1.1 Algoritm

ALGORITM

Kui me midagi suuremat teha tahame, siis võib tulemuse saavutamisel abi olla sellest, kui me kavandatu kuidagi sammudeks jaotame. Seejuures me peaksime muidugi mõtlema, mis järjekorras need sammud tuleb või saab teha. Kas mingeid samme saab teha samaaegselt? Kas midagi tuleb teha korduvalt? Kas mõnda asja tuleb või saab teha ainult teatud tingimustel? Äkki saab hoopis keegi teine meie endi asemel midagi ära teha?

Elus paraku (kahjuks või õnneks) mõeldakse sammude peale sageli alles pärast nende toimumist. Kui aga tahta kellelegi teisele (aga endalegi) ikkagi selgeks teha, kuidas teatud asi toimuma peaks (või toimus), siis peaks selleks kasutama mingit mõlemale poolele arusaadavat viisi. Vast on nii mõnedki kogenud, et teised inimesed (isegi lähedased) ei pruugigi suuta mõtteid lugeda! Saavad asjadest hoopis teistmoodi (et mitte öelda valesti) aru! Või ei saa üldse aru!

Millised on põhimõttelised võimalused suunata ümbritsevat maailma sobivalt käituma? Näiteks saab kehtestada eeskirjad. Need võivad olla negatiivsed - midagi keelavad või piiravad. Ära mine vales kohas üle tänava! Ära söö (joo) nii palju! Ära ületa kiirust! Ära mängi nii palju arvutimänge! Ära …

Siin keskendume pigem positiivsetele eeskirjadele - neile, mis kirjeldavad, mida tuleb teha. Vast on nii mõnedki kogenud, et täpsed juhised võivad aidata tulemuse saamisele kaasa. Pööra ringteelt välja teiselt mahasõidult! Keeda 20 minutit! Lisa kaks labidatäit kruusa! Istuge, palun!

Konkreetse ülesande lahendamiseks võib vaja olla mitmeid juhiseid üksteise järel mitu ja saame rääkida lahenduseeskirjast ehk **algoritmist** (vt Eesti keele seletav sõnaraamat)

ALGORITMIDE ESITAMINE

Algoritme saab esitada erineval moel. Üks levinud viis on plokkskeemi kasutamine. Siingi on erinevaid võimalusi, meie kasutame sellist, kus algoritmi algust ja lõppu kujutatakse ovaalide abil. Algoritmil on alati üks algus ja üks lõpp. Vajadusel võime erinevad lõpud "kokku tõmmata". N-ö tavalise sammu tähistamiseks kasutatakse ristkülikut. Plokkide järgnevust märgitakse nooltega. Nii kirjeldavad (tänaseks juba emeriitprofessorid) Mare Koit, Ülo Kaasik ja Jüri Kiho oma õpikus "Kuidas programmeerida" (1990) kartulisalati tegemist. (Neil oli küll veidi keerulisem retsept.)

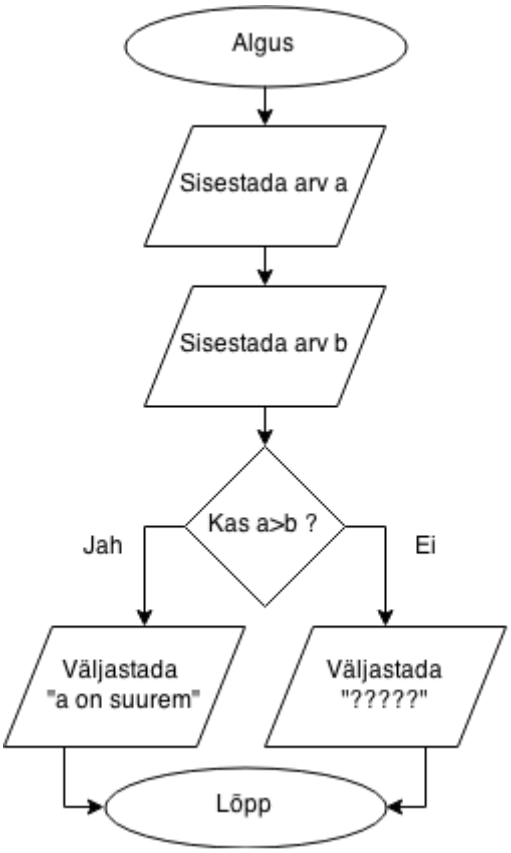
On üsna selge, et samale tulemusele võib jõuda ka mõnevõrra teistsuguse algoritmiga. Mõned sammud võivad olla teises järjekorras, aga mõnede sammude puhul on omavaheline järjekord kindel. See, millised sammud ühte plokki lugeda, võib olla ka üsna vabalt valitav. Kui tahame midagi paralleelselt teha, et tulemust kiiremini saada, siis võiks salati tegemisel kartulite ja hapukurgi tükeldamist eraldi võtta ja neid lasta erinevatel inimestel samal ajal teha. Siinkohal kerkib loomulik küsimus: kas meil on olemas selleks vajalikke ressursse - inimesi, nuge, lõikelaudasid? Paralleelsete protsesside kasutamine programmides on praegusel ajal tegelikult äärmiselt oluline, aga ka keeruline ja see jääb meie kursusest välja.

Ülesanne

Tegemist on enesekontrolliülesandega, mille lahendamise tulemusi ei salvestata. Võite julgesti ka valesti vastata, aga püüdke ikka õigesti. Tuleb silmas pidada, et siin ja mitmetes kohtades edaspidi on ülesannetes elu lihtsustatult käsitletud. Päriselt võib ju kuulutust lugeda nii enne pileti ostmist kui ka pärast seda jms. Siin aga loetakse õigeks üks (arvatavalt levinuim) lahendus.

Mitme tegevuse paralleelselt tegemist me siin ei käsitle. Küll aga on meil olulisel kohal olukorrad, kus tuleb kahest võimalikust jätkust valida üks. Plokkskeemis kasutame selliste hargnemiste - kontrollplokkide - kirjeldamiseks rombi. Kontrollplokis on olulisel kohal tingimus, mille täidetuse põhjal otsustatakse, kumba teed edasi minna. Kontrollplokist väljub alati täpselt kaks noolt, sissetulevate noolte hulk ei ole piiratud.

Kui nüüd tulla rohkem arvutite juurde, siis üsna sageli on vaja mingit sisendit ja tulemuseks on millegi väljastamine. Sellist andmevahetust märgitakse rööpkülikutega. Järgmises algoritmis sisestatakse arvud a ja b ning pärast kontrolli väljastatakse vastav teade.



Ülesanne

Milline on eelnevas skeemis sobiv tekst ????? asemel?

Vali a on suurem

Vali a ei ole suurem

Vali a on väiksem

Vali a ei ole väiksem

Algoritmi koostamisel tuleb olla väga tähelepanelik ka kõikvõimalike erijuhtude arvestamiseks. Näiteks eelmise algoritmi puhul peab väljastatav tekst olema õige ka juhul, kus sisestatud arvud a ja b on võrdsed.

1.2 Programm

PROGRAMMEERIMISKEELED

Mingi ülesande lahendamiseks vajaliku algoritmi kirjapanek plokkskeemina võib küll olla arusaadav inimesele, aga see ei ole väga sobiv arvutile algoritmi arusaadavaks tegemiseks. Selleks otstarbeks tuleb lahendus esitada **programmina** (vt Eesti keele seletav sõnaraamat). Programm kirjutatakse mingis programmeerimiskeeles. Programmeerimiskeeli on olemas mitmeid sadu, võib-olla tuhandeid. Nagu loomulike keelte puhul on ka üks programmeerimiskeel (olemuselt siis tehiskeel) mõne teisega sarnasem, mõnest jälle erinevam. Näiteks eesti keel on suhteliselt sarnane soome keelega, vene keel ukraina keelega ja rootsi keel norra keelega. Samas näiteks kuigi eesti keel ja katalaani keel ei ole väga sarnased, on neis mõlemas ikkagi nimisõnad ja tegusõnad jm paljudele keeltele omased asjad. Nii on ka päris suuresti erinevates programmeerimiskeeltes sarnaseid struktuure. Programmeerimiskeeli saab tinglikult jaotada põlvkondadesse. (Põlvkondadest ja teistest ajaloolistest asjadest on juttu silmaringimaterjalis Katkeid programmeerimise ajaloost.)

Rõhuv enamik programme kirjutatakse tänapäeval mingis kõrgtasemekeeles. Neist levinumad on näiteks C (ka C++, C#), Java, PHP ja Python. Erinevate keelte kasutuslevik on pidevas muutumises ja seda on huvitav jälgida tabeli või graafiku abil.

Meie kursuse aluseks on võetud programmeerimiskeel Python. Ühelt poolt on tegemist keelega, milles tõesti ka päriselt tööstuses programmeeritakse. Teiselt poolt on see sellistest keeltest sobivaim just esimeseks õpitavaks keeleks, sest juba midagigi tegeva programmi saame kirjutada ühe rea abil. Programmeerimiskeele Python esimene versioon loodi 1989. aasta lõpus. Sellel kursusel kasutame versiooni Python 3. Täpsemalt pole versioon (nt. 3.4.3 või 3.5.0 või 3.6.2) nii oluline, kuna meie jaoks on nende erinevused vähetähtsad.

Programmeerimiskeele Python looja nime juurde jõuame aga läbi järgmise küsimuse. Ka edaspidi kasutame vahel küsimusi juba siis, kui vastust veel tekstis toonud pole. Ühelt poolt tundub see võib-olla mõnevõrra kummaline - kuidas küsida õppurilt sellist asja, mida pole veel seletatudki!? Teiselt poolt aga näitavad kogemused, et sellised küsimused aitavad õppija vaimu virge hoida ja paremini teemasse sisse minna. Lisaks ei lähe vastuse hinne kuhugi arvesse, seega pole oma tulemuse pärast vaja liigselt muretseda. Enesekontrolli ülesannete puhul võite julgesti ka meelega valesti vastata, sageli ilmuvad sellisel juhul õpetlikud kommentaarid.

Ülesanne

Kes oli programmeerimiskeele Python looja?

Vali Jean-Claude van Damme

Vali Marco van Basten

Vali Guido van Rossum

Vali Vincent van Gogh

Programmeerimiskeele Python autori kohta leiab natuke infot eesti keeles ja natuke rohkem inglise keeles.

ESIMENE PROGRAMM

Üks põhjus, miks Python on esimese keelena suhteliselt sobiv, on see, et päris esimese programmi kirjutamine on lihtne. Nii saame teha näiteks ühereal ise programmi

```
print("Tere!")
```

See programm toob ekraanile sõna "Tere!".

Tavaliselt muidugi on programmis rohkem ridu. Täiendame meiegi oma programmi.

```
print("Tere!")
print("Head aega!")
```

Proovimegi nüüd päriselt programmeerima hakata.

THONNY (AGA KA PYTHONI) PAIGALDAMINE ARVUTISSE

Selleks, et programmeerimiskeeles Python programmeerima hakata, on vaja Python oma arvutisse saada. Pythoni saab paigaldada eraldi, aga meie kasutame oma kursusel keskkonda Thonny, millega Python juba kaasas on.

Palun paigaldage enda arvutisse Thonny (ja tegelikult siis ka Python 3). Selleks saate abi juhistest.

Püüdke nende juhiste abil eespool olnud lühike Tere-programm käivitada. Tegemist on selle kursuse jaoks väga olulise etapiga. Kui saate esimese programmi tööle, siis on suur võimalus, et saate järgmisedki! Esialgu võivad need sammud võõrad tunduda, aga lähinädalatel hakkate programme koostama, käivitama, parandama ja jälle käivitama kümneid ja sadu kordi.

Pythoni (nagu mistahes teisegi keele) programm on ühelt poolt täiesti tavaline tekstifail, mida võime kirjutada tavalise tekstiredaktoriga (näiteks Notepadi või Wordiga (salvestades vorminduseta tekstina)). Siiski on seda mugavam teha spetsiaalse redaktoriga, nt Thonnyga. (Tegelikult on olemas ka mitmeid teisi toredaid redaktoreid.)

Kui Thonny paigaldamine oma arvutisse ei õnnestunud, siis paluge abi.

Lühema programmiga saime hakkama, püüame natuke pikemaga ka. Seda ei ole mõtet ise sisse toksida, vaid võiks siit kopeerida. Proovige programm käima saada!

```
arv = 1
if (arv > 0):
    print("positiivne")
else:
    print("mittepositiivne")
```

Pythonis on programmi struktuuri korraldamisel äärmiselt oluline õige taandamine. (Teistes keeltes võivad selles rollis olla näiteks looksulud { }). Proovige näiteks taanet muuta.

```
arv = 1
if (arv > 0):
print("positiivne")
else:
    print("mittepositiivne")
```

Kui mitte varem, siis nüüd juhtus selline asi, et me tahtsime käivitada vigast programmi. Seda juhtub programmeerimisel alatihti ja seda ei tohi sugugi karta. Proovige parandada ja uuesti käivitada.

LÕPETUSEKS

Loodetavasti olete nüüd midagi teada saanud algoritmist ja programmist! Ja esimesed programmid töölegi saanud. Seda ei ole üldse vähe! Ikka julgelt edasi!

1.3 Silmaring. Katkeid programmeerimise ajaloost

ALGUS

Ühelt poolt on programmeerimise ajalugu küllaltki lühike, eriti võrreldes mõnede teiste valdkondadega. Teiselt poolt on aga siingi juhtunud palju huvitavaid sündmusi, on olnud erinevaid ajajärke ja tegutsenud värvikaid persoone. Siin saame tutvustada vaid vähest, aga loodetavasti mingi (küll väga fragmentaarse) taustapildi siiski saab.

Programmeerimise ajalugu algab juba enne seda, kui arvutid füüsiliselt valmis said. Nimelt leidis inglise matemaatik Charles Babbage 1812. aastal, et teatud arvutusi võiks teha hoopis masin. Aastakümnete jooksul tegeles ta selle mõtte realiseerimisega. 1842. aastal tutvustas ta loengus universaalse mehaanilise arvuti - analüütilise masina ideed. Teatud eeskujuks olid automaatsed kangasteljed. Varasematest ideedest eristas seda masinat just see, et protsessi pidi juhitama varemkoostatud juhiste järgi. C. Babbage püüdis seda masinat ka valmis ehitada, aga tolleaegsete tehniliste võimaluste piiratuse tõttu see ei õnnestunud. Masina lihtsam kuju tehti valmis aastaid hiljem. Mitmed tema põhimõtted aga leidsid rakendamist hilisemates arvutites.

Analüütilise masina loengu põhjal avaldati artikkel, mille C. Babbage palus inglise keelde tõlkida Ada Lovelace'il, kes oli luuletaja lord Byroni tütar. Lisaks tõlkimisele lisas A. Lovelace artiklile ka kommentaare, mille hulgas olid ka juhised, kuidas selle masina abil leida Bernoulli arve. Hiljem on neid juhiseid hakatud pidama ajaloo esimeseks programmiks ja Ada Lovelace'i esimeseks programmeerijaks. Rohkem programme ta teadaolevalt ei kirjutanud ja ei saanud seda ainsatki masinal reaalselt testida. See, et esimene programmeerija oli naine, on vahel olnud inimestele üllatav. Tegelikult saab programmeerimisega muidugi tegeleda soost sõltumata.

SALADUSED

Hüppame nüüd ajas umbes sada aastat edasi. Vahepeal oli masina abil näiteks edukalt analüüsitud USA rahvaloenduse andmeid, kusjuures masina kasutamine andis tohutu ajavõidu. Perfokaarte oli kasutatud näiteks raamatupidamises. Teise maailmasõja eel ja ajal oli suur osa tegevusest seotud sõjandusega. Võib-olla isegi sõja lõpptulemust oluliselt mõjutanud saaga on seotud saksa šifreerimisaparaadi Enigma koodi lahtimurdmisega. (Sellest on ka juttu filmides, nt Imiteerimismäng (The Imitation Game) .) Nimelt vahetasid saksa staabid, allveelaevad jm omavahel sõnumeid Enigmaga šifreeritult. Selle koodi lahtimurdmisega tegeles spetsiaalne töörühm Londoni lähistel Bletchley Parkis. Töörühmal dešifreerimine õnnestuski ja nende töö andis ka impulsi inglise elektronarvuti Colossus loomiseks. Sakslaste sõnumitest arusaamine võimaldas oma tegevust paremini planeerida ja kui see polnudki põhiline põhjus, miks sõda sedapidi lõppes, siis olulise panuse see kindlasti andis. Bletchley Parki saab ka külastada.

Üks Bletchley Parki töörühma liidreid oli kahtlemata värvikas isiksus Alan Turing. Ühelt poolt oli tegemist kindlasti äärmiselt andeka matemaatiku ja informaatikuga. Tema tööd on arvutiteaduses fundamentaalse tähtsusega: Turingi masin ja Turingi test kannavad lausa tema nime.

Teiselt poolt oli tegu näiteks innustunud pikamaajooksjaga, kes võis Bletchley Parkist üle 60 km kaugusele Londonisse koosolekule joosta. Tema maratonijooksu rekord oli üsna arvestataval tasemel. Kuna ta kannatas heinapalaviku all, siis teatud perioodidel sõitis ta jalgrattaga tööle, gaasimask peas. Isiklik elu oli A. Turingil traagiline. 1952. aastal mõisteti ta süüdi ebasündsuse paragrahvi alusel, mille alla homoseksuaalsus tol ajal käis. 1954. aastal leiti Alan Turing tsüaniidimürgituse tagajärjel surnuna. Käeulatuses oli pooleldisöödud õun … Kuninganna Elisabeth II tühistas süüdimõistva otsuse 2013. aastal.

PROGRAMMEERIMISKEELED

Räägime nüüd ka natuke sellest, kuidas aegade jooksul arvutile oma soove on teada antud. Juba mehaanilistest masinatest peale on olulisel kohal olnud suhteliselt kahevalentne lähenemine nii programmide kui andmete osas. Nii on näiteks perfokaardil või perfolindil mingis konkreetses kohas auk või seda ei ole, mingi lamp põleb või ei põle, mingis pesas on midagi või pole. Arvude kujul on seda mõistlik kirja panna vaid kahe numbri, 1 ja 0 abil.

Programmeerimiskeeled võib (mõnevõrra tinglikult) jaotada põlvkondadesse. Nii saab eristada näiteks:

1. põlvkond - masinkood
2. põlvkond - assemblerkeeled
3. põlvkond - kõrgtasemekeeled

Masinkoodis programmid koosnevad tinglikult ainult ühtedest ja nullidest, näiteks

00000000001000100011000000100000

võib tähendada "liita aadressidel 1 ja 2 olevad arvud ning salvestada resultaat aadressile 6“. Sellist programmi on inimesel raske lugeda ja kirjutada, masinale on see aga hästi “seeditav”. Assemblerkeeles programmitekst on juba inimesele natuke paremini mõistetav, näiteks

add $1, $2, $3

Siiski loetakse assemblerkeeles kirjutatud programme madalatasemeliseks. See tähendab siinkohal arvutilähedust - assemblerkeelt valdavad programmeerijad on ise reeglina just kõrgetasemelised. :-)

Kõrgtaseme keeles programmi suudab ettevalmistunud inimene hästi kirjutada ja lugeda, masina jaoks tuleb seda aga transleerida. (Transleerimine ongi konkreetne termin, kuigi olemuselt see muidugi tõlkimist tähendabki.) Järgmiste põlvkondade programmeerimiskeeled peaksid olema veelgi rohkem programmeerijasõbralikumad ja tehisintellekti abil ülesandeid pigem ülesande (inimkeelse?!) kirjelduse kui juba etteantud lahendussammude järgi lahendama.

1.4 Kontrollülesanne I

Palun joonistage paberile plokkskeem mingi elulise protsessi kohta. Kokku peab seal olema vähemalt 5 plokki ja nende hulgas vähemalt üks kontrollplokk, milles hargnemine toimub kas-küsimuse põhjal.

Kursusele registreerunutele tuleb selle ülesande kohta küsimus nädala lõputestis.

1.5 Maalähedane lugu

Erindveres, mitte kaugel Voorvaili mäest, asuvas talus elavad paikselt väärikas ja vägagi käbe taluperenaine Ada (90) ning tema poeg Lembit (60). Sageli on seal ka Ada lapselapselaps Rasmus (18), viimasel ajal koos oma tüdruksõbra Liisbethiga.

> VAHELEPÕIGE Võib-olla natuke üllatuslikult (või siis mitte nii väga üllatuslikult) on paljugi seotud kuidagi programmeerimisega. Näiteks erindiks nimetatakse teatud erandolukorda programmi töös. Ja miks Voorvaili mäel selline nimi on ...

Lood on vahendanud Eno Tõnisson. Pildid on joonistanud Alar Kriisa.

Maalähedane lugu I

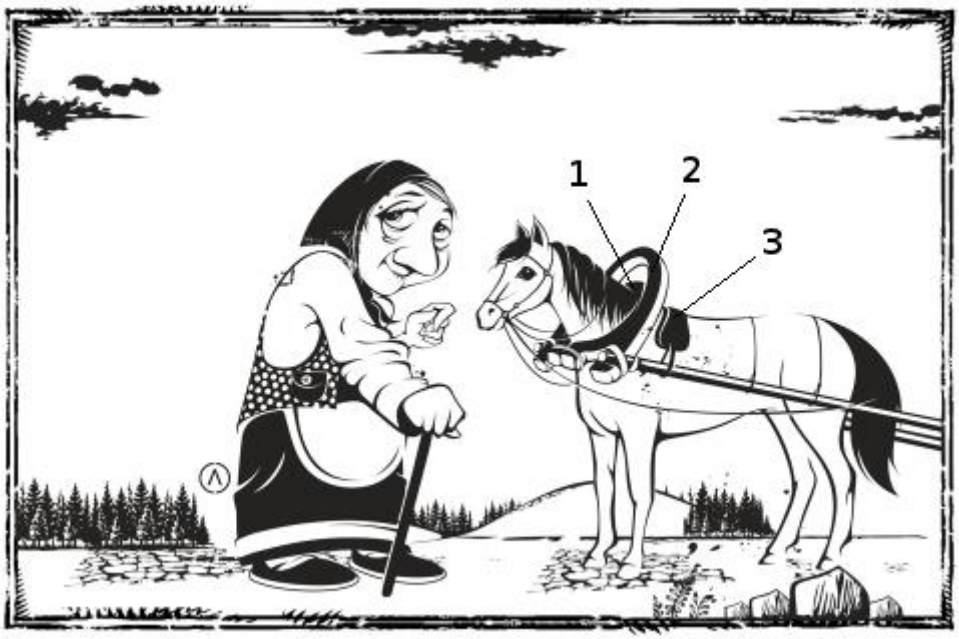
"Oi, kurjam," vandus Ada endamisi, kui oli mingitesse rihmadesse takerdunud. Pidi see Lembit just täna linnas olema?! Ja see antenn peab ka tingimata just pööningul olema! Mis pagana rihmad need üldse on? Sedelgarihmad! Miks ei võiks hobuseriistad tallis olla? Muidugi oli tal meeles, et siin talus sedelgat ei kasutatud ja tallis looga ja rangide juures polnud see olnud vist kunagi.

Alt kostus ukse kolksatus. Kas Lembit oli juba tagasi tulnud? Ei, see oli hoopis Rasmus koos oma pruudiga - või tüdruksõbraga - või kuidas nad nüüd ütlevad. Tubli poiss muidugi, et maal ikka käib, aga kas see ninarõngaga tüdruk just see õige on. Aga no mis minagi sest tean või teadma pean. Vähemalt ilusti räägib see tüdruk küll - ikka ütleb "sääl" ja "hää". Nagu Juhan Liivi luuletuses:

sääl ajab kõrvits ennast laiali
ja küüslauk peenra äärel nagu muru,
sääl kaalik, porgand kasvab priskesti,
siit rohtu kistud, kohendatud puru.

Juhan Liiv oli Ada lemmik. Kui Ada eelmisel sügisel Tartus silmakliinikus käis, siis läks ta jalgsi Raekoja platsile ning juhtus ka Juhan Liivi tänava nurgale - Rahvaste monumendi juures. Esimest korda oli Ada Tartus käinud juba enne sõda koos isaga. Varavalges olid hakanud vankriga minema. Tänapäeva noored vist ei oska hobustki ette rakendada. Muidugi ei oska! Polnud siingi talus hobust juba üle kahekümne aasta olnud. Igatahes võttis Ada nüüd sedelgarihmad kaasa ja kiirustas trepist alla - Rasmus, hea poiss, võib ise pärast antenni sättida.

Üheksakümneaastase kohta oli Ada vägagi käbe ja ta sai tõesti trepist ludinal alla. Kiire oli muidugi ka sellepärast, et saaks noortele kohe hobuse etterakendamist seletama hakata. Ega seal ju midagi keerulist polnud - rangid hobusele kaela, siis loogaga aiste külge ja valmis. Sedelgat siin talus ei kasutatud - väiksemad tagurdamised sai ka ilma ära teha. Tegelikult pidi ikka tähelepanelik ka olema. Kui rangirihmad valele poole loogaotsi panna, siis võis hobune ühel hetkel lihtsalt aiste vahelt välja jalutada.



1 - rangid, 2 - look, 3 - sedelgas

VAHELEPÕIGE Ada, Liisbeth ja Rasmus ei mõelnud muidugi sellest, et hobuse rakendamine on suuresti sarnane programmi kirjutamisega. Mõned asjad peavad olema kindlas järjekorras, mõnede järjekord on suhteliselt vaba. Mõned asjad võib esialgu (või üldse) panemata jätta, aga teatud olukordades võib see väga ebameeldivalt lõppeda.

Rasmus ja Liisbethi jaoks oli hobusetemaatika üsna kauge. Liisu (nagu Liisbethi ikka kutsuti) ei oodanud enam (või veel?) printsi valgel hobusel ja Rasmus oli hobuste asemel pigem mootorratastest huvitatud. Mootorratas, Lembitu vana Jawa, oli muide üks mitmest põhjusest, miks ta heameelega Erindveres käis. Aga Adat kuulasid nad siira uudishimuga, nagu ikka. Ada oli hea jutustaja. Nii saidki nad lisaks hobuse etterakendamise tarkusele, mille jaoks muide talli vana ree juurde mindi ja päriselt looga ja rangidega mütati, teada veel mitmeid asju vanemast ja uuemast minevikust. Kes oleks võinud arvata, et ...