A matematika alkalmazásával foglalkozó emberek régi álma egy olyan intelligens számológép, amely nem csak számokat, hanem képleteket és formulákat is képes kezelni. A számítástechnikai eszközök fejlődésével egyre bonyolultabb programozási feladatok megoldása válik lehetővé. Napjainkra az intelligens számológép álma realitássá vált. Ez egy új korszak kezdetét jelenti a matematika és a számítógép felhasználásában. A szimbolikus számítások egyik eszköze a Maple V számítógépes algebrai rendszer.

A számítógépes algebrai rendszereknek nevezett interaktív programok felhasználói a számokon kívül szimbólumokkal, formulákkal, egyenletekkel stb. is dolgozhatnak. Sok matematikai feladat, így például a differenciálás, integrálás, függvények sorfejtése, szimbolikus elemeket tartalmazó mátrixok invertálása különösebb emberi erőfeszítés nélkül gyorsan, nagy pontossággal megoldható. Hathatós segédeszközei a matematikusoknak, fizikusoknak, mérnököknek – vagyis mindenkinek, aki matematikai számításokat végez. A számítógépes algebrai rendszerek nélkülözhetetlenek a modern elméleti és alkalmazott tudományos kutatásokban, valamint az oktatásban.

A Maple egy komputer algebrai nyelv, azaz a korábban kifejlesztett (pl.: Pascal, C stb.) algoritmikus nyelvekkel ellentétben a matematikai műveleteket nem csak numerikusan és lebegőpontos aritmetikával, hanem ahol ez lehetséges pontosan illetve formális változókkal is lehet végezni. Egy változó a program futása során többféle szerepet is játszhat. Lehet: formális változó, ami felvehet különböző numerikus és nem numerikus értékeket, majd újra vissza lehet alakítani formális változóvá. A formális változók megengedése matematikai szempontból óriási bővítést jelent, mert a formális változó elvileg már bármilyen matematikai objektum lehet: halmaz, függvény, vektor, mátrix, operátor, kifejezés, grafikon, stb. Ezeknek a rendszereknek a flexibilitása azt is lehetővé teszi, hogy a különböző megengedett matematikai objektumok közötti műveleteket is definiáljuk és így szinte tetszőleges matematikai struktúrára megtaníthatjuk a rendszert. Ezeket a lehetőségeket csak a rendszer használatával lehet igazán felmérni, és jelentőségét megérteni.

Ismerkedjünk meg röviden Maple V-tel. A rendszerrel való kommunikáció utasítások kiadásával történik. Az utasítások szerkezete: az utasítás vagy Maple-beli függvény neve, utána `()` alakú zárójelben az utasítások vagy függvények paraméterei (a paraméterek jelentése a függvények leírásában megtalálható), majd az utasítást `;` (pontosvessző) jellel kell lezárni. (Ha nem akarjuk, hogy a rendszer a képernyőre írva válaszoljon, használjuk a `:` jelet a lezárásra.) A továbbiakban tekintsünk néhány példát. A

```
> print('Szia!');
```

kiírja a képernyőre az `' '` jelek közé írt szöveges információt. Jelen esetben a válasz:
Egy egész szám prím felbontását adó utasítás:

```
> ifactor(2520);
```

$$(2)^3(3)^2(5)(7)$$

A $\sum_{i=1}^{n} i^3$összegképlet kiszámítását a következő utasítással számíttathatjuk ki:

```
> sum(i^3,i=1..n);
```

$$\frac{1}{4}(n+1)^4 - \frac{1}{2}(n+1)^3 + \frac{1}{4}(n+1)^2$$

A Maple-rendszer aritmetikai kifejezéseket is elfogad utasításként. Az aritmetikai kifejezéseket a más programnyelvekben is megszokott formában és szintaktikával kell megadnunk.

```
> (x+1)^4*(x+2)^2;
```

$$(x+1)^4(x+2)^2$$

vagy

```
> (x+y)^3-(x-2)^2;
```

$$(x+y)^3 - (x-2)^2$$

A megelőző utasítás értékére a % jellel hivatkozhatunk.

```
> expand(%);
```

$$x^3 + 3x^2y + 3xy^2 + y^3 - x^2 + 4x - 4$$

Az újra szorzattá alakítás a `factor()` függvény segítségével tehető meg.

A Maple-rendszer számos lehetőséget nyújt egyenletek és egyenletrendszerek megoldására, beleértve a lineáris és nem lineáris egyenleteket, egyenletrendszereket és differenciálegyenleteket is. A megoldásra a `solve()` függvény használható.

```
> e1:=3*x+2*y+3*z=110; e2:=5*x+y-4*z=0; e3:=2*x-3*y+z=0;
```

$$e_1 := 3x + 2y + 3z = 110;$$
$$e_2 := 5x + y - 4z = 0;$$
$$e_3 := 2x - 3y + z = 0;$$

```
> solve({e1,e2,e3},{x,y,z});
```

$$\{x = 11,\ y = 13,\ z = 17\}$$

A megoldás során használtuk az értékadó utasítást, melynek alakja:

$$\text{változónév} := \text{kifejezés};.$$

Itt hívnánk fel a figyelmet, hogy **a Maple-rendszer megkülönbözteti a kis- és nagybetűket.**

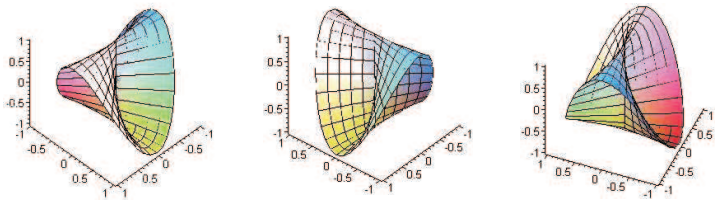
A Maple-verziókhoz részletes angol nyelvű Help program tartozik. A Help használatához a `?` karakter mint utasítás segítségével, vagy az F1 funkcióbillentyűvel kaphatunk eligazítást. Speciálisan egy Maple-függvényről úgy kaphatunk leírást, ha a parancs (vagy függvény) nevét a `?` jel után beírjuk, pl.: `?plot3d;` utasítás után részletes leírást kapunk a háromdimenziós grafikus ábrázolásról.

Ha már említettük a grafikus ábrázolást, nézzük meg milyen lehetőségeket nyújt számunkra a Maple.

A Maple-rendszer egyik leglátványosabb lehetősége a grafikus megjelenítés; ez az egyszerű függvények megjelenítésétől a háromdimenziós ábrák animációjáig, (mozgásban való megjelenítésig) terjed. Speciális grafikus feladatok megoldására külön programcsomag áll rendelkezésre, a plots, amelyben számos grafikus lehetőséget tartalmazó utasítás található. Itt jegyeznénk meg, hogy a rendszerhez különböző programcsomagok tartoznak, melyeket a `with()`, pl.: `with(plots);` utasítással hívhatunk meg, ezáltal bővítve a rendszerben használható függvények számát.

Egy függvény kétdimenziós grafikonját szolgáltató Maple-beli függvény a `plot()`, melynek használata nagyon egyszerű. Az utasítás többféle paraméterezéssel is megadható, ezek kipróbálását a kedves olvasóra bízzuk. Az egyik legizgalmasabb lehetőség, mely csak számítógépekkel valósítható meg a nagy számítási igény miatt, az implicit alakban adott függvények megjelenítése. Erre az `implicitplot()` függvény használható. A háromdimenziós (interaktív) ábrákat a `plot3d()` függvény segítségével állíthatunk elő. Lehetőségünk van paraméteres alakban megadott függvények, térgörbék, felületek megjelenítésére is.

```
> plot3d ([r*cos(t),r*sin(t),cos(2*t)], t=0..2*Pi, r=0..1,
          grid=[45,6], scaling=constrained, axes=frame);
```



Mint említettük a Maple képes animáció készítésre is. Az ábrára kattintva, megjelenik egy gombsor, mely segítségével az animáció lejátszható, és egyéb beállításokat is megadhatunk.

```
> animate( { [cos(u)+k, 1+sin(u), u=0..2*Pi],
  [k*t-sin(k*t),1-cos(k*t),t=0..1],[k-sin(k)+cos(v)/10,
  1-cos(k)+sin(v)/10,v=0..2*Pi]},
  k=0..4*Pi, scaling=constrained);
```

A Maple nagy erénye – amellett, hogy rengeteg beépített függvényt tartalmaz, melyek listája verzióról verzióra bővül – az, hogy programnyelvként is használható. Így a felhasználó maga is írhat programokat, de újabb függvényeket is definiálhat, melyeket aztán ugyanúgy használhat, mintha azok „beépített” függvények lennének. A Maple programozási eszközei nagyon egyszerűek, és könnyen elsajátíthatók. A Maple adatstruktúrájának áttekintése és az adatok kezelésének elsajátítása azonban sokkal több munkát kíván meg, így erre most nem térnénk ki. A programozás elemeivel könnyebben megismerkedhetünk, nézzünk egy gyors összefoglalást.

A programozás eszközei a ciklusszervezés, a feltételvizsgálaton alapuló végrehajtás-ütemezés és az eljárások definiálása. Vegyük sorra ezeket:

1. Ciklusszervezés
   `for` *név* `from` *kifejezés* `by` *kifejezés* `to` *kifejezés* `while` *kifejezés*
   `do`
   a ciklusban végrehajtandó utasítások sorozata
   `od`
   ahol a *név* a ciklusváltozó nevét jelenti, a from utáni *kifejezés* a ciklusváltozó kezdőértékét adja meg, a by utáni *kifejezés* a ciklusváltozó lépésenkénti változásának mértéke, a to utáni *kifejezés* a ciklusváltozó utolsó értékét definiálja, a while-t követő *kifejezés* egy feltételt fogalmaz meg, melynek teljesülése esetén a ciklusban végrehajtandó utasítások sorozata végrehatódik. A fenti forma a legteljesebb, belőle sok rész elhagyható.

2. Feltételvizsgálat
   `if` *logikai kifejezés*
   `then` *a végrehajtandó utasítások*
   `else` *a végrehajtandó utasítások*
   `fi`
   Ha több feltételt szeretnénk vizsgálni, azt a következőképpen tehetjük meg.

`if` *logikai kifejezés*
`then` *a végrehajtandó utasítások*
`elif` *logikai kifejezés*
`then` *a végrehajtandó utasítások*
`else` *a végrehajtandó utasítások*
`fi`

A feltételrendszer a logikai kifejezésekre érvényes művelettábla szerint értékelődik ki. Ha az if utáni logikai kifejezés értéke igaz, akkor az őt követő then utáni végrehajtandó utasítás(ok) lesznek végrehajtva, ellenkező esetben a rendszer áttér a következő feltétel vizsgálatára, vagy eljut a feltételvizsgálatok befejezését jelentő fi kulcsszóhoz.

3. Eljárás (procedúra)
eljárás `:= proc(`*argumentumok*`)`
`local` *változók*
utasítások
`RETURN()`
`end;`

Minden Maple eljárás a `proc(`*argumentumok*`)` formulával, ún. eljárásfejjel kezdődik, és az end kulcsszóval végződik. A visszatérési értéket a RETURN utasítással definiáljuk, és ezzel is lépünk ki az eljárásból.

Néhány egyszerű példa:

a.
```
>negyzet:=proc(x)
>RETURN(x*x)
>end;
negyzet := proc(x) RETURN(x^2) end
>negyzet(3);
9
```

b.
```
>fakt:=proc(n)
local i, f;
f:=1;
for i from 1 to n
do
f:=f*i
od;
RETURN(f)
end;
>fakt(6);
720
```

A Maple széleskörű matematikai és programozási szolgáltatásainak megismeréséhez, és a rendszer tökéletes használatának elsajátításához természetesen további ismeretek szükségesek.

Az Interneten megtalálható letölthető változatai a programnak:

Maple 7 Trial Version:
`http://www.maplesoft.com/trial.shtml` (30 napos próbaverzió)

Maple 3.2 (3,1 MB):
`http://tucows.externet.hu/business/preview/197875.html`

Felhasznált irodalom

André Heck: Bevezetés a Maple használatába.
Molnárka – Gergó – Wettl – Horváth – Kallós: A Maple V és alkalmazásai.