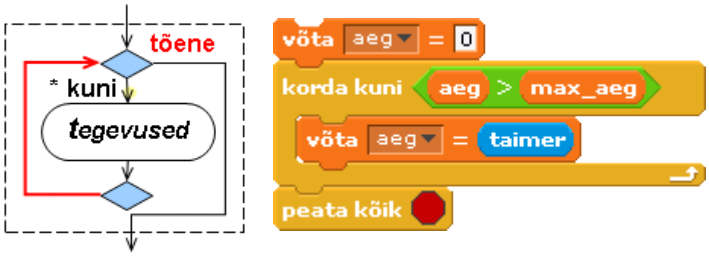
Eesti Teadusagentuur
Estonian Research Council
TeaMe
Euroopa Liit
Euroopa Sotsiaalfond
Eesti tuleviku heaks
TTÜ
informaatikainstituut

Protsesside juhtimine



Programmi skriptides peab määrama vajalikud tegevused ja nende täitmise järjekorra, arvestades ülesande olemust ja Scratchis olevaid vahendeid protsesside juhtimiseks. Selleks on olemas vastavad plokid grupis **Juhtimine**, mille abil saab kirjeldada erinevat liiki kordusi ja valikuid ning esitada muid juhtimisega seotuid korraldusi.

Sisu

Protsesside juhtimine skriptides

Põhimõtteliselt on protsess suvaline tegevuste kogum. Scratchis kujutab protsessi kirjeldus käsuplokkide gruppi, milleks võib olla terve skript või osa sellest. Sõltumata kasutatavatest vahenditest võib ülesannete lahendamisel eristada nelja liiki protsesse:

- järjestikprotsess ehk jada
- tsükliline protsess ehk kordus
- hargnev protsess ehk valik
- paralleelne protsess.

Paralleelseid protsesse saab Scratchis realiseerida ainult nö projekti tasemel: paralleelselt saab täita skripte, skripti sees tegevusi paralleelselt täita ei saa.

Järjestikprotsessid

Järjestikprotsessi korral täidetakse kõik käsud täpselt sellises järjekorras nagu nad on esitatud programmis. Käske ei jäeta vahele ega toimu kordamisi. Mingeid erilisi vahendeid protsessi määratlemisel ei ole vaja, sest täitmine toimub nö loomulikus järjekorras. Illustratsiooniks on toodud kaks näidet.

Esimese skripti korral on tegemist ülesandega, mille lahendamine oma olemuselt kujutab järjestikust protsessi. Programm küsib ja loeb ristküliku külgede pikkused (**a** ja **b**), arvutab selle pindala (**S**) ja ümbermõõdu (**P**) ning leiab pindala ja ümbermõõdu suhte (**suhe**).

NB! käskude järjekorra valimisel peab arvestama, et neid täidetakse täpselt selles järjekorras, nagu need on esitatud programmis. Siin peavad algandmete lugemise käsud kindlasti eelnema pindala ja ümbermõõdu arvutamisele ning viimased omakorda suhte leidmisele. Samal ajal **a** ja **b** lugemise omavaheline järjekord ning **S** ja **P** arvutamise järjekord ei oma tähtsust.

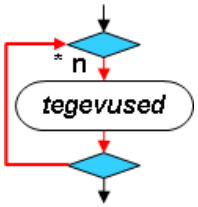
Teises, samuti järjestikku täidetavas skriptis, korduvad kolm korda täpselt kolm ühesugust käsku. Sellisel juhul on otstarbekas kasutada korduste kirjeldamise vahendeid.

Tsüklilised protsessid

Tsükliline protsess seisneb tegevuste (käskude) korduvas täitmises. Programmis tuleb määrata reeglid ja tingimused, kuidas toimub protsessi täitmine. Scratchis on korduste kirjeldamiseks mitu erinevat käsuplokki, mis võimaldavad arvestada erinevat tüüpi kordusega. Vt ka Algoritmimine.

Etteantud korduste arvuga kordus

Selleks on Scratchis plokk [**korda** *n*], kus **n** on kordamiste arv. Viimane võib olla esitatud konstandi, muutuja või avaldise abil. Allpool on toodud ka antud korduse tüübi esitus UMLi diagrammidel ja algoritmikeeles. Samuti on toodud kordusena realiseeritud näide eelmisest jaotisest.



kordus n **korda**

tegevused

....

lõpp kordus

Näide. Karistuslöögid. Fikseeritud arv lööke

Imiteeritakse pealelööke väravale. Palli asukohta muudetakse juhuarvude abil. Laval on sprait nimega **värav**. Löökide (korduste) arv on määratud muutuja **lööke** väärtusega. Väärtust saab muuta liuguri abil.

Objektid: pall, värav, Kraps

Muutujad:

lööke – löökide arv

tabas – tabamuste arv

protsent - tabamisprotsent

X, **Y** – palli koordinaadid

Pall

protseduur Karistus_1

tabas = 0

kordus lööke **korda**

X = juhuarv(x1…x2)

Y = juhuarv(y1…y2)

pall.mine X, Y

kui puudutab värav **siis**

tabas = tabas + 1

teavita Sees

lõpp kui

paus 0,5…1 sek

lõpp kordus

Kraps

protseduur Valmis

protsent = tabas / lööke *100

kui protsent >= 50 **siis**

kuva „Tubli!“

kordus juhuarv(3..5) **korda**

Kraps.y = Kraps.y + hy

paus t sek

Kraps.y = Kraps.y - hy

paus t sek

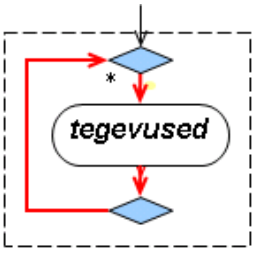
lõpp kordus

lõpp kui

Programm loendab tabamuste arvu. Iga tabamuse korral väljastab **pall** teate **Sees**. Selle võtab vastu Krapsu skript, mis paneb **Krapsu** pöörlema: 36 korda 10° = täisring. Peale korduse lõppu väljastab skript teate **Valmis**. Selle võtab vastu Krapsu skript, mis arvutab tabamuste protsendi ning kui see ei ole väiksem kui 50, Kraps ütleb „Tubli!“ ja teeb 3 kuni 5 hüpet.

Lõputu kordus

Formaalselt täidetakse ploki sees olevaid käske lõputult. Praktiliselt katkestatakse kordus ühel või teisel viisil, kui peatatakse selle või kõikide skriptide töö. Alati saab katkestada kõik tegevused punase nupuga, ühe skripti töö lõpetab hiireklõps skriptil. Sageli katkestatakse kordus käsuga [**peata skript**] või [**peata kõik**], viimane võib asuda ka mõnes teises skriptis.



kordus

tegevused
.....

lõpp kordus

Näide. Karistuslöögid. Fikseeritud aeg

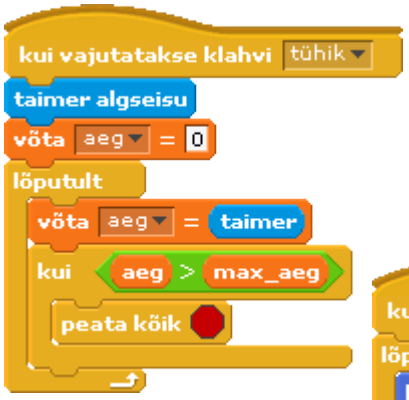
Nagu eelmises näites, imiteeritakse ka siin karistuslööke, muutes palli asukohta juhuarvude abil. Kuid programmi töö lõpetamine toimub siin etteantud aja lõppemisel. Kõik kolm skripti käivituvad vajutusega tühikuklahvile ja töötavad paralleelselt. Kõigis kolmes skriptis on kasutusel lõputu kordus. Esimene skript **Karistus_2** muudab palli asukohta, loendab lööke ja tabamusi. Kolmanda skripti **Sammumine** toimel jalutab Kraps laval edasi-tagasi. Teine skript **Stopper_1** kuvab jooksvat aega ja kontrollib selle väärtust ning kui ületatakse muutuja **max_aeg** väärtus, katkestatakse kogu programmi töö.

protseduur Karistus_2
```
tabas = 0
lööke = 0
kordus
   X = juhuarv(x1…x2)
   Y = juhuarv(y1…y2)
   pall.mine X, Y
   lööke = lööke + 1
   kui puudutab värav siis
      tabas = tabas + 1
   lõpp kui
   protsent  = tabas / lööke * 100
   paus 0,5…1 sek
lõpp kordus
```

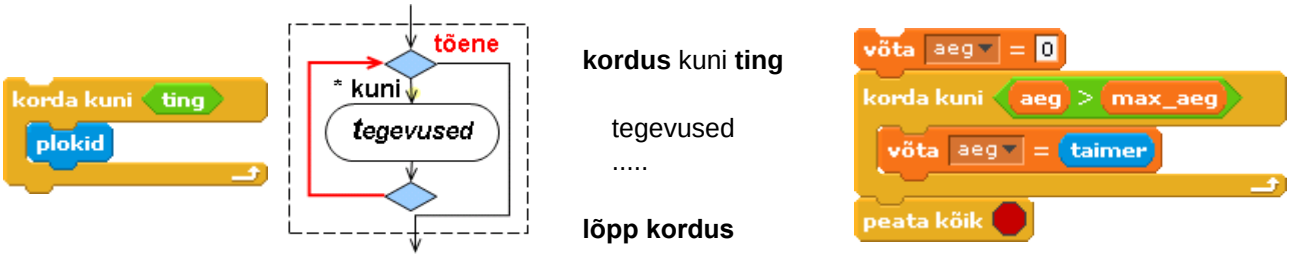
Muutujad: aeg, max_aeg, X, Y, tabas, lööke, protsent

protseduur Stopper_1
```
taimer = 0
aeg = 0
kordus
   aeg = timer
   kui aeg > max_aeg  siis
      Stopp
   lõpp kui
lõpp kordus
```

protseduur Sammumine
```
kordus
   liigu 10 sammu
   vaheta kostüüm
   oota ... sek
   kui puudutab serva siis
      põrka
   lõpp kui
lõpp kordus
```



Eelkontrolliga tingimuslik kordus



Selle korduse liigi korral **korratakse** tegevusi, **kuni** antud tingimus **saab tõeseks**. Taolist korduse liiki nimetatakse mõnikord ka eelkontrolliga **Until**-korduseks (kuni-korduseks). Inglise keeles on see [**repeat until** *tingimus*]. Paljudes programmeerimiskeeltes on selle asemel või lisaks sellele nn **While**-kordus (kui-kordus). Selle korduse tüübi korral **korratakse** tegevusi seni **kuni** tingimus **on tõene**.

Tingimuste esitamiseks kasutatakse Scratchis loogikaavaldisi, mis esitatakse võrdluste ja loogikatehete plokkide abil:

Loogigaavaldise väärtuseks saab olla ainult tõeväärtus: **tõene** (*true*) või **väär** (*false*). Loogikaavaldisi ja ülaltoodud plokke kasutatakse ka järgmistes jaotistes vaadeldavates käskudes.

Näide. Pall ja nõid

Taevas lendab ringi mingi nõiataoline olevus. Kasutaja võib üritada tabada teda palliga ning saada võimalikult palju punkte. Mäng kestab valitud ajani **max_aeg**. Kõigis kolmes toodud skriptis (protseduuris) kasutatakse tingimuslikku kordust. Esimene skript (protseduur) juhib nõia ringliiklust. Välises korduses kasutatakse lõputut kordust, mille katkestab taimeri skript. Sisemises korduses kontrollitakse pidevalt nõia x-asukohta; kui see saab suuremaks lava parempoolsest servast (x>240), peidetakse sprait korraks, viiakse lava vasakusse serva, tehakse nähtavaks ning lend jätkub. Parempoolne skript (protseduur) juhib palli lendu. Siin kontrollitakse korduses tingimust, kas pall jõuab lava ülemise ääreni. Kui jah, lõpetatakse kordus. Keskmine protseduur juhib aja kuvamist ning lõpetab kogu programmi töö, kui ületatakse valitud ajapiir.

Objektid: pall, nõid

Muutujad: aeg, max_aeg, viskeid, tabas

protseduur Lenda_nõid

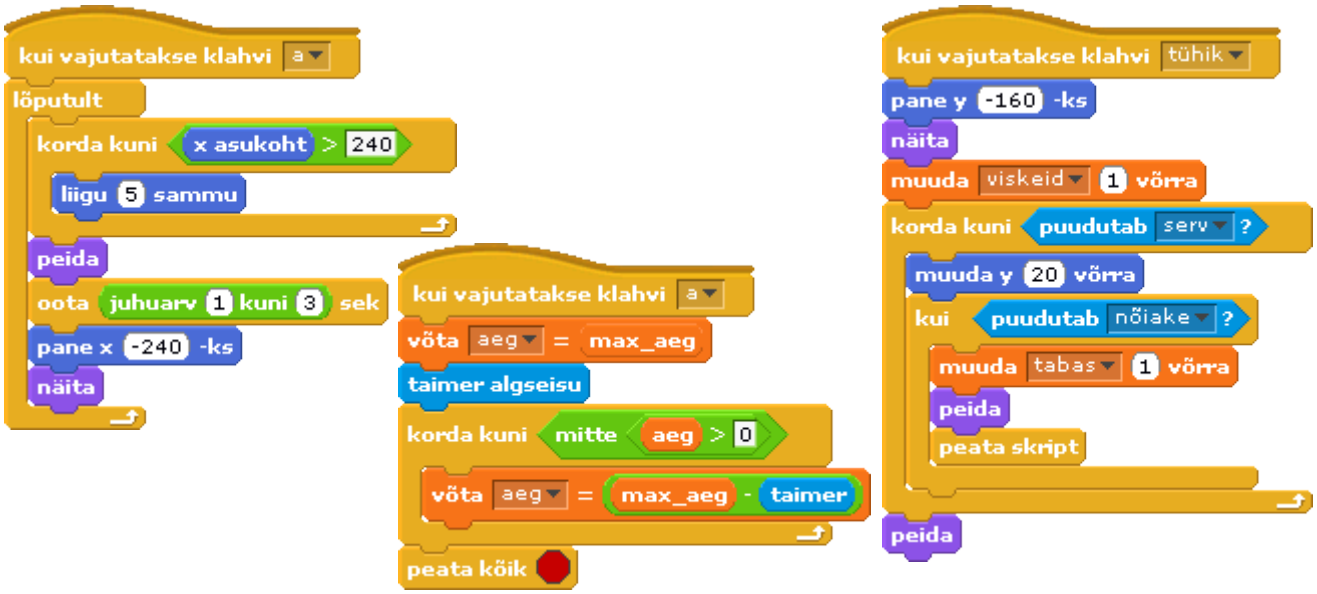
```
kordus
   kordus kuni nõid.x > lava.maxX
      nõid.liigu h
   lõpp kordus
   nõid.peida
   paus 1..3 sek
   nõid.x = lava.minX
   nõid.näita
lõpp kordus
```

protseduur Stopper_2

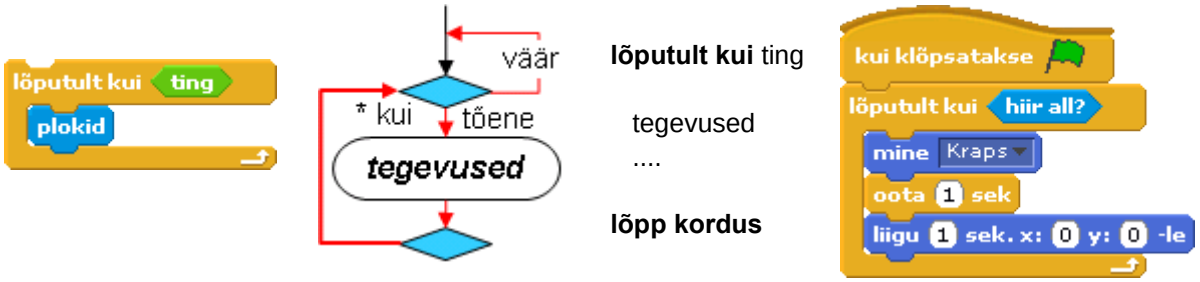
```
taimer = 0
aeg = max_aeg
kordus kuni aeg <= 0
   aeg = max_aeg - timer
lõpp kordus
Stopp
```

protseduur Pall_1

```
pall.y = lava.minY
pall.näita
viskeid = viskeid + 1
kordus kuni serv
   pall.y = pall.y + hy
   kui puudutab nõid siis
      tabas = tabas + 1
      pall.peida
      katkesta
   lõpp kui
lõpp kordus
```



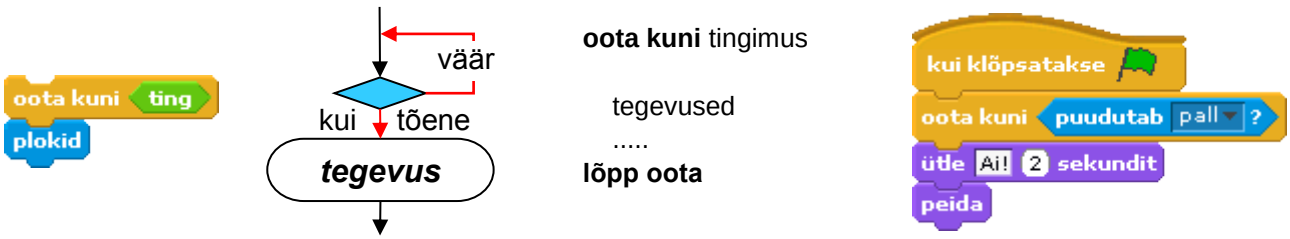
Lõputu tingimuslik kordus



Tegemist on omapärase korduste juhtimise variandiga, taolist teistes programmeerimiskeeltes ei leidu. Selle täitmisel kontrollitakse tingimust pidevalt (lõputult). Kui see on tõene, täidetakse tegevused ja kontrollitakse tingimust uuesti. Kui tingimus on väär, ei tehta midagi, aga jätkub tingimuse kontroll.

Sellise korduse saaks realiseerida ka lõputu kordusena, milles on valikulause (kui-plokk), milles olevad käsud täidetakse, kui tingimus on tõene.

Plokk oota kuni



Tegemist on omapärase „valvuriga“, võiks öelda, et varjatud kordusega. Kui täitmisjärg jõuab antud plokini, hakatakse pidevalt kontrollima tingimust ja skripti töö ei lähe enne edasi, kui tingimus saab tõeseks. Peab aga silmas pidama, et tingimuse väärtust antud skript ise muuta ei saa. Seda peab tegema mõni paralleelselt töötav skript.

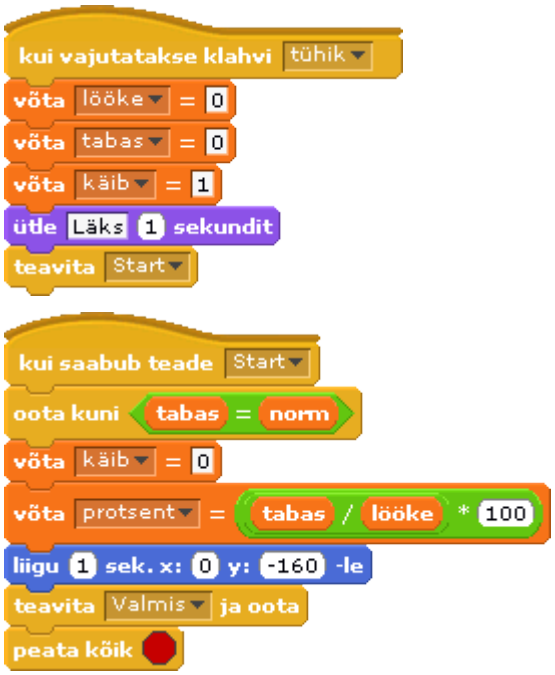
Näide. Karistuslöögid. Etteantud tabamuste arv

Tegemist on jälle karistuslöökidega. Seekord määratakse programmi töö lõpp etteantud tabamuste arvuga. Antud juhul ei saa löökide arv ega aeg olla fikseeritud, sest tegemist on juhuslikkusega. Programmi töö juhtimiseks kasutatakse plokke [**oota kuni** *tingimus*] ja [**lõputult kui** *tingimus*]. Oluline roll on siin ka

olekumuutujal **käib**. Kui **käib** = 1, toimub löökide sooritamine; kui **käib** = 0, lööke ei tehta. Alguses võetakse selle muutuja väärtuseks 0 ja käsuga [**teavita *Start***] käivitatakse kaks skripti. Esimese skripti alguses on plokk [**oota kuni tabas = norm**]. Käsk paneb skripti ooteseisu ja selle töö jätkub siis, kui tabamuste arv saab võrdseks etteantud normiga. Tabamuste arvu aga muudetakse teises skriptis.

Peale tabamuste arvu täitumist peab löökide sooritamine lõppema, kuid teha tuleb veel mõned muud tegevused: leida protsent, viia pall algseisu, Kraps peab tegema hüppeid. Kui praegu kasutaks nö tavalist lõputu korduse plokki, peaks kas peatama terve programmi töö või karistuslöökide löömine jätkuks veel seni, kuni saavad täidetud nimetatud tegevused.

Praegu on vastuolu lahendatud järgmiselt: kui plokis [**oota kuni …**] saab tingimus tõeseks, jätkub selle skripti täitmine ja muutujale **käib** omistatakse null. Sellega „lülitakse välja“ lööke juhtiv skript (tingimus **käib** = 1 ei ole enam tõene). Esimese skripti töö aga jätkub, kuni saavad tehtud kõik vajalikud tegevused.



protseduur Algseaded

```
lööke = 0
tabas = 0
käib = 1
teavita Start
```

protseduur Valvur (Start)

```
oota kuni tabas = norm
käib = 0
protsent  = tabas / lööke *100
pall.liigu lava.minY
teavita Valmis
Stopp
```

protseduur Karistus_3 (Start)

```
kordus kui käib = 1
   X = juhuarv(x1…x2)
   Y = juhuarv(y1…y2)
   pall.mine X, Y
   lööke = lööke + 1
   kui puudutab värav siis
      tabas = tabas + 1
   lõpp kui
   paus 0,5…1 sek
lõpp kordus
```

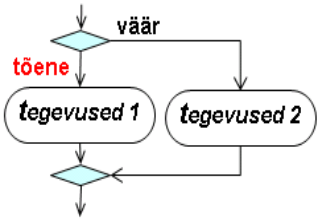
protseduur Valmis

```
kui  protsent >= 50 siis
   kuva „Tubli!“
   kordus juhuarv(3…5) korda
      Kraps.y = Kraps.y + hy
      paus t sek
      Kraps.y = Kraps.y - hy
      Paus t sek
   lõpp kordus
lõpp kui
```

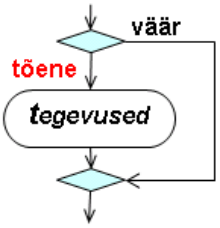
Hargnevad protsessid

Hargnevas protsessis valitakse antud tingimuste alusel mitmest võimalikust tegevusest üks. Vt ka Algoritmimine. Scratchis on kaks valikuplokki: **valik kahest** ja **valik ühest**. Esimesel juhul täidetakse kahest võimalikust plokkide grupist üks. Teisel juhul, olenevalt tingimuse väärtusest, antud plokid kas täidetakse või jäetakse vahele. Iga plokk võib omakorda sisaldada **kui**-plokke. Tingimuste esitamiseks kasutatakse loogikaavaldisi, mis esitatakse võrdluste ja loogikatehete plokkide abil:

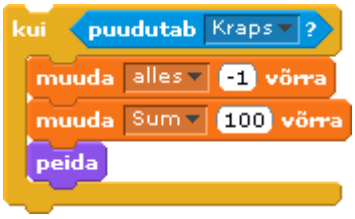
Loogigaavaldise väärtuseks saab olla ainult tõeväärtus: **tõene** (*true*) või **väär** (*false*).



kui tingimus **siis**
tegevused 1
muidu
tegevused 2
lõpp kui



kui tingimus **siis**
tegevused
lõpp kui



Näide. Suurim kolmest arvust

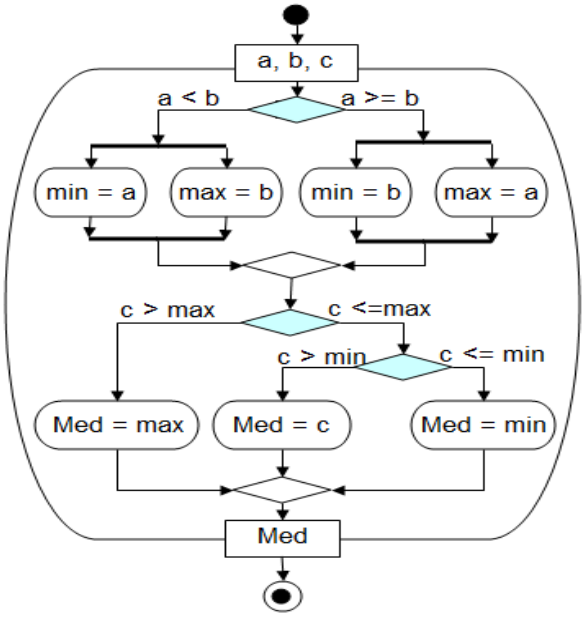
Nn klassikaline loogikaülesanne. On antud kolm arvu: **a**, **b** ja **c**, programm peab leidma neist suurima ja omistama selle väärtuse muutujale **max**. Vaatamata ülesande lihtsusele, on siin võimalikud mitu erinevat varianti. Allpool on toodud kolm.

Esimeses skriptis on kasutusel kolm järjestikust **kui**-plokki. Igas plokis võrreldakse ühte arvu kahe teisega. Kui tingimus on tõene, võetakse vastav arv muutuja **max** väärtuseks. Algoritm ei ole eriti ratsionaalne. Oletame, et suurim arv on **a**, see omistatakse muutujale **max** kohe esimeses plokis, kuid edasi kontrollitakse järgmisi tingimusi ikka. Seega täidetakse 9 tehet: igas tingimuses 2 võrdlust ja üks loogikatehe.

Teises skriptis valitakse suurim väärtustest **a** ja **b** ning omistatakse muutujale **max**. Seejärel võrreldakse **c** väärtust **max**-ga ja kui see on suurem, võetakse **max** väärtuseks **c**. Halvimal juhul on kaks võrdlust ja kaks omistamist. Kolmandas skriptis võetakse kohe **max** algväärtuseks esimene arv **a**. Edasi võrreldakse teistega.

Näide. Kolme arvu mediaan

Valitakse kolmest arvust vahepealne (mediaan). Kõigepealt valitakse **min** väärtuseks väiksem **a**-st ja **b**-st ning **max** väärtuseks suurem. Edasi toimub võrdlus **c**-ga ja tehakse kindlaks mediaan.



funktsioon Mediaan

kui a < b **siis**
  min = a; max = b
muidu
  min = b; max = a
lõpp kui
kui c > max **siis**
  tagasta max
muidu
  kui c > min **siis**
    tagasta c
  muidu
    tagasta min
  lõpp kui
lõpp kui

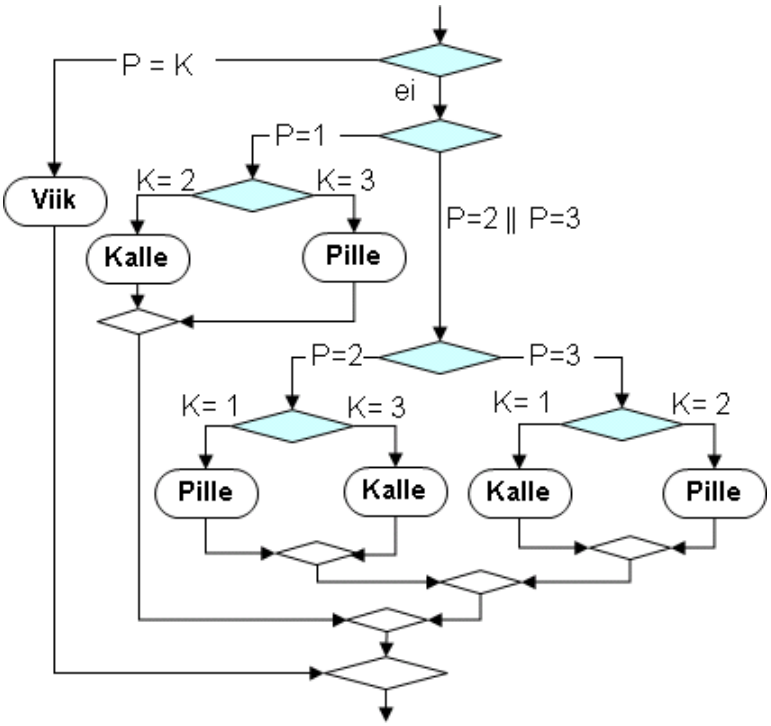
Näide. Fragment mängust „Kivi, paber, käärid“

Kaks mängijad (näiteks **Pill**e ja **Kalle**) näitavad korraga ühe märkidest: kivi, paber või käärid, kasutades mingit kokkulepet. Näiteks üks, kaks või kolm sõrme. Võitja tehakse kindlaks järgmiste reeglite kohaselt:

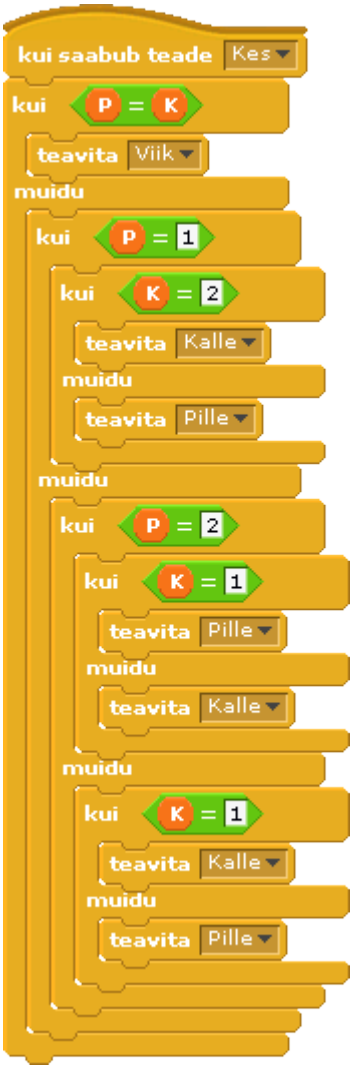
- **kivi** võidab **käärid**,
- **käärid** võidavad **paberi**,
- **paber** võidab **kivi**

Programmis omistatakse kahele muutujale ***P*** ja ***K*** juhuslikud väärtused: 1(kivi), 2 (paber), 3 (käärid). Nüüd tuleb koostada skript (protseduur), mis teeb **P** ja **K** väärtuste järgi kindlaks võitja või fikseerib viigi.

Selleks on väga palju erinevaid variante. Allpool on toodud algoritm, milles kasutatakse järjestikuseid võrdlusi.



kui P = K **siis**
  teavita viik
muidu
  kui P = 1 **siis**
    kui K = 2 **siis**
      teavita Kalle
    muidu
      teavita Pille
    lõpp kui
  muidu
    kui P = 2 **siis**
      kui K = 1 **siis**
        teavita Pille
      muidu
        teavita Kalle
      lõpp kui
    muidu
      kui K = 1 **siis**
        teavita Kalle
      muidu
        teavita Pille
      lõpp kui
    lõpp kui
  lõpp kui
lõpp kui



Nagu näha, on tegemist üsna pika ja võrdlemisi ebaülevaatliku esitusega, seda eriti Scratchi skriptis.

Kui kasutada loogikaavaldisi on lahendus kompaktsem ja selgem, kuigi Scratchi skript läheb veidi suureks teises suunas.