Eesti Teadusagentuur
Estonian Research Council
TeaMe
Euroopa Liit
Euroopa Sotsiaalfond
Eesti tuleviku heaks
TTÜ
informaatikainstituut

Muutujad Scratchis

Muutujad mängivad olulist rolli kõikides programmeerimiskeeltes. Võrreldes traditsiooniliste tekstipõhiste programmeerimiskeeltega on Scratchis muutujate kasutamises teatud iseärasused, kuid muutujate olemus ja kasutamise põhimõtted on samad.

Sisu

Muutujate käsitlemine Scratchis

Muutuja olemus, nimi ja skoop

Muutuja on nimega varustatud koht arvuti mälus, kuhu saab salvestada mingi **väärtuse**: arvu, teksti jms. Salvestatud väärtust saab programm kasutada näiteks uue väärtuse leidmiseks, argumendina mingi tegevuse määramisel jm. Tegemist on **mäluväljaga** ehk lühemalt **väljaga** (öeldakse ka mälupesa ehk pesa). **Viitamiseks** väljale (muutujale) kasutatakse programmis **nime**.
Scratchis luuakse muutujad projekti loomise faasis grupis **Muutujad** asuva korraldusega **Tee muutuja**. Korralduse alusel kuvatakse dialoogiboks, milles küsitatakse muutuja **nime** ja **skoopi**.

Muutuja nimele mingid erilisi nõudeid või piiranguid Scratchis ei ole. Kuid on otstarbekas arvestada teatud väljakujunenud reeglitega, mis kehtivad enamikes programmeerimiskeeltes.

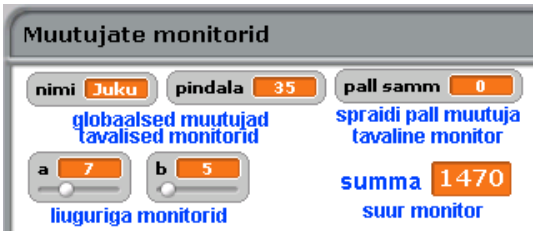
Muutuja **nimi** peab koosnema **ühest tähest või tähtede ja numbrite jadast**, mis peab algama tähega. Nimes ei tohi olla tähtedest ja numbritest erinevaid märke (tehtemärgid, kirjavahemärgid, sulud jmt), välja arvatud allkriipsud. **Nimes ei tohi olla tühikuid**.

Muutuja skoobi saab kasutaja määrata vastava suvandinupu abil. Muutuja skoobi (*scope*) all mõistetakse selle **määramis-piirkonda** ehk kättesaadavuse (nähtavuse) areaali. Scratchis on **globaalsed** ja **lokaalsed** muutujad. Globaalne muutuja on kättesaadav kõikidele spraitidele ja lava skriptidele, lokaalne muutuja on otseselt kättesaadav ainult ühe spraidi jaoks.

Peale esimese muutuja loomist, ilmuvad ka muutujatega seotud **plokid**: muutuja nimega viitamisplokk ning pinuplokid: **võta**, **muuda**, **näita** ja **peida**. Järgmiste muutujate lisamisel ilmuvad ainult muutujate nimedega viitamisplokid.

Muutujate monitorid

Iga muutuja jaoks luuakse automaatselt monitor ehk näidik, mis võimaldab esitada laval muutuja jooksva väärtuse. Monitori saab näidata ja peita, lülitades sisse või välja märkeruudu muutuja nime ees. Monitori saab näidata või peita ka programselt, kasutades skriptides käske **näita muutujat** või **peida muutuja**.



Monitoris kuvatakse muutuja nimi ja jooksev väärtus. Lokaalse muutuja monitoris kuvatakse muutuja nime ees ka spraidi nimi. On kolm monitori esitusviisi ehk tüüpi:

- tavaline ehk normaalne näit, kuvatakse muutuja nimi ja väärtus
- liuguriga näit; **liuguri** ehk kerimisriba abil saab muuta muutuja väärtust. Liuguri jaoks saab vastava käsuga määrata minimaalse ja maksimaalse väärtuse. Oma olemuselt vastab liugur paljude süsteemide kasutajaliidestes esinevale kerimisribale.
- suur näit; muutuja nime ei kuvata. Muutuja identifitseerimiseks võib kasutada nn tekstispraiti.

Vajaliku monitoritüübi saab valida menüüst, mis kuvatakse, kui klõpsata monitoril hiire parema nupuga.

Muutuja, monitori ja liuguri omadused ja meetodid

Muutuja kujutab endast objekti, millele vastab füüsilisel tasemel **mäluväli**. Nii nagu igal objektil, on ka muutujal teatud valik omadusi ja meetodeid. Scratchis on muutuja otse seotud monitoriga, mis kujutab endast muutuja alamobjekti. Viimasel võib olla liugur, mis on omakorda monitori alamobjekt. Tegemist on allpool toodud objektide süsteemiga. Siin, nagu ka teistes jaotistes, tähendab sulgude esinemine meetodi nime järel seda, et vastavat meetodit (tegevust) saab täita ka skriptides. Sulgude puudumine näitab, et antud tegevust saab täita ainult nö „käsitsi“.

Muutuja
nimi, skoop väärtus
loomine eemaldamine väärtuse muutmine() väärtuse lugemine()

Monitor
nimi, väärtus, tüüp, nähtavus, asukoht
näita(), peida() muuda tüüpi muuda asukohta

Liugur
väärtus, min, max
määra min_max muuda väärtust

Monitor võimaldab näha muutuja jooksvat väärtust. Liuguri abil saab seda väärtust ka muuta. Liuguriga monitor on üsna sarnane tekstiboksiga, mis on seotud muutuja ja kerimisribaga. Viimasega saab muuta tekstiboksi väärtust ja selle kaudu ka muutuja väärtust.

Liugurit saab kuvada ja peita nö „käsitsi“ rakenduse loomise ajal. Selle jaoks saab määrata minimaalse ja maksimaalse väärtuse. Liuguri omadus **väärtus**, mis kuvatakse monitoris, on ühtlasi ka muutuja väärtus. Kui skriptis muudetakse muutuja väärtust, muutuvad ka monitori ja liuguri väärtused.

Monitori nimeks on muutuja nimi ning seda eraldi määrata ja muuta ei saa. Monitori tüübi ehk esitusviisi (tavaline, liuguriga või suur) saab määrata monitori objektimenüüst. Monitori nähtavust saab määrata nii käsitsi kui ka programselt. Monitori asukohta saab muuta hiire abil.

Muutuja nimest ja skoobist oli juttu eespool – jaotises „Muutuja olemus, nimi ja skoop“. Muutuja väärtuseks võivad olla arvud, tekstid, tõeväärtused või objektide nimed. **Muutujate tüüpide deklareerimist** (määratlemist), mis on iseloomulik enamikule traditsioonilistele programmeerimiskeeltele, Scratchis **ei toimu**. Erinevatel ajahetkedel võib muutujal olla erinevat tüüpi väärtused: kord arv, teinekord tekst jms. Peab rõhutama, et igal ajahetkel saab muutujal olla ainult üks väärtus. **NB!** Peale programmi töö lõppu, muutuja viimane väärtus säilib!

Muutujate loomine ehk mälu (mäluvälja) eraldamine muutujatele toimub rakenduse loomise faasis nö „käsitsi“ korraldusega **Tee muutuja**. Enamikus traditsioonilistes programmeerimiskeeltes toimub see alati programmi täitmise ajal vastavate deklaratsioonide alusel. Ka muutujate eemaldamine projektist toimub Scratchis „käsitsi“. Muutaja loomisel võetakse selle **algväärtuseks 0** (**null**).

Muutuja väärtuse muutmiseks on Scratchis järgmised võimalused:

- liuguri kasutamine (ainult arvud),
- omistamine käskudega [**võta** *muutuja* **=** *avaldis*] ja [**muuda** *muutuja avaldis* **võrra**]**,**
- lugemine klaviatuurilt käsuga [**küsi** … **ja oota**] + käsk [**võta** *muutuja* **= vastus**].

Muutuja väärtuse lugemine seisneb muutuja väljas salvestatud väärtuse kasutamises näiteks uue väärtuse leidmiseks avaldise abil.

Omistamine ja omistamiskäsk

Muutujale väärtuse omistamine seisneb mingi väärtuse salvestamises antud muutuja väljas. Sageli eelneb sellele väärtuse tuletamine (leidmine) etteantud avaldise alusel või lugemine väliskeskkonnast. Scratchis on omistamise põhivahendiks käsuplokk: 

Siin on **muutuja** esindatud oma nimega; märki „=“ nimetatakse **omistamissümboliks** e **omistamisoperaatoriks** (mitte võrdlusmärgiks); **avaldis** määrab väärtuse leidmise eeskirja. Avaldise erijuhuks on ka konstant ja muutuja. Käsu täitmisel leitakse avaldise väärtus ja saadud tulemus võetakse muutuja väärtuseks, st salvestatakse muutujale eraldatud väljas.

Enamuses tekstipõhistes programmeerimiskeeltes on omistamislausel järgmine kuju:

muutuja = avaldis

Scratchis on ka omistamislause teine variant: 

Muutuja väärtusele lisatakse avaldise väärtus ja saadud tulemus võetakse muutuja uueks väärtuseks.

Mõned omistamiskäskude näited

S = 0 **k = 1** **nimi** = „Juku Naaskel“

muutujatele omistatakse konstantide väärtused, st salvestatakse need vasakus pooles näidatud muutujate väljades

max = a 

loetakse muutuja **a** väärtus ja salvestatakse see muutujas **max** – toimub kopeerimine

D = b * b – 4 * a * c

loetakse muutujate **a**, **b** ja **c** väärtused, leitakse avaldise väärtus ja omistatakse muutujale **D** – salvestatakse selle muutuja väljas

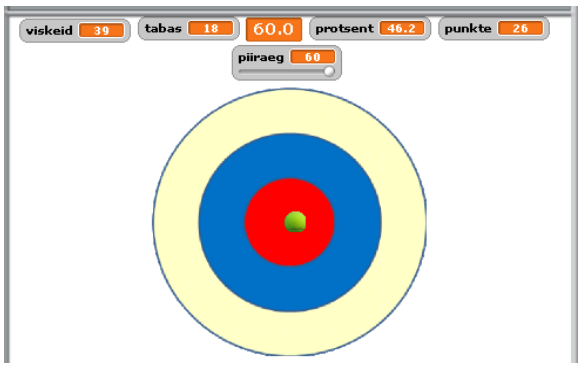
k = k + 1 

loetakse muutuja **k** väärtus, liidetakse sellele **1** ja tulemus salvestatakse tagasi muutujasse **k**

n = n – 5 

loetakse muutuja **n** väärtus, lahutatakse **5**, tulemus salvestatakse tagasi muutujasse **n**

Näide „Täpsusvisked“



Imiteeritakse täpsusviskeid palliga. Lava keskel asub kolmest kontsentrilisest ringist koosnev märklaud. Etteantud aja jooksul (piiraeg) muudetakse teatud sagedusega palli asukohta etteantud piirkonnas juhu-arvude abil. Programm loendab visete arvu, tabamuste arvu, tabamuste protsendi ja punktide arvu.

Tabamuse korral antakse punkte sõltuvalt märklaua koha värvusest, kuhu sattus pall: punane 4 punkti, sinine 2 ja kollane 1 punkt. Demo.

Kasutusel on kaks spraiti: **pall** ja **märklaud**. Kõik skriptid on seotud palliga, märklaud on nö passiivne sprait. Arvestatakse palli sattumist selle sisse ja värve.

Visete juhtimiseks ja „statistika“ pidamiseks on kasutusel järgmised muutujad:

X ja Y – palli sihtkoha koordinaadid**,**

viskeid – visete jooksev ja lõpparv,

tabas – tabamuste jooksev ja lõpparv,

protsent – tabamuse protsent,

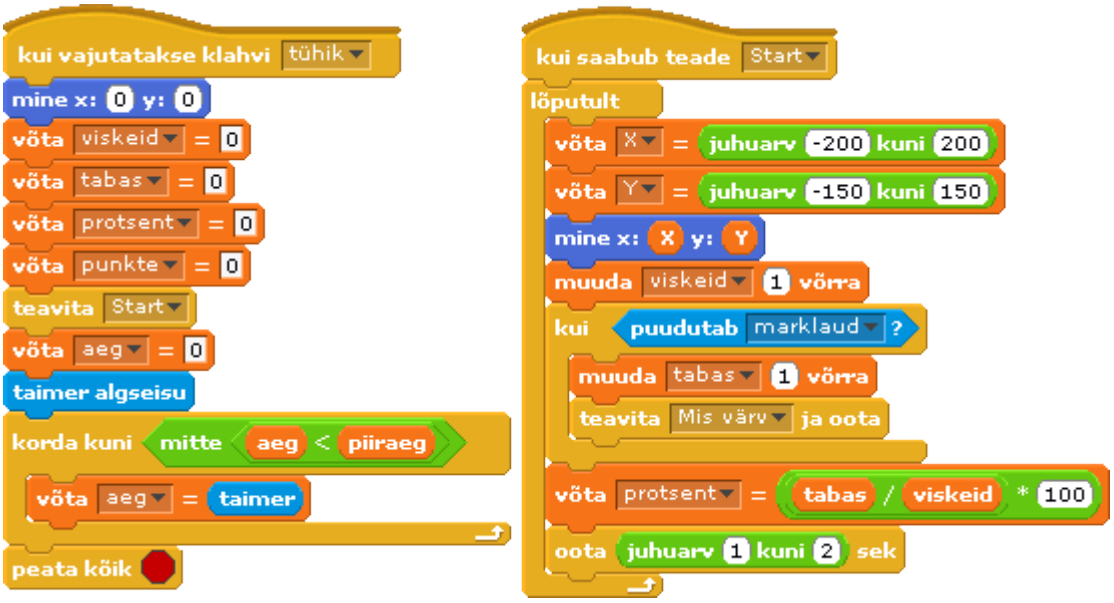
punkte – punktide arv

Aja kuvamiseks ja kontrolliks kasutatakse muutujaid:

aeg – jooksev aeg,

piiraeg – mängu aeg, saab valida kasutaja liuguri abil.

Kõik muutujad on globaalsed. Kuna kõik skriptid kuuluvad ühele spraidile, siis ei ole erilist vahet, kas kasutatakse lokaalseid või globaalseid muutujaid. Viimased on siiski veidi mugavamad monitoride kuvamise seisukohast, sest lokaalsete muutujate monitorides kuvatakse ka spraidi nimi.



Programm käivitub vajutusega klahvile **tühik**. Pall viiakse lava keskele. Muutujatele **viskeid**, **tabas**, **protsent** ja **punkte** omistatakse algväärtuseks 0. Seda peab tegema iga uue mängu alguses, sest muidu jätkatakse eelmise mängu tulemustest. Käsuga [**teavita *Start***] käivitatakse põhiskript ning paralleelselt jätkub antud skripti täitmine tegevustega, mis on seotud aja kontrolliga. Muutuja **aeg** väärtuseks võetakse 0 ja pannakse

taimer algseisu (nulli). Edasi toimib kordus seni, kuni muutuja **aeg** väärtus saab võrdseks või suuremaks piirajast **piiraeg**. Korduse täitmine seisneb praktiliselt aja kuvamises ja kontrollis.

Põhiskript kujutab formaalselt lõputud kordust. See katkestatakse, kui aeg saab täis: esimeses skriptis täidetakse käsk [**peata kõik**].

Kordamisel omistatakse muutujatele **X** ja **Y** uued juhuslikud väärtused, käsuga **mine…** viiakse pall uude kohta ja suurendatakse muutuja **viskeid** väärtust ühe võrra.

Järgnev **kui**-plokk kontrollib, kas pall puudutab märklauda. Kui see on nii, kontrollib alamskript, millist värvi pall puudutab ja suurendab muutujat **punkte** nelja, kahe või ühe võrra. Sõltumata sellest, kas toimus tabamine, arvutatakse **protsent**, tehakse paus 1..2 sekundit ja protsess kordub.

Muutujate kasutamine objektidele viitamiseks

Muutujale võib omistada ka objekti (sprait, heliklipp) nime. Sellisel juhul võib muutujat kasutada viitamiseks erinevatele objektidele. Taolisi muutujaid nimetatakse sageli **viitmuutujateks** ehk **objektimuutujateks**. Objektimuutujat võib kasutada objektile viitamiseks kindla nime (konstandi) asemel plokkides ja käskudes, kus on ette nähtud viit objektile: [**mine** *objekt*], [**osuta** *objekt*], [*omadus* **kujundil** *objekt*], [**mängi** *heli*].

Näide. Laval on **Kraps** ja tema sõbrad - spraidid (objektid) nimedega: **koer**, **part**, **hiir** ja **kass**. Kui klõpsatakse mõnda sõpradest, omistatakse selle nimi muutujale ***NN*** ning väljastatakse teade **Tule**. Selle võtab vastu **Kraps** ja läheb sõbrale külla, tuleb tagasi lava keskele ja ütleb kauguse sõbrani.

Krapsu skript demonstreerib muutuja **NN** kasutamist viitamiseks objektidele (spraitidele).

Käsk [**osuta** NN] pöörab Krapsu spraidi poole, mida klõpsati. Käsk **ütle** kuvab teksti „Oi! *nimi*“, kus *nimi* on spraidi nimi, mida klõpsati. Järgmised kaks käsku omistavad muutujatele X ja Y spraidi koordinaadid, mille nimi on muutujas **NN**. Käsk **liigu** viib Krapsu vastava spraidi juurde.

Käsus **ütle** kasutatakse muutuja **NN** tekstväärtust: kuvatakse tekst „Tere“ ja vastav nimi. Järgmine käsk **liigu** viib Krapsu lava keskkohta. Käsk **võta** leiab ja omistab muutujale **L** vahemaa Krapsu ja selle spraidi vahel, mille nimi on muutujas NN.

Näide „Suur võidusõit“

Näide illustreerib lokaasete ja globaalsete muutujate, omistamise ning erinevat tüüpi monitoride kasutamist. Esineb ka objektimuutuja.

Imiteeritakse autode võidusõitu.

Võistlevad kolm autot. Sõidetavate „ringide“ arvu (muutuja **ringe** väärtuse) saab kasutaja määrata teatud piirides liuguri abil. Programm fikseerib autode sõiduaajad ning teeb kindlaks võitja.

Projektis on neli põhiobjekti: **Kraps** (kohtunik) ja kolm **autot**.

Kasutusel on kaks globaalset muutujat: **ringe** ja **võitja**. Igal autol on kaks lokaalset muutujat: **ring** – jooksva ringi number ja **aeg** – jooksva ringi ja lõppaeg. Lisaks on Krapsul üks lokaalne abimuutuja (**min**).

Ühetüübiliste väärtuste salvestamiseks on lokaalsete muutujate kasutamine üsna kasulik. Tänu sellele on näiteks siin kõikide autode skriptid täiesti ühesugused. Tehes skripti ühe spraidi jaoks ja paljundades seda, saab kohe uue spraidi valmis skriptiga. Globaalsete muutujate korral peaks võtma kuus erineva nimega muutujat näiteks aeg_1, aeg_2, aeg_3, ring_1 jne ning korrigeerima skripte peale kopeerimist.

Pärast algseadete tegemist ja taimeri viimist algseisu, käivitab põhiskript käsuga [**teavita *Start***] autode skriptid. Need juhivad autode ringliiklust, kuvavad iga auto jaoks jooksva ringi aja ja sõidu koguaja.

Kraps. Põhiskript

Käsus **mine** kasutatakse objektile viitamiseks muutujat.

Autod

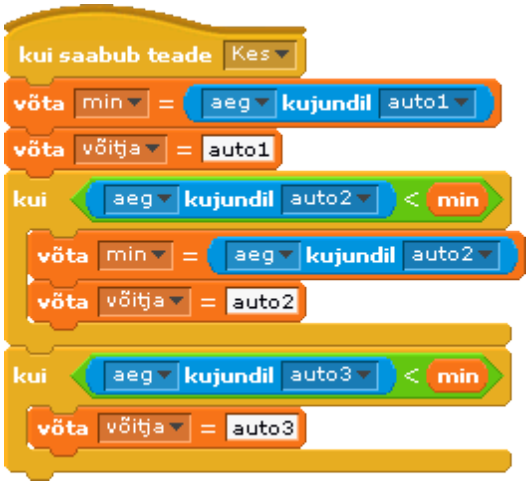
Kõigil kolmel autol on täpselt ühesugused skriptid.

Sõit lava parempoolse servani.

Tagasi vasakusse serva

Jooksva ringi aeg.

Ringi numbri muutmine



Kui kõik autod on sõidu lõpetanud, käivitatakse abiskript **Kes**, mis teeb kindlaks võitja.

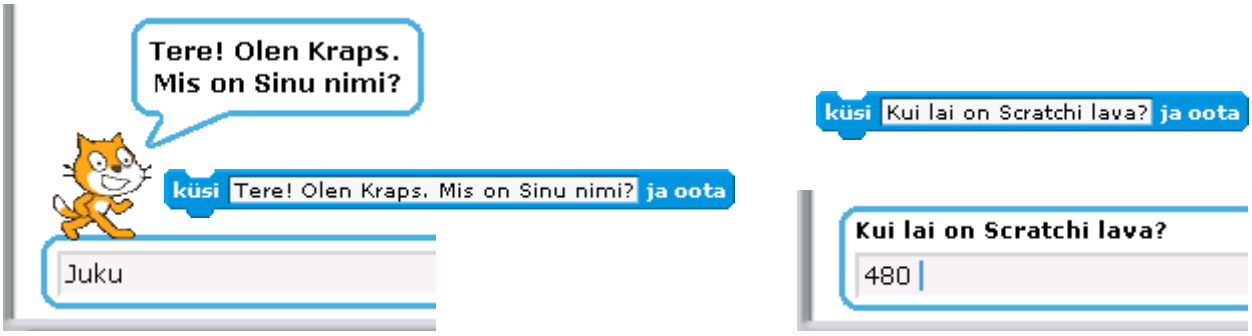
Kõigepealt võetakse muutuja **min** väärtuseks **auto_1** aeg ja muutujale **võitja** omistatakse esimese auto nimi **auto1**. Kui **auto2** aeg on väiksem **min** väärtusest, võetakse see muutuja **min** uueks väärtuseks ning muutujale **võitja** omistatakse nimi **auto2**. Ning lõpuks, kui **auto3** aeg on omakorda väiksem kui **min**, omistatakse muutujale võitja nimi **auto3**.

Muutujat **võitja** kasutakse viitamiseks objektile käsus **mine**, mis viib Krapsu võitnud auto juurde.

Muutujate väärtuste lugemine klaviatuurilt

Väärtuse lugemiseks klaviatuurilt saab kasutada spraidi või lava skriptis **andurigrupi** plokki:

Ploki täitmisel skripti töö peatakse, lava alumisse serva ilmub tekstiväli, kuhu saab tippida vajaliku väärtuse. Ploki sisendväljas olev **teade** (tekst) kuvatakse kõneballoonis vastava spraidi juures (plokk on spraidi skriptis) või tekstivälja ülaosas (plokk on lava skriptis).



Kui kasutaja tipib tekstiväljas väärtuse (tekst või arv) ja vajutab klahvile **Enter** või klõpsab märkeringi tekstivälja paremas otsas, võetakse väärtus ploki vastus vastus väärtuseks. Ploki **vastus** näol on tegemist omamoodi süsteemi sisemuutujaga. Seda võib kasutada avaldistes ja käskudes või selle väärtuse võib omistada muutujale. Järgmise ploki [**küsi** … **ja oota**] täitmisel asendab uus sisend ploki (muutuja) **vastus** eelmise väärtuse.

Näide. „Ideaalne kaal“

Rakendus võimaldab leida inimese „ideaalse“ kaalu ehk massi **mid** (kg) järgmise valemi järgi:

$$m_{id} = \begin{cases} (3 \cdot L - 450 + v) \cdot 0{,}225 + 40{,}5 \text{ naine} \\ (3 \cdot L - 450 + v) \cdot 0{,}250 + 45{,}0 \text{ mees} \end{cases}$$

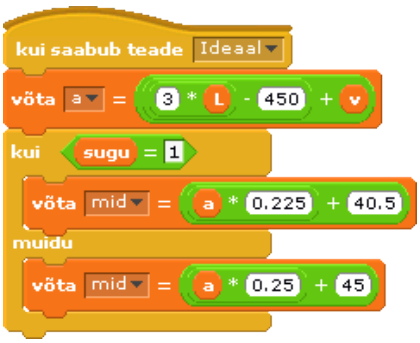
L – pikkus (cm),

v - vanus (aastates)

Esimene käsk [**küsi** …] loeb klaviatuurilt kasutaja nime. Kui kasutaja ei sisesta nime, on tegemist nn tühja väärtusega. Sellisel juhul skripti töö katkestatakse.

Kui kasutaja sisestab nime, omistatakse see muutujale **nimi** ning seda muutujat kasutatakse mitmes käsus.

Järgnevad käsud [**küsi** …] loevad ja omistavad väärtused muutujatele **L**, **v** ja **sugu**.



Skript **Ideaal** leiab ideaalse massi.

Abimuutujale **a** omistatakse avaldiste ühisosa väärtus.

Rakenduses on võimalus sisestada algandmeid ka muutujate monitoride liuguritega.