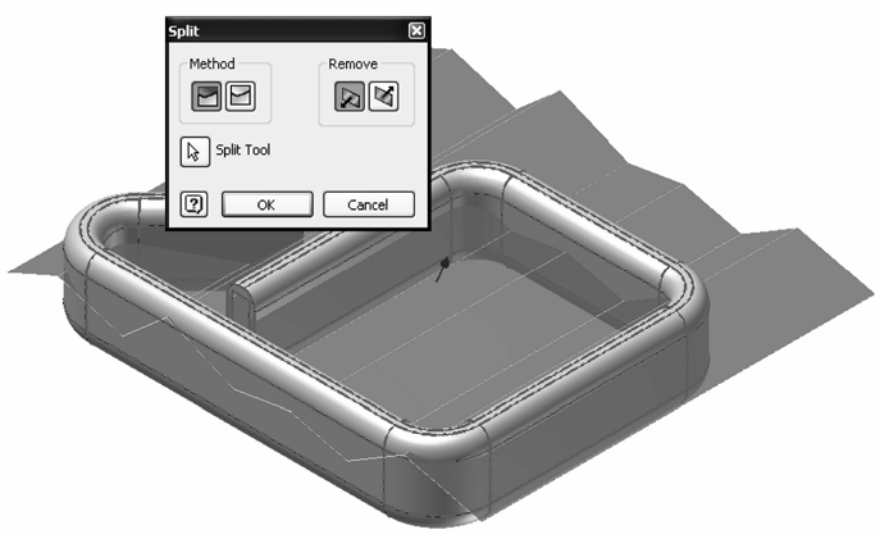
2.91. ábra
A vágási felület

Indítsuk a Split parancsot a Part Features panelből. A megjelenő párbeszédablak a **2.92. ábrán** látható.
Methode – Módszer. Két vágási módszerből választhatunk:

- **Split Part** – a test egyik része – a felhasználó választása alapján – el van távolítva.
- **Split Face** – a test mindkét része a modellen marad.

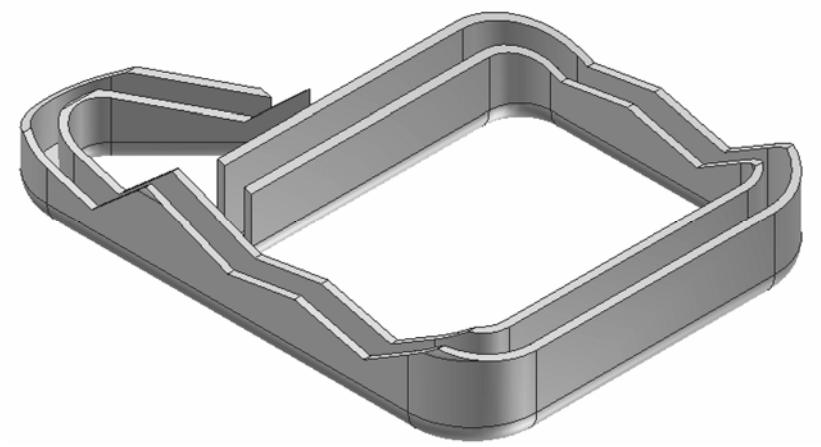
Split Tool – az ikont megnyomva választhatjuk ki a vágó felületet.
Remove – az eltávolítandó részt válasszuk ki a két lehetséges gomb egyikének megnyomásával.
Faces – a második osztási módszernél **All, Faces** és **Faces to Split** opciókkal lehet a vágott felületeket kiválasztani.

Használjuk a Split Part módszert (bal oldali ikon). Jelöljük ki az előzőkben létrehozott felületet a **Split Tool** gomb megnyomása után. A párbeszédablak bal oldalán lévő ikonokból válasszuk ki a test, vágás utáni részének az eltávolítását (Split Part, bal oldali ikon). Felhasználva a **2.92. ábrán** látható elönézetet, a **Remove** ikonokból válasszuk ki azt amelyik eredményeként, a test felső része vágodik le. A Split eredménye a **2.93. ábrán** látható, immár láthatatlanná tett vágófelülettel (Visibility kikapcsolva).



2.92. ábra
A Split parancs párbeszédablaka

A Split is, mint minden sajátosság, megjelenik az Áttekintőtárban. Erre, ha jobb egérgombbal rákattintunk, akkor szerkeszthetjük, vagy a **Supress Features** paranccsal kikapcsolhatjuk, illetve az **Unsupress Features-**el visszakapcsolhatjuk a Split sajátosságot.



2.93. ábra
Testmodellünk a Split parancs használata után

Meg kell jegyeznünk, hogy a fentiekben Split eljárás egyik legegyszerűbb használatát mutattuk be. A szerző személyes tapasztalata szerint [19] az eljárás szinte korlátlan lehetőséget biztosítl az Inventor nyújtotta modellezések terén.

2.9.16. A Coil (Spirál) sajátosság

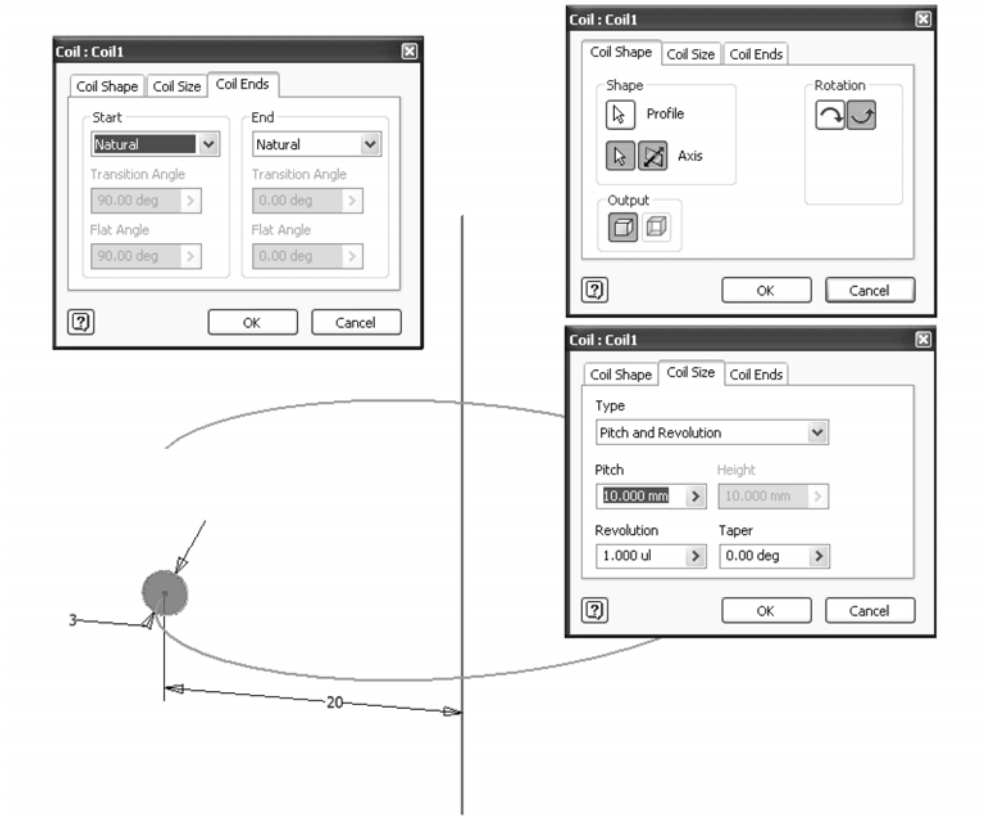
A Coil sajátossággal csavarmeneteket, csavarrugókat és síkbeli spirálrugókat készíthetünk. Elsőként egy csavarrugót készítünk. Vázlatként a rugó szelvényének megfelelő kört, valamint a rugó tengelyét meghatározó vonalat rajzoljunk. A kör átmérője 3 mm, középpontja a vonaltól 20 mm távolságra van.

A Coil parancs ikonja a Part Features eszköztárából adható ki, majd a megjelenő párbeszédablakban állíthatjuk be a további méreteket.

Mivel a vázlat csak egyetlen profilt tartalmaz, ennek kijelölése automatikusan történik meg és következhet a tengely kiválasztása (**2.94. ábra**). A profilnak és a tengelynek ugyanazon a vázlaton kell elhelyezkednie. Ez alól csak az jelent kivételt, ha a spirál rajzolásához tengelyként egy munkatengelyt választunk. Ilyenkor már megjelenik a rugó vonalas rajza, amely alapján eldönthetjük, hogy megfelelő-e az irány. Szükség esetén az irányváltó nyomógomb segítségével változtatunk. A párbeszédablak első lapján kell beállítani, hogy jobbra vagy balra csavarodó spirált akarunk készíteni.

A **Coil Size** (Méret) lapon, a **Type** (Típus) részben állítható be, hogy milyen méretek alapján akarjuk megrajzolni a spirált. Választhatjuk a **Pitch and Revolution** (Menetemelkedés és menetszám), **Revolution and Height** (Menetszám és magasság), vagy az **Pitch and Height** (Menetmelkedés és magasság) megadását, illetve a **Spiral** (Síkbeli spirál) rajzolását. A harmadik lapon, a **Coil**

Ends (Lezárás) részben adhatjuk meg a spirál kezdésére és befejezésére vonatkozó utasítást. **Natural** opció esetében a megadott menetemelkedésnek megfelelően kerül kialakításra a rugó kezdete és vége. A **Flat** kivitelezés választásakor az utolsó menet úgy jön létre. hogy rugó fel tudjon feküdni egy felületre. A két vég kialakítására eltérő kivitelezések is megadhatók.

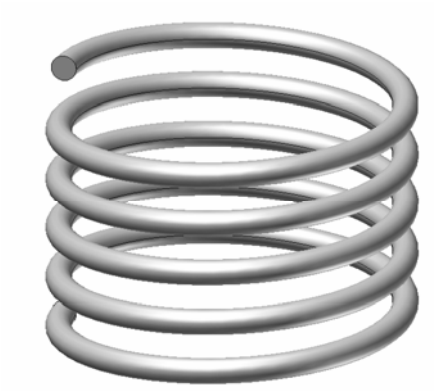


2.94. ábra
A Coil párbeszédablak lapjai

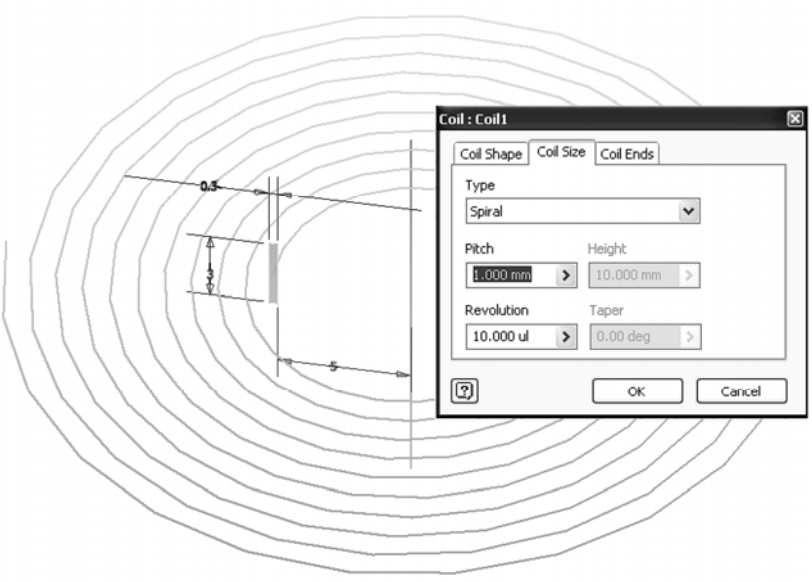
A **Transition Angle** részben megadhatjuk, hogy az utolsó menet lapítása milyen szögtartományban érvényesüljön. A **Flat Angle** értékével határozzuk meg, hogy a rugó utolsó menete által meghatározott sík mekkora szöget zárjon be a rugó tengelyével.

A Pitch and Revolution opció esetén, Pitch-nek 6 mm-t, a Revolution-nak (menetszám) pedig 5-öt állítunk be, és a **2.95. ábrán** látható csavarrugót kapjuk.

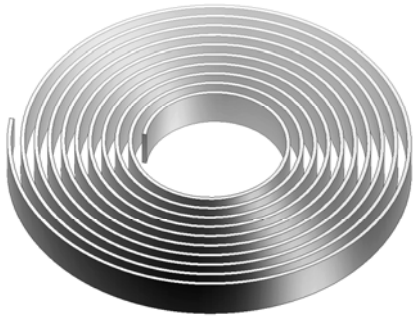
A Coil párbeszédablak **Coil Size** lapján, a **Type** részben, a **Spiral** sort is kiválaszthatjuk. Ennek segítségével archimédeszi spirált hozhatunk létre. Itt is szükségünk van egy profilra, valamint egy vonalra is, amely a forgatás tengelyeként fog szerepelni. A vázlat méretei, a spirálrugó előnézete és méretei, a **2.96. ábrán** láthatók, a **2.97 ábrán** pedig a kész spirálrugó.



2.95. ábra
A Coil paranccsal készült csavarrugó



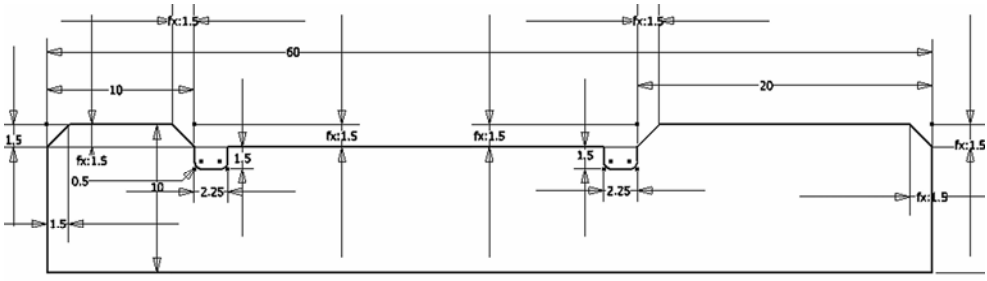
2.96. ábra
A spirálrugó méretei



2.97. ábra
A spirálrugó testmodellje

2.9.17. A Thread (Menet) sajátosság

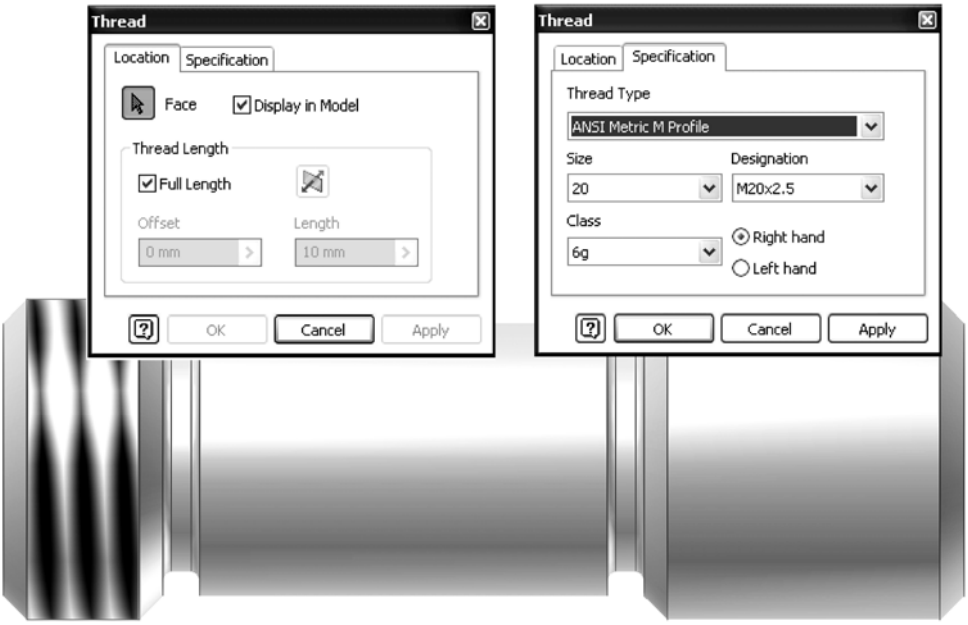
A menetes furatok megjelenítésére két lehetőség van. A már ismertetett Hole sajátosság párbeszédablakában, már a furat létrehozásakor előírhatjuk a csavarmenet megjelenítését, de ez nem alkalmazható külső csavarmenetek esetén. Erre az önálló eszközként rendelkezésre álló Thread sajátosságot használhatjuk. A sajátosság kipróbálására egy olyan hengeres alkatrész modelljét készítjük el, amelyet megforgatással hozunk létre. A vázlat profilja a **2.98. ábrán** látható. A vázlatprofil megforgatásával egy olyan hengeres alkatrészt hozunk létre, amelynek mindkét végén csavarmenet lesz (**2.99. ábra**).



2.98. ábra
A megforgatás vázlata

A Thread sajátosság ikonja párbeszédablakot jelenít meg. A **Location** (Hely) feliratú lapon a hengeres felület kijelölésére, valamint a menet elhelyezésére vonatkozóan végezhetünk beállítást. Ha a **Full Length** (Teljes hossz) kapcsoló nincs bekapcsolva, akkor a menetes rész hosszát és a menetnek az alkatrész kezdéséhez viszonyított eltolását adhatjuk meg (**2.100. ábra**).

2.99. ábra
A megforgatással létrehozott testmodell



2.100. ábra
A Thread párbeszédablak lapjai

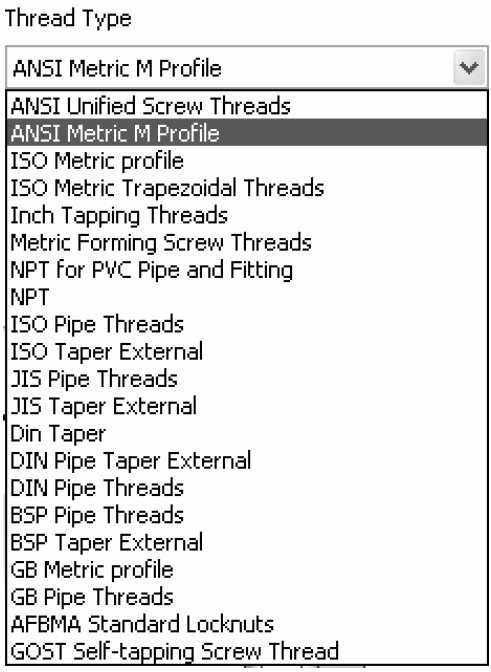
A párbeszédablak **Specification** lapján a Thread Type (Menet típusa) legördülő listából választhatjuk ki, valamint meghatározhatjuk a menet névleges méretét (**Nominal**) és a menetemelkedés (**Pitch**) értékét. Itt állíthatjuk be a menet osztályba sorolását és az irányt is. Példánkban a két végcsapon teljes hosszban M120x1.5 méretű jobbmenet lesz.

Megjegyzés: a sajátosság által megjelenített csavarmenet nem valódi. Csak az ábrázolást segíti elő. Ha azonban az alkatrészmodellből síkrajzot készítettünk az Inventorral, akkor ezen már megjelenik a menet.

Amikor a Thread sajátosság alkalmazásakor megjelenik a párbeszédablak, és a Specification lapon ki kell választanunk a menet típusát (Thread Type), a Nominal, a Pitch értékeit, valamint a menetet jellemző értéket, akkor az Inventor egy előre

meghatározott típusokat tartalmazó, .xls kiterjesztésű állományban tárolt, táblázatot használ. Az alapértelmezés szerinti telepítéskor a Thread.xls állomány a Program Files\Autodesk\Inventor 2009\Design Data mappában található. Ennek következtében Microsoft Excel előzőleges telepítése nélkül a Thread használata lehetetlen. Azonban ha gépünkön telepítve van, akkor meg is változtathatjuk a meneteket tartalmazó állományt, ezeket akár ki is bővíthetjük sajátságos menettípusokkal.

A Thread Type részben a **2.101. ábrán** látható csavarmenet típusok választhatók.



2.101. ábra
Az Inventor 2009 Professional-ban használható menettípusok

A darab két végén Thread sajátsággal ellátott testmodell a 2.102. ábrán látható. A 2.103. ábrán pedig, a Full Length opciót elhagyva, a tengely középső részén egy 10 mm-es részt meneteltünk meg, a darab belső hengeres részének jobb végétől 10 mm-es Offset értékkel.

2.102. ábra
A két végén menetelt darab

2.103. ábra
Menetkészítés Full Length opció kikapcsolása mellett

2.9.18. Munkasík, munkatengely és munkapont

Amint az eddigi példák során is láttuk, elsősorban munkasík nélkül a parametrikus modellezés elképzelhetetlen. A munkasíkok használata azért igen fontos mivel a különböző sajátosságok vázlatainak alapját jelentik.

Mindhárom geometriai elem létrehozásáról csak annyit kell tudni, hogy meghatározásuk ugyanúgy történik, mint az elemi geometriában.

Munkasíkot legegyszerűbben egy, már létező síkfelület párhuzamos eltolásával nyerünk. Ha a Part Features panel Work Plane parancs indítása után bal egérgombbal klikkelünk egy síkfelületre, lenyomva tartva az egérgombot és elhúzva azt a kijelölt felületről, akkor megjelenik egy kis ablak – **Offset** cimkével – amely a munkasíknak az eredeti síktól való távolságát kéri.

Ha egy pontot, élet vagy munkatengelyt jelölünk ki a Work Plane parancsra, utána pedig egy síkot (akár az Browser Bar-ban található koordináta rendszer elemeit is) kijelölve, egy másik ablak jelenik meg **Angle** név alatt, amely a referenciasíkhoz képest kéri a szög megadását. Lehetséges síkokra merőlegesen is munkasíkot építeni, minden esetben az előnézet lehetőséget ad a helyes opciók kiválasztására.

Munkasíkot meghatározhatunk két – nem kitérő – egyenessel is, akár egy egyenes és egy pont-párral, vagy három pont által. Gyakorlatilag minden geometriai elem segítségével, ami a klasszikus geometriában is egy síkot egyértelműen meg tud határozni.

Munkatengelyek létrehozhatók: egy-egy él kijelölésével, egy forgástest, vagy forgástest-részlet forgástengelyeként, két pont segítségével, két sík metszéseként, és minden olyan geometriai meghatározásként, amely segítségével egy egyenes meghatározható.

Munkapontok építhetők: testmodellek éleinek találkozásánál, különböző metszéspontokkal, látható vázlatok elemei segítségével stb.

A bemutatott segítő geometriai elemek használata nélkül elképzelhetetlen lenne bármely parametrikus modellező szoftver hatékony használata, ezért ezek kötelező módon meg is találhatók minden szoftvertípusban.

2.10. Síkrajzok készítése testmodellek alapján

Az alkatrészekről és összeállításokról készített térbeli modellek jól használhatók a kész termék bemutatása mellett, a szerelési folyamatok megtervezésére, az esetleges átfedések felderítésére stb., de legtöbb esetben az elsődleges cél mégiscsak egy precíz síkbeli rajz létrehozása a 3D modellből. Az Inventor ezt a műveletet kevés beavatkozással, csaknem teljesen automatikusan hajtja végre.

Amikor síkbeli rajzot akarunk létrehozni, akkor egy **.idw** kiterjesztésű sablon-állományt kell választanunk. Az **Open** párbeszédablakban az **Default** (Alapértelmezett) lapon a **Standard.idw** áll rendelkezésünkre, de választhatunk a Metric lapon az ISO, DIN. JIS stb. szabványok, illetve az English lapon az ANSI szabvány szerint beállított sablon-állományok közül. A sablon-állomány betöltésekor, alapértelmezés szerint, A3 méretű rajzlap jelenik meg. A szövegmező felosztása függ az alkalmazott szabványtól. Az ISO.idw sablon-állomány esetében az ISO címpecsétnek nevezett szövegmező kerül alkalmazásra. Ha az általunk készítendő rajz az A3 mérettől eltérő rajzlapot igényel, akkor célszerű a vetületek létrehozása előtt módosítani a rajzlap méretét. A rajzlap méretének módosítását a vetületek létrehozása közben, vagy akár a kész rajznál is elvégezhetjük, ilyenkor azonban a vetületek helyét, esetleg méretarányát is utólag kell hozzáigazítani az új mérethez.

Megtehetjük azt is, hogy új, egyéni méretek szerint beállított rajzlapot hozunk létre, amelynél a keretet és a szövegmezőt is egyéni beállítás szerint készíthetjük el. Az új beállítások egyedi névvel elmenthetők, és a későbbiekben más rajzokhoz is felhasználhatók. Ezeknek az előre beállított sablonoknak, a szabványoknak-, valamint a többi rajzolófelület beállításának módja nem a jelen kötet témája.

2.10.1. Vetületek létrehozása

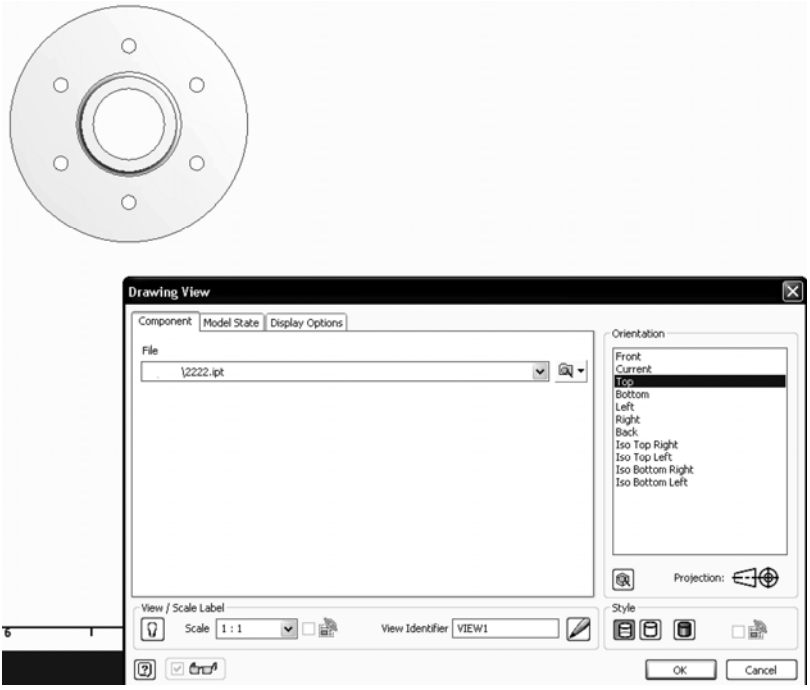
A rajz/lapméret beállítása után következhet a rajzi **nézetek** – metszeti és nézeti vetületek – létrehozása. A **Command Bar** (Parancstár) az .idw kiterjesztésű sablon-állomány betöltésekor, automatikusan a **Drawing Views Panel** (Rajznézetek panel) eszköztár ikonjait jeleníti meg. Ebből az eszköztárból adhatjuk ki a **Base View** (Bázisnézet) parancsot, amely az egérrel történő rákattintáskor párbeszédablakot jelenít meg a beállítások elvégzéséhez.

Alapnézetnek (**Base View**) nevezzük az elsőként létrehozott síkbeli nézetet, amely nem igényli más vetület meglétét, viszont alapját képezheti további nézeti és metszeti vetületeknek, valamint segéd- és résznézeteknek.

Ha az .idw kiterjesztésű sablon-állomány megnyitásakor egy alkatrészrajz is megnyitott állapotban van, akkor a program felkínálja, hogy erről készíthetünk vetületet. A **File** részben ennek a állománynak a neve jelenik meg, de természetesen választhatunk más alkatrészt is Open Existing File ikonra kattintással. Több alkatrész betöltött állapotában az utolsóként szerkesztett alkatrészt jeleníti meg a program.

Példaképpen az egyik már elkészített modellünket használjuk fel (**2.53. ábra**) a vetületek létrehozásának kipróbálására. Keressük meg a állomány nevét (**2.104. ábra**). Megnyitáskor megjelenik az ideiglenes bázisnézet, de a **View** részben

megváltoztathatjuk a vetítési irányt. A párbeszédablakban beállíthatjuk a méretarányt és a nézet stílusát is.

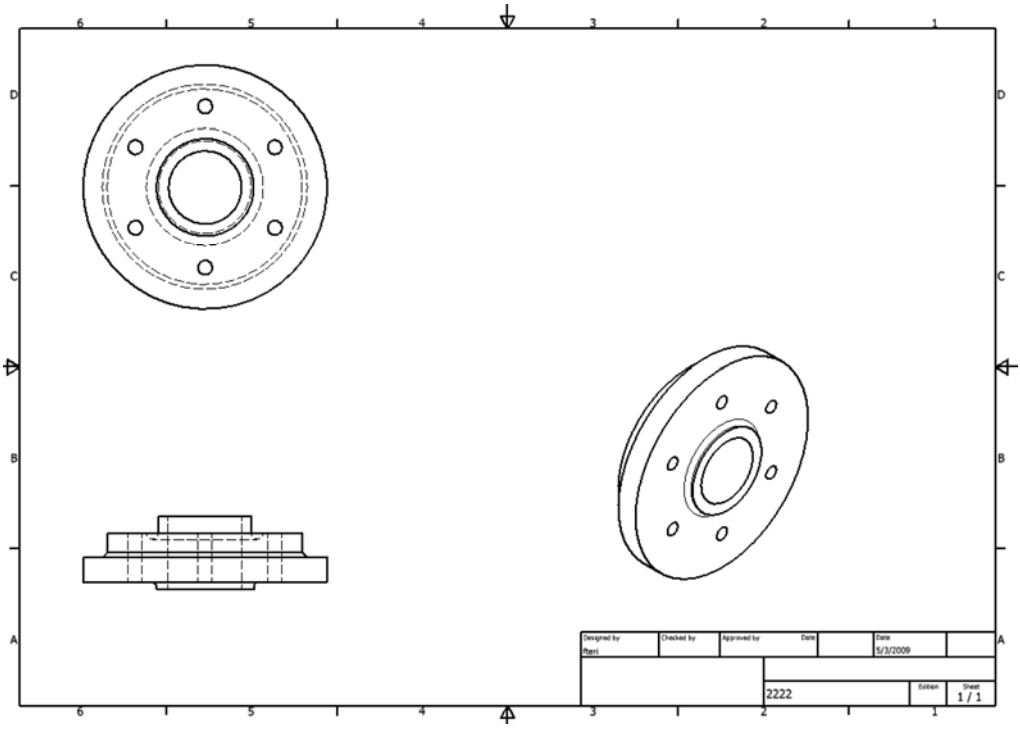


2.104. ábra
A Drawing View párbeszédablaka

A Drawing View párbeszédablakban megnyitva az alkatrész-állományt, ideiglenesen megjelenik a nézet, amelyet az egérkurzorral tudunk a megfelelő helyre elhelyezni (a **2.104. ábrán** a bal felső sarok). Egy alapos műszaki rajznál további vetületre van szükség. Ehhez a Drawing View Panel eszköztárban több lehetőség közül is választhatunk.

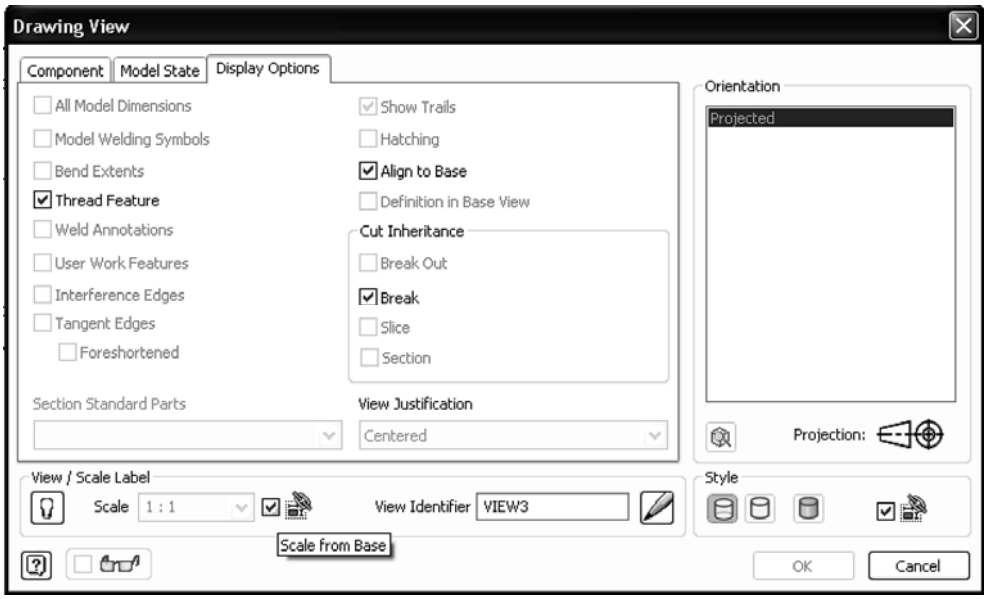
Először is készítsünk egy **származtatott nézetet**, **Projected View** paranccsal. A származtatott nézetek lehetnek a bázisnézet bármely oldalán elhelyezett merőleges vetületek, de izommetrikus nézeteket is a Projected View paranccsal hozhatunk létre. A Panel Bar eszköztárából elindítva a Projected View parancsot, az első kattintással ki kell jelölnünk a bázisnézetet. A megjelenő ideiglenes nézet attól függően változik, hogy az egérkurzort melyik irányba húzzuk el. A rajzunkon jobbra, átlósan lefele mozgatva az egérkurzort, axonometrikus kép jelenik meg, amelynek a helyét egy egérkattintással adhatjuk meg. Ez a kattintás csak egy nézetablakot hoz létre, amelyben úgy véglegesíthetjük a nézet helyét, hogy a jobb egérgombbal megjelenített menüben rákattintunk a **Create** opcióra.

A következő lépésben a bázisnézetről egy vetítést készítünk lefele. Újra elindítjuk a Projected View parancsot, kijelöljük az alapnézetet, és az egérkurzort elmozgatjuk lefele, rákattintunk a rajzfelületre, jobb egérgomb kattintással és a Create paranccsal, az eddigi munkánk alapján, a **2.105. ábrán** látható nézeteket kapjuk.

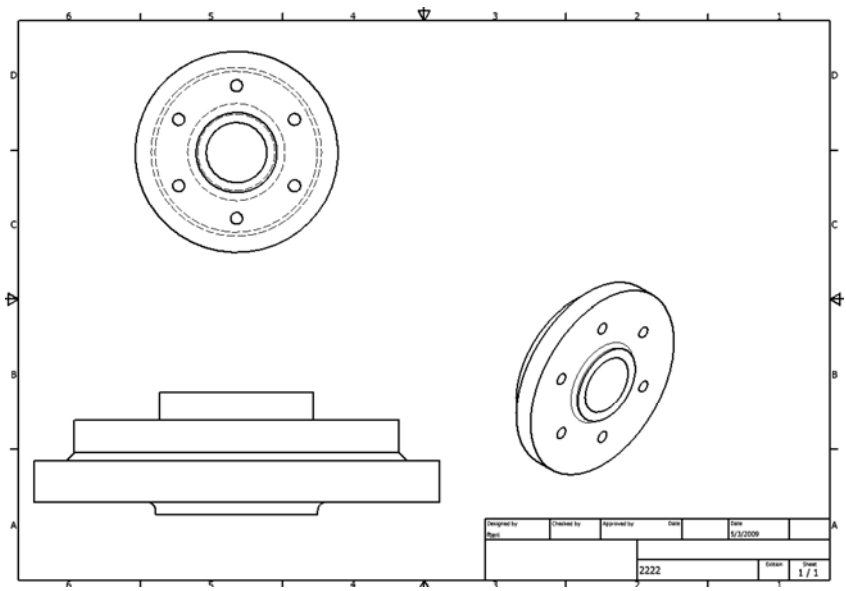


2.105. ábra
Alapnézet és két származtatott nézet

Szerkesszük át a bal alsó nézetet! A jobb egérgombbal a nézetre kattintva, a helyi menüből, az **Edit View...** menüsorral, párbeszédablakot jelenítünk meg (**2.106. ábra**). A párbeszédablak **View/Scale** részéből a **Scale From Base** kipipálását töröljük egy rákattintással, majd lehetőségünk van a **Scale** részben 2:1 léptéket beállítani, a párbeszédablak jobb felén lévő **Style from Base** kapcsolót hasonlóan kikapcsolva, kijelölhetjük a **Hidden Line Removed** opciót. Miután a rajzlapról kilógó nézetet a felső alapnézettel egyszerre visszahúzzuk a rajzlapra, az elvégzett műveletek eredménye a **2.107. ábrán** látható.



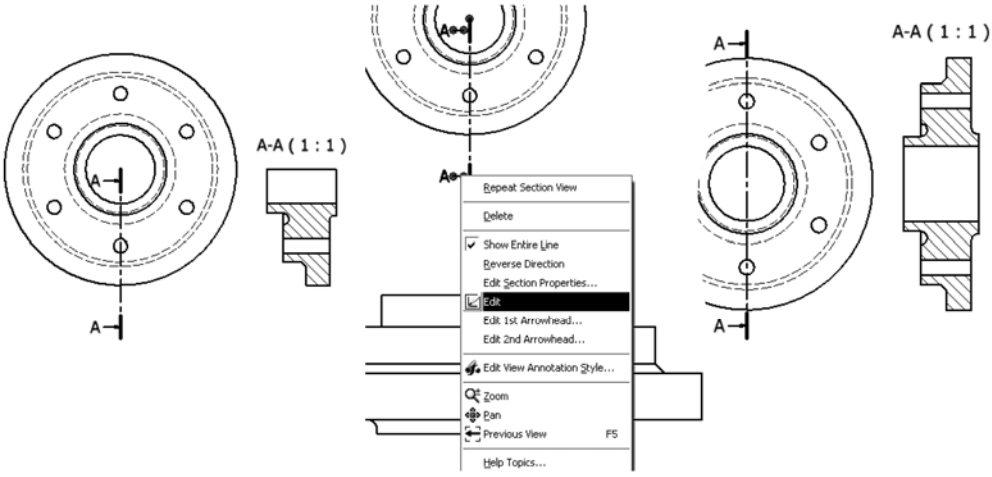
2.106. ábra
Az Edit View parancsra megjelenő párbeszédablak Display Options lapja



2.107. ábra
A módosított nézet

A **metszeti vetület** bármelyik nézetből létrehozható. A metszősík helyénél, kijelölésétől függően készíthetünk teljes-, fél-, lépcsős-, részleges- vagy beforgatott metszetet. Példánkban az elsőt használjuk, amelyet az alapnézetből készítünk. A **Section View** ikonra kattintás után első lépésként rá kell kattintani a bázisnézetre, jelezve, hogy ez lesz az alapja az új vetületnek. A függőleges metszősíknyomvonalat úgy tudjuk pontosan kijelölni, ha kattintás nélkül ráállunk az egérkurzorral a középen lévő körök közepére, ezután meg lefelé toljuk el az egeret, amíg elhagyjuk a darab szélét, majd jobb egérklikkre **Continue**-t nyomunk. Eközben megjelenik a metszet nevét, léptékét, és egyéb adatait tartalmazó párbeszédablak. Az ablakot figyelmen kívül hagyva, jobbra húzzuk az egérkurzort, eszközben előnézetben megjelenik a metszet.

A parancsot a kívánt helyzet elérése után, a rajzlapra való kattintással zárjuk. Amint azt a **2.108. ábra** bal oldali képén is láthatjuk, az eredmény nem kielégítő, nem teljes a metszetünk! De a metszetet szerkeszteni is lehet! Amint az ábra középső képén is látható, jobb egérgombbal rákattintva a metszetvonal alsó nyílára, a megjelenő legördülő menű Edit sorának aktiválására a Draving Views Panel átvált Drawing Sketch Panelre, és a vázlatkészítésnél ismert műveletekkel szerkeszthető a metszet síkjának nyomvonala (amely itt egy vonal), amíg az a nézet felső felét is elhagyja. Példánkban egy vízszintes vonalat húztunk a metszet felé, hogy a vonal továbbra is a darab közepén haladjon át, majd Extend-et használva hosszabbítottuk meg a nyomvonalat. A szerkesztést a Return gomb megnyomásával zárjuk; immár teljes metszetünk a **2.108. ábra** jobb képén látható.

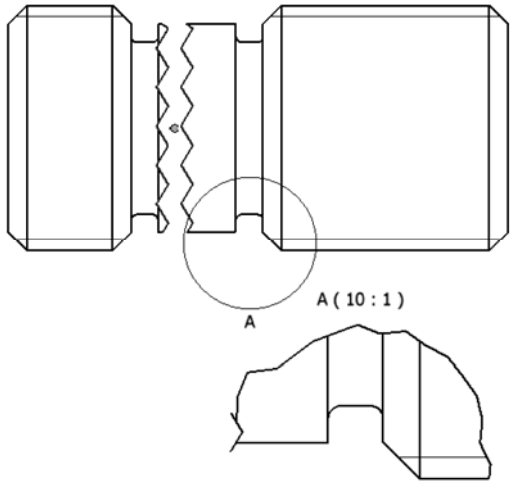


2.108. ábra
A metszet szerkesztésének lépései

Metszetünk a rajzon nem helyezhető el csak az alapnézettel egyvonalban. Ha szükségünk van azt máshova áthelyezni, kattintsunk jobb egérgombbal a metszet-

re (a metszet kerete piros színűre kell váltson!), az Alignment opcióból válasszuk ki a Break parancsot. A művelet után metszetünk bárhova helyezhető a rajzlapon.

Azoknál az alkatrészeknél, amelyeknél viszonylag hosszú szakaszon nem változik a keresztmetszet, alkalmazhatjuk a **töréssel történő ábrázolás**t. Az eszköz a Drawing View Panel eszköztárból indítható el, majd ki kell jelölnünk azt a nézetet, amelynél alkalmazni akarjuk. Példaképpen a **2.103. ábrán** látható hengeres alkatrészt használtuk fel. Az alapnézet létrehozása után a **Broken View** (Megtörés) ikonra kattintással megnyitjuk azt a párbeszédablakot, amelyben elvégezhetjük a megjelenéssel kapcsolatos beállításokat.



2.109. ábra
Törés- és részletnézet

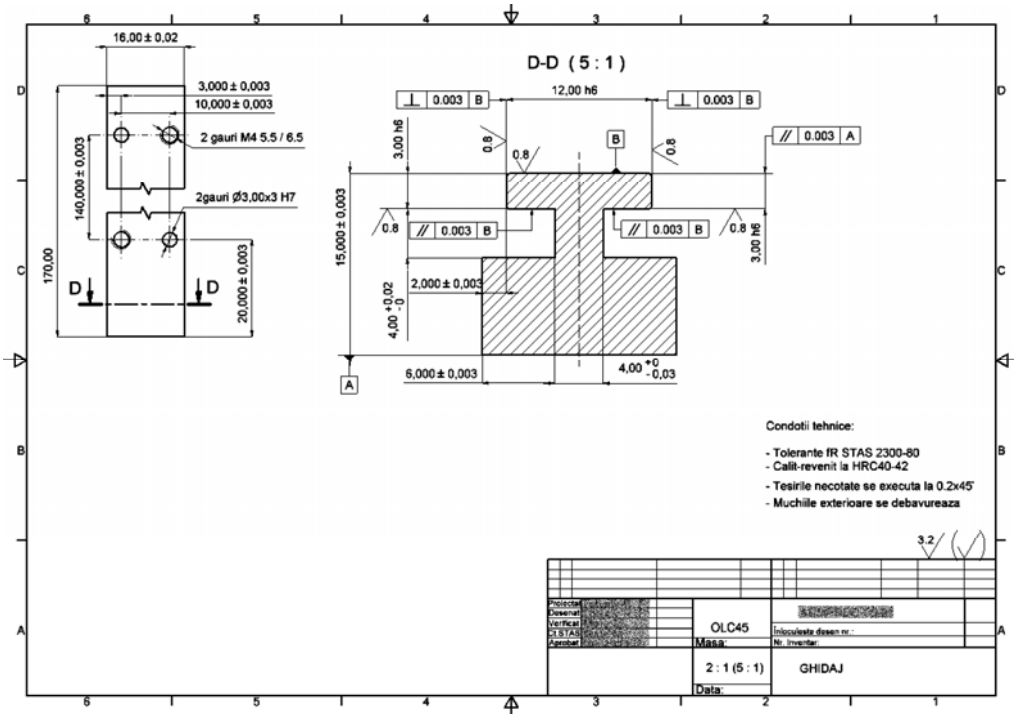
Ha egy rajz részleteit akarjuk kinagyítani, akkor a részletek rajzolására külön eszköz áll rendelkezésünkre az Inventor programban. A **Detail View (Részlet)** parancs párbeszédablakot nyit meg, amelyben szabályozhatjuk a megjelenést. A párbeszédablakban megadhatjuk a **Label** (Címke) betűjét és a méretarányt, valamint szabályozhatjuk ezek megjelenését. A jobb oldali részen, a Stílus (Style) címszó alatt meghatározható a részlet megjelenési módja. Az eszköz elindítása után ki kell jelölni azt a nézetet, amelyből származtatni akarjuk az új vetületet, majd a párbeszédablak megjelenése és a beállítások elvégzése után, körbe kell foglalnunk azt a részt, amelyről a részletet készíteni akarjuk. A kör megrajzolása után a részlet, az egérrel húzható a megfelelő helyre.

A **2.109. ábrán,** a **2.103. ábrán** látható darab tört nézete, és egy részletnézete látható. És végül a **2.110. ábrán** egy teljes szinten befejezett rajz látható, román rajzszabványt (STAS) követve.

Nyilvánvaló, hogy rajzaink az eddig ismertetett eljárások során még csak nézeteket tartalmaznak. A műszaki rajzban használt rajzjeleket, jelöléseket nem mutattuk be, és nem is a jelen anyag témája. Meg kell jegyeznünk, hogy minden rajzot tetszőleges szabvány szerint készíthetünk el, a létező szabványokat vagy akár saját készítésű szabványt használva. A rajzokat a szabványoknak megfelelően **minden** rajzelemmel elláthatjuk az Inventorban, beleértve méreteket, érdességi jeleket, szöveget stb. is.

Az elkészült rajzok, a test- vagy szerelési modellek alapján készültek. Ha a testmodelleket megváltoztatjuk, a rajzok is változnak ennek megfelelően, minden nézetben, metszetben, részletben vagy méretben.

Ezek átfogó bemutatása, valamint a szerelési modellezés vagy bemutató modellezés ismertetése sem a jelent kötet témája. Ezek bemutatása egy következő kötet, a *„Parametrikus modellezés”* témája lesz.



2.110. ábra
Egy kész műhelyrajz

Irodalom

1. *** *Autodesk Inventor 2009 Professional. Help and Tutorials.*
2. *** *Autodesk Inventor 2009. Getting Started*. Autodesk, Inc., 111 McInnis Parkwaz, San Rafael, CA 94903, USA, 2009.
3. Banach, Daniel T. – Jones, Travis – Kalameja, Alan J.: *Autodesk Inventor 6 Essentials with Autodesk Inventor 7 UPDATE*. Ed. Thompson Learning, 2003.
4. Banach, Daniel T.: Autodesk Inventor 2009 Essentials Plus, Autodesk Press, 2009.
5. Halbritter, Ernő: *AutoLISP – AutoLISP alkalmazások*. *Oktatási segédlet*. Szécshényi István Egyetem. Győr, 1996.
6. Harrington, David. – Burchard, Bill – Pitzer, David: *AutoCAD 2002*. Editura Teora, Bucureşti, 2002.
7. Head, George O.: *AutoLISP in Plain English*. Ventana Press, 1992.
8. http://cat2.mit.edu/4.564/tutorials/vlisp/
9. Ivan, Nicolae V. – Berce Petru – Drăgoi, Viorel ş.a., *Sisteme CAD/CAPP/CAM*. *Teorie şi practică*. Editura Technică, Bucureşti 2004.
10. Maczkó, István – Nagy György: *LISP, AutoLISP programozás AutoCAD-ben IBM PC-n*. Budapest., Alkalmazástechnikai Tanácsadó Szolgálat, 1989.
11. Manolea, Dan: *Programare în AutoLISP sub AutoCAD*. Ed. Albastră, Cluj-Napoca 1996.
12. Mocian, Ioan: *Proiectarea technologică asistată de calculator în construcția de maşini*. Editura Universității "Petru Maior", Tg. Mureş, 1999.
13. Pintér Miklós: *Autodesk Inventor 6. Tankönyv és példatár*. Budapest, Computer Books, 2003.
14. Pozdîrcă, Alexandru – Albert, Kálmán – Cheţan, Paul: *Inventor - Modelare parametrică*. Editura Unversității "Petru Maior" din Tg-Mureş, 2007.
15. Pozdîrcă, Alexandru – Mocian, Ioan Albert, Kálmán: *Programare în AutoLISP*. Editura Unversității "Petru Maior" din Tg-Mureş, 2001
16. Pozdîrcă, Alexandru: *AutoLISP. Bazele AutoLISP-ului*. Ed. Universității Techice din Tg. Mureş, 1994.
17. Stăncescu, Constantin – Manolache, Daniel S. – Pârvu Corneliu ş.a.: *Proiectare asistată cu Autodesk Inventor*. Editura FAST, Bucuresti, 2008.
18. Stăncescu, Constantin: *AutoLISP Manual Complet de Programare pentru AutoLISP*. Editura FAST, Bucureşti, 1996.
19. Tolvaly-Roşca, Ferenc: *Studiul preciziei angrenajelor conice prin metoda modelarii parametrice solide*. Teză de doctorat. Univ. "Transilvania" Braşov, 2006 (Doktori tézis).
20. Varga Tibor: *Az AutoCAD programozása. AutoLISP, ADS, R12-R13*. Computer Studio, Győr, 1996.
21. Varga, Tibor: *Autodesk Inventor 2008, 2009, ...* . Computer Studio, Győr, 2008.

The Basics of Computer Aided Design. Basics of AutoLISP Programming and Autodesk Inventor

Summary

This book is intending to offer a written support for Basics of Computer Aided Design discipline, present in the learning programs of mechatronics specialization of Sapientia University.
Presenting the knowledge's of English language software in Hungarian is a little bit unusual situation. The PC programs are widely traduced to Hungarian language, so using them is not a problem for a Hungarian mother tongue technical man. But the software's are traduced only in Hungary!

Based don this facts the, it is a little bit annoying that the book is intending to offer the basic knowledge's of programming in AutoLISP language (under Autodesk's AutoCAD), and the basics use of the parametrical modeling software Autodesk Inventor Professional 2009, in Hungarian language – using English terms from software, and Hungarian for principles and definitions.

The first caption of the book is presenting the base knowledge of AutoLISP programming language, one of the oldest, but the most used programming interface of the AutoCAD.

The second caption is presenting the solid modeling module, and the basics of the drawing module of Autodesk Inventor 2009.

The author does not intend to present an extensive use of these software's; they are matter of extensive practical use, but he want to transfer basic knowledge's in a short learning time. The book is presenting the theoretic knowledge's, trough a large number of examples, or practical application of the notions, and features. For understanding and using this book are necessary god knowledge's from Autodesk AutoCAD software.

Contents

Grundlagen des rechnerunterstütztes Entwerfen und Konstruieren
Elementare Begriffe AutoLISP und Autodesk Inventor

Zusammenfassung

Dieses Buch bemüht sich eine Unterstützung für die Mechatronik Fachgebiet im ungarischer Sprache der Sapientia Universität zu sein zu dem Fach *Basis der Rechnergestüzte Modellierung (CAD)*. Präsentation und Unterricht in Ungarisch von Programmen in englischer Sprache ist irgendwie ungewöhnlich, weil im allgemeinen die Programmen sind in Ungarisch übersetzt. Aber das ist gültig nur in Ungarn! Aus diesem Grund bemüht sich das Buch um das wichtigste Ziel: den Leser die notwendige Programmierung Kenntnisse in AutoLISP unter AutoCAD und die Nutzung der parametrischen Modellierung unter Autodesk Inventor 2009 Software zu lehren. Neben das, der Autor schlägt sich den Erwerb und die Fixierung die Konzepte in ungarischer Sprache der genannten Ziele vor.

Der erste Teil der Arbeit führt die Verwendung von AutoLISP Programmierung vor, an welche der Firma Autodesk möchte noch nicht verzichten und welche zwischen dem fortgeschrittenen Benutzern sehr beliebt ist.

Der zweite Teil zeigt den grundlegenden Ansatz der parametrischen Modellierung und ermöglicht einen kurzen Überblick an der Zeichnung herstellen in die Autodesk Inventor 2009.

Das Buch möchte nicht die präsentierte Themen ausführlich handeln (das hält meistens von Übung und Praxis ab, beziehungsweise von mühsame Arbeit für jedes Programm). Der Autor versucht, wie auch im seine Vorlesung, eine schellen Kenntnis Transfer und Aneignung dieser Kenntnisse durch häufige Beispiele. Damit die grundlegenden theoretischen Konzepte werden durch praktische Beispiele dargestellt.

Dieses Buch ist nicht nur ein Hilfsmittel für Studenten, sondern auch für Ingenieure, die zur Vertiefung ihrer Kenntnisse im Bereich der *Rechnergestüzte Modellierung* bereit sind.

Inhalt

Bazele proiectării asistate de calculator
Noțiuni de bază AutoLISP şi Autodesk Inventor

Rezumat

Cartea ținută în mână de cititor, se străduieşte să ofere un suport scris pentru disciplina Bazele proiectării asistate de calculator (CAD), materie prezentă în programa secției de mecatronică a Universității Sapientia, cu predare în limba maghiară. Prezentarea şi predarea în limba maghiară a unor programe ce rulează în limba engleză, este oarecum neobişnuită, ştiind ca softurile în general sunt traduse în limba maghiară. Dar acest lucru este valabil doar în Ungaria! Datorită acestui fapt, cartea se străduieşte în primul rând să-şi asigure obiectivul principal: predarea cunoştințelor necesare programării în AutoLISP sub AutoCAD şi utilizării la nivel de bază a programului de modelare parametrică Autodesk Inventor 2009. Autorul, pe lângă acesta, îşi propune însuşirea şi fixarea de către cititor, a noțiunilor şi cunoştințelor legate de cele două obiective în limba maghiară.

Prima parte a lucrării prezintă, utilizarea mediului de programare AutoLISP, care rămâne totuşi, limbajul de programare la care nu renunță Autodesk, şi este extrem de popular printre utilizatorii de nivel avansat al programului AutoCAD.

Partea a doua, prezintă abordarea de bază a modelării parametrice solide, şi o scurtă prezentare a modulului realizare a desenelor în Autodesk Inventor 2009.

Lucrarea nu-şi propune o abordare aprofundata a aspectelor prezentate (ele ținând mai ales de exersare şi practică, şi de lucrări laborioase specifice fiecărui program). Autorul încearcă realizarea, ca şi la cursurile proprii, a unui transfer şi a unei modalități de asimilare rapidă a cunoştințelor legate de CAD, prin exemplificări practice frecvente. Astfel noțiunile teoretice de bază sunt prezentate prin exemple practice, punctându-se pe alocuri şi unele aspecte de detaliu.

Prezenta carte este un instrument didactic în primul rând, dar nu doar pentru studenți, ci şi pentru inginerii care vor să-şi aprofundeze cunoştințele legate de domeniul proiectării asistate de calculator.

Cuprins

A sorozat eddig megjelent kötetei:

1. Jodál Endre: *Számítástechnika az ezredforduló küszöbén*. 1992. 35 oldal
2. Pálfalvi Attila: *Porkohászat*. 1993. 39 oldal
3. Bagyinszki Gyula – Bitay Enikő: *Bevezetés az anyagtechnológiák informatikájába*. 2007. 213 oldal
4. Bitay Enikő: *Lézeres felületkezelés és modellezés*. 2007. 174 oldal
5. Bagyinszki Gyula – Bitay Enikő: *Felületkezelés*. 2009. 359 oldal
6. Forgó Zoltán: *Bevezetés a mechatronikába*. 2009. 200 oldal

A könyv, amelyet az olvasó a kezében tart, a Sapientia Magyar Tudományegyetem mechatronika szakán oktatott Számítógépes tervezés tantárgy előadásanyagát igyekszik elsősorban támogatni. Ugyanakkor bizalommal forgathatják, azok a legkülönbözőbb szakosítású mérnök szakemberek is, akik ismereteiket szeretnék bővíteni a számítógépes tervezés ezen a két sajátságos területén. A bemutatott anyag megértéséhez nélkülözhetetlen az AutoCAD középszintű ismerete.

Az első fejezet AutoLISP programozás alapjait ismerteti, a második fejezet pedig, a parametrikus modellezés alapfogalmait mutatja be Autodesk Inventor környezetben. A szerző gyakorlati példákon keresztül igyekszik a legrövidebb úton a legtöbb elméleti ismeretanyagot átadni az olvasónak.

ISBN 978-973-8231-81-8