Uvod u rad i programiranje na HP Prime kalkulatoru

Kožar, Ivica

Ivica Kožar

UVOD U RAD I PROGRAMIRANJE NA HP PRIME KALKULATORU

uniri

UNIVERSITAS STUDIORUM FLUMINENSIS
SVEUČILIŠTE U RIJECI

Udžbenici Sveučilišta u Rijeci
Manualia Universitatis studiorum Fluminenis

Izdavač
Sveučilište u Rijeci
Građevinski fakultet u Rijeci

Autor
Ivica Kožar

Urednik
Sanja Dugonjić Jovančević

Recenzenti
Dražan Kozak
Neira Torić Malić
Boris Podobnik

Lektura
Saša Čohar Mančić

Grafička priprema
Zlatko Majetić

Ilustracija na naslovnici
Ivica Kožar

Tisak
3 Dreams d.o.o.

ISBN 978-953-6953-54-7 (tiskano izdanje)

Prvo izdanje

CIP zapis dostupan u računalnom katalogu Sveučilišne knjižnice Rijeka pod brojem 150203048.

Odlukom Senata Sveučilišta u Rijeci (KLASA: 003-01/21-03/02, URBROJ: 2170-57-01-21-370, od 19. listopada 2021. godine) ovo se djelo objavljuje kao izdanje Sveučilišta u Rijeci.

Ivica Kožar

Uvod u rad i programiranje na HP Prime kalkulatoru

Prvo izdanje

uniri

Rijeka, 2021.

G F Sveučilište u Rijeci **Građevinski fakultet**

Zahvala

Zahvaljujem svima koji su pomogli izlazak ove knjige, u prvom redu mojoj supruzi Danili za pažljivo čitanje prvih verzija teksta i sugestije za poboljšanje jasnoće izlaganja. Zahvaljujem se recenzentima, kolegici Neiri Torić Malić, kolegama Dražanu Kozaku i Borisu Podobniku na korisnim savjetima. Zahvaljujem se urednici Sanji Dugonjić Jovančević na velikom trudu uloženom u izdavanje i tisak ove knjige.

Sadržaj

2 DIO - PROGRAMIRANJE

„U početku bijaše Šiber“

Predgovor

Zašto sam za moto knjige osmislio izjavu „U početku bijaše Šiber“?

Knjiga je prvenstveno namijenjena studentima inženjerskih studija (u starinskom smislu riječi) i predstavlja moju želju da vratim računanje u inženjersku edukaciju. Naravno da je „šiber“ (logaritamsko računalo) stariji od pojam od „inženjera“ (izumiteljem logaritamskog računala, engl. „slide-rule“, smatra se William Oughterd, 1574. – 1660.). „Šiber“ je do 70-ih godina bio nezaobilazni pratilac inženjera koji su njime računali sve što im je trebalo, a kasnije su njegovo mjesto zauzeli kalkulatori. Danas inženjeri malo računaju, uglavnom se koriste propisima i gotovim računalnim programima, a inženjerima se znanja na kojima se oni baziraju prenose uglavnom deklarativno („nauči propis“, „nauči formulu“). Mišljenja sam da je računanje nužno u izobrazbi inženjera i da bitno doprinosi razumijevanju inženjerskih problema i načina njihovog rješavanja. Može se reći da je knjiga pokušaj da se „kultura računanja“ vrati inženjerima, da se sačuva ideja o važnosti kvantifikacije podataka/znanja (npr., kvalitativno znanje jest „ljeta su toplija nego nekada“, a kvantitativno „srednja ljetna temperatura je za 2° veća“). Primjena kvalitativnog znanja je ograničena, a kvalitativno znanje se može uvrstiti u inženjerske modele i načiniti predviđanje koje može dati puno točniji prikaz onoga što nas zanima. Uostalom, odavno je poznata izreka „Misliti je d..k znati!“. Uz računanje, mišljenje postaje argument.

Knjiga se temelji na mom iskustvu upotrebe programabilnog kalkulatora kao studenta i, kasnije, kao asistenta na fakultetu. Naime, moj prvi programabilni kalkulator bio je HP 67, koji sam kupio u ljeto 1979, nakon prve godine studija na Građevinskom fakultetu. Mišljenja sam da je kalkulator HP Prime u potpunosti dovoljan za savladavanje cjelokupnog gradiva preddiplomskog studija, a zadovoljava i većinu potreba na diplomskom studiju (iznimka je upotreba računalnih programa propisanih na pojedinim predmetima jer oni najčešće zahtijevaju računalo s Windows operativnim sustavom). Materijal iznesen u knjizi uglavnom je prezentiran studentima prve godine Građevinskog fakulteta tijekom pet godina moje nastave na predmetu “Uvod u programiranje”.

Prvi dio knjige opisuje upotrebu HP Prime kalkulatora u rješavanju inženjerskih problema, rad s raznim vrstama varijabli (liste, vektori, matrice) te rad s nekim aplikacijama koje dolaze s HP Prime kalkulatorom (funkcije, grafika, statistika). Pretpostavka je da će studenti (pažljivo) pročitati osnovne upute koje dolaze uz

HP Prime („Quick start“, 50-ak stranica teksta) pa se tamo izneseni podaci ne ponavljaju u knjizi.

Drugi dio knjige bavi se uvodom u programiranje. Koristeći primjere opisuju se osnovni elementi programiranja (unos podataka, ispis rezultata, grananje, petlje). Prikazan je i uvod u optimizaciju navodeći probleme u kojem treba maksimizirati jednu, a minimizirati drugu varijablu (odabir predmeta maksimalne vrijednosti i minimalne mase). Rješenje ovog, u biti teškog problema, prikazano je uz pomoć „greedy“ algoritma, uz naznaku mogućih varijanti. U nastavku su navedeni primjeri s grafikom te se tako prikazuje crtanje linija iz zadanih koordinata. Sljedeći je primjer izračun reakcija i reznih sila proste grede, gdje se prikazuju različiti načini interakcije s korisnikom i grafički prikaz rezultata.

Treći dio ili dodatak općenito opisuje primjenu kalkulatora u inženjerstvu, odnosno neke manje uobičajene načine zapisivanja matematičkih operacija i njihovu primjenu na HP kalkulatorima. HP kalkulatori su tradicionalni koristili takav način zapisa (RPN) koji nije koristio zagrade ni tipku jednako. Ilustracija upotrebe i programiranja prikazana je na programabilnom kalkulatoru HP 67 te taj dio teksta predstavlja neobavezno štivo. U nastavku su napomene o pisanju teksta na smartphone uređajima i prebacivanju podataka iz/u aplikaciju HP Prime.

Ivica Kožar
Rijeka, 2021.

Popis pokrata

HP (Hewlett Packard) - tvrtka za proizvodnju kalkulatora i računala

AP (Advanced Placement) - curricula za preddiplomsku razinu studija i završni ispiti srednjih škola u SAD-u

SAT (Scholastic Aptitude Test) - prijemni ispit za upis na preddiplomske studije u SAD-u]

RPN (Reverse Polish Notation) – inverzna poljska notacija

STACK - stack je naziv takve pomične memorije gdje možemo vidjeti samo zadnji uneseni podatak

ASCII (American Standard Code for Information Interchange) – američki standard brojčanog označavanja znakova

WYSIWYG (What You See Is What You Get) – prikaz na ekranu računala koji je istovjetan s prikazom na pisaču

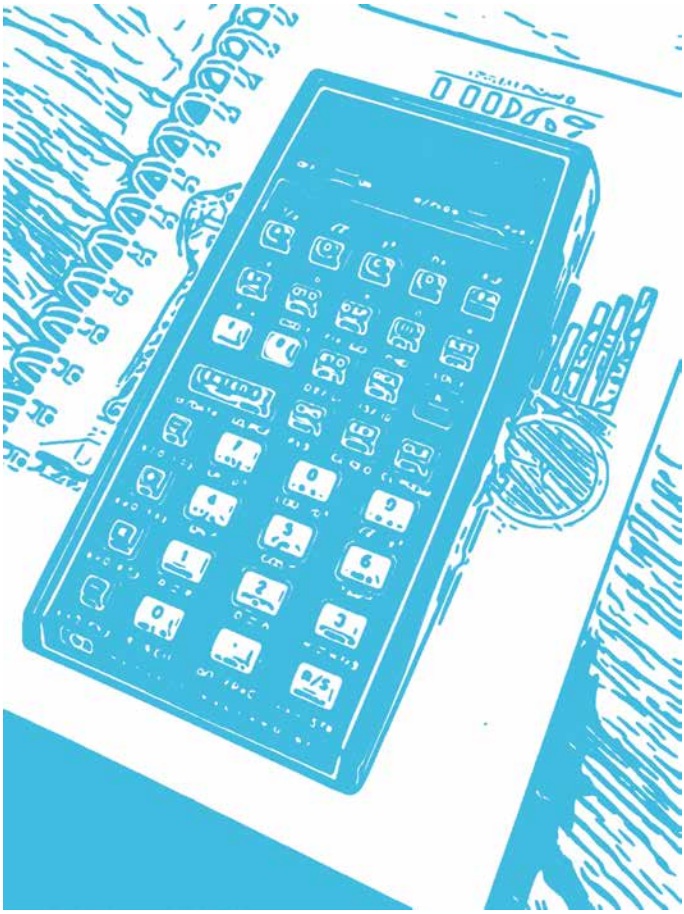
PDF (Portable Document Format) - jezik za prikaz teksta i slika neovisan o software-u, hardware-u i operacijskom sustavu)

RTF (Rich Text Format) – način kodiranja znakova razvijen od strane Microsoft Corporation za potrebe formatiranja teksta

HTML (Hypertext Markup Language) - jezik Web stranica čije naredbe preglednik (Web browser) pretvara u formatirani tekst i slike vidljive na ekranu

XML (eXtensible Markup Language) - jezik za spremanje i prebacivanje podataka na Internetu, načinjen da bude čitljiv i ljudima (i računalima)

SVG (Scalable Vector Graphics) – način zapisa grafičkih naredbi u tekstualnom obliku



Kalkulator HP-67 iz 1976. godine.

1 DIO – RAD S KALKULATOROM

HP Prime - pregled značajki

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz HP Prime Quick Start Guide (iz menija „Help“) :

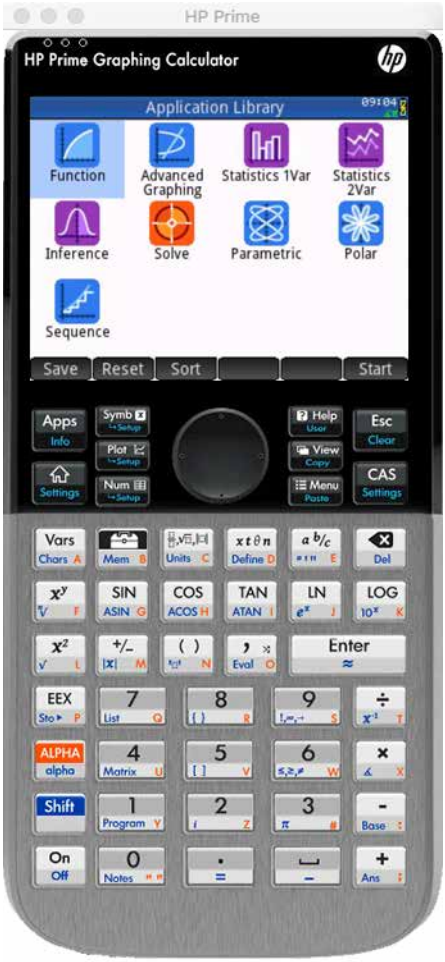
- *Keyborad legend*
- *Getting started*
- *HP apps and their views*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

```
nema pretpostavki o stanju varijabli u memoriji
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite pomoću 'Edit: Paste' naredbi iz menija kalkulatora pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

HP Prime je edukacijski kalkulator s mogućnošću programiranja (službeni naziv je *grafički kalkulator*), posebno pogodan za poduku u STEM (Science Technology Engineering Mahematics) području. Naziv *kalkulator* prihvatio bih uvjetno jer je hardware kalkulatora jači od stolnih računala prije 20-ak godina (procesor ARM9 na 400 MHz, 32 MB centralne memorije, 256 MB memorije za pohranu) te bi možda prikladniji naziv bilo *ručno mikroračunalo*.



Izgled virtualnog HP Prime kalkulatora (lijevo za macOS, desno besplatna verzija za Android). Primjetna je vrlo velika sličnost sa stvarnim uređajem.

Moguće je raditi na stvarnom kalkulatoru, ali je isto tako moguće kalkulator instalirati kao virtualnu aplikaciju na smartphone, tablet ili stolno računalo. Virtualna aplikacija je u primjeni u skoro svemu identična radu na stvarnom uređaju (nije podržano spajanje vanjskih senzora). Podržana je većina operacijskih sustava: iOS, Android, macOS, Windows. Verzija za mobilne operativne sustave se plaća (postoji besplatna inačica smanjenih mogućnosti za Android sustav), a verzije za stolna računala su besplatne.

Instaliranje

Instalacija programa za Android sučelje vrši se putem Google Play (Trgovina Play) web aplikacije. Za Apple iOS sučelje instalacija se vrši putem App Store aplikacije.

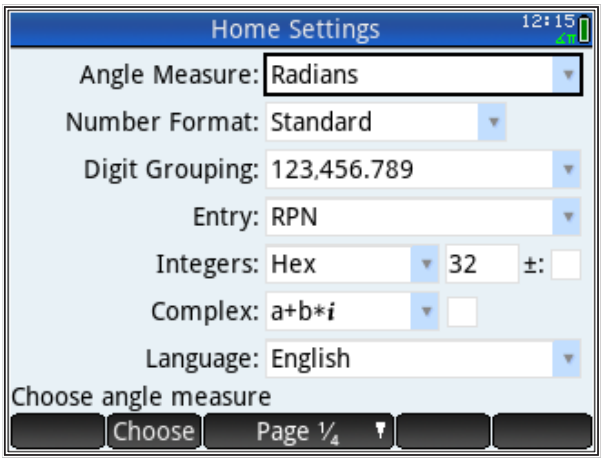
Pregled mogućnosti

HP Prime predviđen je za upotrebu u inženjerstvu, računarstvu, geodeziji, biologiji, kemiji i fizici s aplikacijama za trigonometriju, statistiku i geometriju. HP Prime je dozvoljeno koristiti na AP[1] ispitima integralnog i diferencijalnog računa, statistike, kemije i fizike, na SAT[2] ispitima iz matematike i drugima.

Postavke kalkulatora

Postavke se namještaju u Settings (tipke *Shift* *ENTER*), meniju koji ima 3 ili 4 stranice, ovisno o verziji i računalu. Na dnu slike se vidi koja je stranica prikazana; za 1. stranicu piše 1/4. Meni služi za namještanje mjere kuta, jezika, formata unosa i prikaza brojeva, datuma i vremena i drugih postavki.

1 Advanced Placement, curricula za preddiplomsku razinu studija i završne ispite srednjih škola u SAD-u
2 Scholastic Aptitude Test, prijemni ispit za upis na preddiplomske studije u SAD-u



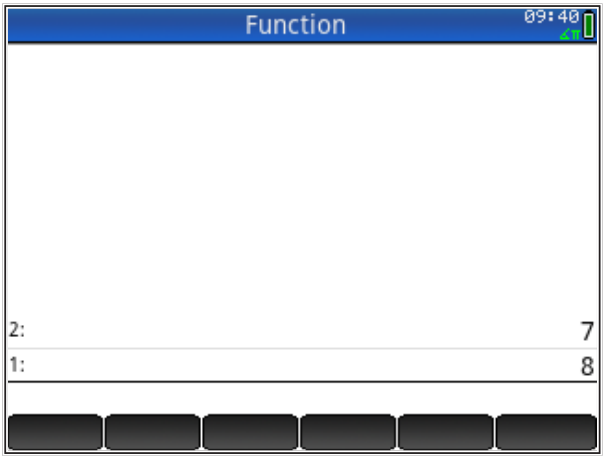
Tu se može namjestiti i RPN[3] format unosa brojeva; tada računamo bez zagrada i znaka jednako (=), npr., umjesto

```
7 + 8 =
```

tipkamo

```
7 ENTER 8 +
```

i rezultat na ekranu jest 15 (tijekom računanja na ekranu smo mogli vidjeti utipkane brojeve kako se pomiču u memoriji kalkulatora[4]). Prije operacije '+', ekran izgleda ovako:



3 Reverse Polish Notation

4 Stack je naziv takve pomične memorije gdje možemo vidjeti samo zadnji uneseni podatak

Mogući su i kompliciraniji izračuni

```
7 + 2*(1-3) =
```

Ako je aktivna RPN notacija, tipkamo

```
1 ENTER 3 - 2 * 7 +
```

i rezultat na ekranu jest 3. Redoslijed operacija u RPN notaciji izravno ovisi o položaju zagrada u algebarskoj notaciji. Tijekom tipkanja mogli smo vidjeti međurezultate kako se izmjenjuju na stack-u. U edukativne svrhe možemo izračun u RPN modu napisati i tako da prvo unesemo sve podatke (brojeve), a onda operacije koje na tim podacima želimo izvršiti.

```
7 ENTER 2 ENTER 1 ENTER 3 - * +
```

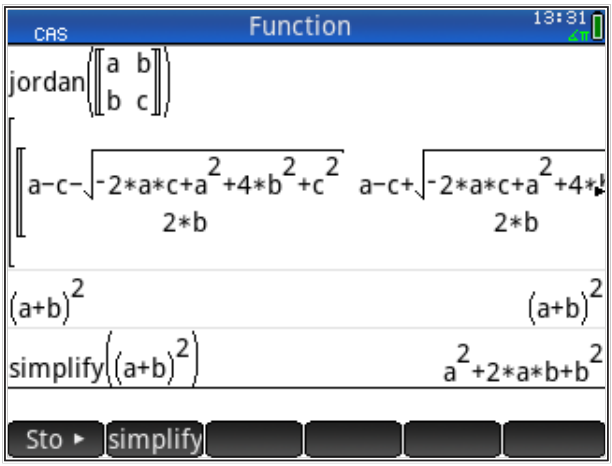
Kao što vidimo na ekranu, ovaj način zahtjeva puno veći stack (više memorije) i obično se ne koristi. Ukoliko je moguće, preferira se prvi način unosa, gdje se miješaju podaci i operacije nad njima; minimalna potreba memorije osigurava se ako se krene „iznutra“, od najniže razine zagrade.

U RPN načinu se, umjesto varijabli, koristi 'STACK', koji je praktički neograničene veličine i iz kojeg se mogu „izvući“ vrijednosti s bilo koje pozicije (unutar '*STACK'*-a).

Poznavanje RPN notacije je dobar uvod u rad s listama, koje su danas često korišten način zapisivanja podataka ('*STACK'* je zapravo lista podataka).

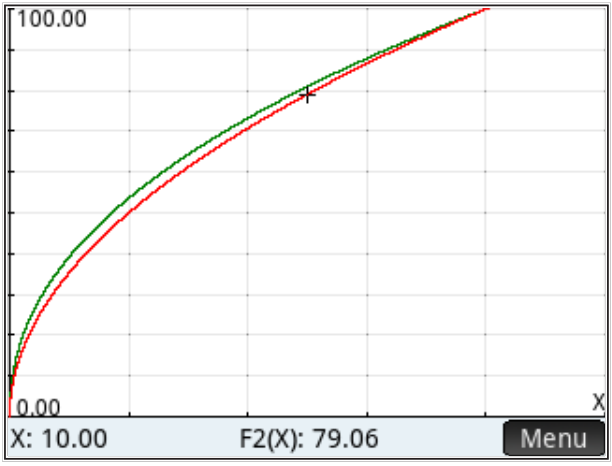
CAS način

CAS je akronim za Computer Algebra System (računalni algebarski program), a omogućuje simboličko računanje. U CAS način ulazimo pritiskom na tipku *CAS*, pri čemu se otvara novi ekran i u lijevom gornjem kutu ekrana piše CAS. Sada alfanumerički upis radi s malim slovima i računi su simbolički, npr., $(a+b)^2$ daje $a^2+2ab+c^2$ (ali samo nakon što na meniju odaberemo 'simplify').



Simboličko tretiranje izraza vrlo je korisno; u nastavku je inženjerski primjer crtanja dvaju nelinearnih funkcija.

Primjer: Fullerova krivulja prolaska agregata kroz sito glasi $p = (\frac{d}{D})^{\frac{1}{2}}$. Tu je formulu modificirala FHWA (Federal Highway Administration) tako da je eksponent s 0,5 promijenjen na 0,45 (a). Obje formule je lako istovremeno prikazati grafički ('d' neka je na apscisi, a 'p' na ordinati).

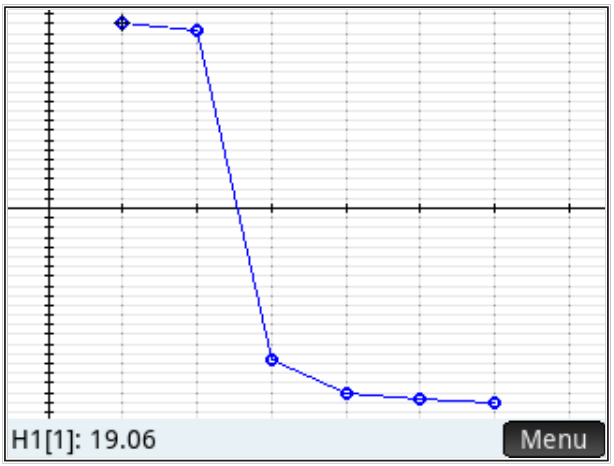


Pregled aplikacija

Besplatna verzija HP Prime nema sve aplikacije kao i stvarni kalkulator (i virtualni koji se plaća). Dostupne aplikacije su vidljive na slici besplatnog virtualnog HP Prime kalkulatora (koji nema mogućnost programiranja):

1. Function - aplikacija za zadavanje, izračun i grafički prikaz eksplicitnih funkcija jedne varijable
2. Advanced Graphing - aplikacija za zadavanje, izračun i grafički prikaz implicitnih funkcija dvije varijable; funkcije mogu biti zadane kao jednakosti ili kao nejednakosti
3. Statistics 1Var - aplikacija za zadavanje niza brojeva kojem se onda mogu izračunati statističke vrijednosti (srednja vrijednost, varijanca, itd.) i koji se (niz) može grafički prikazati na razne načine; ta se aplikacija može koristiti i za grafički prikaz brojeva u listi
4. Statistics 2Var - aplikacije za unos podataka i statistiku 2 varijable, izračun statističkih vrijednosti i grafički prikaz (između ostalog, može izračunati i grafički prikazati regresijsku krivulju zadanih podataka)
5. Inference - testiranje intervala pouzdanosti, x^2 test, ANOVA analiza; podaci se mogu preuzeti iz statistike 1 ili statistike 2 varijable
6. Solve - zadavanje i rješavanje jednadžbi, linearnih i nelinearnih (kako ćemo vidjeti kasnije; to nije klasični rješavač sistema jednadžbi, nego se više radi o odnosima pojedinih varijabli)
7. Parametric - crtanje parametarski zadanih funkcija $X(T)$ i $Y(T)$; crta se u ravnini a parametar je T
8. Polar - crtanje u polarnim koordinatama funkcije $R(\theta)$
9. Sequence - analiza i grafički prikaz rekurzivno zadanih diskretnih funkcija (serija); zadaje se trenutna vrijednost kao funkcija vrijednosti u nekim prethodnim trenucima, $U(N)=f(U(N-1))+g(U(N-2))$.

Kao primjer načinjene aplikacije prikazujem program za izračun temperature slojeva u fizici zgrade (kako se takav dijagram izračunava, uči se na predmetu *Fizika zgrade*). Grafički prikaz rezultata daje

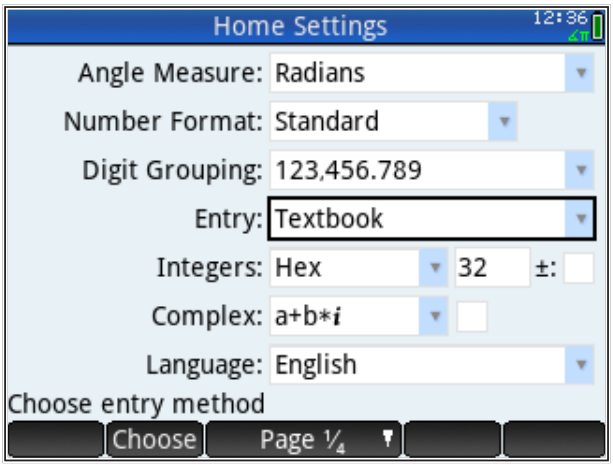


Profesionalna (potpuna) verzija HP Prime kalkulatora ima i dodatne funkcije, a kao najznačajnije spomenimo samo sljedeće:

1. Geometry - kombiniranje i crtanje geometrijskih elemenata: točka, linija, krug, poligon, tangenta
2. Spreadsheet - tablični kalkulator, u mnogo čemu sličan tabličnim kalkulatorima na osobnim računalima, potpuno programabilan (npr., u njemu je načinjen program za računanje proizvoljnih ravninskih rešetki).

Rad s memorijom i varijablama

U RPN načinu se, umjesto varijabli, koristi 'STACK', a za rad s memorijom i varijablama isključujemo RPN način jer u RPN načinu nije moguće zadavati varijable (ali je moguće koristiti prethodno zadane varijable iz 'Textbook' načina). Prelazimo u 'Home' način (tipka 'kućica') i uključujemo 'Textbook' način iz menija 'Settings'.



Zadavanje varijabli

Prvo definirajmo pojam varijable kakav se koristi u radu s HP Prime. Varijable na HP Prime lakše opisujemo velikim slovima. Za varijable možemo koristiti i mala slova, ali velika zahtijevaju manje tipkanja, a i ostale aplikacije HP Prime-a očekuju varijable u velikim slovima. Varijable zadajemo kao,

```
A:=5.5
```

Da bi dobili ovu naredbu, tipkamo

```
ALPHA A   ALPHA :   Shift =   5.5   Enter
```

Vrijednost varijable možemo promijeniti kada želimo, npr.

```
ALPHA A   ALPHA :   Shift =   1.11   Enter
```

S varijablama možemo vršiti sve računske operacije kao i s brojevima, npr.

```
A + 1   Enter
```

daje 2.11 kao rezultat.

Varijabla je simbol koji može poprimiti bilo koju vrijednost i tako nam omogućiti pisanje općenitijih algoritama (postupaka). Ako u algoritmu koristimo varijablu (simbol), tada je algoritam univerzalan jer simbol može poprimiti bilo koju vrijednost.

Npr., postupak rješavanja kvadratne jednadžbe opisuje se formulom (postupkom) za dobivanje rješenja $x_{1,2}$

$$x_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

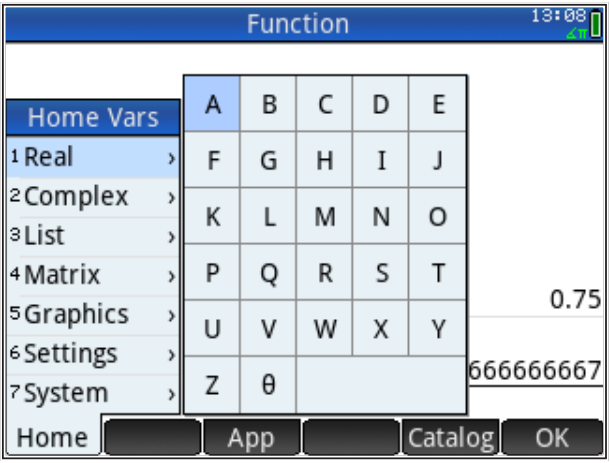
pri čemu predstavljaju bilo koji broj iz funkcije

$$f(x) = A \cdot x^2 + B \cdot x + C$$

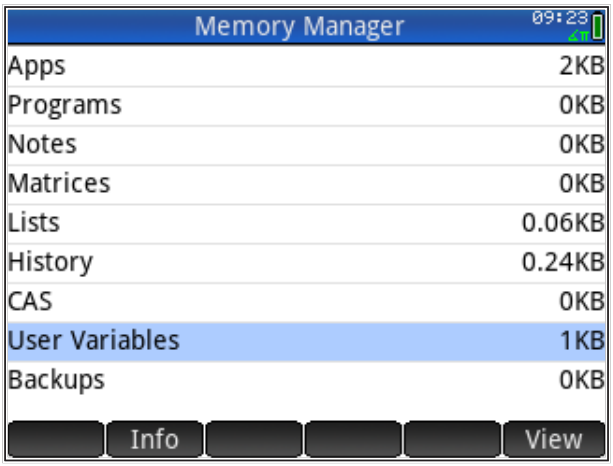
Treba primijetiti da je rješenje opisano formulom univerzalno rješenje jer vrijedi za bilo koju vrijednost A, B, C. Bez primjene varijabli morali bismo posebno računati za npr. $A = 1$, $B = -2$, $C = 3$, posebno za $A = 2$, $B = -7$, $C = -3$, itd.

Prije svake aktivnosti mora se vidjeti početni ekran. Za dobivanje početnog ekrana treba pritisnuti tipku 'kućica'.

Pregled aktivnih varijabli na HP Prime možemo vidjeti upotrebom tipke 'Vars',



a pregled cjelokupne memorije tipkama Shift Mem.



Rad s listama i matricama

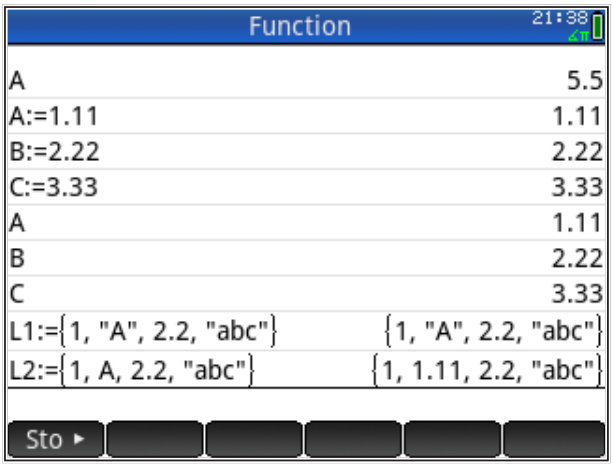
Liste

Liste („popisi“) su takva vrsta varijabli u koje podatke zapisujemo u nizu pri čemu možemo proizvoljno miješati razne vrste podataka (možemo ih usporediti sa sekvencijalno zapisanom tekstualnom datotekom koja je spremljena u memoriji kalkulatora).

Primjeri liste

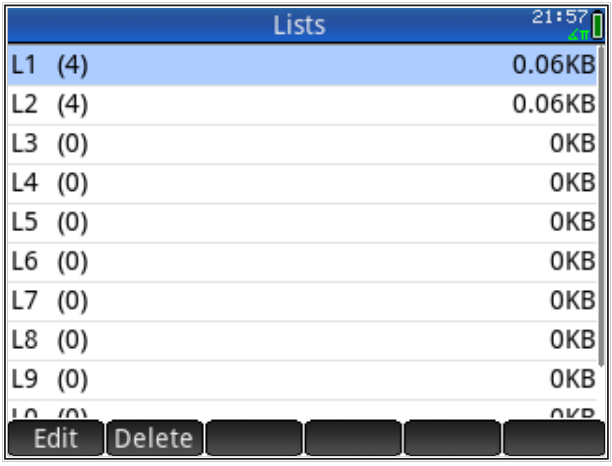
```
L1={1,"A",2.2,"abc"}
L2={1,A,2.2,"abc"}
```

Lista (kao svaka varijabla) ima ime (ovdje 'L'1' i 'L2'); ove dvije liste imaju svaka po 4 elementa, a elementi liste omeđeni su vitičastom zagradom i odijeljeni zarezom. Ove dvije liste na ekranu HP Prime kalkulatora izgledaju ovako:

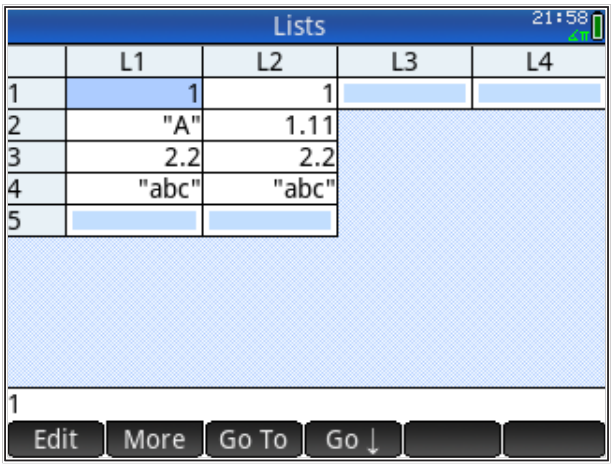


Vidimo da smo liste zadali kao bilo koju varijablu. Također, uočimo da 'A' u listi 'L' nije varijabla 'A', nego slovo 'A' jer smo ga upisali unutar navodnika. Navodnici označavaju slovne znakove (doslovni tekst), varijable predstavljene slovima označavaju se bez navodnika (vidi listu 'L2' na slici).

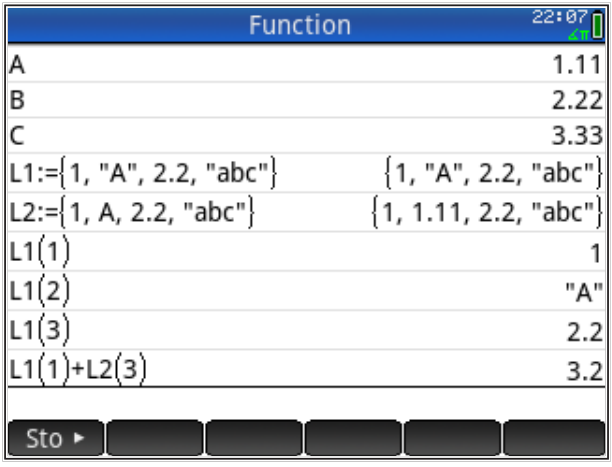
Liste možemo pregledavati i zadavati u posebnom pregledniku (editoru) za liste, gdje pritiskom na *Shift* *List* dobivamo pregled svih lista,



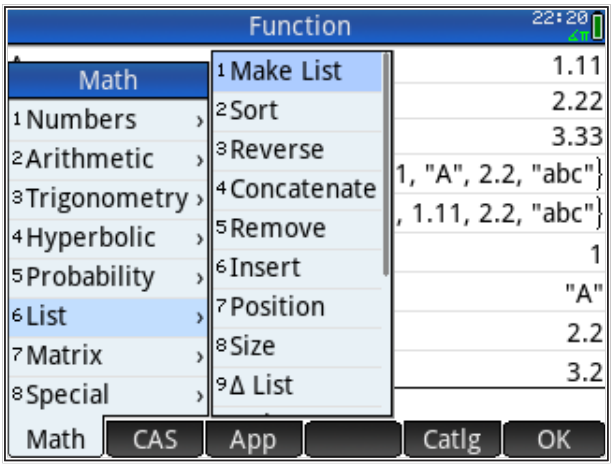
a pritiskom na 'Edit' dobivamo mogućnost mijenjanja (i zadavanja, brisanja) sadržaja pojedine liste.



Pojedine elemente liste možemo dobiti tako da indeksiramo element unutar liste prema njegovoj poziciji u listi; (1) za prvi element, (2) za drugi, itd.



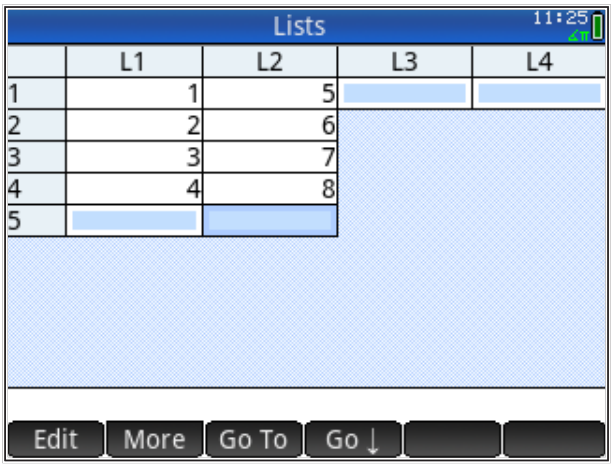
S elementima liste možemo vršiti sve operacije kao i s bilo kojom varijablom. Popis operacija nad listama dobiva se pritiskom na tipku sa slikom kovčežića i nakon toga izborom 'Math' pa 'List' u meniju.



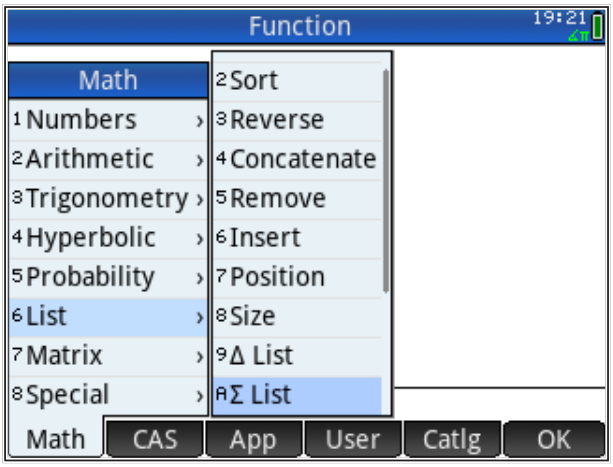
Operacije nad listama u pravilu se izvode po svim elementima liste (kada je to moguće). Npr., produkt dvije liste jednake dužine nije skalarni produkt vektora, nego nova lista s istim brojem elemenata od kojih je svaki produkt elemenata na istoj poziciji u svakoj listi.

Važna primjena lista odnosi se na rad s nizovima numeričkih podataka, gdje možemo reći da omogućuju neku vrstu „vektorizacije“ izračuna (to nisu operacije linearne algebre, koje izvodimo na matricama i vektorima). Ukoliko napravimo usporedbu s „Mathcad“ okruženjem, možemo reći da liste omogućuju rad s podacima za koje kod Mathcada koristimo indekse. Usporedbu ta dva pristupa možemo vidjeti kod izračuna u fizici zgrade; slojeve konstrukcije upisujemo u liste i računamo s listama, umjesto s pojedinačnim brojevima.

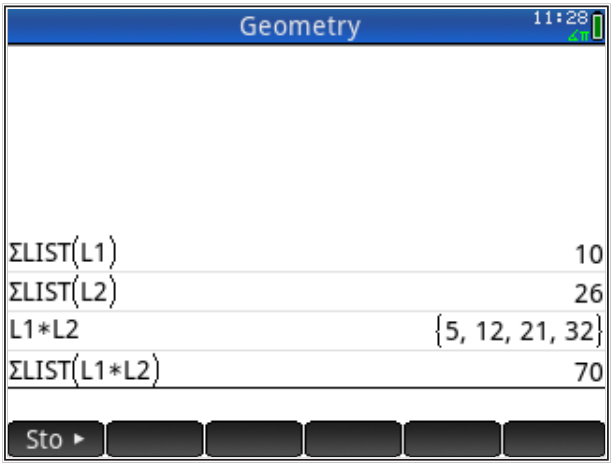
Primjer brojevne liste neka je (zadajemo preko editora u HP kalkulatoru, *Shift* *List*)



Izračun sume elemenata liste možemo jednostavno dobiti pomoću ugrađene funkcije 'suma liste' $\sum List$ (naredbu ne možemo upisati znak po znak, nego se treba odabrati iz popisa naredbi: tipka 'kovčežić', zatim 'Math' pa 'List i onda skrolamo do naredbe $\sum List$)



Primjeri sumiranja listi su

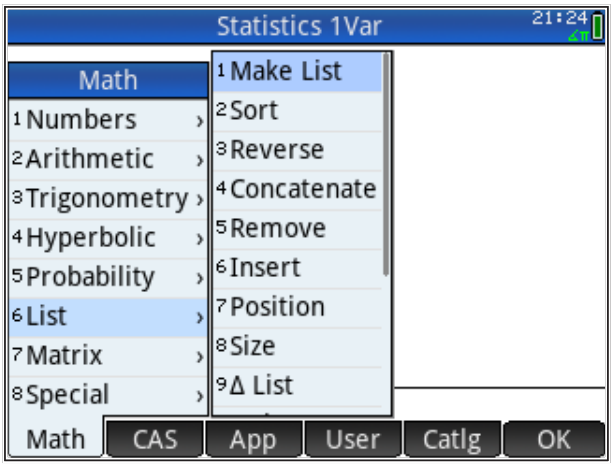


Automatsko generiranje listi

Automatsko generiranje listi omogućuje brzo i jednostavno generiranje pravilnih listi (onih čiji se članovi mogu izraziti formulom). Koristi se naredba 'MAKELIST', čija je sintaksa

```
MAKELIST (izraz, varijabla, početak, kraj, korak)
```

Naredba se može direktno upisati na ekran HP Prime-a ili odabrati iz menija funkcija (tipka s 'kovčežićem').



Npr., ako želimo listu parnih brojeva {2,4,6,8,10} napisat ćemo

```
MAKELIST(2*K,K,1,5,1),
```

gdje je K varijabla (može biti bilo koje slovo) koju koristimo za indeksiranje u intervalu od 1 do 5 po 1, tj., K=1,2,3,4,5. Izraz je 2*K i za K=[1..5] poprimit će vrijednosti 2,4,6,8,10.

Za listu neparnih brojeva {1,3,5,7,9} napisat ćemo

```
MAKELIST(2*K-1,K,1,5,1)
```

gdje je K=1,2,3,4,5. Izraz je 2*K-1 i za K=[1..5] poprimit će vrijednosti 1,3,5,7,9.

Za listu brojeva {1,4,7,10,13}, čiji se članovi povećavaju za 3, napisat ćemo

```
MAKELIST(1+(K-1)*3,K,1,5,1),
```

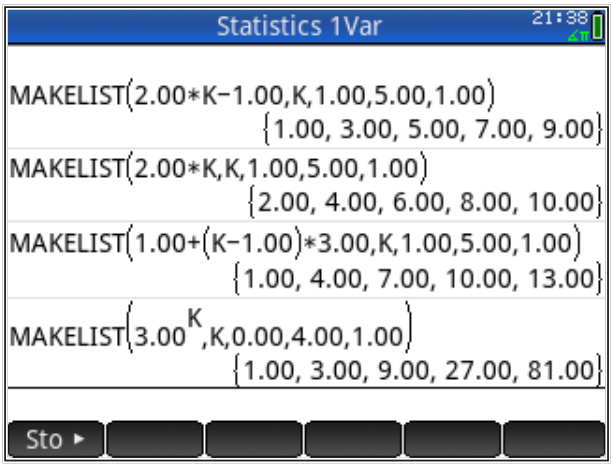
gdje je K=1,2,3,4,5. Izraz je 1+(K-1)*3 i za K=[1..5] poprimit će vrijednosti 1,4,7,10,13.

Za listu brojeva {1,3,9,27,81}, čiji su članovi umnožak prethodnog člana s brojem 3, napisat ćemo

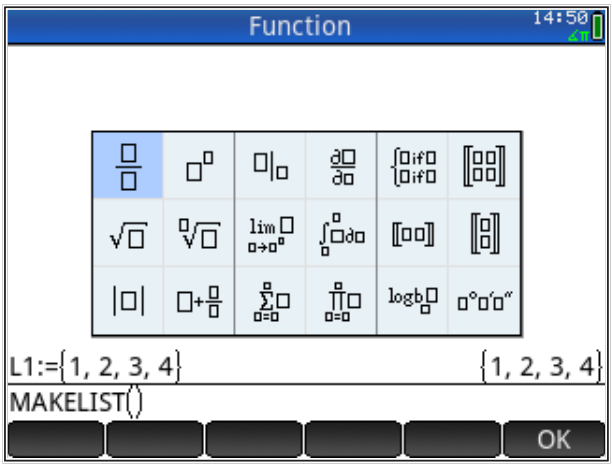
```
MAKELIST(3^K,K,0,4,1),
```

gdje je K=0,1,2,3,4. Izraz je 3^K i za K=[1..5] poprimit će vrijednosti , odnosno 1,3,9,27,81.

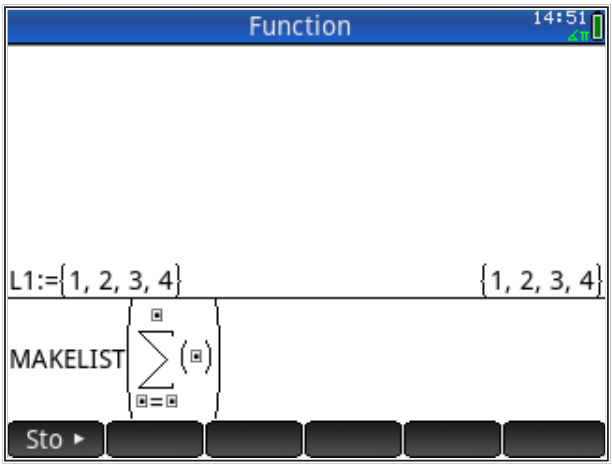
Sve liste vidimo definirane na slici



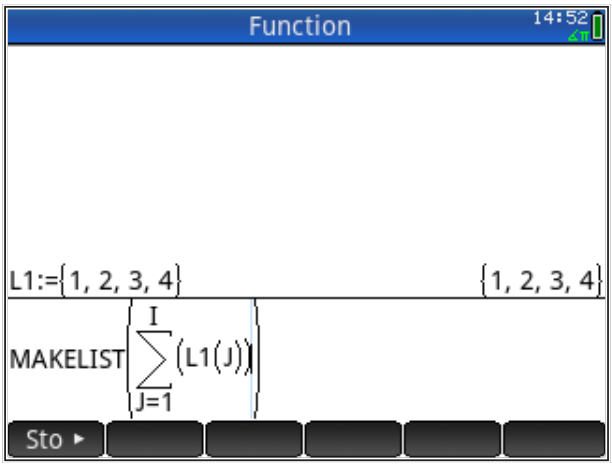
Često je potrebno izračunati parcijalnu sumu elemenata liste, odnosno, načiniti novu listu kojoj su elementi sume elemenata druge liste do tog rednog broja. Za objašnjenje, neka je npr., $L1 = \{1,2,3,4\}$; lista parcijalnih suma je tada $\{1,3,6,10\}$. Ukoliko ne želimo programirati, takvu listu dobijemo kombinacijom naredbe 'Make List' i 'Suma', pri čemu komandu za sumiranje dobivamo iz menija (nalazi se na istoj tipki gdje i slovo C).



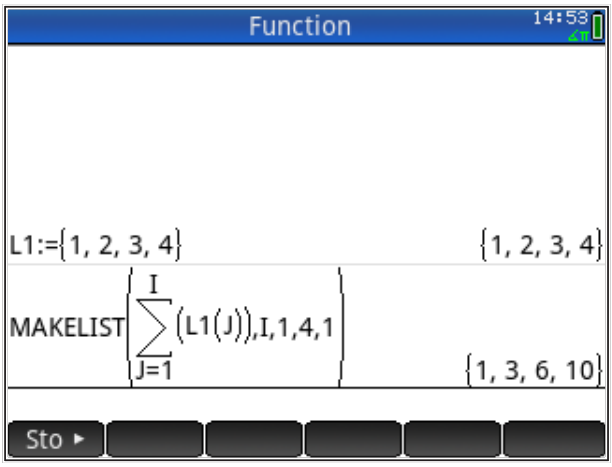
Dobivamo znak sumiranja gdje treba popuniti mjesta za varijable



Time je definiran izraz za izračun liste, a još treba dodati varijablu i njen raspon (od, do, po).



Dodajemo varijablu 'I', koja ide od 1 do 4 po 1, tj., I=1,2,3,4



Dobivena parcijalna suma može se pridodati nekoj listi, npr., listi 'L3'

```
L3:=Ans
```

Treba obratiti pažnju da smo u naredbi kombinirali dva indeksa, 'I' i 'J' i tako dobili promjenjivu granicu sumiranja.

Za I=1 izraz je , tj., J=1; sumira se samo 1. član liste L1.

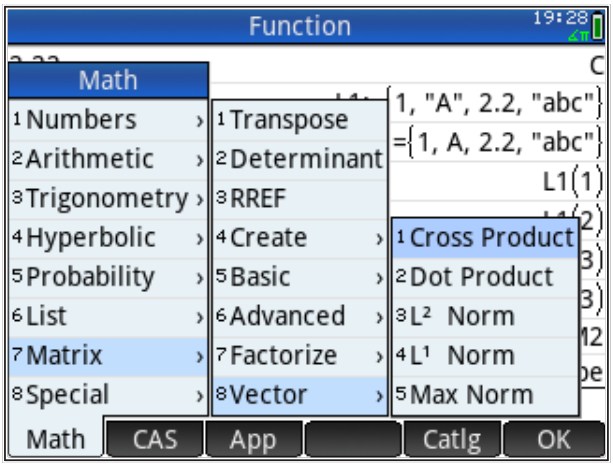
Za I=2 izraz je , tj., J=1,2; sumira se 1. i 2. član liste L1.

Za I=3 izraz je , tj., J=1,2,3; sumira se 1., 2. i 3. član liste L1.

Za I=4 izraz je , tj., J=1,2,3,4; sumira se 1., 2., 3. i 4. član liste L1.

Matrice

Pod pojmom matrice podrazumijevamo i vektore, koji su matrice sa samo jednom dimenzijom. S matricama i vektorima možemo vršiti sve operacije linearne algebre, npr. skalarni i vektorski produkt vektora, množenje matrica, svojstvene vrijednosti i još puno toga. Popis operacija nad matricama/ vektorima možemo dobiti pritiskom na tipku sa slikom kovčežića i nakon toga izborom 'Math' u meniju

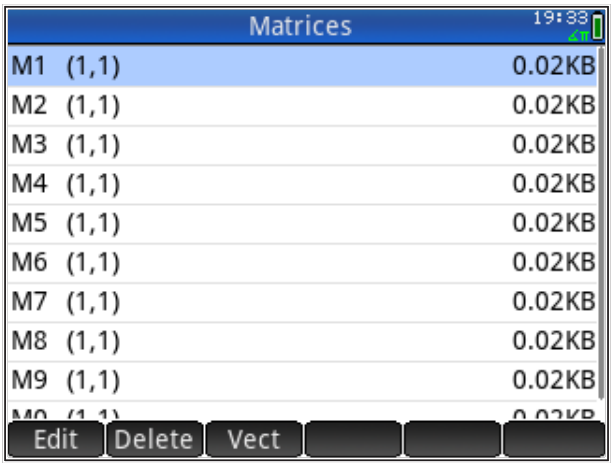


Iako nad listama ne možemo vršiti operacije kao nad vektorima, liste možemo lako prekopirati u vektore. Npr., listu 'L2' u vektor 'M2', tipkamo

```
Alpha L 2 STO> Alpha M 2
```

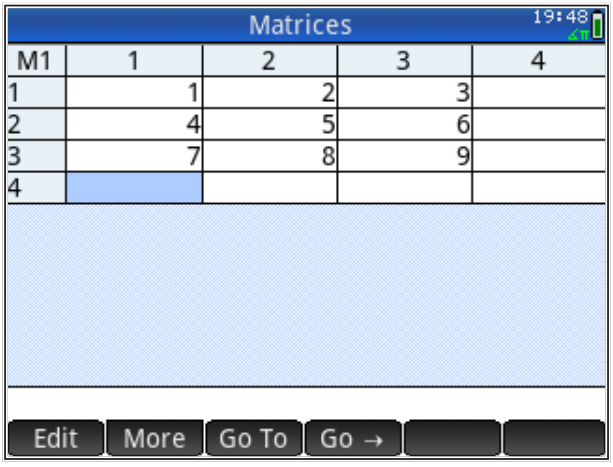
Lista pritom smije sadržavati samo brojeve (tip podatka kompatibilan s vektorom jer se u protivnom javlja greška 'Error: Bad argument type').

Matrice (i vektore) možemo zadavati slično listama, ali u editoru za matrice, tipkamo 'Shift Matrix'



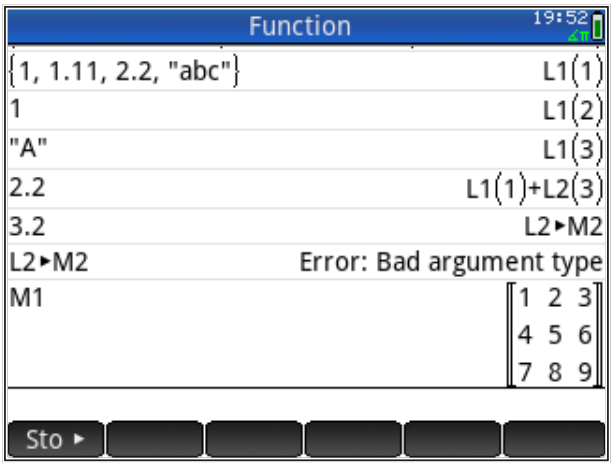
i dobijemo pregled matrica u memoriji kalkulatora. Pritiskom na tipku 'Vect' u meniju na dnu ekrana odabranu matricu mijenjamo u vektor (broj dimenzija mijenja se u 1). Pritiskom na 'Vect' *vektor se mijenja u matricu (dodaje se 1 dimenzija). 'Vect'* se u meniju na dnu ekrana pojavljuje ako je odabran vektor, a ako je odabrana matrica, u meniju na dnu ekrana stoji 'Vect'.

Odabirom matrice i pritiskom na 'Edit' ulazimo u editor kojim uređujemo odabranu matricu, tako da matrica M1 postaje

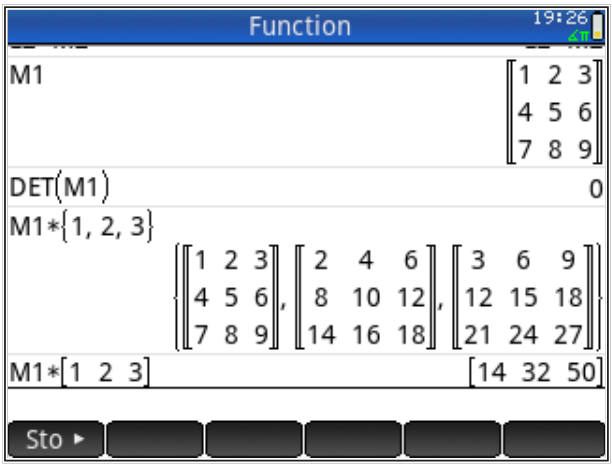


Pritiskom na tipku 'Home' vraćamo se na osnovi ekran. Upravo zadanu matricu možemo vidjeti tipkanjem njenog imena

```
Alpha M 1   Enter
```

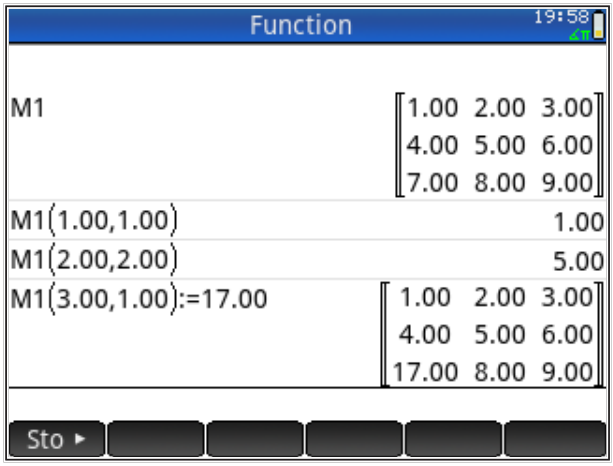


Dalje s matricom M1 možemo vršiti sve operacije linearne algebre, i druge. Npr., ovdje smo izračunali determinantu matrice i pomnožili matricu s vektorom.

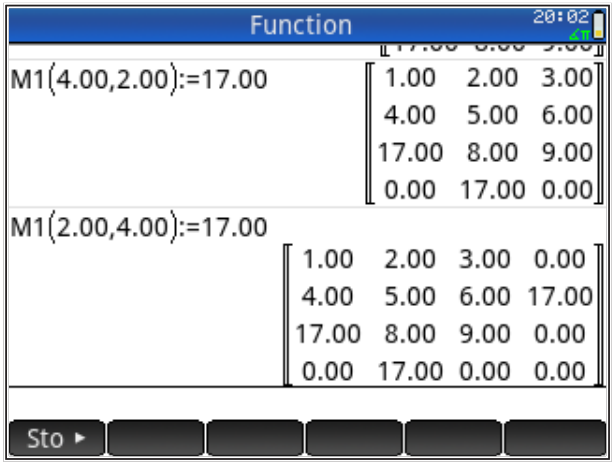


Sa slike se vidi da treba razlikovati množenje listom (operacija se vrši član po član) i množenje vektorom (operacija definirana samo između matrica i vektora, čiji je rezultat matrica ili vektor).

Član matrice dobivamo preko indeksa u zagradama; npr., M(2,2) je element matrice M1, koji se nalazi u drugom redu i drugom stupcu. Na isti način možemo i zadati pojedini član matrice; npr., M1(3,1):=17 zadaje vrijednost u matricu M1 na mjesto u trećem retku i prvom stupcu, kao na slici.

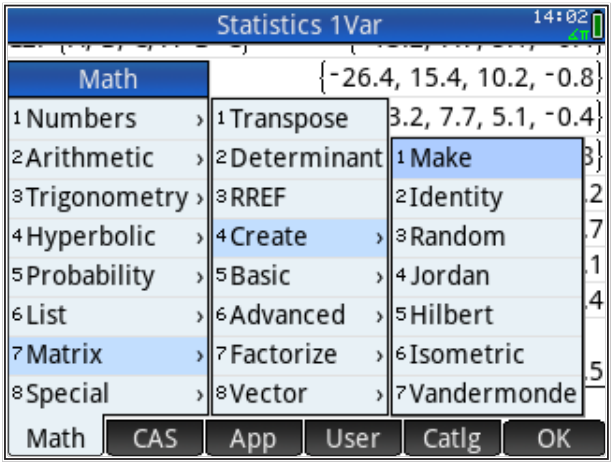


Zadavanjem vrijednosti u redak i stupac koji ne postoje u matrici ona se tim retkom i stupcem proširuje.



Automatsko generiranje matrica

Automatsko generiranje matrica važno je kod programiranja numeričkih metoda, poput metode konačnih razlika ili metode konačnih elemenata. Osnovna naredba za generiranje matrica je 'MAKEMAT', a možemo je direktno otipkati na ekranu HP Prime kalkulatora ili odabrati iz menija.



Sintaksa naredbe je

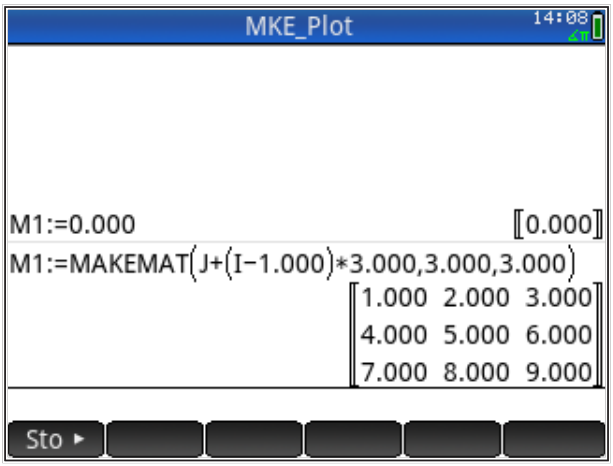
```
MAKEMAT(izraz, broj redaka, broj stupaca)
```

gdje ujedno i definiramo broj redaka i stupaca u matrici.

Npr., možemo kreirati matricu

$$\mathbf{M1} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

preko naredbe MAKEMAT(J+(I-1)*3,3,3)



U naredbi 'MAKEMAT' definirali smo dimenzije matrice 3x3, što znači da indeks redaka ide I=1,2,3, a indeks stupaca je također J=1,2,3.

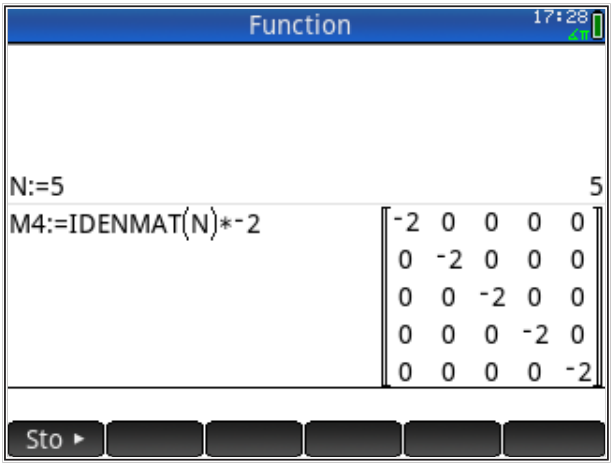
Dakle, za prvi redak matrice I=1, a J=1,2,3 pa formula J+(I-1)x3 daje vrijednosti 1,2,3 (jer je (I-1)=0).
Za drugi redak matrice I=2, a J=1,2,3 pa formula J+(I-1)x3 daje vrijednosti 4,5,6 (jer je (I-1)=1).
Za treći redak matrice I=3, a J=1,2,3 pa formula J+(I-1)x3 daje vrijednosti 7,8,9 (jer je (I-1)=2).

Na sličan način možemo brzo i jednostavno kreirati razne matrice koje se javljaju u inženjerskim zadacima.

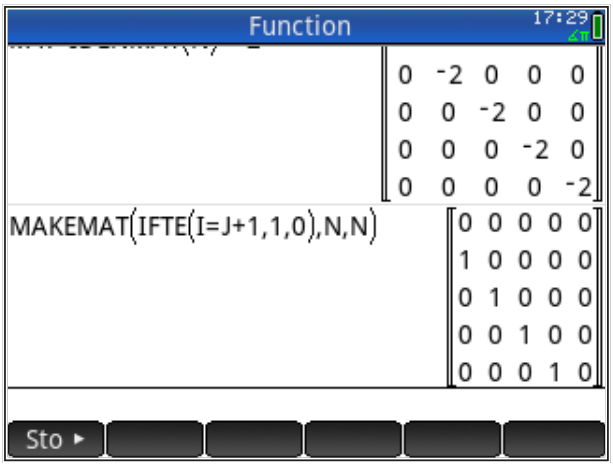
U metodi konačnih razlika često se pojavljuje matrica različite veličine, koja na glavnoj dijagonali ima broj -2 a na susjednim broj 1. Pri programiranju te metode potrebno je generirati takvu matricu proizvoljne veličine.

Pretpostavimo da želimo načiniti takvu matricu s 5 redaka i stupaca, N=5, a spremit ćemo je u varijablu M4.

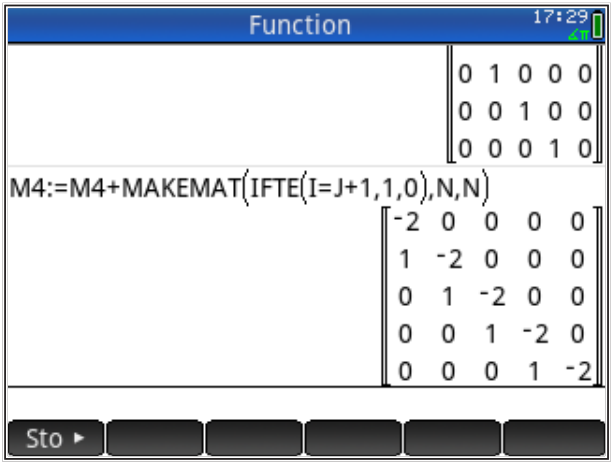
Za generiranje dijagonale iskoristit ćemo naredbu 'IDENMAT', koja kreira matricu identičnosti ('1' na dijagonali, a sve drugo '0'), i pomnožiti sve s '-2'.



Sada još treba dodati dijagonalne elemente iznad i ispod glavne dijagonale. Ispod dijagonale generiramo

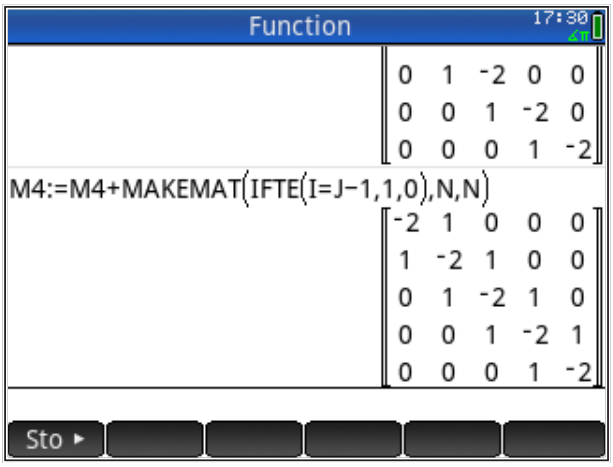


Dobivenu matricu odmah pribrajamo u M4



Vidimo upotrebu nove naredbe 'IFTE', koja omogućuje uvjetno izvršavanje naredbi: 'IFTE (test, točno, netočno)'. Ukoliko je uvjet naveden u 'test' ispunjen (istinit), izvršava se naredba iza prvog zareza, a ukoliko nije, izvršava se zadnja naredba.

Na isti način generiramo element iznad dijagonale (u 'MAKEMAT'-u nam samo treba 'J-1' umjesto 'J+1') i odmah ih pribrajamo matrici M4.



Matrica M4 je sada operator u metodi konačnih razlika i još samo treba uvesti rubne uvjete. Ako promijenimo vrijednost varijable 'N' i ponovimo postupak generiranja (kopiramo vrijednosti s ekrana, ne prepisujemo ih!), možemo generirati matricu bilo koje veličine tako da odgovara broju čvorova u metodi konačnih razlika.

HP Prime funkcije

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide (iz menija 'Help'):*

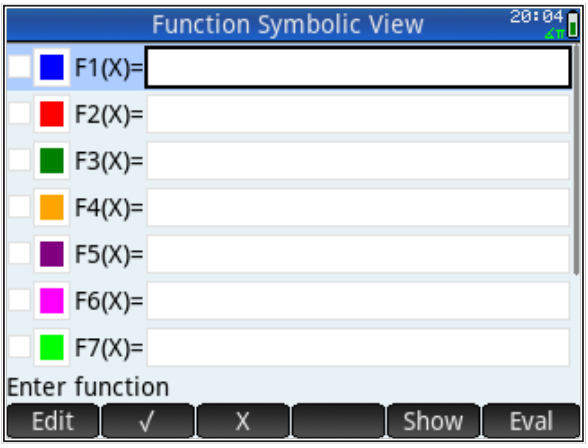
- HP apps and their views
- Primary apps: Function, Polar Catalogs and Editors

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

```
nema pretpostavki o stanju varijabli u memoriji
```

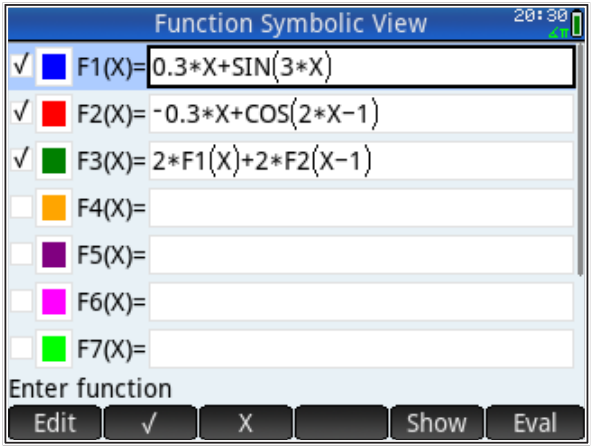
Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite pomoću 'Edit: Paste' naredbi iz menija kalkulatora pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

Prva aplikacija HP Prime kalkulatora je aplikacija za rad s funkcijama i u nju se ulazi pritiskom na tipku 'Apps', nakon čega se izabere 'Functions'. Na ekranu se pojavljuje



Vidimo da je predefinirano deset imena funkcija (F1,...,F0) koje možemo zadati slobodno. Sve one moraju biti funkcije koje čini samo jedna varijabla 'X' (možemo je kliknuti u meniju ispod ekrana radi bržeg unosa). Funkcije više varijabli, implicitne, polarne funkcije i druge, moramo definirati u drugim za to predviđenim aplikacijama.

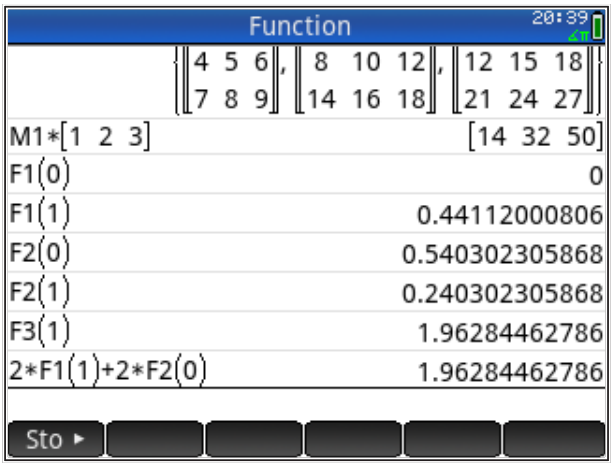
Funkciju unosimo tako da pritisnemo 'Edit' (meni ispod ekrana) i upišemo koeficijente uz 'X' i/ili druge definirane funkcije varijable 'X'. Npr., ako funkciju $0.3x + \sin(3x)$ upišemo kao $F1(x)$, a funkciju $-0.3x + \cos(2x)$ upišemo kao $F2(x)$, to se prikazuje kao na slici, gdje vidimo simbolički prikaz zadanih funkcija.



Definirali smo 3 funkcije, od kojih je treća $F3(X)$ definirana preko $F1(X)$ i $F2(X)$ pa možemo reći da je $F3$ funkcija funkcije. Mogućnost da definiramo funkciju funkcije je iznimno značajna i omogućuje veliku fleksibilnost u njihovu zadavanju.

Vidimo da, osim imena, funkcije radi lakšeg razlikovanja, imaju predefiniranu i boju.

Funkcije (i sve druge aplikacije) možemo gledati na 3 načina, koja dobivamo pritiskom na crne tipke *Symb*, *Plot* i *Num*. Gornja slika prikazuje ekran sa simboličkom definicijom funkcija (preko nepoznanice X), dobiven prilikom ulaska u aplikaciju ili pritiskom na tipku *Symb*. Iz simboličkog prikaza funkcija tipkom ⌂ vraćamo se na početni ekran. Kako bi dobili numeričke vrijednosti funkcija, ovdje možemo koristiti prije definirane funkcije u raznim izračunima, kao na slici

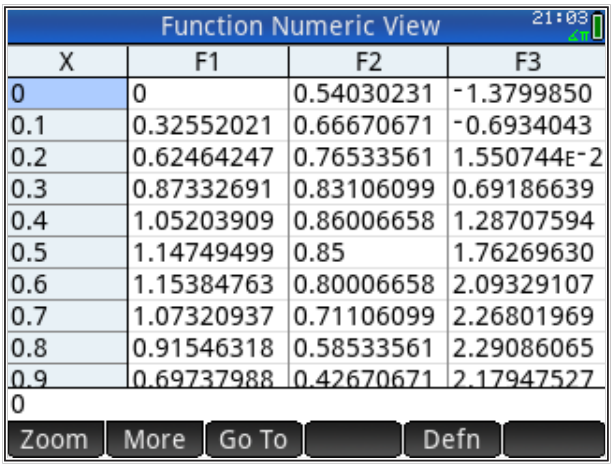


Zadatak: objasni zašto izrazi $F3(1)$ i $2F1(1) + 2F2(0)$ daju isti rezultat. Zašto pritom $F1$ i $F2$ imaju različite argumente?

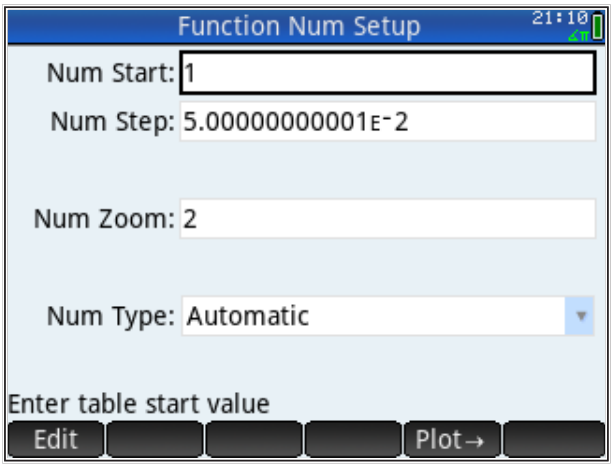
Tabelarni i grafički prikaz funkcija

Osim računanja vrijednosti funkcija za zadani X, funkcije možemo prikazati grafički i numerički. Tijek funkcije možemo na taj način vidjeti bolje, nego da računamo jednu po jednu vrijednost.

Tabelarni prikaz daje vrijednosti definiranih funkcija istovremeno za više vrijednosti varijable X u jednakim razmacima. Tabelarni prikaz dobivamo pritiskom na tipku *Num*.



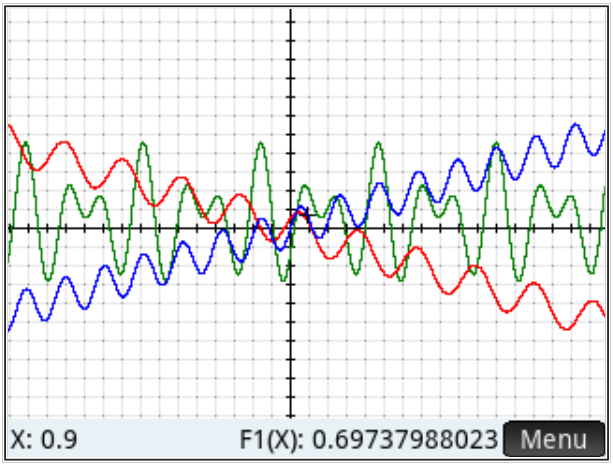
Pritiskom na meni 'Defn' ispod ekrana možemo zadati novi početni za koji se prikazuje tabela vrijednosti zadanih funkcija. Sve relevantne parametre za numerički prikaz možemo zadati pritiskom na *Shift* *Num → Setup* (uređivanje numeričke tablice).



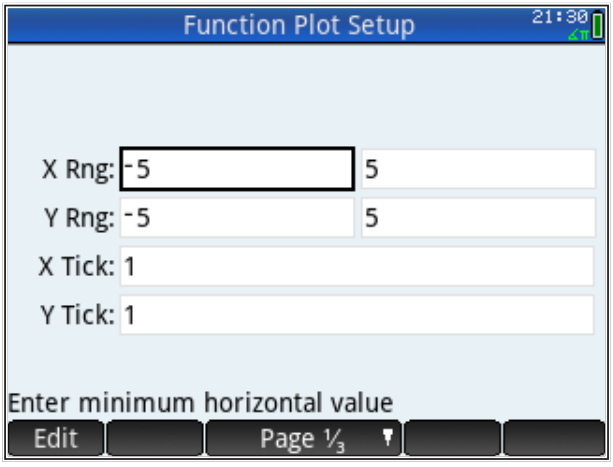
Tu namještamo početnu vrijednost u tablici numeričkog prikaza i korak promjene varijable X. Parametar 'Num Zoom' definira vrijednost koja će se

pojavljivati pri izboru 'Zoom' u meniju ispod tablice. To je faktor kojim se množi ('Zoom Out'), odnosno, dijeli ('Zoom In') odabrana vrijednost koraka promjene varijable X. Na taj način brzo možemo dobiti tabelarni prikaz za gušće ili rijeđe razmaknute vrijednosti X.

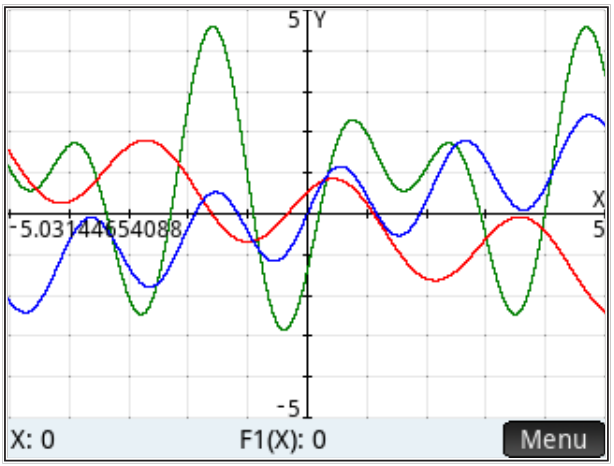
Grafički prikaz daje sliku definiranih funkcija, a svaka funkcija u svojoj je boji. Grafički prikaz aktiviramo pritiskom na tipku *Plot* (tipka ima i ikonu grafa na sebi). Za prije definirane funkcije dobivamo



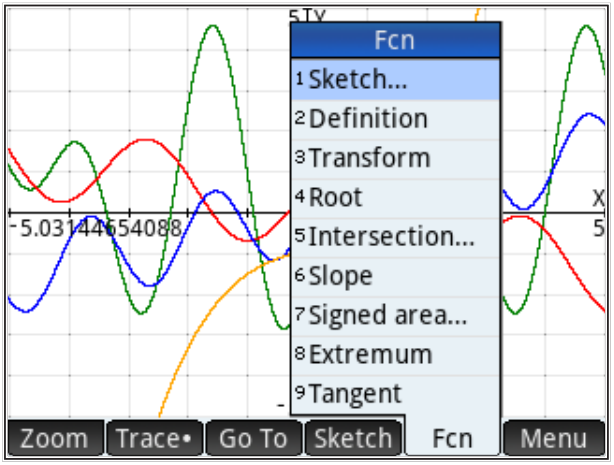
Parametre grafičkog prikaza možemo namjestiti pritiskom na *Shift* *Plot* → *Setup* (uređivanje grafičkog prikaza):



Promijenjeni su parametri 'X Range' i 'Y Range', tako da bismo trebali dobiti već uvećan dio grafičkog prikaza oko ishodišta. Ujedno je aktiviran (označen kvačicom) parametar 'Label' na stranici Page 2/3, tako da osi imaju oznaku

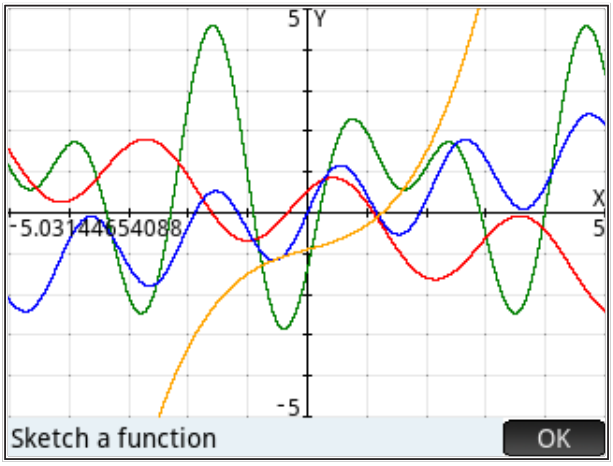


Pri dnu slike vidimo oznaku 'Menu', koji nudi dodatne mogućnosti uređivanja grafičkog prikaza.

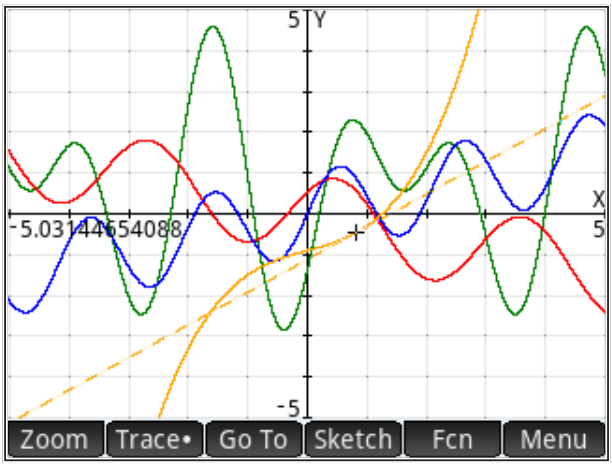


Prva oznaka na novom popisu grafičkog menija je naredba 'Sketch'. Izbor naredbe 'Sketch' omogućuje da prostoručno (koristeći prst ili kursorske tipke, odnosno miša na virtualnom kalkulatoru) skiciramo novu funkciju na ekranu. HP Prime će tada provući regresijsku krivulju što bliže liniji koju smo

prostoručno zadali i napisati jednadžbu koju će dodijeliti novoj, skiciranoj funkciji. Ta će funkcija biti dodana na kraj liste prethodno definiranih funkcija. Na ovom primjeru je to narančasta linija, pridodijeljena funkciji $F4(X)$.

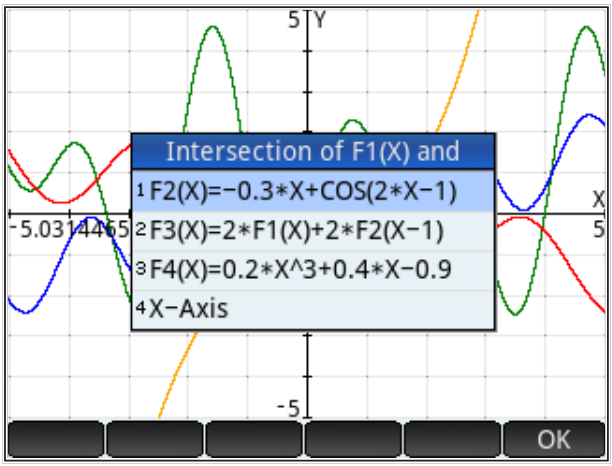


Pogledat ćemo kako se, naredbom 'Tangent' iz grafičkog menija, crta tangenta na funkciju. Nakon odabira te naredbe, kursorom možemo definirati točku x iz koje će se povući tangenta na odabranu funkciju. Izbor aktivne funkcije možemo mijenjati kursorskim kotačem (velika okrugla tipka ispod ekrana) tipkanjem po gornjem ili donjem rubu.

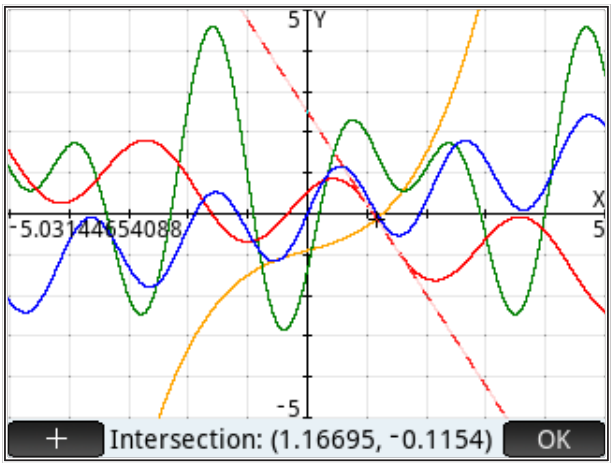


Tangenta se pokazuje crtkanom linijom u boji funkcije na kojoj je tangenta definirana.

Iz menija ćemo još istražiti naredbu 'Intersection', koja omogućuje da nađemo koordinate presjecišta dvaju funkcija. Jedna funkcija je funkcija prethodno odabrana tipkanjem po kursorskom kotaču, a drugu zadajemo na upit.



Vidljivo je da možemo naći i presjecište s osi x (nađeni je zapravo rješenje jednadžbe $F1(X) = 0$, odnosno, umjesto $F1$ može biti bilo koja odabrana funkcija). Presjecište je označeno na ekranu



a prikazane su i koordinate. Naredba '+' samo mijenja izgled kursora koji označava točku presjecišta.

Veza funkcije i algoritma

Funkcije su operatori nad nekim definiranim skupom, domenom funkcije. Funkcija pridružuje (preslikava) svakoj vrijednosti iz domene neku vrijednost u kodomeni (skupu vrijednosti funkcije). U slučaju HP Prime kalkulatora, domena i kodomena su skup realnih brojeva (u nekim slučajevima i skup kompleksnih brojeva). Način pridruživanja (preslikavanja) brojeva iz domene u kodomenu najbolje se vidi na grafičkom prikazu.

Potpunu fleksibilnost u primjeni funkcija dobivamo kad koristimo varijable umjesto konstanti, odnosno, umjesto da pišemo funkciju $0.3x + \sin(3x)$, koristimo varijable i pišemo $ax + \sin(bx)$ ili, još bolje, $ax + c\sin(bx)$. Varijable definiramo po želji pa možemo raditi s varijantama funkcije bez da ponovo definiramo funkciju. To je posebno važno kod rješavanja jednadžbi koje dobivamo iz funkcija tako da vrijednost funkcije izjednačimo s nulom. Tada rješenje dobivamo preko varijabli (a, b, c i druge) i takvo je rješenje općenito. Također, rad s varijablama omogućuje nam parametarsku analizu gdje možemo proučavati utjecaj pojedinog parametra na vrijednost funkcije i na rješenja jednadžbe definirane tom funkcijom.

Primjer

Želimo računati vrijednosti funkcije zadane formulom[5]

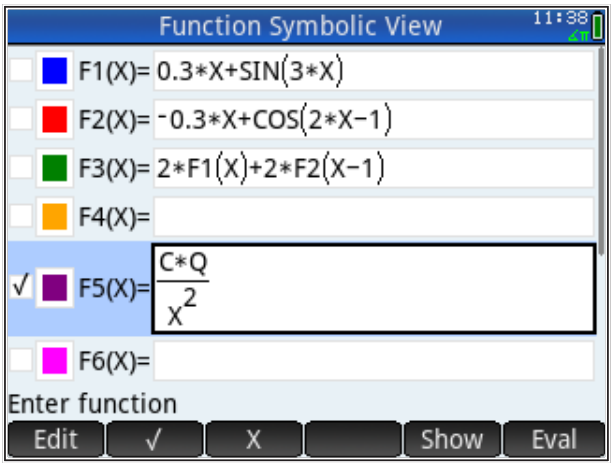
$$v = c \cdot \frac{q}{d^2}$$

gdje je $c = 1.273$. Jednadžba opisuje brzinu tečenja [m/s] u okrugloj cijevi promjera d [m] uz protoku q [m^3/s]. Primijetimo da brzina može biti funkcija promjera $v = v(d)$ ili protoke $v = v(q)$; mi pretpostavljamo funkciju promjera, $v = v(d)$. Naravno, ova formula je valjana samo uz uvjet da su ispunjene pretpostavke da je cijev okruglog presjeka, potpuno ispunjena vodom i da brzina tečenja nije prevelika; hrapavost cijevi ne ulazi u formulu.

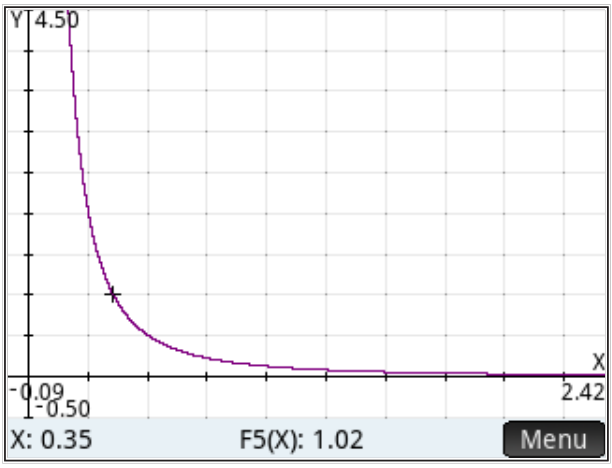
Pretpostavit ćemo da vrijednosti spremamo u varijable C, Q, D. Prebacujemo se na 'Home' način, postavit ćemo konstantu $C := 1.273$, protoku $Q := 0.1$ (koliko je to litara/sekundi?), a umjesto D pisat ćemo X (u aplikaciji 'funkcije'

5 https://www.engineeringtoolbox.com/pipe-velocity-d_1096.html

argument funkcije uvijek mora biti označen slovom 'X'). Formulu smo zadali kao 'F5' i označili kvačicom (grafički se prikazuju samo označene funkcije)



Grafički prikaz zavisnosti brzine tečenja o promjeru cijevi (uz zadanu protoku) prikazan je grafom



Iz slike je vidljivo da, npr., ako želimo da je brzina tečenja u cijevi za protok q=100 l/s manja od 1 m/s, promjer cijevi mora biti veći od 35 cm. Parametri grafa su namješteni tako da je korak po osi X (promjer) 0,25 m i po osi Y (brzina) 0,5 m/s.

Algoritmi su postupci kojima definiramo (opisujemo) neku aktivnost koja nam je od interesa. Možemo razlikovati numeričke, logičke i druge algoritme. Numerički algoritmi u pravilu opisuju matematičke postupke koji vode do rješenja neke grupe matematičkih problema. Za logičke algoritme možemo reći da su uglavnom vezani uz igre i opisuju pobjedničke strategije. U ovome radu ograničit ćemo se na numeričke algoritme, one koji opisuju neke postupke nad funkcijama koje opisuju neke inženjerske modele. Drugim riječima, funkcije mogu predstavljati algoritme (naravno, nisu sve funkcije algoritmi). Tako je, na primjer, zbrajanje brojeva algoritam, koji proces zbrajanja raščlanjuje do tablice zbrajanja (u dekadskom brojevnom sustavu, to je tablica zbrajanja do 20), a postupak i redoslijed operacija vrlo su precizno definirani. Prije nego je taj algoritam definiran, zbrajanje velikih brojeva bila je vrlo cijenjena i rijetka vještina (npr., u starom Egiptu, Grčkoj ili Rimu). Sam algoritam usko je vezan uz indijsko-arapski način zapisivanja brojeva[6]. U računalu je, također, postupak zbrajanja definiran posebnim algoritmom; kada pozivamo operaciju zbrajanja, poziva se taj algoritam. Slično je i s ostalim računskim operacijama; sve su definirane posebnim algoritmima.

Algoritmi se često opisuju tzv. dijagramima toka, sačinjenim od posebnih simbola koji označavaju početak algoritma, postupak računanja, grananja, kraj.

Logički algoritmi se često označavaju drvom algoritma koje ima početak, grane koje odgovaraju mogućim ishodima (izborima) i kraj. Uspješna strategija se onda predstavlja jednim ili više tokova niz drvo algoritma (više o algoritmima u poglavlju o programiranju).

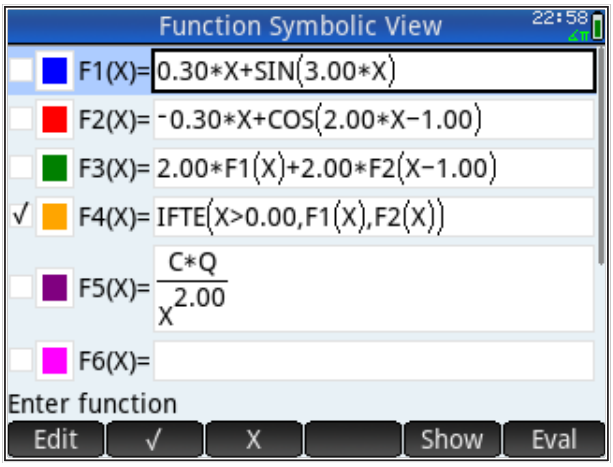
Ovdje ćemo dati primjer algoritma koji je dovoljno kompaktan da se može koristi unutar funkcija definiranih u aplikacijama na HP Prime.

Kao primjer algoritma s grananjem koristit ćemo funkciju 'IFTE' (If Then Else) u 'in-line' izvedbi, tj. 'IFTE' se odvija u jednom retku, ne definiraju se blokovi naredbi. 'IFTE' smo već koristili kod zadavanja matrica, sintaksa je

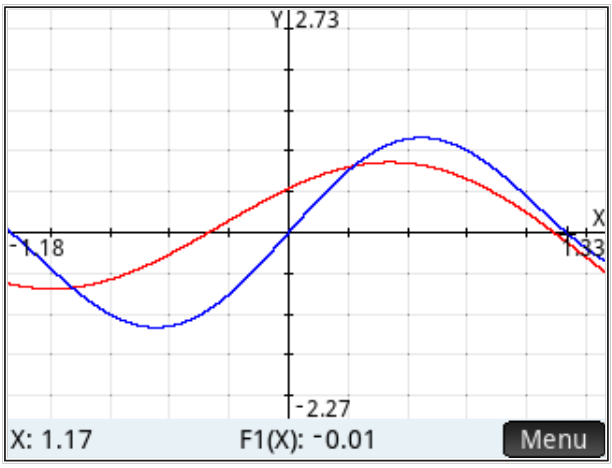
```
IFTE(test, True, False)
```

6 https://en.wikipedia.org/wiki/Hindu–Arabic_numeral_system

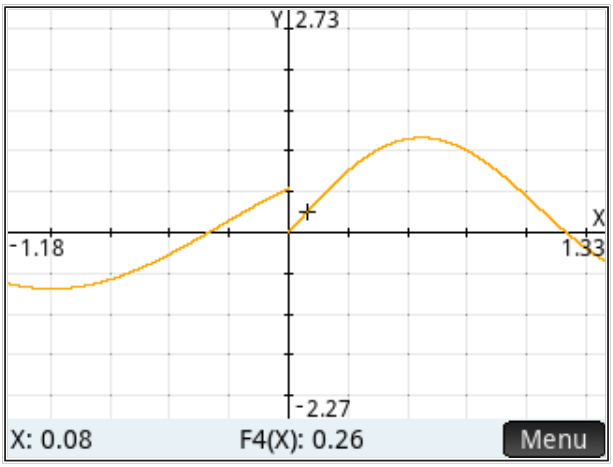
Ukoliko je rezultat testa točan, izvodi se naredba neposredno iza testa (ona koji stoji na mjestu 'True'). Ukoliko nije, izvodi se druga naredba (ona koji stoji na mjestu 'False'). Naredbu 'IFTE' možemo koristiti kod zadavanja funkcija pa je tako F4(X) zadana da je jednaka F1(X) ili F2(X), ovisno i o vrijednosti argumenta 'X'.



Najbolje ćemo vidjeti što radi funkcija F4(x) ako prvo pokažemo graf samo sa F1(X) (plavo) i F2(X) (crveno) te stavimo kvačice na F1(X) i F2(X).



Vidimo da je funkcija F4(X) (žuto) za $X > 0$ jednaka funkciji F1(X), a za $X < 0$ jednaka funkciji F2(X), kao što je vidljivo iz slike



Na taj način možemo definirati izlomljene funkcije ili funkcije sastavljene od više dijelova i sl.

Treba napomenuti da HP Prime ne zna povući tangentu na tako definiranu izlomljenu funkciju, a na funkciju, složenu od drugih funkcija (kao F3(X)), zna. To znači da, treba li nam tangenta na tako sastavljenu funkciju, ona isto mora biti definirana kao sastavljena funkcija (IFTE s derivacijama).

Isto tako, funkciju s definiranom logikom grananja možemo pozivati i s 'Home' ekrana i računati joj vrijednosti.

Zadatak: Nađi neku izlomljenu funkciju (odnosno, funkciju sastavljenu od više jednostavnijih funkcija) koja ima primjenu u građevinarstvu i nacrtaj je.

Veza funkcije i algoritma preko liste

Lista omogućuje zapisivanje podataka u niz i kasniju obradu. Ukoliko se radi o brojčanim podacima, mnoge operacije možemo provesti na listi umjesto da radimo broj po broj; možemo reći da je to neka vrsta jednostavne vektorizacije problema. Rad s listama bez povezivanja funkcija prikazan je u poglavlju 2 *Pregled značajki*.

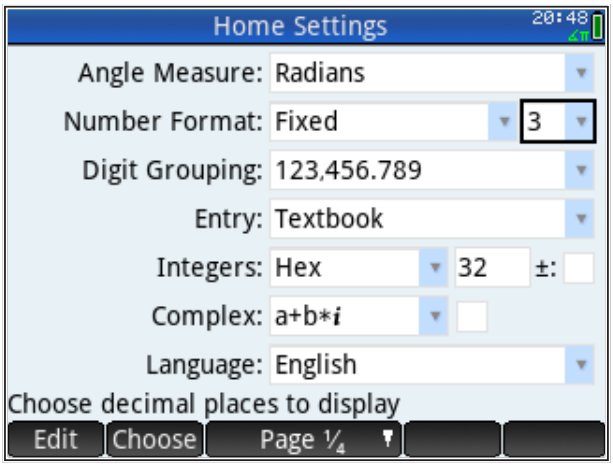
Povezivanje liste i funkcije vrši se putem naredbe

```
MAKELIST(f(X),X,Xmin,Xmax,deltaX)
```

pri čemu je f(X) funkcija koja definira neku operaciju (preslikavanje) nad varijablom 'X', 'X' je ime varijable (koje može biti bilo koji znak a ne samo 'X' ali mora biti veliko slovo), 'Xmin' je početna vrijednost intervala varijable, 'Xmax' je završna vrijednost, 'deltaX' je interval kojim se povećava 'Xmin' sve dok se ne dostigne (ili ne prekorači) 'Xmax' (vidi i *Korisnički priručnik*).

Primjer

Pretpostavlja se da su HP Prime unesene funkcije F1(X) i F2(X), kao što je prije navedeno. Također, format prikaza brojeva je promijenjen na fiksni s tri decimale da se izbjegne prikaz predugačkih brojeva.

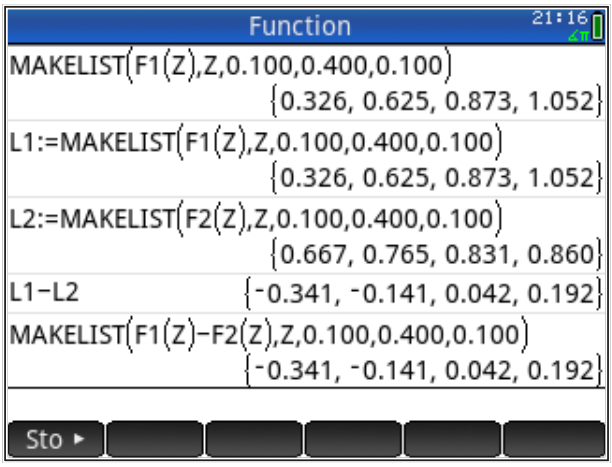


Ukoliko želimo dobiti vrijednosti funkcije F1 u nizu točaka {0.1,0.2,0.3,0.4}, pišemo

```
MAKELIST(F1(Z),Z,0.1,0.4,0.1)
```

Namjerno je upotrijebljena varijabla 'Z' da se vidi da je ime varijable proizvoljno (veliko slovo). Spremimo te vrijednosti u listu L1. Isto možemo napraviti i za funkciju F2 i spremiti vrijednosti u listu L2. Sada možemo vršiti operacije s listama; npr., za izračun razlike u vrijednosti funkcija F1 i F2 u točkama {0.1,

0.2, 0.3, 0.4 }, pišemo samo L1 - L2. Prikaz navedenih izračuna na ekranu HP Prime kalkulatora izgleda ovako:



Vidi se i drugi način na koji smo mogli dobiti isti rezultat; iz tog je postupka razvidno da funkcije možemo kombinirati.

Naravno, funkcije u naredbi 'MAKELIST' ne moraju biti zadane u aplikaciji 'Funkcije'. Npr., možemo napisati

```
MAKELIST(0.3*Z+SIN(3*Z),Z,0.1,0.4,0.1)
```

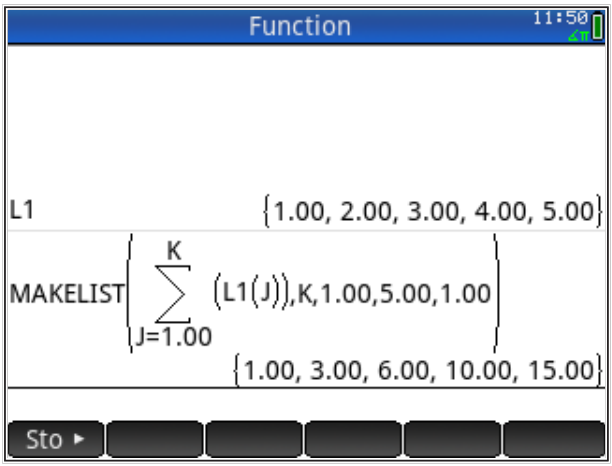
i dobit ćemo isti rezultat {0.326, 0.625, 0.873, 1.052 } kao i za F1.

Isto tako, kao što smo vidjeli u poglavlju 2 *Pregled značajki*, i same liste mogu biti funkcije u naredbi 'MAKELIST'.

Primjer parcijalne sume

Čest zadatak je sumirati članove niza/liste, uključivo do člana koji razmatramo. Npr., neka je polazni niz L1={1,2,3,4,5}; tada je niz parcijalnih suma L2={1,3,6,10,15}; dobiva se po formuli $član(k) = \sum_{i=1}^{i=k} \; član_i$, pa je tako 3. član = 1+2+3=6, 4. član = 1+2+3+4=10, itd.

Na HP Prime to izgleda ovako (promijenili smo u *Settings* formatiranje brojeva na 'Fixed' s 2 decimale, da nam sve stane na ekran):



Naravno, lista L1 može biti bilo kakva, a naredba 'MAKELIST' načinit će parcijalnu sumu.

Primjer Fibonaccijevog niza preko višestrukih listi

Ovo je malo napredniji primjer za vježbanje rada s listama. Fibonaccijev niz dobiva se na način da je svaki član niza jednak sumi prethodna dva člana, a počinje s {1,1} kao prva dva člana; prvih 8 članova niza izgleda ovako {1,1,2,3,5,8,13,21}.

Fibonaccijev niz se može formirati, relativno komplicirano, pomoću sume nekoliko listi s parcijalnih sumama. Jednostavniji pristup preko posebne funkcije prikazat ćemo u nastavku. Ovaj pristup je zanimljiv kao vježba za rad s listama.

Liste parcijalnih sumi koje treba sumirati za prvih 8 članova su

```
{1,1,1,1,1,1,1,1}
{0,0,1,2,3,4,5,6}
{0,0,0,0,1,3,6,10}
{0,0,0,0,0,0,1,4}
+
{1,1,2,3,5,8,13,21}
```

Za više članova niza treba još parcijalnih sumi.

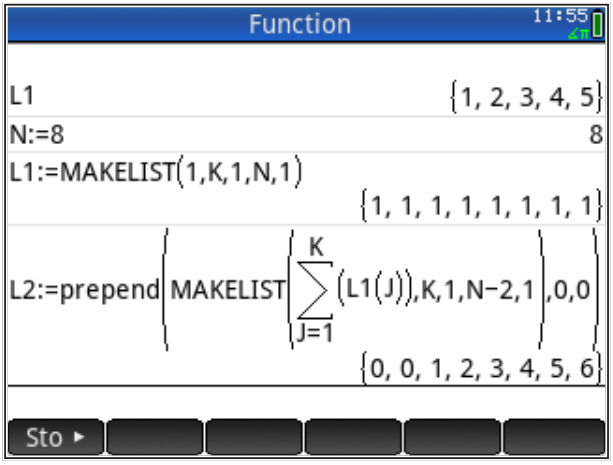
Počinje se s listom jedinica i svaka slijedeća lista počinje s dvije nule iza kojih slijede članovi koji su parcijalna suma prethodne liste. Dvije nule dodajemo naredbom

```
prepend(lista,0,0)
```

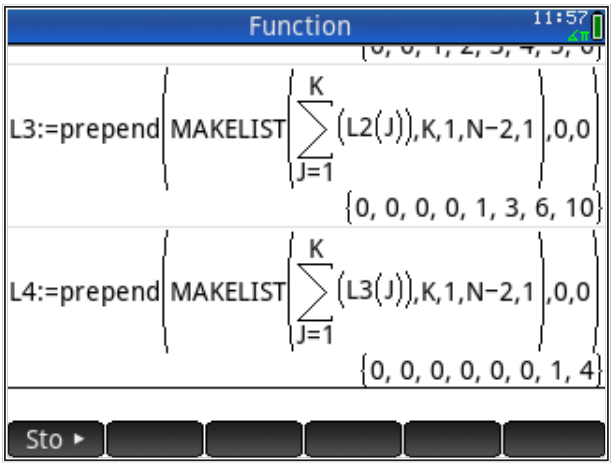
Broj elemenata koje želimo proizvesti je $N = 8$.

Format zapisa brojeva prebacujemo na 'Standard' (tipka *Settings*) da bismo sve izraze imali na ekranu.

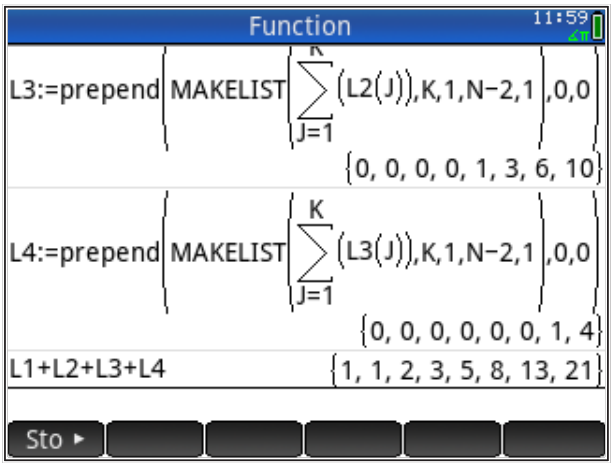
Prve dvije liste su



Druge dvije liste su (izraze za 'L3' i 'L4' najlakše dobijemo kopiranjem izraza za 'L2', gdje samo promijenimo oznake uz liste).



Sada samo trebamo zbrojiti liste parcijalnih suma



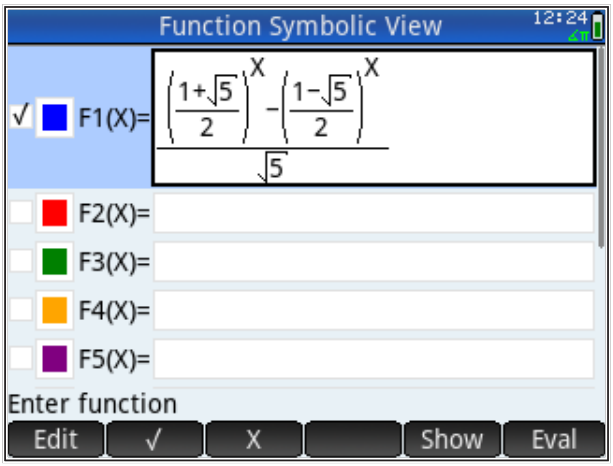
Primjer Fibonaccijevog niza preko razlike funkcija

Za izračun 'n-tih' članova Fibonaccijevog niza a_n koristimo Binetovu formulu iz 1843. godine.

$$a_n = \frac{1}{\sqrt{5}}(\Phi^n - \phi^n),$$

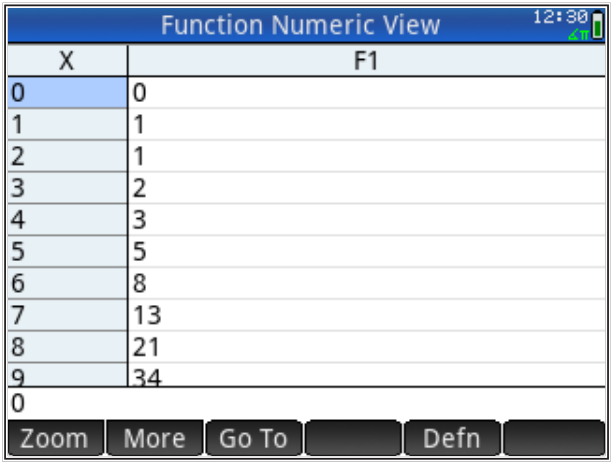
gdje je $\Phi = \frac{1+\sqrt{5}}{2}$ i $\phi = \frac{1-\sqrt{5}}{2}$

Formulu ćemo upisati u aplikaciju 'funkcije'



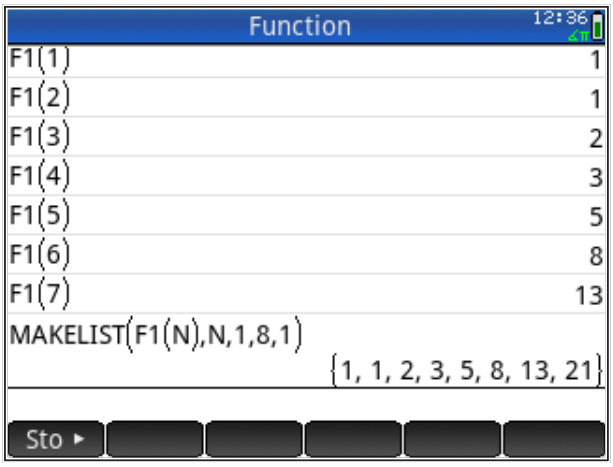
pri čemu je oznaka za eksponent 'X' a ne 'n'; to je zato jer HP Prime aplikacija 'funkcije' pretpostavlja da je ime varijable uvijek samo 'X', a u našem slučaju varijabla je broj člana niza.

Napomena: Zadanu funkciju ne možemo prikazati grafički jer nije definirana (na HP Prime) za eksponent koji nije cjelobrojni, no možemo je prikazati tablično pritiskom na tipku *Shift* *Num → Setup*.



Za prikaz smo u meniju 'zoom' odabrali cjelobrojni prikaz ('Integer').

Uz ovako definiranu funkciju 'F1(X)', listu čiji su članovi elementi Fibonaccijevog niza generiramo naredbom 'MAKELIST'.



Namjerno je kao parametar funkcije navedena varijabla 'N' kako bi se vidjelo da ime varijable može biti proizvoljno kada se poziva funkcija 'F1'; jedino kod definiranja funkcije unutar aplikacije 'funkcija' oznaka parametra treba biti 'X'.

HP Prime CAS

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide* (iz menija 'Help'):

- *Computer algebra system (CAS)*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

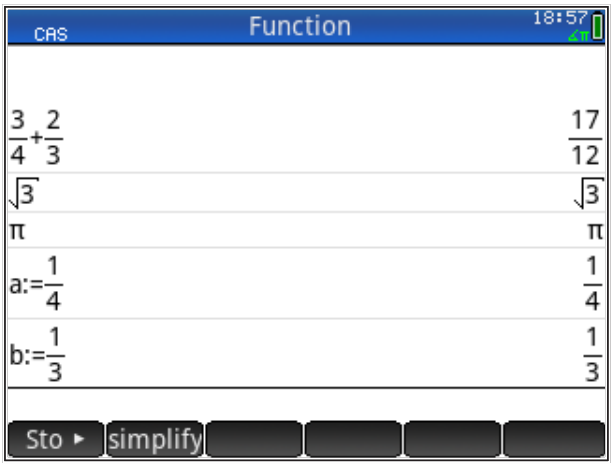
```
nema pretpostavki o stanju varijabli u memoriji
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite pomoću 'Edit: Paste' naredbi iz menija kalkulatora, pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

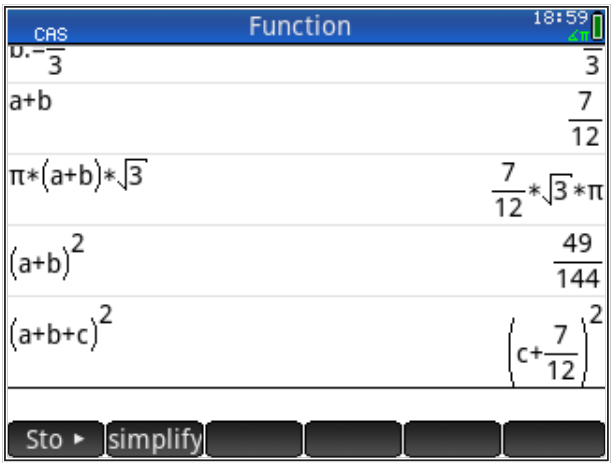
HP Prime kalkulator može izvoditi simboličke operacije nad algebarskim izrazima, tzv. CAS (Computer Algebra System). U numeričkom načinu ('Home'), kalkulator prikazuje približne rezultate (decimalni brojevi), u CAS načinu, kad je to moguće, računa s točnim vrijednostima (npr., izrazi kao razlomci, s poznatim konstantama kao π). Tako 3/4 u 'Home' načinu na ekranu daje 0,75, a u CAS načinu ostaju 3/4. Isto tako, 3/4 + 2/3 daje 17/12 (u

numeričkom načinu to je 1.41667). U CAS mode ulazimo pritiskom na tipku *CAS*, što je vidljivo preko natpisa na vrhu ekrana. U CAS načinu varijable su prikazane malim slovima, za razliku od „standardnog“ načina, gdje se nakon pritiska na tipku *ALPHA* pojavljuju velika slova.

Najznačajnije u CAS načinu jest da možemo računati sa simboličkim vrijednostima, npr.:

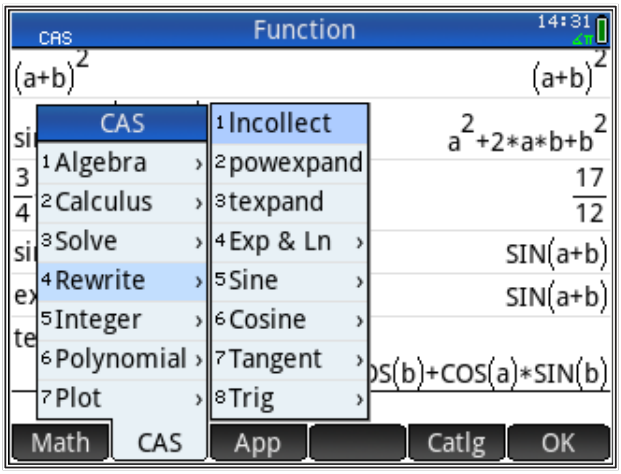


Isto tako,



Vidimo da varijabli možemo pridodati razlomak ili simboličku vrijednost. Zanimljiv je rezultat u zadnjem retku, koji je kombinacija razlomka i simbola 'c'; to je zato što varijable 'a' i 'b' imaju pridodane vrijednosti, a 'c' nema, može poprimiti bilo koju vrijednost i u rezultatu se pojavljuje kao simbol.

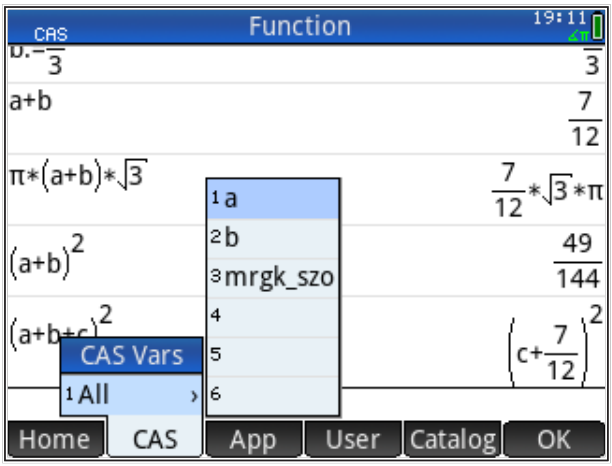
Na simboličke vrijednosti u CAS načinu možemo primijeniti razne funkcije, sve navedene na tipkama HP Prime kalkulatora, kao i još dosta drugih koje su predefinirane u njegovom operativnom sustavu. Pregled funkcija dostupnih u CAS načinu možemo dobiti pritiskom na tipku 'kovčežić' i nakon toga 'CAS'.



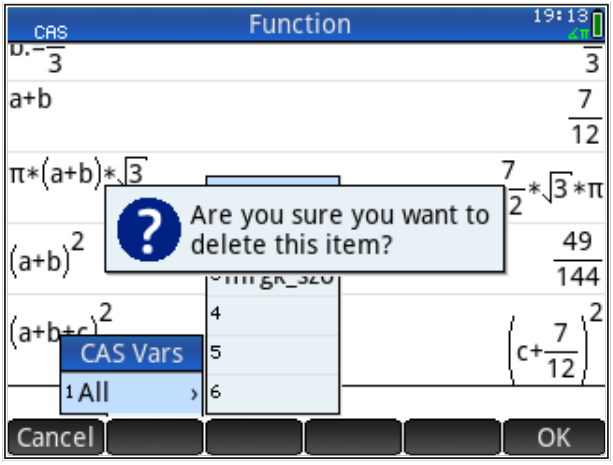
Treba napomenuti da je neke funkcije relativno teško naći; za to treba malo vježbe.

Varijable u CAS načinu

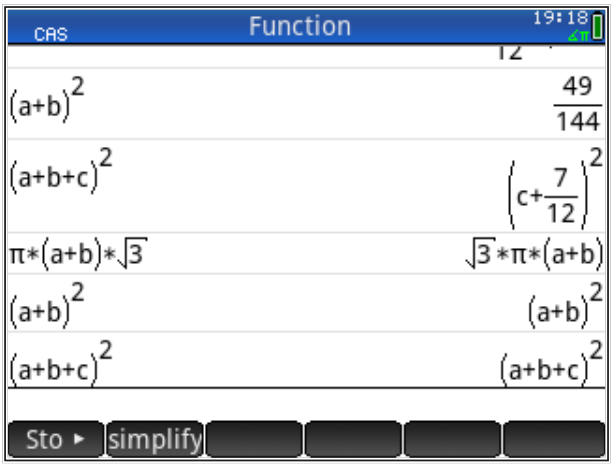
U CAS načinu varijable se ponašaju dvojako (kao što smo vidjeli na jednoj od prethodnih slika): 1) kao vrijednosti koje su spremljene u njih, 2) kao simboli kad im nije pridodana nikakva vrijednosti. Tu razliku najbolje ćemo vidjeti tako da izbrišemo vrijednost dodanu u varijablu. **Jedan od načina** je ulazak u sve varijable kalkulatora, pritiskom na tipku *Vars*. Kao što možemo vidjeti, CAS varijable nisu u korisničkim varijablama. Biramo 'CAS' i zatim 'All', gdje vidimo sve definirane varijable u CAS načinu.



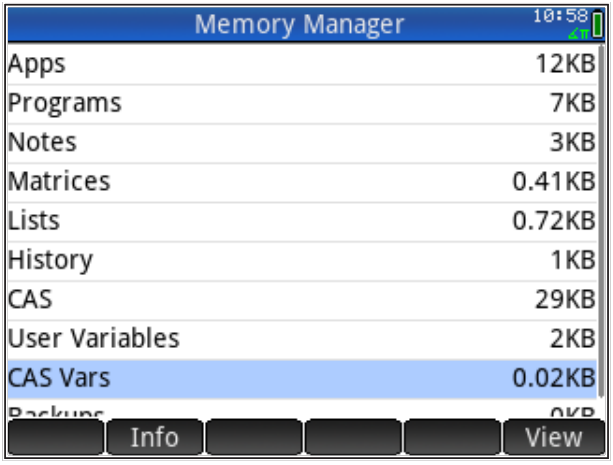
Označimo varijable koje želimo izbrisati (ovdje možemo brisati samo jednu po jednu varijablu, a za više varijabli trebamo ući u memoriju kalkulatora) i pritisnemo tipku *Del*; pojavljuje se pitanje želimo li tu varijablu izbrisati,



odgovaramo pritiskom na 'OK' u meniju i varijabla je izbrisana. Tako izbrišemo varijable 'a' i 'b'; ponovo ulazimo u CAS način (pritiskom na tipku *CAS*). Ako sada ponovimo prijašnje operacije, vidijet ćemo da je rezultat u simboličkom formatu.

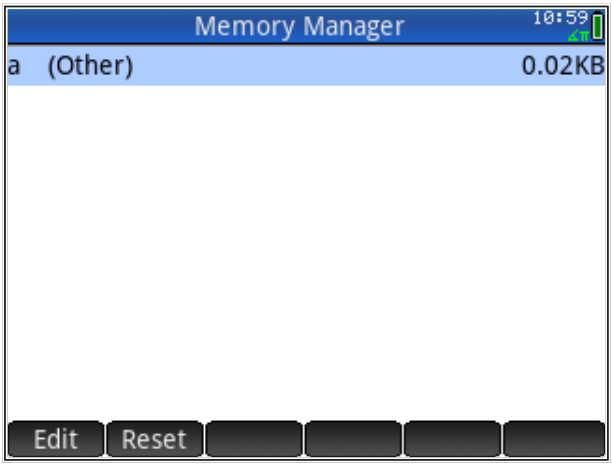


Drugi način poništavanja prethodno zadane vrijednosti u varijablu, koji omogućuje brisanje većeg broja varijabli odjednom, izvodi se ulaskom u memoriju kalkulatora, pritiskom na *Mem*.



Naravno, ako želimo vidjeti varijable 'a' i 'b' u ovom pogledu, moramo ih ponovo zadati (ako smo ih izbrisali prema prethodnim uputama).

Iz menija ispod slike biramo 'View'



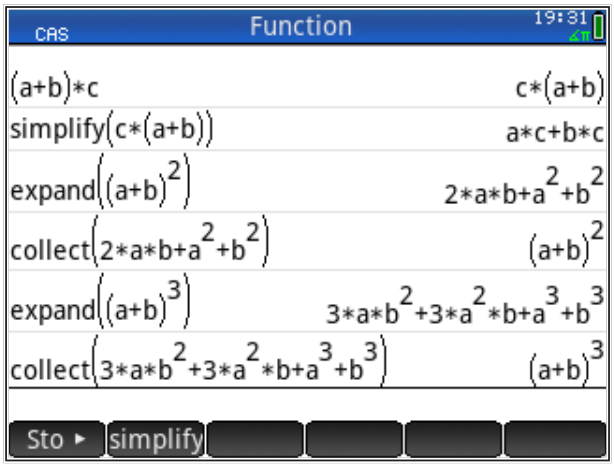
i odabirom 'Reset' u meniju ispod slike brišemo varijablu. Sada ponovljeni izračun daje isti rezultat kao i prije zadavanja vrijednosti varijable 'a'.

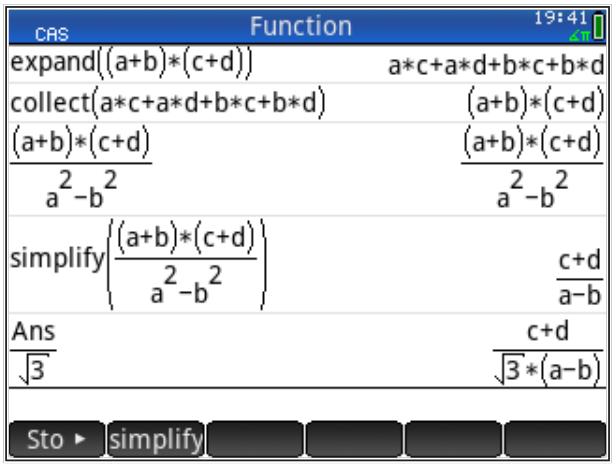
Algebarski izrazi

Na simboličke vrijednosti možemo primijeniti algebarska pravila, kao da sami računamo sa simboličkim vrijednostima. Funkcije koje najčešće koristimo su

```
'simplify' - pokušava napisati zadani izraz u jednostavnijoj formi
'expand' - izvodi naznačene operacije i raspisuje izraz
'collect' - pokušava sažeti zadani izraz grupiranjem elemenata
istog ranga
```

Naredbe se može jednostavno napisati ili odabrati iz menija (tipka 'kovčežić'). Da bi rezultati bili jednaki kao na slikama, potrebno je izbrisati vrijednosti varijabli 'a', 'b' i 'c' (npr., naredbom 'RESET', kao što je prethodno navedeno).





Naravno, moguće je kombinirati simboličke i numeričke vrijednosti. Na slici vidimo da naredbom 'expand' razvijamo binom na kvadrat, ali da ga ne možemo na isti način sažeti natrag u polaznu formu. Za to koristimo naredbu 'collect', koja grupira izraz u kompaktnu formu.

Vidimo i primjer s varijablom 'Ans', koja automatski pamti zadnji rezultat (poziva se pritiskom na tipku *Ans*, a u nekim situacijama kalkulator je poziva automatski).

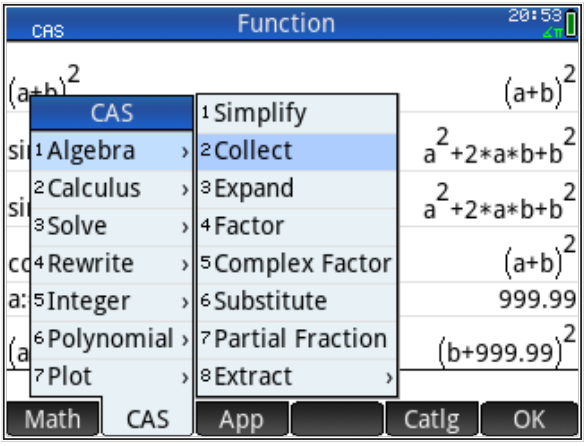
Polinomi

Prikazat ćemo i često korištene operacije pojednostavljivanja polinomima. Umnožak dva polinoma

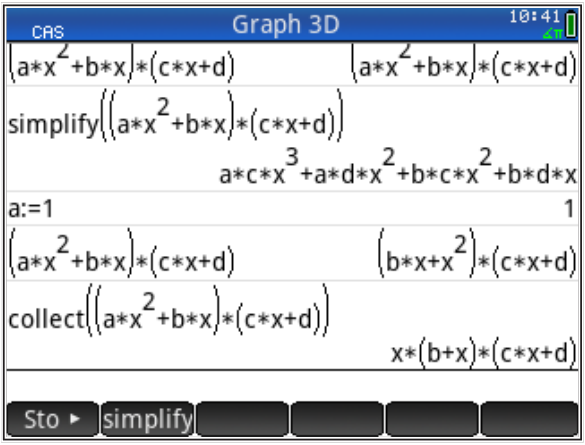
$$(a \cdot x^2 + b \cdot x)(c \cdot x + d)$$

želimo prikazati u jednostavnijem obliku ako je $a = 1$, a koeficijenti b, c, d su neodređeni.

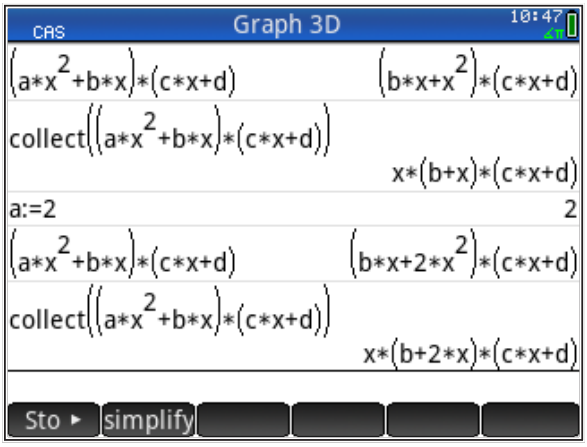
Kalkulator prebacujemo u CAS način rada (tipka 'CAS'). Upotrijebit ćemo naredbu 'collect', koja grupira izraze istog ranga; možemo je dobiti tipkanjem ili izborom iz kataloga CAS operatora (pritisnemo 'kovčežić', pa 'CAS', pa 'Algebra').



Primjer upotrebe vidimo na donjoj slici



Ukoliko promijenimo vrijednost varijable 'a' i ponovimo izračun, dobit ćemo nove, odgovarajuće rezultate

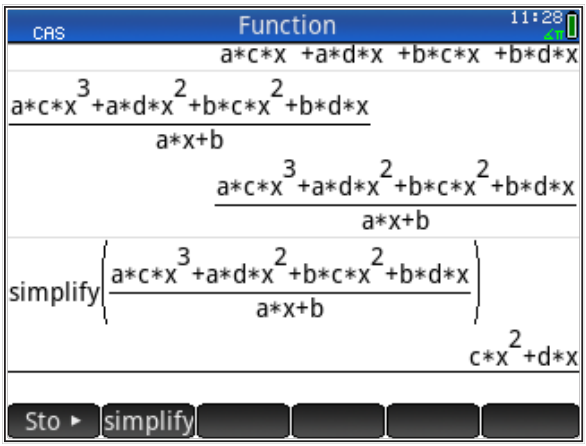


Naravno, ako izbrišemo vrijednost pridodanu varijabli 'a', nju će kalkulator ponovo tretirati kao simbol.

Podijelimo dva polinoma (prvo izbrišimo varijablu 'a')

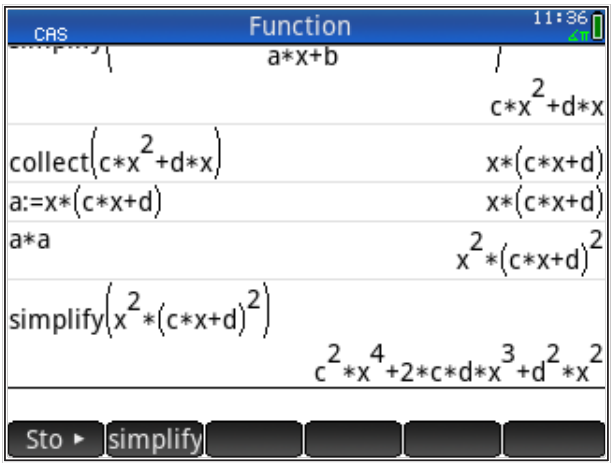
$$\frac{acx^3+adx^2+bcx^2+bdx}{ax+b}$$

Primjećujemo da je polinom u brojniku rezultirajući razvijeni produkt, koji smo prethodno dobili množenjem dva polinoma. Također, polinom $ax + b$ je skriven u prvom izrazu produkta dva polinoma, a vidljiv ako izlučimo 'x'; zato će se polinomi podijeliti bez ostatka. Rezultat je



Napomena: U prethodnim primjerima poželjno je koristiti naredbu 'Copy', a ne prepisivati dugačke izraze.

Izraz se dalje dade „dotjerati“ naredbom 'collect'

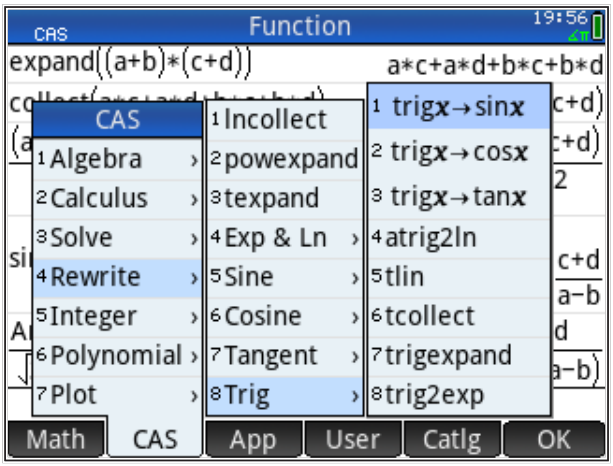


Usput smo dobiveni rezultat (polinom $x(cx + d)$) spremili u varijablu 'a' te dalje umjesto polinoma $x(cx + d)$ možemo pisati 'a'.

Trigonometrijski izrazi

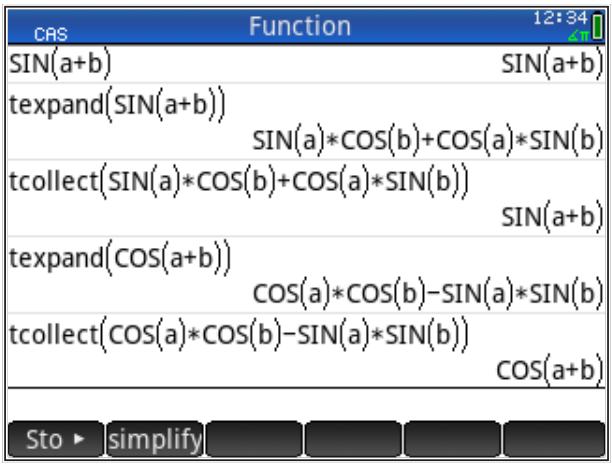
Algebarske funkcije uglavnom ne rade dobro s trigonometrijskim izrazima; za rad s njima postoje posebne trigonometrijske funkcije. Obično se od algebarskih razlikuju samo po dodanom slovu 't' ispred imena. Tako imamo

'trigexpand' - daje prošireni trigonometrijski izraz
'texpand' - izvodi naznačene operacije i raspisuje izraz
'tcollect' - pokušava sažeti zadani trigonometrijski izraz grupirajući zajedno izraze SIN i COS s istim argumentom

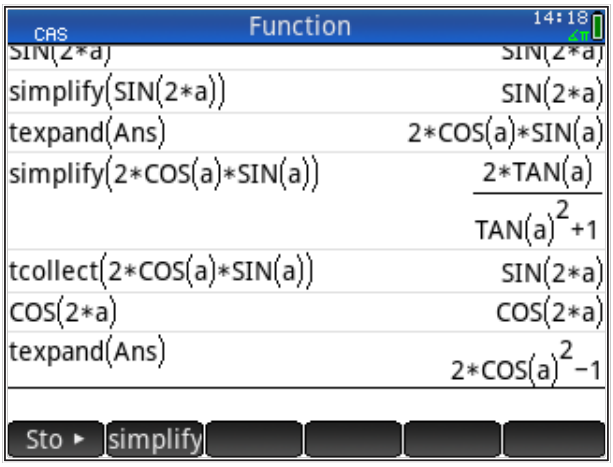


Naravno, kao što vidimo, u CAS načinu rada ima još i drugih trigonometrijskih funkcija.

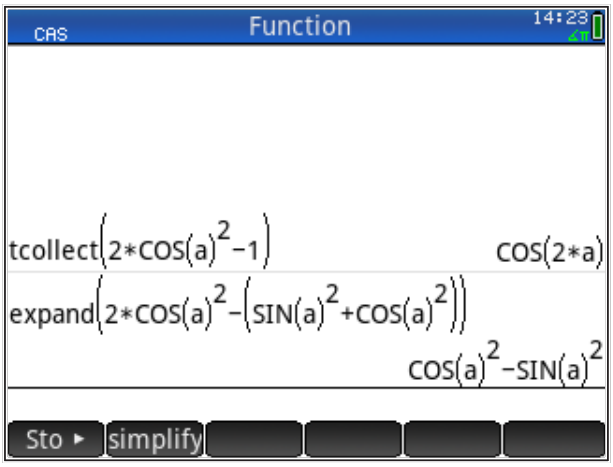
Pogledajmo primjer za razvijeno pisanje sinusa sume $\sin(a+b)$; samo funkcija 'texpand' daje ispravan rezultat ('texpand' je skraćeno od 'trigonometric expansion'). Funkcija 'tcollect' sažet će izraz natrag u formu sume. Usput je prikazan i primjer za kosinus sume $\cos(a+b)$. Da bi primjer ispravno radio, prije treba izbrisati varijable 'a' i 'b'.



Pogledajmo još neke primjere trigonometrijskih transformacija (iz logaritamskih tablica).



Iz slike je vidljivo da trigonometrijske izraze možemo transformirati s 'texpand' i vratiti (pojednostaviti) i sa 'simplify'. No, $\cos(2a)$ ne uspijevamo napisati kao $\cos(2a) = \cos^2(a) - \sin^2(a)$; za to ipak moramo znati $\sin^2(a) + \cos^2(a) = 1$ i ručno uvrstiti umjesto broja '1' te primijeniti funkciju 'expand'.

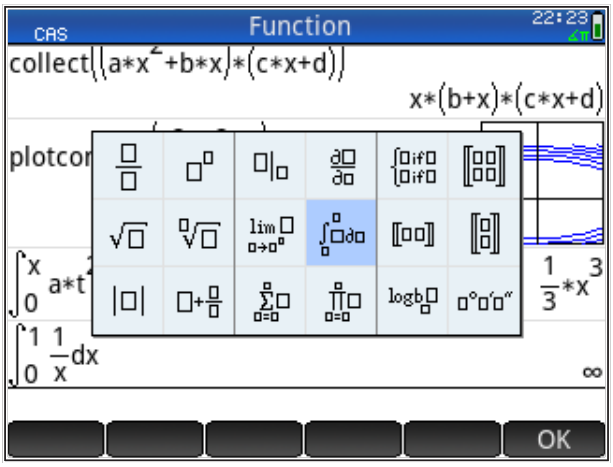


Taj rezultat ne možemo postići funkcijom 'simplify'; dakle, trebamo znati što želimo postići i imati određenu vještinu baratanja matematičkim izrazima (što se postiže vježbom, kao i kod rješavanja „na ruke“).

Integriranje

Za ilustraciju simboličkog integriranja prebacujemo kalkulator u CAS način. HP Prime može rješavati neodređene i određene integrale.

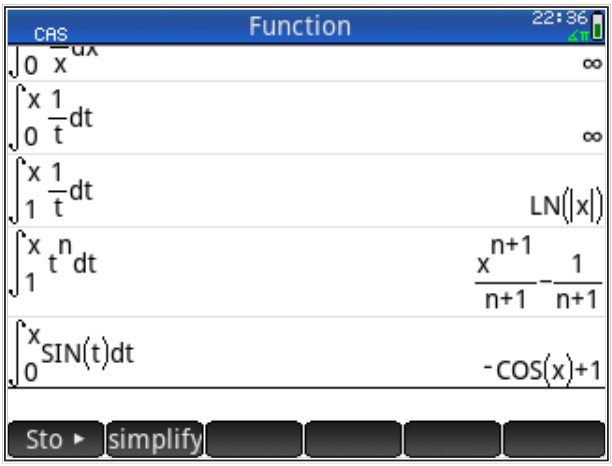
Određene integrale (s definiranim granicama integracije) najzornije možemo predočiti kao računanje površine (ili volumena). Znak za integriranje najlakše se upisuju preko tipke za predloške, nakon čega u predlošku popunimo prazne kvadratiće.



Npr., $\int_0^1 1/x dx$ daje kao rezultat beskonačno ∞.

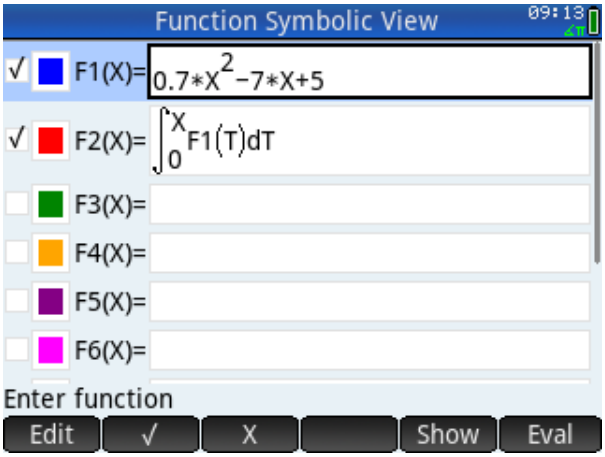
Naročito je važno definirati granice integracije u skladu s tijekom funkcije.

Ukoliko granice integracije zadajemo preko varijable x, možemo dobiti simboličko rješenje; samo treba prilagoditi varijable. Najčešće, za granice integracije se uzimaju vrijednosti od 0 ili 1 do x, a integriranje treba uslijediti prema nekoj internoj varijabli, npr. 't', kao na slici (za neke integrale može se javiti poruka 'No checks were made for singular points of antiderivative ...'; u tom slučaju, samo treba pritisnuti 'ENTER').

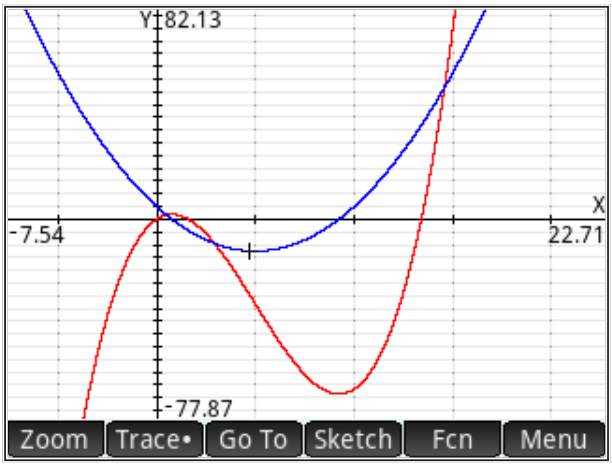


Vidimo da HP Prime poznaje uobičajena pravila za integriranje; samo moramo malo pripaziti kod interpretacije konstante integriranja, koja se može pojaviti u neuobičajenim oblicima.

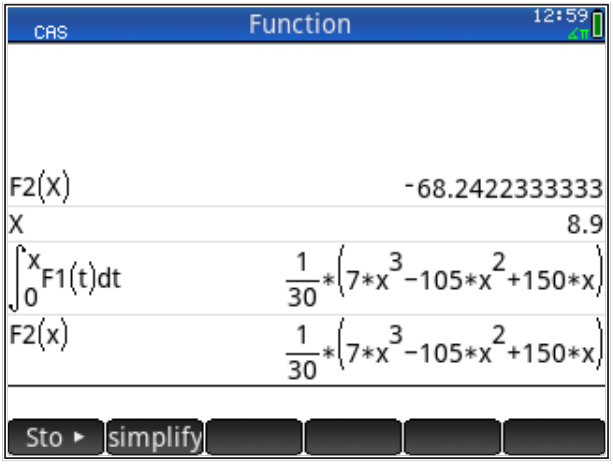
Drugi način integriranja je preko aplikacije 'funkcije'. Definirajmo funkciju $F1(X) = 0.7X^2 - 7X + 5$, a funkcija $F2(X)$ neka je integral funkcije $F1(X)$. Ukoliko ga želite zadržati prije definirane funkcije, onda njih možete definirati kao $F7(X)$ i $F8(X)$.



Grafički je to



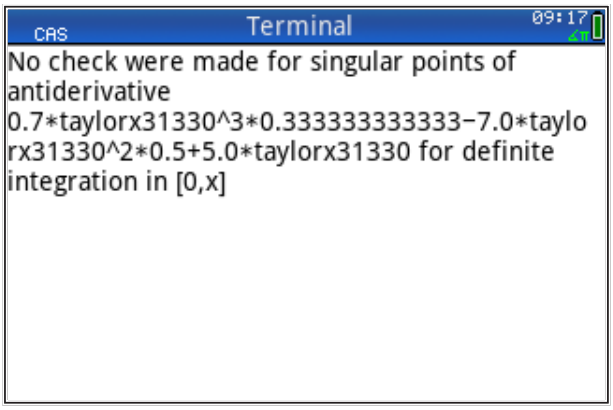
Vrijednost funkcije $F2(X)$ možemo dobiti i numerički i simbolički. Prebacujemo se u CAS način i tipkamo $F2(X)$ (velika slova u CAS načinu dobijemo pomoću tipki *ALPHA* *Shift*); rezultat nije simbolički, kako bismo možda očekivali, nego numerički.



Ako pogledamo prethodnu sliku s grafikom, vidimo da je varijabla X tamo definirana; to je vrijednost koju je kalkulator preuzeo i uvrstio (kao što je razvidno iz slike).

Simboličku vrijednost dobijemo tako da ponovimo izračun integrala u CAS načinu. Pritom, umjesto varijable veliko 'X' trebamo pisati malo 'x' (koje nije

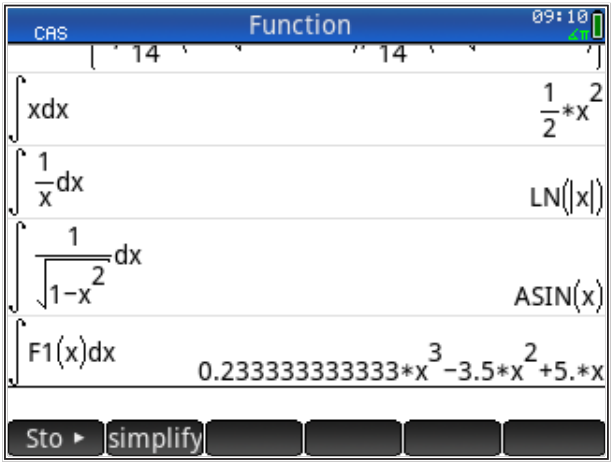
definirano pa ostaje kao simbol); tako i s F2(x) dobivamo simbolički rezultat. Pri izračunu u CAS načinu dobivamo upozorenje kao na slici



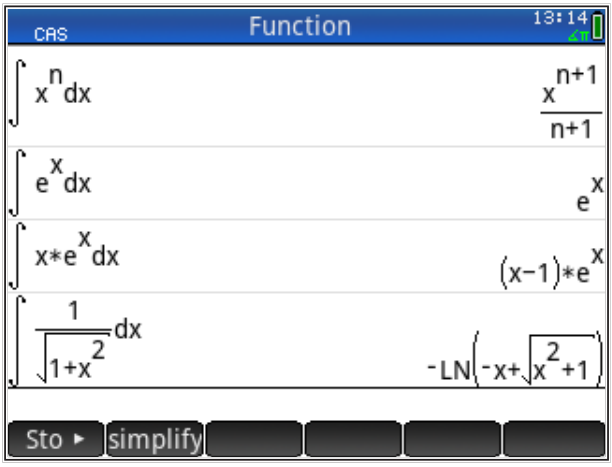
Upozorenje jednostavno znači da kalkulator nije provjerio postoji li singularna točka funkcije u području u kojem se vrši integriranje (pri čemu je napomenuto da se račun vrši tako da se funkcija razvija u Taylorov red, ali to nama nije bitno).

Još jednom smo vidjeli da kalkulator razlikuje mala i velika slova kao imena varijabli i dobra je praksa koristiti velika slova u numeričkom, a mala u CAS načinu (kao što i sam kalkulator sugerira pisanjem varijabli u *ALPHA* načinu).

Neodređeni integrali rješavaju se izborom 'Calculus>', 'Integrate' iz CAS izbornika. Na taj način možemo rješavati i neodređene, ali i određene integrale (jer je upis granica integracije po volji; ukoliko nisu zadane, rješava se neodređeni integral). Npr.,



Još nekoliko primjera neodređenih integrala:



HP Prime „znade“ i osnovni teorem integralnog računa, a to je da je integral od derivacije funkcije jednak samoj funkciji, $\int \frac{dy}{dx} = y$.

Međutim, upotrebom funkcije za integriranje ne zna izračunati složeni integral tipa $\int g \cdot f' dx$. Integriranje takvih složenih funkcija provodi se preko naredbi za parcijalnu integraciju, 'ibpu“ i 'ibpv.

Funkcija 'ibpu' računa izraze tipa $\int u(x) \cdot dv'(x) = u(x)v(x) - \int v(x) \cdot du(x)$, dok funkcija 'ibpv' računa izraze tipa $\int v(x) \cdot du'(x) = v(x)u(x) - \int u(x) \cdot dv(x)$.

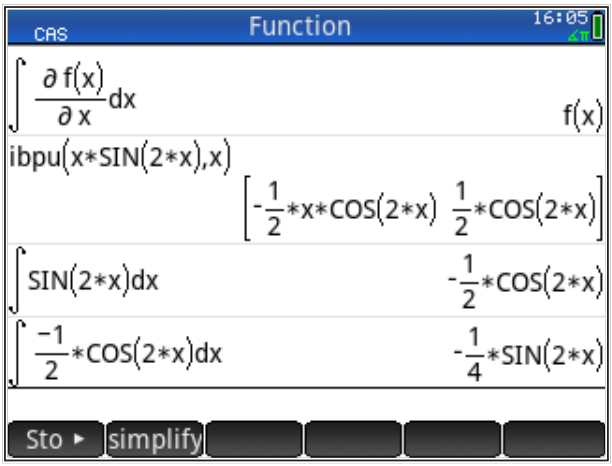
Rezultat ovih funkcija je vektor

$$\begin{bmatrix} u(x) \cdot v(x) \\ v(x) \cdot u'(x) \end{bmatrix}$$

Za konačan rezultat, još treba integrirati drugi član vektora!

Primjer

Izračunajmo integral $\int x \cdot sin(2x) dx$ gdje je $u(x) = x$ i $v'(x) = sin(2x)$. Rješenje je $-\frac{1}{2}xcos(2x) - \int -\frac{1}{2}cos(2x)dx$, odnosno $-\frac{1}{2}xcos(2x) + \frac{1}{4}sin(2x)$ jer je $\int sin(2x)dx = -\frac{1}{2}cos(2x)$ i $\int -\frac{1}{2}cos(2x)dx = -\frac{1}{4}sin(2x)$.



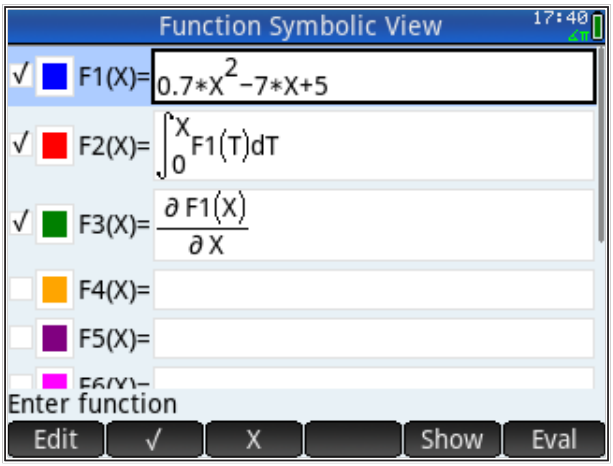
Ukratko, vidimo da kompletno rješenje dobivamo kad u vektoru 'rezultat' (nalazi se u 'Ans'), nakon primjene 'ibpu', drugi član integriramo i pribrojmo prvom članu vektora.

Deriviranje

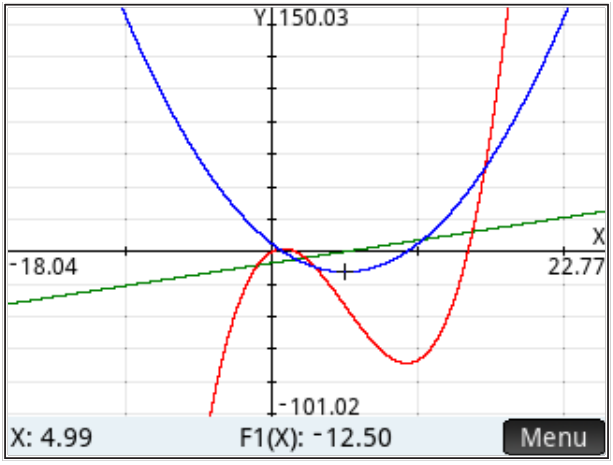
Ilustrativno je deriviranje predočiti grafički, kao traženje tangente na neku funkciju. Prtom, tangentu možemo definirati a) numerički i b) simbolički, tj., a) kao vrijednost nagiba (kuta) tangente u nekoj točki na funkciji ili b) kao definiciju nove funkcije, koja za svaki x daje vrijednost nagiba tangente na mjestu x.

Numeričke derivacije su snažan alat za rješavanje mnogih inžinjerskih problema i njima ćemo se baviti u knjizi o predmetu *Računalno modeliranje* (na diplomskome studiju). Ovdje ćemo prikazati kako dobiti simboličke derivacije funkcija koje zadajemo u aplikaciji 'Funkcije'. Na taj način možemo npr., odrediti ekstreme funkcije (maksimum i minimum), tj., rješavati jednostavne probleme optimizacije bez zadanih uvjeta.

Definiramo funkcije u aplikaciji kao na slici (HP Prime za oznaku derivacije $\frac{dy}{dx}$ koristi simbol kao za parcijalnu derivaciju $\frac{\partial y}{\partial x}$):

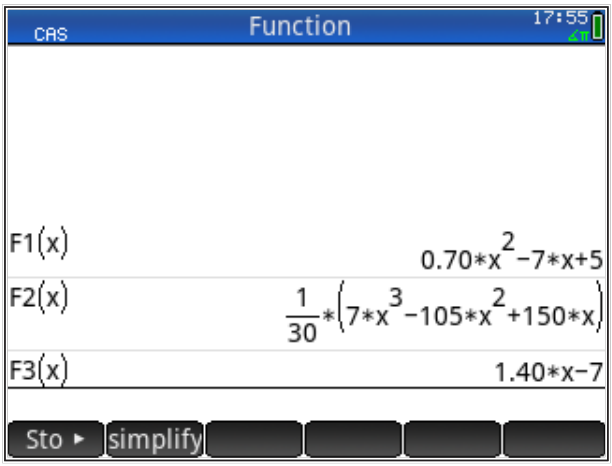


Dakle, zadali smo samo jednu funkciju, F1(X), kao kvadratnu parabolu. F2(X) i F3(X) su integral i derivacija F1(X). Sve tri funkcije možemo grafički prikazati (obratite pažnju na boju svake funkcije):

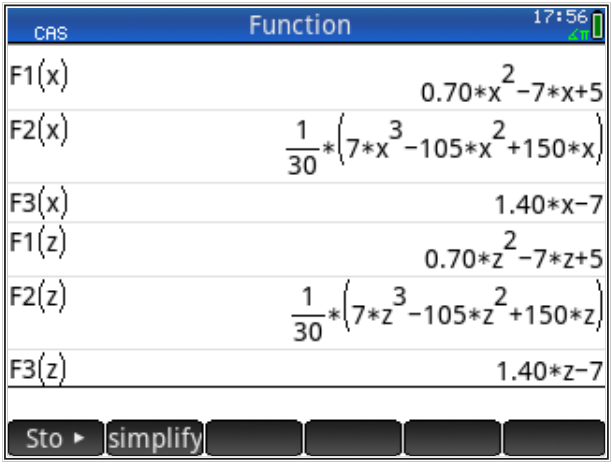


Napomena: Ako je F1(X) kvadratna parabola, onda ima jedan ekstrem, a njezin integral je kubna parabola s dva ekstrema; derivacija je pravac.

Vidimo da smo samo F1(X) zadali analitički (formulom), ali HP Prime svejedno može izračunati analitički izraz za integral, funkciju F2(X) i derivaciju te funkciju F3(X). Moramo se prebaciti u CAS način (pritiskom na tipku *CAS*).



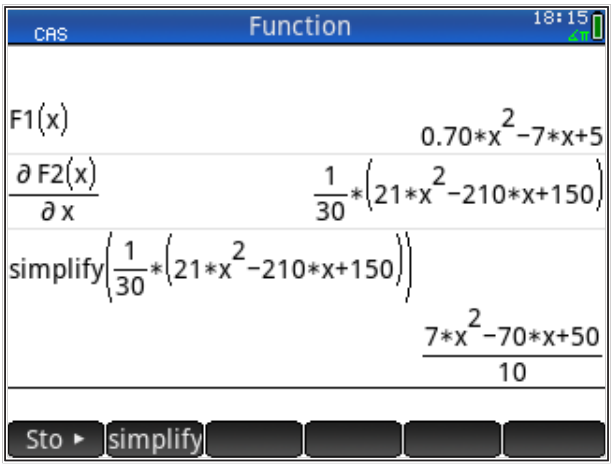
Primijetimo malo slovo 'x' kao argument u CAS načinu (pri čemu varijabli 'x' nije pridodana nikakva vrijednost). Još jednom naglašavam razliku između velikih i malih slova i argumenata u numeričkom i CAS načinu rada. Naravno, ime slobodne (neinicijalizirane) varijable u CAS načinu potpuno je proizvoljno; to je samo simbol, što se zorno vidi na slici.



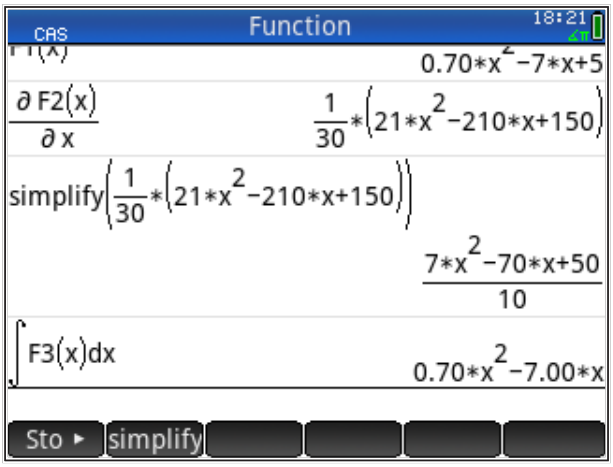
Umjesto 'x' napisali smo 'z', no mogli smo staviti i bilo koji drugi simbol.

Možemo provjeriti i temeljni teorem integralnog i diferencijalnog računa $F(x) = \int_0^x f(t)dt$ gdje je $f(x) = \frac{dF}{dx}$.

Prema tom teoremu treba biti F2'(x) = F1(x).



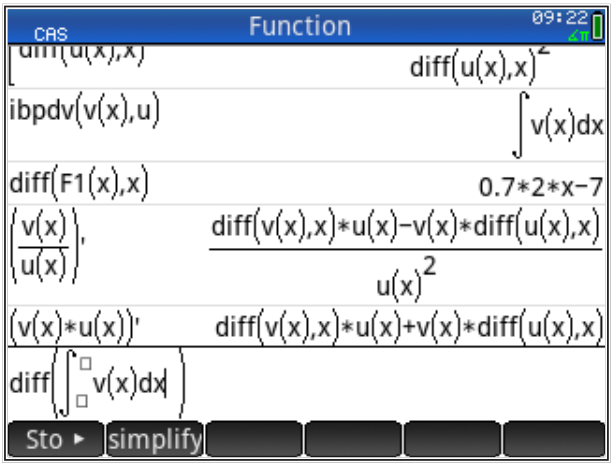
Isto tako, treba biti $\int F3(x) = F1(x)$.



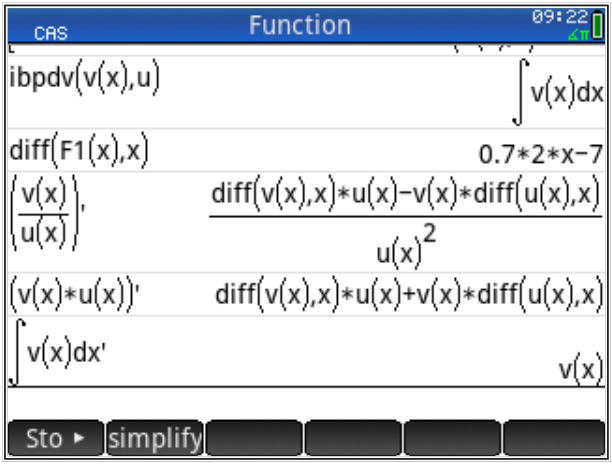
Vidimo da je ovaj uvjet zadovoljen samo do razine konstante, tj., dobili smo kvadratnu parabolu koja je translatirana originalna parabola. Jasno vidimo značaj konstante kod neodređenog integrala (konstanta se određuje iz uvjeta problema koji je opisan jednadžbom).

Zadatak: Prikaži grafički na istoj slici $F1(X)$ i $\int F3(X)$ i uvjeri se da krivulje imaju isti oblik, samo su translatirane po osi X.

Možemo pokazati da HP Prime poznaje pravila za deriviranje složenih funkcija:



Treba paziti jer, nakon što se upiše zadnji izraz, on dobiva drugačiji zapis.



Zapis treba interpretirati kao da je oko integrala zagrada, a apostrof koji označava deriviranje djeluje na sve u zagradi (zagrada postaje vidljiva ako se integral odabere klikom na njega i kopira).

Rješavanje jednadžbi

HP Prime može rješavati jednadžbe i diferencijalne jednadžbe na mnogo načina. Ovdje ćemo samo pogledati mogućnosti koje pruža *funkcija* 'solve' u izborniku CAS, a *aplikaciju* 'Solve' obradit ćemo kasnije. Funkcija 'solve' ima dvije inačice: 'solve', koja (u pravilu) radi s razlomcima, i 'fsolve', koja radi s pomičnim zarezom. Za potrebe praktičnog izračuna preferiramo 'fsolve', a za simbolički rezultat (ako postoji) koristimo 'solve'. Sintaksa im je ista ('solve' ili 'fsolve' tipkamo ili biramo iz izbornika CAS funkcija),

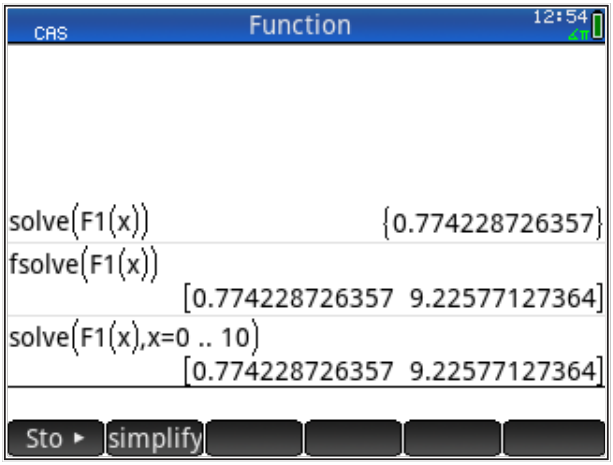
```
solve(F1(x),x=(interval rješenja))     Enter
```

s time da je zadavanje intervala opcija; rezultat će se dobiti i bez toga, ali nećemo biti sigurni jesu li to sva rješenja koja nas mogu zanimati.

Npr., za prije definiranu funkciju F1(X) za interval (0 ... 10) dobivamo dva rješenja kvadratne jednadžbe $x = 0.7742$ i $x = 9.2258$, pri čemu je rješenje u obliku vektora. Prije rješenja dobili smo upozorenje da je F1(X) definiran kao funkcija, a ne kao jednadžba i da će kalkulator od funkcije napraviti jednadžbu tako da joj vrijednost izjednači s nulom, tj., rješavat će jednadžbu F1(X)=0. Tu poruku mogli smo izbjeći tako da smo napisali:

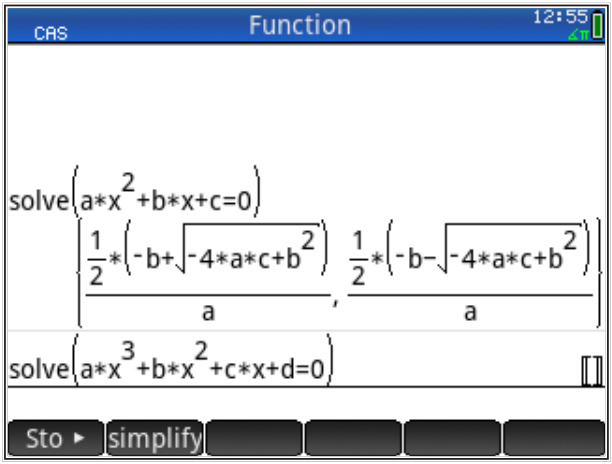
```
solve(F1(x)=0,x=(0 .. 10))     Enter
```

Funkcija 'fsolve' ne daje poruku i ne treba dopisivati '=0'.



Uvidom u grafički prikaz funkcije vidimo da postoje dva realna rješenja, jedno malo veće od nule (x=0.77) i drugo malo manje od 10 (x=9.23). Primijetimo da je numeričko rješenje u obliku vektora (uglate [] zagrade), a 'solve' daje rješenja i u obliku liste (vitičaste {} zagrade).

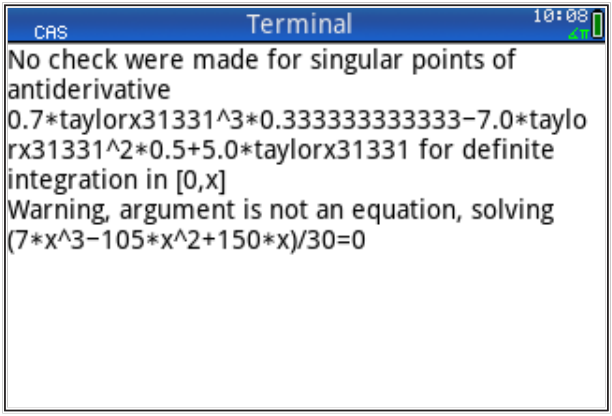
Ta smo rješenja mogli dobiti i bez kalkulatora jer rješenje kvadratne jednadžbe postoji u obliku formule. Malo je više posla ako želimo riješiti jednadžbu F2(X)=0 jer je to kubna jednadžba i rješenje ne postoji u obliku formule (rješenje postoji u obliku postupka/algoritma).



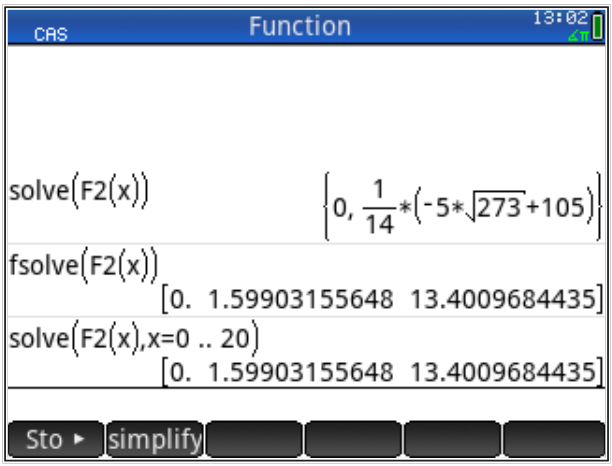
Tipkanjem

```
solve(F2(x))     Enter
```

prvo dobivamo upozorenje,

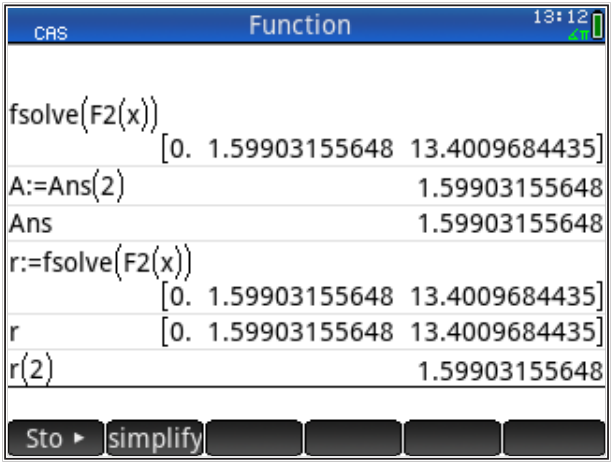


a pritiskom na 'Enter' dobivamo samo dva rješenja (u obliku {liste}), dok 'fsolve' daje sva tri rješenja (u obliku [vektora]). Dodavanjem u 'solve' intervala u kojem želimo rješenje, rezultat postaje isti kao i kod funkcije 'fsolve'.



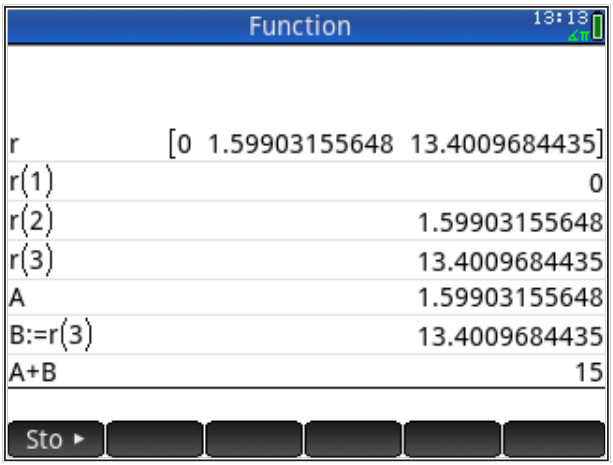
Vidljivo je da je kalkulator prvo izračunao integral, a zatim riješio dobivenu kubnu jednadžbu. U daljnjem radu, radi jednostavnosti, koristit ćemo samo funkciju 'fsolve'.

Ukoliko želimo dalje računati s dobivenim rješenjima, treba ih spremiti u varijable. Ako ćemo nastaviti rad u numeričkom načinu kalkulatora, praktično je rezultat spremiti u varijablu s velikim slovom. Npr., spremit ćemo drugo rješenje u varijablu A i treće rješenje u varijablu B.



Vidimo da za kopiranje jednog rješenja možemo koristiti varijablu 'Ans', koja nakon toga mijenja vrijednost u varijablu koju smo kopirali (zadnja vrijednost na ekranu).

Za kopiranje više rješenja, potrebno je cijeli rezultat kopirati u neku varijablu (na slici, to je 'r', da nas podsjeti na 'rješenje'), a zatim se dalje može raditi s tom varijablom i u CAS i u numeričkom načinu.

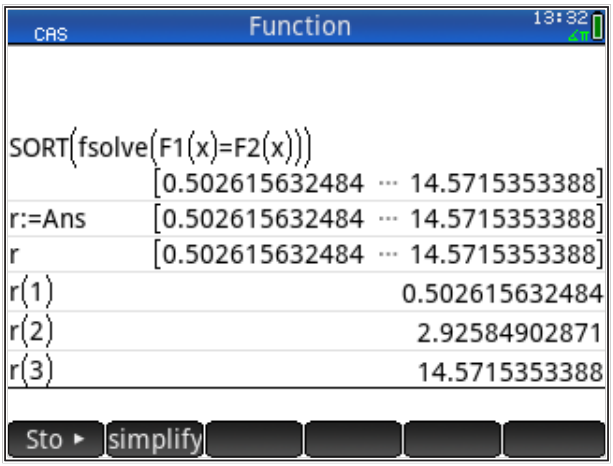


Zadatak: Provjeri rezultat tako da simbolički izračunaš integral F2(x) i riješiš dobivenu jednadžbu!

Presjecište krivulja

Funkcija 'fsolve' može rješavati i sistem polinomijalnih jednadžbi, odnosno, može tražiti presjecišta krivulja opisanih polinomima.

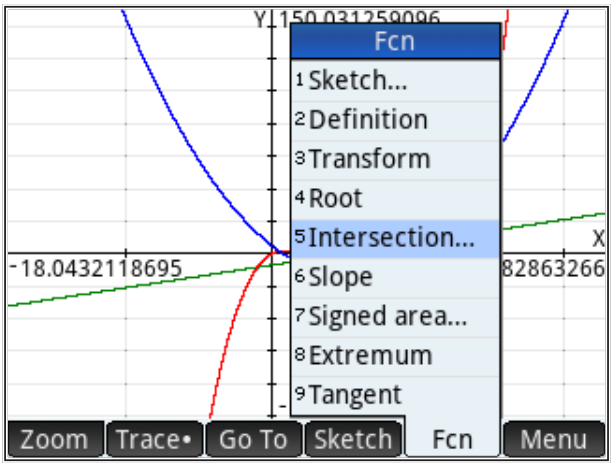
Prebacujemo se u način i rješavamo sistem jednadžbi F1(x)=F2(x); rezultat bi trebale biti zajedničke (presjecišne) točke obaju krivulja. S obzirom da bi kao rezultat trebali dobiti vektor s tri točke, spremit ćemo ga u varijablu 'r' da ga možemo pregledati (izraz je predugačak za ekran).



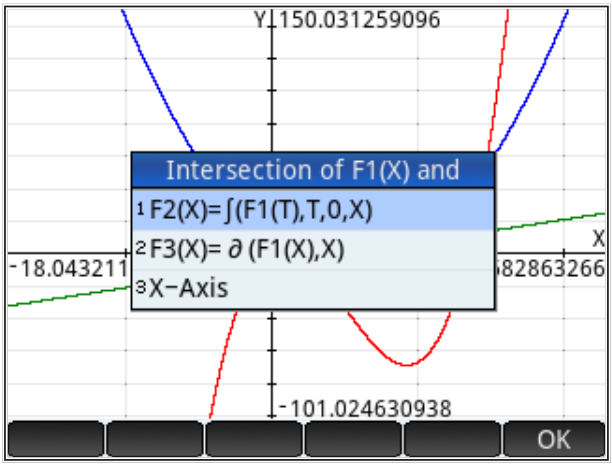
Na slici vidimo da smo vektor rezultata spremili u varijablu 'r'. Budući da 'fsolve' rezultate daje u nepredvidivom redoslijedu, prije spremanja rezultate u vektoru sortirali smo po veličini (naredba 'SORT').

Dobiveni rezultati su samo koordinate 'X' presjecišta; koordinatu 'Y' dobijemo uvrštavanjem rezultata 'X' u F1(X) ili F2(X) te moramo dobiti isti 'Y' jer su to zajedničke točke obaju krivulja.

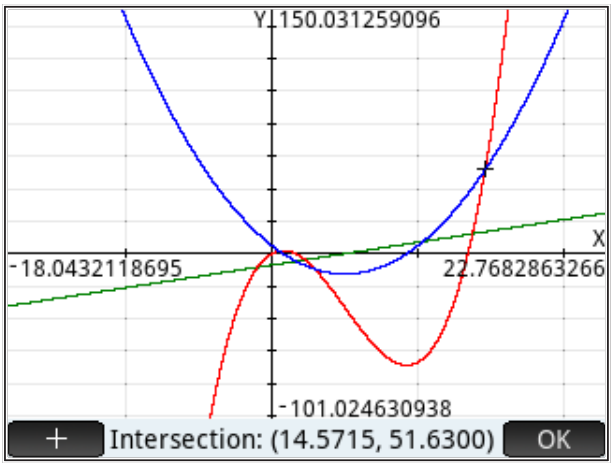
Iste presjecišne točke možemo dobiti i iz grafičkog prikaza funkcija F1(X) i F2(X), koristeći naredbu 'Intersection'.



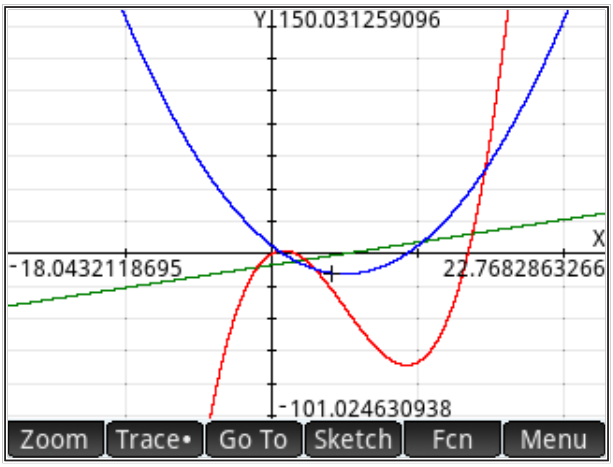
Biramo presjecište F1(X) sa F2(X),



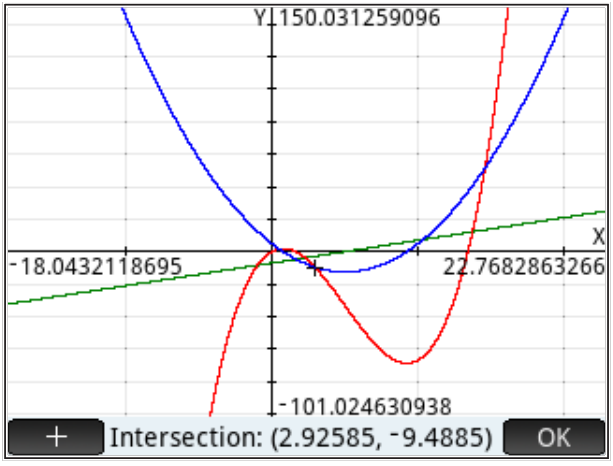
i kao rezultat dobivamo točku (X=14.57,Y=51.63). Primijetimo da je to treće rješenje u prije nađenom vektoru rješenja 'r'.



Drugo rješenje možemo naći tako da koordinatu 'X' pomaknemo na mjesto bliže presjecištu koje tražimo, npr. (uoči križić koji je sada bliže presjecištu oko točke X=3).



Ponovimo li postupak koristeći naredbu 'intersection', dobivamo presjecišnu točku (X=2.93,Y=-9.49), a koordinata 'X' odgovara drugom rješenju iz vektora 'r'.



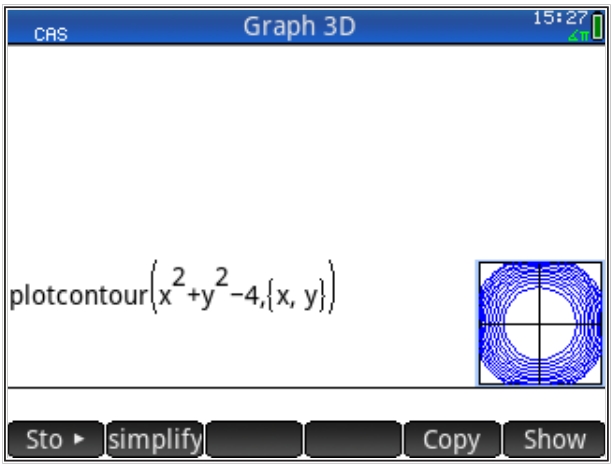
Na isti način možemo naći i presjecište najbliže točki X=0, odnosno, X=0.50.

Ostalo

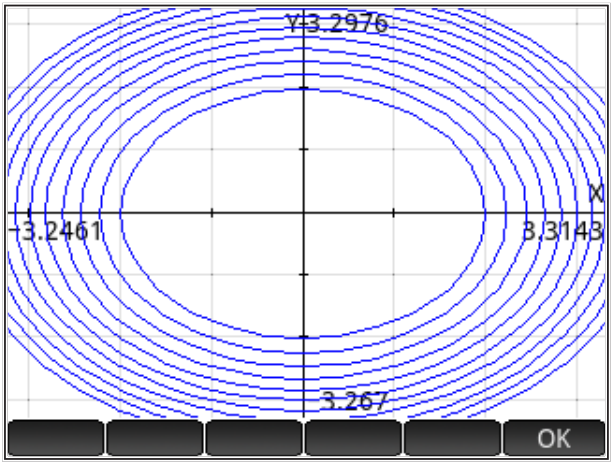
Ostale upotrebe CAS sustava pogledat ćemo u ovom poglavlju samo ukratko. Zanimljiva je mogućnost crtanja kontura krivulje naredbom 'plotcontour()', gdje je neka funcija. Sintaksa je

```
plotcontour(funkcija,{varijable},{x=korak},{y=korak})
```

pri čemu su koraci x i y opcionalni (ne moraju se zadati).



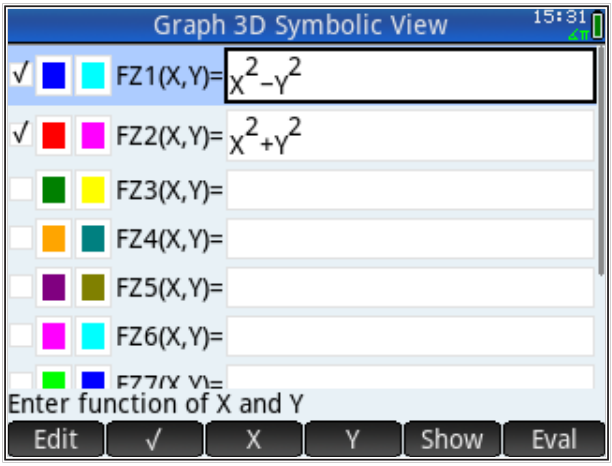
Odabirom slike i pritiskom na 'Show' dobivamo sliku grafike preko cijelog ekrana.



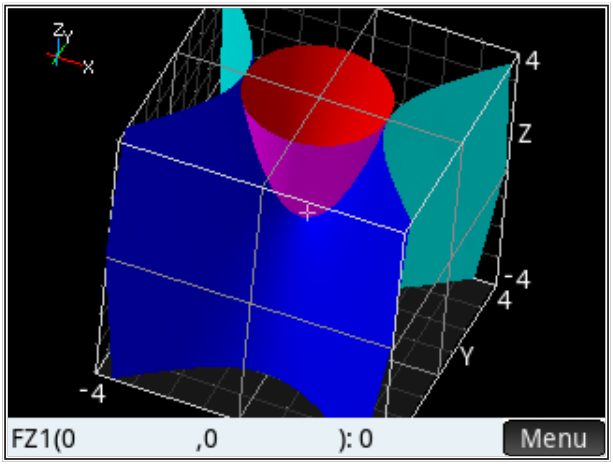
Iz slike izlazimo pritiskom na tipku 'Esc' ili odabirom 'OK' iz menija iznad ekrana.

Napomena: Funkcija 'plotcontour' ponekad daje nepredvidljive rezultate.

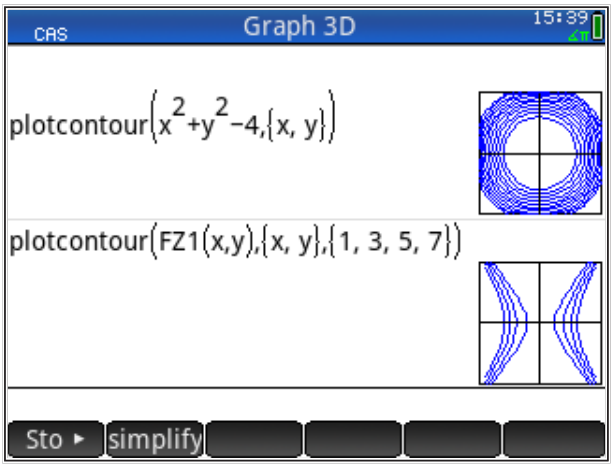
Funkcije koje crtamo s 'plotcontour' mogu biti zadane i u aplikaciji 'Graph 3D' (koja nije dostupna u besplatnoj aplikaciji za smartphone, ali jest za PC).



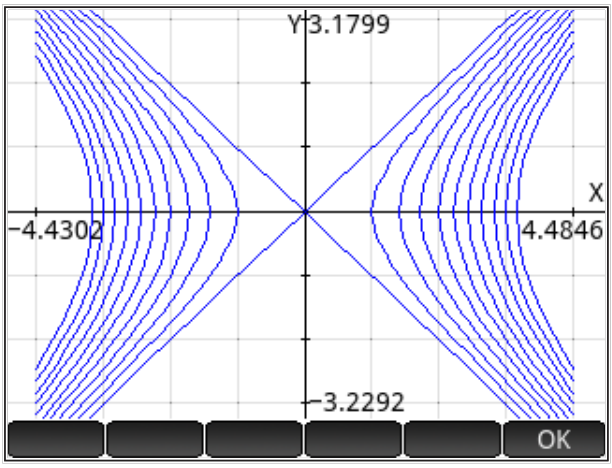
U 3D prikazu vidimo da je FZ1(x,y) hiperboloid, a (FZ2(x,y) je uspravni valjak i crtež će prikazati obje funkcije).



Gustoću kontura možemo regulirati zadavanjem koraka za vrijednost u kojoj će se nacrtati. Npr., konture hiperboloida za zadane vrijednosti funkcije FZ1(x,y), tj., z={1,3,5,7}



Konture hiperboloida FZ1(x,y) izgledaju (nakon pritiska na 'Show') ovako:



Osim ovih, postoje i druge mogućnosti grafičkog prikaza funkcija, ali ih ovdje nećemo posebno obrađivati.

HP Prime rješavanje jednadžbi i nejednadžbi

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide* (iz menija 'Help') :

- *Primary apps: Function app*
- *Primary apps: Advanced Graphing app*
- *Primary apps: Graph 3D app*
- *Primary apps: Solve app*
- *Solver apps: Linear Solver app*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

```
nema pretpostavki o stanju varijabli u memoriji
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite s 'Edit: Paste' naredbama iz menija kalkulatora pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

HP Prime kalkulator nudi više načina rješavanja pojedinačnih jednadžbi i sistema jednadžbi. Pritom, neke aplikacije daju numerička rješenja (rješenje je samo broj, x =...), a neke aplikacije daju simbolička rješenja (rješenje je funkcija parametara jednadžbe, $x = f(a,b,...)$).

Podsjetimo se razlike između rješavanja jedne jednadžbe i sistema jednadžbi.

Jedna jednadžba nastaje kada postavimo uvjet da funkcija jedne nepoznanice ima neku zadanu vrijednost (najčešće nulu). Matematička forma jest

$$f(x) = 0,$$

pri čemu $f(x)$ može biti linearna ili nelinearna funkcija. Ukoliko želimo da jednadžba ima jednoznačno rješenje, smije imati samo jednu nepoznanicu.

Sistem jednadžbi dobivamo kada je problem opisan s više jednadžbi od kojih svaka ima više nepoznanica. Matematička forma sistema jednadžbi jest (primjer s tri jednadžbe f, g, h i tri nepoznanice x, y, z)

$$\begin{aligned} f(x,y,z) &= 0 \\ g(x,y,z) &= 0 \\ h(x,y,z) &= 0 \end{aligned}$$

Funkcije f, g, h mogu biti linearne ili nelinearne (ili miješano), a ukoliko želimo da sistem jednadžbi ima jednoznačno rješenje, broj nepoznanica mora biti jednak broju jednadžbi.

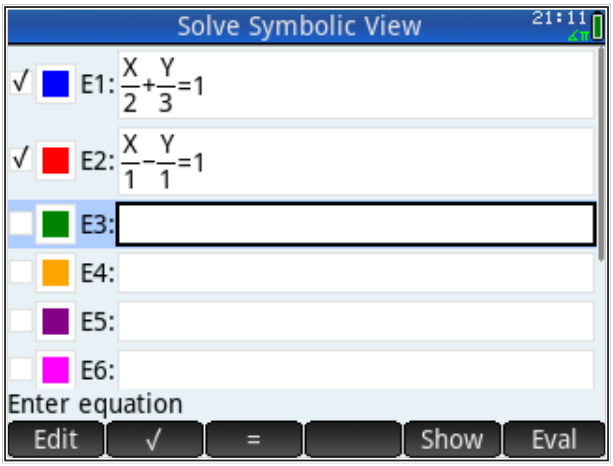
Funkcije f, g, h su pisane implicitno, što znači da su sve nepoznanice s iste strane jednadžbe (i najčešće se ne mogu napisati tako da je jedna nepoznanica sama s jedne strane znaka jednakosti. Dakle, nije, npr., moguće pisati $y = f(x)$ nego samo $g(x,y) = 0$).

Sistem jednadžbi možemo riješiti u aplikaciji 'Solve', ali je grafički prikaz rješenja neuobičajen. Aplikacija 'Advanced Graphing' nameće se (između ostalog) kao proširenje aplikacije 'Solve' na način da omogućuje uobičajeni grafički prikaz rješenja dvije (nelinearne) jednadžbe s dvije nepoznanice.

Aplikacija *rješavanje*

Aplikacija 'Solve' omogućuje numeričko rješavanje do 10 jednadžbi istovremeno te grafički prikaz rješenja jedne od jednadžbi.

Prikažimo način primjene aplikacije kroz jednostavni problem rješavanja dvije linearne jednadžbe s dvije nepoznanice. Otvaramo aplikaciju pritiskom na *Apps* i izborom 'Solve', čime se otvara simbolički prikaz jednadžbi. Upisujemo 2 linearne jednadžbe s 2 nepoznanice

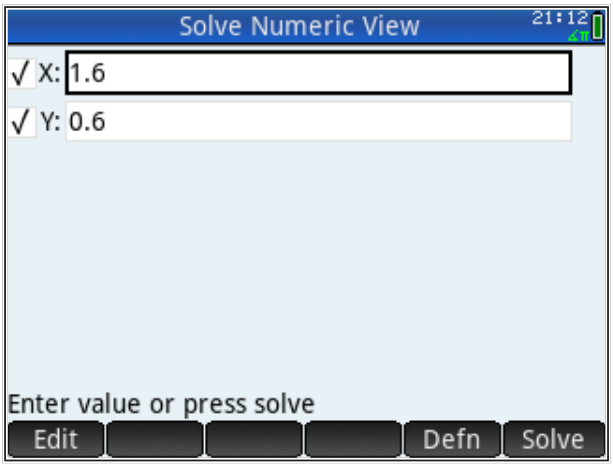


Da bismo naglasili razliku prema aplikaciji 'Function', jednadžbe pravaca zadane su u segmentnom obliku $\frac{x}{m}+\frac{y}{n}=1$, obliku u kojem nije moguće zadati pravac u aplikaciji *funkcije*.

Napomena: u definiciji jednadžbi možemo koristiti i jednadžbe definirane u aplikaciji funkcije ('Function').

Ove dvije jednadžbe rješavat ćemo paralelno u aplikacijama *rješavanje* i *funkcije*, a kako bismo naglasili razlike i prednosti svake od njih.

Rješenje u aplikaciji *rješavanje* dobivamo u brojčanom pogledu ('Numeric view'), pazeći da su u simboličkom prikazu ('Symbolic View') obadvije jednadžbe, E1 i E2, označene kvačicom.

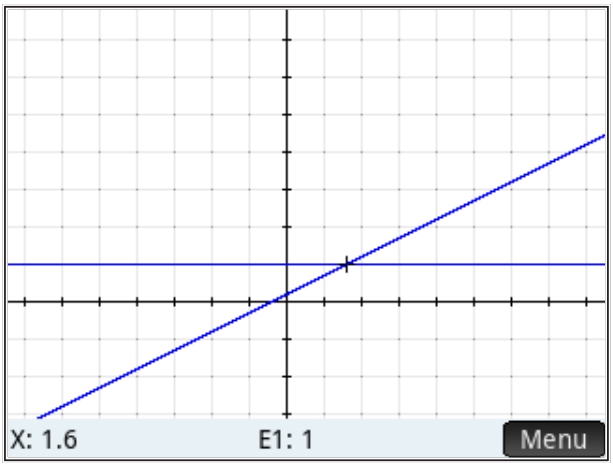


Moguće je da se za X i Y pojave neke vrijednosti preuzete iz drugih aplikacija ili od prijašnje aktivnosti. Npr., nakon naredbe *Plot*, pomicanjem kursora mijenjamo vrijednost varijable X ili Y. Potrebno je kliknuti iz menija ispod slike i tek smo tada sigurni da vidimo točno rješenje. Također, u polja X i Y mogu se zadati neki brojevi koji su početna vrijednost za iterativno računanje rješenja sistema jednadžbi; to je potrebno učiniti kako bi dobili i druga rješenja ako ih sistem ima (npr., kada rješavamo nelinearne jednadžbe).

Napomena: Posebice za rješavanje sistema linearnih jednadžbi, HP Prime kalkulator ima posebnu aplikaciju, 'Linear Solver' (rješavanje linearnih sistema). Prednost aplikacije je nešto jednostavnija upotreba, ali ograničenje je da može riješiti najviše sistem od 3 jednadžbe s 3 nepoznanice. Također, nema grafičkog prikaza rješenja.

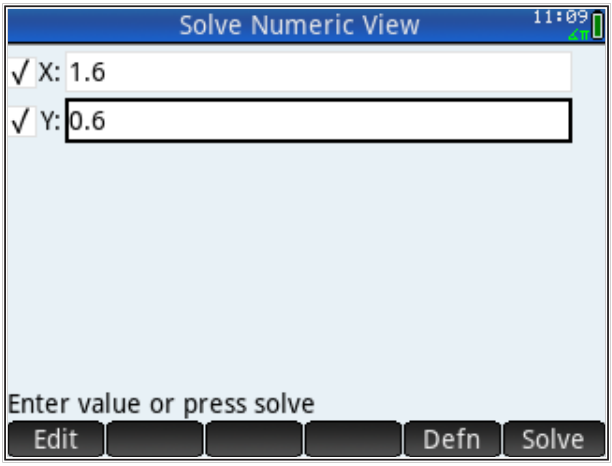
Grafički prikaz u aplikaciji *rješavanje* nije namijenjen vizualizaciji rješenja sistema jednadžbi. Njegova je namjena vizualizacija ponašanja rješenja jedne jednadžbe; na taj način, grafički prikaz, koji se dobiva pritiskom tipke *Plot*, zahtjeva da je u simboličkom prikazu jednadžbi odabrana samo jedna jednadžba (koja će se onda grafički prikazati). Prikaz daje dvije krivulje, posebno za lijevu, a posebno za desnu stranu jednadžbe. Prikaz uzima u obzir numeričke vrijednosti svih varijabli koje se pojavljuju u jednadžbi, a presjecište daje vrijednost odabrane nepoznanice.

Sljedeća slika pokazuje kako se mijenja vrijednost jednadžbe E1 ovisno o promjeni varijable X

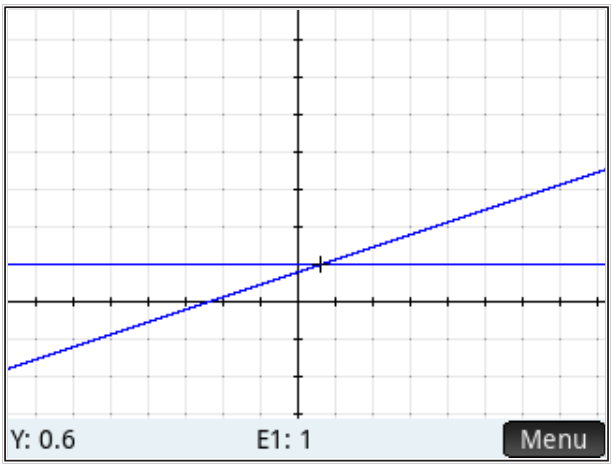


Vrijednost varijable X mijenja se pomicanjem kursora, a vrijednost funkcije E1 se računa. Prikaz na gornjoj slici vrijedi za Y = 0.6, kako je definirano u numeričkom pogledu ('Numeric View'). Tamo možemo zadati neku drugu vrijednost varijable Y i vidjeti kako se jednadžba E1 ponaša kad se mijenja X, a Y ima zadanu vrijednost.

Ukoliko želimo vidjeti kako se mijenja vrijednost funkcije E1 ovisno o varijabli Y, varijablu Y trebamo označiti u numeričkom pogledu.



Sada *Plot* pokazuje ovisnost između funkcije E1 i varijable Y (za zadani X)

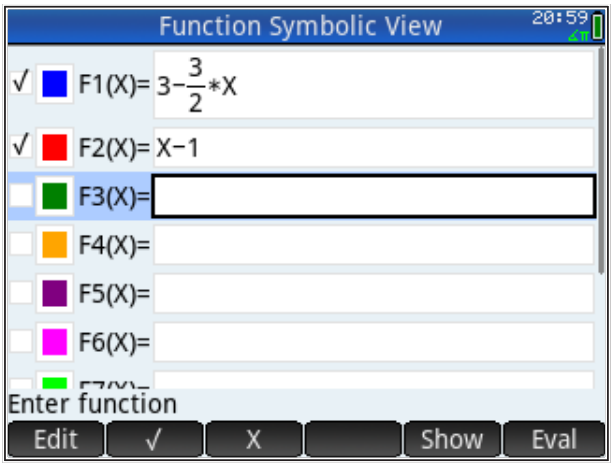


Potrebno je paziti da se u numeričkom pogledu nije promijenila vrijednost varijabli jer kada je označena samo jedna jednadžba, ulaskom u *Num* računa se samo ta. Najsigurniji je način odabir obadviju jednadžbi, ulazak u *Num*, odabir varijable za crtanje, ulazak u *Symb* pa crtanje sa *Plot*.

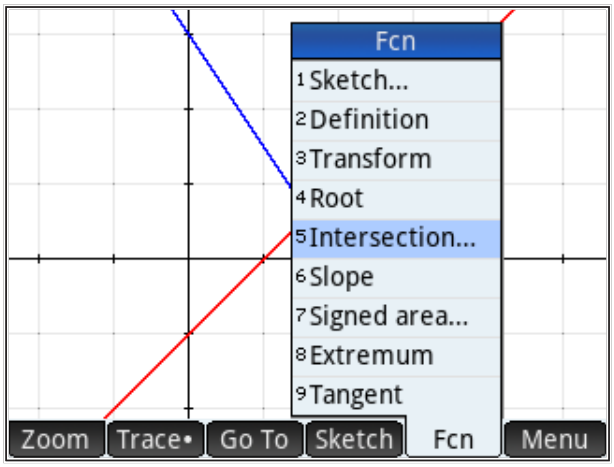
Na isti način možemo pokazati ovisnost između funkcije E2 i varijabli X ili Y (grafički način će sada biti crvene boje, odnosno, one boje koja je dodijeljena odabranoj funkciji).

Prikaz preko aplikacije funkcije

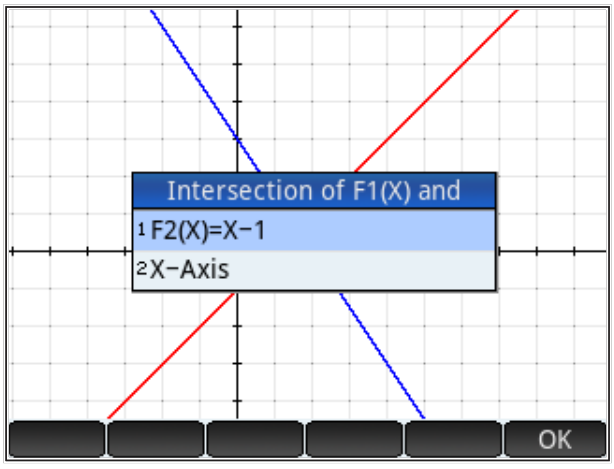
Rješenje prethodne dvije linearne jednadžbe možemo vidjeti kao presjecište dvaju funkcija u aplikaciji *funkcije*. Tipkamo *Apps* i biramo aplikaciju *funkcije* i tipkamo jednadžbe. Za upis u aplikaciju *funkcije*, jednadžbe moramo iz segmentnog oblika pretvoriti u eksplicitni, $y = \frac{1}{n} \cdot (1 - \frac{x}{m})$. U tom obliku, gornje dvije jednadžbe glase



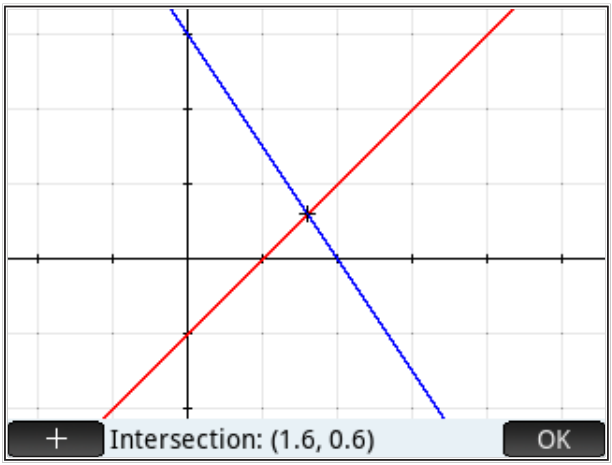
Rješenje ove dvije jednadžbe je zajednička točka s koordinatama $T(x_0, y_0)$, odnosno, geometrijski je to presjecište dva pravca, F1(X) i F2(X). Tipkamo *Plot* i dobivamo grafički prikaz te možemo zumirati da bolje vidimo presjecište pravaca. Da bismo numerički odredili presjecište, s ekrana biramo 'Menu', zatim 'Fcn' pa zatim 'Intersection'.



Nakon toga, biramo nalaženje presjecišta dvaju funkcija (a ne funkcije i X-osi).



Na kraju, vidimo rješenje s ispisom koordinata presjecišta



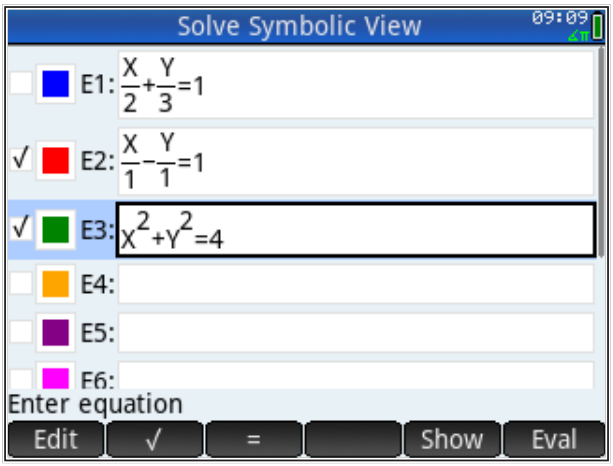
Prema očekivanju, rješenje u obadvije aplikacije je isto!

Nelinearne jednadžbe

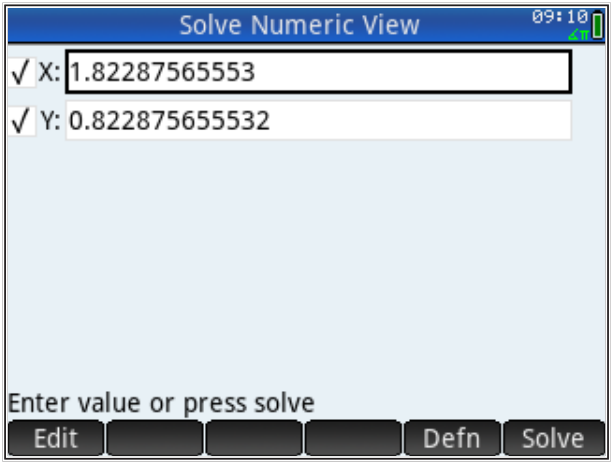
Kod nelinearnih jednadžbi, rješenja, odnosno, zajedničkih točaka (koje se istodobno nalaze i na jednoj i na drugoj krivulji) može biti i više (nelinearne funkcije mogu međusobno imati više presjecišnih točaka). Koju ćemo zajedničku točku dobiti kao rezultat ovisi o tome koja je točka odabrana za početak iterativnog procesa rješavanja.

Za ilustraciju ćemo u prethodne linearne jednadžbe dodati jednadžbu kružnice i naći njeno presjecište s pravcem zadanim kao jednadžba E2 (u aplikaciji *rješavanje*), odnosno, F1 (u aplikaciji *funkcije*).

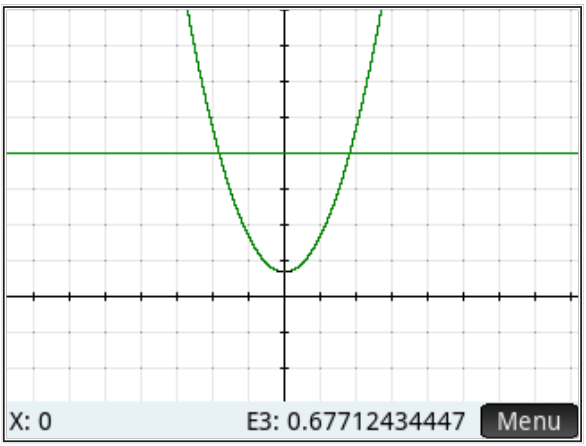
Biramo aplikaciju 'Solve' i zadajemo jednadžbu kružnice



Presjecište jednadžbi E2 i E3 predstavlja rješenje sistema tih dvaju jednadžbi. U numeričkom pregledu biramo 'Solve' i dobivamo jednu presjecišinu točku.



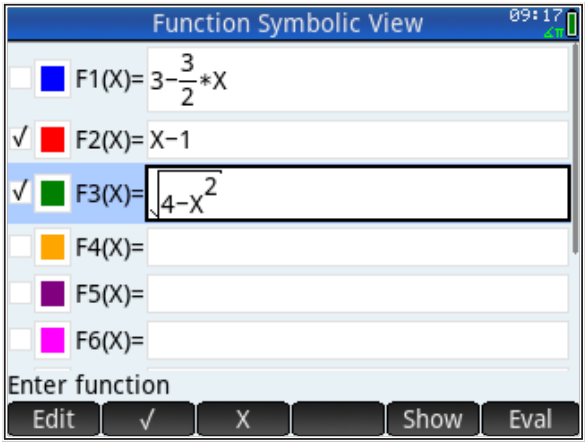
Ukoliko želimo vidjeti grafički prikaz presjecišta, npr., da vidimo postoji li još mogućih rješenja sustava jednadžbi, to nećemo moći učiniti u aplikaciji 'Solve'; ne možemo označiti dvije jednadžbe za grafički prikaz. Ako u simboličkom pregledu označimo samo E3 i otipkamo *Plot*, dobijamo prikaz lijeve i desne strane funkcije E3.



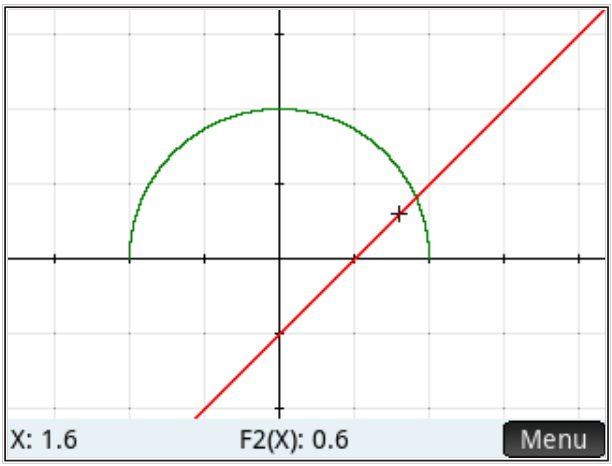
Parabola predstavlja lijevi dio jednadžbe E3, odnosno $y = x^2 + Y_0^2$. Desni dio jednadžbe E3 je pravac, odnosno $y = 4$. Prtom, rješenje sistema jednadžbi je (X_0, Y_0). Njihovo presjecište je X_0, a rješenje jednadžbe E3 ako nepoznanica ima vrijednost upisanu u numeričkom pogledu (u ovom slučaju Y_0, vrijednost koja je ostala u numeričkom pogledu nakon rješavanja sistema jednadžbi). *Napomena: pogledaj kako se grafički prikaz mijenja kako se upisuju različite vrijednosti za Y u polju numeričkog pogleda.*

Istovremenu ovisnost rješenja funkcija E2 i E3 može se vizualizirati u aplikaciji *funkcije*.

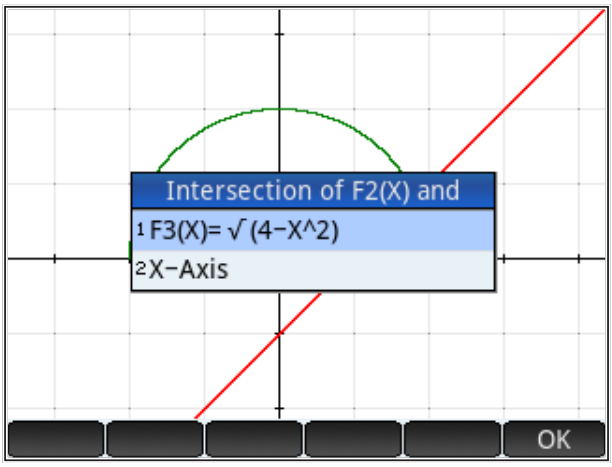
U aplikaciji 'Function' jednadžbu kružnice ne možemo zadati u implicitnom obliku (kao funkciju nepoznanica (x, y)). Stoga zadajemo samo pozitivnu polovicu jednadžbe kružnice kao F3(X).



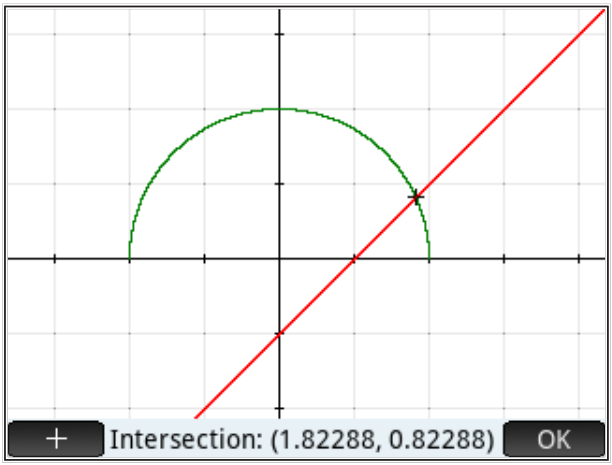
Sjetimo se da je implicitna jednadžba pravca E2(X,Y) iz aplikacije *rješavanje* prepisana kao eksplicitna jednadžba F2(X) u aplikaciji *funkcije*. Označimo jednadžbe F2 i F3 u aplikaciji *funkcije* i grafički ih prikažemo tipkanjem na *Plot*.



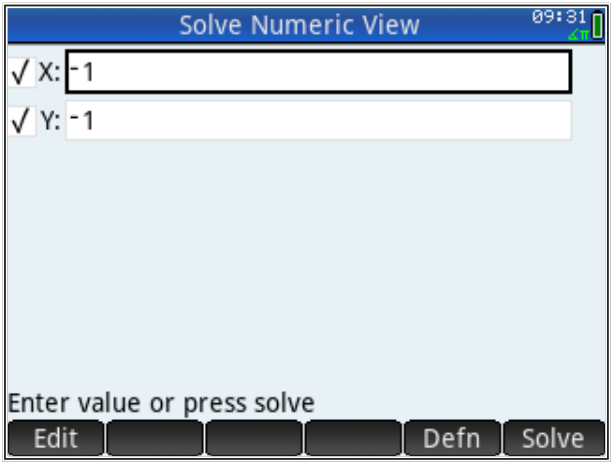
Koordinate presjecišta F2 i F3 nalazimo ulaskom u 'Menu' u grafičkom prikazu, a nakon toga biramo 'Fcn' i 'Intersection', biramo da želimo presjecište F2(X) i F3(X).



Na dnu ekrana dobivamo koordinate (jednog) presjecišta.



Koordinate drugog presjecišta lako dobivamo u aplikaciji *rješavanje*; samo trebamo zadati drugačiju polazišnu točku za postupak iteracije, npr., $(X_0 = -1, Y_0 = -1)$. Tipkamo *Apps* i biramo 'Solve', označimo E2 i E3, tipkamo *Num* i zadajemo nove početne točke za računanje rješenja



Biramo 'Solve' i na ekranu su koordinate drugog presjecišta
$(X_0 = -0.8228756555, Y_0 = -1.822875655)$.

Grafički prikaz drugog presjecišta u aplikaciji *funkcije* zahtjeva pisanje donjeg dijela jednadžbe kružnice.

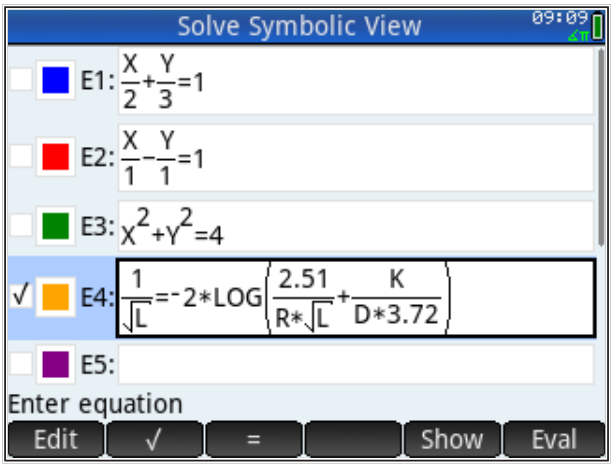
Zadatak: nađi drugo presjecište u aplikaciji 'Function'.

Primjer:
Colebrookova jednadžba

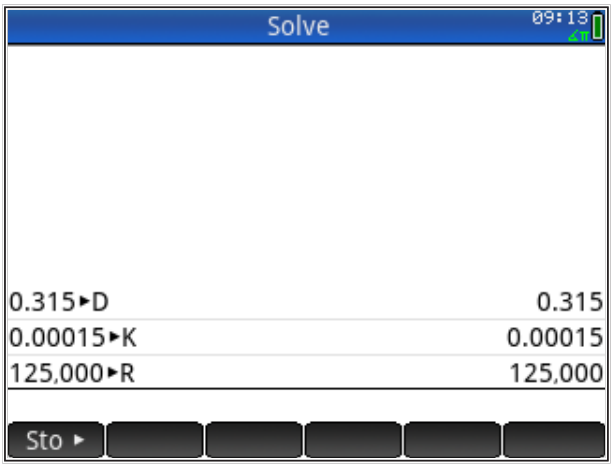
Primjenu aplikacije *rješavanje* prikazat ćemo na praktičnom inženjerskom primjeru: rješavanju jednadžbe tečenja kroz cijev (Colebrookova jednadžba). Colebrookova jednadžba glasi

$$\frac{1}{\sqrt{\lambda}} = -2log(\frac{2.51}{Re \cdot \sqrt{\lambda}} + \frac{\kappa}{3.72D_h})$$

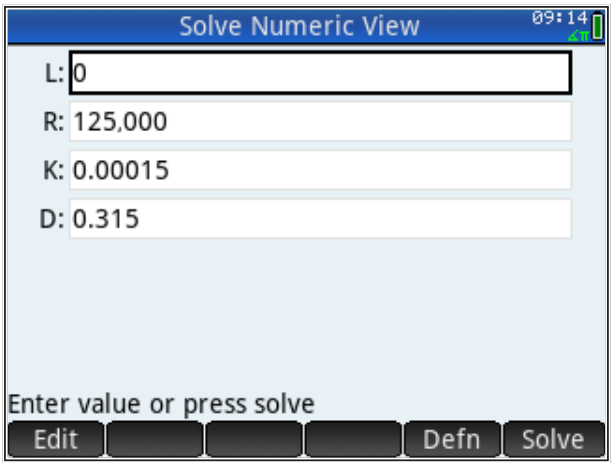
U navedenoj jednadžbi λ je koeficijent trenja, Re je Reynoldsov broj, κ je hrapavost cijevi, D_h je hidraulički promjer. Objašnjenja fizikalnog značaja pojedinih koeficijenata ovdje nećemo prikazivati. Treba izračunati koeficijent trenja λ, a jednadžba je zadana implicitno (λ se nalazi s obje strane znaka jednakosti). Jednadžbu ćemo zapisati u aplikaciju *rješavanje* HP Prime kalkulatora; umjesto oznake λ stavit ćemo L



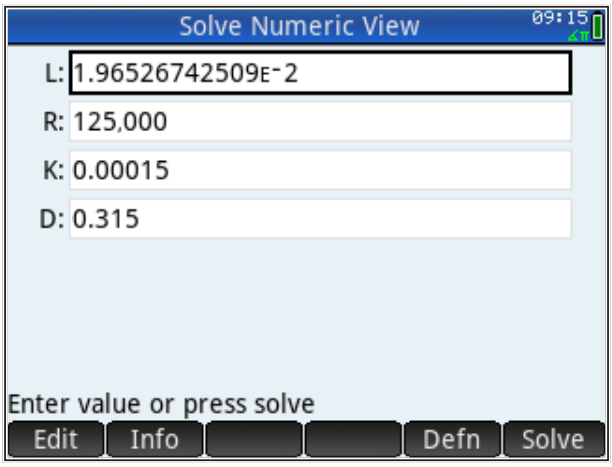
Riješiti ćemo jednadžbu za vrijednosti parametara $D_h = 0.315m$, $\kappa = 0.00015m$ i $Re = 125000$. Zadamo vrijednosti parametara



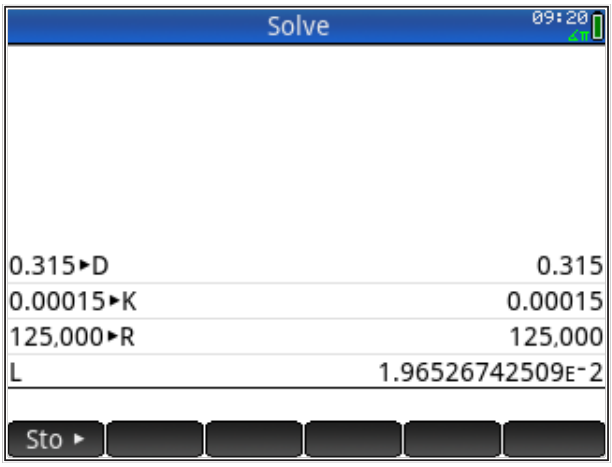
Na slici je prikazan drugačiji način zadavanja vrijednosti varijabli; umjesto ':=' koristimo naredbu 'Sto' = store, koji se dobije pritiskom na 'Sto>' na izborniku u donjem redu ekrana. Sada se prebacimo u numerički način aplikacije *Solve*



Rješenje dobivamo pritiskom na 'Solve' i ono glasi $\lambda = 0.0197$.



Rješenje je spremljeno u varijabli 'L' i možemo ga koristiti u daljnjim izračunima.



Rješavanje sistema linearnih jednadžbi

Posebna aplikacija 'Linear Equation Solver' omogućuje brzo rješavanje linearnih sistema s dvije ili tri jednadžbe, bez grafičkog prikaza. Kako aplikacija 'Linear Equation Solver' ne postoji unutar besplatne verzije HP Prime kalkulatora, pokazat ćemo kako za istu svrhu možemo koristiti aplikaciju 'rješavanje“ (tj. 'Solve').

Zamislimo da trebamo riješiti zadatak raspodjele 100% iznosa na 3 autora tako da 1. dobije 10% više nego drugi, a drugi dobije 10% više nego treći. Matematički zadatak formuliramo kao sistem jednadžbi, gdje je 1. autor označen s 'X', drugi s 'Y', a treći sa 'Z'. Ukupni iznosi koji dobiju sva 3 autora jest 100%, dakle:

$$X + Y + Z = 100$$

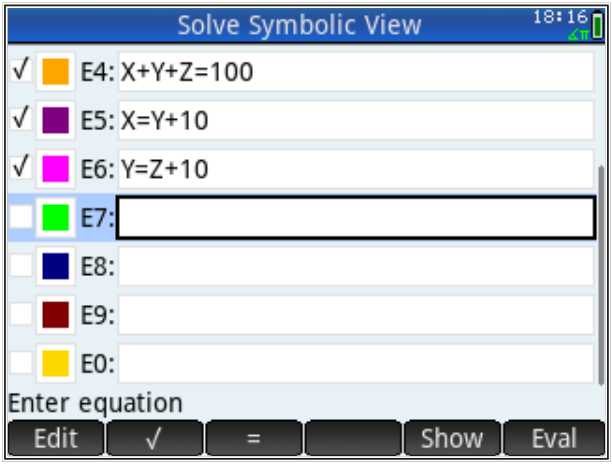
Prvi autor dobije 10% više nego drugi,

$$X = Y + 10$$

drugi autor dobije 10% više nego treći,

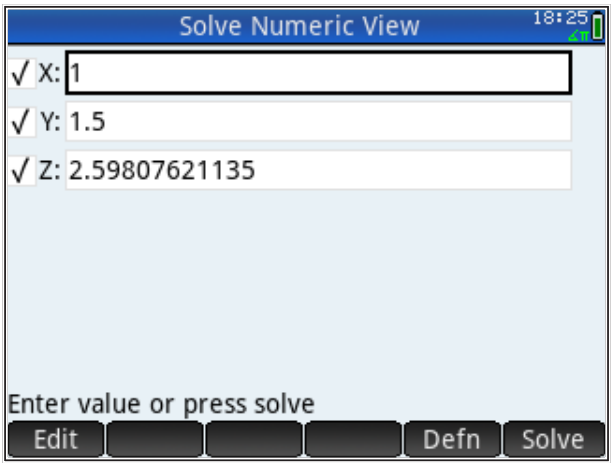
$$Y = Z + 10$$

i dobili smo linearni sistem od 3 jednadžbe s 3 nepoznanice koji unosimo u aplikaciju 'Solve' kao jednadžbe E4, E5 i E6.

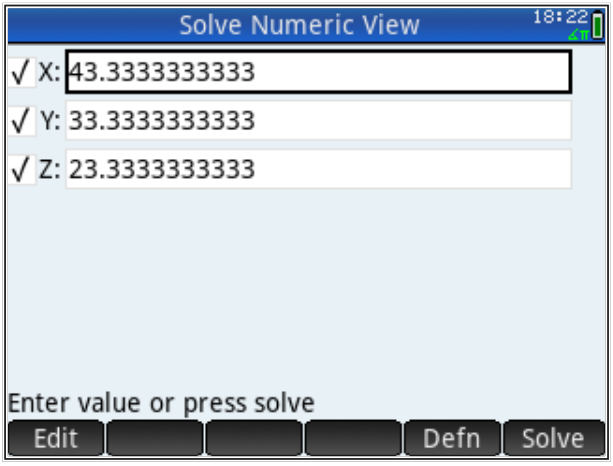


Vidimo da jednadžbe možemo upisati u bilo koju varijablu; bitno je samo da su sve one jednadžbe koje sačinjavaju problem označene kvačicom, a druge nisu.

Sistem rješavamo prelaskom u numerički način tipkom *Num*.



Za 'X,Y,Z' vidimo neki rezultat od prije. To nije rješenje! Rješenje dobijemo pritiskom na 'Solve' u meniju na dnu ekrana i dobivamo rješenje problema (sistema jednadžbi).



Za sisteme s više od 3 nepoznanice, unos jednadžbi u aplikacije 'Linear Equation Solver' i 'Solve' je nepraktičan; brže je i lakše koristiti metode linearne algebre i matrični račun.

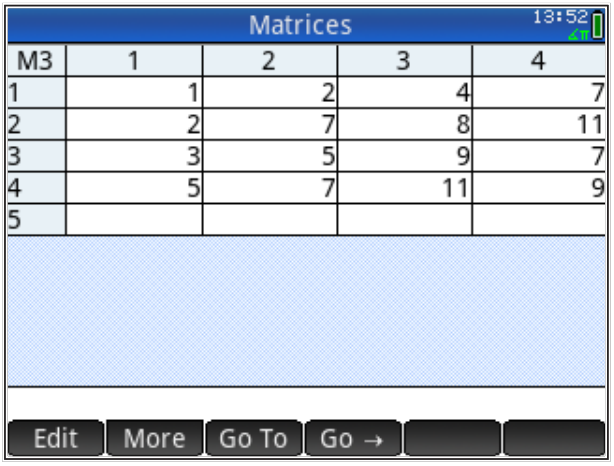
Matrični račun vrlo se lako i intuitivno koristi na HP Prime za rješavanje sistema linearnih jednadžbi. Prije nastavka preporučuje se ponoviti znanje linearne algebre (što je transponirana matrica, Gaussova eliminacija i dr.).

Prije smo vidjeli kako se invertira matrica pa je jasno kako se sistem može riješiti invertiranjem. Sada ćemo prikazati drugi pristup koji ima i edukativni značaj; uvest ćemo funkciju 'RREF' (Reduced Row-Eshalon Form). Navedena funkcija imitira Gaussov postupak eliminacije elemenata matrice i rješavanja sistema jednadžbi.

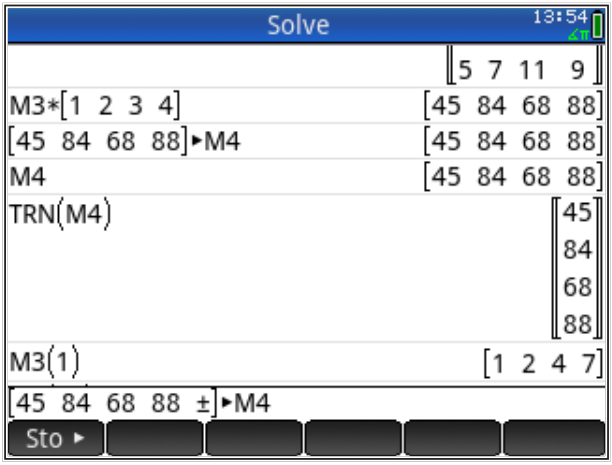
Neka je zadan sistem linearnih jednadžbi

$$\begin{bmatrix} 1 & 2 & 4 & 7 \\ 2 & 7 & 8 & 11 \\ 3 & 5 & 9 & 7 \\ 5 & 7 & 11 & 9 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 45 \\ 84 \\ 68 \\ 88 \end{Bmatrix}$$

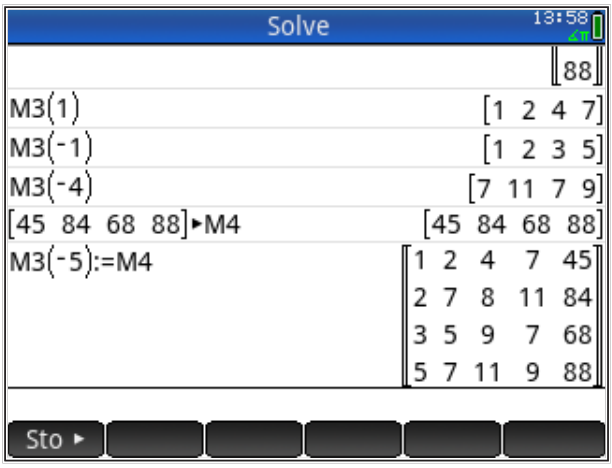
Zadani sistem treba riješiti tako da dobijemo vektor **x**. Koeficijente jednadžbe spremit ćemo u matricu M3 putem editora za matrice, tipkanjem *Shift* *Matrix*.



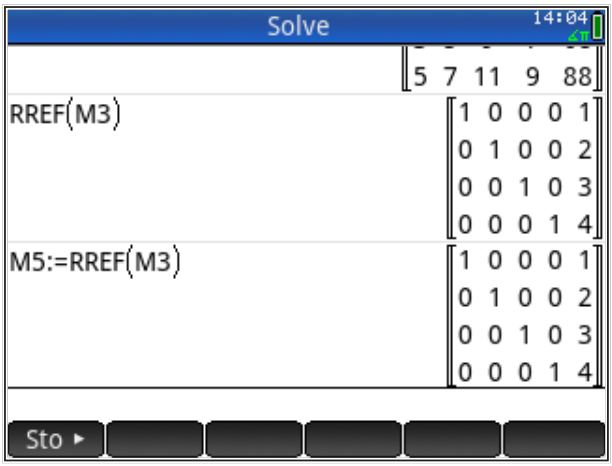
Vektor desnih strana spremit ćemo u matricu (vektor) M4 u 'Home' pogledu



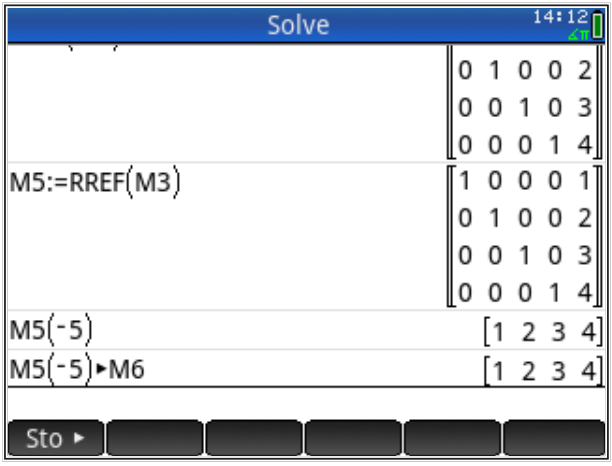
Sada ćemo vektor M4 pridodati matrici M3 kao 5. stupac (primijetimo da pozitivni indeks matrice predstavlja redak, a negativni indeks predstavlja stupac).



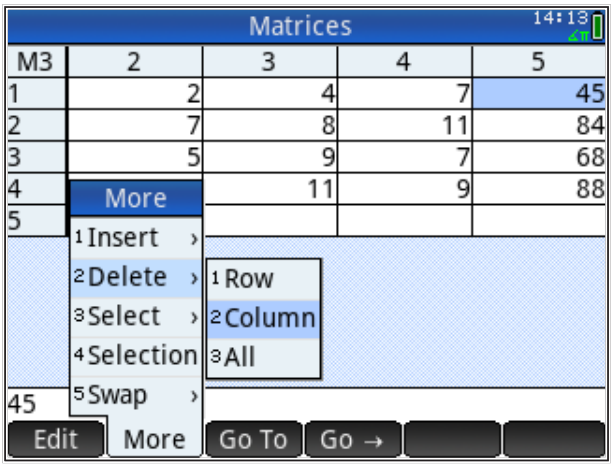
Matrica M3 sada više nema dimenziju [4x4] nego [4x5], kao kada ručno provodimo Gaussov postupak eliminacije. Gaussov postupak eliminacije provodimo putem naredbe 'RREF'. Rješenje će se nalaziti u zadnjem, 5. stupcu i možemo ga prekopirati u, npr., M5.



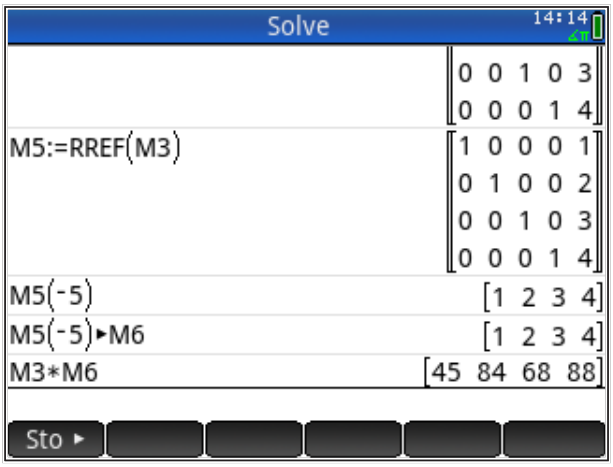
Iz matrice M5 može se izvaditi samo 5. stupac i prekopirati u neki vektor, npr., M6.



Za kontrolu možemo pomnožiti matricu M3 i vektor M6, a rezultat treba biti jednak desnoj strani polazne jednadžbe, tj., vektoru M4. Prije toga, matricu M3 treba vratiti u prvobitno stanje: matricu koeficijenata dimenzije [4x4]. Ulazimo u uređivač matrica i brišemo zadnji stupac



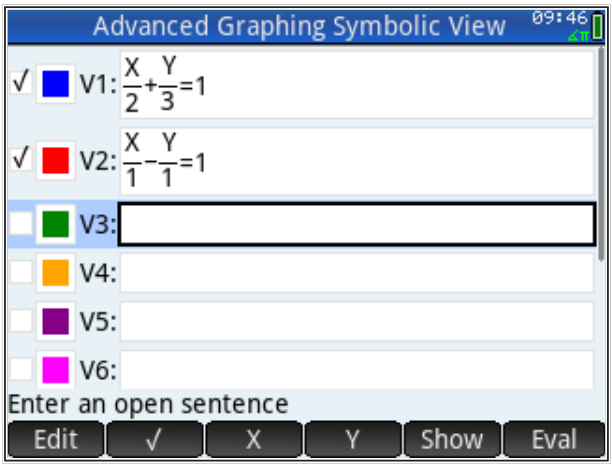
Sada možemo pomnožiti M3 i M6 kontrolirati rezultat



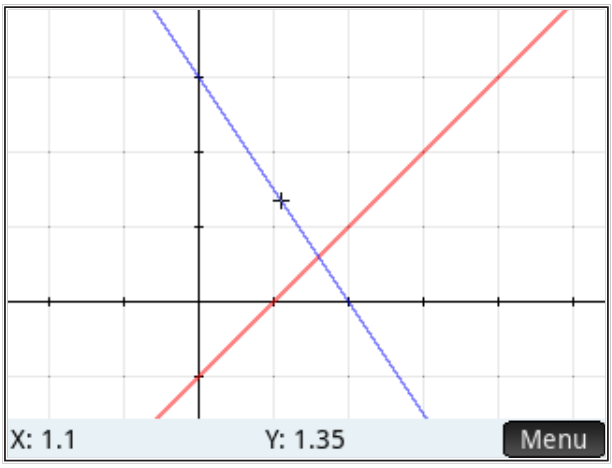
Aplikacija *napredno prikazivanje*

Aplikacija 'Advanced Graphing' omogućuje vizualizaciju do 10 funkcija s dvije nepoznanice, pri čemu su funkcije zapisane implicitno. Također, aplikacija omogućuje i rješavanje **nejednadžbi** jer funkcije mogu biti i u obliku nejednadžbe (osim u obliku jednadžbe). Tu ćemo moći prikazati, osim eksplicitno zadanih (kao u aplikaciji *funkcije*), i implicitno zadane funkcije (s maksimalno dvije nepoznanice).

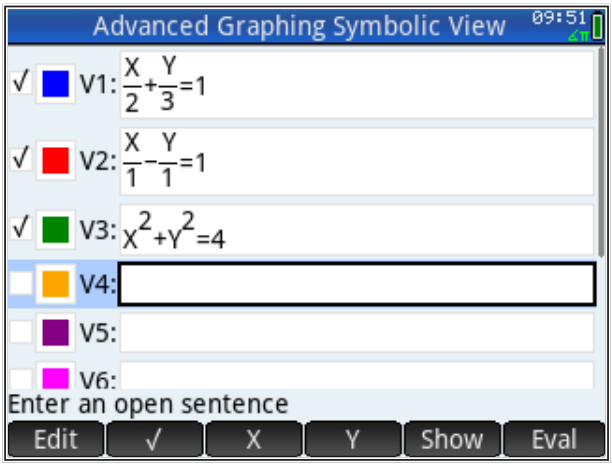
Pogledajmo kako u aplikaciji *napredno prikazivanje* izgledaju prethodne dvije segmentne jednadžbe pravca. Tipkamo [*Apps*] i biramo 'Advanced Graphing'; otvara se simbolički pregled u koji upisujemo jednadžbe



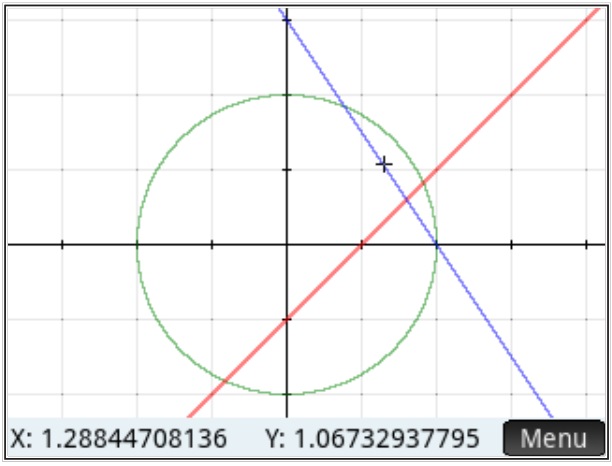
i njihov grafički prikaz



Sada dodajemo i jednadžbu kružnice kao V3



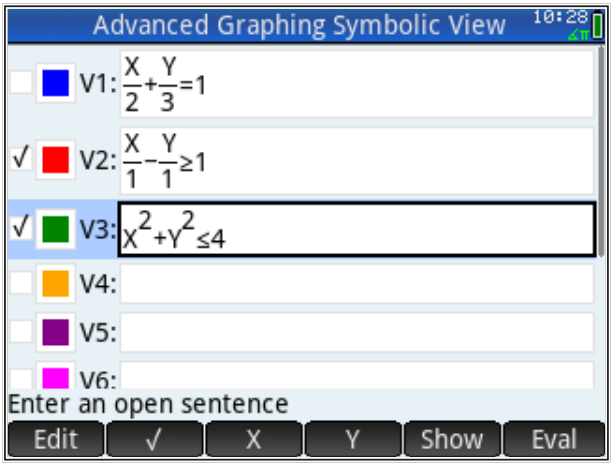
i grafički prikaz sve tri funkcije



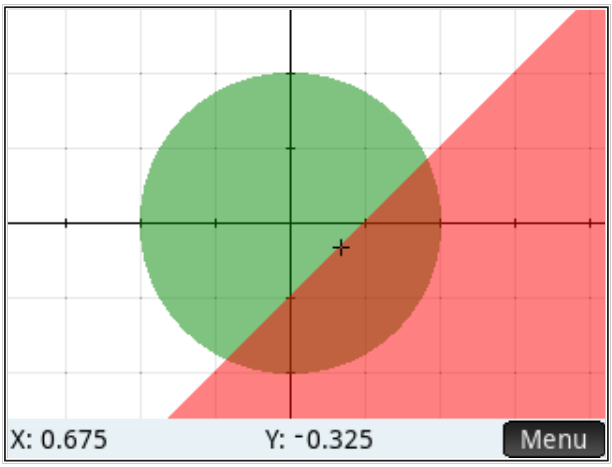
Na slici možemo vidjeti izgled funkcija i broj njihovih zajedničkih točaka, ali u aplikaciji *napredno prikazivanje* ne možemo računati presjecišta, no možemo ih približno odrediti, što nam može poslužiti kao početna pretpostavka rješenja u nekom od postupaka numeričkog izračuna rješenja sistema jednadžbi.

Jedinstvena mogućnost aplikacije 'Advanced Graphing' je mogućnost prikaza nejednadžbi, a time i grafički prikaz rješenja jednostavnih optimizacijskih problema.

Pretpostavimo da nas zanimaju sve točke koje su unutar zadane kružnice i nalaze se ispod zadanog pravca. U našem primjeru, kružnica neka je zadana s V3, a pravac neka je V2. Da bismo vidjeli željene točke, u jednadžbi V2 treba znak jednakosti pretvoriti u ≥, a u V3 u ≤:



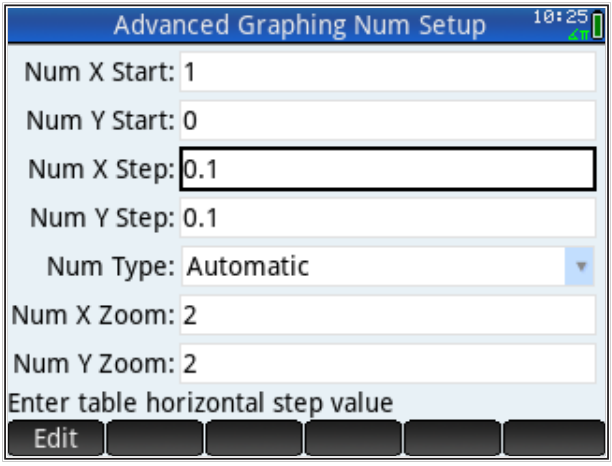
Tipkamo *Plot* i dobivamo grafički prikaz



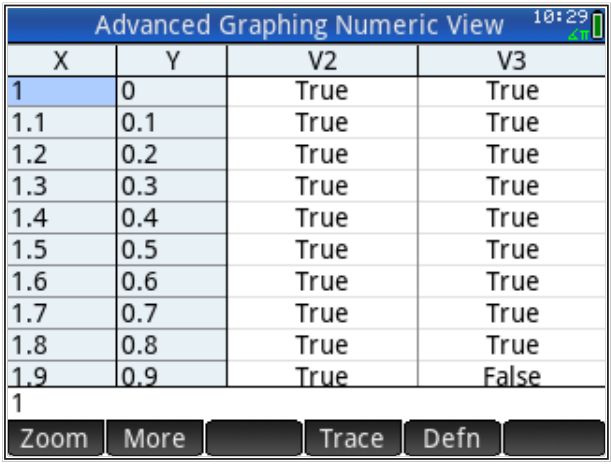
Grafički prikaz točaka koje zadovoljavaju tražene uvjete označen je sjenčanjem, a oba uvjeta su zadovoljena u području koje je dvostruko osjenčano.

Napomena: Rješavanje problema opisanog nejednadžbama je zapravo rješavanje optimizacijskog problema. Npr., neka je problem optimizacije definiran kao: nađi sve točke koje su bliže zadanoj lokaciji (definirano središtem kružnice) od 4 km (radijus), a nalaze se istočno od (neke) ceste (koja je u našem primjeru opisana pravcem). Rješenje je sjenčana zona, kao u našem primjeru.

Numerički prikaz u ovoj aplikaciji ne nudi rješavanje presjecišta, nego pokazuje koje točke zadovoljavaju postavljene uvjete, a koje ne. Npr., ako želimo vidjeti koje točke unutar kvadrata omeđenog s $(X = 1..2, Y = 0..1)$ zadovoljavaju postavljeni uvjet, prvo u numeričkom pogledu definiramo željeni kvadrat, tipkanjem *Shift* i *Num → Setup*



Zadali smo 0.1 za razmak točaka i u numeričkom pogledu dobivamo sljedeće:



Vidimo da točke pri gornjem desnom vrhu zamišljenog kvadrata više ne zadovoljavaju traženi uvjet (vrijednost je (*true, false*) točka zadovoljava uvjet u V2 ali ne i u V3).

Napomena: Za naš izmišljeni problem optimizacije na taj način bismo odredili je li neki objekt sa zadanim koordinatama zadovoljava uvjet pretraživanja (unutar 4 km od zadane lokacije i istočno od zadane prometnice).

Zadatak: prikaži sve točke unutar kružnice V3 koje se nalaze iznad pravca V2 i iznad pravca V1.

HP Prime Bilješke

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide* (iz menija 'Help'):

- *Getting started: Copy and Paste*
- *Catalogs and Editors: Note Catalog and Editor*

Pretpostavljeno stanje varijabli u memoriji kalkulatora jest:

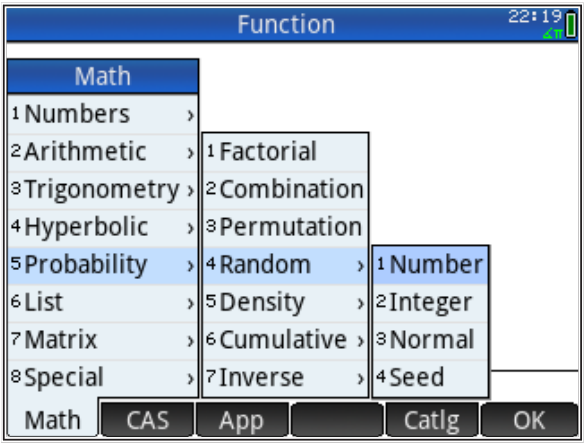
```
L1={1,"A",2.2,"abc"}
L2={1,A,2.2,"abc"}
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite s 'Edit: Paste' naredbama iz menija kalkulatora, pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

Bilješke (Notes) su mjesto u HP Prime kalkulatoru gdje možemo spremati tekst, ali i raditi manipulacije s tekstom. Manipulacije koje želimo pokazati slučajno će kombinirati unaprijed zadani tekst; za šalu možemo reći da ćemo pokušati načiniti da HP Prime simulira „modernog pjesnika“ (bez uvrede pjesnicima). Prvo ćemo pokazati kako generiramo slučajne brojeve, a zatim kako se povezuju liste i bilješke.

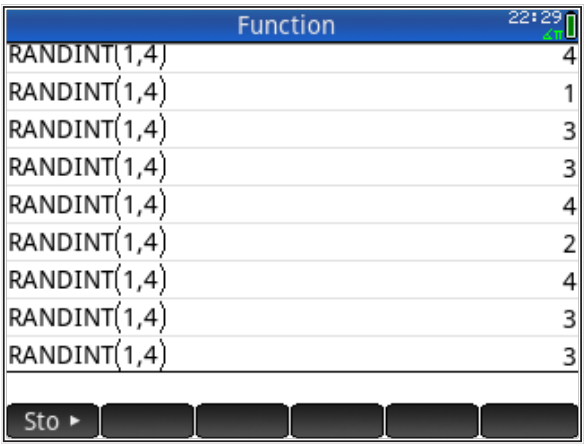
Slučajna varijabla

Pregled funkcija slučajne varijable, koje su nam na raspolaganju na HP Prime, dobit ćemo tipkanjem na 'kovčežić' i zatim izborom 'Math', zatim 'Probability' pa onda 'Random', kao na slici

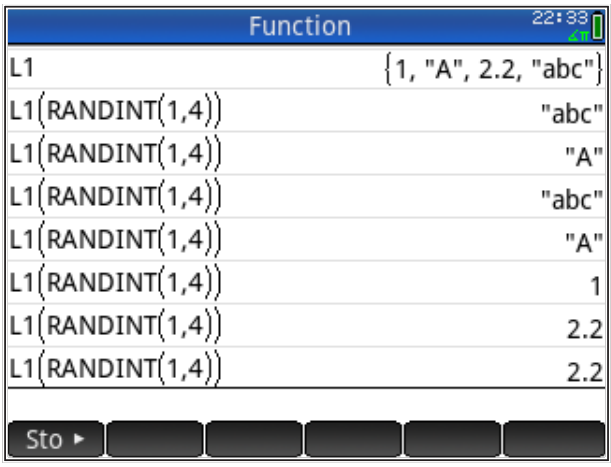


Odabiremo 'Integer' i pojavljuje se funkcija RANDINT(). Izabrali smo funkciju koja generira cijeli slučajni broj jer nam za ovaj primjer trebaju samo cijeli slučajno generirani brojevi.

Iz korisničkog priručnika (*User Guide*) na str. 392 čitamo kako se koristi funkcija 'RANDINT()'; upišu li se dva cijela broja ('integer') unutar zagrada, funkcija generira slučajni broj čija je veličina negdje između ta dva upisana broja. Višestruko pozivanje funkcije daje različite i nepredvidive (slučajne) rezultate, kao na slici



Tako generirani slučajni broj možemo koristiti gdje god možemo upisati cijeli broj. U našem primjeru, slučajno ćemo pozivati element iz prije zadane liste L1.



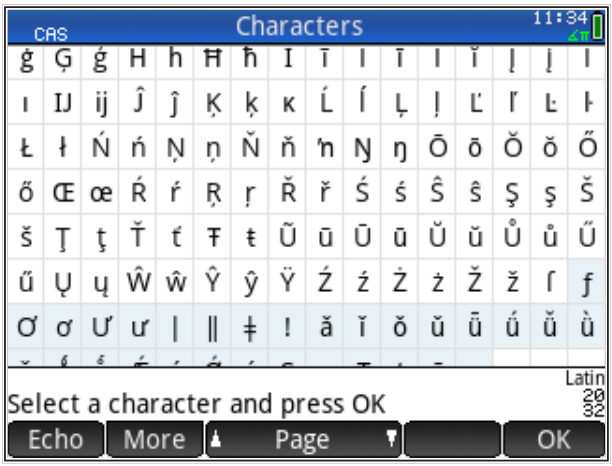
Na ekranu smo pozvali listu L1 da vidimo njezin sadržaj iz kojeg slučajno biramo jedan element (drugim korisnicima se mogu generirati drugi slučajni brojevi i tako se elementi iz liste L1 prikazati drugim redoslijedom).

Sada je jasno zašto smo birali funkciju koja generira slučajni cijeli broj; indeksi koji biraju element iz liste smiju biti samo cijeli brojevi.

Za potrebe primjera definirat ćemo dvije liste samo s tekstualnim elementima. Za zadavanje liste otvorimo katalog svih varijabli tipa list, pritisnemo 'Shift' *List* i onda 'Edit' iz menija ispod ekrana.

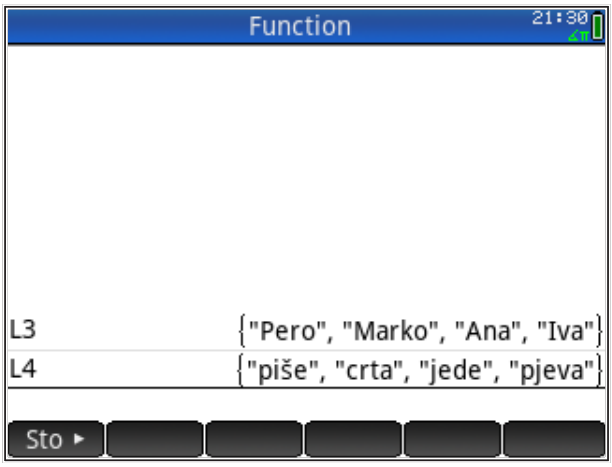
U listu L3 upišemo nekoliko imena, u listu L4 nekoliko aktivnosti, kao na slici

Napomena: Slova s kvačicama i druge posebne znakove dobijemo iz izbornika svih znakova koji se pojavljuje pritiskom na tipke *Chars*; na trećoj stranici (odaberi 'Page V') nalazi se slovo 'š'.

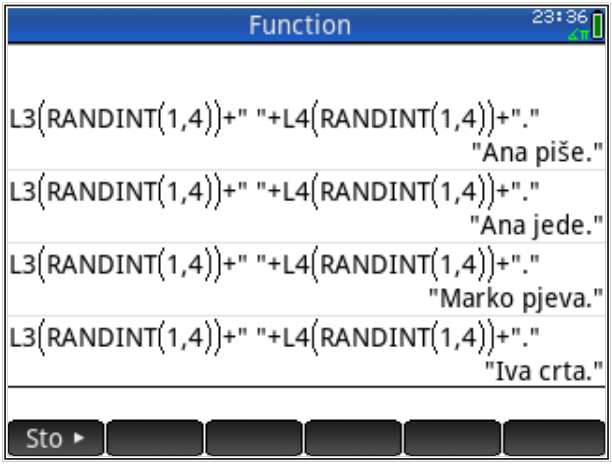


Ne zaboravimo svaki put prije upisa imena i aktivnosti tipkati navodnike *ALPHA* „“ i odabrati 'OK' iz menija ispod ekrana. Sadržaj lista L1 i L2 nije bitan, ostao je iz prethodnih primjera.

Vraćamo se na 'Home' ekran i provjeravamo sadržaj lista L3 i L4 tako da ih ispišemo



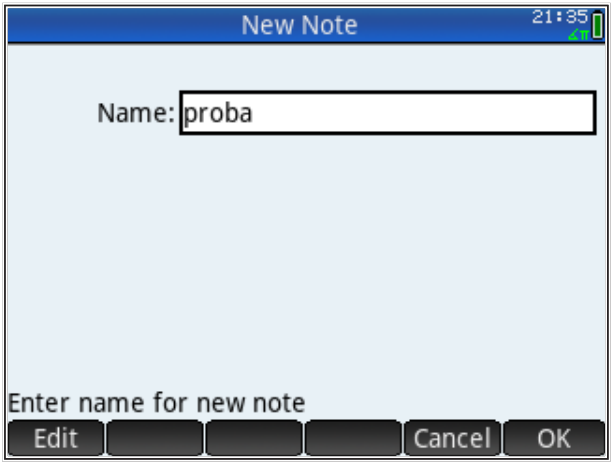
Pogledajmo što dobivamo kada slučajno kombiniramo jedan element iz prve i jedan iz druge liste. Između elemenata liste dodali smo razmak (tipka ␣) i na kraju točku.



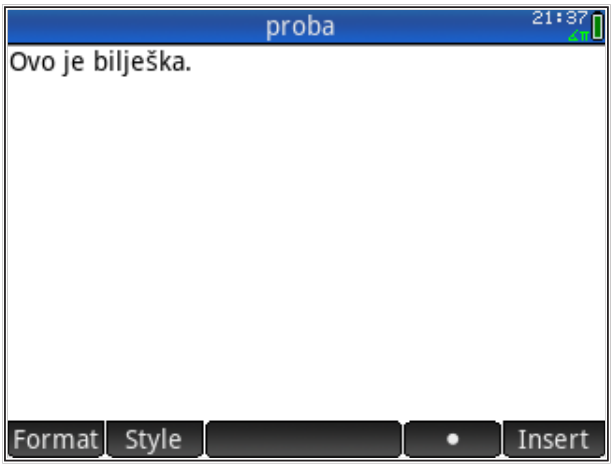
Dobili smo smislene izjave, svaki puta drugačiju (drugim korisnicima mogu se generirati drugi slučajni brojevi i tako se elementi iz listi L3 i L4 prikazati drugim redoslijedom, dajući nešto drugačije rečenice, povezujući druga imena i aktivnosti).

Bilješke su liste

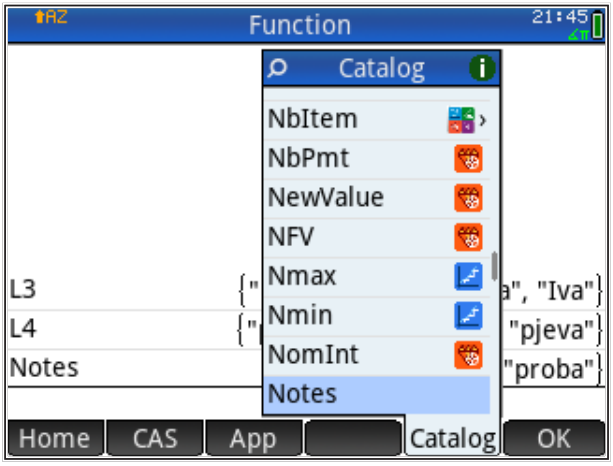
Naše izjave želimo spremiti da ih možemo upotrijebiti kasnije. Bilo koji tekst možemo spremiti u bilješku ('Notes'); za tu svrhu trebamo definirati novu bilješku. Tipkamo 'Shift' *Notes* i zatim biramo 'New', bilješci dajemo ime ('proba' u ovom slučaju) i biramo 'OK'.



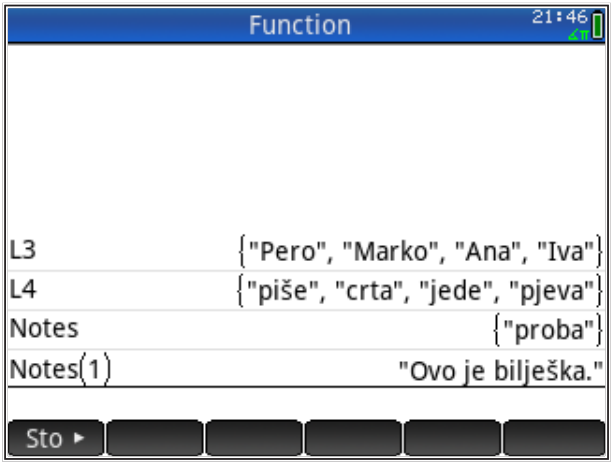
U bilješku možemo upisati neki sadržaj, npr., kao na slici (ne zaboravimo tipkati *ALPHA*, ne trebaju navodnici, bilješka sve prihvaća kao tekst, čak i brojeve ako ih tipkamo).



Tipkamo 'Esc' i vidimo katalog bilješki u kojem je (samo) bilješka 'proba'. Vraćamo se na ekran 'Home', gdje ćemo ispisati nazive svih bilješki koje imamo. Predefinirana varijabla za bilješke je 'Notes' i možemo je samo upisati, ali možemo je i izabrati iz popisa varijabli, kao na slici



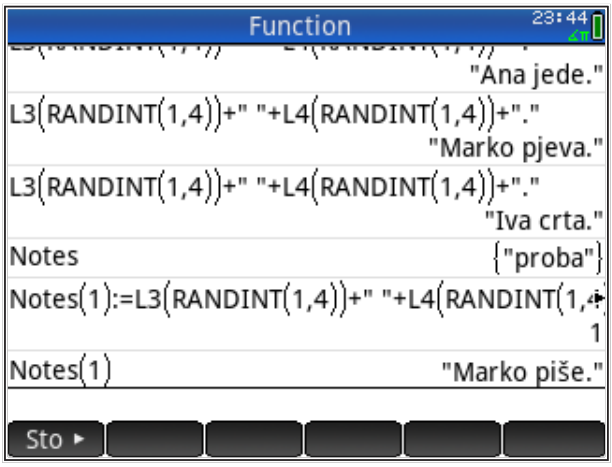
Nakon upisa varijable 'Notes', tipkamo 'Enter' i vidimo da je 'Notes' varijabla tipa liste; u našem slučaju sa samo jednim elementom, 'proba'. Želimo li vidjeti sadržaj prve bilješke (u našem slučaju 'proba'), tipkamo indeks (1) uz listu Notes i vidimo sadržaj koji smo upisali u bilješku 'proba', kao na slici



Na ovaj smo način ustanovili na koji su način bilješke spremljene u sistemu varijabli HP Prime kalkulatora: 'Notes' je ime predefinirane liste; pozivanje elemenata te liste ('Notes(broj)') prikazuje njen sadržaj.

Sada ćemo slučajno generirani tekst spremiti u bilješku. Slučajno izaberimo elemente liste i kombiniramo ih i zatim spremimo. Prva naredba je

```
Notes(1):=L3(RANDINT(1,4))+" "+L4(RANDINT(1,4))+"."
```



Vidimo da je ovaj tekst prebrisao tekst koji je prethodno bio spremljen u bilješku 'proba'.

Dodat ćemo još teksta u bilješku, ali ćemo prije toga pojednostavniti unos podataka. Naime, iako za upis slučajno generiranog teksta iz listi L3 i L4 možemo (i trebamo) koristiti naredbu 'Copy' iz menija ispod ekrana, upotrebom funkcija dodatno možemo pojednostaviti unos podataka.

Funkciju 'RANDINT(1,4)' možemo definirati kao funkciju i pozivati preko naziva funkcije. No, pritom treba znati kako se HP Prime ponaša. Ne možemo napisati

```
FR:=RANDINT(1,4)
```

Time smo definirali varijablu 'FR', čije će pozivanje uvijek davati isti broj, onaj koji je proizvela funkcija 'RANDINT(1,4)' u trenutku definiranje varijable 'FR'. Ako ponovo zadamo varijablu 'FR:=RANDINT(1,4)', imat ćemo neku drugu vrijednost umemoriranu u varijablu 'FR'. Drugim riječima, pozivanje varijable

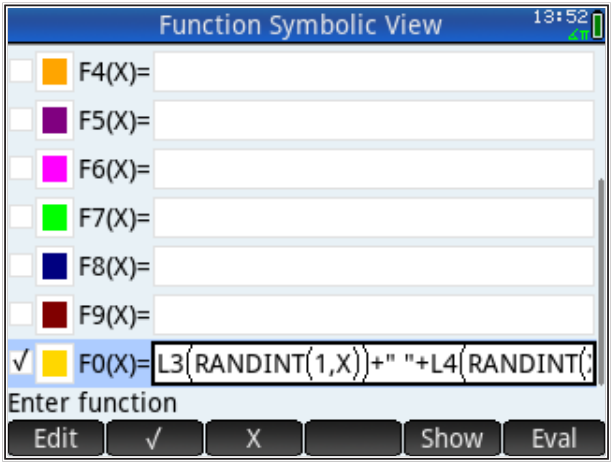
'FR' ne poziva svaki put funkciju 'RANDINT(1,4)'. Funkciju 'RANDINT(1,4)' trebamo definirati kao funkciju

```
FR(X):=RANDINT(1,X)
```

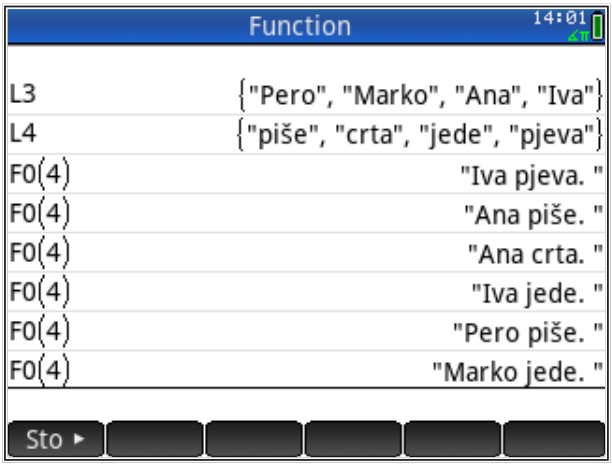
i tek tada će pozivanje (sada funkcije FR(X), a ne varijable 'FR') pozivati 'RANDINT(1,X)'. No, funkciju ne možemo tako zadati preko ekrana, nego treba koristiti aplikaciju 'Funkcije' (*Function*). Također, funkciju se može programirati, ali to ćemo ostaviti za kasnije. Još veće pojednostavljenje upisa dobit ćemo tako da cijelu naredbu

```
L3(RANDINT(1,X))+" "+L4(RANDINT(1,X))+". "
```

upišemo kao funkciju (uočite razmak iza točke, tako da slijedeća izjava počne iza razmaka). *Napomena:* Da bi HP Prime svaki put pozivao 'RANDINT()', funkcija treba imati parametar pa smo umjesto 'RANDINT(1,4)' napisali 'RANDINT(1,X)', pri čemu kod pozivanja funkcije parametar treba biti '4'.



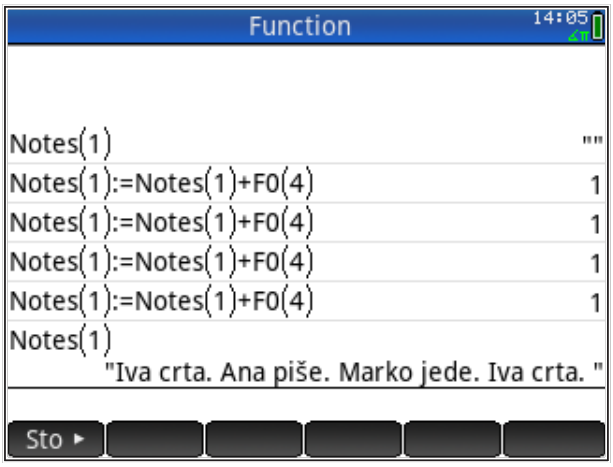
Vidimo da sada jednostavno možemo generirati slučajne izjave



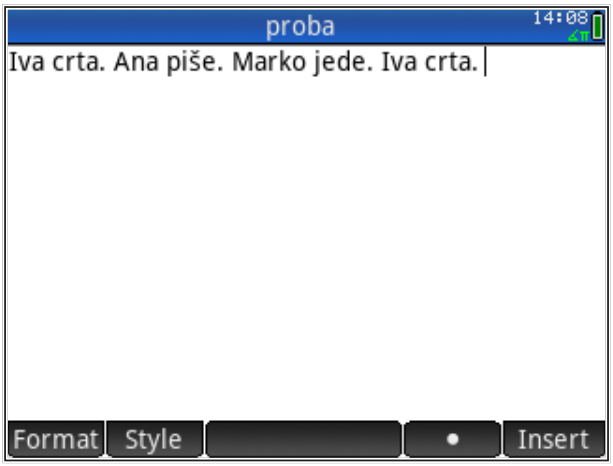
Sada možemo u bilješku jednostavno dodati novi tekst uz već postojeći pa tipkamo

```
Notes(1):=Notes(1)+F0(4)
```

Nakon što gornju naredbu ponovimo nekoliko puta, dobivamo sljedeće:



Za provjeru, sadržaj bilješke možemo vidjeti i u pregledniku bilješki. Tipkamo 'Shift Notes' i biramo 'Edit'.



Zadatak: Dodaj razmake između izjava.

Zadatak: Načini da je svaka izjava u svom retku.

Rješenje ovog zadnjeg zadatka zahtjeva cijelo novo potpoglavlje u kojem ćemo objasniti kako se dodaju znakovi koji se ne mogu napisati, npr., znak za novi redak. Prikazat ćemo i kratki povijesni kontekst kodiranja znakova na računalu.

Kodiranje znakova na računalu

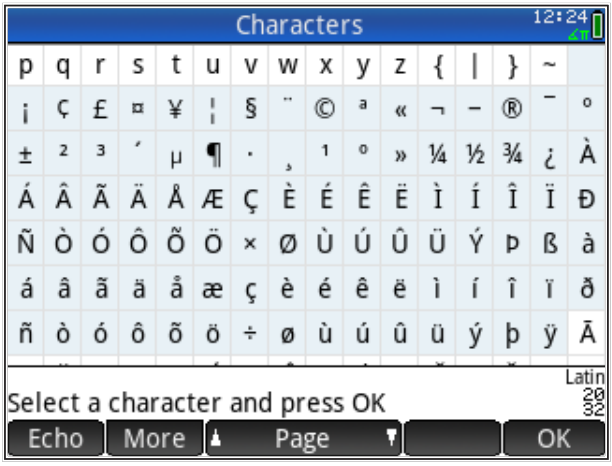
Računalo znakove u svojoj unutrašnjoj i vanjskoj memoriji pamti kao brojeve, kodove, gdje svaki broj predstavlja jedan znak, a prema utvrđenim pravilima koja su pretočena u standarde. Jedan od najstarijih načina kodiranja znakova, koji se zadržao i do danas, jest ASCII[7]. U osnovnom ASCII kodu definirano je 128 brojeva koji predstavljaju razne znakove i naredbe; brojevi od 0 do 31 predstavljaju kontrolne kodove, nije im pridodan niti jedan znak i ne mogu se prikazati na ekranu; brojevi od 32 do 127 predstavljaju razne znakove, npr., broj predstavlja veliko slovo 'A', a broj 97 malo slovo 'a', itd. Osnovna tablica kasnije je proširena s još 128 znakova. Kompletna tablica se može lako naći na Internetu[8]. Može se vidjeti da niti u proširenoj ASCII tablici nema dovoljno znakova za prikaz slova iz svih europskih jezika, pa tako niti iz hrvatskog. To je dovelo do razvoja novih tablica znakova koje su kompatibilne sa starim ASCII

7 American Standard Code for Information Interchange

8 https://en.wikipedia.org/wiki/ASCII

standardom, ali pokrivaju mnogo veći broj jezika i pisama, kao npr. Unicode standard[9] i njegove inačice (npr., UTF-8).

Na HP Prime kalkulatoru tablice raspoloživih znakova možemo vidjeti preko komande *Shift* *Chars*. Pomicanjem 'gore - dolje' ili izborom 'Page' možemo vidjeti sve raspoložive tablice znakova.



Željeni znak biramo kursorom i tipkom *Enter*, nakon čega se tablica zatvori. Izborom naredbe 'Echo' umjesto tipke *Enter* tablica ostaje na ekranu pa dodatne znakove možemo brže izabrati.

Ukoliko koristimo aplikaciju umjesto pravog kalkulatora, jednostavnije je tipkati direktno preko tipkovnice računala; svi znakovi s računala dostupni su i na HP Prime aplikaciji.

Za objašnjenje kontrolnih kodova koji nam trebaju za rješenje navedenog zadatka moramo ići u prošlost, u 60-e godine prošloga stoljeća kada je nastajao ASCII kod, sistem kodiranja znakova za teleprinter (i računalo, između ostalih uređaja). Takvi stari uređaji za pisanje trebali su odvojene naredbe za prelazak u novi red i za vraćanje na početak reda. Tako su nastale naredbe Line Feed (novi red) i Carriage Return (vrati se na početak reda), koje imaju brojčane kodove 10 i 13 (decimalno). Te su naredbe postale dio kodiranja znakova i na računalu, a koriste se i danas. Stoga većina uređivača teksta (editora) na računalu i dalje koristi kombinaciju kodova i za prelazak u novi

9 https://en.wikipedia.org/wiki/Unicode

red. Naravno, ti su brojčani kodovi nevidljivi, tj. editor ih ne prevodi i ne pokazuje kao znakove na ekranu; oni se prevode kao naredbe (a ne kao znakovi).

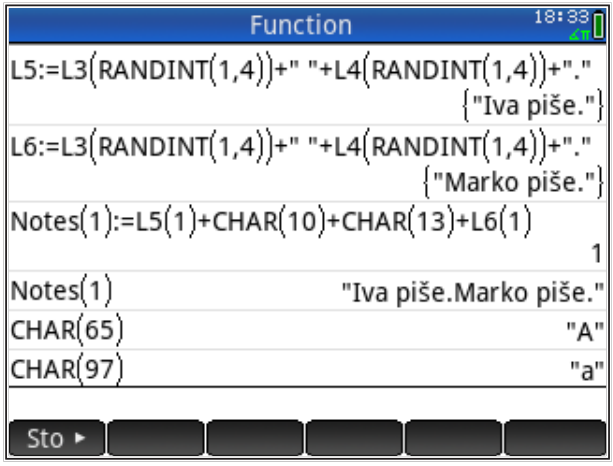
Sada smo spremni pogledati kako se na HP Prime bilješke u više redova pišu naredbama (dakle, bez ručnog odlaska u bilješke (Notes) i upisivanja teksta).

Prvo ćemo u liste L5 i L6 spremiti izjave generirane slučajnim brojevima

```
L5:=L3(RANDINT(1,4))+" "+L4(RANDINT(1,4))+"."
L6:=L3(RANDINT(1,4))+" "+L4(RANDINT(1,4))+"."
```

Zatim ćemo te dvije izjave spojiti u tekst koji ćemo upisati u bilješku 'proba' u dva reda, svaku izjavu u svoj red:

```
Notes(1):=L5(1)+CHAR(10)+CHAR(13)+L6(1)
```



S ekrana se ne vidi da je bilješka (1) napisana u dva reda; naprotiv, dobili smo spojene izjave. Razlog tome jest što naredbe 10-novi red i 13-na početak, nisu znakovi i ne pokazuju se na ekranu. Kada uđemo u uređivanje bilješki i tamo pogledamo, vidimo da je naš tekst zaista ispisan u dva reda.

Naravno, naredbu 'CHAR(10)+CHAR(13)' možemo spremiti u varijablu i sve zapisati kraće.

Zadatak: Definiraj tekstualnu ('string') varijablu 'NR' koja označava novi red i napiši prethodni primjer koristeći 'NR'.

Na prethodnom ekranu pokazano je i što radi naredba 'CHAR(kod)': ispisuje ASCII znak koji odgovara 'kodu' koji je upisan. Na taj način možemo dobiti sve ASCII naredbe i znakove (iako se naredbe ne vide na ekranu, svejedno su tu i djeluju u okruženju koje ih 'razumije', tj., može ih interpretirati). Funkcija 'CHAR(kod)' ima svoju inverznu funkciju 'ASC(„znak“)', koja ispisuje brojčani kod određenog znaka. Tako

```
ASC("A")
```

daje kao odgovor broj 65.

Spremanje funkcija

Nekoliko aplikacija HP Prime-a radi s funkcijama: *funkcije* ('Function'), *napredno prikazivanje* ('Advanced Graphing'), *rješavanje* ('Solve'), *crtanje u 3D* ('Graph 3D'); dvije zadnje aplikacije ne nalaze se u besplatnoj aplikaciji HP Prime. Funkcije se zadaju unutar svake aplikacije u simboličkom pregledniku tako da se upišu ručno, nakon čega se mogu koristiti i u drugim aplikacijama ili direktno iz ekrana *doma* ('Home'). Svaka aplikacija za definiranje funkcija ima rezerviranu varijablu koja počinje drugim (predefiniranim) slovom

'Function'	F
'Advanced Graphing'	V
'Solve'	E
'Graph 3D'	FZ

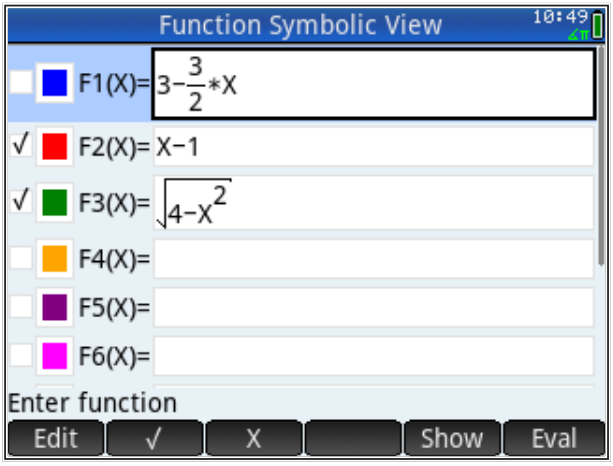
i nastavlja se brojem 1, 2, do 0, čime je omogućeno definiranje do 10 funkcija po aplikaciji. Postoji mehanizam zadavanja funkcija koji nadilazi ovo ograničenje, ali on ima neke druge specifičnosti i u ovom poglavlju ga nećemo spominjati.

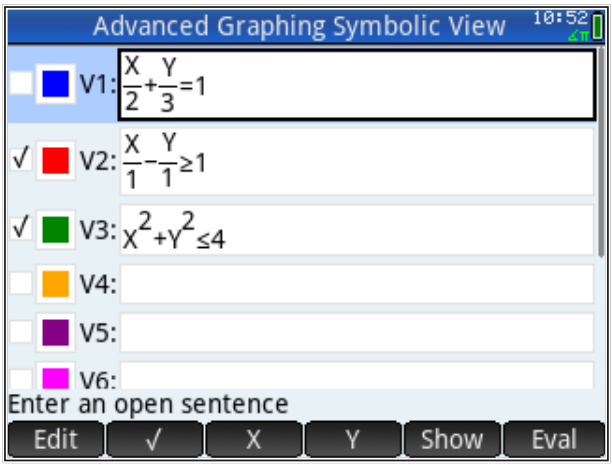
Ukoliko HP Prime koristimo u raznim područjima, vrlo brzo ćemo doći do broja od 10 funkcija, nakon čega treba brisati stare i upisivati nove funkcije. Bilo bi vrlo praktično da funkcije možemo spremati i jednostavno upisivati prema potrebi, a po mogućnosti i jednostavno razmjenjivati (npr., preko tekstualnih datoteka unutar e-maila).

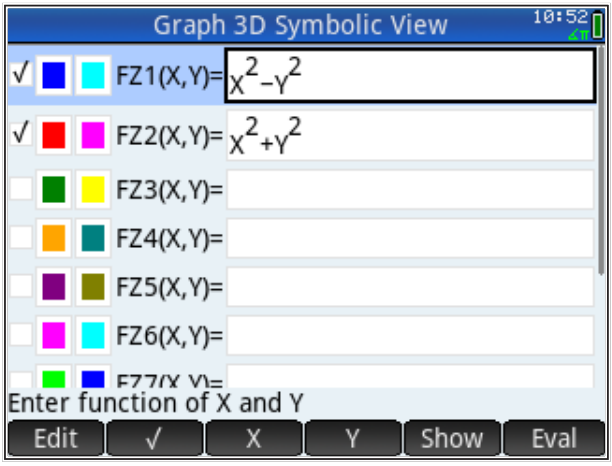
Spremanje funkcija u liste

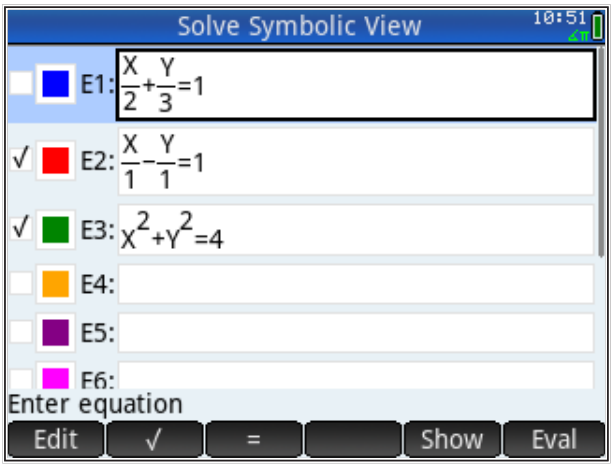
Pokazat ćemo kako se funkcije mogu kopirati iz aplikacije u listu i upisati iz liste u aplikaciju. Također, iz lista ćemo funkcije lako prebaciti u bilješke, i obratno. Na taj način moći ćemo u bilješkama načiniti biblioteku funkcija iz koje ćemo moći birati i upisivati one koje nam trenutno trebaju.

Pogledajmo koje funkcije trenutno imamo definirane u kalkulatoru



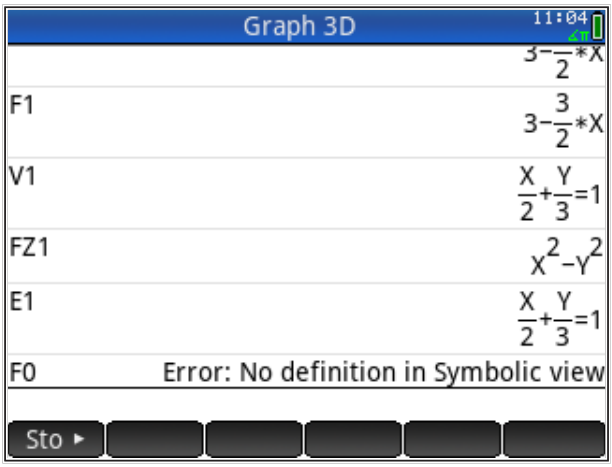

Advanced Graphing Symbolic View
10:52
V1: X/2+Y/3=1
√ V2: X/1−Y/1≥1
√ V3: X^2+Y^2≤4
V4:
V5:
V6:
Enter an open sentence
Edit
√
X
Y
Show
Eval


Graph 3D Symbolic View
10:52
√ FZ1(X,Y)= X^2−Y^2
√ FZ2(X,Y)= X^2+Y^2
FZ3(X,Y)=
FZ4(X,Y)=
FZ5(X,Y)=
FZ6(X,Y)=
Enter function of X and Y
Edit
√
X
Y
Show
Eval


Solve Symbolic View
10:51
E1: X/2+Y/3=1
√ E2: X/1−Y/1=1
√ E3: X^2+Y^2=4
E4:
E5:
E6:
Enter equation
Edit
√
=
Show
Eval

Ukoliko je potrebno, radi praćenja primjera, upišite navedene funkcije u odgovarajuće aplikacije.

Funkcija se na ekran kalkulatora poziva tipkanjem njenog imena, bez zagrada i varijable



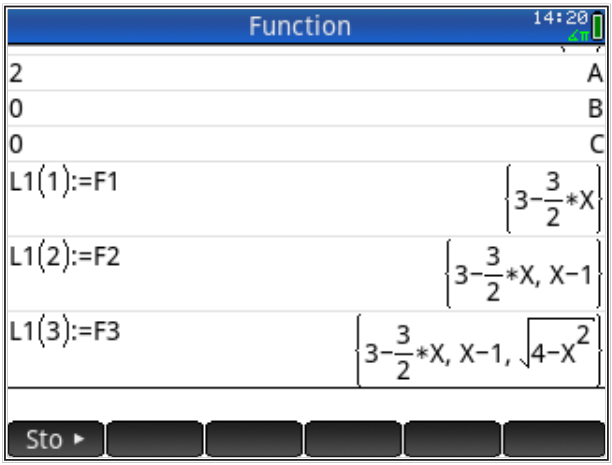
Funkcija mora biti definirana, inače slijedi poruka o grešci.

Ukoliko napišemo naziv funkcije sa zagradom i varijablom, aplikacija računa i ispisuje njenu numeričku vrijednost.

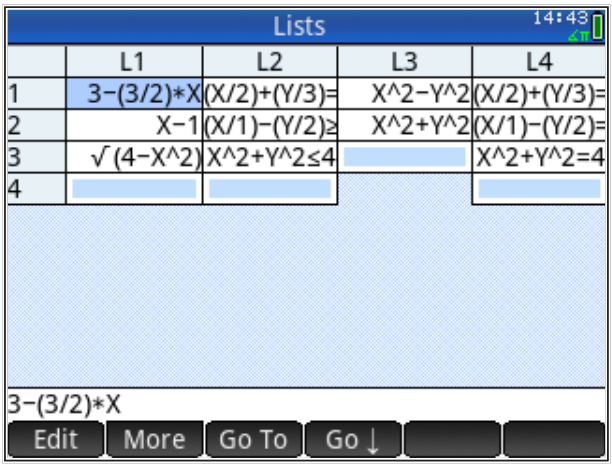
Sada ćemo funkcije spremiti u liste, a radi jednostavnosti izbrisat ćemo prethodni sadržaj listi

```
L1:={}
L2:={}
L3:={}
L4:={}
```

Prvo spremamo funkcije iz aplikacije *funkcije* u listu 'L1'



Na isti način spremamo funkcije iz aplikacija *napredno prikazivanje*, *rješavanje* i *crtanje u 3D* u liste 'L2', 'L3' i 'L4'. Ako liste pregledamo u prikazu lista ('List view', tipke Shift List), vidimo 4 liste, od kojih svaka ima po 3 funkcije.



Da se izbjegne zabuna, treba znati da je zapis funkcija u listama u algebarskoj notaciji ('Entry: Algebraic'), a u simboličkom prikazu u tiskarskoj notaciji ('Entry: Textbook'). Konverziju HP Prime vrši automatski.

Spremanje funkcija u liste omogućuje stvaranje biblioteke funkcija jer, imamo li 10 varijabli za liste (od L1 do L0), veličina indeksa unutar jedne liste ograničena je samo veličinom memorije. Dakle, možemo zapisati funkciju npr. na 12-o mjesto u listi L1

```
L1(12):=F3
```

i dobivamo kao sadržaj liste $\{3-\frac{3}{2}X, X-1, \sqrt{4-X^2}, 0,0,0,0,0,0,0,0, \sqrt{4-X^2}\}$.

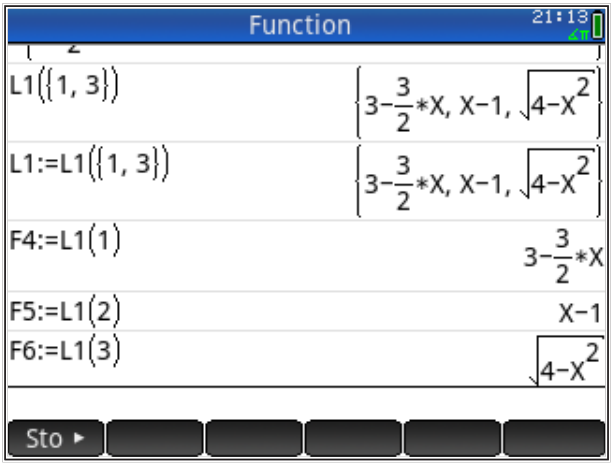
Listu možemo vratiti u prvobitno stanje tako da natrag u listu prepišemo samo prva tri elementa

```
L1:=L1({1,3}).
```

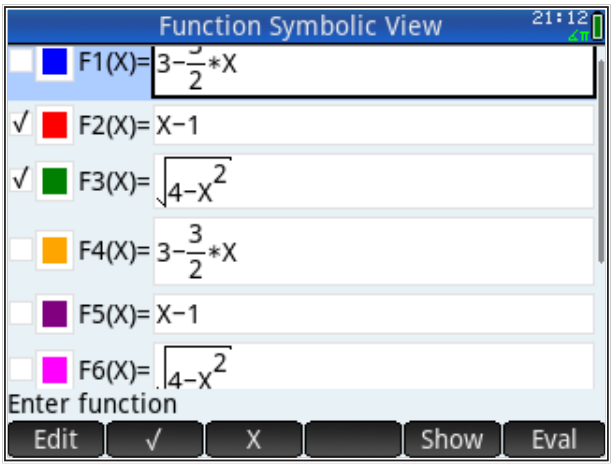
Vraćanje funkcija u aplikacije je jednostavno; prethodno smo imali

```
Lista(index):=Funkcija
```

a sada samo zamijenimo lijevu i desnu stranu prethodnog izraza.



Brojeve u nazivu funkcija smo promijenili iz $F1$, $F2$, $F3$ u $F4$, $F5$, $F6$, da se bolje uoči promjena



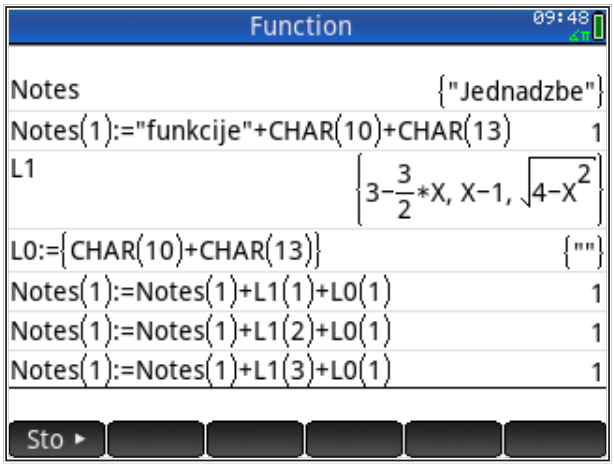
Sada imamo dvostruki broj funkcija, gdje je druga polovica kopija prve. Na isti način možemo vratiti i druge funkcije u odgovarajuće aplikacije.

Spremanje funkcija u bilješke

Funkcije spremljene u liste su praktične, ali nije ih jednostavno poslati mailom i ubaciti u tekst u programu za obradu teksta. Bilješke predstavljaju kompaktni tekst, koji se može kopirati i slati kao jedinstvena cjelina. Za prebacivanje funkcija iz lista u bilješke poslužit ćemo se postupkom koji smo prije definirali kod slaganja tekstualnih izjava.

Definiramo novu bilješku naziva 'Jednadzbe' (izbjegavamo dijakritičke znakove jer ih je sporo pisati). Tipkamo [*Shift*] [*Notes*] i 'New' pa ime 'Jednadzbe' i 'OK'.

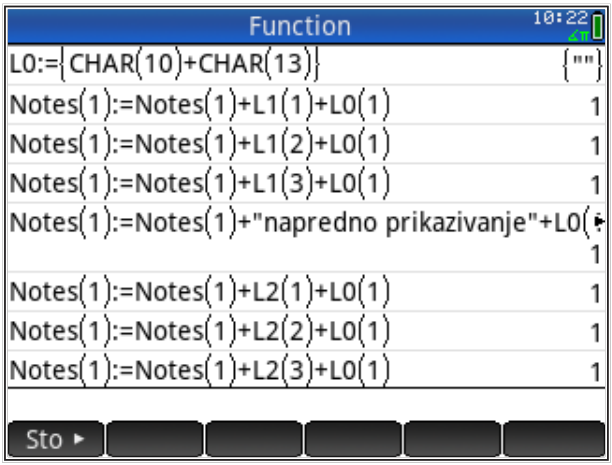
U bilješku upisujemo tekst iz koje aplikacije su funkcije, kao podsjetnik; prebacujemo se u način 'Home' i zatim dodajemo 3 jednadžbe iz aplikacije *funkcije*. Prvo smo ispisali sadržaj *Bilješke* da se uvjerimo da je nova bilješka 'Jednadžbe' na prvom mjestu (može biti još nekih drugih bilješki iza nje); ako to nije slučaj treba otići u *Bilješke* i ući i izaći iz bilješke 'Jednadžbe'; bilješka koja je zadnja uređivana, prva je u listi 'Notes'.



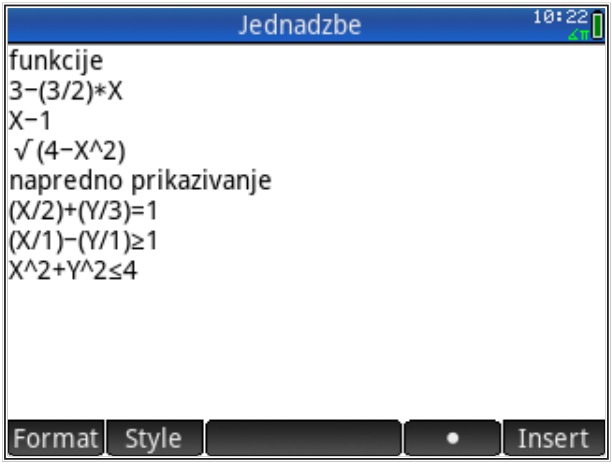
Primijetimo da smo znakove za novi red ubacili u listu 'L0' da skratimo tipkanje. Također, pozivanje naredbe za novi red traži da upisujemo indeks iz liste; dakle, ne 'L0' (jer to daje listu koja je omeđena vitičastim zagradama'{„“}'), nego 'L0(1)' (jer to daje samo element liste '“"'). Također, ne trebaju se tipkati sve naredbe; neki redovi se mogu kopirati i samo promijeniti indeks liste. Nakon ovog upisa, bilješka 'Jednadžbe' izgleda ovako:

Dodajemo funkcije iz drugih aplikacija, *napredno prikazivanje*, *rješavanje* i *crtanje u 3D.*

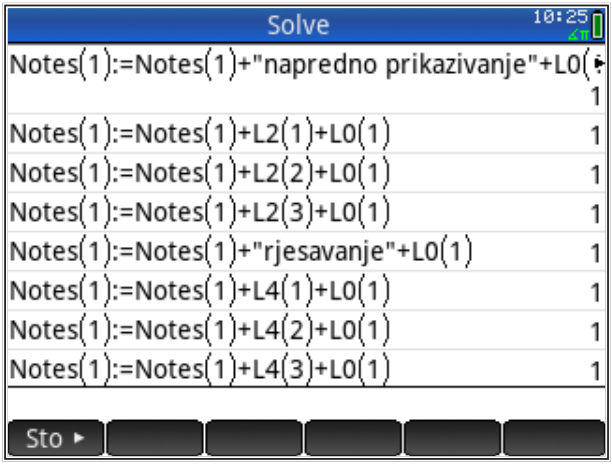
Upisujemo *napredno prikazivanje* ('Advanced Graphing'),



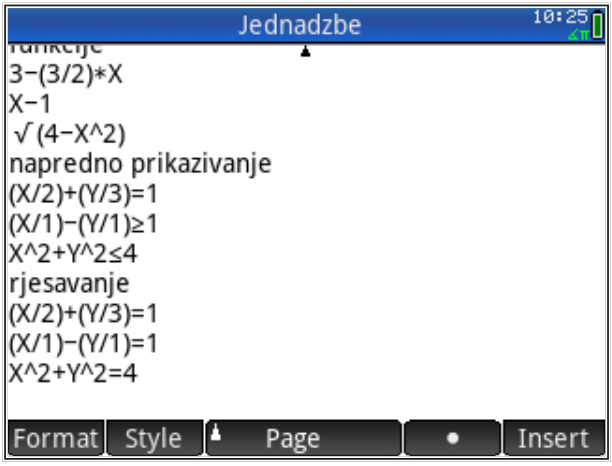
nakon čega bilješke ('Notes') izgledaju ovako:



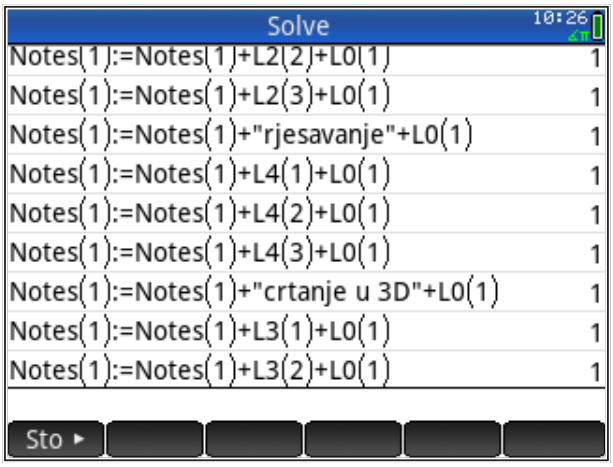
Upisujemo *rješavanje* ('Solve'),



nakon čega bilješke ('Notes') izgledaju ovako:



Upisujemo *crtanje u 3D* ('Graph 3D'). Ukoliko radimo u besplatnoj HP Prime aplikaciji s mobitela, taj dio se preskače (nema aplikacije 'Graph 3D'),



nakon čega bilješke ('Notes') izgledaju ovako:

Sada su sve funkcije upisane u bilješku 'Jednadzbe', od kuda ih iz virtualnog HP Prime kalkulatora jednostavno prebacujemo izvan aplikacije, u 'okoliš' smartphone uređaja ili osobnog računala.

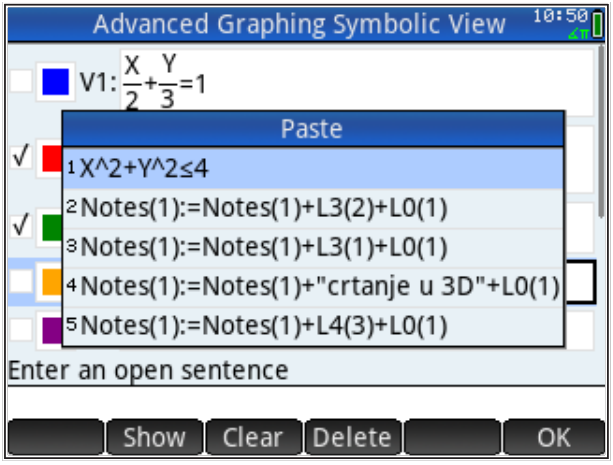
Otvorimo bilješku i iz menija aplikacije HP Prime biramo 'Edit' 'Copy' (bez da smo bilo što selektirali unutar bilješke). Na smartphone uređaju ili tabletu meni aplikacije dobivamo povlačenjem prsta na lijevom rubu virtualnog HP kalkulatora (iOS operativni sustav) ili dodirom 3 crtice ispred teksta 'HP Prime' na lijevom gornjem rubu aplikacije (Android operativni sustav).

Na taj je način cijela bilješka 'Jednadzbe' prebačena u međumemoriju ('buffer') smartphone uređaja/računala i može se umetnuti po želji. U nastavku je primjer umetnutog teksta bilješke

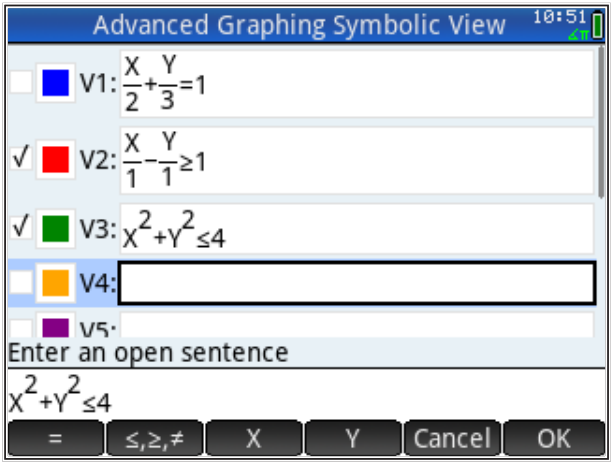
```
funkcije
3-(3/2)*X
X-1
√(4-X^2)
napredno prikazivanje
(X/2)+(Y/3)=1
(X/1)-(Y/1)≥1
X^2+Y^2≤4
rjesavanje
(X/2)+(Y/3)=1
(X/1)-(Y/1)=1
X^2+Y^2=4
crtanje u 3D
X^2-Y^2
X^2+Y^2
```

Ručno vraćanje funkcija iz bilješke vrši se pomoću tipki *Shift* *Copy* i *Shift* *Paste*. Odaberemo i otvorimo bilješku, postavimo kursor na mjesto od kojeg želimo početi kopirati (nije obavezno, ali je brže), tipkamo *Shift* *Copy* nakon čega se na dnu ekrana pojavljuju dodatne komande 'Begin', 'End', 'All', 'Cut', 'Copy'; ove komande omogućuju označavanje i kopiranje (ili izrezivanje, 'Cut') teksta. Odaberemo 'Begin', postavimo kursor na kraj funkcije i odaberemo 'End', pojavi se označeni dio

Biramo 'Copy' iz menija ispod ekrana i prelazimo u aplikaciju 'Advanced Graphing', odabiremo novu funkciju 'V4' i tipkamo *Shift* *Paste*, nakon čega vidimo



Odabiremo 'OK' iz menija ispod ekrana i pojavljuje se



gdje biramo 'OK' iz menija ispod ekrana. Time je u aplikaciji *napredno prikazivanje* zadana nova funkcija 'V4'.

Automatsko vraćanje funkcija u odgovarajuće aplikacije zahtjeva programiranje HP Prime kalkulatora i obradit ćemo ga u drugom dijelu koji se bavi programiranjem.

Biblioteka programa na magnetskim karticama za HP-67.

2 DIO - PROGRAMIRANJE

HP Prime - Jednostavno programiranje

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide* (iz menija 'Help') :

- *Catalogs and Editors: Program Catalog and Editor*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

```
L3:={"Pero", "Marko", "Ana", "Iva"}
```

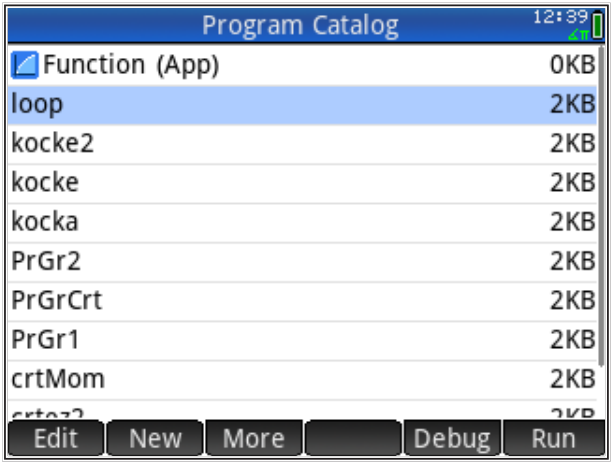
Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite pomoću 'Edit: Paste' naredbama iz menija kalkulatora pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

Ovo je poglavlje posvećeno uvodu u programiranje putem HP Prime kalkulatora. Naime, puna verzija HP Prime kalkulatora može se programirati u svom vlastitom programskom jeziku HP PPL (HP Prime Programing Language). HP PPL suvremeni je programski jezik sa svim osnovnim elementima koji se koriste i u drugim programskim jezicima, npr., Python, Matlab, Mathcad i sl. Savladavanje HP PPL-a omogućuje bezbolni ulazak u svijet programiranja na stolnim računalima.

Primjeri na kraju poglavlja služe kao motivacija za učenje složenijih tehnika programiranja. Najvažnije je uočiti da je pisanje programa usko vezano uz strukturu podataka koji opisuju parametre problema koji se želi riješiti.
Ovaj je tekst namijenjen kao uvod u programiranje studentima prve godine građevinarstva za koje se pretpostavlja da nemaju (ili imaju vrlo malo) prethodnog iskustva u programiranju. Sukladno, pokušalo se prikazati primjere vezane uz tehničku struku; očekuje se da će riješeni primjeri ilustrirati korisnost znanja programiranja i njegovu upotrebljivost u svakodnevnom radu i studiranju.

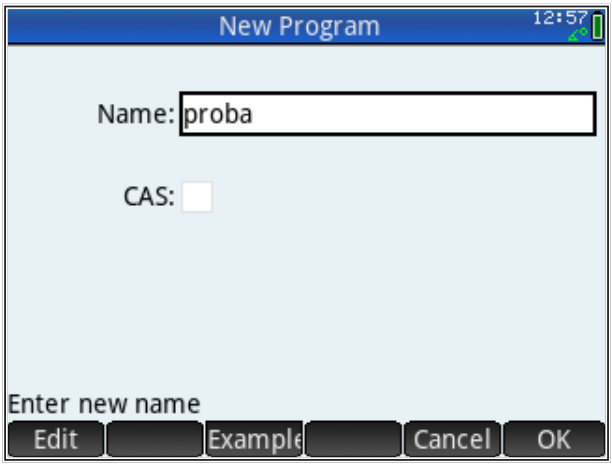
Pisanje programa na HP Prime

HP Prime ima poseban dio memorije odvojen za programe koje pišu korisnici. Ulazak u memoriju za programe ostvaruje se pritiskom na tipke *Shift* '*Program*' nakon čega se pojavljuje katalog svih programa koji se trenutno u memoriji kalkulatora.

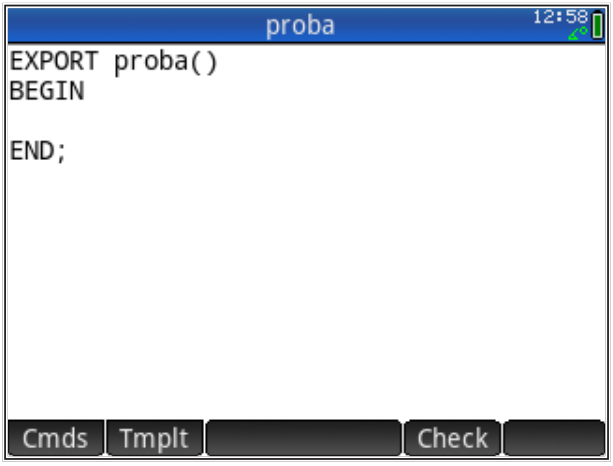


Prvi program u katalogu je 'Function (App)'. To je programsko sučelje za trenutno aktivnu aplikaciju i omogućuje programiranje ugrađene aplikacije. Drugi programi (bez ikone) su programi koje smo sami napisali i neovisni su od trenutno aktivne aplikacije. Ako nismo napisali niti jedan program, tada katalog sadrži samo programsko sučelje za aktivnu aplikaciju (označeno ikonom).

Novi program počinjemo pisati odabirom 'New' iz menija ispod ekrana.

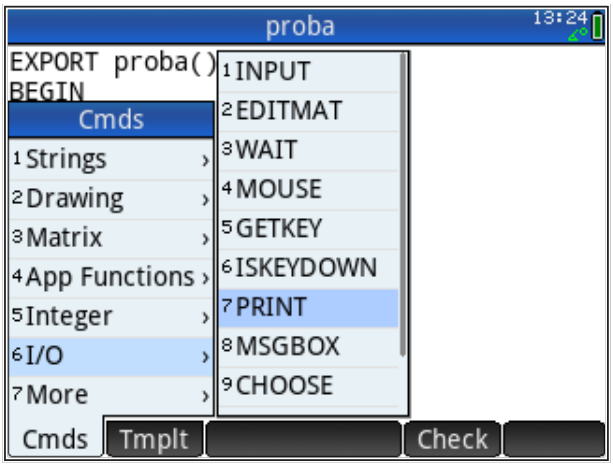


U okvir 'Name' upisujemo ime koje želimo pridružiti novom programu, a u okvir 'CAS' stavljamo kvačicu ako u programu želimo koristiti CAS funkcije. Ispod ekrana je meni gdje 'Edit' omogućuje upis imena a 'Example' poziva listu primjera gotovih programa koji dolaze uz HP Prime. 'Cancel' izlazi iz menija za zadavanje novog programa, a 'OK' potvrđuje upisane podatke (ime) i otvara ekran za pisanje naredbi programa. U našem primjeru, novi program sam nazvao 'proba' i pritiskom na 'OK' otvara se ekran za pisanje naredbi.

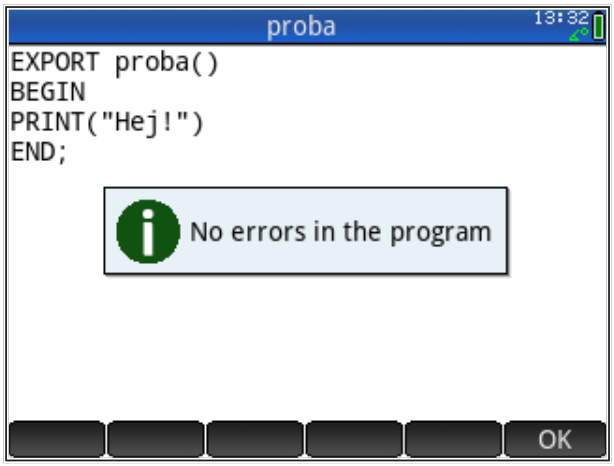


Osnovne naredbe koje svaki program treba imati već su napisane automatski; 'EXPORT' izvozi ime novog programa u katalog tako da je dostupan za pozivanje s 'Home' ekrana, 'BEGIN' i 'END' označavaju početak i kraj programa, a naše naredbe pišemo između te dvije oznake (tu treba postaviti kursor).
Ispod ekrana nalazi se meni s tri naredbe: 'Cmds', 'Tmplt' i 'Check'. 'Cmds' poziva katalog svih naredbi u HP PPL jeziku, 'Tmplt' poziva katalog gotovih primjera koji dolaze uz HP Prime, dok 'Check' provjerava sintaksu programa koji pišemo (jesmo li pravilno napisali sve naredbe).

Za sada ćemo u program upisati samo jednu naredbu: na ekranu ispisati pozdrav „Hej!“. Ima više naredbi za ispis poruka i podataka, a odabrat ćemo naredbu 'PRINT' koja ispisuje rezultat ili tekst na posebni ekran, tzv., 'Terminal'. Naredbu 'PRINT' možemo napisati putem tipkovnice ili pozvati iz menija (valja paziti da je kursor između 'BEGIN' i 'END').

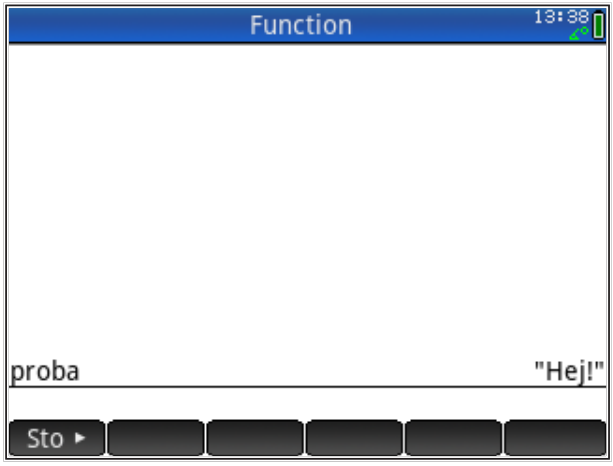


Između zagrada naredbe 'PRINT' napišemo poruku koju želimo (ako je tekst, mora biti unutar navodnika). Odabirom 'Check' u meniju ispod ekrana provjeravamo ima li grešaka u sintaksi programa (ali ne i logici!).



Odaberemo 'OK' i nakon toga izlazimo iz ekrana za pisanje programa, npr., pritisnemo tipku 'Home' ili *Esc*.

Program pozivamo iz 'Home' ekrana tipkanjem njegovog imena i, nakon toga, pritiskom na *Enter*. Pojavljuje se ekran 'Terminal' na kojemu je poruka; pritiskom na bilo koju tipku izlazimo iz 'Terminal' ekrana i vraćamo se u 'Home' ekran.



Višekratnim pozivanjem programa pojavljuje se 'Terminal', ali bez brisanja prethodnog sadržaja; sadržaj ekrana 'Terminal' mora se izbrisati tipkom *Del* ili naredbom iz programa.

Dokumentiranje programa

Svaki računalni program treba precizno dokumentirati jer se ideje i pretpostavke koje je programer koristio prilikom izrade programa vrlo brzo zaborave. Stoga komentare valja pisati unutar programa (komentari su tekst iza posebnog znaka, tako da ga računalo ignorira te se ne obrađuje unutar programa). Treba navesti ulazne parametre, izlazne parametre, kao i model koji se koristi za povezivanje ulaznih i izlaznih parametara. Također, treba navesti eventualna ograničenja programa i područje primjene. Na kraju, treba dati barem jedan primjer korištenja programa.

Osnovni elementi programa

Svaki računalni program ima, ovisno o namjeni, ove elemente:

1. Unos podataka
2. Obrada podataka (izračun)
    1. grananje
    2. petlje
3. Ispis rezultata

Unos podataka je dio programa kojim zadajemo parametre koje želimo da računalni program obradi (učini nešto s njima). Svaki složeniji računalni program treba ulazne podatke. Oni omogućuju fleksibilnost programa, tj. jedan te isti program koristimo za više vrijednosti parametara. Na taj način program postaje univerzalni recept za rješavanje svih slučajeva neke grupe zadataka ili problema.
U gornjem primjeru nemamo zadane parametre, a program ispisuje uvijek isti pozdrav.

Ulazne parametre u HP PPL možemo zadati na više načina, od kojih ćemo pokazati nekoliko.

1. Najjednostavnije je da parametre preuzmemo iz varijabli HP Prime kalkulatora, tj., varijable koje smo definirali u 'Home' načinu koristimo u programu. To je praktično, ali zahtjeva dobro dokumentiranje programa (moramo znati koju smo veličinu spremili u koju varijablu).

2. Drugi način je predvidjeti zadavanje parametara pri pozivu programa. Radi se o vrijednostima koje se navode u zagradi iza imena programa, npr., 'proba(Ime)'. Pritom, varijable možemo jednostavno nabrajati, što je zadovoljavajuće za relativno mali broj parametara. Kod zadavanja

parametara pri pozivu programa (u zagradi) treba poznavati strukturu programa da bismo znali koliko parametara i kakvog tipa (brojke, tekst, lista, vektor, matrica, itd.) trebamo zadati.

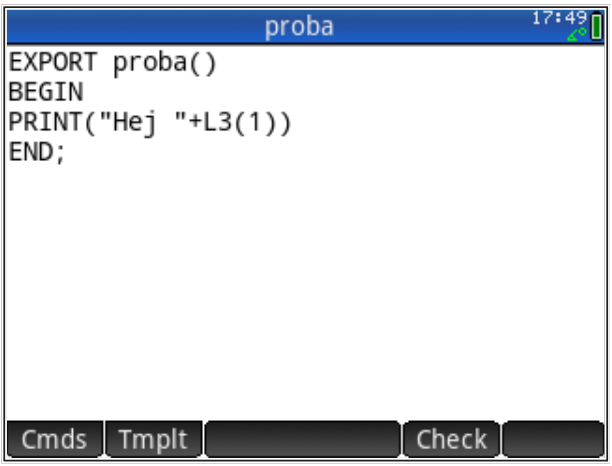
3. Zadavanje parametara nabrajanjem nije pregledno pa se možemo poslužiti naredbom za upis parametara, 'INPUT'. Pozivanje naredbe 'INPUT' otvara poseban ekran na HP Prime s naslovom, tekstom koji opisuje što treba upisati i okvirom gdje se upisuju podaci. Naredba omogućuje upis jednog ili više parametara odjednom.
4. Parametre možemo zadavati i tako da biramo opcije iz ponuđenih menija. Na taj se način smanjuje mogućnost upisa pogrešnog parametra (ne možemo ništa sami upisati, samo odabiremo ponuđeno). Za takav upis možemo koristiti naredbu 'CHOOSE(lista)', gdje u listi nabrajamo parametre koje je moguće odabrati.

Pokažimo na našem jednostavnom primjeru kako izgleda upisivanje parametara. Pretpostavimo da u program za pozdrav želimo dodati ime onoga koga se pozdravlja, pri čemu to ime treba biti parametar jer bismo inače trebali pisati poseban program za svakoga koga želimo pozdraviti.

1. U 'Home' načinu spremimo u varijablu ime koje želimo, ali na HP Prime varijabla za upis teksta može biti lista. Sjetimo se da u listi L3 imamo definirana četiri imena (ako nemamo, treba upisati)

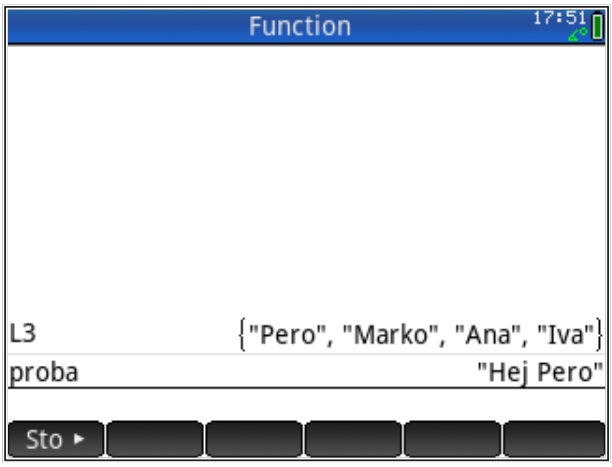
   ```
   L3:={"Pero", "Marko", "Ana", "Iva"}
   ```

   Sada znamo da ime mora biti u listi L3 i u programu predvidimo pozivanje imena iz liste L3. Takav program izgleda ovako



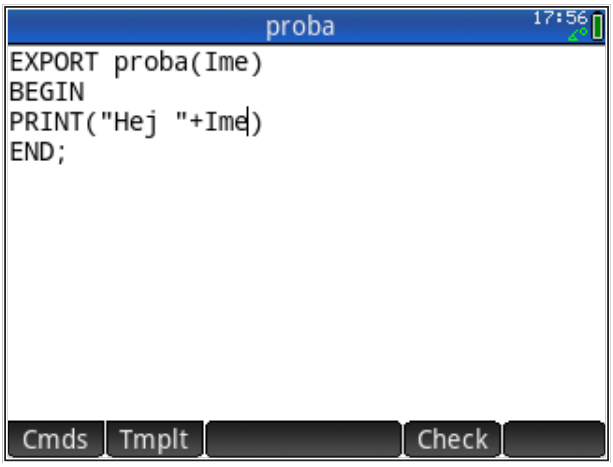
Vidimo da smo uskličnik zamijenili razmakom i dodali smo znak '+' koji spaja tekst i pozvali smo prvo ime iz L3 s L3(1).

Kada pokrenemo program on ispisuje

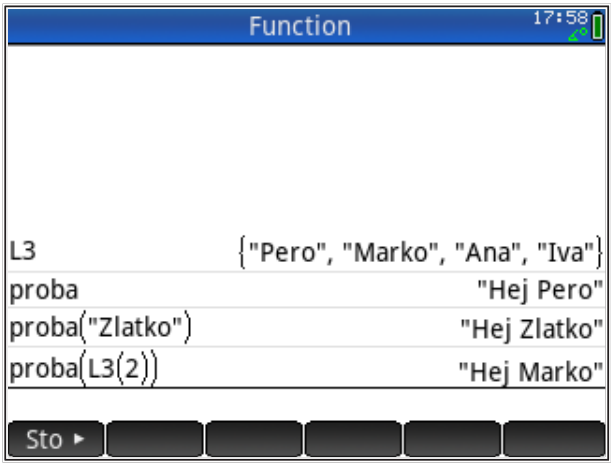


Vidimo da mi moramo odrediti koje ime će program pozvati.

2. Ime možemo zadati kao parametar pri pozivanju programa (u zagradi), pri čemu modificiramo program 'proba' koji sada izgleda ovako (ako želimo zadržati prethodni program, onda moramo zadati novo ime, npr., 'proba2', itd.).



Program se poziva ovako

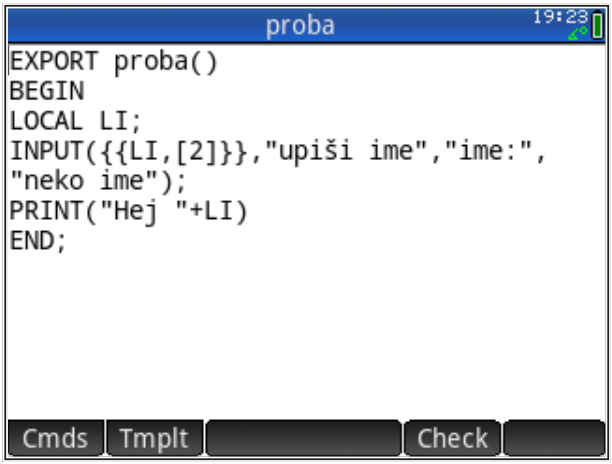


Primijetimo veću fleksibilnost u pozivanju programa.

3. Pozovimo naredbu 'INPUT' za zadavanje parametra. Kada se 'INPUT' koristi za zadavanje brojčanog parametra, upotreba mu je vrlo jednostavna, kao u korisničkom priručniku. Međutim, zadavanje tekstualnog parametra je složeniji zadatak. Sljedeća sintaksa naredbe 'INPUT' nije opisana u HP Prime korisničkom priručniku

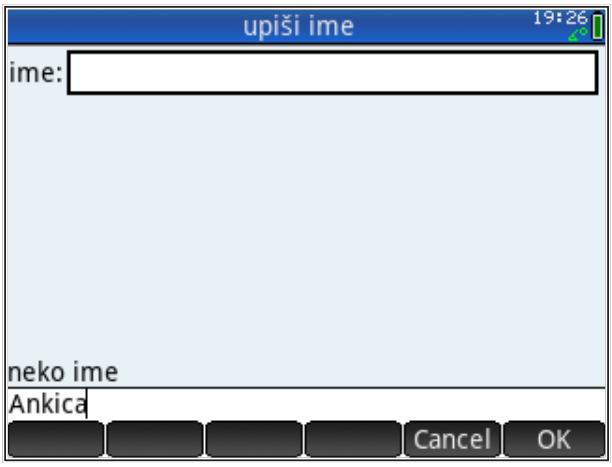
   ```
   INPUT({{var,[TYPE]}},"naslov","ime","pomoćna linija");
   ```

   gdje je 'TYPE' tip varijable prema HP specifikaciji (vidi korisnički priručnik), a tekst (string) varijabla ima 'TYPE=2'.

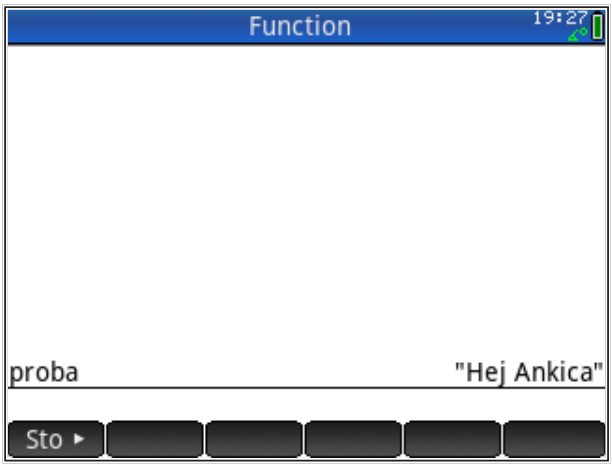


Vidimo da smo varijablu 'LI', koju koristimo u 'INPUT', prvo deklarirali kao LOKALNU varijablu.

Prilikom pozivanja programa pojavljuje se ekran za unos podataka



Prvo odabiremo 'Edit' a zatim 'OK', pojavljuje se ekran 'Terminal', a onda, nakon pritiska na neku tipku, i ekran 'Home'.



LOKALNE I GLOBALNE VARIJABLE u HP PPL razlikuju se u dosegu ('scope'). Lokalne varijable dostupne su samo unutar programa u kojem su definirane i označavaju se naredbom

```
LOCAL ime varijable;
```

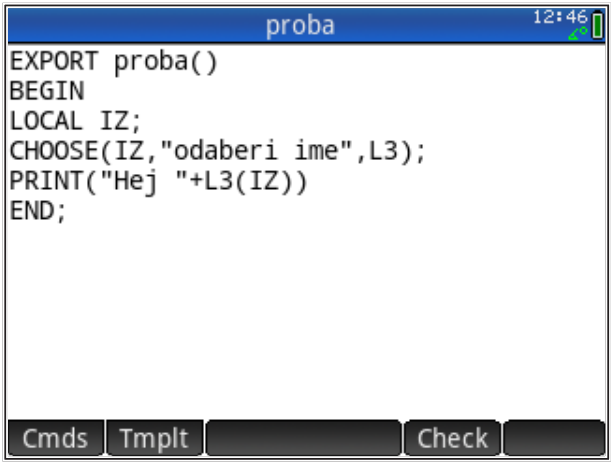
iza 'BEGIN' oznake, pri čemu varijabli može biti više i odvojene su zarezom.

Globalne varijable dostupne su u svim programima HP Prime kalkulatora (i na 'Home' ekranu), a označavaju se naredbom

EXPORT `ime varijable;`

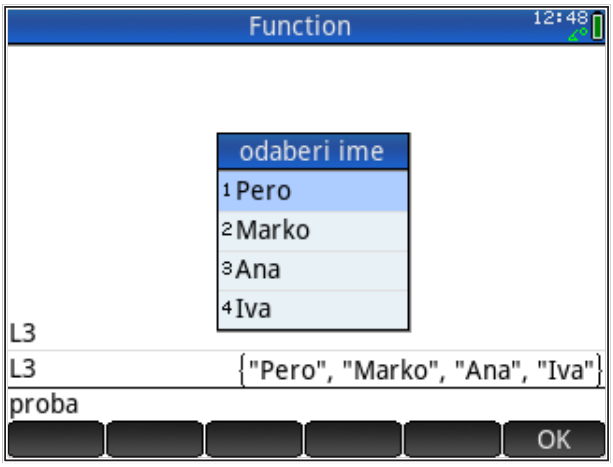
prije 'BEGIN' oznake, pri čemu može biti više varijabli koje su odvojene zarezom.

4. Pozovimo naredbu 'CHOOSE' koja omogućuje biranje parametara iz izbornika (to je moguće izvesti i s naredbom INPUT ali je naredba CHOOSE preporučena za unos predefiniranih podataka iz izbornika). Program sada izgleda ovako



Varijabla 'IZ' poprimit će vrijednost od 1 do 4, ovisno koje ime ćemo odabrati iz menija. Iskoristili smo sadržaj liste L3 da si uštedimo na tipkanju; ujedno je i program fleksibilniji jer ga ne moramo mijenjati svaki puta kad poželimo promijeniti izbornik za upis parametara.

Pozivanjem programa iz 'Home' ekrana pojavljuje se izbornik



Vidimo da se izbornik pojavljuje na 'Home' ekranu. Ako izaberemo “Ana” pojavljuje se pozdrav “Hej Ana”.

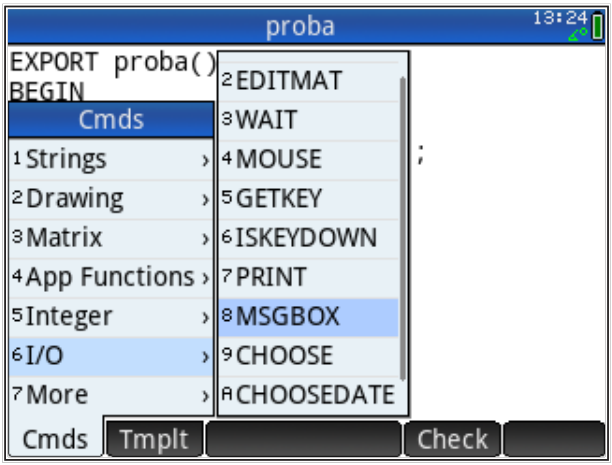
Ispis poruka i rezultata

Računalni program rezultate može dati na više načina, ovisno o problemu i ovisno o koncepciji računalnog programa. Rezultat se može jednostavno ispisati, što je pogodno kada imamo manji opseg rezultata.

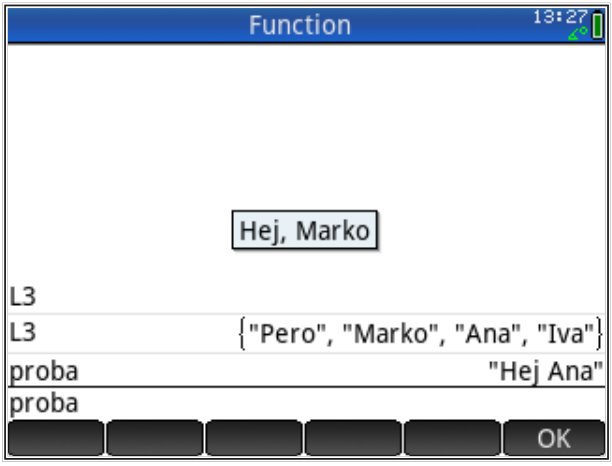
Osnovna naredba za ispis je 'PRINT', koji ispisuje na poseban ekran 'Terminal', a ispis se kasnije vidi i na ekranu 'Home'.

Više mogućnost pri ispisu daje naredba 'MSGBOX' ('poruka'), koja može proslijediti i odgovor korisnika. Moguća je forma naredbe bez odgovora i s tipkama 'OK' i 'Cancel'.

Za naš primjer, upotreba 'MSGBOX' izgleda ovako



Pokrenemo program i prvo se pojavi izbornik, a zatim odgovarajuća poruka



Da bi zatvorili poruku trebamo odabrati 'OK'; kalkulator se prebacuje u ekran 'Home', ali sada ne ispisuje poruku, nego rezultat ovisan o tipki koju smo odabrali (OK = 1, Cancel = 0). Taj odgovor (1 ili 0) je ujedno postavljen u varijablu 'Ans'.

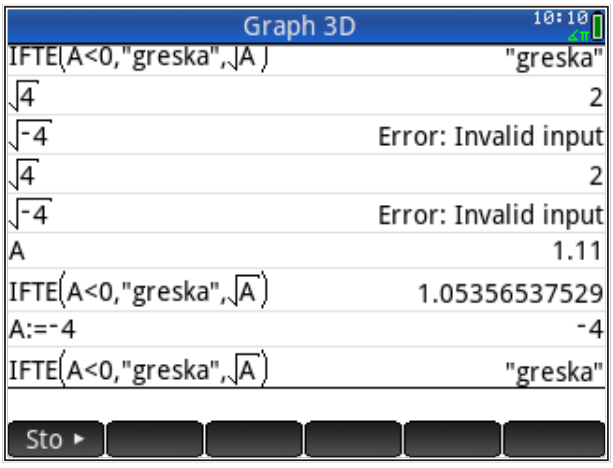
Osim direktnog ispisa rezultata na ekran kalkulatora, rezultate se može ugraditi u postojeće varijable, npr., u tablicu tabličnog kalkulatora, te onda dalje koristiti.

Najsloženiji oblik prikaza rezultata jest grafički prikaz, gdje su rezultati ugrađeni u grafičke parametre problema koji razmatramo (npr., prikaz trase prometnice, koju je računalni program odredio na temelju nekih zadanih parametara, prikaz pomaka ili naprezanja neke konstrukcije, koje je program izračunao prema nekom modelu koji opisuje relevantna svojstva te konstrukcije).

Grananje tijeka programa

Grananje u programu trebamo kada tijek programa ovisi o nekom parametru na način da parametar uvjetuje potrebu za različitim naredbama. Klasični primjer je računanje korijena (ukoliko ne želimo računati s kompleksnim brojevima): negativna vrijednost traži da se korijen računa iz pozitivnog broja, a predznak se postavlja ovisno o predznaku. HP Prime kalkulator u običnom načinu ne zna izvaditi korijen iz negativnog broja (zna u CAS načinu). Tako $\sqrt{-4}$ daje poruku 'Error: Invalid input'. Takvu poruku ne želimo tijekom izvršavanja našeg programa pa se treba osigurati, npr.,

```
A:=-4
IFTE(A<0,”greska”,sqrt(A))
```



Naredba 'IFTE(test,DA,NE)' prvo izvršava 'test' i, ako je uvjet testa ispunjen (test je točan), izvršava se naredba koja stoji na mjestu 'DA', a ako uvjet nije ispunjen, izvršava se naredba pod 'NE'. Naravno, nismo morali napisati poruku „greška“, mogli smo staviti da je rezultat korijen iz pozitivnog broja kojemu smo nadopisali 'i' ili sl.

Naredba 'IFTE' cijela stoji u jednom redu, a naprednija i fleksibilnija varijanta te naredbe je blok 'IF THEN ELSE'

```
IF
test
THEN
... naredbe ako je test ispunjen
ELSE
... naredbe ako test nije ispunjen
END;
```

Blok naredba na 'test' omogućuje kompleksniji odgovor u više programskih linija. Za višestruka testiranja moguće je koristiti u višestruke naredbe 'IF THEN ELSE'. Za izbor željene vrijednosti iz većeg skupa mogućnosti u pravilu koristimo blok 'CASE END' gdje svaki 'CASE' ima svoj 'IF THEN END'.

```
CASE
IF test1 THEN ... naredbe ako je test ispunjen   END;
IF test2 THEN ... naredbe ako je test ispunjen   END;
IF test3 THEN ... naredbe ako je test ispunjen   END;
DEFAULT ... naredbe koje se izvršavaju ako nijedan test nije
zadovoljen
END;
```

Primjer upotrebe 'CASE END' ilustriran je u programu *bacanje kocke*.

Ponovljeno izvršavanje dijelova programa (petlje)

Često imamo slučaj da se pojedini dijelovi programa trebaju izvršavati više puta, nepromijenjeni ili uz minimalne izmjene vezane uz vrijednost nekog parametra pa kažemo da se u programu izvršava *petlja* ('loop').

Naredba koja bezuvjetno izvršava petlju zadani broj puta je naredba 'FOR', koja definira blok s pripadnom naredbom 'END;' na kraju. U primjeru pišemo novi program koji se zove 'loop' i kreira listu od '1' do 'n' s razmakom 'r'.

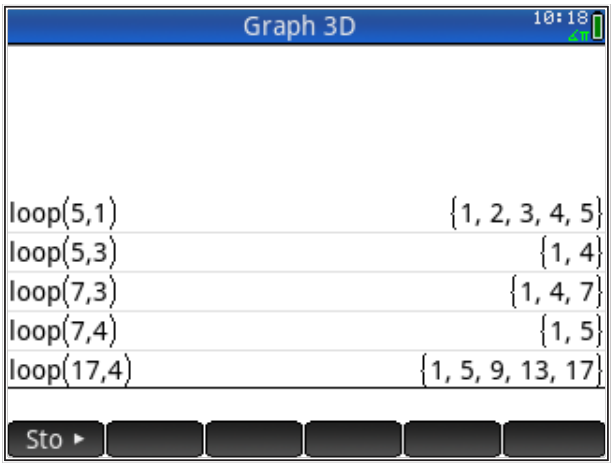
```
EXPORT loop(n,r)
//proizvodi listu brojeva
//n - najveci broj liste
//r - razmak izmedju brojeva
BEGIN
```

```
LOCAL i,lsta;
lsta:={};
FOR i FROM 1 TO n STEP r DO
lsta:=CONCAT(lsta,{i});
END;
RETURN lsta;
END;
```

Tekst iz dvije kose crte // je komentar, nije dio programa i ne izvršava se. Kosa crta '/' dobije se pritiskom na tipke *Shift* *Chars*.

Naredba 'CONCAT' pridodaje element kraju liste (uvrštava ga u listu, tvoreći novu listu s dodatnim elementom na kraju).

Pozivanje programa 'loop' s različitim parametrima daje liste

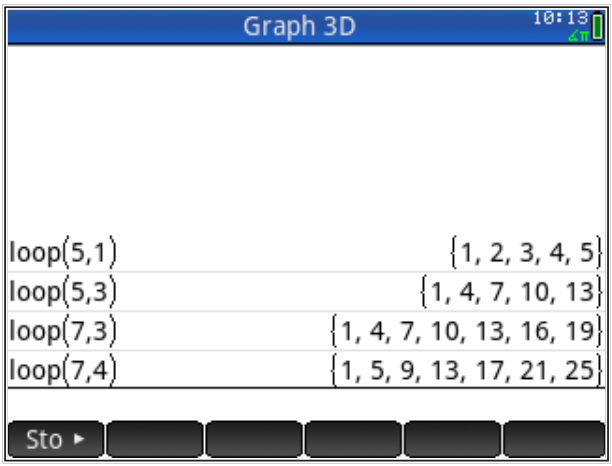


Slična naredba za izvršavanje programa u petlji je 'WHILE END'.

```
WHILE test DO
... naredbe koje se izvršavaju ako je 'test' zadovoljen
END;
```

Naredbu 'WHILE' koristimo kada ne znamo unaprijed broj izvršavanja naredbi u petlji, nego koristimo 'test' da vidimo do kada je željeni uvjet zadovoljen.

Zadatak: Preradi program tako da 'n' ne predstavlja broj do kojeg lista ide, nego broj elemenata u listi (savjet: samo promijeni do kojeg broja ide petlja 'FOR'). Takav program 'loop' s različitim parametrima daje liste



Primjer upotrebe ilustriran je u i varijanti programa *bacanje kocke*.

Programiranje HP PPL - jednostavni primjeri

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime User Guide* (iz menija 'Help') :

- *Programming in HP PPL: The HP Prime programming language*
- *Math menu: Probability*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

```
nema pretpostavki o stanju varijabli u memoriji
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite s 'Edit: Paste' naredbama iz menija kalkulatora pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

U ovom poglavlju prikazat ćemo par jednostavnih programa koji će nam poslužiti kao praktični uvod u osnove programiranja HP PPL (prime programming language).

Bacanje kocke

Ovo je malo modificirani primjer iz Korisničkog priručnika HP Prime kalkulatora. Vrlo je jednostavan i može dobro poslužiti kao ilustracija zadavanja parametara i prikaza rezultata programa.

Program simulira bacanje jedne ili dvije standardne kocke sa 6 strana; 'listing' je u nastavku.

```
EXPORT kocke(brKc)
// brKc - broj kocaka (1 ili 2)
// n    - broj strana (6)
BEGIN
LOCAL n;
n:=6;
IF brKc=1 THEN
 RETURN RANDINT(1,n);
ELSE
 RETURN RANDINT(1,n)+RANDINT(1,n);
END;
END;
```

Navedeni program je jako pojednostavljen, npr., simulira bacanje jedne kocke ako se pozove s parametrom 1, a bacanje dvije kocke ako se zada bilo koji drugi parametar (ne samo broj 2). Taj je pristup u redu ako program ne namjeravamo proširiti tako da može simulirati bacanje bilo kojeg broja kocaka.

Ulazni parametar:

```
brKc - broj kocaka koje bacamo
```

Lokalni parametar:

```
n - broj strana kocke
```

Struktura programa:

Program pozivamo s jednim parametrom koji treba biti cijeli broj, varijabla 'brKc' koja može poprimiti vrijednost '1' ili neku drugu vrijednost.

Lokalna varijabla 'n' definira se kao 'n=6'; uvijek bacamo kocku sa šest strana.

Testiramo na vrijednost varijable 'brKc':

- ako je brKc=1, onda pozivamo funkciju RANDINT(1,6), koja kao rezultat daje slučajni broj u rasponu od 1 do n (n=6),
- ako je brKc 1, onda dva puta pozivamo funkciju RANDINT(1,6) i zbrajamo rezultat te dobivamo slučajni broj u rasponu od 2 do 2*n,
- odgovarajući slučajni broj se vraća (na ekran 'Home') kao rezultat nove funkcije 'kocka'.

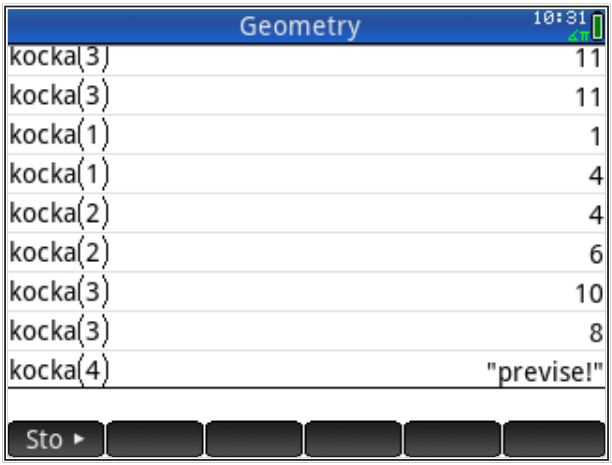
Zadatak: Prikazani program modificiraj za bacanje 3 kocke istovremeno (koristi naredbu 'CASE').

Rješenje: Modificirat ćemo gornji program tako da naredbe ovise o broju kocaka koje bacamo.

```
EXPORT kocka(brKc)
// brKc - broj kocaka (1, 2 ili 3)
// n    - broj strana (6)
BEGIN
LOCAL n;
n:=6;
CASE
IF brKc=1 THEN
   RETURN RANDINT(1,n); END;
IF brKc=2 THEN
   RETURN RANDINT(1,n)+RANDINT(1,n); END;
IF brKc=3 THEN
   RETURN RANDINT(1,n)+RANDINT(1,n)+RANDINT(1,n);
   END;
DEFAULT RETURN "previse!";
END;
END;
```

Primijetimo da s naredbom 'END;' označavamo kraj bloka definiranog početnom naredbom kojom blok započinje ('IF' ili 'CASE' itd.). U slučaju da parametar prilikom pozivanja funkcije nije 1, 2 ili 3, program ispisuje tekst 'previše', dajući do znanja da je načinjen da radi za najviše 3 kocke.

Primjer opetovanog pozivanja programa vidimo na slici



Zadatak: Prikazani program poopći tako da simulira bacanje bilo kojeg broja kocaka istovremeno. *Pomoć: Upotreba naredbe CASE nije u potpunosti prikladna za rješavanje ovog problema jer treba unaprijed znati maksimalni broj kocaka. Elegantno rješenje temelji se na zbrajanju pojedinačnih bacanja kocke.*

Rješenje:

```
EXPORT kocke2(brKc)
// brKc - broj kocaka (proizvoljno)
// n    - broj strana (6)
BEGIN
LOCAL n,i,sum;
brKc:=ABS(IP(brKc));
n:=6;
i:=1;
sum:=0;
WHILE i≤brKc DO
 sum:=sum+RANDINT(1,n);
 i:=i+1;
END;
RETURN sum;
END;
```

U program smo kroz naredbu 'brKc:=ABS(IP(brKc))' ugradili kontrolu ulaznog parametra, tj. broja kocaka koje bacamo. 'ABS' daje apsolutnu vrijednost broja (za slučaj da je netko stavio predznak '-'), 'IP' daje cjelobrojni dio broj (Integer Part) za slučaj da je netko upisao decimalni broj.

Drugi pristup je rješenje temeljeno na zbrajanju elemenata liste, naredba $\sum LIST$ (ne može se napisati, bira se iz liste naredbi, tipka 'kovčežić'). Također, funkciju 'RANDINT' upotrijebit ćemo s tri parametra; funkcija 'RANDINT(n,a,b)' daje 'n' slučajnih brojeva u intervalu [a,b].

```
EXPORT kocke(brKc)
// brKc - broj kocaka (neograničeno)
// n    - broj strana (6)
BEGIN
LOCAL n;
brKc:=ABS(IP(brKc));
n:=6;
 RETURN ΣLIST(RANDINT(brKc,1,n));
END;
```

Kao što vidimo, najopćenitiji program je najelegantniji i najkraći!

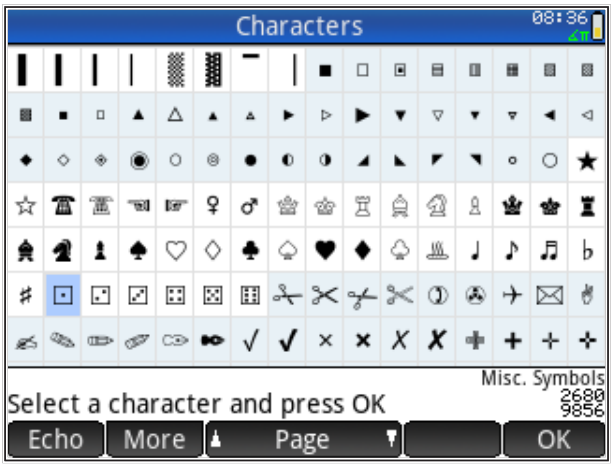
Naravno, u programe bi trebalo ugraditi zaštitu od prevelikog parametra jer ako varijabla brKc postane npr., 1000000, kalkulator će se 'zaglaviti' zbog prevelikog broja operacija koje zauzimaju previše memorije i traju predugo.

Napomena: Istovremeno bacanje više kocaka istovjetno je ponovljenom bacanju jedne kocke, što nam omogućuje da radimo statistiku bacanja.

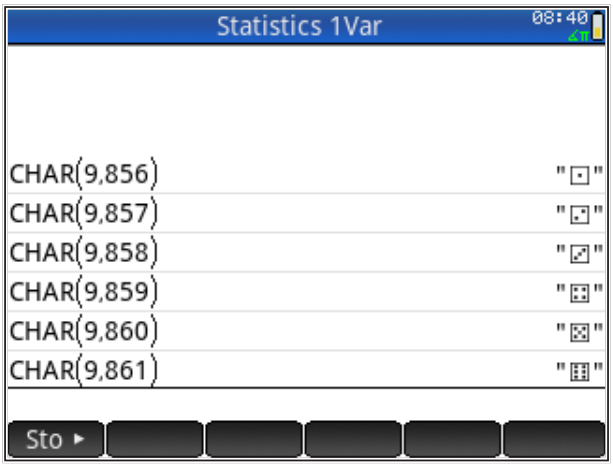
Grafički ispis rezultata

Ispis rezultata programa za simulaciju bacanja kocke je prilično neprivlačan i nezanimljiv te bi grafički prikaz bio puno privlačniji. No, ispis na grafički ekran je nešto složeniji od ispisa na standardni, tekstualni ekran.

Nasreću, postoji kompromis: ispis putem grafičkih znakova. Naime, HP Prime ima veliki broj stranica s različitim znakovima koji su namijenjeni ispisu na raznim jezicima i ispisu koji simulira grafički prikaz. Prikaz svih raspoloživih znakova dobijemo pritiskom na tipke *Shift* *Chars*. Nakon nekoliko uzastopnih pritisaka na 'Page V' dobijemo ekran



Odaberemo znak koji nas zanima (kocka s jednom točkom u sredini) i desno dolje vidimo njenu brojčanu oznaku, '9856'. Odabirom susjednih kocaka vidimo da kocka sa 6 točaka ima oznaku '9861'. Sada lako možemo pozivati simbole kocke na ekran naredbom 'CHAR(brojčani kod)'. Na slici vidimo sve kocke

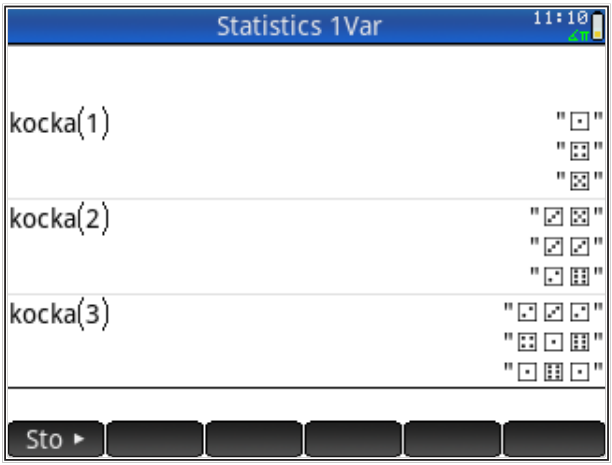


Sada možemo izmijeniti program 'kocka' tako da ima izlaz preko grafičkih znakova

```
EXPORT kocka(brKc)
// brKc - broj kocaka (1 ili 2)
// n    - broj strana (6)
BEGIN
LOCAL n;
n:=6;
CASE
IF brKc=1 THEN
   RETURN CHAR(9855+RANDINT(1,n));
   END;
IF brKc=2 THEN
   RETURN CHAR(9855+RANDINT(1,n))
   + CHAR(9855+RANDINT(1,n));
   END;
IF brKc=3 THEN
   RETURN CHAR(9855+RANDINT(1,n))
   + CHAR(9855+RANDINT(1,n))
   + CHAR(9855+RANDINT(1,n));
   END;
DEFAULT RETURN "previse!”;
END;
END;
```

Vidimo da kod HP Prime naredbe možemo razbiti na više redova, kalkulator ih 'vidi' kao da su napisane u jednom redu.

Primjeri rezultata iz programa su na slici

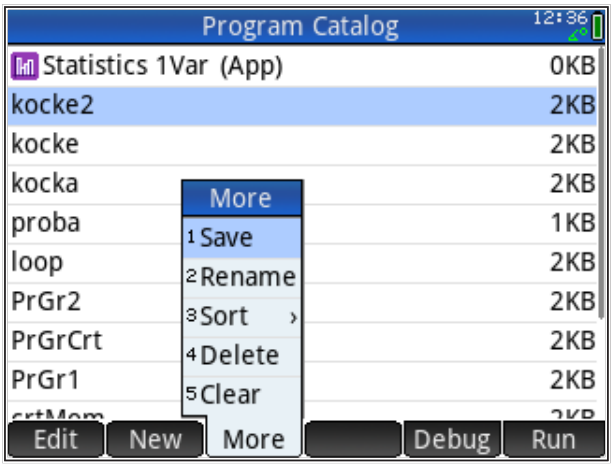


Napomena: Program koji daje samo ukupni zbroj kocaka ne može ovako prikazivati rezultate, nego nam treba zapis za svaku kocku posebno (kao što ćemo vidjeti i u slijedećem poglavlju).

Bacanje kocke sa statistikom

Statistika bacanja će se sastojati od bilježenja rezultata svakog bacanja, pri čemu imamo dvije mogućnosti: 1) rezultate bacanja ćemo prebaciti u program za statistiku (aplikacija 'Statistics 1Var') i statističku analizu obaviti izvan našeg programa, ili 2) statistiku ćemo obraditi lokalno, unutar programa. Svaki pristup ima svoje prednosti i nedostatke. Programe za bacanje kocke sa statistikom načinit ćemo na temelju programa 'kocke2'.

Modificirati ćemo program 'kocke2' tako da definiramo globalnu listu L7 u koju ćemo spremati rezultate bacanja koji će nakon završetka rada programa biti dostupni svim programima na HP Prime. Program ćemo nazvati 'kocke21' i počinjemo tako da kopiramo program 'kocke2' u 'kocke21'. Odemo u katalog programa tipkama , odaberemo program 'kocke2' (bez da ga otvaramo), iz menija ispod ekrana odaberemo 'More' i nakon toga 'Save'.



Pojavljuje se ekran za upis novog imena gdje upisujemo 'kocke21' i biramo 'OK'; sada u katalogu programa imamo novi program 'kocke21'; možemo ga otvoriti i modificirati. U programu ćemo definirati novu varijablu tipa liste, npr., 'LL' i učinit ćemo je globalnom. Za to na samom početku programa dodajemo liniju

```
EXPORT LL;
```

Rezultat svakog bacanja kocke spremit ćemo u listu LL; elemente liste je kasnije jednostavno zbrojiti naredbom $\sum LIST$ pa nam ne treba lokalna varijabla 'sum' te ćemo je izbrisati.

Napomena: Program bi radio i da ne izbrišemo lokalnu varijablu 'sum', ali bi to moglo biti zbunjujuće kada kasnije budemo pregledavali program.

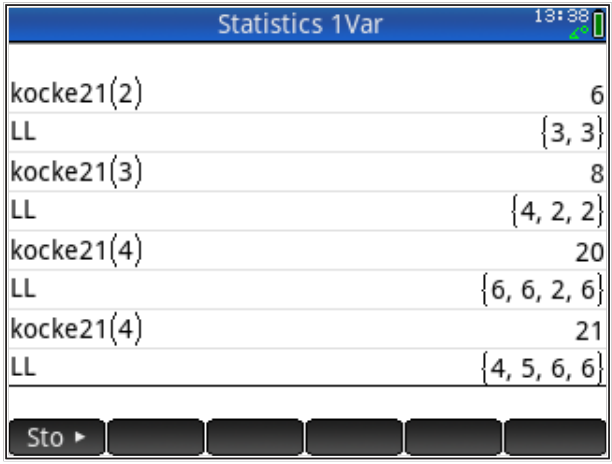
Novi program izgleda ovako

```
EXPORT LL;
EXPORT kocke21(brKc)
// brKc - broj kocaka (proizvoljno)
// n    - broj strana (6)
BEGIN
LOCAL n,i;
brKc:=ABS(IP(brKc));
n:=6;
i:=1;
LL:={};
```

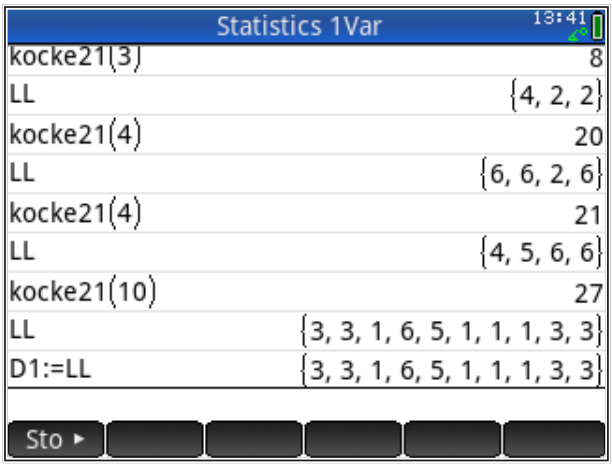
```
WHILE i≤brKc DO
 LL:=CONCAT(LL,RANDINT(1,n));
 i:=i+1;
END;
RETURN SLIST(LL);
END;
```

'SLIST' stoji za naredbu $\sum LIST$. Umjesto inicijalizacije varijable 'sum', inicijaliziramo listu 'LL' u praznu listu 'LL:={}'. Varijabla 'i' ostaje; ona je brojač koji regulira petlju 'WHILE'.

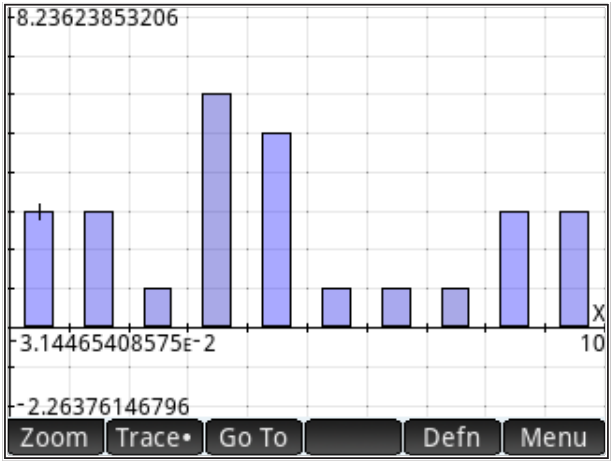
Pri pozivanju programa iz 'Home' ekrana, naizgled se ništa ne mijenja u odnosu na program 'kocke2', međutim, ispišemo li sadržaj liste 'LL' vidimo da su tu zabilježena sva bacanja.



Listu 'LL' možemo kopirati u neku statističku varijablu, npr., D1 i u programu 'Statistics 1Var' načiniti dodatnu analizu i grafički prikazati rezultate bacanja.



Graf dobijemo ulaskom u aplikacije *Apps*, biramo 'Statistics 1Var', tipkamo *Symb*, upisujemo kao varijablu za crtanje 'D1', a za tip crteža 'Plot1' biramo 'Bar'. Grafički prikaz dobijemo pritiskom na tipku *Plot*.



Globalne varijable možemo koristiti i drugačije, tako da ne stvaramo nove varijable, nego da koristimo postojeće. U našem primjeru direktno ćemo pisati u listu 'D1' koja postoji u aplikaciji 'Statistics 1Var'. U postojećem programu 'kocke21' brišemo globalnu varijablu 'LL' i direktno radimo s 'D1'

```
EXPORT kocke21(brKc)
// brKc - broj kocaka (proizvoljno)
```

```
// n    - broj strana (6)
BEGIN
LOCAL n,i;
brKc:=ABS(IP(brKc));
n:=6;
i:=1;
WHILE i≤brKc DO
   Statistics_1Var.D1:=
   CONCAT(Statistics_1Var.D1,RANDINT(1,n));
 i:=i+1;
END;
END;
```

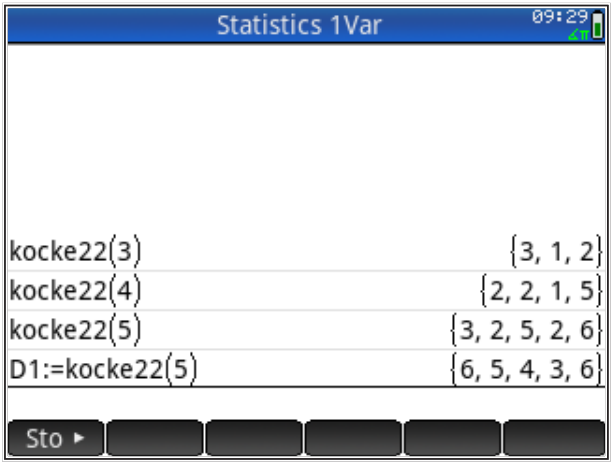
Primijetite da smo D1 pozvali preko imena njene aplikacije 'Statistics_1Var.D1'; konvencija označavanja varijabli preko imena grupe kojoj pripada i dodavanja točke uobičajena je u mnogim programskim jezicima.

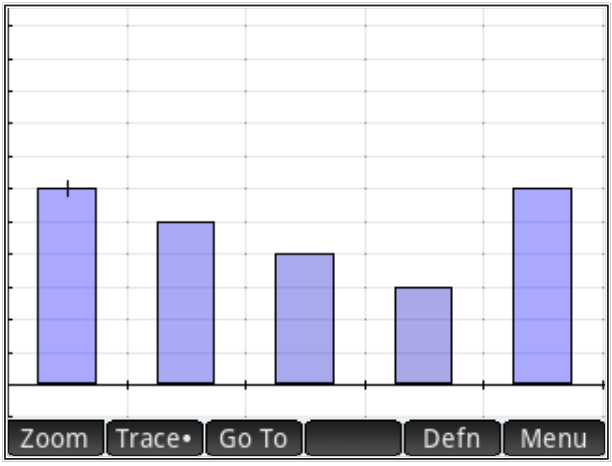
Naravno, statistiku smo mogli odraditi i unutar programa. U tom slučaju možemo raditi samo one analize koje smo predvidjeli i ugradili u program.

Program za lokalnu statistiku bacanja kocke načinit ćemo modifikacijom programa 'kocke2'. Na isti način kao prije načinimo kopiju programa 'kocke2' i nazovemo je 'kocke22'.

```
EXPORT kocke22(brKc)
// brKc - broj kocaka (proizvoljno)
// n    - broj strana (6)
BEGIN
LOCAL n,i,LL;
brKc:=ABS(IP(brKc));
n:=6;
i:=1;
LL:={};
WHILE i≤brKc DO
 LL:=CONCAT(LL,RANDINT(1,n));
 i:=i+1;
END;
RETURN LL;
END;
```

Vidimo da je program vrlo sličan programu 'kocke21', samo 'LL' više nije globalna nego lokalna varijabla. Rezultat programa je lista 'LL'; ona se naravno može spremiti u bilo koju globalnu varijablu, npr., D1, iz koje možemo raditi statistiku i grafički prikaz.





Iako je razlika između programa 'kocke21' i 'kocke22' minimalna što se tiče programiranja, oni predstavljaju dva različita koncepta upravljanja memorijom računala (kalkulatora). Kod većih i složenijih programa ta razlika puno više dolazi do izražaja.

Jednostavni optimizacijski problem

Jednostavni optimizacijski problem predočit ćemo „problemom kradljivca“ (*knapsack problem*), a kao metodu rješavanja upotrijebit ćemo „pohlepni algoritam“ (*greedy algorithm*). Ovaj tekst nije iscrpni ni sveobuhvatni prikaz optimizacijskih problema, niti algoritama za njihovo rješavanje, niti svih svojstava „pohlepnog algoritma“. Cilj teksta je ilustracija mogućnosti primjene HP Prime kalkulatora u rješavanju raznih inženjerskih problema kroz razvoj programa koji omogućuje rješavanje jednostavnih optimizacijskih problema koji se bez računala teško mogu riješiti. Program koji implementira „pohlepni algoritam“ je samo alat, čija primjena onda omogućuje analizu raznih optimizacijskih problema i detaljniju diskusiju svakog od njih.

Knapsack problem (problem kradljivca)

„Problem kradljivca“ formuliran je pred više od 100 godina. To je naziv za optimizaciju odabira n predmeta (aktivnosti) s pozitivnim svojstvom v_i svakog predmeta (aktivnosti) i negativnim svojstvom m_i svakog predmeta (aktivnosti), uz uvjet da je suma negativnih svojstava ograničena na vrijednost M.
Drugim riječima, „kradljivac“ ulazi u sobu u kojoj je n predmeta, svaki vrijednosti v_i i mase m_i. „Kradljivac“ ima vreću kapaciteta M, dakle, ne može nositi više od mase M te treba odlučiti koje predmete odabrati i staviti u vreću tako da mu je plijen (zarada) maksimalan.

Naravno, s ovakvim optimizacijskim zadatkom ne susreću se samo „kradljivci“. On opisuje mnoge praktične probleme: problem smještaja objekata u skladište, problem nalaženja najkraćeg puta između više točaka, problem odabira aktivnosti različitog trajanja, itd.

Matematički, problem možemo formulirati kao odabir vrijednosti iz dvije liste. U prvoj su vrijednosti, u drugoj su težine, a praktično je definirati i treću listu koja označava je li pojedini element iz liste odabran ili nije:

```
Lv:={v1, v2, ..., vn}
Lm:={m1, m2, ..., mn}
Ix:={x1, x2, ..., xn}
```

Ako je $x_i = 0$ onda taj element nije odabran, ako je npr., $x_i = 2$, onda smo odabrali 2 elementa od primjerka na mjestu i iz liste.

Suma težina odabranih predmeta mora biti manja ili jednaka M; vrijednost odabranih predmeta neka je V i mora biti maksimalno moguća.

Razlikujemo nekoliko vrsta *knapsack* problema:

- 0-1 *knapsack* problem: svakog predmeta ima samo po 1 primjerak i ne može ga se dijeliti na manje dijelove; predmet se može samo ostaviti ili uzeti (u „vreću“), odnosno $x_i = 0$ ili $x_i = 1$,
- ograničeni *knapsack* problem (ili *fractional knapsack*): svakog predmeta ima najviše c_i primjeraka, a može ih se uzeti bilo koji broj x_i gdje je $0 \leq x_i \leq c_i$,
- neograničeni *knapsack* problem: broj primjeraka svakog predmeta nije ograničen, a može ih se uzeti bilo koji broj.

Napomena: Neki od navedenih optimizacijskih problema nemaju rješenje u obliku predefiniranog algoritma koji uvijek daje optimalno rješenje.

Greedy (pohlepni) algoritam

„Pohlepni algoritam“ bazira se na principu traženja lokalnog optimuma, u nadi da će tako nađeno rješenje biti i globalni optimum. Algoritam se ne vraća na ispravak prethodnih koraka, nego samo nastavlja dalje do iscrpljivanja podataka. Posljedica je da algoritam uvijek nađe neko rješenje, ali nema garancije da je pronađeno rješenje ujedno i optimalno.

Programiranje algoritma je relativno jednostavno, ali odabir strategije nije jednostavan i ovisi o problemu koji rješavamo.

Primjer

Neka je zadano 5 predmeta čije su mase {1,12,2,1,4} kg i vrijednosti {2,4,2,1,10} tisuća kuna. „Vreća“ za transport neka je kapaciteta M = 15 kg.

```
M:=15
Lv:={2,4,2,1,10}
Lm:={1,12,2,1,4}
```

Ix = {x1,x2,x3,x4,x5} (trebamo odrediti)

Vidimo da je ovo „0-1 knapsack problem“, matematička formulacija problema je

max $\sum v_i x_i$ (izabrani predmeti imaju maksimalnu vrijednost) uz uvjet $\sum m_i x_i \leq M$; $0 \leq x_i \leq 1$ (masa izabranih predmeta manja je ili jednaka dopuštenoj, onoj koja stane u „vreću“). Još jednom, može poprimiti samo vrijednosti 0 ili 1, tj. objekt mase je izabran ($x_i = 1$) ili nije izabran ($x_i = 0$).

Strategija postupka rješavanja nije jasno definirana. Ovo je problem gdje „pohlepni algoritam“ ne garantira optimalno rješenje. Probat ćemo nekoliko strategija, npr., odabir predmeta prema maksimalnoj vrijednosti ili prema minimalnoj težini, itd. Nakon primjene raznih strategija, odlučit ćemo se za onu koja daje bolji (prihvatljiviji) rezultat.

Izbor po maksimalnoj vrijednosti; iz liste vrijednosti predmeta 'Lv' biramo predmete po maksimalnoj vrijednosti i punimo vreću dok ne dosegnemo najveću dopuštenu zadanu masu M.

Postupak je sljedeći:

1. kreiraj listu 'Lw', u koju će se spremati vrijednosti koje su u vreći, u obliku cijelih brojeva koji odgovaraju poziciji u listi vrijednosti 'Lv' predmeta koje smo odabrali i spremili u „vreću2“
2. kreiraj varijablu 'Mu' za ukupnu masu predmeta koji su stavljeni u „vreću“
3. PETLJA:
    3.1. izaberi najveći element iz liste vrijednosti 'Lv' (koristimo funkciju 'MAX'), spremi u varijablu 'vi'
    3.2. nađi na kojoj poziciji u listi je nađena maksimalna vrijednosti, (koristimo funkciju 'POS'), spremi u varijablu 'P'
    3.3. uzmi masu iz liste masa 'Lm' koja odgovara poziciji 'i' na kojoj je maksimalna vrijednost u listi 'Lv' (koristimo funkciju Lm(i)), spremi u varijablu 'mi'
    3.4. umanji liste 'Lv' i 'Lm' za predmet koji je odabran (koristimo naredbu 'DIFFERENCE'), umanjene liste su spremljene u iste varijable, 'Lv' i 'Lm'
    3.5. dodaj masu predmeta 'mi' u ukupnu masu svih predmeta u „vreći“ 'Mu', Mu=Mu+mi
    3.6. provjeri je li ukupna masa 'Mu' manja od dopuštene 'M'
    3.7. ako je masa manja od dopuštene, dodaj odabrani predmet u „vreću“ (stavljanje u „vreću“ se obavlja tako da se vrijednost 'vi' doda u listu vrijednosti odabranih predmeta 'Lw' i masa 'mi' doda u listu mase odabranih predmeta 'Lg')

3.8. provjeri jesi li došao do kraja liste (brojač elemenata u listi 'K' je na nuli),
3.9. ako nisi na kraju liste, ponovi 'PETLJU'

4. ispis rezultata (liste vrijednosti 'Lw' i masa 'Lg' odabranih predmeta).

Program ćemo nazvati npr., 'vrećaV', a liste 'Lv' i 'Lm' će biti lokalne varijable koje će se definirati prilikom pozivanja programa, kao parametri u zagradi.

Algoritam u HP PPL jest

```
EXPORT vrecaV(Lv,Lm)
BEGIN
LOCAL Lw,Lg,Mu,vi,mi,K,P;
Lw:={};
Lg:={};
Mu:=0;
K:=SIZE(Lv);
REPEAT
  vi:=MAX(Lv);
  P:=POS(Lv,vi);
  mi:=Lm(P);
  Lv:=DIFFERENCE(Lv,vi);
  Lm:=DIFFERENCE(Lm,mi);
  IF (Mu+mi)≤M THEN
   Lw(K):=vi;
   Lg(K):=mi;
   Mu:=Mu+mi
    END;
    K:=K-1;
UNTIL K=0;
RETURN {Lw,Lg};
END;
```

Listu odabranih predmeta 'Ix' ne možemo direktno dobiti u programu zbog upotrebe naredbe 'DIFFERENCE' koja iz liste izbacuje SVE elemente koji su isti u dvije liste. Istovremeno, naredba 'POS' daje samo položaj prvog elementa u listi, a koji odgovara traženoj usporedbi. Posljedica je da ne znamo kako će se lista skraćivati nakon naredbe DIFFERENCE pa naredba POS pokazuje na element unutar nove liste čija je duljina nepoznata.

Ukratko, položaje u listi koje dobivamo iz opetovane upotrebe naredbe POS ne možemo povezati s položajima elemenata u orginalnoj (ulaznoj) listi! To je uzrok još jednog ograničenja našeg programa: unutar liste 2 elementa ne smiju imati istu vrijednost. Taj nedostatak otklanjamo relativno jednostavno: (vrlo) malo promijenimo vrijednosti koje su jednake.

Brojčani primjer: Zadano je 5 predmeta, „vreća“ za transport neka je kapaciteta M=15 kg, Tipkamo

```
M:=15
Lv:={2,4,2,1,10} *1000 kuna
Lm:={1,12,2,1,4} kg
```

Odabir predmeta je nepoznat i trebamo ga odrediti.

Možemo se uvjeriti da naš program 'vrecaV' ne radi dobro za ovaj primjer; problem je naredba 'DIFFERENCE'. Međutim, ako vrlo malo promijenimo vrijednosti u listama tako da nemamo ponavljanje istih veličina, program radi dobro. Definiramo korigirane ulazne liste

```
Lv:={2,4,2.01,1,10}
Lm:={1.01,12,2,1,4}
```

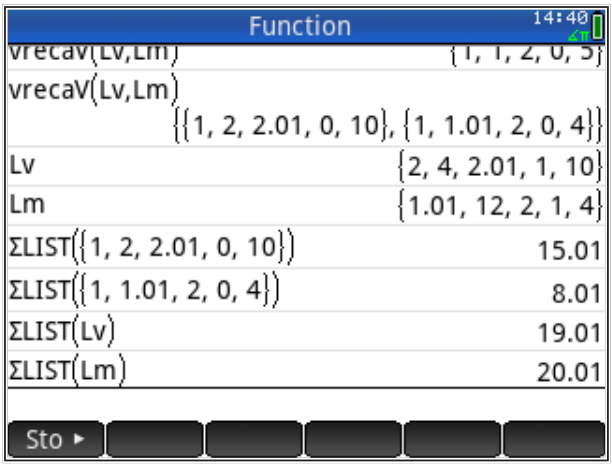
Za kontrolu računamo ukupnu vrijednost i ukupnu masu svih predmeta
Vrijednost svih = $\sum LIST(Lv) = 19.01 \;\; 1000$ kuna,
Masa svih = $\sum LIST(Lm) = 20.01$ kg.

Pozivamo program

```
vrecaV(Lv,Lm)
```

i kao rezultat dobivamo liste vrijednosti i masu odabranih predmeta

```
{{1,2,2.01,0,10},{1,1.01,2,0,4}}
```



Vidimo da je ukupna vrijednost odabranih predmeta 15.01 *1000 kuna i ukupna masa 8.01 kg.

Izračun liste indeksa odabranih predmeta 'Ix'.

Ispis rezultata kao liste vrijednosti i liste masa odabranih predmeta nije uvijek praktičan; željeli bismo listu indeksa odabranih predmeta 'Ix'. Izračun ćemo provesti iz listi 'Lw' i 'Lg' umetanjem naredbi u program na mjesto gdje je sada 'RETURN' naredba. 'Lw' i 'Lg' su na tome mjestu poznati pa ćemo iskoristiti naredbu 'POS' i usporediti odabrane vrijednosti iz liste 'Lw' i predmete iz izvorne liste 'Lv'. Iste rezultate trebali bismo dobiti usporedbom listi masa predmeta 'Lg' i 'Lm' (sjetite se da smo eliminirali duple elemente u obje liste).

Promijenjeni program nazvat ćemo 'vrecaVI' i dobit ćemo ga kopiranjem programa 'vrecaV'.

Dio koji umećemo na mjesto naredbe 'RETURN' u starom programu 'vrecaV' izgleda ovako:

```
Ix:={};
FOR P FROM 1 TO N DO
Ix:=CONCAT(Ix,0)
END;
K:=N;
REPEAT
P:=POS(Nv,Lw(K));
Ix(K):=P;
```

```
K:=K-1;
UNTIL K=0;
RETURN REVERSE(Ix);
```

Napomena: Ukoliko prilikom izvršenja programa dobijemo poruku o grešci, 2 puta kliknemo tipku 'Home' i dobit ćemo poruku u kojem je retku greška.
Na početku smo inicijalizirali listu indeksa 'Ix' tako da ima onoliko nula kolika je duljina liste 'Lv'. U našem primjeru to za 'Ix' daje Ix={0,0,0,0,0}.
Nakon toga, uspoređujemo vrijednosti odabranih predmeta iz liste 'Lw' s orginalnom listom 'Lv' (koja je kopirana u listu 'Nv' jer se 'Lv' mijenja tijekom rada programa). Indekse mjesta gdje se odabrana vrijednost nalazi u orginalnoj listi kopiramo u listu 'Ix'.

Prvi dio programa 'vrecaV' je promijenjen samo koliko treba da bi umetnuti dio radio ispravno: definirane su potrebne lokalne varijable (naredba 'LOCAL' u novom retku) i kopirane su ulazne vrijednosti liste 'Lv' jer se ona tijekom rada programa mijenja.

Kompletni novi program izgleda ovako:

```
EXPORT vrecaVI(Lv,Lm)
BEGIN
LOCAL Lw,Lg,Mu,vi,mi,K,P;
LOCAL N,Ix,Nv;
Lw:={};
Lg:={};
Mu:=0;
Nv:=Lv;
N:=SIZE(Lv);
K:=N;
REPEAT
vi:=MAX(Lv);
P:=POS(Lv,vi);
mi:=Lm(P);
Lv:=DIFFERENCE(Lv,vi);
Lm:=DIFFERENCE(Lm,mi);
IF (Mu+mi)≤M THEN
   Lw(K):=vi;
   Lg(K):=mi;
   Mu:=Mu+mi
```

```
END;
K:=K-1;
UNTIL K=0;
Ix:={};
FOR P FROM 1 TO N DO
Ix:=CONCAT(Ix,0)
END;
K:=N;
REPEAT
P:=POS(Nv,Lw(K));
Ix(K):=P;
K:=K-1;
UNTIL K=0;
RETURN REVERSE(Ix);
END;
```

Listu indeksa 'Ix' ispisujemo obrnutim redom (naredba 'REVERSE') da dobijemo redoslijed biranja predmeta: prvo je 1. odabrani predmet, zatim 2. odabrani predmet, itd.

Brojčani primjer – nastavak. Za prethodno zadanih 5 predmeta dobili smo liste vrijednosti i masa odabranih predmeta

```
{{1,2,2.01,0,10},{1,1.01,2,0,4}}
```

Ako sada s istim parametrima 'Lv' i 'Lm' pokrenemo program 'vrecaVI', kao rezultat za 'Ix' dobijemo

```
{5,0,3,1,4}
```

Prvo je program odabrao 5. predmet iz orginalne liste vrijednosti 'Lv', zatim je preskočio predmet, zatim odabrao 3. predmet, pa 1. predmet i na kraju 4. predmet iz liste 'Lv'.

Rezultat možemo pregledno pisati i ovako

```
L v : = { 2 , 4 , 2 . 0 1 , 1 , 1 0 }
L m : = { 1 . 0 1 , 1 2 , 2 , 1 , 4 }
Ix:={5,0,3,1,4}
```

Zadatak: Modificiraj program 'vrecaVI' tako da se iza liste indeksa 'Ix' kao rezultat pojavljuju još ukupna vrijednost i ukupna masa odabranih predmeta. Tako modificirani program za prethodni primjer trebao bi dati

```
L v := { 2 , 4 , 2 . 0 1 , 1 , 1 0 }
L m := { 1 . 0 1 , 1 2 , 2 , 1 , 4 }
Ix:={{5,0,3,1,4},15.01,8.01}
```

Zadatak: Izračunaj još primjera s raznim ulaznim parametrima i provjeri je li „pohlepni algoritam“ (program 'vrecaVI') našao optimalno rješenje.

Slični problemi. Optimizacijski problem možemo formulirati tako da optimiziramo izbor aktivnosti s obzirom na njihovo trajanje, a varijabla 'M' je tada vremensko ograničenje u kojem te aktivnosti treba obaviti. Listu vrijednosti ćemo prilagoditi problemu: želimo li obaviti što više raznih aktivnosti u zadanom vremenu gdje su nam sve aktivnosti jednako vrijedne ili neke aktivnosti preferiramo, itd. Npr., ako želimo rasporediti veliki građevinski stroj da obavlja poslove na raznim gradilištima, tada je lista vremena trajanje posla, a lista vrijednosti zarada na pojedinom gradilištu.

Izbor po minimalnoj vrijednosti. Iz liste vrijednosti predmeta 'Lv' biramo predmete po minimalnoj vrijednosti i punimo vreću dok ne dosegnemo najveću dopuštenu zadanu masu M. Takvu strategiju bismo primijenili na problemu 'minimiziranja štete (troškova)'. Nije nužno mijenjati postojeći program 'vrecaVI' koji bira maksimalne vrijednosti iz liste 'Lv'; dovoljno je u listu staviti recipročnu vrijednost po kojoj želimo minimum (dakle, 1/trošak). Tada će najveća vrijednost predstavljati najmanji trošak. Na taj način, postojeći program možemo primijeniti i na probleme minimizacije.

HP Prime - Jednostavno programiranje: grafika i crtanje

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime User Guide* (iz menija 'Help') :

- *Programming in HP PPL: Drawing*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

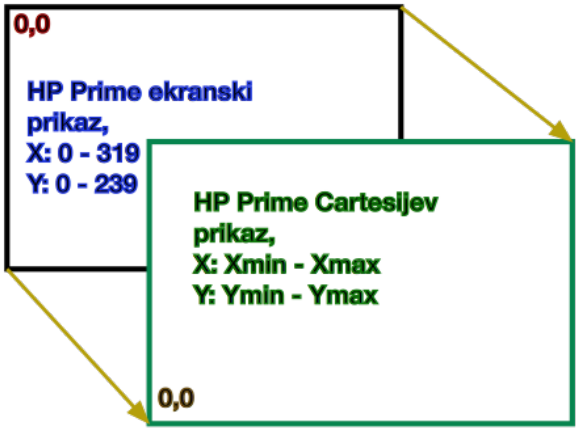
```
nema pretpostavki
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite s 'Edit: Paste' naredbama iz menija kalkulatora, pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

Koordinatni sustav na HP Prime

Grafički prostor koji nam pruža ekran HP Prime kalkulatora sastoji se od 320 točkica po širini ekrana i 240 točkica po visini (točkica na ekranu zove se „piksel“). Početna koordinata (0,0) je u gornjem lijevom kutu, a krajnja (319,239) u donjem desnom. Ukoliko želimo na dnu ekrana ostaviti mjesta za dodatni meni, moramo smanjiti raspoloživu visinu ekrana sa 240 na 220 piksela.
Taj koordinatni sustav zovemo „ekranski“ i većina naredbi za rad s grafikom radi u njegovim koordinatama. Često je praktičnije raditi s proizvoljno zadanim koordinatnim sustavom, koji je definiran varijablama 'Xmin', 'Xmax', 'Ymin', 'Ymax'. Ugrađene aplikacije koje rade s grafikom omogućuju definiranje tih

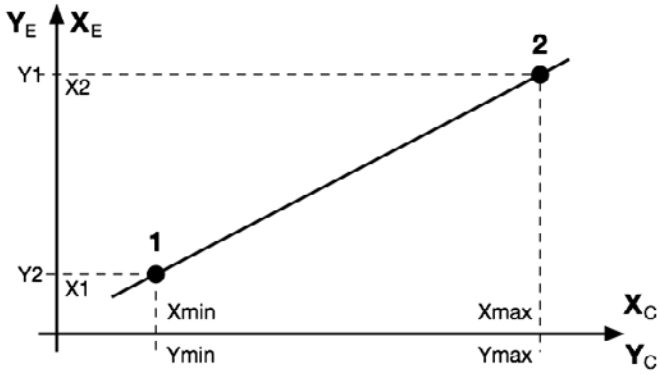
varijabli i automatsko prilagođavanje većine grafičkih naredbi tako da rade u proizvoljno definiranom Cartesijevom koordinatnom sustavu.



Ako u našem programu pozivamo neku od ugrađenih grafičkih aplikacija, možemo koristiti Cartesijev koordinatni sustav definiran varijablama 'Xmin', 'Xmax', 'Ymin', 'Ymax'.

U slučaju da sami želimo načiniti program koji crta grafiku u proizvoljnom koordinatnom sustavu, trebamo načiniti funkcije koje Cartesijeve koordinate preslikavaju u ekranske koordinate.

Preslikavanje koordinata je proces linearne transformacije, koji je najjednostavnije opisati preko jednadžbe pravca kroz dvije točke u koordinatnom sistemu: apscisa - Cartesijev koordinatni sustav (označen indeksom 'C'), ordinata - ekranski koordinatni sustav (označen indeksom 'E').



Posebno su oznake za preslikavanje 'X' koordinata, a posebno za 'Y'. Razvidno je da jednadžba pravca kroz dvije točke

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

daje funkciju izračuna Cartesijevih u ekranske koordinate (ili obrnuto). Funkciju preslikavanja treba načiniti posebno za X os pri čemu je $x = x_C$ i $y = x_E$, i posebno za Y os pri čemu je $x \equiv y_C$ i $y \equiv y_E$.

Za 'X' os ($x \equiv x_C, y \equiv x_E$) usvajamo

$$x_1 = Xmin, x_2 = Xmax,$$
$$y_1 = X1 = 0, y_2 = X2 = 319,$$

Jednadžba pravca kroz dvije točke tada daje funkciju izračuna ekranske 'X' koordinate

$$x_E = \frac{X2}{Xmax - Xmin}(x_C - Xmin)$$

gdje je x_E - ekranska koordinata, x_C - Cartesijeva koordinata.

Za 'Y' os ($x \equiv y_C, y \equiv y_E$) usvajamo

$$x_1 = Ymin, x_2 = Ymax,$$
$$y_1 = Y2 = 239, y_2 = Y1 = 0,$$

Jednadžba pravca kroz dvije točke tada daje funkciju izračuna ekranske 'Y' koordinate

$$y_E = \frac{Y2}{Ymin - Ymax}(y_C - Ymax)$$

gdje je y_E - ekranska koordinata, y_C - Cartesijeva koordinata.

Funkcije koje smo dobili možemo programirati u vlastitoj aplikaciji i tada na ekranu možemo crtati u proizvoljno zadanim koordinatama.

Boje

Ekran može predstaviti 32 tisuće boja po RGB sistemu; naredba RGB(red,green,blue) gdje su 'red', 'green', 'blue' varijable koje označavaju udio pojedine boje u boji koju definiramo i mogu poprimiti vrijednost između 0 i 255. Tako je RGB(0,0,0) potpuno bijela boja, a RGB(255,255,255) crna. Dodatna varijabla 'alpha' u RGB(r,g,b,alpha) određuje prozirnost (alpha=0 definira boju kao prozirnu, alpha=255 kao neprozirnu, što se podrazumijeva ako 'alpha' nije zadano).

HP Prime ima sustav od 10 ekrana označenih varijablama G0, G1, ..., G9. Vidljivi ekran je uvijek G0, a ostali su virtualni.

Naredbe za crtanje

Pod naredbama za crtanje podrazumijevat ćemo naredbe koje mijenjaju izgled aktivnog grafičkog ekrana, a ovdje ćemo spomenuti samo neke, i to u skraćenoj sintaksi (bez svih mogućih opcija). Parametri [u uglatim zagradama] nisu obavezni i mogu se dodati ako korisnik želi. Sve naredbe su prikazane u varijanti za ekranske koordinate (ekstenzija '_P' - piksel).

PIXOFF_P(x,y) - gasi piksel na zadanim koordinatama ekrana,

PIXON_P(x,y,[boja]) - pali piksel na zadanim koordinatama ekrana, u zadanoj boji,

LINE_P(x1,y1,x2,y2,[boja]) - crta liniju zadane boja od točke '1' do točke '2',

RECT_P([x1,y1,x2,y2,boja okvira, boja ispune]) - crta pravokutnik s koordinatom '1' lijevo gore i '2' desno dole, RECT() briše ekran,

ARC_P(x,y,r,[a1,a2,c]) - crta kružnicu radijusa 'r' u točki (x,y), ako se dodaju parametri 'a1' i 'a2', crta kružni luk koji počinje s kutom 'a1' i završava s kutom 'a2' (u smjeru suprotno od kazaljke na satu), 'c' je boja,

FILLPOLY_P({(x1,y1),(x2,y2),...,(xm,yn)},boja,alpha) - crta poligon definiran točkama zadanim u obliku liste, ispunjava ga bojom 'boja'; 'alpha' je indeks prozirnosti, od 0 (potpuno prozirno) do 255 (neprozirno), prva točka (x1,y1) i zadnja (xn,yn) automatski se spajaju (ne treba ponovo na kraju pisati prvu točku),

TEXTOUT_P(„tekst“,x,y) - ispis teksta 'tekst' na grafičkom ekranu, početak teksta je u točki (x,y).

Primjeri crtanja na ekranu

Primjeri ilustriraju grafičke mogućnosti prikaza rezultata na HP Prime kalkulatoru i povezivanje programa s prije definiranim varijablama u kalkulatoru.

Ekranske koordinate su jednostavnije za upotrebu i zadovoljavaju u slučajevima kad grafika treba biti samo ilustrativna, ne i preslika realnosti.

Ovako izgleda program koji crta crvenu dijagonalnu liniju preko cijelog ekrana:

```
EXPORT crtez()
BEGIN
LOCAL boja;
//aktiviraj grafiku
RECT();
//odredi boju
boja:=RGB(250,0,0);
//crtaj liniju preko cijele dijagonale
LINE_P(0,0,319,239,boja);
//ostavi grafiku na ekranu 3 sekunde
WAIT(3);
END;
```

U prethodnom programu naredba WAIT(3) naređuje kalkulatoru da čeka 3 sekunde u stanju u kojem se nalazi, to nam omogućuje da promotrimo grafiku koju smo nacrtali.

Umjesto čekanja, može se formirati MENU u donjem redu grafičkog prikaza kojim se može programirati nastavak rada nakon prikaza grafike. Upute kako se to radi nalaze se u korisničkom priručniku ('User Guide'), uz objašnjenje naredbe DRAWMENU.

Zadatak: Eksperimentiraj s bojom linije.

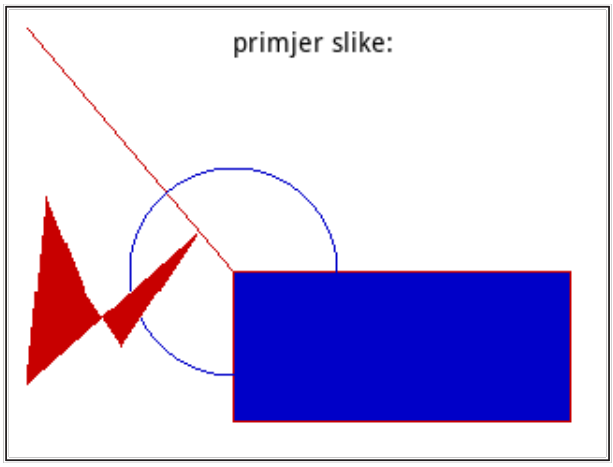
Zadatak: Preinači program tako da kalkulator čeka sve dok ne pritisnemo bilo koje dugme (umjesto naredbe WAIT() koristi naredbu FREEZE ili naredbe GETKEY i REPEAT, uz upute iz korisničkog priručnika).

Program koji kombinira nekoliko grafičkih naredbi, sve koordinate su lokalne (zadane unutar programa)

```
EXPORT crtez1()
BEGIN
```

```
 LOCAL bja1,bja2,t1X,t1Y,t2X,t2Y;
 //aktiviraj grafiku
 RECT();
 //odredi boje
 bja1:=RGB(200,0,0);
 bja2:=RGB(0,0,200);
 //definiraj točke
 t1X:=10;
 t1Y:=10;
 t2X:=120;
 t2Y:=140;
 //crtaj liniju
 LINE_P(t1X,t1Y,t2X,t2Y,bja1);
 //crtaj pravokutnik
 RECT_P(t2X,t2Y,(300),(220),bja1,bja2);
 //crtaj kružnicu
 ARC_P(t2X,t2Y,(t2X-t1X)/2,bja2);
 //crtaj poligon
 FILLPOLY_P({(10,200),(20,100),(40,150),(60,180),(100,120)},bja1,0);
 //stavi naslov na sliku
 TEXTOUT_P("primjer slike:",t2X,t1Y);
 //ostavi grafiku na ekranu
 FREEZE
END;
```

Grafički izlaz je



Poželjno je da je poligon u naredbi 'FILLPOLY' konveksan i da nema križanja linija (a ne kao u primjeru).

Cartesijeve koordinate ili stvarne koordinate, dozvoljavaju realistični grafički prikaz, npr., kada rezultate nekih mjerenja želimo prikazati na ekranu.

Najjednostavniji način da dobijemo prikaz u Cartesijevim (realnim) koordinatora je da crtamo preko aplikacije 'Geometry' (*geometrija*). Tamo je koordinatni sustav Cartesijev, definiran internim varijablama 'Xmin,Xmax' i 'Ymin,Ymax' (preračunavanja iz realnih u ekranske koordinate su automatska).

Ukoliko ne koristimo neku grafičku aplikaciju ugrađenu u HP Prime, koordinate crteža se trebaju preračunavati ('mapirati') iz područja realnih brojeva u cijele brojeve u rasponu broja piksela ekrana. To preračunavanje je jednostavna transformacija, kao što smo prije pokazali i možemo je računati unutar našeg programa.

Primjer poligona u Cartesijevim koordinatama ilustrirat će crtanje realno zadanih objekata. Možemo zamisliti da crtamo mrežu nekakvih građevina koje smo snimili na terenu.
Koordinatni sustav definirat ćemo globalno, preko varijabli 'Xmin', 'Xmax', 'Ymin', 'Ymax' koje ćemo definirati u 'Home' načinu (inače te varijable nisu definirane i HP Prime vraća grešku ako ih pokušamo pozvati). Zadavanje granica ekrana preko globalnih varijabli ima više prednosti: omogućuje nam brzu prilagodbu crteža a kraće se poziva nego da smo granice uključili u parametre programa. Koordinate poligona ćemo definirati u dvije liste, u prvoj 'X', a u drugoj 'Y' koordinate točaka:

```
Px:={x1,x2, ..., xn}
Py:={y1,y2, ..., yn}
```

Dodatnu jednostavnost programa ćemo postići tako da funkcije transformacije koordinata definiramo u aplikaciji *Function*; te funkcije možemo pozvati iz 'Home' načina i modificirati koordinate u listama 'Px' i 'Py'. Nove (ekranske) koordinate spremit ćemo u liste 'Lx' i 'Ly' i proslijediti ih u aplikaciju kao parametre za crtanje.
Transformaciju programiramo preko funkcija F1(X) za koordinate 'X' i F2(X) za koordinate 'Y'. Podsjetimo se da funkcije djeluju na sve elemente liste bez da pozivamo jedan po jedan element.

```
F1(X):= 320/(Xmax-Xmin) *(X - Xmin)
F2(X):= 240/(Ymin-Ymax) *(X - Ymax)
```

Funkcije pretpostavljaju da su granice slike 'Xmin', 'Xmax', 'Ymin', 'Ymax' definirane preko 'Home' ekrana.

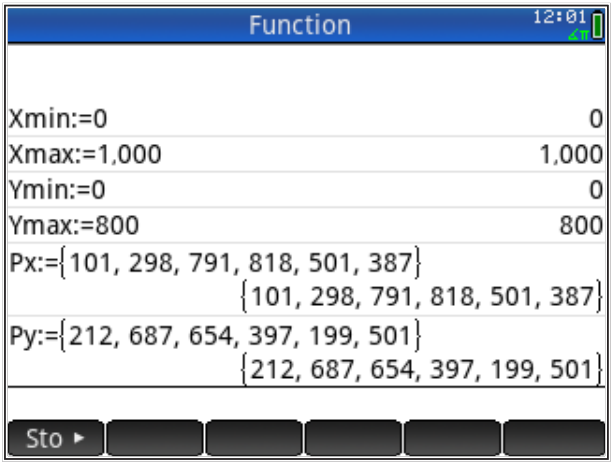
Program ćemo nazvati 'poligon' i načinit ćemo ga modifikacijom programa 'crtez1'. Za crtanje ćemo koristiti naredbu 'LINE' jer želimo nacrtati samo linije (bez ispune).

```
EXPORT poligon(Lx,Ly)
BEGIN
 LOCAL bja1,I,N,t1X,t1Y,t2X,t2Y;
 //aktiviraj grafiku
 RECT();
 //odredi boje
 bja1:=RGB(50,50,50);
 //broj točaka u poligonu
 N:=length(Lx);
 //definiraj prvu točku
 t1X:=Lx(1);
 t1Y:=Ly(1);
 //crtaj poligon
 FOR I FROM 2 TO N DO
   t2X:=Lx(I);
   t2Y:=Ly(I);
   LINE_P(t1X,t1Y,t2X,t2Y,bja1);
   t1X:=t2X;
   t1Y:=t2Y;
 END;
 //stavi naslov na sliku
 TEXTOUT_P("poligon:”,t1X,t1Y);
 //ostavi grafiku na ekranu
 FREEZE;
END;
```

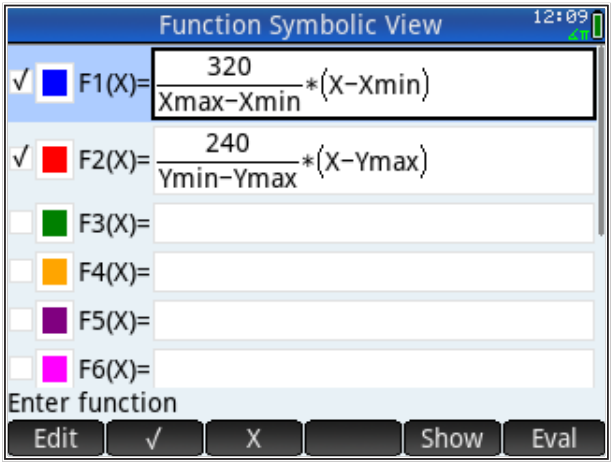
Program 'poligon' je jednostavan jer smo dio posla, transformaciju koordinata iz Cartesijevim u ekranske, obavili izvan programa.
Naravno, program možemo koristiti i za crtanje poligona u ekranskim koordinatama.

Primjer:

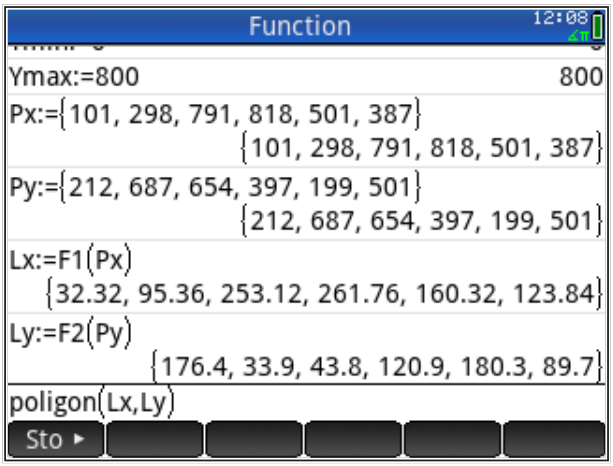
Pretpostavimo da smo na gradilištu približnih dimenzija 1000m sa 800m, trasirali kanalizaciju i imamo koordinate šahti u metrima. Definiramo granice crteža sa (0,1000,0,800) unutar kojih neka je poligon zadan Cartesijevim koordinatama:



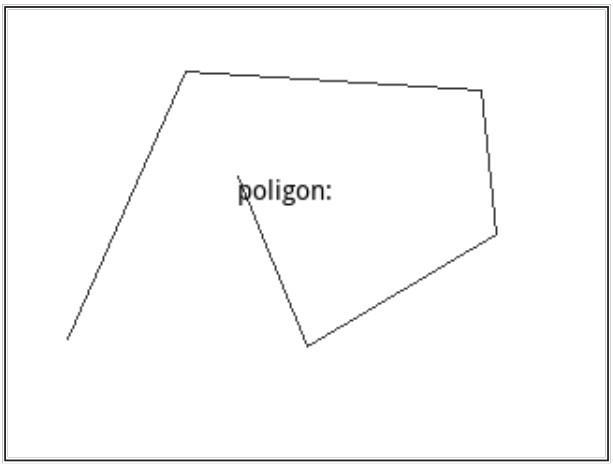
Transformacijske funkcije za HP Prime ekran su



Primijetimo da su ove transformacijske funkcije konstantne za sve točke poligona. Spremanje transformiranih koordinata u liste 'Lx' i 'Ly'



Crtež zadanog poligona je



Modularna verzija

Program 'poligon' može nacrtati pojedini poligon, međutim u praksi često imamo nekoliko poligona koje želimo istovremeno prikazati na slici (na pr., kanalizacija, vodovod, objekti, itd.). Za ponovljeno crtanje istovrsnih grafičkih objekata (poligoni) najpovoljnije je načiniti modularni program: posebno dio za inicijalizaciju crteža i pozivanje crtanja i posebno dio koji crta poligon. Modularni program ćemo načiniti modifikacijom postojećeg programa 'poligon'; taj će program postati upravljački dio, a crtanje ćemo prebaciti u novi program

'poligon2'. Pretpostavimo da ćemo crtati dvija poligona pa ćemo tako predvidjeti u parametrima upravljačkog programa. Program 'poligon' postaje upravljački

```
EXPORT poligon(Lx1,Ly1,Lx2,Ly2)
BEGIN
 LOCAL bja,t1X,t1Y;
 //aktiviraj grafiku
 RECT();
 //odredi boju
 bja:=RGB(50,150,50);
 //- pozovi crtanje 1
 poligon2(Lx1,Ly1,bja);
 //stavi naslov na sliku
 t1X:=Lx1(1);
 t1Y:=Ly1(1);
 TEXTOUT_P("poligon”,t1X,t1Y);
 //- pozovi crtanje 2
 bja:=RGB(150,50,50);
 poligon2(Lx2,Ly2,bja);
 //stavi naslov na sliku
 t1X:=Lx2(1);
 t1Y:=Ly2(1);
 TEXTOUT_P("objekt”,t1X,t1Y);
 //ostavi grafiku na ekranu
 FREEZE;
END;
```

a njegova izvedenica postaje program za crtanje

```
EXPORT poligon2(Lx,Ly,bja)
BEGIN
 LOCAL I,N,t1X,t1Y,t2X,t2Y;
 //broj točaka u poligonu
 N:=length(Lx);
 //definiraj prvu točku
 t1X:=Lx(1);
 t1Y:=Ly(1);
 //crtaj poligon
 FOR I FROM 2 TO N DO
   t2X:=Lx(I);
```

```
   t2Y:=Ly(I);
   LINE_P(t1X,t1Y,t2X,t2Y,bja);
   t1X:=t2X;
   t1Y:=t2Y;
 END;
END;
```

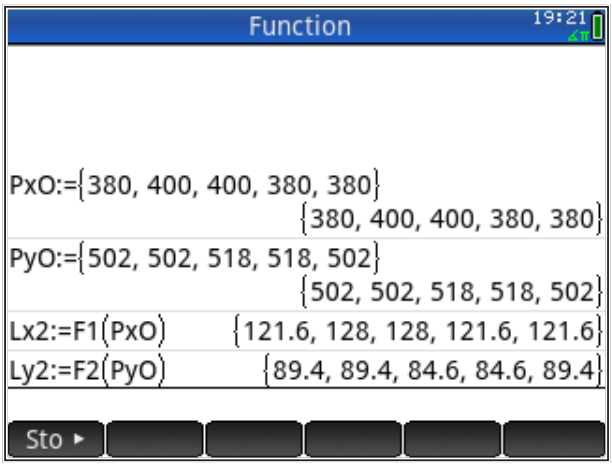
Glavna razlika je da unutar programa 'poligon2' nemamo aktivaciju grafike tako da se poligoni crtaju jedan preko drugog.

Primjer - nastavak:

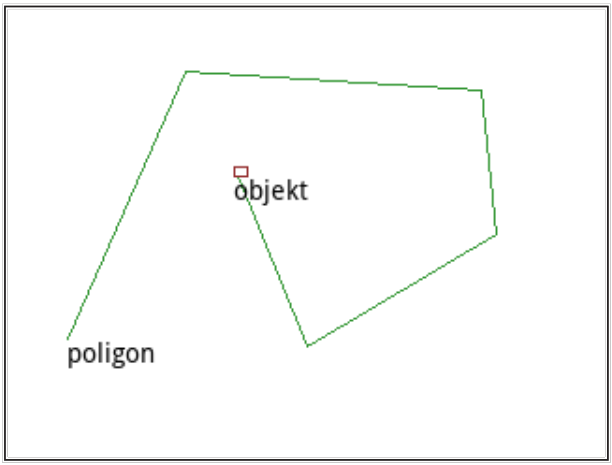
Pretpostavimo da na istom gradilištu kao prije osim kanalizacije imamo i objekt s koordinatama

```
PxO:={380,400,400,380,380}
PyO:={502,502,518,518,508}
```

Koordinate transformiramo i spremimo u liste 'Lx2' i 'Ly2'



Pozovemo program i dobijemo crtež



Po istom principu možemo graditi složene programe koji mogu crtati relativno kompleksne grafike.

“Izvoz” grafike s HP Prime je moguć tako da se iz menija kalkulatora odabere 'Edit: Screen Capture', dobije se rasterska grafika koja se može spremiti u buffer računala ('Copy') ili kao datoteka ('Save As ...').

HP Prime - Jednostavno programiranje - inženjerski primjeri

Sadržaj poglavlja

PREDZNANJE ZA ČITANJE POGLAVLJA

Pročitati iz *HP Prime Quick Start Guide* (iz menija 'Help') :

- *Catalogs and Editors: Program Catalog and Editor*
- *Primary apps: Geometry app*

Pretpostavljeno stanje varijabli u memoriji kalkulatora:

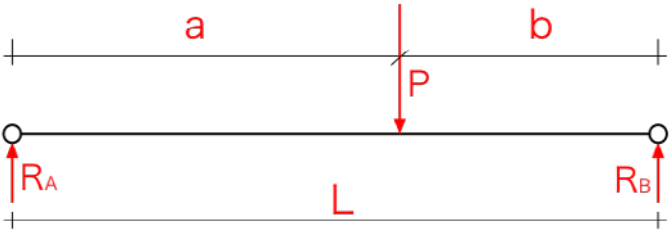
```
nema pretpostavki
```

Ukoliko ove varijable nisu definirane na prikazani način, molim kopirajte ih u kalkulator (označite ih u ovom tekstu, kopirajte sukladno aplikaciji u kojoj čitate ovaj tekst, uđite u HP Prime i zalijepite s 'Edit: Paste' naredbama iz menija kalkulatora, pa pritisnite tipku *ENTER*). Naravno, gornje definicije varijabli možete i jednostavno prepisati u kalkulator.

Primjer: prosta greda

Ovo je pravi inženjerski primjer, nešto složeniji od prethodnih, s više ulaznih i izlaznih parametara. Prije početka rada potrebno je definirati kako će se unositi ulazni parametri i kako će se ispisivati rezultati.

Ulazni parametri neka su geometrijski podaci o gredi i opterećenju, izlazni parametri neka su reakcije grede i moment savijanja. Najjednostavnije je relevantne parametre prikazati grafički



Na slici zorno vidimo značenje pojedinih parametara:

1. P - sila na gredi
2. L - raspon grede
3. a - udaljenost sile od lijevog ruba
4. b - udaljenost sile od desnog ruba ($b = L - a$)
5. R_A - lijeva reakcija grede
6. R_B - desna reakcija grede ($P = R_A + R_B$)

Navedene parametre treba na neki način upisati u program (pritom ne treba zadavati 'b' jer se on može izračunati kao $b = L - a$, a 'b' ćemo tretirati kao lokalnu varijablu).

Veza ulaznih i izlaznih parametara su matematički izrazi

$$R_A = \frac{P \cdot b}{L}$$

$$R_B = \frac{P \cdot a}{L}$$

$$M_{max} = R_A \cdot a = R_B \cdot b = \frac{P \cdot a \cdot b}{L}$$

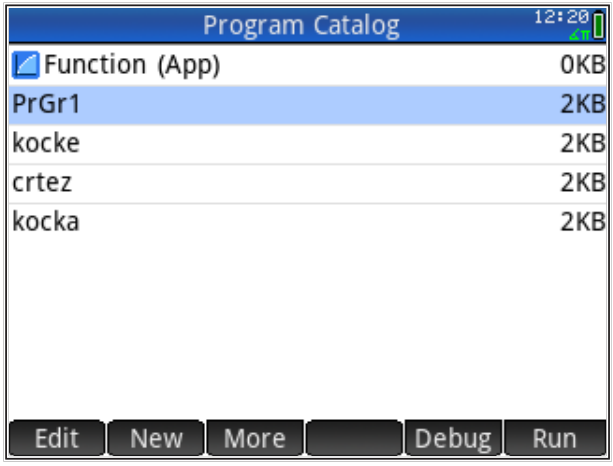
Upis ulaznih podataka

Ulazne podatke ćemo upisati naredbom 'INPUT' redoslijedom L, a, P

```
INPUT({L,a,P},"prosta greda",{"L=","a=","P="},"upiši podatke o gredi");
```

Rezultate ćemo prikazati naredbom 'PRINT' koja ispisuje na terminal. Ukoliko ispisujemo samo jedan rezultat, dovoljno je koristiti naredbu 'RETURN', iza koje slijedi željeni rezultat (u obliku broja ili string varijable).

Prvo ćemo načiniti kratki program gdje ćemo naučiti zadavanje parametara i ispis rezultata. Plan je da taj program kasnije proširimo na potpuni program za računanje reakcija i momenata grede.



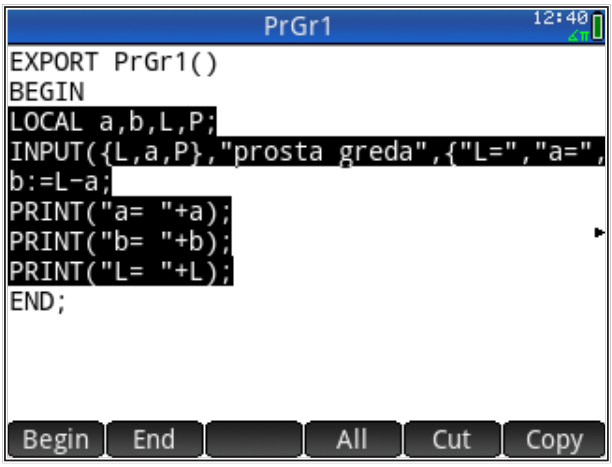
Kompletan program za kontrolu upisa izgleda ovako:

```
EXPORT PrGr1()
BEGIN
LOCAL a,b,L,P;
INPUT({L,a,P},”prosta greda”,{"L=”,”a=”,”P=”},”upiši podatke o gredi”);
b:=L-a;
PRINT("a= "+a);
PRINT("b= "+b);
PRINT("L= "+L);
END;
```

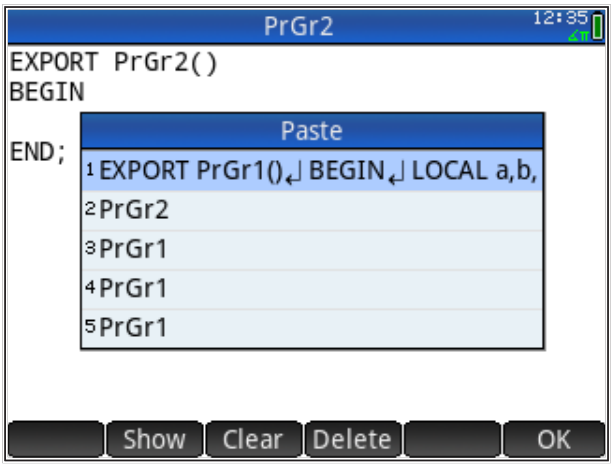
Sada ćemo dodati jednadžbe kojima se iz ulaznih izračunavaju izlazni parametri. Ispis rezultata ćemo s terminala prebaciti na aktivni ekran kalkulatora tako da naredbu za ispis 'PRINT' zamijenimo naredbom 'MSGBOX' koja ispisuje poruku na aktivni ekran.

Novi program ćemo načiniti tako da modificiramo prethodni, ali na način da ga kopiramo u novi program koji ćemo onda mijenjati (želimo zadržati stari program za testiranje upisa parametara).

Postojeći program možemo kopirati u novi iz kataloga programa naredbom 'Save'. Možemo kopirati i samo dio programa. U katalogu programa biramo *Shift* *Copy*, kursor namjestimo na prvu naredbu iza 'BEGIN', iz menija na dnu ekrana biramo 'begin', kursor namjestimo na prvu naredbu ispred 'END;', iz menija na dnu ekrana biramo 'End' i zatim 'Copy'. Izabrani tekst treba izgledati ovako:



Ponovo otvaramo katalog programa i biramo 'New' iz menija na dnu ekrana. Dajemo novo ime programu i biramo dva puta 'OK'. Biramo 'Edit' i namjestimo kursor u prazan red između naredbi 'BEGIN' i 'END', tipkamo *Shift* *Paste* da prenesemo sadržaj programa 'PrGr1' u novi program 'PrGr2'.

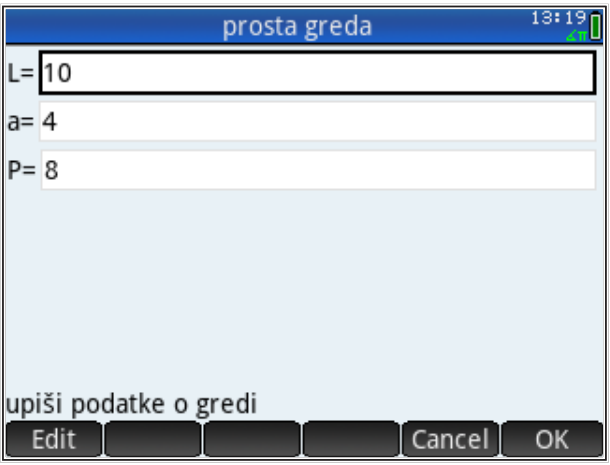


Sada možemo mijenjati program 'PrGr2'. Dodajemo izračun izlaznih parametara i ispis vršimo preko naredbe 'MSGBOX'. Naredba 'MSGBOX' ispisuje samo jednu varijablu tipa 'string' (isto kao i 'PRINT'). Više rezultata možemo u string dodavati jedan za drugim, no program zahtjeva poseban znak kada želimo ispisati rezultate u više redova; potrebno je ubaciti '\n' kao oznaku za novi red. Kompletan listing programa 'PrGr2' sada izgleda ovako:

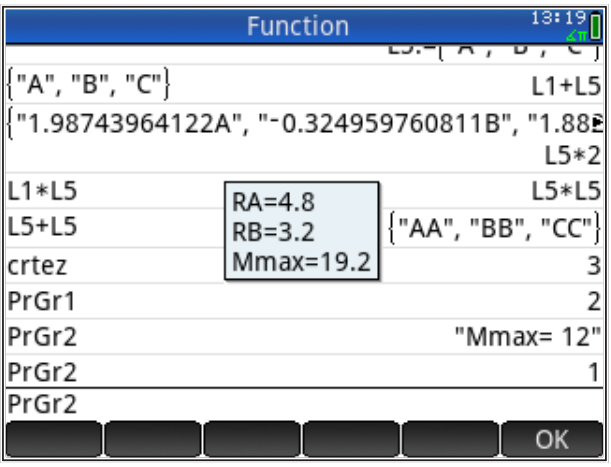
```
EXPORT PrGr2()
BEGIN
LOCAL a,b,L,P;
LOCAL RA,RB,MM;
LOCAL isps;
INPUT({L,a,P},”prosta greda”,{"L=”,”a=”,”P=”},”upiši podatke o
gredi”);
b:=L-a;
RA:=P*b/L;
RB:=P*a/L;
MM:=P*a*b/L;
isps:=”RB=”+RB+”\n”;
MSGBOX("RA=”+RA+”\n”+isps+”Mmax=”+MM);
END;
```

U listingu vidimo korištenje dodatne varijable za olakšavanje ispisa rezultata. Zanimljivo je primijetiti da je najmanji dio programa posvećen računanju vrijednosti parametara, a puno veći dio se odnosi na upis i ispis parametara; to je vrlo čest slučaj s računalnim programima.

Primjer korištenja programa izgleda ovako: prvo upisujemo podatke



Kada je upis završio, biramo 'OK' iz menija na dnu ekrana, program obrađuje ulazne podatke (računa) i pojavljuju se izlazni podaci



Naš program je gotov; obično ćemo ga željeti nadograditi nakon nekog vremena kada primijetimo što bismo još željeli da naš program radi i ispisuje. Najčešće želimo dodati grafički prikaz rezultata, što se prikazuje u slijedećem poglavlju.

Crtanje dijagrama momenata

Crtanje predstavlja dodatni, grafički prikaz izlaznih parametara. S jedne je strane takav prikaz rezultata omiljen jer daje puno informacija („jedna slika vrijedi tisuću riječi“), a s druge strane značajno komplicira pisanje programa.

Crtanju momenata pristupit ćemo modularno: program za crtanje momenata načinit ćemo u posebnom programu koji ćemo povezati s programom za izračun.

Crtež momenta na prostoj gredi od djelovanja sile izgleda kao trokut s vrhom ispod sile. Odredimo karakteristične točke: krajevi grede, točka gdje je sila, točka ispod sile odmaknuta od grede za iznos momenta u nekom odabranom mjerilu. Definiramo varijable:

1. tA - točka A, lijevi rub grede
2. tB - točka B, desni rub grede
3. tC - točka C, na gredi na mjestu djelovanja sile P
4. tM - točka M, ispod sile, odmaknuta za iznos momenta

Te varijable ćemo prenijeti u naš program za crtanje momenta.

Crtat ćemo u realnim koordinatama ekrana da ne bismo pretvarali realne varijable (koordinate) u piksele. To možemo ostvariti tako da aktiviramo grafički ekran ('Plot') neke od aplikacija; odabrat ćemo aplikaciju 'Geometry' (ili njenu kopiju, vidi *Korisnički priručnik*).

Listing programa za crtanje grede u grafičkom izgledu aplikacije *Geometrija* izgleda ovako:

```
EXPORT crtMom(L,a,M)
BEGIN
LOCAL bja;
//aktiviraj aplikaciju
STARTAPP("Geometry”);
//aktiviraj grafički pogled
STARTVIEW(1);
//definiraj boju
bja:=RGB(0,255,0);
//nacrtaj liniju
LINE(0.,0.,L,0.,bja);
FREEZE;
END;
```

Nakon crtanja grafike uvijek moramo zaustaviti kalkulator da bismo mogli vidjeti grafiku. Ako pritisnemo bilo koju tipku, crtež će nestati. Ono što možemo jest prilagoditi položaj i skalu ('Zoom') na ekranu tako da slika izgleda što bolje. Nakon toga, ponovo aktiviramo program i gledamo crtež na ekranu; crtež se može kopirati.

Da bismo dobili ljepšu sliku, s grafičkog prikaza možemo maknuti koordinatne osi.

Vrijednost momenta nije u skali s grafikom (jedinica momenta nije duljina nego sila*duljina); za dobivanje skladnije slike vrijednost momenta treba skalirati tako da je crtež proporcionalan.

Listing programa za crtanje grede i momenta savijanja u grafičkom izgledu aplikacije *Geometrija* izgleda ovako:

```
EXPORT crtMom(L,a,tM)
BEGIN
 LOCAL bja,sklM;
 //odredi boju
 bja:=RGB(0,255,0);
 //aktiviraj aplikaciju i pogled
 STARTAPP("Geometry”);
 STARTVIEW(1);
 //crtaj liniju
 LINE(0.,0.,L,0,bja);
 //crtaj moment u drugoj boji
 bja:=RGB(255,0,0);
 //skala momenta
 sklM:=-0.1;
 LINE(0.,0.,a,tM*sklM,bja);
 LINE(a,tM*sklM,L,0.,bja);
 FREEZE;
END;
```

Ukoliko program pozovemo s navedenim parametrima

```
crtMom(3,1,10) ENTER
```

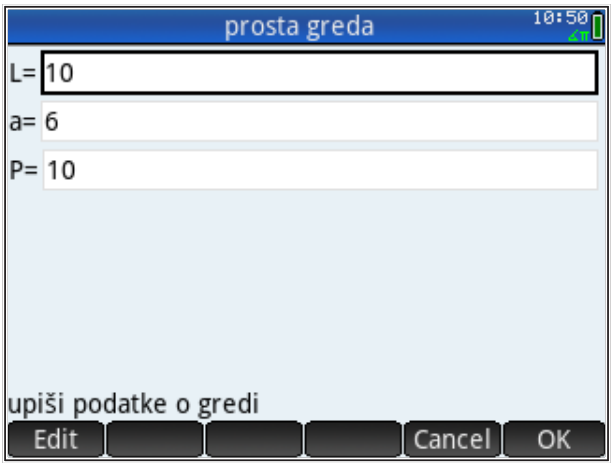
on proizvodi grafiku koja izgleda ovako

Taj ćemo program za crtanje sada povezati s programom za izračun momenata. Prvo kopiramo program 'PrGr2' u program 'PrGrCrt' (lista programa, 'More', 'Save', novo ime, 'OK', 'OK'), a modifikacije radimo u novom programu. Modifikacija je jednostavna; prije kraja programa (iza naredbe 'MSGBOX') pozivamo program za crtanje s odgovarajućim parametrima

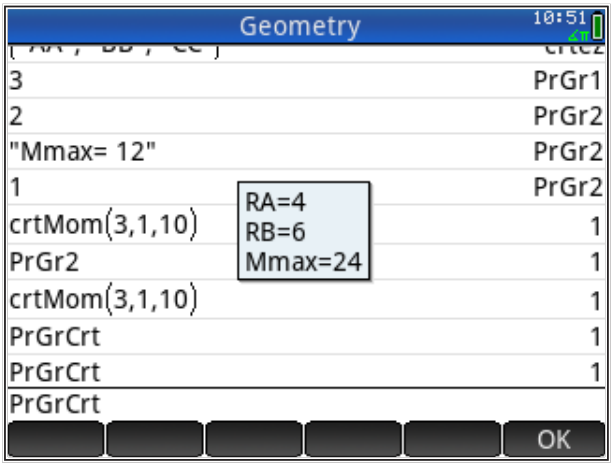
```
crtMom(L,a,MM);
```

Primjer upotrebe programa za gredu duljine 10 m, položaj sile 6 m od lijevog ruba, veličinu sile 10 kN, izgleda ovako:

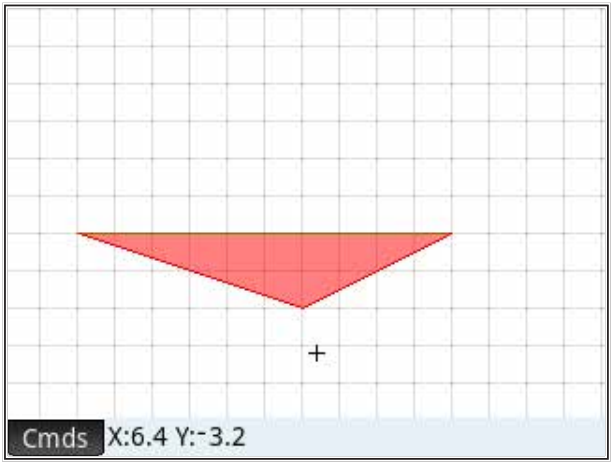
Upis podataka:



Odabiremo 'OK' i dobivamo prikaz rezultata:



Odabiremo 'OK' i dobivamo grafički prikaz rezultata:



Za dobivanje sjenčane poluprozirne slike dodana je naredba 'FILLPOLY()' prije naredbe 'FREEZE'.

Za dobivanje slike koja uvijek optimalno izgleda treba prilagoditi skalu grafičkog prikaza ('Zoom'). S obzirom da grafički rezultat prikazujemo u aplikaciji 'Geometrija', tamo trebamo prilagoditi parametre slike. Aplikacija 'Geometrija' ima nekoliko varijabli kojima regulira veličinu i izgled grafičkog prikaza: 'Xmin',

'Ymin', 'PixSize'. 'Xmin' i 'Ymin' označavaju minimalnu 'X' i 'Y' koordinatu vidljivog ekrana, a 'PixSize' veličinu koraka u 'X' i 'Y' smjeru. Primijetimo da nema parametara 'Xmax' i 'Ymax' koji bi označavali maksimalnu 'X' i 'Y' koordinatu vidljivog ekrana. Te veličine se računaju iz drugih parametara po formuli

```
Xmax = Xmin + PixSize*320
Ymax = Ymin + PixSize*219
```

U programu ćemo zadati 'Xmin' i 'Xmax' i izračunati potrebni 'PixSize' tako da crtež popuni veći dio ekrana.

```
PixSize = (Xmax - Xmin)/320
```

Pritom ćemo 'Xmin' postaviti na -10%, a 'Xmax' na 20% više od duljine grede 'L',

```
Xmin = -L/10
Xmax = L*1.2
PixSize = 1.3*L/320
```

Apscisu crteža (os 'Y') prilagodit ćemo tako da je iznad grede 30% ekrana, pri čemu znamo da je os grede na koordinati 'Y=0'. Veličinu momenta skalirat ćemo tako da ide ispod grede na oko 50% visine ekrana (primijetimo da samo veličina momenta 'tM' određuje visinu ekrana).

Malo računanja će nam pomoći da odredimo parametre crteža po 'Y' osi. Uvjeti koje smo postavili daju

```
Ymin = 1.5*tM*sklM
0.50*(Ymax - Ymin) = tM*sklM
```

Kombiniramo uvjete s prijašnjim izrazom za 'Ymax' (koji je ugrađen u aplikaciju 'Geometrija') i dobivamo

```
2*tM*sklM = PixSize*220
```

pri čemu smo zaokružili broj piksela po 'Y' osi sa 291 na 220. Izraz za faktor skaliranja momenta jest

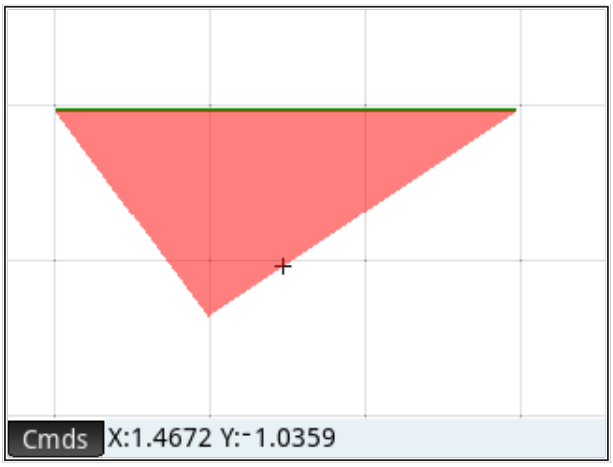
```
sklM = -110*PixSize/tM
```

Kompletan program za crtanje momenta sada je

```
EXPORT crtMomP(L,a,tM)
BEGIN
 LOCAL bja,sklM;
 //odredi boju
 bja:=RGB(0,255,0);
 //aktiviraj aplikaciju i pogled
 STARTAPP("Geometry”);
 STARTVIEW(1);
 //prilagodi okvir slike
 Xmin:=-L/10;
 PixSize:=1.3*L/320;
 sklM:=-110*PixSize/tM;
 Ymin:=1.5*tM*sklM;
 //crtaj liniju
 RECT(0,PixSize,L,0,bja,bja);
 //crtaj moment u drugoj boji
 bja:=RGB(255,0,0);
 FILLPOLY({(0,0),(L,0),(a,tM*sklM)},bja,128);
 FREEZE;
END;
```

Crtanje grede smo sada izveli pravokutnikom (RECT) umjesto linijom (LINE) tako da je greda deblja radi boljeg vizualnog efekta. Crtež momenta izveden je naredbom FILLPOLY uz poluprozirno sjenčanje.

Grafički prikaz iz programa CrtMomP(3,1,10) izgleda ovako



Za bilo koje ulazne podatke, slika dijagrama momenata će biti skalirana tako da zauzima veći centralni dio ekrana.

Prikazani način modifikacije ilustrira kako se programira modularno tako da izrađene programe možemo iskoristiti u što više situacija.

Zadatak: Načini program za izračun i crtanje rezultata izračuna proste grede opterećene koncentriranim momentom.

Zadatak: Načini program za računanje i grafički prikaz proste grede opterećene istovremeno koncentriranom silom i koncentriranim momentom.