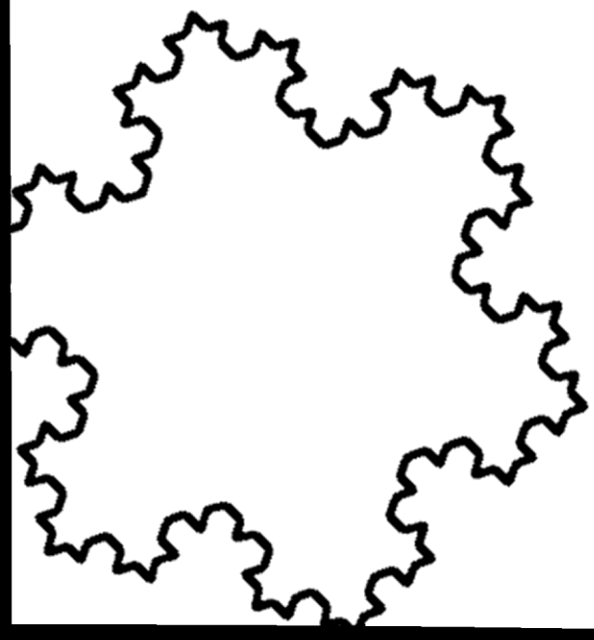
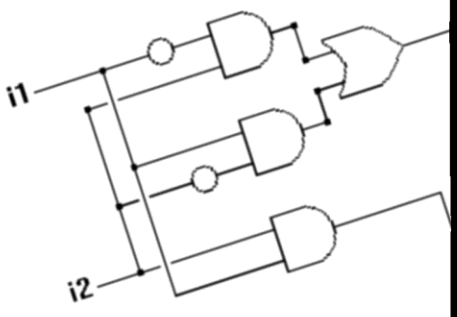
Tölvufræði



Atli Harðarson

Kennslubók í tölvufræði
fyrir framhaldsskóla



Atli Harðarson

Kennslubók í tölvufræði
fyrir framhaldsskóla

Iðnú 2001

Efnisyfirlit

Inngangur

Á árinu 2000 gaf Iðnú út bók eftir mig sem heitir *Java – kennslubók í forritun fyrir framhaldsskóla*. Sú bók var samin sem kennslubók í forritun fyrir áfangana TÖL 103 og TÖL 203 eins og þeim er lýst í *Aðalnámskrá framhaldsskóla*. Í þessum áföngum á, samkvæmt námskránni, að kenna fleira en forritun því þar á einnig að kynna nokkur meginatriði og undirstöðuhugtök tölvufræðinnar. Það vantaði því lesefni í tölvufræði til að nota samhliða kennslubók í forritun. Þessari bók er ætlað að bæta úr þeirri vöntun.

Þótt bókin sé samin til þess að nota samhliða kennslubók í forritun er hún sjálfstæð heild og ætti að nýtast öllum sem hafa áhuga á að fræðast um tölvutækni.

Þegar ég hóf samningu bókarinnar hlaut ég til þess styrk frá menntamálaráðuneytinu. Fyrir það ber að þakka. Einnig þakka ég þeim mörgu sem hafa sent mér ráð, ábendingar og athugasemdir einkum þó eiginkonu minni, Hörpu Hreinsdóttur, sem las allan textann yfir og hjálpaði mér að færa margt til betri vegar.

Október 2001
Atli Harðarson
atli@ismennt.is
http://www.ismennt.is/not/atli/

1. kafli

Saga tölvutækninnar

1.1 Forsagan

Vísindabylting – iðnbylting

Fyrstu tölvurnar voru smíðaðar á árunum milli 1945 og 1950. Smíði þeirra byggði á þekkingu og kunnáttu úr mörgum greinum vísinda og tækni sem sumar eiga sér langa sögu.

Á 17. öldinni urðu raunvísindi nútímans til. Helstu upphafsmenn þeirra eru jafnan taldir Ítalinn *Galíleo Galilei* (1564 – 1642) og Englendingurinn *Isaac Newton* (1642 – 1727). Með uppgötvunum þeirra og annarra raunvísindamanna urðu slík straumhvörf að talað er um *vísindabyltinguna á 17. öld*. Þessi bylting markar upphaf nútímans. Síðan hefur saga mannkynsins mótast öðru fremur af stórstígum framförum í vísindum og tækni.

Framþróun vísindanna gerði það mögulegt að tæknivæða ýmsar atvinnugreinar. Notkun gufuvéla í atvinnulífi hófst í Vestur-Evrópu á seinni hluta 18. aldar. Um það leyti er talað um að *iðnbyltingin* hefjist. Helsta einkenni á *fyrsta hluta* hennar er að vélarafl tekur í æ ríkari mæli við af vöðvaafli og leysir menn undan líkamlegri erfiðisvinnu.

Annar hluti iðnbyltingarinnar hófst á síðustu árum 19. aldar. Þá tóku menn að hagnýta rafmagn í mjög auknum mæli. Fyrsta rafveita í heimi var byggð árið 1882 í New York borg. Það var uppfinningamaðurinn *Thomas A. Edison* (1847 – 1931) sem stjórnaði því verki. Síðan Edison var og hét hefur raftækni hvers konar sett æ meiri svip á líf alls almennings. Tölvutæknin er ein grein raftækninnar en með henni hefst líka *þriðji hluti iðnbyltingarinnar* sem einkennist af sívaxandi sjálfvirkni og því að vélar taka við alls konar hugarstarfi og leysa menn þannig undan andlegri erfiðisvinnu.

Tölvutæknin byggir á hugmyndum og hugsunarhætti sem urðu til með vísindabyltingunni, framleiðsluaðferðum og samfélagsgerð sem mótuðust í iðnbyltingunni og raftækni sem tók að mótast á síðustu árum 19. aldar. Auk þess sækir hún mjög til nýjunga í stærðfræði og rökfræði sem komu fram á fyrstu áratugum 20. aldar (og er sagt ofurlítið frá í kafla 7.3).

Reiknivélar og vefstólar – Pascal, Leibniz og Jacquard

Fyrstu reiknivélarnar voru búnar til á 17. öld. Meðal frægra reiknivélasmiða frá þeim tíma má nefna Frakkann *Blaise Pascal* (1623 – 1662) og Þjóðverjann *Gottfried Wilhelm Leibniz* (1646 – 1716). Reiknivélar sem *Leibniz* smíðaði gátu lagt saman, dregið frá, margfaldað og deilt. En hann lét sig dreyma um að smíða vél sem gæti meira en þetta og setti fram kenningar um að vinna mætti hvers kyns útreikninga og margt annað hugarstarf með vélrænum aðferðum. Þessar bollaleggingar Leibniz minna um margt á

hugmyndir og kenningar í tölvufræðum nútímans. Þær vöktu því miður litla athygli fyrr en í byrjun þessarar aldar, enda áttu fyrri tíðar menn bágt með að gera sér grein fyrir mikilvægi þeirra.

Á sautjándu og átjándu öld þróaðist ýmiss konar véltækni og um aldamótin 1800 fann Frakkinn *Joseph M. Jacquard* (1752 –1834) upp forritanlega vefstóla. Vefstólar þessir gátu ofið eftir forskriftum sem þeim voru gefnar á gataspjöldum. Hægt var að mata þá á nánast hvaða munstri sem er með því að hengja mörg gataspjöld saman í keðju. Þegar keðjan hafði snúist heilan hring endurtók munstrið sig.

The Analytical Engine – Babbage og Ada Lovelace

Í byrjun 19. aldar voru komnar fram forritanlegar vélar og reiknivélar. En enginn hafði reynt að smíða forritanlega reiknivél, þótt Leibniz hafi látið sig dreyma um slíkan grip. Sá sem fyrstur gerði tilraun til þess að smíða forritanlega reiknivél var Englendingurinn *Charles Babbage* (1792 –1871). Hann hannaði og reyndi að smíða vél sem hægt væri að mata á reikniformúlum á svipaðan hátt og vefstólar *Jacquard* voru mataðir á munstrum. Vél þessi átti að vefa munstur algebrunnar eins og vinkona *Babbage* og samstarfsmaður, *Ada Lovelace* (1815 – 1852), komst að orði.

Vélin sem Babbage reyndi að smíða var lík tölvum nútímans að því leyti að hún átti að vera forritanleg. Að vél sé forritanleg þýðir að hægt sé að búa til runu af skipunum og afhenda henni og láta hana svo sjálfa um að vinna eftir þeim. Forritanlegar vélar „skilja“ einhvers konar mál sem hægt er að nota til þess að orða uppskriftir eða forrit handa þeim til að vinna eftir. Venjuleg reiknivél er ekki forritanleg, heldur tekur aðeins við einni skipun í senn, framkvæmir hana og bíður svo eftir næstu skipun. Tölvur eru forritanlegar en það eru til fleiri forritanlegar vélar en tölvur, eins og til dæmis sjálfvirkir vefstólar. Sérstaða tölvanna er fólgin í því að málin sem þær „skilja“ duga til þess að orða hvaða reiknireglu eða stærðfræðilega aðferð sem er.

Babbage taldi að mögulegt yrði að forrita vélina sína til þess að reikna hvaða reiknisdæmi sem er. Hann gat þó ekki fært nein viðhlítandi rök fyrir þessu. Það var ekki fyrr en á þessari öld sem *Alan Turing* færði rök að því að hægt sé að smíða vél sem forrita má til þess að reikna eftir öllum mögulegum aðferðum.

Babbage tókst aldrei að smíða þessa vél. Á hans tíma var rafeindatækni nær óþekkt og hann ætlaði að búa vélina til úr tannhjólum, öxlum og álíka málmhlutum. Það er ekki útilokað að smíða tölvu úr tannhjólum. Mönnum tókst þó ekki að smíða nothæfar tölvur fyrr en þeir höfðu náð tökum á rafeindtækni, enda væri tölva úr tannhjólum og öðrum hreyfanlegum hlutum ansi flókin smíð og auk þess rándýr, þung og seinvirk.

Babbage kallaði vélina sína *The Analytical Engine* og þótt hún væri aldrei smíðuð þá voru samin nokkur forrit fyrir hana. Af þeim sem unnu með Babbage að gerð forrita fyrir vélina er *Ada Lovelace* þekktust. Stundum hefur hún verið kölluð fyrsti forritarinn. Það var um 1830 sem Babbage og Ada Lovelace unnu þessi fyrstu afrek á sviði tölvufræði. En fæstir tóku mark á þeim og því fór með verk þeirra, eins og bollaleggingar heimspekingsins *Leibniz*, að þau gleymdust þar til á þessari öld.

Talningarvélar – Hollerith, IBM

Næsta mikilvæga skref í þróun reiknivéla og upplýsingatækni var tekið í Bandaríkjunum á síðasta áratug 19. aldar af verkfræðingi sem hét *Hermann Hollerith* (1860 – 1929). Hann smíðaði vélar til að vinna úr manntali sem gert var í Bandaríkjunum árið

1890. Vélar Holleriths notuðu gataspjöld svipuð þeim sem notuð voru til að forrita vefstóla *Jacquards*. Hvert spjald geymdi upplýsingar um einn einstakling þar sem trúflokkur, þjóðerni og fleiri eiginleikar voru táknaðir með mynstri úr götum. Til að vinna úr manntalinu, telja t.d. hvað margir íbúar einhvers svæðis voru grísk-kaþólskir og af rússneskum uppruna, var stafli af spjöldum fyrir allan hópinn settur í velina og hún látin fletta í gegnum þau og telja hve oft tiltekin munstur komu fyrir. Vélar Holleriths gengu fyrir rafmagni og þær mörkuðu þáttaskil í vélrænni úrvinnslu upplýsinga.

Árið 1886 stofnaði Hollerith fyrirtæki til að framleiða og markaðssetja talningarvélar sínar. Árið 1924 sameinaðist þetta fyrirtæki tveim öðrum og myndaði með þeim fyrirtækið *International Business Machines Corporation*, sem yfirleitt er kallað *IBM*. Þetta fyrirtæki hefur frá upphafi gegnt forystu í framleiðslu skrifstofuvéla og reiknitækja, meðal annars var það fyrst til að hefja framleiðslu og sölu á rafmagnsritvélum á árunum upp úr 1920 og það átti mikinn þátt í þróun og smíði fyrstu tölvanna. *IBM* á dótturfyrirtæki í mörgum löndum. *IBM á Íslandi* tók til starfa árið 1967.

Zuse – Aiken

Á árunum milli 1930 og 1945 voru gerðar nokkrar tilraunir til að smíða sjálfvirkar eða forritanlegar reiknivélar. Fyrstu forritanlegu reiknivélarnar sem byggðu á notkun tvíundakerfis voru smíðaðar af Þjóðverjanum *Konrad Zuse* á árunum 1936 til 1938. Reikniverki þeirra svipaði um margt til gjörva í tölvu en þær voru forritaðar með því að breyta tengingum. Minnið geymdi aðeins gögn, ekki forrit.

Uppfinningar Zuse vísuðu veginn til tölvutækni nútímans en á þessum árum voru þýsk vísindi að einangrast og vélarnar sem Zuse smíðaði eyðilögðust í loftárásum bandamanna á Hamborg árið 1943. Þó Konrads Zuse sé nú minnst sem eins af helstu upphafsmönnum tölvutækninnar höfðu uppgötvanir hans lítil áhrif á þróun tölva og reiknitækja á árunum eftir heimstyrjöldina síðari.

Af amerískum reiknivélasmiðum á fjórða áratugnum er *Howard Aiken*, prófessor í stærðfræði við Harvard háskóla, með þeim frægustu. Hann hafði kynnt sér hugmyndir Babbage og Ada Lovelace og hannaði vél sem hafði svipaða eiginleika og *The Analytical Engine* en gekk fyrir rafmagni. Hann gerði stjórnendum *IBM* grein fyrir hugmyndum sínum og þeir samþykktu að verja milljón dölum til að smíða vélina. Vélin var fullgerð 1944 og hlaut nafnið *Mark I*. Hún var um 5 tonn að þyngd og sagan segir að það hafi þurft nokkur tonn af ís á dag til að kæla hana. *Mark I* var millistig milli gamaldags vélar úr tannhjólum, öxlum og öðrum hreyfanlegum hlutum og rafeindatækis. En það styttist í að menn smíðuðu reiknitæki sem byggðu algerlega á rafeindatækni.

ENIAC – Atanasoff, Mauchly

Um svipað leyti og Howard Aiken vann að hugmyndum sínum setti *John V. Atanasoff*, prófessor í eðlisfræði við fylkisháskólann í Iowa, fram hugmyndir um forritanlega reiknivél sem ynni stafrænt, notaði tvíundakerfi og byggði algerlega á rafeindatækni. Þessum hugmyndum kynntist eðlisfræðingurinn *John Mauchly* sem vann að smíði reiknivéla fyrir bandaríska herinn og hann og félagar hans notuðu þær til að smíða vél sem hlaut nafnið *Electronically Numerical Integrator and Calculator* eða *ENIAC*. Hún var gangsett árið 1945 við fylkisháskólann í Pennsylvaníu.

ENIAC vó 30 tonn, fyllti um 180 fermetra gólfflöt, notaði 174 kílówött af rafafli og var samsett úr meira en 18.000 útvarpslömpum og mörgum kílómetrum af vír. Hún gat margfaldað saman tvær tölur á 2,8 millisekúndum, eða með öðrum orðum reiknað meira en 300 margföldunardæmi á sekúndu. En þótt hún hafi verið á stærð við hús og þetta hraðvirk þá var hún ófær um sumt af því sem vasareiknivélar nútímans geta. *ENIAC* var að því leyti eins og vélar Zuse að hún gat ekki geymt forrit í minni, eins og tölvur sem nú eru í notkun gera, heldur var hún forrituð með því að breyta tengingum eða stöðu rofa í henni.

1.2 Fyrstu tölvurnar

Turing og von Neumann

Hér hafa verið nefndir til sögu uppfinningamenn eins og *Hollerith*, *Zuse*, *Aiken* og *Atanasoff* sem áttu upphaf að merkum nýjungum á sviði reikni- og upplýsingatækni. Enn eru þó ótaldir merkustu hugmyndasmiðirnir við upphaf tölvualdar, þeir *Alan Turing* (1912 – 1954) og *John von Neumann* (1903 – 1957)

Englendingurinn *Alan Turing* var menntaður í stærðfræði. Á árunum upp úr 1930 vann hann það afrek að skýra hvernig hægt er að byggja alls konar útreikninga og reikniaðferðir úr aðgerðum sem eru svo einfaldar að vél geti framkvæmt þær. Turing gerði nákvæma grein fyrir því hverjar þessar einföldu aðgerðir þyrftu að vera og lýsti ímynduðum vélum sem gætu unnið eftir þeim. Þessar ímynduðu vélar eru nefndar eftir höfundi sínum og kallaðar *Turingvélar*.

Ásamt starfsbróður sínum, *Alonso Church* (1903 – 1937), setti Turing fram þá kenningu að hægt sé að forrita Turingvél til þess að vinna eftir hvaða stærðfræðilegri aðferð sem er. Þessi kenning gengur yfirleitt undir nafninu *tilgáta Church og Turing*. Hún hefur aldrei verið sönnuð, en allar rannsóknir sem gerðar hafa verið síðan benda til þess að hún sé rétt og nú til dags draga hana fáir í efa. Af þessari kenningu leiðir meðal annars að ef Babbage hefði tekist að smíða The Analytical Engine þá hefði verið mögulegt að forrita hana til að vinna hvaða útreikninga sem er, því Turingvélar eru bæði jafngildar vél Babbage og öllum tölvum sem smíðaðar hafa verið að því leyti að hægt er að láta þessar vélar vinna sömu útreikninga.

Turing gafst ekki tóm til þess að fullmóta hugmyndir sínar um smíði forritanlegra reiknivéla fyrr en síðari heimsstyrjöldinni lauk. Í stríðinu vann hann fyrir ensku leyniþjónustuna. Verkefni hans var að þýða dulmálsskeyti Þjóðverja. Tæknin sem hann þróaði til þess var langt á undan sinni samtíð og olli byltingu bæði í dulmálsfræðum og smíði véla til þess að vinna flókna útreikninga. Þessi tækni nýttist honum síðar þegar hann tók til við að hanna raunverulegar tölvur.

Þegar hér var komið sögu hafði ungverski stærðfræðingurinn *John von Neumann* bætt ýmsu við kenningar Turings og mótað hugmyndir um hvernig best væri að hanna og smíða tölvu. Hann var gyðingur, fæddur og uppalinn í Búdapest. Eins og fjölmargir aðrir menntamenn af gyðingaættum flutti hann til Bandaríkjanna þegar nazistar og aðrir þjóðernissinnaðir öfgahópar fóru að hafa veruleg áhrif á evrópsk stjórnmál á 4. áratugnum.

John von Neumann fylgdist með smíði *Mark I*. Sú vél hafði minni til að geyma tölur og niðurstöður útreikninga en forritin sem hún vann eftir voru ekki geymd í minni heldur voru þau „skráð“ í vélina með því að breyta tengingum og stöðu færanlegra vélarhluta. Ein af merkustu hugmyndum von Neumann var að láta forritin vera í minninu eins og gögnin.

MADM, EDSAC, UNIVAC

Á árunum 1948 til 1950 voru smíðaðar nokkrar vélar sem byggðu á hugmyndum von Neumann um að hafa forritin í minni eins og gögnin sem þau unnu með. Ein sú fyrsta var smíðuð á Englandi við háskólann í Manchester. Hún hlaut nafnið *Manchester Automatic Digital Machine*, eða *MADM*. Meðal þeirra sem unnu að gerð hennar var Alan Turing. Hún komst í gang árið 1948. Árið eftir var gangsett í Bandaríkjunum tölva sem byggð var eftir forskrift von Neumann og hét *EDSAC*.

Árið 1950 voru í gangi um það bil 20 tölvur og forritanlegar reiknivélar í Bandaríkjunum. Engar tvær þessara véla voru eins, fjöldaframleiðsla á tölvum var ekki hafin. Fyrstu tölvurnar sem voru framleiddar og seldar í mörgum eintökum komu á markað 1951. Frægasta vörumerkið frá því ári er *UNIVAC I* sem framleidd var af *The Eckert-Mauchly Computer Corporation*. Annars af stofnendum þess fyrirtækis, *Mauchly*, hefur áður verið getið. UNIVAC I var gerð úr 5.000 útvarpslömpum, vélin var um $20m^2$ að flatarmáli og meira en 2m há og um 5 tonn að þyngd. Hún kostaði 250.000 dali.

Þessar fyrstu tölvur voru næsta frumstæðar á mælikvarða nútímans. Þær voru að mestu smíðaðar úr þúsundum útvarpslampa og mörgum kílómetrum af vír. Þær höfðu hvorki disklingadrif né skjái og yfirleitt ekkert af þeim jaðartækjum sem nú þykja sjálfsögð. Forrit og gögn voru sett inn í þær með frumstæðum aðferðum á borð við þá að gata pappírsstrimla og renna þeim í gegnum „lesara“ sem voru tengdir tölvunum. Fljótlega var þó farið að nota segulbönd í svipuðum tilgangi og disklingadrif eru notuð nú til dags. Um 1955 kom fram tæki sem svipar til harðra diska eins og nú eru notaðir. Disklingadrif þróuðust á 7. áratugnum. Prentarar urðu til fljótlega eftir 1950. Tölvuskjáir komu líka fram á fyrri hluta 6. áratugarins en notkun þeirra varð þó ekki algeng fyrr en seinna.

Fyrstu tölvurnar voru mest notaðar við talnareikning, einkum útreikninga fyrir heri Bretlands og Bandaríkjanna. Áður en smíði tölva hófst hafði *Alan Turing* þó gert grein fyrir ótæmandi möguleikum tölvanna, að þær gætu unnið hvers kyns hugarstarf. Frá upphafi tölvualdar hefur fjölbreytni þeirra verka sem tölvur vinna verið að aukast og nú til dags eru þær notaðar við ótal margt annað en reikning.

1.3 Þróun tölvutækni frá 1948 til nútímans

Fyrsta kynslóð (1948 – 1955) og önnur kynslóð (1955 – 1965)

Tölvur af *fyrstu kynslóð*, sem smíðaðar voru fyrir 1955, voru að mestu gerðar úr útvarpslömpum. Síðan hefur tæknin einkum þróast með þeim hætti að hlutarnir sem tölvur eru búnar til úr hafa orðið minni, hraðvirkari, ódýrari, öruggari (það er ólíklegri til að bila) og sparneytnari á rafmagn. Þessi þróun hefur tekið þrjú stökk eftir að fyrstu tölvurnar voru smíðaðar. Þess vegna er talað um fjórar kynslóðir tölva.

Árið 1947 var *smárinn* (transistorinn) fundinn upp. Smári vinnur sama verk og útvarpslampi en er margfalt minni, ódýrari, sparneytnari og endingarbetri. Árið 1955 var tekið að nota smára til að smíða tölvur. Þá varð *önnur kynslóð* tölva til. Við þetta minnkuðu tölvurnar talsvert og lækkuðu í verði og tekið var að nota þær við gagnavinnslu og útreikninga í ýmsum greinum atvinnulífs. Tölvur af annarri kynslóð áttu sitt blómaskeið á árunum milli 1955 til 1965. Á þessu tímabili má segja að tölvutæknin hafi slitið barnskónum. Fyrstu forritunarmálin voru búin til og í kringum 1960 komu fram frumstæð stýrikerfi. Á fyrri hluta 7. áratugarins voru til stýrikerfi sem gátu stjórnað víxlvinnslu, þ.e. skipt tölvu milli notenda svo fleiri en einn gætu notað hana á sama tíma.

Undir lok þessa tímabils, árið 1964, hófst tölvuvæðing hér á landi því *Háskóli Íslands* og *Skýrsluvélar ríkisins og Reykjavíkurborgar* eignuðust sínar fyrstu tölvur það ár. Þær kostuðu hvor um sig eins og einbýlishús og þóttu hin mestu galdratæki. Nú fást margfalt öflugri tölvur fyrir nokkur þúsund krónur.

Þriðja kynslóð (1965 – 1975) og fjórða kynslóð (1975 – ?)

Snemma á 7. áratugnum var tekið að smíða samrásir úr kísli. Slík rás inniheldur fjölda smára í einni heilsteyptri einingu. Tölvur sem smíðaðar eru úr samrásum teljast til *þriðju kynslóðar*. Með tilkomu samrásanna hófu tölvur fyrir alvöru innreið sína í atvinnulífið. Samrásatölvur voru þó býsna dýrar og stórskornar miðað við þær tölvur sem nú eru notaðar.

Tölvum af 3. kynslóð fylgdu yfirleitt stýrikerfi sem réðu við víxlvinnslu og gátu þjónað mörgum notendum í senn. Af stýrikerfum frá þessum tíma er *UNIX* einna þekktast. *UNIX* var búið til við *Bell* símafyrirtækið í Bandaríkjunum um 1970 og hefur verið að þróast síðan og á enn vaxandi vinsældum að fagna.

Þegar kom fram á miðjan 8. áratuginn höfðu samrásir þróast þannig að hægt var að koma þúsundum smára fyrir á einni kísilflögu sem var minni en sykurmoli. Þegar farið var að smíða örgjörva, þ.e. setja heila gjörva á einstakar kísilflögur, tók *fjórða kynslóð* tölva við af þeirri þriðju. Nú (árið 2001) eru til kísilflögur sem eru minni en krónupeningur og hafa brenndar í sig margar milljónir smára ásamt tilheyrandi tengingum. Vélarhlutar, sem áður voru samsettir úr mörgum pörtum og fylltu heil herbergi, eru nú fólgnir í einni lítilli kísilflögu.

Lögmál Moores og fjölgun tölva til 1980

Árið 1965 setti *Gordon Moore*, sem síðar stofnaði fyrirtækið *Intel*, fram þá tilgátu að fjöldi smára í kísilflögu mundi tvöfaldast á hverjum 18 mánuðum. Þessi tilgáta er stundum kölluð *lögmál Moores*. Ekki er fjarri lagi að hún hafi staðist til þessa og ekkert bendir til að það hægi á þróuninni á næstu árum.

Í vissum skilningi geta allar tölvur gert það sama, þær geta unnið eftir hvaða stærðfræðilegri aðferð sem er. En hraðvirkar nútímatölvur bjóða samt upp á ýmsa möguleika umfram hægvirkari tölvur fyrri ára. Þær geta til dæmis sýnt flóknar hreyfimyndir á eðlilegum hraða sem eldri tölvur hefðu verið óratíma að mjaka um skjáinn. Sem tölvur hafa orðið ódýrari, smávaxnari og hraðvirkari hefur notkunarsviðum þeirra og notkunarmöguleikum fjölgað og eftirspurn eftir þeim aukist. Um 1950 skiptu tölvur í heiminum nokkrum tugum og fáum datt í hug að nokkurn tíma yrði þörf fyrir mörg þúsund, hvað þá milljónir tölva. En 1960 voru til meira en 1.000 tölvur. Árið 1970 var

tala þeirra komin yfir 100.000 og 1980 voru meira en 1.000.000 tölvur í notkun í heiminum og síðan þá hefur tala þeirra margfaldast.

Helstu flokkar tölva við lok 20. aldar

Þegar farið var að fjöldaframleiða ódýrar kísilflögur sköpuðust möguleikar á smíði ódýrra smátölva bæði til að tengja við skjái og lyklaborð og nota sem skrifstofuvélar eða leiktæki og til að stjórna ýmsum tækjabúnaði eins og þvottavélum, hemlakerfum bíla, myndbandstækjum, geislaspilurum og örbylgjuofnum. Nú til dags eru tölvur út um allt. Helstu flokkar eru taldir upp í eftirfarandi töflu. Rétt er að taka tölum um verð með hæfilegum fyrirvara.

	Verð	Dæmigerð notkun
Iðntölva	10^3 kr.	Stjórna sjálfvirkum vélum t.d. þvottavél eða hemlakerfi í bíl.
Leikjatölva	10^4 kr.	Tengd við sjónvarp á heimili til að leika tölvuleiki.
Einmenningstölva	10^5 kr.	Skrifstofuvinna, nám, vefráp, tölvuleikir.
Netþjónn/Vinnustöð	10^6 kr.	Miðlari á staðarneti, vefþjónn á Interneti, tölvustudd hönnun.
Miðlungstölva	10^7 kr.	Samnota tölva í meðalstóru fyrirtæki.
Stórtölva	10^8 kr.	Gagnavinnsla í stórum bönkum.
Ofurtölva	10^9 kr.	Langtíma veðurspár, vísindarannsóknir.

1.4 Þróun einmenningstölva frá 1970 til 1999[1]

Á 8. áratugnum var algengt að tengja tölvu nokkrum skjám og lyklaborðum. Þetta var vegna þess að ekki þótti vit að láta einn mann sitja að svo dýrri vél sem tölvu. Nú orðið er svo ódýrt að smíða litlar tölvur að lítill eða enginn sparnaður er að því að tengja þær mörgum útstöðvum, þess vegna eru langflestar tölvur, sem nú eru notaðar við skrifstofustörf, einmenningstölvur. Á undanförnum áratugum hefur þróun þeirra og útbreiðsla verið kraftaverki líkust.

1970 – 1974

Árið 1970 komu örgjörvar fyrst á almennan markað. Einn sá fyrsti hét *Intel 4004* og var smíðaður af fyrirtækinu *Intel* sem enn þann dag í dag er stærsti örgjörvaframleiðandi í heimi. Árið áður hafði *Intel* kynnt minniskubba sem voru heilt kílóbæti að stærð. *Intel 4004* gekk á hraðanum 0,1 megarið og gat framkvæmt um 60.000 aðgerðir á sekúndu. Hann innihélt 2300 smára og gat nýtt 640 bæta minni.

Árið 1972 hóf *Intel* svo framleiðslu á *Intel 8008* örgjörvanum. Hann innihélt 3500 smára, hafði 8 bita gisti, gekk á hraðanum 0,2 megarið og gat nýtt allt að 16 kílóbæta minni. Tveim árum seinna, 1974, kynnti *Intel* enn einn örgjörva *Intel 8080*. Sama ár hóf fyrirtækið *Motorola* framleiðslu á örgjörvanum *Mororola 6800*. Næstu 20 árin var *Motorola* helsti keppinautur *Intel* í framleiðslu á örgjörvum.

[1] Öll útlend fyrirtæki sem nefnd eru í þessum kafla starfa í Bandaríkjunum nema annað sé tekið fram. Þegar talað er um verð í dölum er átt við Bandaríkjadali.

Með tilkomu fjöldaframleiddra örgjörva og minniseininga opnuðust möguleikar á smíði ódýrra tölva. 1973 hófst framleiðsla á fyrstu einmenningstölvunni, *Scelbi-8H*. Hún notaði *Intel 8008* örgjörvann og hafði 1 kílóbætis minni (stækkanlegt í 16 kílóbæti) og kostaði um 600 dali. Árið eftir kynnti fyrirtækið *Popular Electronics* einmenningstölvuna *Altair 8800*. Hún hafði örgjörva af gerðinni *Intel 8080* og kostaði um 450 dali.

Þessar fyrstu einmenningstölvur voru nær eingöngu keyptar af áhugamönnum um tölvu- og rafeindatækni. Þeim fylgdu engin forrit, notendur urðu að mata þær á vélamálsskipunum með frumstæðum inntakstækjum. En þetta stóð til bóta því um þetta leyti bjó *Gary Kildall* til stýrikerfi fyrir einmenningstölvur. Það hlaut nafnið *CP/M*.

1975 – 1979

Á þessu tímabili kemst verulegur skriður á þróun einmenningstölvanna. 1975 hefst framleiðsla á örgjörvunum *Z80* frá *Zilog* og *6502* frá *MOS Technology*. Þessi örgjörvar urðu ríkjandi í ódýrum tölvum fram á 9. áratuginn.

Árið 1979 kynnti *Intel* örgjörvann *Intel 8088*. Hann gekk á 4,77 megariðum, hafði 16 bita gisti en aðeins 8 bita gagnabraut og innihélt 29.000 smára. *Intel 8088* var notaður í fyrstu *pésana* (*PC-tölvurnar*) þegar framleiðsla þeirra hófst 1981. Sama ár kom *Motorola 68000* örgjörvinn frá *Motorola* sem var notaður í fyrstu *Macintosh* tölvurnar þegar framleiðsla þeirra hófst árið 1984. *Motorola 68000* dró nafn sitt af því að í honum voru 68000 smárar. Hann hafði 16 bita gisti og 16 bita gagnabraut.

Á þessu tímabili komu fram ódýrar tölvur sem hægt var að nota til að vinna skrifstofuvinnu og leika tölvuleiki. Mörg öflugustu tölvu- og hugbúnaðarfyrirtæki heims voru stofnuð á þessu tímabili. Þau frægustu eru líklega *Apple Computer* sem *Steve Jobs* og *Steve Wozniac* stofnuðu árið 1976 og *Microsoft* sem *Bill Gates* og *Paul Allen* stofnuðu 1977.

Árið 1977 hófst framleiðsla á fyrstu einmenningstölvunum sem náðu verulegri útbreiðslu. Þetta voru *Commodore Pet* og *Apple II*. Þær voru báðar með örgjörva af tegundinni *6502* frá *MOS Technology*.

Commodore Pet var framleidd af *Commodore Business Machines*. Hún kostaði um 600 dali með 4 kílóbæta ritminni, 14 kílóbæta lesminni, lyklaborði, skjá og segulbandsstöð (kasettutæki) sem gegndi svipuðu hlutverki og disklingadrif gerðu síðar.

Apple II var með 16 kílóbæta ritminni, 16 kílóbæta lesminni, lyklaborð, skjá og tengi fyrir segulband og kostaði um það bil 1.300 dali út úr búð í Bandaríkjunum. Þessi vél náði feikilegum vinsældum og sem árin liðu komu öflugri og fullkomnari útgáfur af henni. Þegar *Apple Computer* hætti loks framleiðslu *Apple II* tölva árið 1993 höfðu selst um 5 milljónir véla.

Á þessu tímabili varð til hugbúnaður sem gerði tölvur að nýtilegum verkfærum fyrir fleiri en tæknimenn og grúskara. Þegar árið 1975 bjuggu stofnendur *Microsoft* þeir *Bill Gates* og *Paul Allen* til *BASIC* túlk fyrir *Altair 8800*. Þetta var fyrsta forritunarmálið fyrir einmenningstölvur og fram á 9. áratuginn fylgdi *BASIC* túlkur með flestum einmenningstölvum sem seldar voru. Í sumum tilvikum var hann innbyggður í lesminni tölvunnar og gegndi að nokkru leyti hlutverki stýrikerfis. Forritin sem hafa fylgt einmenningstölvum alla tíð síðan komu svo fram eitt af öðru: 1976 bjó *Michael Shrayer* til *Electric Pencil,* fyrsta ritvinnsluforritið fyrir einmenningstölvur; 1978 skrifuðu *Dan Bricklin* og *Bob Frankston* fyrsta töflureiknin, *VisiCalc*; 1979 kom svo *WordStar*

ritvinnsluforritið frá *Micropro* og það sama ár sló tölvuleikurinn *PacMan,* eftir Japanann *Toru Iwatani*, í gegn.

Til þessa tímabils má einnig rekja upphaf margra jaðartækja sem náðu útbreiðslu á 9. áratugnum. 1978 hóf *Epson* framleiðslu á ódýrum nálaprenturum fyrir einmenningstölvur og það sama ár hóf *Apple Computer* framleiðslu á disklingadrifum fyrir *Apple II* vélar og á næstu árum tóku disklingar við af segulbandsspólum sem gagnageymslur fyrir smátölvur. Árið eftir, 1979, byrjaði fyrirtækið *Hayes* að framleiða mótöld fyrir einmenningstölvur.

Á þessu tímabili voru einmenningstölvur ekki notaðar að ráði í atvinnulífi en tæknin sem átti eftir að gerbreyta allri skrifstofuvinnu á næstu 20 árum mótaðist að miklu leyti. Það fóru líka að fást tölvutímarit í venjulegum bókabúðum. Eitt það merkasta var *Byte* sem hóf göngu sína árið 1975.

Þegar á árunum í kringum 1970 varð fyrirrennari *Internetsins*, *ARPA* (*Advanced Research Projects Agency*) tölvunetið til við Kaliforníuháskóla. Það þjónaði einkum þörfum bandaríska hersins og á tímabilinu milli 1975 og 1980 þróaði bandaríska varnarmálaráðuneytið samskiptastaðalinn *TCP/IP*. Á þeim tíma datt líklega fáum í hug að þessi samskiptastaðall yrði notaður á tölvuneti sem næði inn á milljónir heimila um allan heim fyrir aldamót.

1980 – 1984

Örgjörvar héldu áfram að þróast. Árið 1982 kom *Intel 80286*. Hann hafði 16 bita gisti og 16 bita gagnabraut og gat notað allt að 16 megabæta minni.

Á þessum tíma komu fram tvær gerðir einmenningstölva sem höfðu mikil áhrif á þróunina: *IBM PC* og *Apple Macintosh*.

Fyrsta *PC-tölvan*, *IBM-PC*, var sett á markað af *IBM* árið 1981. Hún kostaði um það bil 3.000 dali, hafði *Intel 8088* örgjörva, 64 kílóbæta ritminni, eitt 5¼ tommu disklingadrif, lyklaborð og skjá. Henni fylgdi stýrikerfi frá fyrirtækinu *Microsoft* sem gekk undir nafninu *PC DOS* ef það var keypt í umbúðum frá *IMB* en *MS DOS* ef það var selt af *Microsoft*. Með tilkomu *IBM PC* urðu þáttaskil í sögu einmenningstölvanna og notkun þeirra í atvinnulífi hófst fyrir alvöru. Alla tíð síðan hafa *PC-tölvur* verið að þróast. Árið 1983 hóf *IBM* sölu á *PC-tölvu* með hörðum diski. Sú vél var kölluð *IBM XT*. Diskurinn í henni var 10 megabæti. Þessi vél hafði 128 kílóbæta minni og kostaði 5.000 dali. 1984 komu svo *PC-tölvur* með Intel 80286 örgörva frá *IBM*. Þær voru kallaðar *IBM AT*. Slík vél með 256 kílóbæta minni, disklingadrifi, lyklaborði og skjá kostaði um það bil 5.500 dali.

IBM birti allar upplýsingar um gerð *PC-vélanna* og leyfði öðrum fyrirtækjum að smíða eftirlíkingar af þeim. Framleiðsla á slíkum eftirlíkingum hófst þegar árið 1982.

Fyrsta *Macintosh* tölvan frá *Apple Computer* kom á markað 1984. *Macintosh* með 8 megariða *Motorola 68000* örgjörva og 128 kílóbæta minni, disklingadrifi, lyklaborði og skjá kostaði um það bil 2.500 dali. Vélinni fylgdi mús og stýrikerfi með myndrænum notendaskilum. Með því urðu þáttaskil í gerð stýrikerfa fyrir einmenningstölvur.

Meðal annarra merkilegra tölva frá þessu tímabili má telja bresku vélina *Sinclair ZX80* frá 1980. Hún var fyrsta tölvan sem seld var fyrir minna en 200 dali. Þessi vél hafði *Z80* örgjörva, 1 kílóbætis ritminni, 4 kílóbæta lesminni, lyklaborð og tengi fyrir segulband og sjónvarpsskjá. Hún var að ýmsu leyti fyrirrennari leikjatölva sem nú eru tengdar við sjónvarp á mörgum heimilum. Einnig er vert að nefna *Commodore VIC 20*

frá 1981, með *M6502* örgjörva og 5 kílóbæta ritminni (stækkanlegt í 32 kílóbæti). Hún var fyrsta tölvan sem seldist í meira en milljón eintökum.

Ein af merkustu nýjungum þessa tíma var harðir diskar fyrir einmenningstölvur. Sá fyrsti kom á markað árið 1980. Hann var framleiddur af fyrirtækinu *Seagate*, rúmaði 5 megabæti og kostaði 600 dali.

Í byrjun 9. áratugsins voru stýrikerfi fyrir einmenningstölvur heldur frumstæð. Fyrsta útgáfa *MS-DOS* frá *Microsoft* gat t.d. ekki skipt gagnageymslum í skráasöfn. Úr þessu var bætt með útgáfu 2 sem kom 1983. Hún var gerð fyrir harða diska. Forrit voru yfirleitt keyrð frá skipanalínu og stjórnað með því að slá skipanir á lyklaborð. Þetta breyttist 1984 með *Macintosh* stýrikerfinu sem hlýtur að teljast mikilvægasta nýjungin í hugbúnaðargerð frá þessu tímabili. Gerð þess byggði að miklu leyti á rannsókna- og tilraunastarfi á rannsóknastöð fyrirtækisins *Xerox* í Palo Alto í Kaliforníu.

Með merkustu hugbúnaðarpökkum frá fyrri hluta 9. áratugarins má nefna gagnagrunnin *dBASE II* sem fyrirtækið *Ashton Tate* hóf að selja 1981. Árið 1982 kom á markað töflureiknirinn *Lotus 1-2-3* frá fyrirtækinu *Lotus*. Það sama ár kom líka fyrsta útgáfa *AutoCad* hönnunarforritsins frá *Autodesk*. 1983 kynnti *Microsoft* ritvinnsluforritið *Microsoft Word 1.0* fyrir *MS DOS*. Með því var hægt að nota nýtt inntakstæki frá fyrirtækinu, mús. Árið eftir sendi *Microsoft* svo frá sér töflureikninn *Multiplan*. Þá hóf *Satellite Software* sölu á helsta keppinaut *Microsoft Word* um árabil, ritvinnsluforritinu *Word Perfect*.

Á árunum milli 1980 og 1984 hófst notkun einmenningstölva í atvinnulífi fyrir alvöru. Ritvinnsluforrit, töflureiknar og gagnagrunnar urðu sjálfsögð verkfæri við ýmsa skrifstofuvinnu. Vélarnar voru yfirleitt ekki nettengdar en þróun í þá átt var að hefjast. Árið 1981 kynnti *Xerox* fyrirtækið *Ethernet* staðarnet og fyrsta útgáfa af *NetWare* stýrikerfi fyrir netþjóna frá *Novell* kom 1983.

Á þessum árum varð tölvutæknin almenningseign. 1984 áttu 15 milljónir Bandaríkjamanna tölvu og þá voru tölvuforrit í fyrsta sinn auglýst í sjónvarpi þar í landi.

1985 – 1989

Í upphafi þessa tímbils, árið 1985, kynnti *Intel* nýjan örgjörva, *Intel 80386*. Hann gekk á 16 megariðum, hafði 32 bita gisti og 32 bita gagnabraut, gat nýtt 4 gígabæta minni og innihélt um 275.000 smára. Við lok tímabilsins, árið 1989, kom svo *Intel 80486*. Hann var 32 bita eins og fyrirrennari hans en hafði innbyggt reikniverk fyrir kommutölureikning og 8 kílóbæta flýtiminni. *Intel* 80486 innihélt meira en milljón smára og gekk á 25 megariða hraða.

Á þessu tímabili komu líka nýir örgjörvar frá *Motorola, 68030* árið 1987 og fyrstu *68040* örgjörvarnir frá *Motorola* voru smíðaðir árið 1989 og settir á markað árið eftir. Þessir örgjörvar voru báðir 32 bita eins og nýju örgjörvarnir frá *Intel* og sá síðarnefndi innihélt meira en milljón smára.

Með tilkomu þessara nýju örgjörva opnuðust möguleikar á að smíða hraðvirkar tölvur sem réðu við myndræn notendaskil og margmiðlun. Þegar árið *1986* komu á markað *PC*-tölvur með 16 megariða *Intel 80386* örgjörva. Árið 1988 kynnti *Steve Jobs*, annar af stofnendum *Apple Computer*, tölvuna *NeXT* sem þótti bera af öðrum einmenningstölvum með 25 megariða *Motorola 68030* örgjörva, 8 megabæta minni og 17 þumlunga skjá. Þessi vél náði ekki verulegri útbreiðslu en möguleikar hennar á sviði margmiðlunar vísuðu veginn fram á 10. áratuginn. Önnur tölva sem kom fram á þessum

tíma og hafði mikil áhrif var *Sun 4* vinnustöðin sem *Sun Microsystems* hóf að framleiða árið 1987. Henni fylgdi stýrikerfið *UNIX*.

Undir lok þessa tímabils jókst mjög framleiðsla á fartölvum. Árið 1989 voru komnar á markað vélar sem keyðu *MS DOS* og vógu minna en hálft kílógramm. Það ár hóf *Apple Computer* líka sölu á *Macintosh* fartölvum.

Ein af merkari nýjungum þessa tíma voru geisladrif (*CD-ROM*) fyrir tölvur sem komu fram árið 1985.

Þegar kom fram á seinni hluta 9. áratugsins varð ljóst að *MS-DOS* stýrikerfið svaraði ekki lengur kalli tímans. Notendaskil þess voru heldur dapurleg, bara blikkandi bendill í skipanalínu. Það gat aðeins keyrt eitt forrit í senn og réð ekki við að úthluta forritum meira en 640 kílóbæta minni. *Microsoft* reyndi að bæta úr þessum annmörkum með *Microsoft Windows 1.0* árið 1985. Þessi fyrsta útgáfa *Windows* var í raun lítið annað en myndræn notendaskil fyrir *MS DOS*.

Árið 1988 sendu *Microsoft* og *IBM* frá sér nýtt stýrikerfi, *OS/*2, sem var að ýmsu leyti svipað stýrikerfi *Macintosh* en með betri möguleikum til víxlvinnslu. Það var orðið lýðum ljóst að víxlvinnsla, gluggakerfi, mýs og myndræn notendaskil voru að taka við af gamla *DOS*-inu. Um þessar mundir var líka veruleg aukning á notkun *UNIX* fyrir einmenningstölvur.

Af merkilegum hugbúnaðarpökkum frá þessum tíma má nefna umbrotsforritið *PageMaker* fyrir *Macintosh* tölvur sem *Aldus* sendi frá sér 1985 og fyrstu útgáfu af töflureikninum *Microsoft Excel* fyrir *Windows* sem kom 1987.

Á þessum árum færðist hægt og hægt í vöxt að einmenningstölvur í fyrirtækjum væru tengdar saman í staðarnet. *Internetið* tengdi saman tölvur við háskóla og rannsóknarstofnanir vítt og breitt um heiminn og var meðal annars notað til að senda tölvupóst. Hér á landi og annars staðar var farið að kenna framhaldsskólanemum á skrifstofuhugbúnað og bjóða almenningi upp á tölvunámskeið þar sem fólk gat lært á *WordPerfect*, *Multiplan, dBASE* og fleiri vinsæla hugbúnaðarpakka.

1990 – 1994

Árið 1992 kynntu *IBM* og *Motorola* nýjan 64 bita örgjörva sem kallast *Power PC 601*. Árið áður höfðu *Apple Computer* og *IBM* gert samning sín á milli um að þessi örgjörvi yrði notaður í *Apple Macintosh* tölvurnar. *Intel* var líka með nýjungar á prjónunum: Árið 1992 kynnti fyrirtækið *PCI braut* (*PCI local bus*) fyrir einmenningstölvur með hraða frá 8 upp í 33 megarið og árið eftir kom *Intel Pentium* örgjörvinn. Hann var með 32 bita gisti, 64 bita gagnabraut, gat nýtt 4 gígabæta minni og innihélt yfir 3 milljónir smára. Þessi fyrsta útgáfa *Pentium* örgjörvans keyrði á 66 megariða hraða og gat framkvæmt um 100 milljónir skipana á sekúndu.

Undir lok þessa tímabils fengust *PC-tölvur* með *Pentium* örgjörva og *PCI* braut og *Apple Macintosh* vélar með *PowerPC* örgjörva. Með þessum vélum fylgdu geisladrif, hljóðkort og fullkomin skjátengi og almenningur kynntist margmiðlun og tölvuleikjum með fullkominni þrívíddargrafík og víðóma hljóði.

Árið 1990 kom útgáfa 3.0 af *Microsoft Windows* stýrikerfinu. Með útgáfu 3.0 náði *Microsoft Windows* fyrst verulegri útbreiðslu og segja má að allt þetta tímabil einkennist af mikilli velgengni *Microsoft*.

Árið 1991 kom *Mac OS 7.0* fyrir *Apple Macintosh*, verulega endurbætt gerð af *Macintosh* stýrikerfinu sem hafði verið að þróast frá 1984. Það sama ár skrifaði Finninn

Linus Thorvalds stýrikerfið *Linux*. Það er nánast eins og *UNIX* sem hafði verið til frá því um 1970 en er hugsað fyrir einmenningstölvur og dreift ókeypis. Í upphafi vakti *Linux* litla athygli en þar sem það hentar vel sem stýrikerfi fyrir miðlara á *Internetinu* fór áhugi á því vaxandi þegar leið á 10. áratuginn.

Merkasta nýjung þessara ára er tvímælalaust veraldarvefurinn. Fyrsta útgáfa *HTML* skipanamálsins leit dagsins ljós 1991 og síðan hefur vefurinn vaxið hraðar en nokkurn óraði fyrir. Fyrsti myndræni vafrinn, *MOSAIC*, kom fram árið 1993. Í kjölfarið fylgdi svo *Netscape Navigator* 1994. Fyrir tíma myndrænna vafra voru til vafrar eins og *Lynx* sem aðeins gátu birt texta.

Fyrir 1990 var orðið nokkuð um að tölvur í fyrirtækjum væru tengdar saman í staðarnet. Á þessu tímabili tengjast staðarnet í stórum stíl við *Internetið* og samhliða þróun margmiðlunar verða tölvur í auknum mæli samskiptatæki.

1995 – 1999

Enn koma öflugri örgjörvar. 1995 kynnti *Intel* nýja gerð *Pentium* örgjörva, *Pentium Pro,* sem keyrði á allt að 200 megariðum og í kjölfarið komu sífellt öflugri afbrigði af *Intel Pentium* örgjörvum. 1998 kom 400 megariða *Intel Pentium II* og við lok tímabilsins voru 1000 megariða gjörvar í sjónmáli.

Meðal nýjunga á tölvumarkaði má nefna nýja gerð *Apple Macintosh* véla sem kallast *iMac* og kom á markað 1998.

Í upphafi þessa tímabils, árið 1995, sendi *Mirsosoft* frá sér stýrikerfið *Windows 95* sem var verulega endurbætt gerð *Windows* stýrikerfisins. Seinna kom svo *Windows 98.* Þessar nýju útgáfur *Windows* tóku í arf ýmis vandamál eldri gerðanna enda þurftu þær að geta keyrt forrit sem voru skrifuð fyrir *Windows 3.0* og *MS-DOS.* Samhliða þróun *Windows 95/98* vann *Microsoft* að gerð *Windows NT* sem átti að vera laust við þessi vandamál. Fyrsta útgáfa *Windows NT* sem náði verulegri útbreiðslu var *Windows NT 4.0* sem kom á markað *1996* og hefur síðan verið mikið notað sem stýrikerfi fyrir vefþjóna, bæði á staðarnetum og á *Internetinu*.

Uppgangur *Microsoft* hélt áfram og árið 1998 var það orðið verðmætasta fyrirtæki heims. En *Microsoft* hefur þó ekki náð að verða allsráðandi í framleiðslu stýrikerfa. Á þessu tímabili jukust vinsældir *Linux* til mikilla muna og notendur þess fóru að skipta milljónum. Einnig hélt þróun *Macintosh* stýrikerfisins áfram.

Meðal helstu nýjunga í hugbúnaðargerð á þessum tíma eru æ fullkomnari vafrar og ýmiss konar hugbúnaður til tölvusamskipta, vefsíðugerðar og upplýsingamiðlunar á *Internetinu*. Á þessari stundu er ómögulegt að meta hvað stendur upp úr af öllum þeim hugbúnaði sem komið hefur fram undanfarin 5 ár.

Samhliða þróun *Internetsins* hefur aukist áhugi á dreifðri vinnslu og gerð forrita sem keyra af vefsíðum og sækja gögn af netinu. Engin leið er að spá um hvert þessi þróun leiðir en nú þegar hafa tölvunet, fjölmiðlar og fjarskiptakerfi að nokkru runnið saman í eina heild. Útvarpsstöðvar senda út á *Internetinu* og það er hægt að nota farsíma til að lesa vefsíður og taka við skilaboðum frá tölvum.

1.5 Forritun í hálfa öld[2]

Fyrsta kynslóð forritunarmála

Stundum er talað um fjórar kynslóðir forritunarmála. Forritunarmál af *fyrstu kynslóð* voru eiginlega ekki nein mál. Skipanir voru gefnar með því að hreyfa til rofa eða hlaða einstaka bita í minni tölvunnar annað hvort handvirkt eða með gataspjöldum. Til að forrita á þessum málum þurfti að vita nákvæmlega hvernig vélbúnaðurinn var byggður. Fyrstu tölvurnar voru forritaðar með þessum hætti. Það var erfitt verk og tæpast í mannlegu valdi að búa til nema einföld forrit.

Önnur kynslóð forritunarmála

Forritunarmál af *annarri kynslóð* eru stundum kölluð *smalamál* (*assembly languages*). Þessi mál hafa eitt orð fyrir hverja aðgerð sem gjörvi vélarinnar getur framkvæmt. Hvert smalamál er bundið einni gerð gjörva og til að nota þau þarf verulega þekkingu á innviðum gjörvans og minnisins í tölvunni.

Þótt smalamál séu erfið og henti illa til að orða flóknar hugmyndir voru þau mikil framför frá forritunarmálum af fyrstu kynslóð. Það er auðveldara fyrir fólk að skilja skipun á smalalmáli eins og

```
ADD 5 #513
```

sem gæti merkt: Bættu 5 við töluna í minnishólfi númer 513 heldur en vélamálskóta á borð við

```
10001110 00000101 00000010 00000001
```

eða fyrirmæli um að hleypa straumi á tilteknar rásir.

Notkun forritunarmála af annarri kynslóð hófst í byrjun 6. áratugarins og þau eru enn notuð við gerð rekla fyrir jaðartæki og önnur verkefni þar sem forritari þarf fullkomið vald yfir innviðum vélarinnar.

Þriðja kynslóð forritunarmála

Mál af *þriðju kynslóð* eru stundum kölluð æðri forritunarmál. Hægt er að nota þau til að orða aðferðir handa tölvu til að vinna eftir án þess að vita mikið um gerð og byggingu vélbúnaðarins – forritari getur þá einbeitt sér að því að hugsa um aðferðirnar sem forritið á að vinna eftir en þarf ekki að hugsa um innviði gjörva og minnis.

Fyrsta forritunarmálið af þriðju kynslóð heitir *Fortran*. Það var fullgert árið 1954 og fóru 25 ársverk í að semja þýðandann. (Þýðandi er forrit sem þýðir af forritunarmáli á vélamálið sem tölvan „skilur“.)

Í árdaga þriðju kynslóðar mála voru forrit oftast skráð á gataspjöld. Skipanirnar voru vélritaðar á vélar sem umrituðu þær sem mynstur úr götum. Tölvurnar voru svo tengdar lesurum sem gátu flett gegnum bunka af spjöldum og lesið mynstrin af þeim. Þegar búið var að skrifa forrit, t.d. á *Fortran*, var bunkinn með spjöldunum tekinn og farið með hann þangað sem tölvan var. Til að keyra forritið var byrjað á að láta tölvuna lesa spjaldabunka með *Fortan*-þýðanda, síðan voru spjöldin með forritinu sett í lesarann,

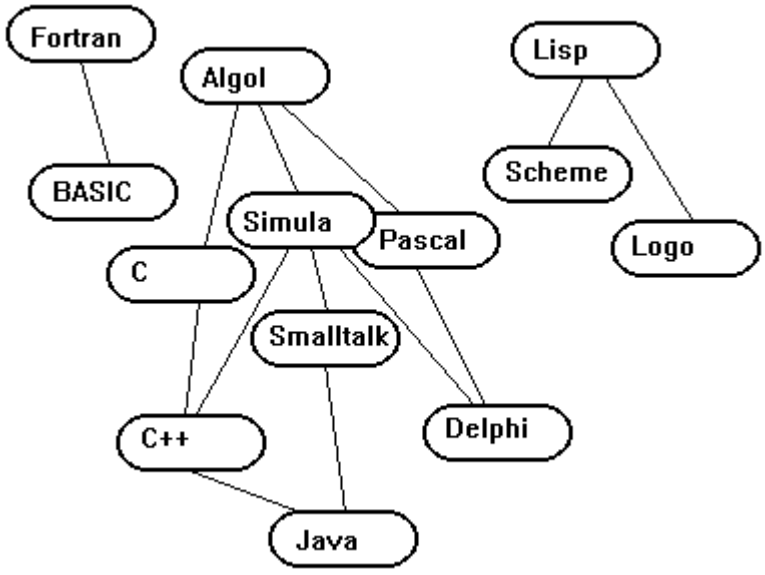
[2] Hér er aðeins sagt lauslega frá þróun forritunarmála. Ítarlegri umfjöllun um forritun og forritunarmál er að finna í 8. kafla.

tölvan las þau í minni og keyrði svo þýðandann og þýddi forritið af *Fortran* á vélamál. Ef forritið var villulaust sendi tölvan vélamálsþýðinguna til götunarvélar sem skrifaði þýðinguna á ný spjöld. Nú var hægt að mata tölvuna á nýju spjöldunum og keyra forritið. Úttakið var svo yfirleitt sent til prentara sem skrifaði tölur eða aðrar niðurstöður á blað.

Fortran málið frá 1954 var frumstætt og það var erfitt að nota það við lausn flókinna verkefna. Á árunum kringum 1960 komu fram tvö forritunarmál sem öðrum fremur vörðuðu leiðina til nútímans. Þau eru *Lisp* og *Algol*. Eins og myndin sýnir eru mörg algeng forritunarmál sem notuð eru nú til dags „afkomendur“ þessara þriggja „fornmála“ *Fortran, Lisp* og *Algol*.

Lisp er elst svokallaðra fallamála (á ensku *functional languages*). *Lisp* og mál sem af því eru dregin hafa ekki verið mikið notuð til að framleiða hugbúnað til að selja en þess meira við tilraunir og rannsóknir í ýmsum greinum tölvufræða. Enn þann dag í dag gegna mál af þessari málaætt, eins og *Scheme*, mikilvægu hlutverki í rannsóknum, t.d. á sviði gervigreindarfræða. Forritunarmálið *Logo*, sem er líkt *Lisp*, hefur víða um heim verið notað til að kenna börnum og unglingum undirstöðuatriði tölvufræða.

Mikilvægustu afkomendur *Fortran* eru ýmis afbrigði af *BASIC*. Fyrsta gerð *BASIC* málsins var búin til um 1965 og þegar einmenningstölvur urðu algengar á árunum í kringum 1980 fylgdi þeim yfirleitt *BASIC* túlkur. Eldri gerðir *Fortran* og *BASIC* hentuðu illa til að vinna flókin verkefni vegna þess að forrit á þessum málum voru skrifuð sem ein halarófa af skipunum. Til að sigrast á þessum takmörkunum voru búin til svokölluð mótuð forritunarmál. Hið fyrsta þeirra var *Algol*.



Þegar forritunarmál eru kölluð *mótuð* er átt við að á þeim sé hægt að:

- Pakka mörgum skipunum saman í blokk og setja heilar blokkir inn í slaufur eða skilyrðissetningar;
- Skilgreina tegundir gagna og láta einstakar breytur geyma svo flóknar og margsamsettar upplýsingar sem verkast vill;
- Skipta verki í undirforrit eða föll sem hafa staðværar breytur.

Algol hafði mikil áhrif á þróun forritunarmála. Þó það næði aldrei mikilli útbreiðslu urðu málin *Pascal* og C sem samin voru undir áhrifum frá *Algol* mjög vinsæl. Þessi mál voru bæði búin til rétt fyrir 1970.

Mótuðu málin, *Pascal* og *C*, fullnægðu flestum þörfum forritara fram yfir 1980 en með tilkomu gluggakerfa og myndrænna notendaskila urðu forrit flóknari og efiðara að skrifa þau á þessum málum. Frá því um 1970 hafði vísindamönnum á sviði hugbúnaðarfræða verið ljóst að hlutbundin mál auðvelduðu mjög alla forritun fyrir gluggakerfi og greiddu fyrir samvinnu margra forritara við lausn flókinna verkefna. Fyrsta málið sem bauð að einhverju leyti upp á hlutbundnar aðferðir við forritun heitir *Simula*. Það var búið til um miðjan 7. áratuginn. Fyrsta forritunarmálið sem bauð upp á alla kosti hlutbundinnar forritunar heitir *Smalltalk*. Það varð til rétt fyrir 1970. Síðar komu svo hlutbundnar útgáfur af *Pascal* og *C*.

Þegar sagt er að forritunarmál sé hlutbundið er átt við að það hafi eftirtalin einkenni til viðbótar við mótuðu málin:

- Hægt er að skilgreina tegundir af hlutum og byggja inn í þær svo flókin gögn sem verkast vill og aðferðir til að vinna úr gögnunum, bregðast við fyrirmælum, eða senda skilaboð til annarra hluta. Slík tegund kallast *klasi*.
- Þegar *klasi* er skilgreindur er hægt að láta hann *erfa* eiginleika og aðferðir *klasa* sem til eru fyrir. Sé til dæmis búið að búa til einfaldan glugga sem hægt er að birta á skjánum þá þarf ekki að byrja upp á nýtt frá grunni ef þörf er á glugga með valmynd. Það er hægt að láta nýja *klasann* erfa aðferðir og eiginleika þess gamla og bæta aðeins við því sem hann á að hafa til viðbótar.

Það forritunarmál sem nýtur mestra vinsælda meðal atvinnumanna í forritun nú til dags er hlutbundin gerð af *C* sem heitir *C++* og var búin til um miðjan 9. áratuginn. Önnur hlutbundin mál sem njóta vinsælda meðal forritara um þessar mundir eru *Java* og ýmsar hlutbundnar útgáfur af *Pascal*, þeirra þekktust er *Delphi*.

Forritunarmálið *Java* var búið til á árunum upp úr 1990. Það er fyrsta forritunarmálið sem er hannað frá grunni til að nýta möguleika *Internetsins*.

Fjórða kynslóð forritunarmála

Forritunarmál af *fjórðu kynslóð* eru enn sem komið er lítið notuð við gerð stórra forrita eða hugbúnaðarpakka. Þau eru enn í mótun. Munurinn á þeim og málum af þriðju kynslóð er sá að til að skipa tölvu fyrir verkum á máli af *fjórðu kynslóð* þarf ekki að stafa nákvæmlega ofan í hana hvernig á að vinna verkin, það dugar að segja henni hvað á að gera. Þau mál af *fjórðu kynslóð* sem hafa náð mestri útbreiðslu eru *Prolog* og *SQL*. *Prolog* hefur einkum verið notað til að forrita sérfræðikerfi, þ.e. hugbúnað sem dregur ályktanir af upplýsingum og svarar spurningum á einhverju sérsviði. *SQL* (*Structured Query Language*) er einkum notað til að fiska upplýsingar úr gagnasöfnum.

Forritun einmenningstölva

Fram á miðjan 9. áratuginn fylgdi yfirleitt *BASIC* túlkur með einmenningstölvum og fram til 1983 voru flest forrit fyrir þær annað hvort skrifuð á *BASIC* og keyrð af mishægvirkum túlkum eða þá á smalamálum.

1983 hóf fyrirtækið *Borland International* sölu á *Pascal* þýðanda og forritunarumhverfi fyrir tölvur með *MS DOS* og *CP/M* stýrikefi sem kallaðist *Turbo Pascal*. Í

kjölfarið komu svo þýðendur og forritunarumhverfi fyrir fleiri mótuð forritunarmál eins og *C* og það urðu þáttaskil í gerð hugbúnaðar fyrir einmenningstölvur. Þegar glugga-stýrikerfin komu á *Macintosh* 1984 og seinna fyrir *PC-tölvur* skapaðist þörf fyrir enn fullkomnari forritunarmál og hlutbundin mál, einkum *C++*, tóku við af *Pascal* og *C*. Þó hélt *BASIC* vinsældum sínum meðal þeirra sem smíða lítil og einföld forrit. Fram komu sífellt fullkomnari afbrigði af *BASIC* eins og *Visual BASIC* frá *Microsoft* sem hefur flesta kosti mótaðra mála. Nú til dags eru ýmis afbrigði af *BASIC* notuð til að skrifa fjölva og alls konar smáforrit fyrir einmenningstölvur, en stórir hugbúnaðarpakkar eru flestir skrifaðir á *C++*.

2. kafli

Tvíundakerfi og Boole-algebra

2.1 Tvíundakerfi

Sætisritháttur og tugakerfi

Við erum vön að skrifa tölur með tölustöfunum 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9 og láta tölustafinn lengst til hægri tákna einingar en hvern staf svo vega 10 sinnum meira en þann næsta til hægri. Í talnakerfinu sem við erum vön að nota er talan tíu *grunntala* og

2354 merkir $2\cdot10^3 + 3\cdot10^2 + 5\cdot10^1 + 4\cdot10^0$

Þessi aðferð til að skrifa tölur kallast *sætisritháttur*. Einkenni hans er að talan sem nefnd er með tölustafnum í sæti númer n er margfölduð með G^{n-1}, þar sem G er grunntalan.

Evrópubúar lærðu af Aröbum að nota sætisrithátt með grunntölunni 10. Þessi aðferð tók að breiðast út um 1120 þegar kennslubók í reikningi sem arabíski stærðfræðingurinn *al-Khwarizimi* skrifaði um 825 var þýdd á latínu. Í þessari bók eru kenndar aðferðir til að leggja saman, draga frá, margfalda og deila svipaðar þeim sem grunnskólanemar læra enn þann dag í dag. Þessar aðferðir byggja algerlega á sætisrithætti og voru mikil framför frá eldri aðferðum sem notaðar voru við talnareikning meðan tölur voru skrifaðar með rómverskum tölustöfum. Orðið *algorithm* í ensku (og svipuð orð í öðrum málum eins og *Algorithmus* í þýsku) er dregið af nafni *al-Khwarizimi*. Íslenska orðið yfir þetta lykilhugtak tölvufræðinnar er *algrím*. Reikniaðferðirnar fjórar sem kenndar eru í bók *al-Khwarizimi* eru velþekkt dæmi um *algrím*, þ.e. endanlegar, öruggar og ótvíræðar aðferðir til að leysa tiltekin verkefni.

Mörgum þykir aðferðin sem við höfum til að skrifa tölur og gefa þeim nöfn svo sjálfsögð að þeir gera ekki greinarmun á tölustöfunum og tölunum sjálfum. Runa af tölustöfum á blaði er þó ekki tala heldur nafn á tölu og það eru ótal leiðir til að nefna sömu töluna. Hér fara á eftir nokkur mismunandi heiti tölunnar 92:

XCII	Rómverskir tölustafir
níutíu og tveir	Íslensk töluorð
92	Sætisritháttur með grunntölunni 10
5C	Sætisritháttur með grunntölunni 16
1011100	Sætisritháttur með grunntölunni 2

Tvíundakerfi

Frá því fyrst var tekið að smíða tölvur hafa þær svo til allar unnið með tölur í tvíundakerfi, þ.e. notað sætisrithátt með grunntölunni 2. Kosturinn við þennan rithátt er að það þarf aðeins að nota tvo mismunandi tölustafi: 0 og 1. Vél sem byggir á tvíundakerfi þarf því aðeins að geta gert greinarmun á tvenns konar merkjum eða táknum og smíði slíkrar

vélar er mun auðveldari en smíði tækja sem aðgreina 3, 4 eða jafnvel 10 mismunandi tákn.

Í tugakerfi merkir fyrsta sætið 10^0 (þ.e. 1), annað sætið 10^1, það þriðja 10^2 og sæti númer n merkir 10^{n-1}. Í tvíundakerfi merkir fyrsta sætið 2^0 (þ.e. 1), annað sætið 2^1, hið þriðja 2^2 og sæti númer n merkir 2^{n-1} . Það er sama hvaða grunntala er notuð, lægsta tveggja stafa talan er alltaf rituð 10 og í tvíundakerfi merkir 10 töluna $1\cdot2^1 + 0\cdot2^0$ eða með öðrum orðum töluna tvo.

Tala sem er rituð 1011100 í tvíundakerfi jafngildir

$1\cdot2^6 + 0\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 =$

$64 + 0 + 16 + 8 + 4 + 0 + 0 = 92.$

Verkefni 2.1.a

Hér fara á eftir nokkrar tölur ritaðar í tvíundakerfi. Skrifaðu þessar sömu tölur í tugakerfi.

100	1000	10000
11001101	11111111	10001000

Lágstafir hægra megin við tölu eru oft notaðir til að tákna grunntölu. 1327_{10} merkir t.d. að átt sé við töluna sem er rituð 1327 í tugakerfi og 235_8 merkir að átt sé við töluna sem er rituð 235 í talnakerfi með grunntöluna 8.

Að breyta úr tugakerfi í tvíundakerfi

Það er vandalítið að breyta úr tvíundakerfi í tugakerfi. Það dugar að vita vægi hvers sætis og leggja saman þau sæti sem innihalda 1. Það er ögn meira mál að umrita tölu úr tugakerfi í tvíundakerfi. Aðferðin til þess byggir á því að síðasti stafur í tölu er talan sem gengur af ef deilt er með grunntölunni. Sé t.d. deilt með 10_{10} í 1327_{10} þá koma út 132_{10} og 7 ganga af, enda er 7 síðasti tölustafurinn í 1327. Næstsíðasti tölustafurinn er svo það sem gengur af þegar 10_{10} er deilt í 132_{10}, þ.e. í útkomuna úr því að deila 10_{10} í 1327_{10}. Kerfið er semsagt svona:

10 er deilt í	1327	útkoman er	132	afgangur er	7
10 er deilt í	132	útkoman er	13	afgangur er	2
10 er deilt í	13	útkoman er	1	afgangur er	3
10 er deilt í	1	útkoman er	0	afgangur er	1

Afgangarnir sem lenda í aftasta dálki eru tölustafirnir í tölunni, sá aftasti efst o.s.fr. Þetta sama gildir um tvíundakerfi.

2 er deilt í	1327	útkoman er	663	afgangur er	1
2 er deilt í	663	útkoman er	331	afgangur er	1
2 er deilt í	331	útkoman er	165	afgangur er	1
2 er deilt í	165	útkoman er	82	afgangur er	1
2 er deilt í	82	útkoman er	41	afgangur er	0
2 er deilt í	41	útkoman er	20	afgangur er	1
2 er deilt í	20	útkoman er	10	afgangur er	0

2 er deilt í	10	útkoman er	5	afgangur er	0
2 er deilt í	5	útkoman er	2	afgangur er	1
2 er deilt í	2	útkoman er	1	afgangur er	0
2 er deilt í	1	útkoman er	0	afgangur er	1

Talan sem er rituð 1327 í tugakerfi er því rituð 10100101111 í tvíundakerfi. (Útreikningarnir hér eru allir skrifaðir í tugakerfi.)

Með þessari sömu aðferð er hægt að umrita tölur úr tugakerfi í hvaða talnakerfi sem er.

Verkefni 2.1.b

Hér fara á eftir nokkrar tölur ritaðar í tugakerfi. Skrifaðu þessar sömu tölur í tvíundakerfi og í talnakerfi með grunntöluna 8.

8 32 512 15 45 777

Reikningur í tvíundakerfi

Reiknireglur eru óháðar því hvaða grunntala er notuð svo það gilda sömu reiknireglur í tvíundakerfi og í tugakerfi. Þegar tvær tölur eru lagðar saman er einn geymdur ef útkoma úr dálki er jöfn eða hærri grunntölunni. Sé tugakerfi notað er geymt ef útkoma er jöfn eða hærri en tíu og sé tvíundakerfi notað er geymt ef útkoma er jöfn eða hærri en tveir. Samlagningartafla fyrir tugakerfi er nokkuð löng. Hér á eftir er hluti af henni (4 línur af 100).

0 + 0 = 0
0 + 1 = 1
…
7 + 2 = 9
7 + 3 = 0 og 1 er geymdur

Samlagningartafla fyrir tvíundakerfi er hins vegar mjög stutt:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 og 1 er geymdur

Þessa töflu getum við notað til að leggja saman tvær tölur í tvíundakerfi svona:

Geymt	1		1	1	1			
Tala 1		1	0	1	1	1	0	0
Tala 2		1	0	0	1	1	0	1
Útkoma	1	0	1	0	1	0	0	1

Margföldunartafla fyrir tvíundakerfi er líka afar stutt og einföld:

0 · 0 = 0
0 · 1 = 0
1 · 0 = 0
1 · 1 = 1

Með þessa töflu að vopni getum við margfaldað saman tvær tölur í tvíundakerfi svona:

Tala 1				1	0	1	1
Tala 2					1	0	1
Margfaldað með aftasta staf í tölu 2				1	0	1	1
Margfaldað með miðstaf í tölu 2			0	0	0	0	
Margfaldað með fremsta staf í tölu 2		1	0	1	1		
Útkoma (summan af 3. til 5. línu)		1	1	0	1	1	1

Séu sömu tölur ritaðar í tugakerfi og margfaldaðar saman fáum við út reiknisdæmið $11 \cdot 5 = 55$.

Alveg eins og margföldun með 10_{10}, 100_{10}, 1000_{10} o.s.fr. er auðveld í tugakerfi er auðvelt að margfalda með 10_2, 100_2 og 1000_2 (þ.e. 2_{10}, 4_{10}, og 8_{10}) o.s.fr. í tvíundakerfi, það bætast bara 0 aftan við töluna. Dæmi: $11011_2 \cdot 100_2 = 1101100_2$.

Verkefni 2.1.c

Reiknaðu þessi dæmi í tvíundakerfi:

1000 + 1000	1000 · 10
11100011 + 1001110	111001 · 110
10001000 + 1110111	1001 · 11

Verkefni 2.1.d

Skrifaðu samlagningar- og margföldunartöflu fyrir talnakerfi með grunntöluna 3.

Sextándakerfi

Tvíundakerfið er einfalt að því leyti að það hefur aðeins tvö tákn og margföldunar og samlagningartöflur eru mjög stuttar, aðeins 4 línur hvor. Hins vegar verða tölur ansi langar. Það þarf t.d. 11_{10} stafi til að skrifa 1327_{10} sem er rituð með 4 stöfum í tugakerfi. Það er þægilegra fyrir fólk að lesa og skrifa tölur með færri stöfum en gert er í tvíundakerfi, þess vegna er talnakerfi með grunntölunni 16_{10} oft notað til að umrita tölur af tvíundakerfi. Þetta kerfi kallast *hexadecimal system* á ensku og á íslensku er það ýmist nefnt *hexadesímalkerfi* eða *sextándakerfi*.

Í sextándakerfi eru notaðir 16 mismunandi tölustafir:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E og F.

Talan sem er rituð 4B7 í sextándakerfi jafngildir tugakerfistölunni $4 \cdot 16^2 + 11 \cdot 16^1 + 7 \cdot 16^0 = 1207$

Þar sem 16_{10} er heilt veldi af 2 ($2^4 = 16_{10}$) er mun auðveldara að breyta úr tvíundakerfi í sextándakerfi heldur en í tugakerfi. Fjórir tölustafir í tvíundakerfi samsvara ævinlega einum tölustaf í sextándakerfi en ýmist einum eða tveim tölustöfum í tugakerfi. Eftirfarandi tafla sýnir allar mögulegar 4 stafa tölur í tvíundakerfi umritaðar í tuga- og sextándakerfi.

Tvíundakerfi	Tugakerfi	Sextándakerfi
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Hægt er að nota þessa töflu til að umrita tölur milli tvíunda- og sextándakerfis. Tökum dæmi: Talan sem er rituð 1110111001 í tvíundakerfi er svona ef tölustafir eru teknir saman 4 og 4:

0011 1011 1001

Það þurfti að bæta 2 núllum framan við til að fjöldi stafa yrði heilt margfeldi af 4. Með töflunni er hægt að umrita hana í sextándakerfi svona:

3 B 9

Það er jafnauðvelt að fara í hina áttina. Talan sem er rituð 7F05 í sextándakerfi er t.d. rituð svona í tvíundakerfi:

0111 1111 0000 0101

Verkefni 2.1.e

Skrifaðu í tvíundakerfi:	$1BC_{16}$	100_{16}	FF_{16}
Skrifaðu í sextándakerfi:	1100_2	110000_2	10000000_2

2.2 Boole-algebra

Einfaldar og samsettar yrðingar

Árið 1852 sendi enski stærðfræðingurinn *George Boole* frá sér bók sem hann kallaði *Rannsókn á lögmálum hugsunarinnar* (*An Investigation of the Laws of Thought*). Í þessari bók setti *Boole* fram algebru fyrir yrðingar sem hver um sig getur haft tvö gildi: *Satt* og *ósatt*.

Við getum táknað satt með 1 og ósatt með 0. Samkvæmt venju látum við bókstafina p, q og r standa fyrir yrðingar. Ef við hugsum okkur til dæmis að

p standi fyrir yrðinguna: „Kýr eru spendýr“
q standi fyrir yrðinguna: „2 + 2 = 5“ og
r standi fyrir yrðinguna: „Fagurt er í Fjörðum“

þá getum við sagt að

p hafi gildið 1 og
q hafi gildið 0.
Um sanngildi r getur svo hver og einn haft sína skoðun.

Úr einföldum yrðingum eins og p og q og rökaðgerðum er hægt að mynda samsettar yrðingar. Helstu rökaðgerðir eru:

og	hér táknuð með &.	Stundum táknuð með: AND, &&, $\wedge$, $\cdot$, $\cap$
eða	hér táknuð með $\vee$.	Stundum táknuð með: OR, |, ||, +, $\cup$
ekki	hér táknuð með ~.	Stundum táknuð með: NOT, –, $\neg$, !
ef-þá	hér táknuð með $\Rightarrow$.	Stundum táknuð með: $\rightarrow$, $\supset$

Ekki er líka stundum táknað með striki ofan við yrðingu, svona: $\bar{p}$

Með þessum táknum getum við myndað samsettar yrðingar eins og

p & r	Kýr eru spendýr og fagurt er í Fjörðum.
r $\Rightarrow$ q	Ef fagurt er í Fjörðum þá er 2 + 2 = 5.
r $\vee$ ~r	Fagurt er í Fjörðum eða ekki er fagurt í Fjörðum.

Sanntöflur

Sanngildi samsettra yrðinga veltur á sanngildi einföldu yrðinganna sem þær eru settar saman úr. Ef tvær yrðingar eru t.d. tengdar saman með & þá verður útkoman sönn yrðing ef þær eru báðar sannar en ósönn ella. Þetta er hægt að setja fram með *sanntöflu* eins og gert er hér til hægri.

p	q	p & q
1	1	1
1	0	0
0	1	0
0	0	0

Taflan hefur 4 línur því hægt er að úthluta einföldu yrðingunum p og q sanngildum á fjóra vegu. Samsetta yrðingin p & q fær gildið 1 ef bæði p og q hafa gildið 1, annars gildið 0.

Lítum nú á sanntöflur fyrir $\vee$, $\Rightarrow$ og ~.

p	q	p $\vee$ q
1	1	1
1	0	1
0	1	1
0	0	0

p	q	p $\Rightarrow$ q
1	1	1
1	0	0
0	1	1
0	0	1

p	~p
1	0
0	1

Eins og sést af töflunni fyrir $\vee$ merkir samtengingin að a.m.k. önnur yrðingin sé sönn. Taflan fyrir p $\Rightarrow$ q sýnir að tengingin segir það eitt að ef p er satt þá er q það líka. Yrðingin p $\Rightarrow$ q útilokar semsagt aðeins þann möguleika að p sé satt en q ósatt. Í stað p $\Rightarrow$ q er því eins hægt að rita ~(p & ~q).

Aðgerðin ~ er ólík hinum að því leyti að henni er aðeins beitt á eina yrðingu.

Með hliðsjón af þessum 4 sanntöflum er hægt að búa til sanntöflu fyrir flóknari yrðingar eins og t.d. (~p & q) $\vee$ (p & ~q). Aðferðin er sú að búa fyrst til töflur fyrir

(~p & q) annars vegar (5. dálkur í töflunni hér að neðan) og (p & ~q) hins vegar (6. dálkur) og beita svo aðgerðinni $\vee$ á þær (7. dálkur).

		Leitt af 1. og 2. dálki og sanntöflu fyrir ~		Leitt af 2. og 3. dálki og sann-töflu fyrir &	Leitt af 1. og 4. dálki og sann-töflu fyrir &	Leitt af 5. og 6. dálki og sanntöflu fyrir $\vee$.
p	q	~p	~q	~p & q	p & ~q	(~p & q) $\vee$ (p & ~q)
1	1	0	0	0	0	0
1	0	0	1	0	1	1
0	1	1	0	1	0	1
0	0	1	1	0	0	0

Verkefni 2.2.a

Búðu til sanntöflur fyrir þessar yrðingar:

p & ~q	~(p $\vee$ q)	p $\Rightarrow$ ~q
~(p & q)	p & ~(p & q)	(p $\vee$ q) & (p $\vee$ ~q)

Mótsögn og klifun

Samsett yrðing sem er sönn undir öllum kringumstæðum kallast *klifun*. Yrðingin p $\vee$ ~p er dæmi um klifun enda er það augljóst að hún er sönn hvort sem p er satt eða ósatt.

Samsett yrðing sem er ósönn undir öllum kringumstæðum kallast *mótsögn*. Yrðingin p & ~p er dæmi um mótsögn enda er það augljóst að hún er ósönn hvort sem p er satt eða ósatt.

Stundum er ekki augljóst við fyrstu sýn hvort yrðing er klifun, mótsögn eða hvorugt en það er alltaf hægt að ganga úr skugga um það með því að búa til sanntöflu. Hér fer á eftir sanntafla sem leiðir í ljós að yrðingin p $\vee$ ~(p & q) er klifun.

		Leitt af 1. og 2. dálki og sanntöflu fyrir &	Leitt af 3. dálki og sanntöflu fyrir ~	Leitt af 1. og 4. dálki og sanntöflu fyrir $\vee$.
p	q	p & q	~(p & q)	p $\vee$ ~(p & q)
1	1	1	0	1
1	0	0	1	1
0	1	0	1	1
0	0	0	1	1

Verkefni 2.2.b

Notaðu sanntöflur til að finna hverjar þessara yrðinga eru klifun, hverjar eru mótsögn og hverjar eru hvorki klifun né mótsögn.

(p & q) & (~p $\vee$ ~q)	(~p $\vee$ q) & (p $\vee$ ~ q)
(p & q) $\vee$ (~p $\vee$ ~q)	~((p $\Rightarrow$ q) & p) $\vee$ q

Sannföll af mörgum yrðingum

			Leitt af 1. og 2. dálki og sanntöflu fyrir &	Leitt af 3. og 4. dálki og sanntöflu fyrir $\vee$.
p	q	r	p & q	(p & q) $\vee$ r
1	1	1	1	1
1	1	0	1	1
1	0	1	0	1
1	0	0	0	0
0	1	1	0	1
0	1	0	0	0
0	0	1	0	1
0	0	0	0	0

Sanntafla yfir yrðingu sem er samsett úr tveim einföldum yrðingum hefur 4 línur því hægt er að raða sanngildunum 1 og 0 á tvær yrðingar á 2^2 vegu. Séu einföldu yrðingarnar 3 eru línurnar í sanntöflunni 8 (þ.e. 2^3) og almennt gildir að fyrir n einfaldar yrðingar þarf sanntöflu með 2^n línum. Hér til vinstri er dæmi um sanntöflu fyrir yrðinguna (p & q) $\vee$ r.

Til eru 16 mismunandi sannföll af tveim einföldum yrðingum því í dálk með 4 línum er hægt að raða gildunum 1 og 0 á 2^4 vegu. Séu yrðingarnar 3 verða línurnar $2^3 = 8$ og fjöldi mismunandi sannfalla $2^8 = 256$. Almennt gildir að séu n mismunandi einfaldar yrðingar þá eru línurnar í sanntöflunni 2^n og fjöldi mögulegra sannfalla 2^{2^n}. Ekki þarf margar yrðingar til að þetta verði mjög háar tölur.

n	2^n	2^{2^n}
1	2	4
2	4	16
3	8	256
4	16	65536
5	32	4294967296
6	64	$\approx 1{,}8\times10^{19}$
7	128	$\approx 3{,}4\times10^{38}$
8	256	$\approx 1{,}2\times10^{77}$

Sannföll og staðlað eða-form

Eins og nefnt hefur verið eru til 16 möguleg sannföll af 2 yrðingum, eða jafn mörg og 4 stafa tölur í tvíundakerfi. Þau eru öll talin upp í þessari töflu.

p	q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Fall nr. 0 er mótsögn og nr. 15 er klifun. Önnur kunnugleg föll eru:

- Fall númer 8 sem er p & q
- Fall númer 11 sem er $p \Rightarrow q$
- Fall númer 14 sem er $p \vee q$

Tvö önnur föll er vert að nefna:

- Fall númer 6 er *útilokandi eða* (þ.e. annar en ekki báðir). Þetta fall er stundum kallað XOR og táknað: p XOR q.
- Fall númer 9 er *jafngildi* (p og q hafa sama sanngildi). Þetta fall er táknað með: $p \equiv q$.

Öll þessi sannföll er hægt að mynda með tengjunum &, $\vee$ og ~. Ein aðferð til að búa til formúlu fyrir gefið sannfall úr p, q og þessum tengjum er að nota *staðlað eða form*.

p	q	5
1	1	0
1	0	1
0	1	0
0	0	1

Lítum til dæmis á dálk númer 5. Við myndum formúlu á *stöðluðu eða formi* með því að skoða þær línur sem hafa sanngildið 1. Hér eru það línur 2 og 4. Í línu 2 er p satt en q ósatt, sem við táknum með (p & ~q). Í línu 4 er p ósatt og q líka, sem við táknum með (~p & ~q). Formúlan er sönn ef annað af þessu er satt sem við táknum með (p & ~q) $\vee$ (~p & ~q).

p	q	r	
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	1

Tökum annað dæmi. Við ætlum að búa til formúlu úr þrem einföldum yrðingum, p, q og r og láta hana hafa sanntöflu eins og hér til hægri. Við búum fyrst til eina &-setningu fyrir hverja línu þar sem formúlan á að hafa gildið 1 og tengjum &-setningarnar svo saman með $\vee$. Gildið 1 kemur fyrir í 3 línum og &-setningarnar sem samsvara þessum línum eru:

(p & ~q & ~r) fyrir 4. línu.
(~p & q & r) fyrir 5. línu.
(~p & ~q & ~r) fyrir 8. línu.

Setningin verður því (p & ~q & ~r) $\vee$ (~p & q & r) $\vee$ (~p & ~q & ~r).

Verkefni 2.2.c

Búðu til yrðingar á samræmdu eða formi sem hafa sanntöflur sem eru eins og dálkar 2, 3, 10 og 11 í töflunni yfir öll möguleg sannföll af tveim yrðingum.

Lögmál De Morgans

Ekki er nóg með að hægt sé að mynda öll möguleg sannföll með tengjunum &, $\vee$ og ~. Það dugar að nota ~ og annað hinna því

(p & q) er jafngilt ~(~p $\vee$ ~q) og
(p $\vee$ q) er jafngilt ~(~p & ~q) eins og þessar sanntöflur sýna:

p	q	~p	~q	**p & q**	(~p $\vee$ ~q)	**~(~p $\vee$ ~q)**
1	1	0	0	1	0	1
1	0	0	1	0	1	0
0	1	1	0	0	1	0
0	0	1	1	0	1	0

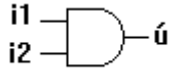
p	q	~p	~q	**p $\vee$ q**	(~p & ~q)	**~(~p & ~q)**
1	1	0	0	1	0	1
1	0	0	1	1	0	1
0	1	1	0	1	0	1
0	0	1	1	0	1	0

Þessi tvö jafngildi eru kennd við rökfræðinginn De Morgan (1806 – 1871) og kölluð *lögmál De Morgans*.

2.3 Boole-algebra, rökrásir og reikningur í tvíundakerfi

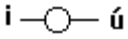
Rökrásir

Tölvur eru smíðaðar úr *rökrásum*, þ.e. raftækjum sem samsvara sannföllum. Allar rökrásir er hægt að byggja úr einingum sem samsvara tengjunum &, $\vee$ og $\sim$ og kallast *og-rás*, *eða-rás* og *ekki-rás*. Hér fyrir neðan eru táknmyndir af þessum þrem rásum.



og-rás

eða-rás

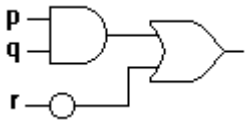


ekki-rás

Og-rás hagar sér eins og rökaðgerðin & að því leyti að straumur kemur út um úttakið (ú) ef straumur fer inn um bæði inntökin (i1 og i2) en annars ekki. Ef við táknum straum með 1 og engan straum með 0 getum við sagt að úttak *og-rásar* hafi gildið 1 ef bæði inntökin hafa gildið 1.

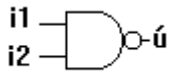
Svipaða sögu er að segja um *eða-rás*. Hún hagar sér eins og rökaðgerðin $\vee$ að því leyti að úttakið fær gildið 1 ef a.m.k. annað inntakið hefur gildið 1.

Úttak *ekki-rásar* er 0 ef inntakið er 1 og öfugt. Úr þessum þrem rásum er hægt að smíða rásir sem samsvara samsettum yrðingum. Myndin hér til hægri sýnir t.d. rás sem samsvarar yrðingunni $(p \,\&\, q) \vee \sim r$.



nand og nor

Til að smíða *og-rás* þarf að minnsta kosti 3 smára og það sama gildir um *eða-rás*. Hins vegar er hægt að smíða rásir sem samsvara yrðingunum $\sim(p \,\&\, q)$ og $\sim(p \vee q)$ úr tveim smárum hvora. Þessar rásir eru táknaðar með svona myndum.

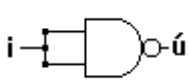


nand-rás

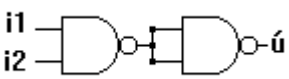
nor-rás

Þær eru yfirleitt nefndar með enskuslettunum *nand* (skammstöfun á *not and*) og *nor* (skammstöfun á *not or*) þó til séu íslensku orðin *eibeggjarás* og *samneitunarrás*.

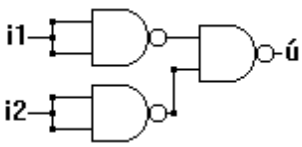
Þar sem það þarf færri smára í þessar rásir heldur en í *og-* og *eða-rásir* eru þær mikið notaðar í tölvur og annan rafeindabúnað. Þær hafa líka þá sérstöðu að úr hvorri þeirra sem er má byggja allar mögulegar rökrásir. Úr *nand-rásum* er t.d. hægt að byggja *ekki-*, *og-*, og *eða-rásir* eins og myndin sýnir.



Ekki-rás úr einni *nand-rás*



Og-rás úr 2 *nand-rásum*



Eða-rás úr 3 *nand-rásum*

Eða-rásin sem byggð er úr *nand-rásum* er leidd beint af aðferðinni til að smíða *ekki-rás* og *lögmáli De Morgans* sem segir að $(p \vee q)$ sé jafngilt $\sim(\sim p \,\&\, \sim q)$.

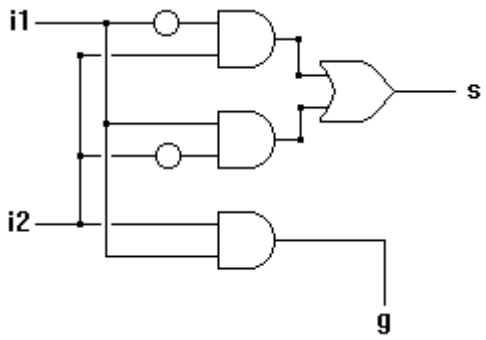
Verkefni 2.3.a

Hvernig er hægt að byggja *ekki-*, *eða-,* og *og-rásir* úr tómum *nor-rásum*?

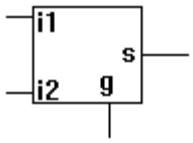
Samlagning úr rökaðgerðum

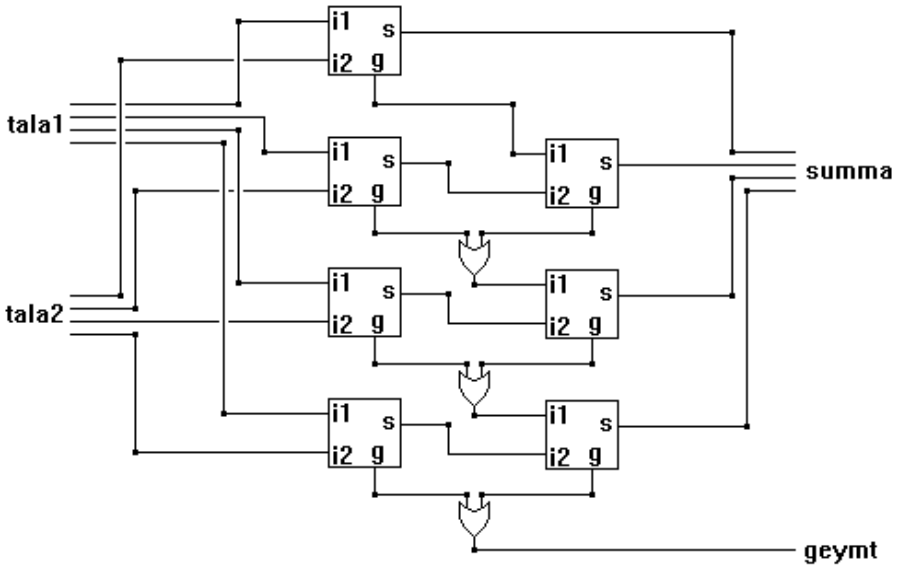
Þar sem tölur í tvíundakerfi eru myndaðar úr tveim táknum er hægt að láta sannföll samsvara reikniaðgerðum í tvíundakerfi. Úr *og-*, *eða-*, og *ekki-rásum* er til dæmis hægt að mynda rásir sem leggja saman tvíundakerfistölur.

Myndin sýnir rás sem samsvarar samlagningartöflu fyrir tvíundatölur. Tölurnar koma inn um inntökin i1 og i2. Summa þeirra kemur út um s og það sem er geymt kemur út um g.



Þessi rás leggur saman tvær eins stafs tvíundatölur. Hún er kölluð *hálfsamleggjari*. Hægt er að búa til rás sem leggur saman stærri tölur með því að tengja saman marga hálfsamleggjara og *eða-rásir.* Ef við látum mynd eins og hér er til hægri standa fyrir einn *hálfsamleggjara* getum við teiknað samleggjara fyrir 4 stafa tvíundatölur eins og gert er á myndinni hér að neðan.





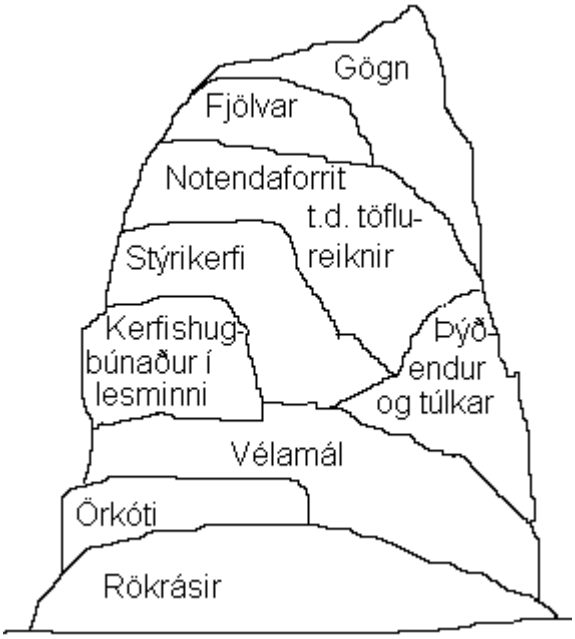
Verkefni 2.3.b

Sannreyndu að rásin á myndinni leggi í raun og veru saman tvær 4 stafa tvíundatölur.

Frá hálfleiðurum til hugarstarfs

Úr kísli og öðrum hálfleiðurum er hægt að smíða smára og úr smárunum tæki eins og *og-*, *eða-* og *ekki-rásir*. Úr þessum rásum er svo hægt að smíða minniseiningar, reikni- og rökverk og aðra vélarhluta tölvu.

Þegar tölvan hefur verið smíðuð er hægt að mata hana á forritum. Þessi forrit eru í raun runur af tvenns konar táknum sem vistuð eru í minni tölvunnar. Við getum litið á þessi tákn sem tvíundatölur og túlkað þau sem tölustafina 0 og 1. Við getum líka litið á hvern bita minnis sem yrðingu og túlkað táknin sem sanngildi. Báðar þessar túlkanir eru réttar svo langt sem þær ná en hvorug gagnast vel til að skilja hvernig flókin forrit vinna.



Að ætla sér að finna út hvað tölva er að gera með því að skoða inntak og úttak einstakra *og-*, *eða-* og *ekki-rása* í henni er eins og að reyna að grafast fyrir um hvað fólk er að sýsla með því að telja upp hvað gerist í hverri frumu líkamans. Þegar tölva er að vinna eitthvert verk, eins og til dæmis að reikna dæmi sem hefur verið sett upp í töflureikni, er nær engin leið að skilja hvað hún er að gera með því að hugsa bara um rökrásir. Nær er að skoða gögnin sem hún er að vinna úr og formúlurnar sem slegnar hafa verið inn í reiti töflunnar. Ef til vill þarf líka að athuga fjölva (þ.e. smáforrit) sem töflureiknirinn keyrir.

Eigi að kafa dýpra getur þurft að athuga hvernig töflureiknisforritið er gert og túlkurinn sem er innbyggður í það og keyrir fjölvana.

Rétt eins og möguleikar tölvunnar á að vinna úr gögnunum í töflunni hvíla á eiginleikum töflureiknisins byggir töflureiknirinn á undirstöðum sem eru neðar í stigveldinu því hann er forrit sem nýtir þjónustu stýrikerfis og er annað hvort á vélamáli eða keyrt af túlk. Sé hann á vélamáli eins og flest viðamikil forrit þá er hann væntanlega þýddur af öðru forritunarmáli með þýðanda. Stýrikerfið notar svo ýmsan kerfishugbúnað í lesminni tölvunnar. Þessi kerfishugbúnaður er byggður ofan á vélamálið sem gjörvinn vinnur eftir.

Það er langt og flókið stigveldi frá formúlu í töflureikni sem tölva vinnur eftir niður í rökrásirnar sem hún er byggð úr. Þarna á milli geta verið mörg lög af hugbúnaði.

Víst er hægt að lýsa tölvukerfi með hugtökum Boole-algebru og rökrásafræða, en hætt er við að þá sjáist ekki í skóginn fyrir trjánum. Það er líka hægt að einblína á tölur, vélamálsskipanir og reikniaðgerðir. Næsta skref væri svo að nota hugtök úr forritun eins og breytur, aðferðir og atburði. Ef tölvan keyrir töflureikni og er að vinna eftir formúlum er líka hægt að nota hugtök úr heimi viðfangsefnisins og segja t.d. að hún sé að reikna jöfnu bestu línu gegnum safn af punktum og útskýra svo hvernig hún fer að því með tilvísun til þeirrar stærðfræði sem er notuð við þetta verk. Að þessu leyti eru tölvukerfi eins og lífverur, það er hægt að lýsa þeim á marga ólíka vegu sem allir eiga

rétt á sér því hver þeirra beinir sjónum okkar að vissum eiginleikum kerfisins sem ekki er gott að átta sig á eftir öðrum leiðum.

Í töflunni hér að neðan eru annars vegar talin upp sex ólík hugtakakerfi sem hægt er að nota til að fjalla um tölvur. Hins vegar er svipuð upptalning fyrir lífverur. Í þessum kafla hefur verið fjallað ofurlítið um miðhæðirnar í stigveldinu, þ.e. rökrásir og tvíundakerfi.

Tölvukerfi	**Lífvera**
Verkefni sem tölvur leysa, leikreglur o. þ. u. l.	Vistfræði og félagsfræði
Forritun	Atferlis- og sálarfræði
Vélamál og tvíundakerfi	Líffæra og lífeðlisfræði
Rökrásir	Frumulíffræði
Rafeindatækni og eðlisfræði hálfleiðara	Efnafræði
Öreindafræði	Öreindafræði

Til upprifjunar

1. Hvað merkja táknin: &, $\sim$, $\vee$, $\wedge$, $\supset$, $\Rightarrow$, $\mid$, $\neg$.

2. Hvað eru:
   Hálfsamleggjari, hexadesímalkerfi, klifun, lögmál de Morgans, mótsögn, nand-rás, nor-rás, samræmt eða form, sætisritháttur, tvíundakerfi, XOR.

3. kafli

Bygging tölvu

3.1 Helstu vélarhlutar

Gjörvi, minni, braut, klukka og tengibúnaður

Sé tölva tekin í sundur getur að líta nokkra kubba og víra eða rásir sem tengja þá saman. Innan í hverjum kubbi er kísilflaga sem inniheldur flókið net rökrása. Þessar rásir eru svo fíngerðar að þær sjást ekki með berum augum.

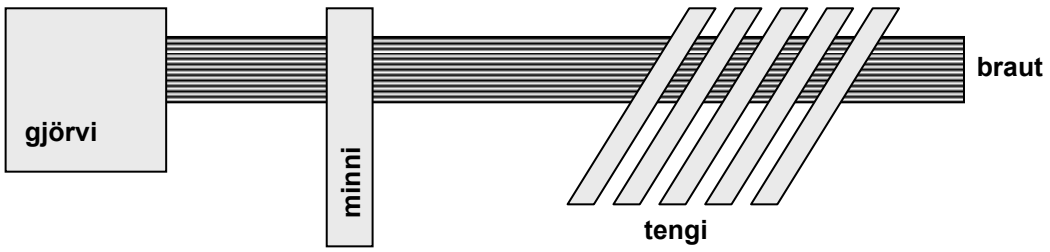
Mikilvægustu vélarhlutar tölvu eru *gjörvi*, *minni*, *braut* og *tengibúnaður* af ýmsu tagi sem stjórnar samskiptum tölvunnar við tæki eins og skjá, lyklaborð, diska, mús, prentara eða aðrar tölvur.

Minnið geymir forritin, sem vélin vinnur eftir, og gögnin sem þeim er beitt á. Gjörvinn framkvæmir skipanirnar í forritunum. Brautin er gerð úr mörgum samhliða rafrásum sem tengja gjörvann við minni og tengibúnað. Brautin liggur yfirleitt á plötu sem kallast móðurborð. Áfastir henni eru einnig kubbar með rökrásum sem stjórna umferð um brautina. Oftast eru minnið og gjörvinn í einingum sem smellt er í innstungur á móðurborðinu. Í sumum vélum eru minni og gjörvi þó lóðuð föst við það.

Nokkuð er misjafnt hvaða tengibúnaður er áfastur móðurborði. Í mörgum einmenningstölvum er tengibúnaður fyrir lyklaborð, mús, skjá, diska og fleiri tæki innbyggður í móðurborðið en stundum eru sum þessara tengja á plötum sem smellt er í raufar á því. Móðurborð í einmenningstölvum hafa yfirleitt nokkrar raufar sem hægt er að nota til að bæta tengibúnaði við vélina.

Í nýlegum tölvum er gjörvinn fólginn í einni kísilflögu. Slíkur gjörvi kallast *örgjörvi* til aðgreiningar frá eldri og stórskornari gjörvum. Minnið er oftast í nokkrum kubbum sem sitja saman á bretti.

Myndin sýnir hvernig helstu hlutar tölvu tengjast. Frá því byrjað var að framleiða tölvur fyrir um það bil hálfri öld hafa þær flestar haft í megindráttum þessa sömu byggingu sem oft er kennd við von Neumann og kölluð von Neumann arkítektúr.



Þegar tölva er í gangi sækir gjörvinn skipun í minnið, framkvæmir hana, sækir svo næstu og framkvæmir hana. Skipanirnar berast gjörvanum eftir brautinni og til að framkvæma þær getur þurft að flytja gögn eftir þessari sömu braut milli gjörva og minnis eða einhvers tengibúnaðar. Hraða þessarar atburðarásar er stjórnað af eins konar takt-

mæli sem kallast *klukka* og er oftast innbyggður í örgjörvann. Klukkuhraði tölvu er mældur í megariðum (sem einnig kallast megaHerz, skst. MHz). 1 megarið (þ.e. 1MHz) er milljón taktar á sekúndu.

Hraði tölvu

Í gjörva sem er 1000MHz slær klukkan eittþúsund milljónir ($=10^9$) takta á sekúndu. Það er öðru fremur þessi ævintýralegi hraði sem gerir tölvur slík furðuverk sem þær eru. Hver einstakur atburður sem gerist inni í vélinni er afskaplega einfaldur. Mynstur úr tvenns konar merkjum eða rafboðum (sem vani er að túlka sem tölur á formi tvíundakerfis) færast milli vélarhluta og á þeim eru framkvæmdar einfaldar reikniaðgerðir. Til að vinna flóknari verk eins og að birta einn bókstaf á skjá eða lesa innslátt frá lyklaborð þarf tölva að framkvæma fjölda skipana. Okkur virðist stafur birtast á skjá í sömu andrá og slegið er á lyklaborðið vegna þess að skipanirnar eru framkvæmdar á sekúndubroti.

Ef klukka í tölvu slægi 1 takt á sekúndu og keyrsla forrita væri nógu hæg til að við gætum séð framvinduna skipun fyrir skipun þætti okkur trúlega lítið til koma. Það sem tekur 1000MHz tölvu eina sekúndu að gera tæki þá 10^9 sekúndur sem eru um það bil 30 ár. Það liðu margir dagar frá því smellt er á hnapp á lyklaborðinu þar til stafur birtist á skjánum.

Allar tölvur hafa svipaða hæfileika. Þær geta unnið allt hugarstarf sem hægt er að vinna eftir nákvæmri forskrift eða algrími. (Um þetta verður fjallað nánar í 7. kafla.) Framþróun tölvutækninnar undanfarna hálfa öld hefur fyrst og fremst snúist um að skapa sífellt smágerðari, ódýrari, traustari en umfram allt hraðvirkari tölvur.

Hér á eftir eru talin upp helstu atriði sem hafa áhrif á hve hraðvirk tölva er:

- **Klukkuhraði**. Hann er eins og fyrr segir mældur í megariðum. Flestar nýlegar einmenningstölvur eru með nokkur hundruð megariða klukku. Iðntölvur og ódýrar leikjatölvur ganga hægar.
- **Bygging gjörva**. Það veltur á byggingu gjörva hvað mikið gerist við hvert klukkuslag. Gjörvi sem er byrjaður að sækja næstu skipun í minni meðan hann er að framkvæma skipunina á undan vinnur hraðar en gjörvi sem lýkur einni skipun áður en hann byrjar á þeirri næstu. Gjörvi sem sækir skipun og lýkur við að framkvæma hana á 1 til 4 klukkuslögum er að jafnaði afkastameiri en gjörvi sem þarf 10 til 20 klukkuslög fyrir hverja skipun.
- **Breidd brautar** ræður því hvað hægt er að flytja mikið af gögnum milli vélarhluta við hvert klukkuslag. Ódýrustu iðntölvur hafa 8 bita (víra) til gagnaflutninga og geta því aðeins flutt átta stafa tvíundatölu við hvert klukkuslag. Slík vél þarf nokkur slög til að koma einni vélamálsskipun frá minni til gjörva. Flestar nýlegar einmenningstölvur og vinnustöðvar hafa 64 bita gagnabraut.
- **Stærð minnis**. Þegar tölva vinnur með stærri forrit eða viðameiri gögn en svo að allt rúmist í senn í minni hennar nýtir hún yfirleitt hluta af hörðum diski fyrir sýndarminni (þ.e. sem viðbót til að geyma það sem ekki rúmast í minni þá stundina). Vélin þarf þá oft að flytja gögn milli disks og minnis og iðulega tefur þetta mjög fyrir henni enda er mun seinlegra að lesa og skrifa á disk en í minni. Vél sem hefur stórt minni verður mun síður fyrir töfum af þessu tagi en vél með lítið minni. Oft er því hægt að auka hraða tölvu mjög mikið með því að stækka í henni minnið.
- **Flýtiminni**. Flestir gjörvar geta unnið hraðar en svo að ódýrar minniseiningar hafi við að mata þá á gögnum og skipunum. Minnið er því flöskuháls í mörgum tölvukerfum. Til að bæta úr þessu er sett *flýtiminni* (á ensku *cache*) milli gjörvans og aðalminnisins.

Það er iðulega mun minna en aðalminnið en hraðvirkara enda byggt úr dýrari einingum. Skipanir og gögn sem flutt eru frá minni til gjörva lenda í flýtiminni svo þar er iðulega afrit af innihaldi þess hluta minnisins sem nýlega hefur verið lesið úr. Í hvert sinni sem gjörvinn les úr minni nýtir hann flýtiminnið ef það sem sækja skal er þar, en aðalminnið að öðrum kosti. Flest forrit innihalda slaufur þar sem sömu skipanir eru endurteknar aftur og aftur og innihald sömu breytanna uppfært í hvert sinn. Slaufurnar eru tímafrekustu hlutar flestra forrita svo flýtiminni sem rúmar heila slaufu ásamt breytunum sem hún vinnur með getur aukið mjög afkastagetu tölvu.

*

Hér hafa verið talin fimm atriði sem hafa mikil áhrif á afkastagetu tölva. Fleira mætti tína til. Hraði á hörðum diskum hefur til dæmis veruleg áhrif á hve fljótt tölvur vinna. Við ýmsa tölvuvinnu munar líka mikið um nettengingar, hve mikið þær geta borið af gögnum á hverri tímaeiningu.

Afköst tölvu eru stundum mæld í MIPS. Þetta er ensk skammstöfun fyrir *Millions of instructions per second* sem þýðir *milljónir skipana á sekúndu*.

Fram til þessa hefur afkastageta tölva aukist ár frá ári og sér ekki fyrir endann á þeirri þróun. Einhver efri mörk hljóta þó að vera á því hvað vél með einni braut og einum gjörva getur unnið hratt því boð geta ekki borist á meira en ljóshraða (sem er um það bil $3 \cdot 10^8$ m/s) og sameindabygging efnisins setur því mörk hvað vélarhlutarnir geta orðið smágerðir.

Eigi gögn að komast eftir 0,1m langri braut á einu klukkuslagi verða að líða a.m.k. $0{,}1\text{m} \div 3 \cdot 10^8\text{m/s} \approx 3{,}33 \cdot 10^{-10}$ sekúndur milli slaga. Klukkuhraðinn má þá ekki vera meira en 3000MHz. Eigi að smíða braut sem gengur á meiri hraða en þetta verður að hafa hana styttri.

Ritminni og lesminni

Þegar talað er um minni tölvu er oftast átt við *vinnsluminni*, þ.e. *ritminni* (á ensku *RAM*, *random access memory*) sem í venjulegri einmenningstölvu er yfirleitt nokkrir tugir *megabæta* en getur skipt *gígabætum* í dýrari tölvum. Auk ritminnisins hafa tölvur *lesminni* (á ensku *ROM*, *read only memory*) sem yfirleitt er nokkrir tugir *kílóbæta* og geymir hluta af kerfishugbúnaði vélarinnar. Lesminni hefur óbreytanlegt innihald og tæmist ekki þó slökkt sé á tölvunni. Gjörvinn getur lesið úr því en ekki skrifað í það. Hann getur hins vegar breytt innihaldi ritminnisins á alla mögulega vegu og það tæmist þegar slökkt er á vélinni.

Verkefni 3.1.a

Finndu út:
Hvað gjörvinn í tölvunni sem þú notar heitir;
Hvað hann keyrir á margra megariða hraða;
Hvað gagnabrautin í henni er breið;
Hvað minnið er stórt;
Hvað vélin framkvæmir margar skipanir á sekúndu.

Bitar bæti og orð

Stærð minnis er oftast mæld í *bætum*, *kílóbætum*, *megabætum* og *gígabætum*.

- *Gígabæti* = 2^{10} (=1024) megabæti = 2^{20} kílóbæti = 2^{30} bæti.
- *Megabæti* = 2^{10} (=1024) kílóbæti = 2^{20} bæti.
- *Kílóbæti* = 2^{10} (=1024) bæti.
- *Bæti* er 8 bitar.

Biti er tæki sem getur innihaldið tvö mismunandi gildi sem oftast eru táknuð með 0 og 1, þ.e. tölustöfunum sem notaðir eru í tvíundakerfinu, en stundum líka með *satt* og *ósatt*.

Þar sem 1 bæti er 8 bitar og hver biti rúmar einn tölustaf í tvíundakerfi getur hvert bæti í minni tölvu geymt tölur á bilinu frá 00000000_2 til 11111111_2 þ.e. allar tölur milli 0 og FF_{16} (= 255_{10}). Þetta eru 256_{10} mismunandi tölur.

Auk mælieininganna *biti* og *bæti* er stærð minnis stundum mæld í *orðum*. Eitt *orð* er sá bitafjöldi sem gjörvinn meðhöndlar í senn. Það er misjafnt eftir tölvugerðum hvað *orð* er stórt. Hafi tölva 32 bita gagnabraut og 32 bita gisti er eðlilegast að hún noti 32 bita orð en séu gagnabrautin og gistin 64 bita er hvert orð líkast til 64 bitar eða 8 bæti.

Brautin

Braut í tölvu skiptist í þrjá hluta sem kallast *gagnabraut*, *vistfangsrás* og *stjórnbraut*. Hvert hólf, eða bæti, í minninu hefur sitt númer sem kallast *vistfang*. Vistfang fyrsta bætis, eða hólfs, í minninu er 0, þess annars 1, þess þriðja 2 o.s.fr. Hafi tölva t.d. 128 megabæta minni er fyrsta bætið númer 0, næsta númer 1 og það síðasta númer 2^{27} - 1.

Gjörvinn flytur gögn milli sín og minnisins með því að senda viðeigandi vistfang eftir vistfangsrásinni og boð eftir stjórnbrautinni sem segja til um hvort skrifað skal í minnið eða lesið úr því. Gögnin sjálf eru svo flutt eftir gagnabrautinni.

Til þess að átta okkur betur á því sem fer um þessar þrjár rásir skulum við taka sem dæmi að gjörvi sendi töluna 01110011_2 í vistfang númer 1040. Fyrst sendir hann tölu eftir stjórnbrautinni sem táknar að skrifa skuli í minnið. Jafnframt sendir hann töluna 1040 eftir vistfangsrásinni. Þessi boð opna leið eftir gagnabrautinni milli gjörvans og minnishólfs númer 1040 og munstrið 01110011 rennur eftir gagnabrautinni og sest í minnishólfið.

Í þeim tölvum sem hafa 8 víra í gagnabraut er hægt að flytja 8 bita, eða 1 bæti, í senn milli hinna ýmsu vélarhluta með því að setja einn bita á hvern vír. Til að flytja munstrið 01110011 er þá hleypt straumi á víra númer 1, 2, 3, 6 og 7 en ekki á víra númer 0, 4 og 5.

Sagt er að gagnabraut sem hefur 8 víra sé 8 bita. Gagnabraut með 16 víra er sögð vera 16 bita og 16 bita gagnabraut getur flutt 2 bæti í senn milli hinna ýmsu vélarhluta. 32 bita gagnabraut getur flutt 4 bæti í senn og sé hún 64 bita ber hún 8 bæti í senn.

Vistfangsrásin ber vistfang á sama hátt og gagnabrautin ber gögn. Þau eru táknuð eins og tölur í minni tölvunnar. 8 bita vistfangsrás getur því borið 256 mismunandi númer. En minni flestra tölva er meira en 256 bæti svo þörf er á breiðari vistfangsrás. Því breiðari sem rásin er því fleiri mismunandi númer getur hún flutt. Við hvern vír sem bætt er við tvöfaldast númerafjöldinn. (Af hverju skyldi hann tvöfaldast?). Algengt er að vistfangsrás í einmenningstölvu sé 32 bita breið. Það hve stórt minni tölva getur notað ræðst af breidd vistfangsrásar.

Hafi tölva 32 bita vistfangsrás þá getur hún nýtt 2^{32} bæta (= 4 gígabæta) minni ef hún á þess kost að vísa í einstök bæti. Geti gjörvinn hins vegar aðeins sótt og vistað heil 32 bita (= 4 bæta) orð þá er nóg að númera fjórða hvert bæti og 32 bita vistfangsrásin dugar til að nýta 4×2^{32} bæta (= 16 gígabæta) minni.

Minnisrýmd og sýndarminni

Flestar tölvur hafa minna vinnsluminni en breidd vistfangsrásar leyfir. Til dæmis geta *Intel Pentium* örgjörvar nýtt 64 gígabæta minni. En vélar með *Pentium* örgjörva hafa yfirleitt ekki nærri því svo stórt minni. Forrit sem keyrð eru á *Pentium-vélum* geta þó yfirleitt vísað í vistföng eins og minnið í vélinni væri mun stærra en það er í raun og veru. Þetta er vegna þess að flestar tölvur nota hluta af hörðum diski sem sýndarminni. Tæknin sem notuð er við þetta byggist á því að skipta minninu í *síður*, sem gætu t.d. verið 16 kílóbæti hver. Hugsum okkur að vinnsluminnið sé t.d. 64 megabæti (= 2^{26} bæti) en forrit vísi í vistfang með hærra númeri t.d. 2^{27}. Það sem gerist er að ein síða af minninu er skrifuð í „sýndarminnið“ á diskinum og síðan sem geymir bæti með vistfang 2^{27} er sótt af diskinum í hennar stað. Vistföngum á þessum hluta minnisins er um leið breytt þannig að númerið 2^{27} vísi þangað.

Verkefni 3.1.b

Hvað þarf breiða vistfangsrás til að nýta 128 megabæta minni?
Hvað er hægt að mynda mörg mismundandi 32 bita orð?

Stigveldi minnis

Tegund minnis	Stærð í dæmigerðri einmennings-tölvu
Gisti í gjörva	$10^1 - 10^2$ bæti
Flýtiminni	$10^{15} - 10^{20}$ bæti
Vinnsluminni	$2^{25} - 2^{28}$ bæti
Harður diskur	$2^{32} - 2^{36}$ bæti
Geisladiskar	$2^{29} - 2^{34}$ bæti

Forritarar geta látið sem öll vistföng vísi á hólf í vinnsluminni tölvunnar þótt ef til vill sé aðeins hluti þeirra þar í senn, hluti í *sýndarminni* á diski.

Eins og nefnt hefur verið hafa flestar tölvur flýtiminni auk hins venjulega vinnsluminnis. Flýtiminnið er að því leyti eins og sýndarminnið að forritarar (aðrir en þeir sem skrifa stýrikerfi) geta látið sem það sé ekki til, flutningur í það og úr því fer fram sjálfkrafa bak við tjöldin, forritin vísa aðeins á vistföng eins og öll gögn og allar skipanir væru í minni sem er ein hala-rófa af bætum, númeruð frá 0 og upp úr. Stýrikerfið og kerfishugbúnaðurinn í les-minninu sjá um að tölva hagi sér eins og hún sé mun einfaldari að gerð en rafeindavirkið í henni er í raun og veru.

Í töflunni hér að ofan er yfirlit yfir helstu tegundir minnis í röð eftir hraða. Klukkuhraði gjörvans stjórnar því hve hratt er hægt að uppfæra gögn í gistum hans, flutningur í og úr flýtiminni tekur ögn lengri tíma, vinnsluminnið er svo enn hægvirkara o.s.fr.

Ekki er nóg með að hluti af diski geti verið *sýndarminni*. Það er líka hægt að nota hluta af vinnsluminni fyrir *sýndardisk* og vista í því skrár rétt eins og á diski. En við skulum láta þetta liggja milli hluta. Yfirleitt er allt það minni sem forrit vísa í með vistfangi einfaldlega kallað *minni* þótt hluti þess sé ef til vill staddur í flýtiminni um

stund og annar hluti sé á sama tíma aðeins til sem sýndarminni á diski. Allt það sem geymir skrár er kallað *gagnageymslur* hvort sem það er á hörðum diski, á geisladiskum eða í ritminni sem er nýtt sem sýndardiskur.

Gjörvi

Gjörvi (á ensku *CPU*, *central processing unit*) eða *miðverk* er mikilvægasta og flóknasta eining hvers tölvukerfis. Nýlegir gjörvar eru samsettir úr milljónum smára og mjög flóknu neti af rafrásum. Þótt þeir séu fólgnir í einni kísilflögu er hægt að skipta þeim í nokkra vélarhluta. Þeir helstu eru *gisti*, *reikni- og rökverk* (á ensku *ALU, arithmetic logic unit*) og *stýriverk* (á ensku *control unit*). Margir gjörvar hafa auk þessa innbyggt flýtiminni og sérstakt reikniverk fyrir kommutöluaðgerðir (á ensku *FPU, floating point unit*).

Gisti (á ensku *register*) eru í raun minnishólf til að geyma þau gögn sem gjörvinn vinnur við þá stundina. Hvaðeina sem hann sækir í minnið er flutt í gisti og allt sem gjörvinn sendir eftir brautinni er sent úr einhverju gisti. Auk nokkurra *gagnagista* til almennra nota hefur hver gjörvi sérhæfð gisti, meðal þeirra mikilvægustu eru *merkjagisti*, *skipanagisti* og *vistfangsgisti*.

Vistfangsgistið geymir vistfang þess minnishólfs sem næst skal sækja skipun til. Númerið í því hækkar sjálfkrafa í hvert sinn sem skipun er framkvæmd. *Skipanagisti* geymir skipunina sem gjörvinn er að framkvæma þá stundina. Um leið og því er lokið er næsta skipun sótt og þannig koll af kolli því innihald vistfangsgistisins hækkar í hvert sinn sem skipun er framkvæmd.

Merkjagistið geymir upplýsingar um útkomu reikni- og rökaðgerða. Í dæmigerðum gjörva er til dæmis einn biti í merkjagisti sem fær gildið 1 ef samanburður á innihaldi tveggja annarra gista leiðir í ljós að þau innihaldi sömu tölu og 0 ef samanburðurinn leiðir í ljós að þau innihaldi ólíkar tölur. Aðrir bitar í merkjagistinu geyma svo upplýsingar um hvort summa eða margfeldi tveggja annarra gista rúmast í því þriðja eða hvort hún er jákvæð eða neikvæð.

Gisti til almennra nota hafa yfirleitt sömu breidd og gagnabraut tölvunnar þannig að sé gjörvi gerður fyrir 32 bita gagnabraut þá hefur hann 32 bita gisti. Hér eru *Pentium* gjörvar þó undantekning því þeir hafa 64 bita gagnabraut en 32 bita gisti.

Þegar tölva er í gangi rúllar starfsemi gjörvans stöðugt sama hring í takt við slátt klukkunnar: Skipun er sótt úr minni í skipanagisti; stýriverkið túlkar hana og virkjar viðeigandi hluta reikni og rökverks til að framkvæma skipunina, útkoman lendir svo í einhverju gistinu.

Einfaldir gjörvar ljúka einni skipun áður en sú næsta er sótt. Flóknari gjörvar hafa margar á færibandinu í senn, sækja eina meðan önnur er framkvæmd og niðurstöðu þeirrar þriðju skilað í gisti. Sumir gjörvar (eins og t.d. *Intel Pentium*) hafa jafnvel tvöfalt reikniverk og geta því framkvæmt tvær skipanir úr forriti samtímis þegar ekki er þörf á að nota útkomuna úr þeirri fyrri sem inntak fyrir þá seinni.

Vélamál

Skipanirnar sem gjörvi hlýðir eru á svokölluðu *vélamáli*. Hver tegund gjörva hefur sitt sérstaka vélamál svo forrit sem hægt er að keyra á vél með einni gerð gjörva er ekki hægt að keyra á tölvu með öðru vísi gjörva.

Þótt til séu mörg og ólík vélamál eiga þau ýmislegt sameiginlegt.[1] Orðin eru yfirleitt 8, 16 eða 32 bita munstur (þ.e. runur úr táknunum 0 og 1) og setningar eru myndaðar úr nokkrum slíkum munstrum þar sem það fyrsta er skipun en þau sem á eftir koma vistföng, tölur eða annað sem skipuninni skal beitt á. Hver skipun mælir fyrir um afar einfalda aðgerð. Það er hægt að skipta þeim í 3 meginflokka:

1. **Gildisgjöf, afritun og flutningur**. Einstakar skipanir geta sagt gjörva að flytja bæti eða orð frá tilteknum stað í minni í eitthvert gagnagisti eða milli tveggja gagnagista eða frá gagnagisti til vistfangs í minni. Einnig er hægt að rita tilteknar tölur í gisti eða vistfang í minninu.
2. **Reikni og rökaðgerðir**. Skipanir af þessu tagi láta gjörva framkvæma einhverjar aðgerðir á innihaldi gagnagista eða vistfangs í minni. Þessar aðgerðir geta til dæmis verið samlagning, margföldun, samanburður á innihaldi tveggja gista eða rökaðgerðir eins og *ekki* eða *og*. Þær eru framkvæmdar af reikni- og rökverkinu og útkoman lendir yfirleitt í einhverju gagnagisti. Útkomur slíkra aðgerða hafa líka oft áhrif á innihald merkjagistis t.d. þannig að biti fái gildið 1 ef samanburður leiðir í ljós að tvö gisti hafi sama innihald en 0 ef innihald þeirra er ólíkt.
3. **Hlaup og köll**. Skipanir í þessum flokki láta gjörva breyta innihaldi vistfangsgistis. Sumar þeirra láta hann breyta því skilyrðislaust en aðrar segja honum að breyta því þá aðeins að tiltekinn biti í merkjagisti sé hlaðinn. Þannig er til dæmis hægt að segja gjörva að hækka númerið í vistfangsgistinu um 100 ef tiltekinn biti í merkjagisti er hlaðinn. Með þessu móti er mögulegt að láta gjörva framkvæma skipanir í annarri röð en þær standa í minninu og hlaupa til og frá í forriti á alla mögulega vegu.

   Með skipunum af þessu tagi er hægt að forrita slaufur (þ.e. endurtekningu) einhvern veginn svona:

```
1000  Setja 100 í gisti A.
1001  Setja 0 í gisti B.
1002  Bæta 1 við gisti B.
1003  Láta bita n í merkjagisti verða 1 ef A = B.
…
1006  Setja 1002 í vistfangsgisti ef biti n í merkjagisti er ekki 1.
```

Hér eru sett íslensk orð í stað vélamálsskipana og tölurnar til vinstri eiga að segja hvar í minninu skipanirnar eru. Við hugsum okkur að vistföng 1004 og 1005 innihaldi einhverjar skipanir (þó þær séu ekki sýndar hér) og að biti n í merkjagisti sé upphaflega stilltur á 0. Þessi forritsbútur endurtekur 100 sinnum skipanirnar frá 1004 til 1005. Skipunin með vistfang 1006 er svokallað hlaup (eða stökk, á ensku *jump*).

Einn mikilvægur flokkur skipana sem breyta innihaldi vistfangsgistis eru köll (eintala kall) á undirforrit. Hvert slíkt undirforrit endar á skipun um að snúa til baka (á ensku *return*). Þegar kallað er á undirforrit er númerið í vistfangsgistinu sett í geymslu og þegar næst er gefin skipun um að snúa til baka er það endurheimt. Ef við hugsum okkur að vistfang 10000 geymi upphaf skipanarunu sem sækir tölu úr gisti C og ritar hana á skjáinn þá skrifar þetta forrit tölurnar frá 1 upp í 100 á skjá tölvunnar.

[1] Um vélamál verður aftur fjallað í kafla 8.2.

1000 Setja 100 í gisti A.
1001 Setja 0 í gisti B.
1002 Bæta 1 við gisti B.
1003 Láta bita n í merkjagisti verða 1 ef A = B.
1004 Afrita gisti B í gisti C.
1005 Kalla á 10000.
1006 Setja 1002 í vistfangsgisti ef biti n í merkjagisti er ekki 1.

Það kann að virðast dálítið ótrúlegt að skipanir úr þessum þrem flokkum dugi til að láta tölvur vinna hvaðeina sem þær gera. En flókin forrit sem tefla skák, spila lög, teikna myndir eða leita að gögnum á diski eru samt samsett úr vélamálsskipunum sem hver um sig lætur tölvuna gera ósköp lítið. En með því að framkvæma milljónir slíkra skipana á hverri sekúndu getur tölvan unnið flókin verk á augabragði.

Tölva gangsett

Um leið og kveikt er á tölvu fer eitthvert númer í vistfangsgistið, yfirleitt 0. Í tölvum af sömu gerð er þetta alltaf sama númerið. Þetta er vistfang þess staðar í minninu sem geymir fyrstu skipunina sem framkvæma skal. Það er vitaskuld í lesminni tölvunnar því ritminnið er tómt þegar vélin er ræst. Þegar þessi fyrsta skipun hefur verið framkvæmd er sú næsta sótt og þannig koll af kolli þar til kemur að skipun sem segir gjörvanum að breyta númerinu í vistfangsgistinu, þá er stokkið til í forritinu og haldið áfram frá þeim stað.

Þessar fyrstu skipanir sem gjörvinn hlýðir eftir að kveikt er á tölvunni láta hann yfirleitt framkvæma nokkur próf sem skera úr um það hvort vélin sé í lagi og sækja síðan *stýrikerfi* af diski (eða e.t.v. geisladiski eða disklingi), hlaða því inn í ritminnið og breyta svo númerinu í vistfangsgistinu þannig að vísað sé á fyrstu skipun stýrikerfisins. Eftir það er tölvan undir stjórn stýrikerfisins. Það sér svo um að hlaða öðrum forritum í minni. Um stýrikerfi verður fjallað nánar í kafla 5.2.

RISC og CISC

Nýlegir örgjörvar skiptast í tvo meginflokka sem kallaðir eru *RISC* og *CISC*. Þetta eru skammstafanir sem standa fyrir ensku orðasamböndin *reduced instructions set computer* og *complex instructions set comuter*.

Vélamál CISC-gjörva er flóknara en vélamál RISC-gjörva og skipanirnar eru ekki keyrðar beint á vélbúnaðinn heldur túlkaðar af skipanatúlki í stýriverki gjörvans sem umritar þær á einfaldara mál, svokallaðan *örkóta* (á ensku *microcode*). Ein vélamálsskipun getur orðið að nokkrum skipunum á örkóta. Örkótinn leikur beint á rökrásir gjörvans. Í dæmigerðum CISC-gjörva tekur keyrsla vélamálsskipana mislangan tíma. Flóknustu skipanirnar verða að alllangri örkótaromsu, þær einföldustu að einni skipun á örkótanum og í sumum CISC-gjörvum eru einföldustu vélamálsskipanirnar ekki túlkaðar heldur keyrðar beint á rökrásir gjörvans.

Vélamál RISC-gjörva hefur fremur fáar og einfaldar skipanir en á móti kemur að gjörvinn inniheldur rökrásir sem framkvæma hverja skipun svo ekki er þörf á neinum skipanatúlki. Í flestum RISC-gjörvum taka allar vélamálsskipanir sama tíma.

Ef RISC og CISC-gjörvi ganga á sömu klukkutíðni þá framkvæmir RISC gjörvinn fleiri skipanir á tímaeiningu, því hann þarf færri klukkuslög á hverja skipun. Á móti

kemur að sum verk sem hægt er að vinna með einni vélamálsskipun í CISC-gjörva útheimta margar vélamálsskipanir í RISC-gjörva.

Sem dæmi um algenga RISC-gjörva má nefna, *PowerPC* sem er í nýlegum Macintosh tölvum, *Alpha* frá DEC, og *RISC 6000* frá IBM. Algengastu CISC-gjörvarnir eru ýmsar útgáfur af *Pentium* sem framleiddir eru af fyrirtækinu Intel og meðal annars notaðir í PC-tölvur.

Öðru vísi hönnun – margir gjörvar

Þótt flestar tölvur sem smíðaðar hafa verið til þessa byggi á von Neumann arkítektúr, eins og hér hefur verið lýst, hafa tölvufræðingar rannsakað ýmsa kosti á að smíða vélar með mörgum gjörvum. Ef margir gjörvar vinna hlið við hlið þá geta þeir ef til vill í sameiningu framkvæmt fleiri skipanir á tímaeiningu heldur en nokkur einn gjörvi getur. Nokkuð er um að dýrar tölvur eins og t.d. netþjónar hafi tvo eða jafnvel fleiri gjörva.

Tilraunir hafa verið gerðar til að smíða tölvur með mjög mörgum gjörvum. Einn flokkur slíkra véla er svokölluð tauganet sem eru eins konar eftirlíking af heila eða miðtaugakerfi í dýri eða manni. Tauganet byggja á mörgum einföldum *gjörvum* sem tengjast saman í net þannig að úttak frá sumum er inntak fyrir aðra. Tauganet eru forrituð með því að aðlaga hversu mikil áhrif úttak hvers gjörva hefur á viðbrögð annarra gjörva og með þessu er reynt að líkja eftir því hvernig taugakerfi í lífverum er talið bregðast við áreiti.

Tölvur með mörgum gjörvum og tauganet eru ekki einu leiðirnar til að sigrast á takmörkunum hefðbundins byggingarlags á tölvum. Ein þeirra leiða sem mestar vonir eru bundnar við er að láta margar venjulegar tölvur vinna saman. Þegar keyrslu forrits er dreift á margar vélar er talað um *dreifða vinnslu* og tilraunir til að vinna umfangsmikil verk með þeim hætti hafa færst í vöxt samhliða hraðvirkari og betri nettengingum milli tölva. Nú þegar fer fram verulega mikil dreifð vinnsla á Internetinu og í tölvunetum ýmissa fyrirtækja. Þótt samskipti milli tölva séu margfalt hægari en samskipti milli minnis og gjörva í sömu vél getur það flýtt sumum verkum mjög mikið að dreifa vinnunni á margar nettengdar tölvur.

3.2 Samband tölvu við umheiminn

Jaðartæki og tengi

Til að hægt sé að nota tölvu þarf að tengja hana við inntaks- og úttakstæki svo hægt sé að koma boðum til hennar og hún geti komið upplýsingum frá sér. Algengustu inntakstæki einmenningstölva eru lyklaborð og mús. Helstu úttakstækin eru skjár og prentari.

Tölvur eru yfirleitt tengdar fleiri vélum en eiginlegum inn- og úttakstækjum. Hér má til dæmis nefna mótöld, nettengi, og gagnageymslur á borð við harða diska og disklingadrif. Sum þessara tækja eru afar flókin og stundum inniheldur stjórnbúnaður þeirra eina eða fleiri tölvur.

Tæki sem eru tengd við tölvu (eins og mýs, lyklaborð, prentarar, skjáir, hátalarar, mótöld og hljóðnemar) kallast einu nafni *jaðartæki*.

Á venjulegri einmenningstölvu eru nokkur tengi sem hægt er að nota til að tengja margs konar jaðartæki við vélina. Helstu tegundir slíkra fjölnota tengja eru *samhliðatengi*, *raðtengi*, *USB* og *SCSI braut*.

Samhliðatengi eru stundum kölluð prentaratengi enda eru þau mest notuð til að tengja tölvur við prentara þótt ýmis önnur tæki fáist með tengisnúru fyrir samhliðatengi eins og til dæmis utanáliggjandi gagnageymslur og myndlesarar.

Raðtengi eru oftast notuð til að tengja mús við tölvu. Mótöld eru einnig oft tengd við raðtengi.

Flestar nýlegar einmenningstölvur hafa *USB* tengi og eru þau að taka við af hefðbundnum raðtengjum. Nafnið er skammstöfun á ensku orðunum *universal serial bus* (sem mætti þýða með *alhliða raðtengibraut*). Við USB er hægt að tengja halarófu af tækjum þannig að það fyrsta sé tengt við USB innstungu á tölvunni, það næsta við USB innstungu á því tæki o.s.fr. Einnig eru til USB tengibox með mörgum innstungum. Þegar slíkt tæki hefur verið tengt við USB tengi tölvu er hægt að tengja eitt jaðartæki í hverja innstungu á því og láta tölvuna nýta þau öll samtímis. Fjölmargar gerðir jaðartækja tengjast tölvu um USB, m.a. gagnageymslur, myndlesarar, prentarar og mýs.

SCSI braut er einkum notuð til að tengja tölvur við gagnageymslur. Hún er mjög hraðvirk og hentar vel sem tengibúnaður fyrir harða diska þegar þörf er á miklum hraða. Hægt er að tengja nokkur tæki í halarófu á sömu SCSI brautina.

Algengt er að PC-tölvur hafi eitt samhliðatengi, tvö raðtengi og tvö USB-tengi. Hægt er að bæta í þær spjaldi með SCSI-tengi.

Auk tengja til almennra nota hafa flestar tölvur sérhæfðan tengibúnað fyrir skjá og lyklaborð. Á hljóðkortum sem fylgja mörgum vélum eru sérhæfð tengi fyrir hátalara, hljóðnema og hljómflutningstæki.

Samband við minni

Tengibúnaður, sem tengir tölvu við jaðartæki, hefur aðgang að einhverjum hluta af minni hennar. Stundum eru minnisrásir í tengibúnaðinum sem taka yfir einhver vistföng í vinnsluminninu og stundum hefur tengibúnaðurinn beinan aðgang eftir brautinni að einhverjum hluta vinnsluminnisins.

Forrit heimta gögn frá inntakstækjum með því að lesa úr þeim hluta minnisins sem tækið tengist og þau senda boð til úttakstækis með því að skrifa í þann hluta minnisins sem tengdur er við það. Sum tæki hafa aðeins nokkur bæti af minni, önnur meira. Skjátengi þurfa til dæmis nægilega stórt minni til að vista upplýsingar um lit hvers punkts á skjánum. Í nýlegum tölvum er litur hvers punkts táknaður með 3 til 4 bætum af minni. Ef skjáupplausnin er 600·800 punktar og hver punktur fær 4 bæti þarf skjátengið 4·600·800 = 1.920.000 bæti eða næstum 2 megabæti. Fyrir meiri upplausn þarf stærra skjáminni.

Í minni tölvu er skjámynd aðeins runa af tölum. Forrit sem teiknar á skjáinn gerir því í raun ekkert annað en að hringla með tölur í skjáminninu. Tónar, sem sendir eru hljóðkorti, eru líka bara tölur í minninu og þegar lamið er á stafina á lyklaborðinu lenda tölur í þeim hluta minnisins sem tengdur er við lyklaborðstengið.

Rof

Í hvert sinn sem jaðartæki þarf á athygli gjörvans að halda sendir það honum *rof* (á ensku *interrupt*) það er að segja tölu sem táknar að sinna þurfi tiltekinni þörf þessa jaðartækis. Rof er merki sem berst gjörva eftir sérstakri rás. Hann bregst við með því að ljúka skipuninni sem hann er að framkvæma, vista síðan innihald vistfangsgistis í

vinnsluminni og sækja þangað númer sem rofið vísar á og bendir á vistfang fyrstu skipunar í forritsbút sem bregst við þörf viðkomandi jaðartækis. Keyrslu þessa búts lýkur á að gjörvinn endurheimtir númerið sem var í vistfangsgistinu þegar rofið barst.

Þegar rof berst til gjörva hverfur hann frá því verki sem hann er að vinna og framkvæmir skipanarunu, sem yfirleitt er annað hvort í lesminninu eða hluti af stýrikerfinu og segir hvernig á að bregðast við þessu tiltekna rofi. Að því búnu heldur gjörvinn áfram fyrra verki þar sem frá var horfið. Vinna hans er rofin þegar hann þarf að sinna jaðartækinu. Þess vegna er orðið *rof* notað yfir þetta.

Þegar við sláum á lykil á lyklaborði tölvu virðist okkur sem stafurinn birtist strax á skjánum. Það sem raunverulega gerist er flókin atburðarás sem hefst á því að númer berst frá lyklaborði í minni og lyklaborðstengið sendir gjörvanum rof. Það verður til þess að hann framkvæmir skipanarunu sem sækir töluna úr lyklaborðsminninu. Áður en stafurinn birtist á skjánum þarf að keyra nokkra búta úr stýrikerfinu sem m.a. para töluna við staf úr stafatöflu og færa mynd af honum í skjáminnið. Næst þegar skjátengið les skjáminnið og uppfærir myndina á skjánum (sem það gerir í tugi skipta á hverri sekúndu) birtist stafurinn þar. Þá eru ef til vill liðnir tugir millisekúndna frá því smellt var á hnappinn á lyklaborðinu. Á 10 millisekúndum framkvæmir tölva með afköst upp á 100 MIPS heilar 1.000.000 skipanir. Hún hefur varið hluta þessa tíma til að keyra skipanir sem koma stafnum frá lyklaborðinu á skjáinn en inn á milli framkvæmt nokkrar skipanir úr forritum sem eru í gangi og ef til vill svarað rofi frá einhverju öðru jaðartæki.

Til upprifjunar

1. Hvað merkja skammstafanirnar:
   CISC, MHz, MIPS, RAM, RISC, ROM, USB.

2. Hvað eru:
   Flýtiminni, gagnabraut, gjörvi, klukka, merkjagisti, reikni- og rökverk, rof, skipanatúlkur, skjáminni, stjórnbraut, vistfangsgisti, vistfangsrás, örkóti.

3. Gerðu grein fyrir 5 atriðum sem hafa áhrif á hve hraðvirk tölva er.

4. kafli

Gögn og breytur

4.1 Gögn

Munstur úr tvenns konar táknum

Meðan tölva keyrir forrit er það í minni hennar og þótt forritið sé ef til vill sífellt að skrifa í gagnageymslur og sækja ný gögn úr þeim eru upplýsingarnar, sem það vinnur með, í minni rétt á meðan.

Öll gögn í minni tölvu eru á tvíundaformi, þ.e. munstur úr tvenns konar táknum sem túlka má sem tölustafina 0 og 1. Það er lítill vandi að skilja hvernig þessi tvenns konar tákn eru notuð til að skrifa jákvæðar heilar tölur. Það er einfaldlega gert með því að skrifa tölurnar í tvíundakerfi. Öllu snúnara er að tákna neikvæðar tölur, brot, bókstafi og þaðan af flóknari gögn eins og myndir og hljóð.

Neikvæðar tölur og tvíundafylli

Nokkrar aðferðir hafa verið notaðar til að tákna neikvæðar heilar tölur. Sú einfaldasta er að túlka fyrsta bita tölunnar sem formerki, láta t.d. 1 í fyrsta sæti tákna mínus og 0 tákna plús. Með þessari aðferð er hægt geyma töluna -515 í tveim bætum (þ.e. 16 bitum) með munstrinu 1000001000000011.

Fyrsti biti hafður fyrir formerki

Sæti númer	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Vægi	+/-	16382	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Talan -515	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

Þótt þessi aðferð sé einföld og auðskilin er hún fremur lítið notuð. Algengasta aðferðin til að tákna neikvæðar tölur er að nota *tvíundafylli* (á ensku *two's complement*).

Tvíundafylli byggir á því að túlka alla bitana eins og venjulega er gert í tvíundakerfi nema þann fyrsta, hann er túlkaður sem neikvæð tala. Talan -515 er t.d. geymd í tveim bætum með munstrinu 1111110111111101. Eins og taflan sýnir jafngildir þetta munstur tölunni

-32768 + 16384 + 8192 + 4096 + 2048 + 1024 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 1 = - 515

Tvíundafylli

Sæti númer	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Vægi	-32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Talan -515	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1

Þegar tvíundafylli er notuð eru jákvæðu tölurnar skrifaðar eins og venjulega er gert í tvíundakerfi, fremsti bitinn þó hafður 0. Hægt er að breyta formerki úr plús í mínus með

því að breyta alls staðar úr 0 í 1 og úr 1 í 0 og bæta svo einum við. Tökum dæmi. Ef talan 45 er geymd í einu bæti er hún svona:

Sæti númer	7	6	5	4	3	2	1	0
Vægi	-128	64	32	16	8	4	2	1
Talan 45	0	0	1	0	1	1	0	1

Talan -45 er búin til með því að umhverfa hverjum bita, svona:

Sæti númer	7	6	5	4	3	2	1	0
Vægi	-128	64	32	16	8	4	2	1
	1	1	0	1	0	0	1	0

Og bæta svo einum við, svona:

Sæti númer	7	6	5	4	3	2	1	0
Vægi	-128	64	32	16	8	4	2	1
Talan -45	1	1	0	1	0	0	1	1

Útkoman úr þessu er -128 + 64 + 16 + 2 + 1 = -45

Með þessari aðferð er hægt að geyma heilar tölur frá -128 upp í 127 í 8 bitum, þ.e. einu bæti, og tölur frá -32768 upp í 32767 í 16 bitum og almennt gildir að í n bitum rúmast tölur frá -2^{n-1} upp í $2^{n-1}-1$. Í mörgum forritunarmálum taka heiltölubreytur gildi á þessum bilum. Í *Pascal* geta breytur af tegundinni *Integer* t.d. tekið gildi frá -32768 upp í 32767 og í forritunarmálinu *Java* eru fjórar tegundir af heiltölubreytum sem geyma tölur á formi tvíundafylli eins og taflan sýnir.

Heiltölubreytur í Java

Heiti tegundar	möguleg gildi
byte	Heilar tölur frá -128 til 127
short	Heilar tölur frá -32768 til 32767
int	Heilar tölur frá -2147483648 til 2147483647
long	Heilar tölur frá -9223372036854775808 til 9223372036854775807

Þegar tvíundafylli er notuð eru mögulegar jákvæðar tölur alltaf einum færri en mögulegar neikvæðar tölur.

Verkefni 4.1.a

Skrifaðu sem tvíundafylli: -1, -7, -30, -125.

Verkefni 4.1.b

Hvað taka breytur af tegundunum *byte*, *short*, *int* og *long* í forritunarmálinu Java mikið pláss í minni?

Kommutölur og IEEE 754

Í stærðfræðibókum er kennt að kalla mengi *náttúrulegra talna* N, mengi *heilla talna* Z, mengi *ræðra talna* Q og mengi *rauntalna* R.

- N er allar jákvæðar heilar tölur: 1, 2, 3, 4, 5…
- Z er allar heilar tölur: … -3, -2, -1, 0, 1, 2, 3, …
- Q er allar tölur sem hægt er að skrifa sem almenn brot, þ.e. á forminu a/b þar sem $a \in Z$ og $b \in Z$ og $b \neq 0$.
- R er allar tölur á talnalínunni.

Í vissum skilningi er hægt að halda því fram að minni tölvu innihaldi aldrei neitt annað en tölur úr menginu N skrifaðar í tvíundakerfi. En samt er hægt að geyma neikvæðar tölur í minni tölvu með því að búa til reglu sem parar saman tölur úr mengi heilla talna, Z, og tölur úr mengi náttúrulegra talna, N. Tvíundafylli er ein slík regla.

Með svipuðum hætti er hægt að geyma ræðar tölur í minni tölvu með því að para saman heilar tölur og tölur úr menginu Q. Ein möguleg leið væri t.d. að láta 2 bæta (16 bita) heiltölur standa fyrir brot með því að fyrstu 8 bitarnir væru teljari og þeir næstu 8 nefnari. Þá mundi 0000001000000011 tákna brotið $00000010_2/00000011_2$ sem ritað er 2/3 í tugakerfi. Þótt þessi aðferð sé einföld er hún yfirleitt ekki notuð. Aðferðin sem notuð er í nær öllum tölvum byggir á staðalformi þar sem hver ræð tala er táknuð með tveim heilum tölum, *broti* og *veldisvísi*. Í tugakerfi lítur þetta svona út:

- Tölurnar -123456 og 04 tákna $-0{,}123456 \cdot 10^4 = -1234{,}56$
- Tölurnar 123456 og -04 tákna $0{,}123456 \cdot 10^{-4} = 0{,}0000123456$

Ef við höldum okkur við tugakerfi og staðalform og skrifum fyrri töluna með 6 stöfum auk formerkis og þá seinni með 2 stöfum auk formerkis þá getum við táknað töluna 0 og tölur frá $\pm 0{,}100000 \cdot 10^{-99}$ upp í $0{,}999999 \cdot 10^{99}$. Með þessu móti er engin leið að tákna tölur með hærra tölugildi en $0{,}999999 \cdot 10^{99}$. Með því að víkja frá staðalformi og leyfa tölugildi brotsins að fara niður í 0,000001 er hægt að tákna tölur með tölugildi niður í $0{,}000001 \cdot 10^{-99}$

Milli hverra tveggja talna í Q eru óendanlega margar aðrar. En með 8 tölustöfum og tveim formerkjum er aðeins hægt að tákna $2 \cdot 10^8$ mismunandi tölur. Þessi aðferð til að tákna ræðar tölur er því tvenns konar takmörkum háð:

- Aðeins er hægt að tákna tölur á sumum hlutum talnalínunnar – tölur með mjög hátt tölugildi og tölur sem eru mjög nálægt núlli verða út undan.
- Aðeins er hægt að skrifa tölur með tilteknum fjölda markverðra stafa (6 í dæminu sem hér var tekið) svo útkomur verður að rúnna af.

Við takmörk af þessu tagi verður að una, því í minni af endanlegri stærð er aðeins hægt að setja endanlegan fjölda mismunandi gilda.

Aðferðin sem flestallar tölvur nota til að geyma ræðar tölur fylgir staðli sem heitir *IEEE 754*. Margir nýlegir örgjörvar, eins og t.d. *Intel Pentium* hafa innbyggðar reikni-aðgerðir fyrir kommutölur sem skráðar eru samkvæmt þessum staðli. Hann lýsir tveim leiðum til að skrá kommutölur. Önnur notar 32 bita, hin 64 bita. Í báðum tilvikum er fyrsti bitinn notaður fyrir formerki, sé hann 1 er talan neikvæð en jákvæð ef hann er 0.

Ræðar tölur sem vistaðar eru í 32 bitum samkvæmt *IEEE 754* kallast *kommutölur með einfaldri stafanákvæmni*. Næstu 8 bitar á eftir formerkinu eru veldisvísir sem er

fenginn með því að draga 127 frá innihaldi þeirra. Þeir 23 bitar sem eftir eru tákna brot á tvíundaformi.

Ræðar tölur sem vistaðar eru í 64 bitum samkvæmt *IEEE 754* kallast *kommutölur með tvöfaldri stafanákvæmni*. Næstu 11 bitar á eftir formerkinu eru veldisvísir sem er fenginn með því að draga 1023 frá innihaldi þeirra. Þeir 52 bitar sem eftir eru tákna brot á tvíundaformi.

Þar sem fyrsti stafur aftan við kommu í tvíundakerfistölu á staðalformi er alltaf 1 er óþarfi að vista fyrsta stafinn í brotinu. Talan 0<u>10000010</u>01100000000000000000000 táknar samkvæmt þessu brotið 5,5. (Ath. veldisvísirinn er undirstrikaður.)

- Fyrsti bitinn er 0 sem þýðir að talan er jákvæð.
- Næstu 8 bitar eru $10000010_2 = 130_{10}$ sem táknar veldisvísinn $130_{10} - 127_{10} = 3$.
- Síðast er $01100000000000000000000_2$ sem táknar $0{,}10110000000000000000000_2 = 2^{-1} + 2^{-3} + 2^{-4} = 0{,}6875_{10}$
- Talan er því $0{,}6875_{10} \cdot 2^3 = 5{,}5_{10}$.

Staðallinn gerir ráð fyrir því að talan 0 sé táknuð með því að hafa alla bita í veldisvísi og broti 0. Veldisvísir allra annarra ræðra talna er skráður með tölu milli 0 og hæsta gildis, sem fæst með því að hafa alla bita veldisvísisins 1.

Óendanlega hátt gildi er táknað með því að hafa alla bita veldisvísis 1 og alla bita í brotinu 0. Yfirleitt er þetta notað fyrir útkomu ef deilt er með 0.

Ekkert gildi eða engin tala er táknað með því að hafa alla bita veldisvísis 1 en ekki alla bita í brotinu 0. Þetta er yfirleitt notað fyrir útkomu ef reynt er að deila með óendaleika í óendanleika.

Nokkur atriði úr IEEE 754

	Einföld stafanákvæmni	Tvöföld stafanákvæmi
Bitar í formerki	1	1
Bitar í veldisvísi	8	11
Bitar í broti	23	52
Fjöldi bita alls	32	64
Veldi af 2 á bilinu	$\pm 2^{-126} - \pm 2^{127}$	$\pm 2^{-1022} - \pm 2^{1023}$
Samsvarandi veldi af 10	um $\pm 10^{-38} - \pm 10^{38}$	um $\pm 10^{-308} - \pm 10^{308}$

Verkefni 4.1.c

Hvernig á að rita í tugakerfi töluna sem samkvæmt *IEEE 754* er rituð: 1<u>10001100</u>11011000000000000000000?

ASCII og Unicode

Gjörvar í tölvum hafa yfirleitt innbyggt reikniverk til að vinna með heiltölur og kommutölur samkvæmt *IEEE 754*. Annars konar gögn eru svo byggð úr þessu. Til dæmis eru bókstafir einfaldlega táknaðir með tölum.

Tölvutæknin er að mestu upprunnin í Bandaríkjunum og fyrsti staðallinn um geymslu bókstafa í minni tölvu sem náði verulegri útbreiðslu gerði aðeins ráð fyrir þeim rittáknum sem notuð eru í ameríkskri ensku. Þessi staðall kallast *ASCII* (*American Standard Code for Information Interchange*). Samkæmt honum er hver stafur táknaður

með 7 bitum, þ.e. tölu á bilinu 0000000_2 til $1111111_2 = 127_{10}$. Séu stafir táknaðir með 7 bitum er hægt að nota 8. bita hvers bætis fyrir *tvístæðubita* (á ensku *parity bit*), þ.e. gefa honum gildið 1 eða 0 eftir því hvort fjöldi 1-bita í tákninu er oddatala eða slétt tala. Ef einn biti misferst við afritun eða gagnasendingar tryggir *tvístæðubitinn* að villan komi í ljós.

Tákn númer 0 til 31 í *ASCII* stafatöflunni eru ekki eiginleg rittákn heldur *stýristafir* eins og línuskil, síðuskil og hopstafur. Tákn númer 32 til 127 eru í töflunni hér að neðan.

ASCII stafataflan dugar aðeins þeim sem láta sér nægja enska stafrófið. Í henni eru ekki þýskir, danskir og íslenskir stafir eins og ä, ü, å, ø, ð, ý, þ, æ, ö. Til að ráða bót á þessu hafa verið búnar til 8 bita stafatöflur. Stöfunum er þar með fjölgað úr 128 í 265 en möguleikanum á tvístæðubita fórnað í staðinn. Sú 8 bita stafatafla sem mestri útbreiðslu hefur náð í okkar heimshluta heitir *Latin-1*. Fyrri helmingur hennar er eins og ASCII. Seinni helmingur hennar (tákn númer 128 til 255) inniheldur m.a. rittákn úr ýmsum Evrópumálum sem ekki eru í ensku. Þar eru m.a. þýsku, dönsku og íslensku stafirnir sem taldir eru upp hér að ofan.

Tákn númer 32 til 127 í ASCII stafatöflunni

Númer	tákn	Númer	tákn	Númer	tákn
32	⟨orðabil⟩	64	@	96	`
33	!	65	A	97	a
34	”	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	’	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	___	127	DEL

Rittákn sem notuð eru í tungumálum jarðarbúa skipta hundruðum þúsunda. Stafataflan *Latin 1* inniheldur því ekki nema brot af þeim. Fram undir þetta hafa því verið í notkun margar ólíkar 8 bita stafatöflur. Með auknum tölvusamskiptum og útbreiðslu Internetsins hefur þörfin fyrir alþjóðlega stöðlun á stafatöflum farið vaxandi. Til að mæta þessari þörf var búinn til staðall sem kallast *Unicode*. Hann er nú notaður af algengum vöfrum eins og *Netscape Navigator* og *Internet Explorer*.

Unicode staðallinn táknar hvern staf með tveim bætum, þ.e. 16 bitum, og hefur því rúm fyrir $2^{16} = 65.536$ tákn. Stafatöflunni er skipt í 256 síður sem hver inniheldur 256 tákn. Annað bætið táknar því númer síðu og hitt númer tákns á síðu. Fyrsta síðan, sem er númer 0, er eins og *Latin 1* stafataflan. Þar fyrir aftan koma svo síður fyrir ýmis mál. Síða númer 4 er t.d. með rússneskum rittáknum og númer 5 með hebreskum stöfum.[1]

Strengir, myndir og önnur gögn

Forrit vinna með alls konar gögn: orðalista, landakort, veðurspár og spil og allt þar á milli en öll þessi gögn eru skráð í minni tölvunnar sem munstur úr tvenns konar táknum sem við getum litið á sem tölur í tvíundakerfi.

Flest forritunarmál búa yfir aðferðum til að vinna með strengi, þ.e. runur af bókstöfum. Algengt er að skrá strengi sem runur af *ASCII*, *Latin 1* eða *Unicode* táknum og láta tákn númer 0 merkja hvar runan endar.

Margar aðferðir eru til að skrá myndir sem runur af tölum. Þegar mynd er geymd sem *punktafylki* (á ensku *bitmap*) standa tölur fyrir liti. Séu litir til dæmis táknaðir með 3 bætum, einu fyrir rauðan lit, einu fyrir grænan og einu fyrir bláan er hægt að skrá ferningslaga mynd sem er 100 punktar á hvern veg í $3 \cdot 100 \cdot 100$ bæti með því að skrifa lit fyrsta punktsins á myndinni í þrjú fyrstu bætin, lit annars punkts í þau þrjú næstu o.s.fr.

Önnur gögn eins og t.d. kvikmyndir og hljóð eru líka skráð sem runur af tölum í tvíundakerfi. Með heilu tölunum 0, 1, 2, 3, … er hægt að tákna ræðar tölur, stafi, myndir og fleira því allt sem verður tölum talið má para við mengið N eða hlutmengi þess.

Stafræn og flaumræn gögn

Öll gögn sem tölvur fást við eru á *stafrænu* (á ensku *digital*) formi sem þýðir að þau eru kóðuð sem runa af táknum og líkjast yfirleitt ekki þeim veruleika sem þau lýsa. Halarófa af tölum getur sagt hvernig sérhver punktur á mynd er á litinn eða hvernig lag á að hljóma en hún líkist ekki myndinni eða hljóðinu. Gögn sem ekki eru á stafrænu formi, eins og til dæmis ljósmynd á filmu, lag á gamaldags vínilplötu eða prentað landakort kallast ýmist *hliðræn* eða *flaumræn* (á ensku *analog*).

Þegar flaumrænir geymslumiðlar rispast, mást eða skemmast á annan hátt glatast að jafnaði hluti upplýsinganna sem eru geymdar á þeim. Stafræn geymsla er ekki eins viðkvæm því gögn glatast ekki nema stafur skemmist svo mikið að hann verði ólæsilegur.

[1] Upplýsingar um *Unicode* og myndir af öllum 256 síðunum er hægt að nálgast frá vefsíðunni http://www.unicode.org/

Með tölvubyltingunni hafa stafrænir geymslumiðlar að miklu leyti tekið við af flaumrænum og nú eru hljóðupptökur oftast unnar með stafrænni tækni og geisladiskar fyrir hljómflutningstæki geyma hljóð á stafrænu formi.

Myndbandsspólur geyma myndir og hljóð á *flaumrænu* formi en með tilkomu DVD-diska færist í vöxt að kvikmyndum sé dreift á stafrænu formi.

4.2 Breytur og bendar

Gögn í minni tölvu

Forrit geyma gögn í breytum. Eigi forrit á C eða Java t.d. að nota heilu töluna 65 og bókstafinn A þá eru skilgreindar breytur af viðkomandi tegundum og þeim gefin gildi með skipunum á borð við

```
int i;    Breyta af tegundinni int (þ.e. heil tala) skilgreind og gefið nafnið i.
i = 65;   Breytunni i gefið gildið 65.
char c;   Breyta af tegundinni char (þ.e. bókstafur) skilgreind og gefið nafnið c.
c = 'A';  Breytunni c gefið gildið A.
```

Forrit sem eru samin á mótuðum málum (eða hlutbundnum) skiptast í sjálfstæða verkþætti sem oftast eru kallaðir *undirforrit* en líka stundum *föll* eða *aðferðir*. Hvert undirforrit getur haft sínar eigin breytur sem aðeins eru til meðan verið er að framkvæma það og eru ekki aðgengilegar úr öðrum undirforritum. Slíkar breytur kallast *staðværar* (á ensku *local*). Breytur sem eru til allan tímann sem forrit er í gangi og eru aðgengilegar úr öllum aðferðum þess kallast *víðværar* (á ensku *global*).

Sum gögn eru þess eðlis að engin leið er að vita fyrirfram hvað þau munu taka mikið rúm í minni tölu. Sem dæmi má taka texta sem er sleginn inn í ritvinnsluforrit. Þegar forritið er búið til er engin leið að áætla hvað stafaromsurnar sem lamdar eru inn verða langar og margar. Það hentar því illa að geyma þær í víðværum breytum af fastri stærð. Forritið þarf því að búa til nýjar breytur og ákvarða hlutum stað í minni meðan það er í gangi. Breytur sem eru ekki „naglfastar“ við forritið heldur búnar til jafnóðum og á þarf að halda kallast ýmist *hverfular* eða *kviklegar* (á ensku *dynamic*).

Yfirleitt er minnisplássi sem forrit fær til umráða skipt í fjögur svæði þar sem eitt er fyrir *forritskóta*, annað fyrir *varanleg gögn*, hið þriðja fyrir *stafla* og fjórða fyrir *haug*.

- *Forritskótinn* er forritið sjálft, þ.e. runa af skipunum eða fyrirmælum, oftast á vélamáli. Hin þrjú svæðin eru fyrir gögnin sem forritið notar.
- Svæðið fyrir *varanleg gögn* geymir víðværar breytur og fasta sem eru til allan tímann sem forritið er í gangi.
- *Staflinn* geymir staðværar breytur. Staðværar breytur hvers undirforrits fá rúm á staflanum um leið og keyrsla þess hefst.
- *Haugurinn* er rúm fyrir hverful gögn.

Stafli og staðværar breytur

Stór forrit eru iðulega samsett úr tugum, hundruðum eða jafnvel þúsundum undirforrita. Hvert undirforrit hefur sínar eigin breytur sem geta tekið mikið eða lítið rúm eftir atvikum. Ætti að taka frá rúm í minni fyrir allar breytur í öllum undirforritum um leið

og forrit er ræst þá gæti það þurft ansi mikið minni. Minnið nýtist því betur ef hvert undirforrit fær rými fyrir sínar breytur um leið og keyrsla þess hefst og skilar því aftur að keyrslu lokinni.

Flest forritunarmál leyfa *endurkomu* (á ensku *recursion*) sem þýðir að undirforrit A getur kallað á sjálft sig annað hvort beint eða þá óbeint, þannig að A ræsi B sem ræsir svo A. Þegar þetta gerist þarf að geyma tvö eintök af breytum A í senn því önnur keyrsla þess er í gangi þó þeirri fyrstu sé ekki lokið. Þetta væri illgerlegt ef breytur hvers undirforrits hefðu fyrirfram ákveðið rúm í minni tölvunnar því engin leið er að áætla fyrirfram hversu mörg eintök sama undirforrits verða í gangi samtímis og þar með hvort þörf er á einföldu, tvöföldu eða þreföldu eða enn meira rými undir breytur þess.

Til að nýta minnið sem best og gera forritum mögulegt að nota endurkomu er notaður *stafli* (á ensku: *stack*) til að geyma breytur undirforrita.

Matardiskar í eldhússkáp eru oftast geymdir í stafla, hver ofan á öðrum. Þegar diskur er sóttur í skápinn er sá efsti tekinn, þ.e. sá sem síðast var settur í skápinn. Af þessu draga staflar í minni tölvu nafn sitt, það sem næst er tekið af staflanum er það sem síðast var sett á hann. Þeir síðustu inn eru fyrstir út og þeir fyrstu síðastir.

Hér á eftir fer forrit á C sem hefur tvö undirforrit, u1 og u2, auk main sem fer af stað þegar keyrsla forritsins hefst. Þetta forrit inniheldur nokkrar breytur og þó þær séu ekki notaðar neitt fá þær samt rúm í minni. Raunar gerir forritið ekkert annað en að skrifa:

```
Byrja keyrslu main
Byrja keyrslu u2
Byrja keyrslu u1
Enda keyrslu u1
Enda keyrslu u2
Enda keyrslu main
```

Textinn vinstra megin er skýringar en ekki hluti af forritinu.

```
Víðværar breytur skilgreindar.          int j, k;

Undirforritið u1 byrjar hér.            u1()
                                        {
Staðværar breytur í u1 skilgreindar.        int a, b;
Skipunin printf skrifar strenginn sem       printf("Byrja keyrslu u1\n");
er innan gæsalappa.  \n merkir línuskil.    printf("Enda keyrslu u1\n");
Undirforritið u1 endar hér.             }

Undirforritið u2 byrjar hér.            u2()
                                        {
Staðværar breytur í u2 skilgreindar.        int c, d;
                                            printf("Byrja keyrslu u2\n");
Kallað á u1.                                u1();
                                            printf("Enda keyrslu u2\n");
Undirforritið u1 endar hér.             }

Hér hefst main. (Keyrsla C forrits      main()
byrjar efst í main.)                    {
                                            printf("Byrja keyrslu main\n");
Kallað á u2.                                u2();
                                            printf("Enda keyrslu main\n");
Hér endar main.                         }
```

Taflan sem hér fer á eftir sýnir hvaða breytur eru í svæði fyrir varanleg gögn og hvaða breytur eru á stafla á hverjum stað í forritinu. Víðværu breyturnar j og k eru til allan tímann. Staðværu breyturnar a, b, c, d eru aðeins til á meðan undirforritin sem þær tilheyra eru í gangi. Þar sem u1 er kallað úr u2 er keyrslu u2 ekki lokið þegar keyrsla u1 hefst. Þess vegna verður að geyma breytur u2 meðan u1 er í gangi svo u2 geti haldið áfram að nota þær eftir að keyrslu u1 er lokið og u2 er að framkvæma skipanirnar fyrir neðan línuna sem kallar á u1.

Þegar keyrsla u1 hefst eru tvær breytur á staflanum en u1 getur ekki notað þær því það getur aðeins notað það rúm á staflanum sem er laust þegar keyrsla þess hefst.

Forritið skrifar út	Varanleg gögn	Stafli	Skýringar
Byrja keyrslu main	j, k		Keyrsla main að hefjast
Byrja keyrslu u2	j, k	c, d	Keyrir u2
Byrja keyrslu u1	j, k	c, d, a, b	Keyrir u1, u2 ekki lokið
Enda keyrslu u1	j, k	c, d, a, b	Keyrir u1, u2 ekki lokið
Enda keyrslu u2	j, k	c, d	Keyrir u2. Keyrslu u1 lokið
Enda keyrslu main	j, k		Keyrslu u2 lokið

Bendar og hverful gögn

Sumar breytur geyma gögn eins og tölur, stafi og strengi. Breytan er þá nafn á svæði í minninu þar sem gögnin er að finna. Flest forrit hafa líka breytur sem innihalda vistfang. Slíkar breytur kallast *bendar* (á ensku *pointers*). Bendar innihalda ekki gögn heldur vistfang sem segir hvar gögnin er að finna.

Bendar eru notaðir til að vísa á hverful gögn. Í forritunarmálinu *C* eru þeir auðkenndir með *. Til að skilgreina breytu sem heitir i og inniheldur heila tölu er notuð skipunin:

```
int i;
```

En til að skilgreina bendi sem heitir *i* og bendir á stað á haugnum þar sem heil tala er geymd er skipað:

```
int *i;
```

Til að taka frá pláss fyrir 5000 tölur frá og með þeim stað sem *i* bendir á er svo hægt að skipa:

```
i = (int*) malloc(5000);
```

Skipunin *malloc* tekur frá pláss á haugnum. Í stað tölunnar 5000 gæti komið talnabreyta sem inniheldur gildi sem er breytilegt frá einni keyrslu forrits til annarrar. Bendirinn *i* vísar þá á upphaf talnarunu sem ekki hefur fyrirfram ákveðna lengd.

Nú er hægt að vista töluna 65 í fimmta sæti frá þeim stað sem *i* vísar til með

```
*(i + 5) = 65;
```

Flest forritunarmál bjóða upp á notkun benda. Í *C* eru þeir auðkenndir með * og í *Pascal* með ^. Í sumum málum, eins og t.d. *Java* eru bendar ekki auðkenndir sérstaklega heldur er það látið ráðast af tegund hvort breyta er bendir eða ekki.

Í *Java* eru 8 einfaldar tegundir. Þetta eru 4 tegundir af heilum tölum (*byte*, *short*, *int* og *long*) 2 af kommutölum (*float* og *double*), 1 af bókstöfum (*char*) og tegundin *boolean*. Allar breytur sem ekki tilheyra neinni þessara tegunda eru bendar.

Ruslatínsla

Hér að ofan var sýnt hvernig hægt er að búa til hverfula geymslu fyrir 5000 heiltölur í *C*. Í *Java* er hægt að skilgreina bendi sem vísar á upphaf talnarunu með

```
int i[];
```

Upphaflega inniheldur i gildið *null* og vísar ekki á neitt. En með skipuninni

```
i = new int[5000];
```

er tekið frá rými sem dugar til að geyma 5000 tölur og i látin benda á upphaf þess. Síðar í forritinu er skipunin

```
i = new int[3000];
```

ef til vill gefin og i þar með látin vísa á aðra talnarunu. Gamla runan er enn í minni en ef engin bendir vísar á hana lengur getur forritið ekki gert neitt við hana. Hún er eins og hvert annað rusl sem tekur upp pláss án þess að vera til nokkurs nýt.

Þegar forrit eru skrifuð í *Pascal*, *C* eða *C++* þurfa forritarar að gefa skipanir til að taka frá minni á haugnum fyrir hverful gögn og til að endurheimta það aftur þegar hætt er að nota þau. Sum forritunarmál eins og t.d. *Java* bjóða upp á *sjálfvirka ruslatínslu* sem þýðir að forritarar þurfa ekki að hafa áhyggjur af minnisplássi, rúm fyrir hverful gögn er tekið frá sjálfkrafa og skilað aftur þegar hætt er að nota þau. Þegar haugurinn er að fyllast setja *Java* forrit, eða túlkurinn sem keyrir þau, í gang *sjálfvirka ruslatínslu* sem sér um að henda gögnum sem hætt er að nota.

Til upprifjunar

Hvað eru:
ASCII, bendir, haugur, IEEE 754, kvikleg breyta, Latin 1, punktafylki, sjálfvirk ruslatínsla, staðvær breyta, stafli, strengur, tvístæðubiti, tvíundafylli, Unicode, víðvær breyta.

5. kafli

Hugbúnaður

5.1 Nokkrar gerðir hugbúnaðar

Vélbúnaður + hugbúnaður = verkfæri

Flestar vélar eru gerðar til að vinna eftir fáeinum einföldum reglum. Þvottavélar eru til dæmis gerðar til að þvo á nokkra mismunandi vegu, ljósritunarvélar til að afrita af einu blaði á annað á nokkra vegu og reiknivélar geta reiknað eftir fáeinum reiknireglum. Tölvur hafa sérstöðu í heimi véla því þær eru ekki gerðar til að vinna eftir neinni einni reglu öðrum fremur. Það er hægt að mata þær á hvaða reglu sem er og fá þær til að vinna eftir henni.

Sérhæfðar vélar eins og þvottavélar, ljósritunarvélar og reiknivélar eru albúnar að vinna sín fáu einföldu verk um leið og lokið er við að smíða þær. En þótt búið sé að smíða tölvu vantar enn mikið á að hún sé nýtileg til nokkurrar vinnu.

Til þess að gera tölvu að nýtilegu verkfæri þarf að mata hana *hugbúnaði*, þ.e. forritum og gögnum. Tölvan sjálf og jaðartækin sem tengjast henni kallast einu nafni *vélbúnaður*. Með viðeigandi hugbúnaði er hægt að breyta vélbúnaðinum í verkfæri sem vinnur eftir einhverjum tilteknum reglum.

Í venjulegu tölvukerfi sem notað er við skrifstofustörf er vélbúnaðurinn oftast tölva, skjár, lyklaborð, mús, disklingadrif, harður diskur, geisladrif, prentari og ef til vill fleiri jaðartæki. Með því að mata tölvuna á stýrikerfi og ritvinnsluforriti má breyta vélbúnaðinum í verkfæri sem hægt er að nota til að vélrita og vinna með texta.

Vélbúnaðurinn er smíðaður úr plasti, kísli, málmum, gleri og öðrum efnum. Hugbúnaður er hins vegar ekki smíðaður úr neinum áþreifanlegum efnum. Hann er eins og skáldskapur, búinn til úr engu þótt einhvern efnivið þurfi til að geyma hann, rétt eins og það þarf skinn, pappír, geisladiska eða aðra efniskennda miðla til að geyma skáldskap.

Eins og fram hefur komið má skipta hugbúnaði í gögn og forrit. Varast ber þó að taka þessa tvískiptingu of hátíðlega því greinarmunur gagna og forrita er alls ekki mjög skýr. Eitt forrit getur notað annað forrit fyrir gögn og stundum má nota gögn á borð við talnaromsur sem hluta af forskrift eða reglu sem tölva er látin vinna eftir. Til að gera langa sögu stutta skulum við segja að forrit segi tölvu hvaða reglu hún á að vinna eftir og gögnin séu það sem reglunni er beitt á.

Ef við látum tölvu til dæmis raða orðum í stafrófsröð þá eru orðin sem hún raðar gögn en forskriftin, sem segir henni hvernig á að fara að þessu, forrit. Ef við notum tölvu til að skrifa sendibréf þá er bréfið gögn en ritvinnsluforritið sem við notum forrit. En ef bréfið er nú ekki venjulegt sendibréf heldur vélrituð runa af skipunum sem tölvan getur unnið eftir þá má seinna nota það fyrir forrit þótt nú sé það í hlutverki gagna.

Endalaus fjölbreytni

Fyrstu tölvurnar sem smíðaðar voru á árunum í kringum 1950 voru einkum notaðar við talnareikning. Þótt Alan Turing hefði fært rök fyrir því að tölvur gætu unnið hvers kyns hugarstarf og stjórnað vélum og tækjum eftir öllum mögulegum reglum og aðferðum gerðu fáir sér grein fyrir endalausum möguleikum tölvutækninnar.

Þegar þróun tölvutækninnar er lýst er oft lögð áhersla á framfarir í smíði vélbúnaðar. Þessar framfarir eru einkum í því fólgnar að vélarhlutarnir sem tölvukerfi eru byggð úr verða sífellt hraðvirkari, smágerðari og ódýrari. En vélbúnaður nútímatölvu býr samt í aðalatriðum yfir sams konar möguleikum og vélbúnaður í eldri tölvum. Til að skilja hvernig notkunarsvið tölva verða fjölbreytilegri með hverju ári sem líður dugar ekki að einblína á þróun vélbúnaðar. Framfarir í gerð hugbúnaðar hafa ekki síður verið stórstígar. Frá því Turing lýsti því yfir að hægt sé að forrita tölvur til að vinna hvers kyns hugarstarf hefur ekkert lát verið á nýjum uppfinningum á sviði forritunar og hugbúnaðargerðar.

Flokkar og gerðir hugbúnaðar eru nú fleiri en svo að nokkur leið sé að segja frá þeim öllum í stuttu máli. Hér skal því látið duga að nefna nokkrar algengar tegundir hugbúnaðar fyrir einmenningstölvur.

- *Stærðfræði og útreikningar*. *Töflureiknar* (eins og t.d. *Excel*) eru mest notaðir við talnareikning og úrvinnslu tölulegra upplýsinga. Einnig er til úrval af tölfræðiforritum til að vinna tölfræðilega útreikninga og hjálpa fólki að draga ályktanir af tölfræðilegum upplýsingum. (Sem dæmi um tölfræðiforrit má t.d. nefna *SPSS* og *Statistica*).

  Symbólskar reiknivélar (eins og t.d. *Maple*, *MathCad*, *Mathematica* og *Dervie*) eru forrit sem ráða við algebru og stærðfræðigreiningu og geta t.d. einfaldað segðir, fundið heildi og afleiður og einangrað stærð í jöfnum.

- *Ritun og textavinnsla*. Flestir kannast við *ritvinnsluforrit* sem nota má til að vélrita texta og stjórna uppsetningu hans. Til viðbótar við venjuleg ritvinnsluforrit (eins og *Word* eða *WordPerfect*) eru til *umbrotsforrit* (eins og t.d. *Quark* og *PageMaker*) sem notuð eru við uppsetningu texta og bókagerð. Eftir því sem ritvinnsluforrit verða fullkomnari bjóða þau upp á meira og meira af þeim möguleikum sem umbrotsforrit hafa til að stjórna útliti og uppsetningu texta. Sum nýleg ritvinnsluforrit geta líka vistað texta og myndir sem vefsíður og þannig komið að nokkru í stað sérhæfðra vefsmíðaforrita.

  Auk ritvinnslu- og umbrotsforrita er til ýmis hugbúnaður til að aðstoða fólk við ritun og frágang texta. Hér má t.d. nefna forrit til að leiðrétta stafsetningu (eins og t.d. *Púkann* frá Friðriki Skúlasyni sem aðstoðar við réttritun á íslensku) og forrit til að skipta orðum rétt milli lína eða laga hnökra á stíl og greinamerkjasetningu.

- *Gagnagrunnar* (eins og t.d. *Access* eða *Paradox*) eru notaðir til að skrá *gagnasöfn* á borð við símaskrá, þjóðskrá eða nemendaskrá skóla og vinna úr þeim. Slík forrit gegna mikilvægu hlutverki í fjármálastofnunum og viðskiptalífi þar sem þau eru notuð til að halda utan um upplýsingar og búa til reikninga og gluggapóst af ýmsu tagi.

  Í mörgum greinum atvinnulífs eru notaðir sérhæfðir gagnagrunnar. Sem dæmi má nefna íslenska framhaldsskóla sem nota slíkan hugbúnað til að skrá nemendur í áfanga, búa til stundatöflur, prenta út námsferla og einkunnablöð eða henda reiður á skrópi og fjarvistum.

  Gagnagrunnar henta best þegar sams konar upplýsingar (t.d. nafn, heimili og sími) eru skráðar um mjög marga einstaklinga. Upplýsingar um einn einstakling kallast þá *færsla* og hvert atriði sem skráð er (t.d. nafn eða heimili) kallast *svið*. Um gagnagrunna

og gagnasöfn mætti rita langt mál enda er *gagnasafnsfræði* viðamikil grein innan tölvufræðinnar.

- *Bókhaldsforrit* eru eins og nafnið bendir til notuð við bókhald og til að hjálpa stjórnendum fyrirtækja að gera sér grein fyrir hvernig reksturinn gengur.
- *Grafík og myndvinnsla*. Rétt eins og gagnasafnsfræðin er tölvugrafík heil fræðigrein. *Myndvinnsluforrit* (eins og t.d. *Photoshop*) eru m.a. notuð til að vinna listrænar myndir og lappa upp á ljósmyndir eða ganga frá þeim til prentunar eða birtingar á vefsíðum. *Teiknforrit* (eins og t.d. *CorelDraw*) eru notuð við grafíska hönnun og teikningu. Af nokkuð öðru tagi eru svo *hönnunarforrit* (eins og *AutoCad*) sem notuð eru við tækniteikningu og hönnum iðnvarnings og mannvirkja af ýmsu tagi.
- *Vefur, tölvupóstur og internetið*. Flestir tölvunotendur kannast við *vafra* (eins og *Netscape*, *Mosaic* eða *Internet Explorer*) og hafa notað slík forrit. Auk vafra sem eru til að skoða vefsíður er til mikið úrval af hugbúnaði til að búa til vefi og halda utan um vinnu við viðhald þeirra. Aðrir flokkar forrita sem nýta möguleika Internetsins eru t.d. *póstforrit* (til að sækja og senda tölvupóst), *FTP-forrit* (til að flytja gögn milli tölva) og *IRC-forrit* (til að taka þátt í samræðum á spjallrásum).

Hér hafa verið nefndir nokkrir algengir flokkar *notendaforrita* (þ.e. forrita sem fólk notar við leiki, nám og störf). Þessi listi er langt frá því að vera tæmandi. Það mætti bæta við hann *tölvuleikjum*, *tónsmíðaforritum*, *hermilíkönum* og ótal gerðum sérhæfðra forrita til að reikna veðurspár, stjórna flugumferð, aðstoða jarðfræðinga við olíuleit o.s.fr. o.s.fr.

Oft er nokkrum notendaforritum pakkað saman í einn vöndul. Sem dæmi má taka skrifstofuvöndullinn *Office* frá *Microsoft* sem inniheldur *Word* ritvinnslu, *Excel* töflureikni, *PowerPoint* glærugerðarforrit, *Access* gagnagrunninn og nokkur önnur forrit.

Auk notendaforrita er til hugbúnaður sem venjulegir tölvunotendur nota ekki beinlínis en þarf samt að vera til svo hægt sé að keyra notendaforrit. Þessi hugbúnaður kallast einu nafni *kerfishugbúnaður*. Mikilvægustu flokkar kerfishugbúnaðar eru annars vegar *þýðendur* og *túlkar* og hins vegar *stýrikerfi*, *grunnforrit* og *reklar*. (Um stýrikerfi verður fjallað í seinni hluta þessa kafla.)

Þýðendur og túlkar

Keyrsluhæf forrit eru á vélamáli en fólk sem semur forrit skrifar þau yfirleitt á svokölluðum *æðri forritunarmálum*. Til að tölva geti keyrt forrit sem samið er á öðru máli en vélamálinu sem gjörvi hennar vinnur eftir þarf að þýða forritið. Hugbúnaður sem þýðir forrit af einu forritunarmáli á annað er einkum af tvennu tagi: *þýðendur* og *túlkar*.

Þýðandi er forrit sem þýðir af einu forritunarmáli á annað, oftast á vélamál. Þegar hann hefur lokið því verki er forritið til á tveim málum og hægt að keyra vélmálsþýðinguna. *Túlkur* skilar hins vegar ekki heilli þýðingu heldur snarar einni skipun, sér um að hún sé framkvæmd, snarar þá þeirri næstu og þannig koll af kolli uns keyrslu forrits er lokið. Um þýðendur og túlka verður fjallað nánar í kafla 8.2.

Fjölmörg notendaforrit innihalda túlk. Til dæmis er túlkur fyrir forritunarmálin *Java* og *JavaScript* innbyggður í vafra eins og *Netscape* og *Internet Explorer* og túlkur fyrir *Visual BASIC for Applications* er innbyggður í forritin í *Office* vöndlinum frá *Microsoft*. Það er því hægt að hengja forrit á *Visual BASIC for Applications* við *Excel*-töflur, *Word*-skjöl eða *PowerPoint*-glærusýningu. Slíkt forrit sem er hluti af gagnaskrá kallast *fjölvi* (á ensku *macro*). Innbyggðir túlkar og möguleikar á að skrifa fjölva gera notend-

um algengra forrita kleift að bæta við þau nýjum skipunum og láta sjálfvirkni koma í stað seinlegrar vinnu án þess að skrifa heilt sjálfstætt forrit frá grunni.

Veirur og sóttvarnir

Þótt flest forrit séu annað hvort notendaforrit eða kerfishugbúnaður er þessi tvískipting ekki einhlít. Það er til dæmis álitamál hvort túlkur sem er innbyggður í ritvinnsluforrit eða vafra er kerfishugbúnaður eða bara partur af notendaforriti. Einnig eru til flokkar af hugbúnaði eins og *veirur* sem tilheyra hvorugum flokknum.

Veira (á ensku *virus*) er forrit sem tekur afrit af sjálfu sér og dreifir þeim annað hvort um tölvunet eða með því að lauma þeim á disklinga og aðrar gagnageymslur sem flakka milli tölva. Auk þess sem vírusar „æxlast“ og dreifa sér gera flestir þeirra einhverjar skammir af sér. Þeir hættulegustu eyða gögnum af diskum og spilla uppsetningu á hugbúnaði. Veirur eru búnar til af skemmdarvörgum og tilgangur þeirra er oftast sá einn að gera eigendum og notendum tölvukerfa lífið leitt.

Tölvuveirur eru til af nokkrum mismunandi gerðum. Hér verður aðeins getið um fáar þeirra:

Ræsigeiraveirur eru smáforrit sem komið er fyrir á ræsigeira disklings eða annarrar gagnageymslu. Í hvert sinn sem nýr disklingur er settur í disklingadrif les stýrikerfi tölvunnar ræsigeira hans og keyrir forritsbút sem þar er að finna. Sé ræsigeirinn sýktur af veiru afritar hún sig yfirleitt á ræsigeira harðra diska í tölvunni og fer eftir það í gang í hvert sinn sem vélin er ræst og skrifar afrit af sjálfri sér á alla disklinga sem notaðir eru. Sé mikið gert af því að bera gögn milli véla á disklingum geta ræsigeiraveirur breiðst hratt út.

Fjölvaveirur eru fjölvar sem fylgja skjölum og smita yfirleitt önnur skjöl sem eru opin í sama forriti. Mörg forrit opna sama skjal alltaf þegar þau eru ræst (t.d. opnar ritvinnsluforritið *Word* alltaf skjal sem heitir normal.dot og geymir upplýsingar um stillingar og stíla). Ef fjölvaveira afritar sig í þetta skjal fer hún eftirleiðis í öll skjöl sem opnuð eru eða búin til í forritinu. Sé þessum skjölum síðan dreift (t.d. sem viðhengjum með tölvupósti eða með því að bera þau milli véla á disklingum) þá breiðist veiran út.

Póstormar eru forrit sem yfirleitt eru send sem viðhengi með tölvupósti. Séu þau keyrð (t.d. í þeirri trú að þau innihaldi skemmtiefni af einhverju tagi) fá þau póstforritið til að senda sig út um hvippinn og hvappinn (t.d. á póstföng í bréfum sem liggja á diskum í tölvunni).

Algengast er að veirusmit og önnur óværa berist með tölvupósti. Hér er ekki bara um að ræða fjölvaveirur og orma heldur líka alls konar forrit sem send eru í viðhengjum og spilla uppsetningu á hugbúnaði ef þau eru keyrð.

Til að forðast veirusmit er ráðlegt að hafa veiruvarnarforrit stöðugt í gangi og uppfæra það reglulega því gamalt veiruvarnarforrit ræður yfirleitt ekki við nýjustu veirur. Einnig er ráðlegt að láta vera að keyra viðhengi nema vitað sé með vissu hvað í þeim felst. Flest forrit sem keyra fjölva er hægt að stilla svo að fjölvar verði óvirkir og ráðlegt er að gera það áður en grunsamleg skjöl eru opnuð.

5.2 Stýrikerfi

Stýrikerfi og grunnforrit

Þegar venjuleg einmenningstölva er ræst keyrir hún fyrst forrit sem er í lesminni hennar. Þetta forrit byrjar á að láta vélina prófa helstu hluta sína og athuga hvort vélbúnaðurinn sé í lagi. Það endar á að hlaða inn stýrikerfi af diski. Þegar stýrikerfið er komið inn tekur það við stjórn vélarinnar.

Forritin í lesminninu kallast einu nafni *grunnforrit* (á ensku *BIOS*, *basic input-output system*). Þau mynda ásamt *stýrikerfinu* (á ensku *operating system*) og *reklum* fyrir jaðartæki (á ensku *drivers*) mikilvægustu hlutana af kerfishugbúnaði vélarinnar, þá hluta sem gera mögulegt að keyra notendaforrit á henni.

Það er misjafnt eftir tölvutegundum hver verkaskipting er milli grunnforrits og stýrikerfis sem sótt er af diski. Í sumum leikjatölvum og iðntölvum er allt stýrikerfið í lesminninu. Oftast er þó meirihluti þess lesinn inn í ritminni. Í lesminninu eru þó að minnsta kosti skipanaromsur sem sjá um að koma vélinni af stað og sækja stýrikerfi af gagnageymslu og láta það taka við stjórn tölvunnar. Þegar kemur að því að hlaða inn stýrikerfi af diski les tölvan *ræsigeira* (á ensku *boot sector*) disksins. Í honum eru fyrirmæli um hvernig stýrikerfinu skuli hlaðið inn.

Þau stýrikerfi fyrir einmenningstölvur sem útbreiddust eru um þessar mundir eru:

- Ýmsar gerðir af *Windows* frá *Microsoft* (*Windows 95*, *Windows 98*, *Windows NT* og *Windows 2000*).
- Ýmsar útgáfur af *UNIX* sem er upprunnið hjá *Bell* símafyrirtækinu í Bandaríkjunum á árunum kringum 1970 og hefur verið að þróast síðan. *UNIX* er ekki sérstaklega fyrir einmenningstölvur og er ekki síður notað sem stýrikerfi fyrir stórar samnotavélar.
- *Mac OS* fyrir *Apple Macintosh* tölvur. Nýjustu gerðir þessa stýrikerfis eru afbrigði af *UNIX*.
- *Linux* sem var skrifað af Finnanum *Linus Thorvalds* á árunum kring um 1990. *Linux* er næstum alveg eins og *UNIX* en hefur þann kost að vera ókeypis og er auk þess einkum miðað við einmenningstölvur fremur en stærri vélar.

Hlutverk stýrikerfis

Til að átta okkur á hlutverki stýrikerfis skulum við hugsa okkur að við eigum forritanlegt vélmenni sem hægt er að mata á skipunum sem segja því að hreyfa einstök liðamót svo og svo mikið eða lesa merki frá nemum (eða skynfærum). Einstakar skipanir gætu verið eitthvað á þessa leið.

```
Beygja liðamót númer 75 um 8 gráður.
Ef merki berst frá nema númer 544 þá á að beygja liðamót númer 31 um 12 gráður.
Beygja liðamót númer 13 um 1 gráðu aftur og aftur þar til merki berst frá nema númer 71.
```

Ef við ætlum að láta vélmennið fara í sendiferð þurfum við að skrifa mörg þúsund skipanir til að láta það reima á sig skó, opna útidyrnar og ganga yfir þröskuldinn. Til að forrita vélmennið þarf að vita nákvæmlega hvernig það er sett saman og jafnvel þótt við búum yfir þeirri þekkingu er forritunin seinleg og erfið.

Hugsum okkur nú að einhver skrifi safn forrita sem segja vélmenninu hvernig á að vinna algeng verk eins og að taka eitt skref, opna dyr, fara í skó, ganga yfir götu. Ef þessu safni er hlaðið inn í minni áður en við byrjum að forrita sendiferðina verður

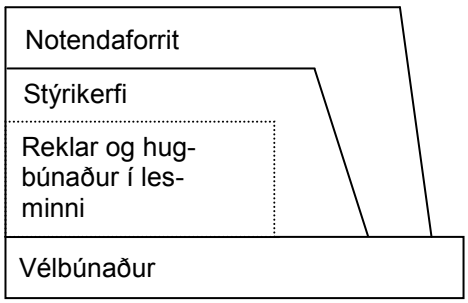
verkið miklu auðveldara. Í staðinn fyrir ótal skipanir sem lýsa hreyfingum einstakra liðamóta kemur nú ein skipun um að finna og framkvæma forritsbút sem segir hvernig á að reima skó.

Stýrikerfi í tölvu hefur svipað hlutverk og forritasafn vélmennisins. Það inniheldur forritsbúta sem segja hvernig á að vinna algeng verk eins og að skrifa staf á skjá, vista skrá á diski eða lesa merki frá lyklaborði og mús. Það gerir forriturum mögulegt að skrifa hugbúnað án þess að vita smáatriði um jaðartæki, tengibúnað og innviði vélarinnar.

Forritari sem skrifar ritvinnsluforrit þarf ekki að vita nákvæmlega hvernig gagnageymslur virka til að geta látið forritið vista gögn og opna skrár. Hann þarf heldur ekki að vita mikið um skjá og skjátengi til að láta forritið birta stafi og myndir á skjánum. Ritvinnsluforrit inniheldur ekki fyrirmæli um hvernig þetta er gert heldur aðeins skipanir um að framkvæma þá hluta stýrikerfisins sem sjá um að gera þetta.

Stýrikerfið býr yfirleitt ekki yfir upplýsingum um hvernig stjórna skal öllum jaðartækjum sem tengd eru tölvunni. Það lætur því rekla annast stjórn þeirra. En *rekill* fyrir jaðartæki, t.d. prentara, er forrit sem yfirleitt fylgir tækinu og inniheldur skipanir til að stjórna því. Það er því ekki nóg með að forritarar, sem skrifa notendaforrit, geti treyst á stýrikerfið í stað þess að skrifa nákvæm fyrirmæli um hvernig vinna skal algeng verk á borð við að vista skrá eða birta letur á skjánum. Þeir sem búa til stýrikerfin þurfa ekki að vita smáatriði um allar mögulegar gerðir jaðartækja. Þeir treysta á að reklar og hugbúnaður í lesminni tækjanna sjái að mestu um stjórn þeirra stillingar.

Í vissum skilningi er kerfishugbúnaðurinn á milli vélbúnaðarins og notendaforritanna. Þegar maður biður ritvinnsluforrit að vista skrá eða prenta þá tekur forritið við skipuninni og vinnur úr henni að einhverju leyti en biður stýrikerfið að vinna suma verkþættina. Stýrikerfið inniheldur skipanir til að sinna hluta þess sem það er beðið fyrir en biður rekla, grunnforrit og hugbúnað í lesminni tengja og jaðartækja að vinna suma verkþætti. Þessi lagskipting er lykill að skipulegri verkaskiptingu milli forritara og þeirra sem smíða vélbúnað. Hún gerir forriturum mögulegt að skrifa notendahugbúnað sem hægt er að keyra á tölvum sem hafa að einhverju leyti ólíkan vélbúnað og ólík jaðartæki og hlífir þeim við að þurfa að vita tæknileg smáatriði um gerð vélbúnaðarins.



Hér hefur verið gerð lausleg grein fyrir hlutverki stýrikerfis. Hægt er að skýra með skipulegri hætti hvað það gerir með því að skipta hlutverkum þess í þrjá flokka sem eru:

1. Að *einfalda tölvuna* fyrir notendum og forriturum og hlífa þeim við að þurfa að vita smáatriði um byggingu og gerð vélbúnaðarins. Undir þetta fellur m.a. að:
   - ✓ Skipuleggja gagnageymslur sem skráakerfi;
   - ✓ Annast samskipti við jaðartæki eins og gagnageymslur, skjá, lyklaborð og prentara;
   - ✓ Leggja til samræmd notendaskil (þ.e. sjá um að búa til valmyndir, textareiti, hnappa o.þ.u.l. fyrir notendaforrit).

2. *Úthluta* forritum og notendum *aðgangi að gögnum vélbúnaði*. Undir þetta fellur m.a. að:
   - ✓ Skipta minni tölvunnar milli forrita sem eru í gangi;
   - ✓ Stjórna víxlvinnslu, þ.e. láta gjörvann sinna forritum til skiptis þannig að eftir að búið er að framkvæma nokkrar skipanir í einu forriti sé staða þess vistuð og keyrðar skipanir í því næsta og svo koll af kolli þar til öll forrit sem í gangi eru hafa fengið dálítinn tíma og gjörvinn snýr sér aftur að því fyrsta og fer annan hring;
   - ✓ Úthluta aðgangi að jaðartækjum og skrám þannig að prentun frá einu forriti bíði t.d. meðan annað forrit er að nota prentarann og skrá sem eitt forrit hefur opna sé lokuð fyrir öðrum forritum.
3. *Tryggja öryggi* í rekstri tölvukerfis. Undir þetta fellur m.a. að:
   - ✓ Koma í veg fyrir að villa í einu forriti spilli keyrslu annarra forrita;
   - ✓ Gera kerfisstjóra kleift að stjórna því með aðgangs- og lykilorðum hvað einstakir notendur geta gert (hvaða forrit þeir geta keyrt og hvern aðgang þeir hafa að gögnum).

Stýrikerfið ræður mestu um hvernig tölva kemur fram við notanda sinn. Tvær tölvur með sama stýrikerfi haga sér nær alveg eins þótt þær kunni að vera ólíkrar gerðar. En séu tvær tölvur sömu gerðar látnar hafa sitt stýrikerfið hvor þá getur hegðun þeirra orðið býsna ólík.

Notendaforrit eru háð þjónustu stýrikerfis. Forrit sem skrifuð eru fyrir tiltekna vélargerð og stýrikerfi (t.d. PC-tölvur með Windows stýrikerfi) og þýdd hafa verið á vélamál, er að jafnaði hvorki hægt að keyra á öðru vísi vél né undir öðru stýrikerfi.

Skráakerfi

Hvert stýrikerfi *forsníður* diska og aðrar gagnageymslur á sinn sérstaka hátt og raðar skrám á þær eftir sínu höfði. Þess vegna er upp og ofan hvernig gengur að flytja disklinga, geisladiska og aðrar gagnageymslur milli véla sem nota ólík stýrikerfi.

Þegar gagnageymsla er forsniðin tekur stýrikerfið frá visst svæði á henni undir *efnisyfirlit* (á ensku *FAT, file allocation table*). Þar ritar það upplýsingar um hvar hverja skrá er að finna.

Flest stýrikerfi skipuleggja skráakerfi með svipuðum hætti þannig að hægt er að pakka skrám saman í *skráasöfn* (á ensku *directory*) sem geta verið hvert inn af öðru. Í Windows hefur hver gagnageymsla sitt eigið nafn sem er myndað úr bókstaf og tvípunkti (disklingadrif heitir t.d. oftast A: og fyrsti harði diskur C:). Í UNIX og skyldum stýrikerfum birtast einstakir geymslumiðlar hins vegar sem skráasöfn. Í augum þeirra sem hafa vanist Windows lítur þetta út eins og sérhver tölva sem keyrir UNIX hafi aðeins eina gagnageymslu og disklingadrif eða geisladrif séu skráasöfn innan hennar.

Nokkuð er ólíkt eftir stýrikerfum hvaða reglur gilda um skráanöfn. Í UNIX getur slíkt nafn t.d. ekki innihaldið orðabil. Þegar gögn eru flutt milli tölva sem nota ólík stýrikerfi (t.d. sem viðhengi í tölvupósti) valda ólíkar venjur um skráanöfn stundum vandræðum. Stutt nöfn sem aðeins eru mynduð úr enskum bókstöfum (A, B, C … Z) ganga yfirleitt á öllum tölvum svo þeim sem senda gögn og vita ekki deili á tölvukosti viðtakanda er ráðlagt að halda sig við enska stafrófið og forðast greinarmerki (önnur en punkt milli nafns og nafnauka), bil eða bókstafi sem ekki tilheyra ensku ritmáli (Á, Å, Ä, Đ, É o.s.fr.) í skráanöfnum.

Algengt er að láta skráanöfn enda á punkti og nafnauka, þ.e. skammstöfun sem segir til um hvers konar skrá er um að ræða. Hér fara á eftir nokkrir nafnaukar sem algengt er að nota með Windows stýrikerfum frá Microsoft.

BAT	Skipanaskrá sem skipanatúlkur stýrikerfis getur keyrt.
BMP	Mynd (óþjöppuð á því formi sem Windows birtir á skjánum).
COM	Keyrsluhæft forrit á vélamáli.
DLL	„Library“ þ.e. safn forritsbúta sem exe-skrár geta kallað á.
DOC	Word-ritvinnsluskrá.
EXE	Keyrsluhæft forrit á vélamáli.
INI	Stillingar á forriti.
SYS	Partur af stýrikerfinu.
TXT	Textaskrá.
ZIP	Þjöppuð gögn (getur innihaldið margar skrár og skráasöfn).

Verkefni 5.2.a

Finndu út hvaða stýrikerfi er á tölvunni sem þú notar og hvaða önnur stýrikerfi er hægt að fá á hana.

Verkefni 5.2.b

Hvaða notendahugbúnaður er settur upp á tölvunni sem þú notar?
Getur þú giskað á hvað hann kostar?
Er hægt að fá ódýrari hugbúnað sem gerir sama gagn?

Verkefni 5.2.c

Væntanlega eru notendaskil í nokkrum forritum sem þú kannast við samræmd þannig að valmyndir eru t.d. svipaðar og valblöðin sem koma upp þegar þú vistar skrá eða prentar nokkurn veginn eins í þeim öllum. Notar þú einhver forrit sem eru undantekning frá þessu og nýta ekki þau notendaskil sem stýrikerfið býður upp á?

Til upprifjunar

1. Hvað er:
   Fjölvi, færsla, gagnagrunnur, grunnforrit, hugbúnaður, kerfishugbúnaður, Linux, Mac OS, nafnauki, notendaforrit, póstormur, rekill, ræsigeiri, skráasafn, stýrikerfi, svið, túlkur, umbrotsforrit, UNIX, veira, Windows, þýðandi?

2. Hver eru helstu hlutverk stýrikerfis?

3. Nefndu fjögur stýrikerfi og fimm flokka notendaforrita.

4. Hvaða nafnauka hafa keyrsluhæf forrit í Windows?

6. kafli
Tölvunet

6.1 Hlutverk og helstu flokkar tölvuneta

Staðarnet og víðnet

Tvær eða fleiri tölvur sem eru tengdar saman þannig að þær geti skipst á boðum eða sent gögn sín á milli kallast *tölvunet*. Séu vélarnar allar á sama svæði (í sömu byggingu eða á sömu lóð) er talað um *staðarnet* (á ensku *LAN*, *local area network*). Sé tölvurnar dreifðar þannig að löng leið sé á milli sumra þeirra er talað um *víðnet* (á ensku *WAN*, *wide area network*). Stundum eru net, sem dreifast um stærra svæði en staðarnet en þó þannig að vélarnar séu allar innan sama þéttbýlis eða byggðarlags, kölluð *borgarnet* (á ensku *MAN*, *metropolitan area network*).

Flest fyrirtæki sem nota margar tölvur reka staðarnet. Dæmigert staðarnet samanstendur af nokkrum *vinnustöðvum* (þ.e. tölvum sem fólk vinnur við) og einum eða fleiri *netþjónum* (á ensku *server*, þ.e. tölvum sem veita öðrum vélum einhvers konar þjónustu) og tengingum á milli þessara véla. Oftast er a.m.k. einn netþjónn búinn stórum og hraðvirkum gagnageymslum sem vinnustöðvar hafa aðgang að. Slíkur netþjónn kallast *skráaþjónn* (á ensku *file server*). Önnur algeng hlutverk *netþjóna* eru að veita vinnustöðvum aðgang að prenturum, miðla tölvupósti eða fletta upp í gagnasöfnum. Þjónar sem sinna þessum hlutverkum kallast *prentþjónar* (á ensku *print server*), *póstþjónar* (á ensku *mail server*) og *gagnasafnsþjónar* (á ensku *database server*). Á litlu staðarneti getur ein venjuleg PC-tölva gegnt í senn hlutverki skráa-, prent- og póstþjóns. Á stóru neti getur þurft marga dýra og öfluga netþjóna.

Netþjónar keyra *stýrikerfi* sem sjá um að úthluta samnýttum tækjum, miðla boðum milli véla og stjórna samskiptum á netinu. Flest venjuleg stýrikerfi eins og *UNIX*, *Linux* og ýmsar útgáfur af *Windows* búa yfir möguleikum á að sinna þessum hlutverkum og raunar eru *UNIX*, *Linux*, og *Windows NT* algengustu stýrikerfin á netþjónum á *Internetinu*, auk þess sem þau eru mikið notuð á netþjónum á staðarnetum. Einnig eru til sérhæfð netstýrikerfi eins og *NetWare* frá *Novell* sem er sérstaklega gert fyrir netþjóna á staðarnetum en hentar engan veginn sem stýrikerfi fyrir vinnustöðvar.

Flest stærri staðarnet eru *biðlara-miðlaranet* (á ensku *client-server network*) sem þýðir að sumar tölvur eru eingöngu vinnustöðvar (þ.e. biðlarar) og aðrar eingöngu *netþjónar* (þ.e. miðlarar) og ekki notaðar sem *vinnustöðvar*. Stundum eru lítil staðarnet sett upp sem *jafningjanet* (á ensku *peer-to-peer network*) sem þýðir að allar tölvurnar eru vinnustöðvar og sumar þeirra veita öðrum aðgang að diski eða prentara. Yfirleitt eru allar tölvur á jafningjaneti látnar keyra stýrikerfi (eins og t.d. *Windows* eða *Linux*) sem gerir þeim kleift að leika í senn netþjón og vinnustöð.

Þegar tölvur eru tengdar í staðarnet eru yfirleitt lagðir vírar milli þeirra eða settir upp sendar og loftnet ef netið er þráðlaust. Þegar tölvur eru tengdar í víðnet er oftast reynt að nota símavíra eða aðrar lagnir sem til eru fyrir. Þannig eru tölvur Íslenskrar

getspár (þ.e. lottóvélarnar sem eru í söluturnum um allt land) t.d. tengdar saman gegnum símakerfið enda væri ansi dýrt að leggja sérstakar lagnir fyrir þær.

Það víðnet sem flestir þekkja er *Internetið*. Það nær um allan heim og tengir ekki bara saman einstakar tölvur heldur líka minni net. Í fyrirtækjum sem reka staðarnet er algengt að ein vél á staðarnetinu hafi fast samband við *Internetið*.

Annað víðnet sem margir kannast við er rekið af fjármála- og greiðslukortafyrirtækjum og tengir saman hraðbanka, posa í verslunum og fjármálastofnanir um allan heim þannig að fari Íslendingur með greiðslukortið sitt til Asíu og taki þar út peninga í hraðbanka eða vörur í verslun þá færist úttektin sjálfkrafa á reikning hans hér á landi.

Hlutverk staðarneta

Staðarnet eru til af ýmsu tagi og hlutverk þeirra eru mörg og ólík. Hér verða aðeins nefnd nokkur þau helstu:

- *Samnýting vélbúnaðar*. Þar sem margar tölvur eru tengdar saman á staðarneti geta þær samnýtt tæki eins og t.d. prentara, diska, skanna, teiknivél eða afritunarbúnað. Væru vélarnar ekki tengdar saman þyrftu menn annað hvort að tengja einn prentara við hverja þeirra (sem er nokkuð dýrt) eða láta sér lynda að vinna við tölvur sem ekki geta prentað á blað.
- *Samnýting gagna og hugbúnaðar*. Þar sem margar vélar hafa aðgang að sömu diskum er kostur á hópvinnu og samstarfi af ýmsu tagi, t.d. geta þá margir unnið við að slá gögn inn í sama gagnasafn og einn notandi getur lesið upplýsingar augnabliki eftir að annar hefur slegið þær inn.

  Sameiginlegar gagnageymslur á skráaþjóni geta líka sparað vinnu við uppsetningu og viðhald á hugbúnaði. Það er að jafnaði fljótlegra að setja hugbúnað upp á einni sameiginlegri gagnageymslu en að ferðast með geisladisk eða disklinga milli vinnustöðva og setja hann upp á hverri og einni.
- *Miðlæg keyrsla*. Eins og nefnt hefur verið getur venjulegur skráaþjónn sparað mikla vinnu við uppsetningu og viðhald á hugbúnaði. Stundum er hægt að ná enn meiri vinnusparnaði með því að hafa *hugbúnaðarþjón* (á ensku *application server*) á staðarneti. Slíkur þjónn keyrir forrit fyrir vinnustöðvar og sendir þeim bara skjámyndir og tekur við merkjum sem þær senda frá mús, lyklaborði eða öðrum inntakstækjum. Þar sem forrit eru keyrð miðlægt af hugbúnaðarþjóni þurfa vinnustöðvar aðeins að keyra *biðlara* (á ensku *client*) sem tekur við skjámyndum og sendir merki frá inntakstækjum. Það þarf því aðeins að setja upp eitt forrit á hverri vinnustöð, öll hin eru sett upp og keyrð á hugbúnaðarþjóni. Mestöll vinna við uppsetningu og viðhald hugbúnaðar getur því farið fram á einum stað.
- *Öryggisafritun*. Ef allir tölvunotendur vista gögn sín á diski í einum skráaþjóni er hægt að taka öryggisafrit fyrir þá alla með því að afrita þann eina disk á segulband eða aðra geymslu. Þetta er mun fljótlegra og auðveldara en að taka afrit af hörðum diskum í mörgum tölvum sem dreifðar eru um stórt svæði. Í flestum fyrirtækjum sjá umsjónarmenn tölvuneta um að taka reglulega öryggisafrit af verðmætum gögnum og geyma þau á öruggum stað þannig að þau glatist ekki þó tölvubúnaður bili eða eyðileggist.
- *Aðgangur að Internetinu*. Með því að tengja einn þjón á staðarneti við *Internetið* er hægt að veita öllum vinnustöðvum aðgang að vef, tölvupósti og annarri þjónustu sem *Internetið* býður upp á. Hægt er að láta hugbúnað á þessum netþjóni stýra aðgangi

vinnustöðva að Internetinu t.d. með því að útiloka aðgang að tilteknum vefsíðum (t.d. klámi) eða tilteknum þjónustuflokkum (t.d. spjallrásum).

- *Að stjórna aðgangi að tækjum og hugbúnaði*. Þegar vinnustöðvar tengjast staðarneti biðja þær yfirleitt um notandanafn og lykilorð. Notendanöfn eru yfirleitt öllum kunn en lykilorð engum nema þeim eina sem notar það og ef til vill umsjónarmönnum netsins. Yfirleitt er svo búið um hnúta að engin leið er að tengjast undir tilteknu notendanafni nema vita hvaða lykilorð fylgir.

  Umsjónarmenn tölvuneta úthluta réttindum til notenda, ákveða til dæmis að sá sem tengist undir nafninu nonni megi nota þennan prentara en ekki hinn, lesa skrár í tilteknu skráasafni en ekki eyða þeim eða breyta meðan sá sem tengist undir nafninu sigga má nota alla prentara og skrifa í skrárnar sem nonni fær aðeins að lesa o.s.fr. Með þessu móti er hægt að stjórna aðgangi að sameiginlegum tækjum og gagnageymslum.

  Sjálfir tengjast umsjónarmenn staðarneta undir notendanöfnum sem veita ótakmörkuð réttindi. (Þar sem netþjónar keyra *Novell NetWare* netstýrikerfið hefur notandi sem tengist undir nafninu admin yfirleitt ótakmörkuð réttindi. Á *Linux* og *UNIX* veitir notendanafnið *root* oftast ótakmörkuð réttindi. Til að tryggja öryggi tölvuneta er mikilvægt að lykilorðum þessara ofurnotenda sé haldið vandlega leyndum.)

- *Eftirlit með notendum*. Á sumum staðarnetum geta umsjónarmenn netkerfis eða stjórnendur fyrirtækis fylgst með tölvunotkun starfsmanna, séð hvenær þeir eru tengdir, hvaða forrit þeir nota, hvað vefi þeir heimsækja, hverjum þeir senda tölvupóst o.s.fr.

*

Með hverju ári sem líður verða tölvur ódýrari en kostnaður við rekstur tölvukerfa í fyrirtækjum lækkar ekki sem nemur lækkun á verði vélanna því með aukinni tölvunotkun eykst þörf á þjónustu við notendur. Það þarf að aðstoða þá á ýmsa lund og bjarga þeim úr vandræðum vegna kenjótts vélbúnaðar, gagna sem týnast eða forrita sem haga sér öðru vísi en til er ætlast. Í mörgum fyrirtækjum kostar rekstur einmenningstölvu í eitt ár mun meira en kaupverð vélarinnar. Skynsamleg uppbygging staðarneta og góð stjórn á þeim er helsta leið fyrirtækja til að draga úr útgjöldum af þessu tagi.

Verkefni 6.1.a

Finndu út hvað eru margir netþjónar á staðarnetinu í skólanum (eða vinnustaðnum) þínum, hvaða hlutverk hver og einn hefur og hvaða stýrikerfi þeir nota.

Verkefni 6.1.b

Hér að framan voru talin upp 7 hlutverk staðarneta. Hverjum þessara hlutverka sinnir staðarnetið í skólanum eða á vinnustaðnum þínum?

6.2 Uppbygging tölvuneta

OSI staðallinn

Tölvur eru tengdar saman í net með ýmsu móti. Stundum eru notuð netkort og tvinnaður vír, stundum mótöld og símalagnir, stundum þráðlaust samband. Ólík netstýrikerfi pakka gögnum líka á ólíka vegu og merkja sendingar með misjöfnum hætti. Alls konar

forrit nýta sér samskipti tölva á neti algerlega óháð því hvers eðlis tengingarnar eru og hvaða staðla netstýrikerfin nota. Ritvinnsluforrit getur t.d. prentað á prentara sem samband er við gegnum prentþjón og höfundur ritvinnsluforritsins þarf ekki að hafa neina hugmynd um hvor sambandið við prentþjóninn er um tvinnaðan koparvír, ljósleiðara eða með útvarpsbylgjum af einhverju tagi. Eins þurfa þeir sem skrifa póstforrit og vafra sem sækja gögn út á Internetið ekki að taka neina afstöðu til þess hvort sambandið er um mótald og símalínu, ADSL-tengingu, ethernet-búnað eða eitthvað allt annað.

Í síðasta kafla var fjallað um það nokkrum orðum hvernig lagskipting, þar sem kerfishugbúnaður kemur milli vélbúnaðar og notendahugbúnaðar, gerir forriturum mögulegt að skrifa notendahugbúnað sem hægt er að keyra á tölvum þótt þær hafi að einhverju leyti ólíkan vélbúnað og ólík jaðartæki og hlífir þeim við að þurfa að vita tæknileg smáatriði um gerð vélbúnaðarins. Hliðstæð lagskipting gerir mönnum mögulegt að tengja ólíkar tölvur sem keyra ólík stýrikerfi saman í net og skrifa forrit sem miðla gögnum milli tölva óháð því hvers konar tengibúnaður er notaður og á hvaða formi gögnin eru send.

Flestir framleiðendur tengibúnaðar, netstýrikerfa og hugbúnaðar fyrir tölvunet fylgja staðli frá *Alþjóðlegu staðlasamtökunum* sem oft eru nefnd með skammstöfuninni *ISO* (en hún stendur fyrir *International Standards Organization*). Staðall þessi nefnist *OSI* (*open systems interconnection*) og kveður meðal annars á um hvernig tengingum milli tölva skuli skipt í sjö lög sem talin eru upp í eftirfarandi töflu.

7	Notkunarlag	Allt sem notendur sjá tilheyrir þessu efsta lagi.
6	Framsetningarlag	Þessi fimm millilög sjá um að koma gögnum frá notkunarlagi (þ.e. notendaforritum) á form sem hægt er að senda eftir eðlislaginu (þ.e. tengibúnaðinum) m.a. með því að skipta gögnum í pakka og merkja þá með einkennisstöfum viðtakanda. Þau taka líka við sendingum frá eðlislagi og koma þeim á form sem notkunarlag getur tekið við.
5	Fundarlag	
4	Flutningslag	
3	Netlag	
2	Greinalag	
1	Eðlislag	Allur vélbúnaður (netkort, vírar o.s.fr.) tilheyra þessu neðsta lagi.

Þar sem þessum staðli er fylgt sér hugbúnaðurinn í greinalaginu um öll samskipti við eðlislagið. Þetta þýðir að sé tengingum breytt (hægvirkum netkortum skipt út fyrir önnur hraðvirkari eða skipt úr upphringisambandi í fasta línu) þá dugar að skipta um þann hugbúnað sem tilheyrir greinalaginu.

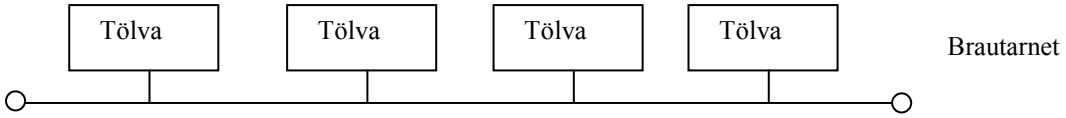
Hér er ekki rúm til að gera grein fyrir hverju lagi en það sem gerist í grófum dráttum þegar notendaforrit (t.d. póstforrit) á einni vél sendir gögn til forrits (t.d. póstþjóns) á annarri vél er á þessa leið:

Notendaforritið afhendir forriti eða forritshluta sem tilheyrir framsetningarlaginu gögnin ásamt upplýsingum um hvert skuli senda þau. Forritið í framsetningarlaginu skiptir gögnunum niður í pakka og í sumum tilvikum þjappar það þeim saman eða dulkóðar þau. Það sendir pakkana til forrita sem tilheyra fundarlaginu. Fundarlagið sendir merki til næsta lags fyrir neðan um að koma á sambandi við viðtakanda. Þegar svar berst um að samband sé komið á sendir það gagnapakkana einn af öðrum til hugbúnaðar í flutningslagi. Í lögunum þar fyrir neðan er upplýsingum til að greina hvort gögn berast villulaust bætt framan við hvern gagnapakka. (Þetta getur t.d. verið talan sem fæst með því að leggja gögnin í pakkanum saman bæti fyrir bæti). Í neðstu millilögunum er upplýsingum um hvert gögnin skuli berast og hvaðan þau koma líka breytt í kennitölur á netkortum eða aðra einkennisstafi sem bætt er framan við hvern gagnapakka svo neðstu lögin á viðtökuvél þekki hvað þeim er ætlað. Gagnapakkarnir eru svo sendir um eðlislagið.

Á leiðinni frá notendaforriti niður í vír er gögnunum skipt í pakka og alls konar merkingum bætt framan og aftan við þá. Hjá viðtakanda er atburðarásin í öfugri röð þannig að það sem t.d. netlag á vél viðtakanda fær er nákvæmlega það sama og netlag á vél sendanda lét frá sér fara.

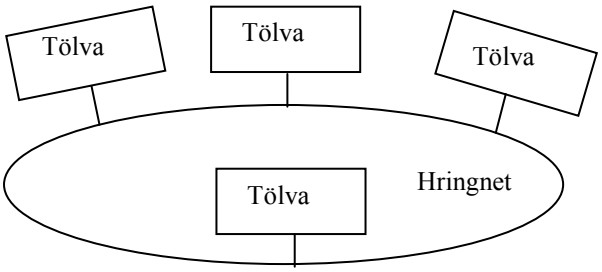
Eðlislag og greinalag á staðarneti

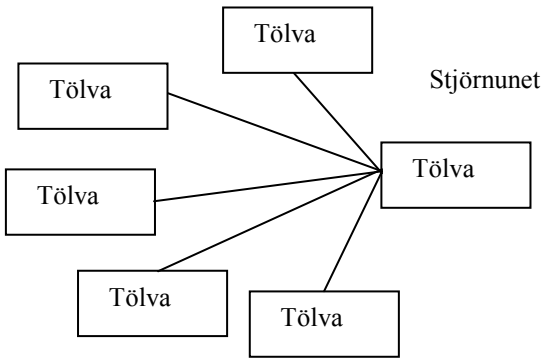
Algengasti tengibúnaður fyrir staðarnet fylgir svokölluðum *Ethernet* stöðlum sem lýsa uppbyggingu tengibúnaðar (sem tilheyrir eðlislaginu) og neðsta lagi hugbúnaðar, þ.e. greinalaginu sem inniheldur rekla fyrir tengibúnaðinn eða annan hugbúnað sem stjórnar samskiptum tölvunnar við hann.



Hver tölva á Ethernet-neti hefur eitt eða fleiri Ethernet tengi. Slík tengi eru í sumum tilvikum áföst móðurborði en oftar eru þau á lausum spjöldum. Sum Ethernet tengi nota þráðlaust samband eða tengjast við ljósleiðara en algengast er að ódýr Ethernet-spjöld hafi innstungur fyrir *samrása streng* (á ensku *coaxial cable*) eða *tvinnaða línu* (á ensku *twisted-pair*). Ethernet er *brautarnet* sem þýðir að tölvurnar tengjast í röð á eina lögn eins og myndin að ofan sýnir. (Á endum lagnarinnar eru endaviðnám sem eru táknuð með litlum hringjum.)

Þótt *Ethernet* sé algengasti tengibúnaður fyrir staðarnet eru til fleiri gerðir eins og t.d. *Tókahringnet* (á ensku *Token ring network*) frá IBM. Á slíku neti tengjast tölvurnar á eina lögn sem liggur í hring eins og myndin hér til vinstri sýnir.



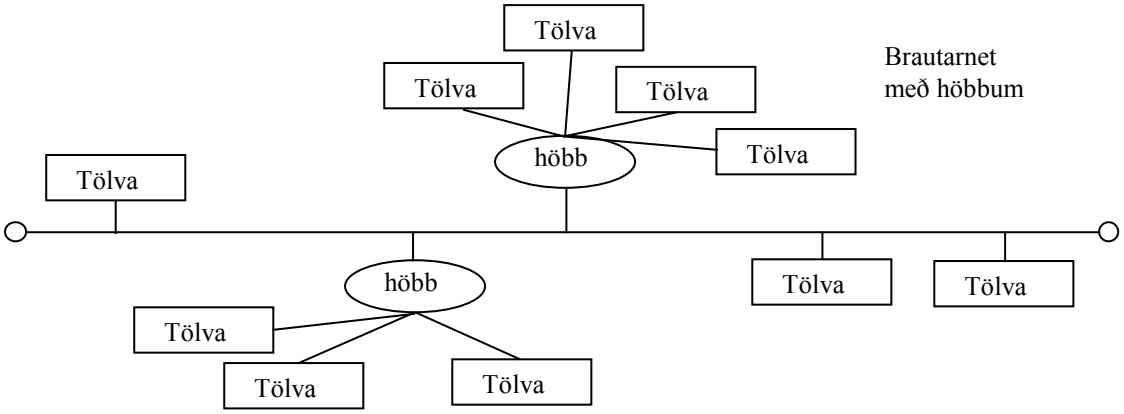


Þar sem brautarnet (eins og t.d. Ethernet) eða hringnet (eins t.d. Tókahringnet) eru lögð þarf fremur lítið af vírum og köplum því allar vélarnar nýta sömu lögnina. Á móti kemur að bilun í einni vél getur truflað öll samskipti á netinu og vélarnar þurfa að skiptast á að nota kapalinn því á hverju augnabliki getur aðeins ein sent gögn eftir honum.

Ef tvær vélar á Ethernet neti reyna samtímis að senda pakka eftir lögninni verður truflun og tengibúnaðurinn í vélunum bregst við með því að bíða stundarkorn og nota slembifall (þ.e. fall sem líkir eftir tilviljun) til að ákvarða biðtímann.

Stjörnunet eru laus við þá galla á brautar- og hringnetum sem hér hafa verið nefndir en krefjast ansi mikilla lagna því lagður er sérstakur kapall frá einni miðstöð í hverja vél. Þó ein vél (önnur en miðstöðin) bili eða einn kapall slitni hefur það ekki áhrif á aðrar vélar á netinu.

Þótt Ethernet sé hugsað sem brautarnet er hægt að njóta að nokkru kosta stjörnunets með því að nota tengibúnað sem kallast *höbb* (á ensku *hub*). Slíkt tæki tengist annars vegar Ethernet braut og hins vegar nokkrum tölvum þannig að sér lögn er úr höbb í hverja tölvu. Þótt tenging úr höbb í eina tölvu bili (vír fari í sundur eða tengibúnaður skemmist) helst samband milli hinna. Næsta mynd lýsir brautarneti með tveim höbbum og nokkrum tölvum sem tengjast beint við Ethernet brautina.



Hér hafa verið nefnd stjörnunet, hringnet og brautarnet og fjallað dálítið um algengustu gerð brautarneta sem er Ethernet. Þessi upptalning er langt frá því að vera tæmandi. Það eru til margar fleiri leiðir til að tengja tölvur saman í staðarnet.

Eðlislag og greinalag á víðneti

Tengibúnaður fyrr víðnet er til af margvíslegu tagi, allt frá mótöldum sem tengjast við símainnstungu upp í ljósleiðara sem tengjast við hraðvirk netkort. Í fyrrnefnda tilvikinu er eðlislagið mótald og símavír og greinalagið rekill fyrir mótaldið eða annar hugbúnaður sem stjórnar samskiptum tölvunnar við það. Í því síðarnefnda er eðlislagið netkort og ljósleiðari.

Algengustu lagnir fyrir víðnet eru símavírar (þeir sömu og notaðir eru fyrir talsíma og fax-tæki). Ýmsar leiðir eru til að tengja tölvu við símainnstungu. Hingað til hefur verið algengast að nota mótald. En *mótald* er tæki sem breytir merkjum frá tölvu (þ.e. tölum á formi tvíundakerfis) í runu af tvenns konar tónum og túlkar tóna, sem berast eftir vírnum, fyrir tölvuna með því að snara þeim yfir í tvíundakerfistölur. Þessi tækni byggist á því að símavírar eru hannaðir til að bera hljóð. Á undanförnum árum hafa fundist leiðir til að senda stafræn gögn um símavír án þess að breyta þeim í tóna eða hljóðmerki. Tvær slíkar aðferðir hafa náð verulegri útbreiðslu. Önnur kallast *ISDN* (eða *samnet*) og hin *ADSL*.

Þar sem sett hafa verið upp ISDN símatengi er hægt að nota tvö tæki í senn á sama símavír (t.d. tölvu og talsíma eða talsíma og fax) því ISDN-samband hefur tvær rásir. Ef aðeins eitt tæki er í notkun leggur það undir sig báðar rásirnar. ADSL samband leyfir líka að talsími sé notaður samtímis tengingu við tölvunet.

Bandbreidd

Hugtakið *bandbreidd* eða *bandvídd* er notað til að lýsa flutningsgetu netlagnar, þ.e. því hvað hægt er að flytja miklar upplýsingar á tímaeiningu. Mælieiningarnar kb/sek (kíló-

bitar á sekúndu) og Mb/sek (megabitar á sekúndu) eru oft notaðar til að lýsa bandbreidd. 1 Mb/sek jafngildir 128 kílóbætum á sekúndu (því eitt bæti er 8 bitar).

Ódýr Ethernettengi geta flutt 10 Mb/sek og betri Ethernettengi ráða við allt að 100 Mb/sek. Þetta segir þó ekki alla sögu um flutningshraða á Ethernetbraut. Ef skráaþjónn hefur netkort sem getur miðlað 100 Mb/sek og 10 vélar eru að lesa af diskum hans þá getur hver og ein ekki lesið nema í mesta lagi 100/10 = 10 Mb/sek að meðaltali.

Algengt er að mótöld sem tengd eru venjulegri símalínu ráði við 56 kb/sek. Með því að nota gagnaþjöppun (þ.e. með því að „skammstafa“ eins og hægt er það efni sem sent er og nota til þess reglu sem mótaldið hinu megin „kann“ svo það geti leyst úr „skammstöfununum“) er hægt að senda töluvert hraðar en þessar tölur gefa til kynna.

Hvor rás í ISDN sambandi er 64 kb/sek sem þýðir að ef tölva fær báðar rásirnar getur hún sent eða sótt 128 kb/sek. Með ADSL sambandi er hægt að sækja allt að 1,5 Mb/sek og senda allt að 384 kb/sek. Margir kaupa þó ADSL samband með minni burðargetu (t.d. 256 kb/sek í aðra áttina og 128 kb/sek í hina) enda er það ódýrara. *Breiðband* og *VDSL* samband bjóða upp á mun meiri bandvídd en ADSL og búast má við því að á næstu árum aukist flutningsgeta netlagna fyrir víðnet mjög mikið.

Netlag, flutningslag og samskiptastaðlar

Næstu lög fyrir ofan greinalagið eru netlag og flutningslag. Hugbúnaðurinn sem tilheyrir þessum lögum sér um að samskiptin fylgi *samskiptastöðlum* (á ensku *protocol*). Þessir staðlar kveða meðal annars á um hvernig pakkar sem sendir eru um netið skuli merktir, þ.e. hverju skuli skeytt framan eða aftan við þá til að tilgreina frá hverjum þeir eru og hvert þeir eiga að fara.

Til eru allmargir samskiptastaðlar en meðal þeirra algengustu eru:

IPX/SPX	Notaður þar sem netþjónar keyra NetWare netstýrikerfið frá Novell.
TCP/IP	Notaður á Internetinu. Notkun TCP/IP á staðarnetum fer vaxandi.
NetBEUI	Fylgir Windows stýrikerfinu frá Microsoft.
AppleTalk	Er einkum notaður þar sem Macintosh tölvur frá Apple eru tengdar saman í net.

Að tengja saman tvö eða fleiri tölvunet

Eins og nefnt hefur verið tengjast mörg staðarnet við Internetið. Einnig er algengt að tvö eða fleiri staðarnet séu samtengd. Ýmiss konar búnaður er notaður til að tengja saman tvö eða fleiri tölvunet:

- *Millinetagátt* (á ensku *gateway*) er notuð til að tengja saman net sem byggja á ólíkum samskiptastöðlum (og e.t.v. líka ólíkum tengibúnaði).
- *Beinir* (á ensku *router*) tengir saman tvö net sem nota sama samskiptastaðal en ef til vill ólíkan tengibúnað (t.d. tvö TCP/IP net þar sem annað notar Ethernet tengingar og hitt notar Tókahring). Til að tengja tvö net með beini þurfa netlög beggja að vera eins en lögin þar fyrir neðan mega vera ólík.
- *Brú* (á ensku *bridge*) er notuð til að tengja saman tvö net sem eru sömu gerðar, þ.e. nota sams konar greinalag og sama samskiptastaðal. Eðlislögin mega vera ólík.

Þegar staðarnet eru tengd við Internetið eða annað víðnet er tölvan sem tengist millinetagátt eða beini oftast látin keyra hugbúnað sem kallast *eldveggur*. Hlutverk eldveggja er að fylgjast með umferð inn á staðarnetið og út af því og gæta þess að pakkar berist ekki nema þeir uppfylli skilyrði sem sett eru af umsjónarmönnum staðarnetsins. Þessi skilyrði geta verið af ýmsu tagi en yfirleitt er þeim a.m.k. ætlað að koma í

veg fyrir að tölvuþrjótar geti notað aðgang að víðneti til að snuðra eða grauta í gögnum á staðarnetinu.

Verkefni 6.2.a

Finndu út hvaða samskiptastaðlar og hvers konar tengibúnaður er notaður á netinu í þínum skóla eða vinnustað. Finndu einnig út hvort netið í skólanum (eða á vinnustaðnum) er tengt við önnur net og ef svo er hvers konar búnaður er þá notaður til þess.

Verkefni 6.2.b

Ef þú hefur tölvu heima hjá þér sem tengist Internetinu gerðu þá grein fyrir:
a) Eðlis- og greinalagi tengibúnaðarins;
b) Hvaða staðlar eru notaðir í net- og flutningslagi;
c) Hversu hraðvirkt sambandið er;
d) Hvaða möguleikar eru á hraðvirkara sambandi og hvað þeir kosta.

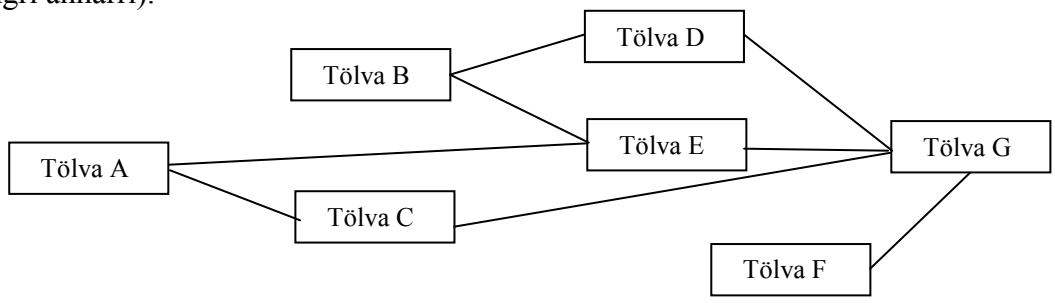
6.3 Internetið

Internetið og TCP/IP

Internetið (sem sumir kalla bara *Netið* með ákveðnum greini og stórum staf) er víðnet sem nær um allan heim. Það tengir saman ólíkar vélar (PC tölvur, Apple Macintosh tölvur, UNIX vinnustöðvar o.fl.) og leyfir alls konar tengibúnað. Með þeim hugtökum sem kynnt voru í kafla 6.2 má orða þetta svo að eðlis- og greinalag nettenginga á Internetinu megi vera af hvaða tegund sem er. Hins vegar verða allar tölvur sem tengjast því að fylgja sama samskiptastaðli á netlagi. Þessir samskiptastaðall heitir *IP* (*Internet protocol*). Flestar vélar á Internetinu fylgja líka staðli sem heitir *TCP* (*Transmission Control Protocol*) og tekur til flutningslagsins. Oft er nöfnum þessara tveggja staðla slengt saman og talað um *TCP/IP net*.

Önnur net en Internetið sem nota TCP/IP samskiptastaðlana eru stundum kölluð *internet* með litlum upphafsstaf og TCP/IP staðarnet er stundum kallað *innranet* (á ensku *intranet*).

Internetið hefur vaxið mjög hratt frá því um 1990. Þessi hraði vöxtur er mögulegur vegna þess að netinu er ekki miðstýrt. Til að bæta einni tölvu við það dugar að tengjast einhverri vél sem fyrir er. Annað mikilvægt einkenni Internetsins er að allar tölvur sem tengjast því eru jafnréttháar, það er engin miðstöð og engin vél er ómissandi. Ef ein bilar geta hinar samt verið í sambandi (nema þær sem tengjast aðeins biluðu vélinni og engri annarri).



Hugsum okkur til dæmis að sjö tölvur tengist eins og myndin sýnir og A hafi venjulega samband við B gegnum E. Ef E bilar getur A áfram haft samband við B gegnum C, G og D. Ef G bilar hins vegar þá er F sambandslaus.

Vélarnar á myndinni tengjast hvorki í brautarnet, hringnet né stjörnunet og sama gildir um Internetið, tengingarnar á því mynda ekkert einfalt form.

Flestir sem hafa samband við Internetið kannast við *Veraldarvefinn* (sem margir kalla *Vefinn*) og í daglegu tali er stundum ekki gerður greinarmunur á Netinu og Vefnum. Netið er þó notað til að gera margt fleira en að miðla vefsíðum. Meðal annars til að flytja tölvupóst, veita fólki aðgang að spjallrásum (*IRC*, *internet relay chat*), flytja skrár með *FTP* (*file transfer protocol*) flutningi, taka þátt í ráðstefnum og umræðuhópum og leika tölvuleiki.

IP-tölur

Til að tölvur geti skipst á boðum eftir IP staðli verða þær hver og ein að hafa einkennisnúmer sem kallast *IP-tala*. Á sama neti mega engar tvær tölvur hafa sömu *IP-tölu*. Þótt enginn eigi Internetið eða ráði yfir því þarf að vera stjórn á úthlutun IP-talna til að allir sem á þurfa að halda geti fengið tölur og engir tveir fái sömu tölu. Yfirstjórn þessara mála er hjá bandarískri stofnun sem kallast *InterNIC* (*Internet Network Information Center*). Hún úthlutar stofnunum á borð við *Intís* (*Internet á Íslandi hf.*) IP-tölum sem þær svo úthluta til fyrirtækja og einstaklinga.

IP-tölur eru yfirleitt ritaðar með 4 tölum á bilinu 0 til 255 og hafður punktur á milli. Til dæmis hefur ein af vélum Fjölbrautaskóla Vesturlands IP-töluna 212.30.219.178. Hægt er að rita 4 bæti (þ.e. fjórar tölur milli 0 og 255) á $256^4 = 4.294.967.296$ vegu. Á Internetinu er því pláss fyrir rúmlega fjóra milljarða véla. Þetta rúm nýtist ekki nema að hluta því IP tölum er úthlutað í blokkum. Stofnun fær kannski til ráðstöfunar allar IP-tölur sem byrja á 167.1. Þetta eru $256^2 = 65.536$ tölur. Ef stofnunin rekur 536 tölvur á Internetinu eru 65.000 af þessum tölum ónotaðar. Til að tryggja rúm fyrir nógu margar vélar er í ráði að taka upp nýjan staðal fyrir IP-tölur og rita þær með 16 bætum (16 tölum milli 0 og 255). Með því móti verður rúm fyrir um það bil $3.4 \cdot 10^{38}$ vélar (þ.e. um það bil 50.000.000.000.000.000.000.000.000.000 á hvern jarðarbúa). Þetta ætti að duga um langan aldur.

Hverri IP-tölu má skipta í tvo hluta þar sem sá fyrri táknar *undirnet* (á ensku *subnet*) og sá seinni táknar einstaka vél. Ef fyrirtæki hefur til dæmis fengið úthlutað öllum IP-tölum sem byrja á 167.1 þá mynda tölurnar 167 og 1 númer undirnets sem það rekur. Tölurnar tvær sem koma þar fyrir aftan aðgreina svo einstakar vélar á undirnetinu. Undirnetum á Internetinu er skipt í þrjá flokka eftir því hvort númer þeirra er táknað með 1, 2 eða 3 bætum. Þessir flokkar kallast A, B og C.

- Undirnet í A flokki hafa númer á bilinu 1 til 9 eða 11 til 126. Hvert slíkt undirnet getur haft 253^3 vélar því aðeins eitt bæti fer í að númera undirnetið og þá eru 3 eftir til að auðkenna vélar innan þess. Langt er síðan hætt var að úthluta IP-tölum í þessum flokki.
- Undirnet í B flokki hafa tveggja bæta númer á bilinu 128.1 til 191.254. Hvert slíkt undirnet getur haft 256^2 vélar.
- Undirnet í C flokki hafa 3 bæta númer á bilinu 193.0.0 til 223.255.254. Hvert slíkt undirnet getur haft 256 vélar.

Þótt engar tvær tölvur á Internetinu geti haft sömu IP-tölu er ekkert því til fyrirstöðu að tvær tölvur sem tilheyra hvor sínu staðarneti hafi sömu IP-tölu. Það er til siðs að láta

tölvur á staðarnetum sem nota IP-samskiptastaðalinn hafa IP-tölur sem eru ekki notaðar á Internetinu, þetta eru tölur sem byrja á 10, 127 eða 192.

Umdæmisheiti og DNS

Það er erfitt fyrir fólk að muna IP tölur. Þess vegna er flestum vélum, sem hafa fast samband við Internetið, gefið nafn. Slíkt nafn kallast *lén* eða *umdæmisheiti* (á ensku *domain name*). Vélar sem ekki hafa fast samband við Internetið heldur tengjast aðeins stund og stund, t.d. gegnum mótald, hafa yfirleitt ekki neitt nafn heldur aðeins IP-tölu. Stundum hafa þær fasta IP-tölu (þ.e. alltaf þá sömu) en algengast er að heimilistölvur sem tengjast með því að hringja í vél hjá einhverri Internetþjónustu fái IP-tölu (af undirneti þjónustuaðila) úthlutað um leið og þær ná sambandi og ekki endilega sömu IP-tölu í hvert skipti. Þetta er allt í lagi ef aðeins á að nota sambandið til að sækja póst, gramsa á vef eða tengjast spjallrás en eigi tölva að geyma gögn, t.d. vefsíður, og veita öðrum aðgang að þeim þarf hún að hafa fasta IP tölu, vera stöðugt í sambandi og helst að hafa umdæmisheiti.

Sama vél getur haft mörg nöfn (alveg eins og símanúmer getur verið skráð á mörg nöfn í símaskránni). Til dæmis gengur vél sem hefur númerið 194.105.226.1 bæði undir nafninu www.simnet.is og www.isholf.is. Hér eru nokkur dæmi um umdæmisheiti og samsvarandi IP tölur:

www.fva.is	212.30.219.178
www.ismennt.is	212.30.217.70
pop.ismennt.is	212.30.217.11
www.simnet.is	194.105.226.1
www.isholf.is	194.105.226.1
ftp.funet.fi	193.166.3.2

Sé tölva á Internetinu beðin um samband við 212.30.217.70 þá nær hún sambandi við eina af vélum Íslenska menntanetsins en sé hún beðin um samband við www.ismennt.is þarf hún fyrst að komast að því hvaða IP-tala samsvarar þessu nafni (alveg eins og ef við erum beðin að hringja í 431-1200 þá getum við gert það strax en ef við erum beðin að hringja í Hróa hött á Akranesi þá þurfum við fyrst að fletta því upp hvaða síma-númer samsvarar nafninu Hrói höttur).

Til þess að finna hvaða IP tala samsvarar umdæmisheiti eins og www.ismennt.is eða ftp.funet.fi hefur tölva samband við *umdæmisheitaþjónustu* (á ensku *DNS* eða *domain name server*). *Umdæmisheitaþjónusta* eða *DNS* er tölva sem keyrir forrit sem sjá um að fletta upp IP-tölum fyrir aðrar tölvur. Til að tölva (t.d. heimilistölva með mótald) geti náð sambandi við vélar sem nefndar eru nöfnum, en ekki bara IP-tölum, þarf hún að „vita“ IP tölu a.m.k. einnar tölvu sem sinnir hlutverki DNS.

Hér á landi er umdæmisheitum úthlutað af Intís. Íslensk umdæmisheiti enda á is, sem er skammstöfun landsins. Síðasti liður umdæmisheitis er tveggja stafa skamm-stöfun sem segir til um í hvaða landi netfangið er skráð nema ef það er í Bandaríkjunum þá er hann þriggja stafa skammstöfun sem segir til um hvaða flokki stofnunin sem á tölvuna tilheyrir. Dæmi:

dk	Danmörk	fi	Finnland
is	Ísland	no	Noregur
se	Svíþjóð	uk	Bretland
edu	Menntastofnun	mil	Stofnun á vegum bandaríska hersins
gov	Stofnun á vegum bandaríska ríkisins	com	Ýmis fyrirtæki
net	Ýmis fyrirtæki, félög og stofnanir	org	Ýmis fyrirtæki, félög og stofnanir

Oftast er næstsíðasti liður umdæmisheitis nafn (eða skammstöfun á nafni) fyrirtækis eða stofnunar. Þannig hafa allar vélar Íslenska menntanetsins nafn sem endar á ismennt.is og vélar Háskóla Íslands hafa nafn sem endar á hi.is.

Á vélum sem miðla vefsíðum er fyrsti liður umdæmisheitis oft www (fyrir *World wide web* sem er enskt heiti Veraldarvefsins) eins og í www.ismennt.is og www.fva.is. Vélar sem sinna póstþjónustu hafa oft umdæmisheiti sem byrjar á pop eða mail eins og pop.ismennt.is og vélar sem bjóða upp á skráaflutning með ftp hafa gjarna umdæmisheiti sem hefst á ftp eins og ftp.funet.fi.

Póstfang og veffang (URL)

Tengist maður vél sem heitir ismennt.is undir nafninu atli þá er *póstfang* (sem einnig kallast *netfang*, á ensku *E-mail address*) hans

atli@ismennt.is.

Sé vefsíða geymd í skrá sem heitir karamella.htm í skráasafninu /nammi á vélinni www.fva.is þá er *veffang* (á ensku *URL*, *uniform resource locator*) hennar:

http://www.fva.is/nammi/karamella.htm

Þar sem www.fva.is stendur fyrir IP töluna 212.30.219.178 má eins rita veffangið:

http://212.30.219.178/nammi/karamella.htm

Veffang byrjar á skammstöfun sem segir hvers konar gögn er um að ræða. Dæmi:

file	skrá á eigin diski
ftp	skrá sem miðlað er af FTP þjóni
http	vefsíða sem miðlað er af vefþjóni
news	Usenet ráðstefna
telnet	tenging með skjáhermi

Á eftir skammstöfun (sem oftast er http) kemur tvípunktur og tvö skástrik og svo nafn vélar (annað hvort IP tala eða umdæmisheiti), þá vísun í skrá sem geymd er á vélinni.

Á tölvum sem keyra stýrikerfið UNIX eða Linux fá notendur oft skráasöfn sem hægt að finna með því að setja ~ framan við nöfn þeirra. Notandi sem tengist www.fva.is undir nafninu atli á til dæmis skráasafn sem hægt er að komast í með því að biðja um ~atli. Vefsíður sem atli setur í skráasafn sitt er því hægt að nálgast með því að gefa upp veffangið http://www.fva.is/~atli/

Það er nokkuð misjafnt hvað vefþjónar gera ef engin skrá er tilgreind í veffangi heldur aðeins skráasafn eins og í

http://www.fva.is/~atli/

eða

http://212.30.219.178/nammi/

Sumir vefþjónar senda skrá með heitinu Welcome.html sé hana að finna í skráasafninu, sumir senda index.html og sumir senda lista yfir allar skrár í skráasafninu.

Miðlarar, biðlarar og hlið

Tölva sem miðlar gögnum til annarra véla, t.d. pósti eða vefsíðum kallast *miðlari* (á ensku *server*). Vélin www.fva.is er til dæmis *vefmiðlari* því hún geymir vefsíður og keyrir forrit sem kallast *vefþjónn* og miðlar vefsíðum til annarra véla.

Tölva sem nýtir sér þjónustu miðlara kallast *biðlari* (á ensku *client*). Flestar heimilistölvur sem tengjast Internetinu um mótald eru aðeins biðlarar, þær nota sér þjónustu annarra véla en láta ekkert í té.

Tæknilega er ekkert því til fyrirstöðu að venjuleg heimilistölva sé miðlari. Hún getur til dæmis keyrt vefþjón og veitt öðrum vélum aðgang að vefsíðum sem geymdar eru á diski hennar. Yfirleitt er þó lítið vit í að láta vél bjóða upp á slíka þjónustu nema hún sé í stöðugu sambandi og hafi a.m.k. fasta IP-tölu og helst líka umdæmisheiti. Það getur enginn notað þjónustuna nema hann nái sambandi við vélina og það er vandkvæðum bundið ef hún er ekki tengd nema við og við eða hefur eina IP-tölu í dag og aðra á morgun.

Til að tölva geti verið miðlari þarf hún að keyra *þjónustuforrit*, t.d. *póstþjón* ef hún á að miðla pósti, *vefþjón* ef hún á að miðla vefsíðum, *IRC-þjón* ef hún á að miðla spjallrásum. Fyrir hvern þjónustuflokk á Internetinu þurfa að vera til tvenns konar forrit, þjónustuforrit sem miðlarar keyra og notendaforrit sem keyrð eru á biðlurum.

Þegar biðlari hefur samband við miðlara og biður hann að miðla sér einhverju eins og pósti eða vefsíðu þarf hann að tilgreina hvers konar þjónustu hann vill. Þetta gerir hann með því að tilgreina númer sem kallað er *hlið* (á ensku *port*). Algeng hlið eru:

20	fyrir ftp gögn
21	fyrir ftp skipanir
25	til að senda póst
80	fyrir vefsíður
110	til að sækja póst
119	fyrir Usenet ráðstefnur

Í undantekningartilvikum eru vefsíður afgreiddar um annað port en númer 80. Þá er hægt að nálgast þær með því að rita númer portsins á eftir umdæmisheitinu. Dæmi:

http://www.afbrigdileg_vjel.is:81/nammi/karamella.html

Verkefni 6.3.a

Hverjar þessara IP-talna tilheyra sömu undirnetum

124.231.7.7	124.199.11.13	125.199.11.13
189.5.5.1	189.5.6.1	189.6.6.1
194.2.2.1	194.2.2.2	194.2.3.1?

Verkefni 6.3.b

Vefsíða hefur veffangið http://www.pt.dk/nik/uff.htm
Hvert er umdæmisheiti vélarinnar sem hún er geymd á og í hvaða landi ætli hún sé?

Verkefni 6.3.c

Ef þú hefur heimilistölvu sem tengist Internetinu finndu þá

a) Umdæmisheiti vélarinnar sem þú tengist.

b) IP-tölu vélarinnar sem tölvan þín tengist. Ef þú notar Windows stýrikerfið getur þú gert þetta með því að keyra forrit sem heitir *ping* og fylgir með Windows. Þú þarft að opna DOS-glugga með því að velja *Run* á *Start* valmyndinni og skrifa *command* í reitinn sem þá birtist. Þú keyrir svo *ping* með því að rita nafn þess í skipanalínuna og umdæmisheiti vélarinnar þar fyrir aftan, dæmi:

    ping www.fva.is

c) Finndu IP tölu vélarinnar sem þú notar. Ef hún keyrir Windows stýrikerfið getur þú gert þetta með því að keyra forrit sem heitir *winIPcfg* og fylgir með Windows. (Ef vélin er ekki tengd við TCP/IP net segir þetta forrit að hún hafi IP-töluna 0.0.0.0.)

Til upprifjunar

1. Hvað er:
   ADSL, AppleTalk, bandbreidd, beinir, biðlari, brautarnet, brú, eðlislag, eldveggur, Ethernet, greinalag, hringnet, hub, hugbúnaðarþjónn, innranet, internet, Internetið, InterNIC, Intís, IP-tala, IPX/SPX, ISDN, jafningjanet, LAN, lén, lykilorð, MAN, miðlara-biðlaranet, miðlari, millinetagátt, mótald, NetBEUI, netstýrikerfi, NetWare, netþjónn, notandanafn, notkunarlag, OSI, samskiptastaðall, skráaþjónn, staðarnet, stjörnunet, TCP/IP, Tókahringur, umdæmisheiti, Vefurinn, vinnustöð, víðnet, WAN?

2. Hver er tilgangur þess að tengja tölvur saman í staðarnet?

3. Hver er sérstaða IP-talna sem byrja á 10, 127 eða 192?

4. Hvaða munur er á undirnetum í A, B og C flokki?

7. kafli

Algrím, flækjustig, reiknanleiki

7.1 Algrím

Skilgreining á hugtakinu *algrím*

Algrím (á ensku *algorithm*) er eitt af undirstöðuhugtökum tölvufræðinnar. Það merkir aðferð sem er í senn *endanleg*, *örugg* og *ótvíræð*. Lítum á hvað þessi þrjú orð merkja:

- Þegar sagt er að aðferð sé *endanleg* er átt við að hægt sé að ljúka við hana í endanlegum fjölda skrefa. Skrefin geta skipt milljónum en þau mega ekki vera óendanlega mörg.
- Að aðferð sé *örugg* þýðir að sé henni fylgt þá takist örugglega að ljúka verki eða komast að niðurstöðu. Þetta útilokar til dæmis að það að kaupa happdrættismiða geti talist *algrím* til að græða peninga.
- Þriðja skilyrðið, að aðferðin sé *ótvíræð*, felur í sér að eftir hvert skref sé alveg klárt hvað á að gera næst þannig að til að fylgja fyrirmælunum dugi smásmuguleg nákvæmni, það þurfi hvorki að nota ímyndunarafl né sköpunargáfu. Þetta útilokar ekki að aðferð innifeli t.d. teningakast eða aðra aðgerð sem hefur tilviljanakennda útkomu, aðeins að fyrirmæli séu tvíræð eða ónákvæm.

Fjölmörg algrím eru vel þekkt úr daglegu lífi. Sem dæmi má taka reikniaðferðirnar fjórar sem kenndar eru í grunnskólum, uppskriftir í matreiðslubókum, prjónauppskriftir í handavinnublöðum og leiðbeiningar og forskriftir af ýmsu tagi.

Tilgáta Church og Turings

Af ástæðum sem fjallað verður um í kafla 7.3 höfðu stærðfræðingar og rökfræðingar á fyrri helmingi 20. aldar áhuga á að skilgreina nákvæmlega hvað flest í hugtakinu *stærðfræðileg aðferð*, eða með öðrum orðum að skilgreina *algrím til að vinna stærðfræðileg verkefni*. Fram komu nokkrar ólíkar skilgreiningar. Sú þekktasta var sett fram af *Alan Turing* (1912 – 1954) í grein sem hann birti árið 1936. Hann skilgreindi vél sem gat beitt nokkrum einföldum aðgerðum á runur af tvenns konar merkjum sem rituð voru á renning og færði rök að því að hægt væri að byggja allar *stærðfræðilegar aðferðir* úr þessum einföldu aðgerðum. Eins og nefnt var í kafla 1.2. kallast vélar eins og þær sem Turing lýsti *Turingvélar*. Aðrar skilgreiningar voru settar fram af *Alonso Church* (1903 – 1937) og *Emil L. Post* (1897 – 1954). Skilgreiningar Church og Post byggðust á því að búa til eins konar forritunarmál og rökstyðja að hægt sé að nota þau til að orða allar *stærðfræðilegar aðferðir*. Síðar varð ljóst að skilgreiningar þessara þriggja manna á stærðfræðilegri aðferð eru jafngildar og allar aðferðir sem hægt er að lýsa samkvæmt þeim er hægt að orða á öllum fullgildum forritunarmálum og láta hvaða tölvu sem er vinna eftir. Af þessu leiðir að öll forritunarmál eru jafngild og þótt eitt mál henti betur en önnur til að vinna eitthvert verk er ekkert algrím til sem bara er hægt að orða á sumum forritunarmálum en ekki öðrum. Þar sem möguleikar tölvu ráðast algerlega af

vélamálinu sem gjörvinn í henni vinnur eftir og vélamál eru fullgild forritunarmál leiðir líka af þessu að allar tölvur geta það sama. Ein tölva er kannski margfalt fljótari en önnur að vinna eitthvert verk en það er ekki til neitt algrím sem sumar tölvur geta framkvæmt en aðrar ekki.

Það hefur ekki beinlínis verið sannað að skilgreiningar *Post*, *Church* og *Turing* á *stærðfræðilegri aðferð* séu réttar þannig að útilokað sé að nokkurn tíma finnist aðferðir sem ekki falla undir þær. En þar sem allar stærðfræðilegar aðferðir sem þekktar eru falla undir þessar skilgreiningar og enginn hefur minnstu hugmynd um hvernig aðferð sem gerir það ekki gæti hugsanlega verið trúa flestir þeirri tilgátu sem kennd er við *Church* og *Turing* að það sé *útilokað að búa til stærðfræðilegt algrím sem ekki fellur undir þessar skilgreiningar*.

Þar sem sýnt hefur verið fram á að allar tölvur eru jafngildar *Turing vélum* leiðir af *tilgátu Church og Turings* að hægt er að *láta hvaða tölvu sem er vinna eftir hvaða stærðfræðilegu algrími sem er*. Stundum er einfaldlega talað um að tölvur geti unnið eftir hvaða algrími sem er og að forritunarmál dugi til að orða hvaða algrím sem er. Hér þarf að setja nokkra fyrirvara. Prjónauppskrift í handavinnublaði er fullgilt algrím en venjuleg borðtölva getur ekki unnið eftir henni og yfirleitt alls ekki haldið á prjónum, enda hefur hún engar hendur.

Eins og fjallað var um í 4. kafla eru öll gögn sem tölva vinnur með skráð í minni hennar sem náttúrulegar tölur á formi tvíundakerfis. Við getum látið þessar tölur standa fyrir bókstafi, myndir og ótalmargt annað. Við getum líka tengt ýmiss konar vélar og tæki við tölvu og látið hreyfingar þeirra stjórnast af innihaldi einstakra staða í minninu. Það er t.d. ekkert útilokað að láta tölvu stjórna prjónavél. Tölva sem getur unnið eftir hvaða stærðfræðilegu algrími sem er getur því unnið úr gögnum sem hægt er að skrá með tölum eftir hvaða formúlu sem er og sent sjálfvirkum vélum merki og skipanir eftir hvaða reglu sem vera skal.

Alan Turing áleit mögulegt að forrita tölvur til að vinna hvaða hugarstarf sem er. Hvort hann hafði rétt fyrir sér um þetta efni veit enginn með vissu enda veit enginn nógu vel hvernig mannshugur virkar til að geta fullyrt af eða á um hvort hann vinnur einhver verk sem ekki er hægt að vinna samkvæmt stærðfræðilegri aðferð. Hitt er hins vegar ljóst að síðan Turing setti kenningar sínar fram hefur tekist að forrita tölvur til að vinna ótal verk sem samtímamenn hans óraði ekki fyrir að vél gæti nokkurn tíma unnið.

Runa, skilyrði, endurtekning

Algrím sem tölvur vinna eftir eru mynduð úr þrenns konar einingum sem eru *runur*, *skilyrði* og *endurtekning*.

Með *runu* er ekki átt við neitt flóknara en að skipanir séu framkvæmdar í röð, hver á eftir annarri. Eftirfarandi algrím til sjóða kartöflur er til dæmis *runa* af 6 skipunum:

```
1.  Settu kartöflur í pott.
2.  Láttu renna heitt vatn í pottinn þar til það flæðir
    yfir kartöflurnar.
3.  Settu pottinn á eldavélarhellu.
4.  Kveiktu á hellunni.
5.  Bíddu í 30 mínútur.
6.  Slökktu á hellunni.
```

Til að framkvæma algrímið þarf að byrja á fyrstu skipuninni, framkvæma hverja skipun einu sinni og ljúka henni áður en sú næsta er framkvæmd. Þegar lokið er við að framkvæma síðustu skipunina er búið að vinna það verk sem algrímið lýsir.

Þessi aðferð til að sjóða kartöflur er gölluð. Ef kartöflurnar eru litlar þá verða þær mauksoðnar eftir 30 mínútur og ef þær eru stórar verða þær hálfhráar þegar slökkt er á hellunni. Með því að nota *endurtekningu* og *skilyrði* er hægt að endurbæta algrímið svona:

```
1.  Settu kartöflur í pott.
2.  Láttu renna heitt vatn í pottinn þar til það flæðir
    yfir kartöflurnar.
3.  Settu pottinn á eldavélarhellu.
4.  Kveiktu á hellunni.
5.  Bíddu í 20 mínútur.
6.  Stingdu prjóni í eina kartöflu.
7.  Ef prjónninn stingst í gegnum kartöfluna
    þá farðu í línu númer 10
8.  Bíddu í 3 mínútur.
9.  Farðu í línu númer 6.
10. Slökktu á hellunni.
```

Skilyrðissetningin í línu númer 7 segir að það eigi að gera eitthvað (nefnilega að fara í línu númer 10) ef tiltekið *skilyrði* (að prjónn stingist gegnum kartöflu) er uppfyllt.

Þessi endurbætta aðferð til að sjóða kartöflur felur í sér *endurtekningu* því þegar komið er í línu númer 9 er stokkið upp í línu númer 6. Þessari *endurtekningu* lýkur þegar *skilyrðið* í línu 7 er satt því þá er stokkið í línu númer 10 og með henni endar algrímið. Það mætti líka nota sérstaka skipun fyrir *endurtekningu* og orða algrímið einhvern veginn svona:

```
1. Settu kartöflur í pott.
2. Láttu renna heitt vatn í pottinn þar til það flæðir
   yfir kartöflurnar.
3. Settu pottinn á eldavélarhellu.
4. Kveiktu á hellunni.
5. Bíddu í 20 mínútur.
6. Endurtaktu
7.    Stingdu prjóni í eina kartöflu.
8.    Bíddu í 3 mínútur.
9. þar til prjónn stingst gegnum kartöflu
9. Slökktu á hellunni.
```

Verkefni 7.1.a

Orðaðu á venjulegu máli:
Algrím til að sjóða 4 egg í potti;
Algrím til að skipta um dekk á bíl.

Einfalt mál sem dugar

Skipanirnar sem kartöflusuðualgrímið er myndað úr eru ansi flóknar. Til að framkvæma þær þarf flóknar hreyfingar enda er matreiðsla erfið list. Til að lýsa reikniaðferðum þarf ekki neinar svona flóknar skipanir. Það dugar að hafa eftirfarandi:

- Skipun til að *gefa breytum gildi*. Við getum t.d. táknað hana með jafnaðarmerki og látið **`x = 7`** tákna að breytan x skuli fá gildið 7.
- *Reikniaðgerðir*. Öll forritunarmál innihalda reikniaðgerðirnar *samlagningu*, *frádrátt*, *margföldun* og *deilingu* en það má komast af með minna, raunar dugar að hafa aðferð til að bæta einum við gildi breytu. Við getum notað ++ og látið **`x++`** tákna að gildi x skuli hækkað um 1.
- Möguleika á að *bera saman tvær tölur*. Öll forritunarmál innihalda samanburðaraðgerðirnar *jafnt*, *stærra* og *minna*. Strangt tekið dugar að hafa eina þessara aðgerða t.d. aðgerð til að athuga hvort tvær tölur séu jafnar. Við getum til dæmis notað tvö jafnaðarmerki og látið **`(x == 7)`** tákna setningu sem er sönn ef gildi breytunnar x er 7.
- *Endurtekningu*. Öll mótuð forritunarmál hafa skipanir til að tákna endurtekningu samsvarandi þeirri sem er notuð í síðustu gerð kartöflusuðualgrímsins hér að framan. Það má þó komast af með skipun til að hoppa um forritið. Við getum til dæmis notað orðið *hoppa* og látið **`hoppa 6`** tákna að stokkið skuli í línu númer 6 og **`hoppa x`** að hoppað skuli í línu með númerinu sem geymt er í breytunni x.
- *Skilyrði*. Við getum til dæmis notað orðið *ef* og látið skipunina **`ef (x == 5) hoppa 6`** tákna að stokkið skuli í línu 6 ef gildi breytunnar x er 5 og skipunina **`ef (x == y) n++`** tákna að ef x og y innihalda sama gildi þá skuli gildi breytunnar n hækkað um einn.

  Hér hefur verið lýst afar einföldu forritunarmáli. Sé gert ráð fyrir að breytur geti geymt hvað háar heiltölur sem er þá samsvarar þetta mál skilgreiningum Turing, Church og Post á stærðfræðilegri aðferð. Á því er hægt að orða allar aðferðir sem tölvur geta unnið eftir og ef *tilgáta Church og Turings* er rétt þá dugar það til að orða fyrirmæli um alla útreikninga sem yfirleitt er hægt að vinna eftir nokkurri aðferð.

- Þótt það sem talið var hér að ofan dugi skulum við bæta orðinu *skrifa* við orðaforða málsins þannig að sé skipunin **`skrifa(x)`** gefin þá sé gildi breytunnar x skrifað á skjá eða prentara.

  Þótt þetta mál sé einfalt er það miklu erfiðara í notkun er nokkurt venjulegt forritunarmál. Það kostar dálítil heilabrot að búa til forrit á því sem vinnur jafneinfalt verk og að leggja saman tvær tölur. Hér slíkt forrit. Það setur tölurnar 5 og 7 í breyturnar x og y, leggur þær saman og setur útkomuna í breytuna z og skrifar gildi hennar.

```
1.   x = 5
2.   y = 7
3.   z = y
4.   n = 0
5.   ef (n == x) hoppa 9
6.   n++
7.   z++
8.   hoppa 5
9.   skrifa(z)
```

  Þetta forrit virkar þannig að z er látið vera jafnt og y og n jafnt og núll og x svo bætt við z með því að hækka n og z um einn aftur og aftur þar til n er jafnt og x.

Verkefni 7.1.a

Búðu til forrit á lágmarksmálinu sem setur tölur í tvær breytur x og y og reiknar y - x.

Auðugra mál og dæmi um algrím

Það lágmarksmál sem hér hefur verið lýst er áhugavert vegna þess eins að það er dæmi um hvað lítið þarf til að smíða allar mögulegar reikniaðferðir. Öll raunveruleg forritunarmál hafa auðugri orðaforða en þetta lágmarksmál þannig að ein lína getur lýst aðferð sem fyllti margar blaðsíður ef hún væri skrifuð á lágmarksmálinu.

Í þeim dæmum sem hér fara á eftir verður notað ímyndað forritunarmál sem svipar til mótaðra mála en er vonandi skiljanlegra því skipanirnar eru látnar minna á setningar á íslensku með sömu merkingu.

Í kafla 2.1 var kynnt aðferð til að umrita jákvæðar heiltölur af tvíundakerfi í tugakerfi. Á blendingi af íslensku og máli sem líkist dálítið *C* eða *Java* gæti þessi aðferð litið svona út:

```
heiltala x;
heiltala t;
strengur s;
lesa(x);
endurtaka meðan (x > 0)
{
   t = x % 2;
   x = x / 2;
   ef (t == 1)
   {
      bæta '1' framan á strenginn s;
   }
   annars
   {
      bæta '0' framan á strenginn s;
   }
}
```

Hér er gert ráð fyrir að skipunin `t = x % 2` gefi breytunni t gildi sem er fengið með því að reikna hvað gengur af þegar 2 er deilt upp í x. Skipunin `x = x / 2` á að gefa x gildi sem fæst með því að deila 2 í gildið sem hún hefur fyrir. Þar sem x er heiltala táknar / heiltöludeilingu þannig að ef x er t.d. 5 á fæst útkoman 2 (en ekki 2,5) úr `x / 2`.

Þegar búið er að skrifa algrím eins og þetta er hægt að spyrja ýmissa spurninga um það eins og:

- Er algrímið rétt? (Vinnur það verkið sem það á að vinna rétt undir öllum kringumstæðum?)
- Hvað tekur keyrsla þess langan tíma?
- Hvað þarf algrímið mikið minni?

Um réttmæti algríms og villur í forritum verður fjallað í kafla 8.3. Hér skal þess aðeins getið að ef aðferðinni er ætlað að umrita jákvæðar heilar tölur sem ekki eru stærri en svo að þær rúmist í einni heiltölubreytu þá er hún rétt. En ef henni er ætlað að umrita hvaða heiltölu sem er þá er hún ekki rétt. Hún virkar hvorki á neikvæðar tölur né á tölur sem eru of stórar til að rúmast í einni heiltölubreytu.

Við getum ekki vitað hvað keyrsla algrímsins tekur langan tíma nema við vitum hvað vélin (eða maðurinn) sem framkvæmir það er lengi með hverja skipun og það er

auðvitað misjafnt eftir vélartegundum. Við getum samt gert ráð fyrir að þegar sama skipanaromsa er endurtekin aftur og aftur þá taki það álíka langan tíma í hvert skipti.

Gerum ráð fyrir að það taki alls k sekúndur að framkvæma reikniaðgerðirnar tvær `t = x % 2` og `x = x / 2`, samanburðaraðgerðina `(t == 1)` og að bæta staf framan á streng. Hver umferð gegnum slaufuna (þ.e. skipanablokkina sem er endurtekin) tekur þá k sekúndur. Það hve oft slaufan er endurtekin veltur á stærð tölunnar x. Þar sem x helmingast í hvert skipti og það er hætt þegar x er komið niður í 0 þá er fjöldi umferða um það bil $\log_2(x)$ (þ.e. það veldi sem þarf að hefja 2 í til að x komi út). Tíminn sem tekur að framkvæma forritið er þá um það bil $k \cdot \log_2(x)$ þar sem k er breytilegt frá einni vél til annarrar.

Minnisplássið sem þetta algrím þarf er summan af því rúmi sem allar breytur þess taka í minni tölvunnar.

7.2 Röðunaraðferðir og flækjustig

Flækjurými og flækjutími

Stundum er talað um flækjustig aðferða. Þegar sagt er að ein aðferð hafi hærra flækjustig en önnur er stundum ekki átt við neitt annað en að hún sé flóknari, a.m.k. í augum þess sem talar. Stundum er verið að vísa til fræðilegra mælikvarða á *flækjutíma* (á ensku *time complexity*) og *flækjurými* (á ensku *space complexity*).

Algrímið hér að framan til að skrifa tölur í tvíundakerfi hafði flækjutímann $k \cdot \log_2(x)$ þar sem k var fasti háður gerð vélar og x talan sem skrifa átti í tvíundakerfi. Oft er flækjutími algríms fall af stærð eða umfangi þeirra gagna sem því er beitt á. Flækjurými er það minnispláss sem algrím þarf undir útreikninga og eins og flækjutíminn er það oft fall af stærð eða umfangi gagna.

Flest verk er hægt að vinna á marga vegu. Það eru til dæmis til margar mismunandi aðferðir til að margfalda og leggja saman og þessar aðferðir eru misfljótvirkar og útheimta mismikla skriffinnsku eða hafa með öðrum orðum misjafnan flækjutíma og mismikið flækjurými. Hér verða skoðuð tvö algrím til að raða fylki af tölum og fjallað nokkrum orðum um flækjutíma þeirra.

Röðunaraðferðir

Gerum ráð fyrir að fylki, sem við skulum kalla a, geymi 10 tölur, t.d. tölurnar 11, 9, 7, 3, 1, 12, 15, 8, 4 og 2 eins og myndin sýnir.

Fylkið a	11	9	7	3	1	12	15	8	4	2
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Ein leið til að raða tölunum í röð frá hæstu til lægstu er að byrja á að fara yfir allt fylkið og finna lægstu töluna og skipta svo á henni og tölunni í hólfi númer 1. Hér er lægsta talan 1 (sem er í hólfi númer 5) og þegar búið er að skipta á henni og þeirri fremstu er innihald fylkisins:

Fylkið a	1	9	7	3	11	12	15	8	4	2
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Næst þarf svo að fara yfir fylkið frá hólfi númer 2 til enda og finna lægstu töluna (í þessu tilviki er það talan 2 í hólfi númer 10) og skipta á henni og tölunni í hólfi númer 2. Útkoman úr þessu verður svona:

Fylkið a	1	2	7	3	11	12	15	8	4	9
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Framhaldið er augljóst. Næst er farið yfir fylkið frá hólfi númer 3 til enda, lægsta talan fundin (í þessu tilviki er það talan 3 í hólfi númer 4) og skipt á henni og tölunni í hólfi númer 3.

Við getum nú orðað röðunaralgrímið á blöndu af íslensku og mótuðu forritunarmáli. Við köllum fylkið a, fjöldi talna í því er n og við vísum í tölu númer k í fylkinu með a[k].

```
fylki a = {11, 9, 7, 3, 1, 12, 15, 8, 4, 2};
heiltala n = 10;
heiltölur i, j, min, x;

fyrir öll gildi á i frá 1 til (n-1)
{
   min = i;
   fyrir öll gildi á j frá (i+1) til n
   {
      ef (a[j] < a[min])
      {
         min = j;
      }
   }
   x = a[i];
   a[i] = a[min];
   a[min] = x;
}
```

Ytri slaufan er endurtekin n-1 (í þessu tilviki 9) sinnum. Í fyrsta sinn hefur i gildið 1, næst 2 o.s.fr. Fyrsta skipunin í slaufunni (`min = i`) segir að minnsta tala sem fundist hefur sé í hólfi númer i. Innri slaufan fer svo yfir öll hólf frá númer i+1 til enda og í hvert sinn sem fyrir verður hólf með tölu sem er lægri en talan í hólfi númer min er innihaldi min breytt í númer þess hólfs. Þegar lokið er við að fara gegnum innri slaufuna taka við 3 skipanir sem skipta á innihaldi í hólfum númer min og i.

Til að reikna flækjutíma þessa algríms þarf að finna hve oft farið er gegnum slaufurnar. Auðvelt er að sjá að farið er n-1 sinnum gegnum ytri slaufuna (þar sem n er fjöldi talna í fylkinu). Ef n er há tala munar litlu á n og n-1 svo við getum sagt að farið sé um það bil n sinnum gegnum ytri slaufuna. Við hverja ferð gegnum ytri slaufuna er farið nokkrum sinnum gegnum þá innri: n-1 sinnum fyrst, næst n-2 sinnum þá n-3 sinnum o.s.fr. og að síðustu einu sinni. Þetta þýðir að farið er að meðaltali um það bil ½·n sinnum gegnum innri slaufuna í hvert sinn sem farið er gegnum þá ytri. Alls er því farið um það bil $\frac{1}{2} \cdot n^2$ sinnum gegnum innri slaufuna.

Tíminn sem tekur að framkvæma hverja skipun er háður vélargerð. Við skulum segja að það taki alls k_g sekúndur að framkvæma eina gildisgjöf og k_s sekúndur að framkvæma eina samanburðaraðgerð. Innri slaufan inniheldur eina samanburðaraðgerð og eina gildisgjöf sem er að meðaltali framkvæmd í annað hvert skipti, nefnilega þegar `(a[j] < a[min])` er satt. Ytri slaufan inniheldur 4 gildisgjafir að auki. Flækjutími

algrímsins er þá um það bil $\frac{1}{2}(k_s + \frac{1}{2}k_g)n^2 + 4k_g n$ þar sem n er fjöldi talna í fylkinu. Þetta flækjutímafall er annars stigs margliða. Til eru röðunaraðferðir sem hafa mun minni flækjutíma en þetta. Sú hraðvirkasta sem vitað er um kallast *quicksort* á ensku og hefur ekki enn fengið nafn á íslensku.

Quicksort aðferðin er nokkuð flókin og því verður henni aðeins lýst í grófum dráttum. Gerum ráð fyrir að við ætlum að raða fylki af 10 tölum eins og sýnt er hér fyrir neðan.

Fylkið a	11	9	7	3	1	12	15	8	4	2
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Við byrjum á að giska á miðgildi talnasafnsins. Þar sem tölurnar eru allar milli 1 og 15 getum við giskað á að miðgildið sé 8.

Næst er farið gegnum fylkið og allar tölur sem eru framan við miðgildið settar fremst og tölurnar sem eru ekki framan við miðgildið þar fyrir aftan. Útkoman úr því er:

Fylkið a	7	3	1	4	2	11	9	12	15	8
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Með þessu er raunar búið að skipta fylkinu í tvennt þannig að tölurnar í fyrri helmingnum eru lægri en tölurnar í seinni helmingnum.

Næst er það sama gert við hvorn helming: Giskað er á miðgildi fyrri hlutans t.d. með því að velja tölu mitt á milli 1 og upphaflega miðgildisins sem var 8. Þessi tala er 4,5 sem við rúnnum að 5. Svo setjum við tölurnar framan við 5 fremst og tölurnar aftan við 5 þar fyrir aftan. Þetta sama er gert við seinni hlutann. Þar er miðgildið 12. Eftir aðra umferð er búið að skipta fylkinu í fernt og innihald þess er:

Fylkið a	3	1	4	2	7	11	9	8	12	15
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Í þriðju umferð er hverjum þessara fjögurra hluta skipt í tvennt, nema þeim sem aðeins hafa 1 hólf, við þá er ekkert gert. Útkoman úr þessu verður:

Fylkið a	1	2	3	4	7	9	8	11	12	15
Sæti nr.	1	2	3	4	5	6	7	8	9	10

Eftir fjórðu umferð verður enginn hluti stærri en eitt hólf og tölurnar komnar í rétta röð.

Þótt *quicksort* aðferðinni hafi aðeins verið lýst í grófum dráttum höfum við nægar upplýsingar til að áætla hvers konar fall af stærð fylkis flækjutími hennar er. Þar sem lengd hvers svæðis helmingast í hverri umferð er hægt að komast af með um það bil $\log_2(n)$ umferðir. Í hverri umferð þarf að fara gegnum allt fylkið og framkvæma nokkrar samanburðaraðgerðir og nokkrar gildisgjafir. Ef þessar aðgerðir taka samtals k_q sekúndur þá er flækjutími algrímsins um það bil $k_q \cdot n \cdot \log_2(n)$ sekúndur þar sem n er fjöldi talna í fylkinu sem þarf að raða.

Samanburður á flækjutíma

Flækjutími fyrri röðunaraðferðarinnar er $\frac{1}{2}(k_s + \frac{1}{2}k_g)n^2 + 4k_g n$. Ef n er há tala munar lítið um seinni liðinn og þetta nálgast að vera $\frac{1}{2}(k_s + \frac{1}{2}k_g)n^2$. Fastinn k_q í flækjutíma quicksort og $\frac{1}{2}(k_s + \frac{1}{2}k_g)$ eru háðir gerð vélar og smáatriðum í útfærslu forritsins en

yfirleitt má gera ráð fyrir að munurinn á ½(k_s + ½k_g) og k_q sé lítill svo við getum borið saman flækjutíma þessara tveggja aðferða með því að bera saman n^2 og $n \cdot \log_2(n)$ eins og gert er í eftirfarandi töflu.

n	n^2	$n \cdot \log_2(n)$
$10 = 10^1$	10^2	um $3 \cdot 10^1$
$1.000 = 10^3$	10^6	um $1 \cdot 10^4$
$100.000 = 10^5$	10^{10}	um $2 \cdot 10^6$
$10.000.000 = 10^7$	10^{17}	um $2 \cdot 10^8$

Það er aðeins um þrefaldur munur á flækjutíma þegar fylkið inniheldur 10 stök en fallið n^2 vex hraðar en $n \cdot \log_2(n)$ svo þegar stökin eru orðin 100.000 er tímamunurinn 5.000 faldur og þegar stökin eru 10.000.000 talsins þá er munurinn orðin 500.000.000 faldur. Af þessu má vera ljóst að ólík algrím til að vinna sama verk geta verið misfljótvirk og tímamunurinn getur orðið býsna mikill.

Það hversu tölva er lengi að vinna verk veltur bæði á því hve hraðvirkur vélabúnaðurinn í henni er og á flækjutíma algrímsins sem hún vinnur eftir. Hraðamunur á dýrri tölvu og ódýrri getur verið meira en þúsundfaldur en stundum er sá munur lítill í samanburði við muninn á flækjutíma ólíkra aðferða.

Þessi samanburður á tveim röðunaraðferðum hefur verið *quicksort* mjög í vil. Rétt er þó að geta þess að flækjurými *quicksort* er mun meira en hinnar aðferðarinnar. *Quicksort* er oftast forritað með því að nota endurkomu svo forrit sem beitir þessari aðferð á stórt fylki þarf verulega mikið minnisrými á stafla til að geyma staðværar breytur.

Verkefni 7.2.a

Taktu öll spil í einum lit úr spilastokk (t.d. öll hjörtun) og raðaðu þeim fyrst með einföldu aðferðinni (sem kynnt var hér að framan og byggist á því að finna minnsta spil eða lægstu tölu í þeim hluta bunkans sem ekki er búið að raða) og svo með quicksort.

Verkefni 7.2.b

Finndu eina röðunaraðferð til viðbótar við þær tvær sem hér hefur verið rætt um.

Margliður, veldisvísisföll og raunhæfar lausnir

Við höfum nú skoðað flækjutíma þriggja algríma. Aðferðin til að skrifa jákvæða heiltölu, n, á formi tvíundakerfis hafði flækjutíma $k \cdot \log_2(n)$. Röðunaraðferðirnar höfðu flækjutíma nokkurn veginn $k \cdot n \cdot \log_2(n)$ og $k \cdot n^2$. Til er mikill fjöldi algríma þar sem flækjutíminn er annaðhvort lografall eða margliðufall af umfangi gagna (eða stærð verkefnis).

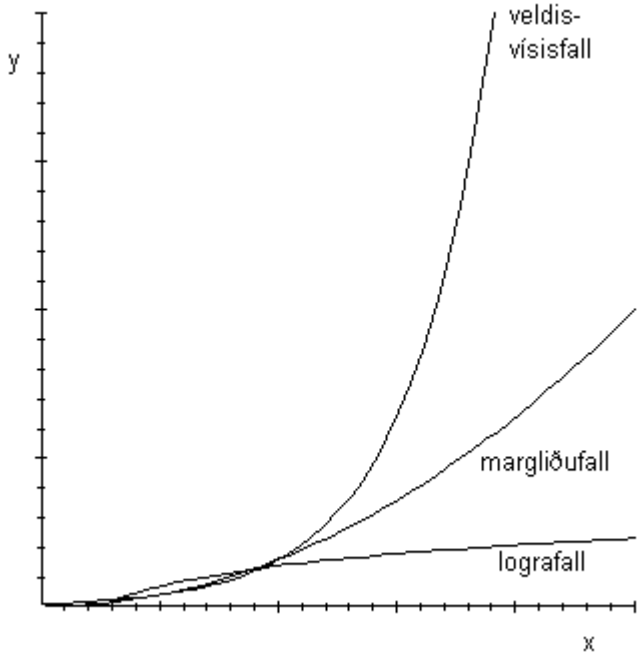
Lograföll vaxa afar hægt. Ef flækjutími er í hlutfalli við log(n) þá lýkur keyrslu algríms yfirleitt á skömmum tíma jafnvel þótt því sé beitt á umfangsmikil gögn.

Margliðuföll með háum veldisvísum eins og t.d. x^{100} vaxa að vísu ansi hratt en oftast er þó raunhæft að nota algrím þar sem flækjutími er margliðufall af umfangi gagna.

Sum föll vaxa miklu hraðar en nokkurt margliðufall. Dæmi um slík föll eru veldisvísisföll eins og 2^x eða 10^x. Að vísu tekur margliðufall með háa veldisvísa eins og x^{100} hærri gildi en veldisvísisfall af lágri tölu eins og 2^x meðan x er lág tala en veldisvísisfallið stingur margliðuna samt af þegar x stækkar.

Fjölmörg algrím hafa flækjutíma sem er veldisvísisfall af stærð eða umfangi gagna. Slík algrím er yfirleitt ekki raunhæft að nota enda þarf x ekki að vera nema 45 til að 2^x sekúndur jafngildi milljónum ára og löngu áður en x hefur náð 100 er 2^x sekúndur orðið svo langur tími að slokknað verður á sólinni áður en hann er liðinn.

Ef algrím fer 2^x sinnum gegnum slaufu og kemst milljón ferðir gegnum slaufuna á hverri sekúndu þá tekur keyrsla áratugi ef x er 50, mörg þúsund ár ef x er 60, milljónir ára ef x er 70 og mörg þúsund milljónir ára ef x er 80. Slíkt algrím er ekki nothæft á viðamikil gögn.



Fjölmörg verkefni eru þannig að öll þekkt algrím til að leysa þau hafa flækjutímafall sem vex hraðar en nokkurt margliðufall.[1] Oftast eru þetta veldisvísisföll af n en stundum líka n!, þar sem n er tala sem segir til um stærð eða umfang gagna[2]. Hér má til dæmis nefna það verkefni að þátta tölur. Þetta þýðir að þótt vandalaust sé að þátta 10 stafa tölur er óraunhæft að ætla sér að þátta 1000 stafa tölur. Annað dæmi er að búa til stundatöflur fyrir áfangaskóla þannig að göt í töflum nemenda verði eins fá og vera má, enn annað er að finna bestu leið til að lesta flutningatæki.

Hugsum okkur að við höfum fjölda vörubíla sem mega flytja 10 tonn hver og stafla af misþungum kössum sem sumir eru 1 tonn, sumir 3 tonn, sumir 5 tonn o.s.fr. Hvernig á að raða kössunum á bílana þannig að hver þeirra beri sem næst 10 tonn? Allar þekktar aðferðir til að leysa þetta vandamál hafa flækjutíma sem vex hraðar en nokkurt margliðufall þegar kössunum fjölgar.

Þótt ekki séu til raunhæfar lausnir á vandamálum eins og stundatöflugerð og lestun vörubíla tekst mönnum samt einhvern veginn að lesta flutningatæki og búa til stunda-töflur og það eru meira að segja til forrit sem reikna þokkalegar lausnir á vandamálum af þessu tagi því þótt ekki sé til raunhæf aðferð til að finna bestu lausnina eru til raun-hæfar aðferðir sem finna viðunandi lausn í flestum tilvikum.

Þegar öll algrím til að leysa verk eru ónothæf vegna mikils flækjutíma er oft hægt að bjarga sér með því að skilgreina léttara verk sem gerir nokkurn veginn sama gagn og hægt er að vinna eftir raunhæfri aðferð. Þótt ekki sé raunhæft að reikna bestu mögulega

[1] Um þetta er stundum notað orðalagið að föll hafi ólíka stærðargráðu. Þegar sagt er að f(x) sé af hærri stærðargráðu en g(x) er átt við að $g(x)/f(x) \to 0$ þegar $x \to \infty$, þ.e. það sé sama hve lága tölu, ε, við hugsum okkur alltaf sé hægt að finna tölu n þannig að fyrir öll x gildi að ef $x > n$ þá sé $g(x)/f(x) < \varepsilon$.

[2] n! er $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot \ldots \cdot (n-1) \cdot n$, þ.e. allar tölur frá 1 upp í n margfaldaðar saman. Þetta fall vex hraðar en veldisvísisföll.

stundatöflu fyrir alla nemendur í 600 manna fjölbrautaskóla kann að vera raunhæft að láta tölvu reikna stundatöflur þannig að í flestum tilvikum verði lítið af ónauðsynlegum eyðum.

Útvíkkun á tilgátu Church og Turings

Hér hefur verið talað um flækjutíma sem eiginleika algríms fremur en verkefnis, enda er hægt að vinna flest verk á marga vegu. Það eru til dæmis til mörg röðunaralgrím sem hægt er að orða á ólíka vegu á ólíkum forritunarmálum. Ekki er útilokað að ólík tilbrigði sömu aðferðar hafi ólíkan flækjutíma og það er heldur ekki útilokað að útkoman úr því að þýða algrím af forritunarmáli á vélamál einnar tölvu verði ólík útkomunni úr því að þýða það á vélamál annarrar tölvu þannig að vélamálsþýðingarnar tvær hafi ólíkan flækjutíma. Af þessum ástæðum er erfitt að eigna verkefnum ákveðin flækjutíma án nokkurra fyrirvara.

Við getum þó sagt að miðað við tiltekna vél sé flækjutími verks margliðufall af umfangi þess ef keyrslutími fljótvirkasta algríms sem vélin getur keyrt til að vinna verkið er margliðufall af umfangi þess. En ætli það séu til verkefni þar sem flækjutíminn er margliðufall á einni tölvu en ekki á annarri?

Fyrr í þessum kafla var minnst á *tilgátu Church og Turings*. Af henni leiðir að allar tölvur eru jafngildar, þær geta allar unnið sömu verk, nefnilega alla útreikninga sem hægt er að vinna eftir algrími. Af þessari tilgátu leiðir líka að hægt er að láta hvaða tölvu sem er herma eftir hvaða annarri tölvu sem er því fyrir hverja tölvu er til algrím til að herma eftir henni sem hver önnur tölva getur unnið eftir.

Sé algrím til að herma eftir vinnu einnar tölvu keyrt á annarri tölvu þá er keyrslutíminn ævinlega margliðufall af umfangi vinnunnar. Þetta gildir um allar tölvur sem gerðar hafa verið. Margir álíta sennilegt að þetta gildi ekki bara um allar tölvur sem smíðaðar hafa verið heldur um allar tölvur sem mögulegt er að smíða. Ef þetta er rétt þá getum við bætt við *tilgátu Church og Turings* og sagt að ekki sé nóg með að allar mögulegar tölvur geti unnið sömu verk heldur gildi líka að *ef flækjutími verks er margliðufall á einni tölvu þá sé hann margliðufall á öllum tölvum og ef hann er veldisvísisfall á einni tölvu þá sé hann veldisvísisfall á öllum tölvum*. Af þessari tilgátu leiðir að flest verk sem óraunhæft er að vinna á tölvum sem til eru nú verði einnig óraunhæft að vinna á þeim tölvum sem til verða í framtíðinni.

7.3 Reiknanleiki og óleysanleg vandamál

Áætlun Hilberts og sönnun Gödels

Hér hefur verið minnst á algrím sem ekki er raunhæft að nota því tíminn sem það tekur er veldisvísisfall af umfangi gagnanna sem þeim er beitt á. En ætli það séu til einhver verkefni sem ekki er hægt að leysa með nokkru algrími? Já, slík verkefni eru til og þar er ekki bara um að ræða illa skilgreind verkefni eins og að fá fólk til að hlæja eða yrkja ódauðleg ljóð (sem trúlega er ekki til algrím fyrir). Það eru til óendanlega mörg stærðfræðileg verkefni sem ekki er hægt að leysa með neinni aðferð.

Áður en fjallað verður meira um óleysanleg verkefni er rétt að rifja upp sögulegan aðdraganda þess að menn uppgötvuðu tilveru slíkra verkefna.

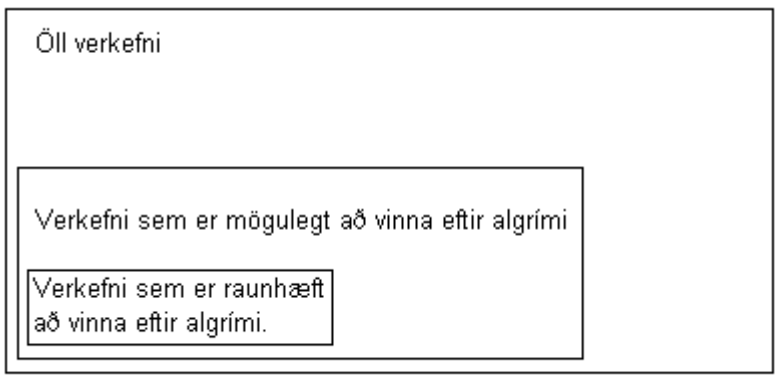
Á fyrstu áratugum 20. aldar var Þjóðverjinn *David Hilbert* (1862 – 1943) frægastur allra stærðfræðinga. Hann hafði ekki aðeins áhuga á að rannsaka mengi, tölur, föll og ferla og önnur efni sem stærðfræðingar höfðu fengist við heldur velti hann líka fyrir sér eðli stærðfræðinnar sjálfrar. Hilbert taldi að hægt væri að orða öll stærðfræðileg sannindi á formlegu táknmáli (þ.e. máli með endanlegum orðaforða og nákvæmum reglum um hvernig orð og tákn mynda setningar) og setja fram strangar rökfræðireglur um hvernig leiða má eina setningu af öðrum. Hilbert áleit ennfremur að á þessu máli væri hægt að orða frumsetningar allrar stærðfræði og binda hana þannig í kerfi sem væri í senn *sjálfu sér samkvæmt*, *fullkomið*, og *ákvarðanlegt*.

- Að kerfið sé *sjálfu sér samkvæmt* merkir að af frumsetningunum sé ekki hægt að leiða mótsagnir, þ.e. að það geti aldrei gerst að tvær setningar sem leiddar eru af frumsetningunum stangist á þannig að önnur neiti hinni. Þar sem hægt er að leiða hvað sem er af mótsögn er stærðfræðilegt kerfi sem ekki er sjálfu sér samkvæmt algerlega ónýtt.
- Að kerfið sé *fullkomið* merkir að hægt sé að leiða allar sannar stærðfræðisetningar af frumsetningunum, þ.e. að það verði engin stærðfræðileg sannindi útundan.
- Að kerfið sé *ákvarðanlegt* þýðir að til sé algrím til að komast að því hvort setning sé afleiðing af frumsetningunum.

Þessar hugmyndir Hilberts voru í dúr við skoðanir flestra stærðfræðinga. Þeir litu á fræði sín sem formlegt kerfi þar sem öll sannindi væru leidd með ströngum rökfærslum af fáeinum grundvallarreglum eða frumsetningum.

Um svipað leyti og Hilbert setti hugmyndir sínar fram jókst mjög áhugi á rökfræði, formlegum málum og frumsetningakerfum og gerðar voru nokkrar tilraunir til að binda stærðfræðina í kerfi eins og Hilbert lagði til. Frægust þessara tilrauna er mikið rit eftir Englendingana *Bertrand Russell* (1872 – 1970) og *Alfred North Whitehead* (1861 – 1947) sem heitir *Principia Mathematica* og kom út á árunum 1910 til 1913. Rannsóknir Russells, Whiteheads og fleiri á formlegum málum urðu síðar ein af undirstöðum tölvufræðinnar og rannsókna á forritunarmálum.

Eftir að *Principia Mathematica* kom út urðu geysilegar framfarir í rökfræði og þekkingu manna á eðli formlegra mála og stærðfræðilegra kerfa. Flestir bjuggust við að draumur Hilberts um að öll stærðfræði yrði bundin í formlegt kerfi mundi senn rætast, þar til árið 1931 þegar austurríski stærðfræðingurinn *Kurt Gödel* (1906 – 1978) sannaði að það sé ómögulegt að kerfisbinda talnafræði (þá grein stærðfræðinnar sem fjallar um heilar tölur) með þeim hætti sem Hilbert hugsaði sér.



Með *sönnun Gödels* var sýnt fram á að ómögulegt sé að búa til frumsetningakerfi fyrir talnafræði sem sé *í senn fullkomið og sjálfu sér samkvæmt*. Þar sem nauðsynlegt er að gera þá kröfu að slík kerfi séu sjálfum

sér samkvæm þýðir þetta í raun að útilokað sé að setja fram frumsetningar sem dugi til að leiða út allar sannar stærðfræðilegar setningar um heilar tölur, alltaf verði einhver sannindi út undan.

Ef draumar Hilberts hefðu ræst þá væri stærðfræðilegum rannsóknum að vissu leyti lokið því það þyrfti ekki lengur hugmyndaflug og sköpunargáfu til að komast að því hvort stærðfræðileg fullyrðing sé sönn eða ósönn, það væri hægt að láta tölvu framkvæma algrímið sem sker úr um hvort setning er afleiðing af frumsetningunum eða ekki. Sönnun Gödels útilokar að vísu ekki að slíkt algrím sé til. Hún útilokar aðeins að stærðfræðilegt kerfi gæti verið *fullkomið* en eftir að hún var komin fram var það enn opin spurning hvort það gæti verið *ákvarðanlegt*. Kerfi sem er *ákvarðanlegt* en *ekki fullkomið* er þannig að til er algrím til að skera úr um hvort setning er afleiðing af frumsetningunum en þar sem sumar sannar setningar eru ekki afleiðing af frumsetningunum dugar ákvörðunaraðferðin ekki til að skera úr um það í öllum tilvikum hvort setning sé sönn, aðeins hvort hún sé sannanleg.

Turing og „das Entscheidungsproblem“

Það verkefni að finna ákvörðunaraðferð fyrir stærðfræðilegt kerfi kallaði Hilbert *das Entscheidungsproblem*. Eins og fyrr hefur verið getið birti Alan Turing skilgreiningu á *algrími* árið 1936. Í ritgerð eftir hann sem kom út það ár lýsti Turing ímyndaðri vél, eins konar tölvu, og sagði að algrím sé aðferð sem slík vél gæti unnið eftir. Það sem vakti fyrir Turing þegar hann setti þessar hugmyndir fram var að glíma við *das Entscheidungsproblem*. En í leiðinni mótaði hann margar af undirstöðuhugmyndum tölvufræðinnar.

Niðurstaða Turings var að ekki væri mögulegt að búa til ákvörðunaraðferð af því tagi sem Hilbert hafði talað um. Með rökum hans fyrir þessari niðurstöðu varð til ný grein innan stærðfræðinnar sem fjallar um reiknanleika, þ.e.a.s. um hvaða verkefni er hægt að leysa með algrími og hvaða verkefni er ekki til algrím fyrir.

Uppgötvanir Turings og Gödels sem hér hefur verið sagt frá eru með merkustu vísindaafrekum 20. aldar. Með þeim varð að engu draumurinn um að setja alla stærðfræði fram sem *sjálfu sér samkvæmt*, *fullkomið* og *ákvarðanlegt* kerfi. En um leið kviknaði nýr draumur um altæka vél sem gæti unnið eftir hvaða aðferð sem er. Þótt áætlanir Hilberts hafi verið vonlausar og tilgátur hans um eðli stærðfræðinnar rangar urðu þær kveikjan að merkilegum pælingum sem af spruttu heilar vísindagreinar sem síðan hafa vaxið og dafnað. Í sögu vísindanna finnast mörg dæmi um heimskuleg sannindi sem bera engan ávöxt. En þar eru líka dæmi sem þetta um viturleg ósannindi sem leiða til merkilegra pælinga.

Óleysanleg verkefni

Árið 1936 komst Turing að því að ekki sé til algrím sem sker úr um hvort setning í talnafræði sé afleiðing af gefnum frumsetningum eða ekki. Síðan hafa mörg verkefni verið dæmd óvinnandi með svipuðum rökum. Meðal annars hefur verið sýnt fram á að ekki sé hægt að búa til algrím sem sker úr um hvort forrit lýkur keyrslu eða hvort það heldur áfram endalaust. Það hefur líka verið sýnt fram á að ekki sé til algrím sem sker úr um það hvort tvö forrit eru jafngild (þ.e. skila alltaf sömu útkomu ef þau fá sömu gögn). Svona mætti lengi telja en við skulum staldra aðeins við spurninguna um hvort forrit stöðvast eða heldur áfram endalaust.

Í sumum tilvikum er auðvelt að sýna fram á að forrit lýkur keyrslu og í sumum tilvikum er augljóst að það inniheldur endalausa slaufu og keyrslu þess getur aldrei lokið. Hér er dæmi um endalausa slaufu:

```
heiltala x = 1;
endurtaktu meðan (x < 2)
{
   x = -1 * x;
}
```

Ekkert sem gerist innan í slaufunni getur valdið því að x hætti að vera minna en 2 svo keyrslu hennar lýkur aldrei.

Stundum er erfitt að átta sig á hvort slaufa er endalaus eða hvort hún endar einhvern tíma. Hér er dæmi um undirforrit sem heitir *endalaust?* og inniheldur slíka slaufu.

```
undirforrit endalaust?(heiltala x)
{
   meðan (x > 1)
   {
      ef (x er slétt tala)
      {
         x = x/2;
      }
      annars
      {
         x = 3*x + 1;
      }
   }
}
```

Í sumum tilvikum er augljóst að þetta undirforrit lýkur keyrslu. Slaufan í því er til dæmis bara endurtekin tvisvar ef skipað er `endalaust?(4)` en það er allt annað en auðvelt að átta sig á hvað gerist ef skipað er `endalaust?(17311037)`.

Þegar sagt er að ekki sé til algrím til að skera úr um hvort forrit lýkur keyrslu er ekki verið að útiloka að hægt sé að komast að því í einstökum tilvikum. Ef forrit lýkur einhvern tíma keyrslu er alltaf hægt að komast að því að svo sé með því að setja það af stað og bíða. Ef keyrslunni lýkur einhvern tíma þá tekur biðin enda. Það verður á endanum fullreynt að keyrslu ljúki en það verður aldrei fullreynt að henni ljúki ekki. Ef forrit keyrir endalaust þá dugar ekki að bíða í trilljón ár til að leiða það í ljós.

Það er útilokað að búa til algrím sem sker úr um það í öllum tilvikum hvort forrit endar eða ekki. Þessa niðurstöðu er hægt að rökstyðja með fremur einföldum hætti með því að leiða mótsögn af þeirri tilgátu að slíkt algrím sé til.

Gerum ráð fyrir að til sé algrím sem tekur við forriti og inntaksgögnum handa forritinu og sker úr um hvort forritið stoppar eða ekki. Ef slíkt algrím er til þá er hægt að búa til undirforrit sem tekur við tveim gildum (forriti og inntaksgögnum) og skrifar „stoppar“ ef forritið sem því er sent stoppar en „stoppar ekki“ ef það keyrir endalaust (er t.d. með endalausa slaufu). Þar sem forrit og gögn eru bara runur af stöfum eða táknum sem hægt er að skrifa sem tölustafi í tvíundakerfi getum við skrifað hvaða forrit sem er sem runu af tölustöfum og sömuleiðis er hægt að tákna hvaða gögn sem er með tölustöfum. Við getum því hugsað okkur að undirforritið taki við forriti, F, og gögnum, G, sem heilum tölum svona:

```
undirforrit stoppar?(heiltala F, heiltala G)
{
   ef (F stoppar þegar það fær gögnin G)
   {
      skrifa("stoppar");
   }
   annars
   {
      skrifa("stoppar ekki");
   }
}
```

Ef mögulegt er að búa til algrím sem sker úr um það í öllum tilvikum hvort forrit á einhverju forritunarmáli lýkur keyrslu eða ekki þá hlýtur að vera hægt að búa til undirforrit sem vinnur sama verk og *stoppar?*. Og ef það er hægt þá er hægt að búa til undirforrit eins og þetta:

```
undirforrit mótsögn(heiltala X)
{
   ef (stoppar?(X, X) skrifar "stoppar")
   {
      fara í endalausa slaufu;
   }
   annars
   {
      stoppa;
   }
}
```

Hugsum okkur nú að við keyrum undirforritið *mótsögn* og sendum því sjálft sig svona `mótsögn(mótsögn)`. Undirforritið segir að ef *mótsögn* stoppar þegar það fær sjálft sig sem gögn þá eigi að fara í endalausa slaufu en ef það stoppar ekki þá eigi að stoppa svo ef það stoppar þá heldur það endalaust áfram og ef það heldur endalaust áfram þá stoppar það. Þetta er mótsögn svo forsendan, að hægt sé að búa til undirforritið *stoppar?*, hlýtur að vera röng.

Sönnun á setningu Gödels

Upphafleg sönnun Gödels á því að ekki sé hægt að búa til *samkvæmt* og *fullkomið* frumsetningakerfi fyrir talnafræði er flókin. En það er hægt að leiða sömu niðurstöðu með mun einfaldari hætti af þeirri forsendu að ekki sé til algrím til að skera úr um það í öllum tilvikum hvort forrit lýkur keyrslu.

Þar sem hægt er að skrifa forrit og gögn sem heilar tölur er hægt að skilgreina fall, H, sem tekur við tveim heilum tölum, F og G og skilar útkomunni 1 ef forritið sem táknað er með F lýkur keyrslu þegar það fær gögnin sem táknuð eru með G en útkomunni 0 ef það lýkur ekki keyrslu.

H(F, G) = 1 ef forritið sem táknað er með F lýkur keyrslu þegar það fær gögnin sem táknuð eru með G.

H(F, G) = 0 ef forritið sem táknað er með F lýkur aldrei keyrslu þegar það fær gögnin sem táknuð eru með G.

Einhverjum kann að þykja þetta undarlegt fall en ef tölvukerfi er lýst af fullkominni nákvæmni þá er hægt að umrita lýsinguna á stærðfræðilegt táknmál og skilgreina fall á borð við H.

Sannanir eru ekkert annað en strengir af táknum á einhverju táknmáli. Það er hægt að raða öllum slíkum strengjum í röð þannig að fyrst komi í starfrófsröð allir þeir sem innihalda aðeins eitt tákn, næst allir sem innihalda tvö tákn o.s.fr. Flestir strengirnir í þessari röð eru auðvitað merkingarlaust bull en innan um eru sannanir. Ef hægt væri að sanna allar sannar setningar um heilar tölur þá væri einhvers staðar í þessari röð sönnun á setningunni *H(F, G) = 0* ef forritið sem táknað er með F stöðvast aldrei þegar því er beitt á gögn sem táknuð eru með G.

Við getum nú sýnt fram á að ef hægt væri að sanna allar sannar setningar um heilar tölur þá væri hægt að búa til aðferð til að skera úr um hvort forrit stöðvast eða heldur áfram endalaust. Aðferðin gæti til dæmis verið á þessa leið: Til að skera úr um hvort forrit sem táknað er með F stöðvast þegar það fær gögn sem táknuð eru með G skal setja það í gang á tölvu númer 1 og láta hana skrifa „stoppar“ um leið og keyrslu forritsins lýkur. Ef keyrslunni lýkur þá kemur að því að tölva númer 1 gefur það til kynna.

Jafnframt því sem forritið er sett af stað á tölvu númer 1 skal setja tölvu númer 2 í gang og láta hana fara gegnum lista yfir alla strengi af táknum sem notuð eru til að skrifa stærðfræðilegar sannanir. Ef hún kemur að sönnun á setningunni H(F, G) = 0 þá á hún að skrifa „stoppar ekki“. Ef keyrslunni lýkur aldrei þá kemur að því að tölva númer 2 gefur það til kynna.

Ef setning Gödels væri ósönn þá væri sem sagt hægt að búa til aðferð til að skera úr um það í öllum tilvikum hvort keyrslu forrits lýkur eða hvort hún heldur áfram endalaust. Þar sem sannað hefur verið að slík aðferð geti ekki verið til þá hlýtur setning Gödels að vera sönn.

Til upprifjunar

1. Hvað eru:
   Algrím, áætlun Hilberts, das Entscheidungsproblem, flækjurými, flækjutími, sönnun Gödels, tilgáta Church og Turings, Turingvél.

2. Nefndu dæmi um:
   Verk sem hefur of mikinn flækjutíma til að raunhæft sé að tölva geti lokið því;
   Verkefni sem er ekki hægt að leysa eftir algrími.

3. Hvað er átt við þegar sagt er að:
   Kerfi sé fullkomið, sjálfu sér samkvæmt og ákvarðanlegt?
   Aðferð sé endanleg, örugg og ótvíræð?

8. kafli

Forritun og forritunarmál

8.1 Málskipan og merkingarfræði

Forritunarmál og mannamál

Forritunarmál eru notuð eru til að orða algrím. Þau eru að því leyti eins og tungumál að hægt er að nota endanlegan orðaforða til að mynda óendanlega margar mismunandi setningar og setningarunur. En *málskipan* þeirra og *merkingarfræði* eru miklu einfaldari en í tungumálum eins og íslensku eða ensku.

- *Málskipan* (á ensku *syntax*) forritunarmáls er reglur um hvernig raða má orðum og öðrum táknum saman í skipanir og heil forrit. Þessar reglur skera úr um hvort runa af orðum og táknum er rétt mynduð skipun (eða rétt myndað forrit eða forritshluti) eða ekki.
- *Merkingarfræði* (á ensku *semantics*) forritunarmáls er reglur um hvaða merkingu rétt myndaðar skipanir hafa.

Til að forritunarmál sé nothæft þarf í fyrsta lagi að vera til nothæft algrím sem sker úr um hvort runa af táknum er rétt mynduð eða ekki og í öðru lagi þarf að vera til nothæft algrím sem þýðir texta af forritunarmálinu á vélamál einhverrar tölvu. Fyrrnefnda algrímið byggir á reglum um *málskipan* forritunarmálsins og það seinna byggir á reglum um *merkingarfræði* þess. Til að þessi algrím séu nothæf þarf flækjutími þeirra að vera innan hóflegra marka.

Það er álitamál hvort til eru algrím sem skera úr um hvort setningar á tungumáli (eins og t.d. íslensku eða ensku) eru rétt myndaðar. Ef slík algrím eru til hljóta þau að vera afar flókin. Hins vegar er ljóst að ekki eru til reglur um merkingarfræði tungumála sem duga til að ákvarða merkingu setninga. Þetta er vegna þess að það hvernig við skiljum setningar á tungumálum ræðst að nokkru leyti af aðstæðum og vitneskju sem eru utan við svið merkingarfræðinnar.

Sem dæmi um hvernig merking ræðst af aðstæðum getum við tekið setninguna „Það er hundur í honum“. Ef mælandi horfir á mann þá getur hún merkt að hann sé í vondu skapi. En sé bent á kofa þá getur hún merkt að þar inni sé dýr af ákveðinni tegund.

Ef við hugsum um setningar eins og „Sigga gat ekki farið í búðina því hún var lasin“ og „Sigga gat ekki farið í búðina því hún var lokuð“ þá áttum við okkur strax á að fornafnið „hún“ vísar á Siggu í fyrri setningunni en á búðina í þeirri seinni. Við gerum okkur grein fyrir þessu vegna þess að við vitum að konum er hættara við lasleika en búðum.

Til að skilja þessar (og aðrar) setningar á venjulegum tungumálum dugar ekki að fletta upp í málfræðireglum og orðabókum. Til að skilja mannamál þarf að styðjast við heil ókjör af vitneskju til viðbótar við reglur um *málskipan* og *merkingarfræði*. Þegar forritunarmál eiga í hlut duga málfræðireglur hins vegar fullkomlega til að skera úr um

hvort runa af táknum er rétt mynduð setning (eða forrit eða forritshluti) og þegar það hefur verið gert er hægt að komast að merkingu hennar án þess að styðjast við neitt annað en orðabækur og formlegar reglur sem yfirleitt eru ekki fyrirferðarmeiri en svo að hægt er að koma þeim fyrir í einni lítilli bók.

Backus - Naur ritháttur

Ýmsar leiðir eru til að lýsa málskipan forritunarmála. Ein sú algengasta er *Backus-Naur ritháttur*. Táknin sem eru notuð til að gera grein fyrir málskipan með þessum hætti eru $\rightarrow$, | og oddbogar. (Stundum er ::= eða => notað í staðinn fyrir $\rightarrow$.)

Við skulum taka sem dæmi skilgreiningu á gildisgjöf í máli eins og *Java* eða *C*. Hún gæti verið svona:

⟨gildisgjöf⟩ → ⟨nafn⟩ **=** ⟨segð⟩

Þetta þýðir að gildisgjöf sé skilgreind þannig að fyrst komi nafn, svo komi merkið = og svo komi segð. Merkið = er ekki innan oddboga því það er *lokatákn*, þ.e.a.s. tákn sem ekki þarf að skilgreina nánar. Orðið *nafn* er hins vegar innan oddboga því það þarf að skilgreina nánar hvað nafn er.

Ef við gerum ráð fyrir að nafn sé myndað úr einum eða fleiri litlum bókstöfum úr enska stafrófinu og engu öðru þá getum við skilgreint nafn svona:

⟨nafn⟩ → ⟨bókstafur⟩ | ⟨bókstafur⟩⟨nafn⟩

Táknið | þýðir *eða* svo þessi skilgreining segir að nafn sé annað hvort einn bókstafur eða einn bókstafur og nafn. Til að þessi skilgreining sé nothæf þarf að skilgreina bókstaf. Ef við höldum okkur við litlu stafina í enska stafrófinu getum við gert það svona:

⟨bókstafur⟩ → **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**

Samkvæmt þessu er *x* nafn því það er einn bókstafur. *ex* er líka nafn því það er bókstafur (nefnilega *e*) með nafni fyrir aftan (nefnilega *x*). Með sömu rökum er *kex* nafn því það er bókstafur (nefnilega *k*) með nafni (nefnilega *ex*) fyrir aftan.

Hér er bókstafur skilgreindur með tómum *lokatáknum*. Skilgreiningar með *Backus-Naur rithætti* geta innihaldið atriði sem þarf að skilgreina nánar. Skilgreiningar á þessum atriðum geta svo innihaldið enn önnur atriði sem þarfnast skilgreiningar en þetta getur ekki haldið endalaust áfram. Við endum alltaf á *lokatáknum* sem eru hluti af orðaforða eða stafrófi málsins sem fjallað er um.

Skilgreiningar á málskipan heils forritunarmáls eins og *Java* eða *C* fylla nokkrar blaðsíður. Hér mun því látið duga að skilgreina ofurlítinn part af máli sem svipar til *Java* og *C*. Þessi partur inniheldur:

- nöfn úr litlum stöfum úr enska stafrófinu;
- gildisgjöf (með merkinu =);
- jákvæðar heilar tölur;
- segðir sem eru myndaðar úr heilum tölum, nöfnum, reikniaðgerðunum +, - og * og svigum;

Með þessu getum við myndað skipanir eins og

```
breidd = 7
lengd = 5*(y-3)
magn = 3 + 5 * (y - 1)
```

Við vorum búin að skilgreina gildisgjöf, nafn og bókstaf svona.

1. ⟨gildisgjöf⟩ → ⟨nafn⟩ **=** ⟨segð⟩
2. ⟨nafn⟩ → ⟨bókstafur⟩ | ⟨bókstafur⟩⟨nafn⟩
3. ⟨bókstafur⟩ → **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**

Þá á eftir að skilgreina segð og tölu. Það er gert hér fyrir neðan.

4. ⟨segð⟩ → ⟨liður⟩ | ⟨liður⟩ **+** ⟨segð⟩ | ⟨liður⟩ **-** ⟨segð⟩
5. ⟨liður⟩ → ⟨þáttur⟩ | ⟨þáttur⟩ ***** ⟨liður⟩
6. ⟨þáttur⟩ → ⟨nafn⟩ | ⟨tala⟩ | **(** ⟨segð⟩ **)**
7. ⟨tala⟩ → ⟨tölustafur⟩ | ⟨tölustafur⟩⟨tala⟩
8. ⟨tölustafur⟩ → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

Athugum nú hvernig hægt er að leiða út setninguna

```
magn = 3 + 5 * (y - 1)
```

Setningin er gildisgjöf á forminu

⟨nafn⟩ **=** ⟨segð⟩

magn er nafn samkvæmt reglum númer 2 og 3. Þetta er hægt að leiða út skref fyrir skref svona.

⟨nafn⟩	
⟨bókstafur⟩⟨nafn⟩	Skv. 2. reglu má setja ⟨bókstafur⟩⟨nafn⟩ í stað ⟨nafn⟩.
m⟨nafn⟩	Skv. 3. reglu má setja m í stað ⟨bókstafur⟩.
m⟨bókstafur⟩⟨nafn⟩	Skv. 2. reglu má setja ⟨bókstafur⟩⟨nafn⟩ í stað ⟨nafn⟩.
ma⟨nafn⟩	Skv. 3. reglu má setja a í stað ⟨bókstafur⟩.
ma⟨bókstafur⟩⟨nafn⟩	Skv. 2. reglu má setja ⟨bókstafur⟩⟨nafn⟩ í stað ⟨nafn⟩.
mag⟨nafn⟩	Skv. 3. reglu má setja g í stað ⟨bókstafur⟩.
mag⟨bókstafur⟩	Skv. 2. reglu má setja ⟨bókstafur⟩ í stað ⟨nafn⟩.
magn	Skv. 3. reglu má setja n í stað ⟨bókstafur⟩.

3 + 5 * (y - 1) er segð samkvæmt reglunum. Þetta er hægt að leiða út skref fyrir skref svona:

⟨segð⟩	
⟨liður⟩ **+** ⟨segð⟩	4. regla: ⟨segð⟩ → ⟨liður⟩ + ⟨segð⟩.[1]
⟨þáttur⟩ **+** ⟨segð⟩	5. regla: ⟨liður⟩ → ⟨þáttur⟩.
⟨tala⟩ **+** ⟨segð⟩	6. regla: ⟨þáttur⟩ → ⟨tala⟩.
⟨tölustafur⟩ **+** ⟨segð⟩	7. regla: ⟨tala⟩ → ⟨tölustafur⟩.
3 **+** ⟨segð⟩	8. regla: ⟨tölustafur⟩ → 3.
3 **+** ⟨liður⟩	4. regla: ⟨segð⟩ → ⟨liður⟩.
3 **+** ⟨þáttur⟩ ***** ⟨liður⟩	5. regla: ⟨liður⟩ → ⟨þáttur⟩ * ⟨liður⟩.
3 **+** ⟨tala⟩ ***** ⟨liður⟩	6. regla: ⟨þáttur⟩ → ⟨tala⟩.
3 **+** ⟨tölustafur⟩ ***** ⟨liður⟩	7. regla: ⟨tala⟩ → ⟨tölustafur⟩.
3 **+** 5 ***** ⟨liður⟩	8. regla: ⟨tölustafur⟩ → 5.
3 **+** 5 ***** ⟨þáttur⟩	5. regla: ⟨liður⟩ → ⟨þáttur⟩.
3 **+** 5 ***** (⟨segð⟩)	6. regla: ⟨þáttur⟩ → (⟨segð⟩).
3 **+** 5 ***** (⟨liður⟩ **-** ⟨segð⟩)	4. regla: ⟨segð⟩ → ⟨liður⟩ - ⟨segð⟩.
3 **+** 5 ***** (⟨þáttur⟩ **-** ⟨segð⟩)	5. regla: ⟨liður⟩ → ⟨þáttur⟩ .
3 **+** 5 ***** (⟨nafn⟩ **-** ⟨segð⟩)	6. regla: ⟨þáttur⟩ → ⟨nafn⟩.

[1] Þetta þýðir að 4. regla leyfi að í stað ⟨segð⟩ sé sett ⟨liður⟩ + ⟨segð⟩.

3 **+** 5 ***** (⟨bókstafur⟩ **-** ⟨segð⟩) 2. regla: ⟨nafn⟩ → ⟨bókstafur⟩.
3 **+** 5 ***** (y **-** ⟨segð⟩) 3. regla: ⟨bókstafur⟩ → y.
3 **+** 5 ***** (y **-** ⟨liður⟩) 4. regla: ⟨segð⟩ → ⟨liður⟩.
3 **+** 5 ***** (y **-** ⟨þáttur⟩) 5. regla: ⟨liður⟩ → ⟨þáttur⟩.
3 **+** 5 ***** (y **-** ⟨tala⟩) 6. regla: ⟨þáttur⟩ → ⟨tala⟩.
3 **+** 5 ***** (y **-** ⟨tölustafur⟩) 7. regla: ⟨tala⟩ → ⟨tölustafur⟩.
3 **+** 5 ***** (y **-** 1) 8. regla: ⟨tölustafur⟩ → 1.

Reglur 1 til 8 lýsa aðeins ofurlitlum hluta af forritunarmáli. En það er hægt að bæta við reglum eins og t.d. þessum:

9. ⟨skilyrðissetning⟩ → **if (** ⟨röksegð⟩ **)** ⟨skipanablokk⟩
10. ⟨röksegð⟩ → ⟨segð⟩⟨samanburðarvirki⟩⟨segð⟩
11. ⟨samanburðarvirki⟩ → **==** | **<** | **>**
12. ⟨skipanablokk⟩ → **{** ⟨skipanaruna⟩ **}**
13. ⟨skipanaruna⟩ → ⟨skipun⟩ **;** | ⟨skipun⟩ **;** ⟨skipanaruna⟩
14. ⟨skipun⟩ → ⟨gildisgjöf⟩ | ⟨skilyrðissetning⟩

Samkvæmt reglum 1 til 14 er hægt að mynda setningar eins og:

```
if (x > 3)
{
   y = x * 2;
}
```

eða

```
if (x + 5 > y)
{
   y = x * 2;
   z = x + 2;
}
```

Verkefni 8.1.a

Leiddu eftirtaldar setningar af myndunarreglum 1 til 14.

```
x = 5 - 3
```

```
ax = x - 2 * y
```

```
if (x > 3)
{
   y = x * 2;
}
```

Verkefni 8.1.b

Myndunarreglur 8 til 9 duga til að mynda jákvæðar heilar tölur. Settu fram myndunarreglur sem duga til að mynda bæði jákvæðar og neikvæðar kommutölur.

Verkefni 8.1.c

Myndunarreglur 2 og 3 duga til að mynda nöfn úr litlum bókstöfum úr enska stafrófinu. Settu fram myndunarreglur fyrir nöfn sem eru mynduð úr bókstöfum og tölustöfum þannig að fyrsti stafurinn sé þó alltaf bókstafur.

Viðbætur við Backus-Naur rithátt og flæðirit

Þau dæmi sem hér hafa verið skoðuð sýna að *Backus-Naur ritháttur* er nokkuð langorður. Það þarf margar reglur til að gera grein fyrir einföldustu atriðum í málskipan forritunarmála og langar útleiðslur til að leiða setningar af reglunum. Til að bæta úr þessu hafa verið settar fram viðbætur við *Backus-Naur ritháttinn* (á ensku *extended*

Backus-Naur form skst. *EBNF*). Þessar viðbætur eru einkum fólgnar í því að nota hornklofa til að tákna að atriði komi fyrir 0 eða 1 sinni og slaufusviga til að tákna að það komi 0 sinnum eða oftar. Með þessum viðbótum er hægt að skrifa myndunarreglur fyrir nafn sem byrjar á bókstaf og inniheldur bókstafi og tölustafi svona:

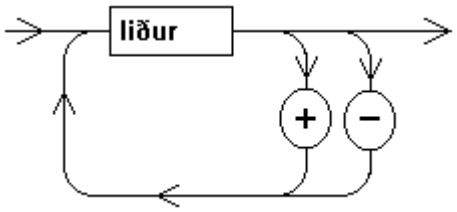
⟨nafn⟩ → ⟨bókstafur⟩ { ⟨bókstafur⟩ | ⟨tölustafur⟩ }

Þetta þýðir að fyrst komi bókstafur svo komi 0 sinnum eða oftar bókstafur eða tölustafur. Með hornklofum og slaufusvigum er líka fremur auðvelt að skrifa myndunarreglur fyrir tölur sem duga til að mynda alls konar tölur, jákvæðar og neikvæðar, heiltölur og brot.

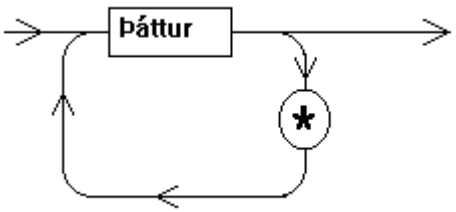
⟨tala⟩ → [⟨formerki⟩] ⟨tölustafur⟩ {⟨tölustafur⟩} [**,** ⟨tölustafur⟩{⟨tölustafur⟩}]
⟨formerki⟩ → **+** | **-**
⟨tölustafur⟩ → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

Myndunarreglan fyrir tölu segir að fyrst komi [⟨formerki⟩] þ.e. ekkert eða eitt formerki. Það má sem sagt sleppa formerkinu og það má setja eitt formerki. Næst kemur ⟨tölustafur⟩ og þar fyrir aftan {⟨tölustafur⟩} þ.e. 0 eða fleiri tölustafir. Aftan við þetta kemur [**,** ⟨tölustafur⟩{⟨tölustafur⟩}] sem merkir að setja megi einu sinni (eða aldrei) kommu með tölustöfum fyrir aftan.

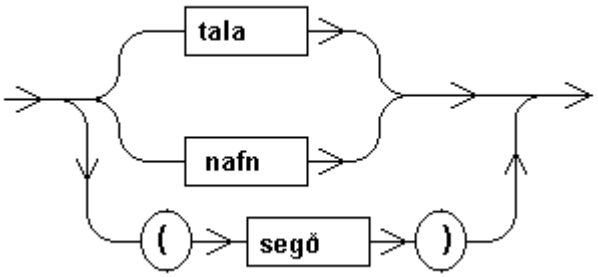
Til eru fleiri aðferðir til að gera grein fyrir málskipan forritunarmála. Ein þeirra algengustu er að teikna flæðirit eins og hér fyrir neðan. Lokatákn eru höfð innan í hring og tákn sem þarf að skilgreina nánar eru skrifuð innan í ferhyrning. Til að mynda setningu þarf að fylgja stefnu örvanna. Fyrsta myndin sýnir t.d. að hægt er að mynda segð úr einum lið eða með því að skrifa einn lið og svo annað hvort plús eða mínus og fara svo aftur gegnum alla myndina.



Flæðirit sem samsvarar myndunarreglunni
⟨segð⟩ → ⟨liður⟩ | ⟨liður⟩ **+** ⟨segð⟩ | ⟨liður⟩ **-** ⟨segð⟩



Flæðirit sem samsvarar myndunarreglunni
⟨liður⟩ → ⟨þáttur⟩ | ⟨þáttur⟩ * ⟨liður⟩



Flæðirit sem samsvarar myndunarreglunni
⟨þáttur⟩ → ⟨nafn⟩ | ⟨tala⟩ | **(** ⟨segð⟩ **)**

Til eru þokkalega fljótvirk algrím sem taka við myndunarreglum á *Backus-Naur formi* og texta og skera úr um hvort textinn er í samræmi við myndunarreglurnar. Með *Backus-Naur rithætti* eða flæðiritum er hægt að setja fram reglur sem skera úr um hvort setning er í samræmi við málfræðireglur en þessar reglur segja ekkert um merkingu þeirra.

Verkefni 8.1.d

Teiknaðu flæðirit sem samsvara myndunarreglunum

⟨skilyrðissetning⟩ → **if (** ⟨röksegð⟩ **)** ⟨skipanablokk⟩
⟨skipanablokk⟩ → **{** ⟨skipanaruna⟩ **}**
⟨skipanaruna⟩ → ⟨skipun⟩ **;** | ⟨skipun⟩ **;** ⟨skipanaruna⟩
⟨skipun⟩ → ⟨gildisgjöf⟩ | ⟨skilyrðissetning⟩

8.2 Forritunarmál

Vélamál og æðri forritunarmál

Hver tegund gjörva getur hlýtt skipunum á einu vélamáli. *Vélamál* eru yfirleitt mjög einföld að því leyti að þau hafa einfalda málskipun, lítinn orðaforða og merking hverrar setningar er óbrotin og auðskilgreinanleg. Þessi einfaldleiki veldur því að hægt er að smíða ódýrar vélar (gjörva) sem „skilja“ vélamálin en hann veldur því líka að vélamál eru afar erfið fyrir fólk því:

- Það er erfitt að læra þau þar sem skipanir samsvara ekki hugtökum sem fólki er tamt að nota;
- Hver skipun segir mjög lítið svo það þarf langar romsur til að lýsa einföldustu verkum;
- Það er erfitt að lesa forrit á vélamáli til að breyta þeim eða finna og leiðrétta villur.

Flest forrit eru samin á svokölluðum *æðri forritunarmálum*. Þetta eru mál sem hægt er að þýða á vélamál hvaða gjörva sem er því, eins og nefnt hefur verið, eru til algrím til að ákvarða hvort runur af táknum eru rétt mynduð forrit á tilteknu forritunarmáli og hvaða merkingu þau þá hafa.

Helstu kostir sem æðri forritunarmál hafa umfram vélamál eru:

- Það er auðvelt að læra þau þar sem skipanir samsvara nokkurn veginn hugtökum sem fólki er tamt að nota;
- Forrit eru þokkalega læsileg svo auðvelt er að breyta þeim og finna villur og leiðrétta þær;
- Auðvelt er að færa forrit milli ólíkra véla því fyrir flest tölvukerfi eru til þýðendur sem þýða af mörgum algengum forritunarmálum.

Smalamál eru eins konar millistig milli vélamála og æðri forritunarmála. Hver skipun á *smalamáli* samsvarar einni skipun á vélamáli. Munurinn er einkum í því fólginn að þar sem skipanir á vélamálum eru ritaðar sem tölur í tvíundakerfi eru skipanir á smalamálum ritaðar með skammstöfunum. Einnig leyfa smalamál yfirleitt að breytum séu gefin lýsandi nöfn.

Hér er forritsbútur á *Pascal* sem mælir fyrir um að skrifa skuli enska stafrófið frá A til Z.

```
FOR s := 'A' TO 'Z' DO
BEGIN
   Write(s);
END;
```

Hér kemur þýðing á smalamál fyrir *Intel 8086* örgjörva:

```
          MOV DL, 'A'
          MOV CX, 26
slaufa:   CALL skrifa
          LOOP slaufa
          INT 32
skrifa:   MOV AH, 2
          INT 33
          INC DL
          RET
```

Og er þýðing á vélamál *Intel 8086* örgjörvans:

```
10110010  01000001
10111001  00011010  00000000
11101000  00000100  00000000
11100010  11111011
11001101  00100000
10110100  00000010
11001101  00100001
11111110  11000010
11000011
```

Þegar rætt er um vélamál er yfirleitt átt við mál sem þarf að þýða forrit af æðri málum á til að tölva geti keyrt þau. Stundum er nokkur munur á þessum málum og þeim sem vélbúnaður tölva getur framkvæmt milliliðalaust. Þær skipanir á vélamáli sem ekki er hægt að keyra beint á vélbúnaðinum eru oftast túlkaðar af hugbúnaði sem er falinn í lesminni.

Forritsbúturinn sem skrifaður er á Pascal, smalamáli og vélamáli hér að ofan skrifar enska stafrófið (frá A til Z) á skjá tölvunnar. Skipanirnar sem birta stafina á skjánum sjást hvorki í smalamáls- né vélamálskótanum. Í stað þeirra eru á smalamálinu

```
          MOV AH, 2
          INT 33
```

og á vélamálinu

```
10110100  00000010
11001101  00100001
```

Þessar skipanir hlaða tölunni 2 inn í eitt af gistum gjörvans og kalla síðan á rof númer 33 með þeim afleiðingum að framkvæmd er skipanaromsa í lesminninu sem birtir staf á skjánum. Þessi skipanaromsa er ansi löng og ef henni væri skotið inn í vélamálskótann þá kæmist hann ekki á eina blaðsíðu.

Til að keyra forrit á því vélamál sem forrit eru þýdd á af æðri málum getur tölva þurft að nota heilmikið af hugbúnaði, en sá hugbúnaður er yfirleitt innbyggður í lesminni hennar og forritarar þurfa ekki að hafa neinar áhyggjur af því hvar mörkin liggja milli vélbúnaðar og hugbúnaðar í lesminni.

Stundum er forritunarmálum skipað í stigveldi eftir því hvað þau eru „nálægt“ vélbúnaðinum. Vélamál er neðarlega í þessu stigveldi. Ofan við vélamálin koma svo smalamál og þar fyrir ofan æðri mál. Æðri forritunarmál eru ekki öll á sama stað í þessu stigveldi. Sum mál eins og t.d. *C* eru neðarlega vegna þess að þau gefa forriturum kost á að vísa á tiltekna staði í minni eða tiltekin gisti í gjörva. Önnur mál eins og *Pascal* eða *Java* eru ofar vegna þess að þau gefa forriturum ekki kost á þessu. Forrit sem standa neðarlega í stigveldinu eru stundum kölluð *lágtæknimál* (á ensku *low-level languages*).

Almennt má segja að *lágtæknimál* hafi þá kosti að veita forritara fullt vald yfir vélinni með því að stjórna innihaldi einstakra vistfanga í minni og gista í gjörva. Þeir sem skrifa stýrikerfi og rekla fyrir jaðartæki þurfa oft á þessum möguleikum lágtæknimálanna að halda. Stundum er líka hægt að nýta þá til að skrifa forrit sem taka minna pláss og keyra hraðar en forrit sem skrifuð eru á æðri málum. En það er líka hægt að misnota þá til að gera verri vitleysur en hægt er með góðu móti að koma orðum að á æðri málum. Yfirleitt þarf töluverða þekkingu á vélbúnaði til að geta nýtt kosti *lágtæknimála*.

Þýðendur og túlkar

Forrit sem þýða af einu forritunarmáli á annað eru af tvennu tagi: *þýðendur* og *túlkar*. Í flestum tilvikum þýða þessi forrit af æðri forritunarmálum á vélamál. Það eru þó til þýðendur og túlkar sem þýða af einu vélamáli á annað eða milli annarra mála en vélamála.

Þýðandi (á ensku *compiler*) les heilt forrit, þýðir það allt frá upphafi til enda og skrifar þýðinguna í skrá. Ef þýðingin er á vélamál er skráin sem þýðandinn skrifar út yfirleitt keyrsluhæft forrit. Eftir að þýðandi hefur lokið verki sínu er forritið til á tveim málum: málinu sem það var skrifað á og málinu sem það var þýtt á, sem er oftast vélamál.

Verkinu sem þýðandi vinnur er hægt að lýsa á þessa leið:

```
byrja á upphafi forrits;
endurtaka meðan (ekki er komið að enda forrits)
{
   lesa næstu skipun;
   þýða skipunina sem var lesin;
   bæta þýðingunni við úttakið;
}
skrifa allt úttakið í skrá;
```

Túlkur (á ensku *interpreter*) les forrit skipun fyrir skipun og í hvert sinn sem heil skipun hefur verið lesin er hún þýdd og tölvan látin framkvæma hana. Þegar túlkur

hefur lokið störfum er búið að framkvæma forrit einu sinni en þýðingin er hvergi til. Verkinu sem túlkur vinnur er hægt að lýsa á þessa leið:

```
byrja á upphafi forrits;
endurtaka meðan (ekki er komið að enda forrits)
{
   lesa næstu skipun;
   þýða skipunina sem var lesin;
   láta tölvu framkvæma þýðinguna;
}
```

Hvort sem þýðandi eða túlkur á í hlut má skipta þýðingunni í þrjá verkþætti sem eru:

- *Lesgreining*, þ.e. að skipta runu af stöfum niður í einstök orð og merkingarbær tákn.
- *Setningafræðileg greining* sem er í því fólgin að máta þá runu af orðum og táknum sem fæst út úr lesgreiningunni við myndunarreglur sem lýsa málskipun forritunarmálsins.
- *Kótasmíð* sem er hin eiginlega þýðing. Búin er til runa af táknum á öðru máli (oftast vélamáli) sem hefur sömu merkingu og forritið.

Hugsum okkur að þýðandi taki við eftirfarandi skipun á forritunarmálinu *C*.

```
if (xhnit == 7)
{
   dx = xhnit-1;
}
```

Lesgreining skiptir þessu í orð og tákn svona:

if	(	xhnit	==	7	)	{	dx	=	xhnit	-	1	;	}

Setningafræðileg greining leiðir svo í ljós að þetta er skilyrðissetning á forminu

if (⟨röksegð⟩ **) {** ⟨gildisgjöf⟩ **}**

Kótasmíðin býr svo til vélamálskóta sem samsvarar þessari skipun.

Eftir að kótasmíði er lokið yfirfara flestir þýðendur vélamálsromsuna og stytta hana og lagfæra eins og kostur er til að forritið verði sem hraðvirkast og fyrirferðarminnst.

Nokkur algeng forritunarmál

Æðri forritunarmálum er stundum skipt í nokkrar málaættir. Flest algengustu málin, eins og *BASIC*, *Pascal, Delphi*, *C*, *C++* og *Java* tilheyra flokki *gildingarmála* (á ensku *imperative languages*). Samkenni þessara mála er að gildisgjöf (skipanir til að gefa breytum gildi) gegnir lykilhlutverki. Feikilega mörg forritunarmál tilheyra þessum flokki. Hér verða aðeins talin nokkur þau þekktustu:

- *Fortran* er elst allra æðri forritunarmála. Fyrsti *Fortran þýðandinn* var búinn til árið 1954. Nafnið er stytting á *For*mula *tran*slator. *Fortran* er enn töluvert notað af verkfræðingum og öðrum sem forrita tölvur til að vinna útreikninga einkum á sviði tækni- og raunvísinda.
- *COBOL* er eitt af elstu málunum. Það var búið til í kringum 1960. Nafnið er stytting á *Co*mmon *B*uisness *O*riented *L*anguage. *COBOL* var einkum notað til að skrifa forrit fyrir viðskiptalífið.
- *PL/1* er litlu yngra en *COBOL* og því var einnig ætlað að þjóna þörfum viðskiptalífsins.

- *BASIC* var upphaflega búið til um 1965 en þær útgáfur sem nú eru í notkun eru ansi frábrugðnar fyrstu gerð málsins. Nafnið stendur fyrir *Beginner's Advanced Symbolic Instruction Code*.

  Basic er fremur einfalt mál og auðlært. Upphaflega tóku túlkar fyrir það lítið rúm í minni og gerðu litlar kröfur til vélbúnaðar. Þegar einmenningstölvur komu fyrst fram á 8. áratugnum fylgdi þeim yfirleitt *BASIC* túlkur og síðan má segja að málið hafi þróast með einmenningstölvunum og það er enn þann dag í dag mikið notað til að semja smáforrit og fjölva fyrir einmenningstölvur.

  Sú útgáfa af *BASIC* sem nú nýtur mestra vinsælda er *Visual BASIC* frá *Miccrosoft*. Því fylgir hugbúnaður til sjálfvirkrar kótunar á notendaskilum fyrir Windows forrit. Með þessum búnaði er hægt að raða upp valmyndum, hnöppum, tækjastikum og fleiri sýnilegum hlutum og forritskótinn sem segir fyrir um gerð þeirra og staðsetningu verður til sjálfkrafa. Forritin í *Microsoft* Office pakkanum (*Word*, *Excel*, *PowerPoint*, *Access*) hafa innbyggðan túlk fyrir afbrigði af *Visual BASIC* sem heitir *Visual BASIC for Applications*. Á því máli er hægt að skrifa fjölva til að keyra í þessum forritum.

  Öfugt við flest gildingarmál (sem eru yfirleitt þýdd) er *BASIC* oftast túlkað. Fyrir sumar útgáfur eru þó bæði til þýðendur og túlkar.

- *Pascal* var búið til um 1970. Það var upphaflega hugsað sem kennslumál en náði fljótt miklum vinsældum og hefur um 30 ára skeið verið notað við hugbúnaðargerð á mörgum sviðum.

  Pascal er að verulegu leyti sniðið eftir eldra máli sem heitir *Algol* og er yfirleitt talið fyrsta mótaða forritunarmálið. Málfræði *Pascal* er þó mun einfaldari en *Algol* og raunar eru fá forritunarmál með jafn einfalda málskipan og merkingarfræði eins og *Pascal*. Fyrir vikið er það fremur auðlært og auðvelt er að búa til trausta þýðendur fyrir það.

  Ýmis nýrri forritunarmál hafa verið búin til undir áhrifum frá *Pascal*. Hér má t.d. nefna *Ada* (sem er mikið notað af bandaríska hernum) *Modula 2, Concurrent Pascal* og *Delphi*. Í sumum tilvikum er álitamál hvort þetta eru ný mál eða afbrigði af *Pascal* þau eru a.m.k. ekkert ólíkari innbyrðis en þau afbrigði sem til eru af *BASIC*.

  Meðal þeirra sem semja forrit fyrir *Microsoft Windows* stýrikerfið er *Delphi* vinsælast þessara *Pascal* afbrigða. *Delphi* býður bæði upp á alla kosti hlutbundinnar forritunar og sjálfvirka kótun á notendaskilum.

- *C* var búið til um 1970. Það er *mótað* mál eins og *Pascal* og byggir líka að nokkru leyti á *Algol*. Frá upphafi hefur *C* haft náin tengsl við *UNIX* stýrikerfið þótt þýðendur séu til fyrir flest stýrikerfi.

  C hefur sérstöðu meðal æðri forritunarmála vegna þess hvað það veitir forriturum mikið vald yfir innviðum vélarinnar. Þýðendur fyrir C skila yfirleitt mjög samþjöppuðum og hraðvirkum kóta. *C* hefur því flesta kosti *lágtæknimála* (smalamála og vélamála) en er laust við verstu galla þeirra. Fyrir vikið nýtur *C* vinsælda meðal þeirra sem semja kerfishugbúnað (stýrikerfi, rekla o.þ.u.l.).

- *C++* var búið til snemma á 9. áratugnum. Það inniheldur viðbætur við *C* sem gera mögulegt að skrifa hlutbundin forrit. Þessar viðbætur gagnast einkum við gerð stórra hugbúnaðarverkefna en auðvelda líka mjög forritun fyrir gluggakerfi.

  Síðan *C++* kom fram hefur það átt vaxandi vinsældum að fagna og er nú mest notað allra forritunarmála við lausn stórra hugbúnaðarverkefna. Það sameinar kosti

lágtæknimála og *hlutbundinnar forritunar* og gefur forriturum því bæði möguleika á að ná fullkomnu valdi yfir vélbúnaðinum og að skipuleggja flókin forrit.

- *Java* er ungt mál (búið til á árunum eftir 1990). Java markar nokkur tímamót því það var frá upphafi hugsað fyrir hlutbundna forritun, samspil tölva á neti og keyrslu á vefsíðum.

  Java svipar um margt til *C++* en er þó mun einfaldara og það býður ekki upp á möguleika á *lágtækniforritun*. Meðal mikilvægustu einkenna þess er að það er í senn þýtt og túlkað. *Java þýðandi* þýðir ekki á vélamál neins tiltekins gjörva heldur á *millimál* sem kallast *Bytecode*. *Bytecode* er afar einfalt mál og fyrir öll vélamál er auðvelt að búa til hraðvirkan túlk sem túlkar af *Bytecode*. Þegar búið er að þýða forrit af *Java* á *Bytecode* er hægt að keyra það á hvaða tölvu sem er að því tilskildu að hún hafi *Bytecode* túlk. Flest önnur þýdd mál eru þýdd á vélamál einhver tiltekins örgjörva og gerð til að nýta sér þjónustu tiltekins stýrikerfis. Hafi forrit t.d. verið skrifað á *C* og þýtt á *PC tölvu* sem keyrir *Microsoft Windows* þá er aðeins hægt að keyra þýðinguna á *PC tölvum* sem nota *Windows* stýrikerfið.

  Bytecode túlkur er innbyggður í flesta vafra (eins og *Netscape Navigator* og *Internet Explorer*) svo þeir geta keyrt *Java* forrit sem vísað er í af vefsíðum. Slík forrit kallast *Applet*. Einnig er hægt að láta vefþjóna keyra *Java* forrit og kallast þau þá *Servlet*. *Java* er þó ekki eingöngu til að skrifa forrit fyrir vefinn. Það er fullkomið alhliða forritunarmál og hentar m.a. vel til að skrifa forrit fyrir iðntölvur og sjálfvirknibúnað af ýmsu tagi.

- *Perl* er mál sem svipar um margt til *C* en er sérstakt að því leyti að það hentar vel til að vinna með strengi (þ.e. runur af stöfum og táknum). Á síðustu árum hefur *Perl* notið vaxandi vinsælda meðal þeirra sem skrifa *CGI-bin* forrit, þ.e. forrit sem keyrð eru af vefþjónum, yfirleitt til að bregðast við einhverjum skilaboðum frá vafra. Dæmi um *CGI-bin* forrit eru til dæmis leitarvélar á vefnum.

- *Javascript*. Með þróun og útbreiðslu veraldarvefsins hefur skapast þörf fyrir forrit af ýmsu tagi til að keyra í vafra og á vefþjónum. Þegar hefur verið minnst á *Java* og *Perl*. Þetta eru fullkomin alhliða forritunarmál sem eru ekkert endilega til að skrifa forrit sem tengjast vafra eða vefþjóni. *Javascript* er hins vegar einfalt mál (sem á fátt sameiginlegt með *Java*) og það er eingöngu til að skrifa forrit til að setja á vefsíður. Forrit á þessu máli eru skrifuð inn á vefsíðurnar og flestir vafrar innihalda túlk til að túlka *Javascript* forrit.

Frá spaghettíkóta til mótaðra mála

Undir lok kafla 7.1. var fjallað um algrím til að skrifa tölu í tvíundakerfi. Skoðum hvernig þetta algrím lítur út á gamaldags *BASIC*.

```
1 INPUT X%
2 S$ = ""
3 T% = X% MOD 2
4 IF T% = 1 THEN S$ = "1" + S$ ELSE S$ = "0" + S$
5 x% = INT(X% / 2)
6 IF X% > 0 THEN GOTO 3
7 PRINT S$
```

Nöfn breyta ráða tegundum þeirra (nafn sem endar á $ táknar streng og % tölu). Skilyrði eru táknuð með IF - THEN eða IF - THEN - ELSE en það er engin skipun fyrir

endurtekningu önnur en GOTO til að hoppa fram og aftur um forritið. Skipanirnar eru framkvæmdar í röð og það er engin leið að pakka þeim saman í blokkir eða undirforrit.

Langt forrit á gamaldags BASIC getur innihaldið margar GOTO skipanir sem láta keyrsluna hoppa hingað og þangað þannig að reyni maður að þræða sig gegnum forritið verði úr flækja sem minnir helst á fullan disk af spaghettí. Slík forrit eru illæsileg og það er erfitt að leiðrétta villur í þeim og fyrir vikið fékk *BASIC* hálfgert óorð á sig. Nýlegar útgáfur *BASIC* hafa þó helstu kosti mótaðra mála svo það er óþarfi að láta glósur um spaghettíforritun fæla sig frá að nota *BASIC*.

Skoðum nú hvernig þetta sama algrím lítur út ef það er skrifað á nýlegri gerð af *BASIC*.

```
INPUT N%
PRINT tuga2tvi$(N%)

FUNCTION tuga2tvi$(x%)
 S$ = ""
 WHILE x% > 0
   T% = x% MOD 2
   IF T% = 1 THEN S$ = "1" + S$ ELSE S$ = "0" + S$
   x% = INT(x% / 2)
 WEND
 tuga2tvi$ = S$
END FUNCTION
```

Aðalforritið er aðeins tvær línur. Sú fyrri (`INPUT x%`) les gildi inni í talnabreytuna N% og sú seinni skrifar útkomuna úr því að senda fallinu tuga2tvi$ gildi breytunnar N%. Fallið er sjálfstætt undirforrit með sínar eigin breytur. Endurtekning er forrituð með

```
WHILE ⟨skilyrði⟩
   ⟨skipanir⟩
WEND
```

Skipanir milli WHILE og END mynda eins konar blokk sem er endurtekin. Á svipaðan hátt er hægt að pakka mörgum skipunum saman inn í skilyrðissetningu svona:

```
IF ⟨skilyrði⟩ THEN
   ⟨skipanir⟩
END IF
```

Þessi útgáfa af *BASIC* hefur tvö af megineinkennum mótaðra mála sem eru að:

- Hægt er að skilgreina sjálfstæð undirforrit (eða föll) sem hafa staðværar breytur.
- Hægt er að pakka mörgum skipunum saman í blokkir innan í endurtekningu eða skilyrðissetningu.

BASIC er þó ekki *bálkað* eins og *Pascal* eða *C* þar sem allar skipanir tilheyra skipana-blokk og þessar blokkir geta verið hver innan í annarri. Í *Pascal* afmarkast þær af orðunum *BEGIN* og *END*, í *C* afmarkast þær af slaufusvigum. Hér á eftir er algrímið sem ritar tölu í tvíundakerfi skrifað á *Pascal*.

```
PROGRAM ur_tuga_i_tviundakerfi;
VAR x : Integer;

     FUNCTION tuga2tvi(x : Integer) : String;
     VAR s, t : String;
     BEGIN
        s := '';
        WHILE (x > 0) DO
        BEGIN
           Str((x MOD 2), t);
           x := x DIV 2;
           s := t + s;
        END;
        tuga2tvi := s;
     END;

BEGIN
     Readln(x);
     Writeln(tuga2tvi(x));
END.
```

Fallið *tuga2tvi* tekur við einni heiltölu og skilar streng. Skipanirnar í því mynda eina blokk og afmarkast af orðunum *BEGIN* og *END*. Aðalforritið er neðst. Það inniheldur tvær skipanir sem einnig mynda blokk.

Pascal er *tagað* eins og flest mótuð mál. Þetta þýðir að hver einasta breyta er skilgreind áður en hún er notuð og tekið fram af hvaða tegund hún er. T.d. eru staðværu breyturnar *s* og *t* í fallinu *tuga2tvi* af tegundinni *String*.

- Þriðja megineinkenni mótaðra mála er að hægt er að skilgreina nýjar tegundir af gögnum og byggja flóknar gagnagrindur.

  Í *BASIC* er hægt að nota fylki auk strengja, bókstafa og talna en það býður ekki upp á sömu möguleika á að skilgreina nýjar tegundir eins og mótuðu málin *Pascal* og *C*.

  Lítum t.d. á hvernig hægt er að skilgreina spilastokk í *Pascal*.

```
TYPE
   litur = (hjarta, spadi, tigull, lauf);
   gildi = 1..13;
   spil = RECORD
           l : litur;
           g : gildi;
          END;
   spilastokkur = ARRAY[1..52] OF spil;
```

Orðið TYPE táknar að á eftir komi skilgreiningar á tegundum. Fyrst er tegundin *litur* skilgreind þannig að hún geti tekið gildin *hjarta*, *spadi*, *tigull*, eða *lauf* og tegundin *gildi* er skilgreind þannig að hún geti geymt tölu milli 1 og 13. Næst er *spil* skilgreint sem færsla sem inniheldur eina breytu af tegundinni *litur* og eina af tegundinni *gildi* og að síðustu er *spilastokkur* skilgreindur sem fylki af 52 spilum.

Þegar búið er að smíða tegundina *spilastokkur* er svo hægt að skilgreina breytu af þeirri tegund svona:

```
VAR X : spilastokkur;
```

Mótuð mál og ofansækin hönnun

Mótuð forritunarmál voru hönnuð með það fyrir augum að auðvelda forriturum að skipta flóknum verkum niður í einfaldari verkþætti og nota *ofansæknar* (á ensku *top down*) aðferðir við lausn þeirra.

Við skulum taka sem dæmi að skrifa eigi forrit sem teiknar turn eins og er hér til hægri. Forritunarmálið sem við notum er blendingur af íslensku og mótuðu forritunarmáli og við gerum ráð fyrir að það hafi ekki aðrar teikniskipanir en en skipun til að teikna strik. Látum

```
strik(x1, y1, x2, y2)
```

þýða að teiknað skuli strik frá punkti með hnitin (x1, y1) til punkts með hnitin (x2, y2). Það er erfitt að teikna turninn með því að nota eingöngu þessar skipanir svo við ímyndum okkur að við höfum skipun til að teikna ferhyrning þannig að

```
ferhyrningur(x1, y1, h, b)
```

teikni ferhyrning með horn neðst til vinstri í punktinum (x1, y1), hæðina h og breiddina b. Með þessa (ímynduðu) skipun að vopni getum við búið til undirforrit sem teiknar turn úr 5 ferhyrningum sem hver um sig er af stærðinni 40 sinnum 25. Hornið neðst til vinstri er í punktinum (0, 0).

```
undirforrit turn
{
   fyrir öll gildi á i frá 0 til 4
   {
      ferhyrningur(0, i*25, 25, 40);
   }
}
```

Til að þetta virki þarf að búa til skipunina *ferhyrningur* því hún er ekki innbyggð í málið. Það er hægt að gera svona:

```
ferhyrningur(heiltölur x1, y1, h, b)
{
   strik(x1, y1, x1+b, y1);
   strik(x1+b, y1, x1+b, y1+h);
   strik(y1+h, x1+b, x1, y1+h);
   strik(x1, y1+h, x1, y1);
}
```

Þessi aðferð byggist á því að byrja á að lýsa verki í stórum dráttum með því að ímynda sér að maður hafi allar þær skipanir sem hugurinn girnist (í þessu tilviki skipunina *ferhyrningur*). Sumar þeirra eru í raun og veru innbyggðar í málið en sumar ekki. Í næstu umferð eru búin til undirforrit fyrir allar þær skipanir sem eru ekki innbyggðar í málið. Við hvert þeirra er notuð sama aðferð þar til kemur að undirforritum sem eru nógu einföld til að hægt sé að skrifa þau á orðaforða málsins.

Með mótuðum forritunarmálum fylgir iðulega safn undirforrita til að vinna algeng verk og verkþætti.

Hlutbundin forritun

Hlutbundin forritunarmál hafa alla kosti mótaðra mála en að auki bjóða þau upp á möguleika á að skilgreina *klasa* (á ensku *class*), þ.e. tegundir sem innihalda bæði gögn og aðferðir.

Hér hefur verið sýnt hvernig hægt er að skilgreina spilastokk í *Pascal*. Þessi spilastokkur getur ekkert gert annað en geymt upplýsingar um lit og gildi 52 spila. Það er engin leið að kenna honum neinar aðferðir. Eigi til dæmis að stokka hann eða sýna spilin á skjánum þá þarf að skrifa undirforrit til þess og þau eru ekki hluti af spilastokknum. Á hlutbundnum málum eins og *C++*, *Delphi*, og *Java* er hins vegar hægt að skilgreina spilastokk þannig að hann innihaldi bæði spilin og aðferðir til að gera eitthvað við þau eins og t.d. að stokka þau og sýna þau á skjánum.

Lítum á hvernig hægt er að skrifa algrímið til að umrita tölu á tvíundaform á *Java*. Eðlilegast er að skilgreina klasa sem geymir eina heila tölu og kann aðferð til að skrifa hana sem streng á tvíundaformi. Ef við látum klasann heita *Tala* þá getur hann litið svona út:

```
public class Tala
{
   private int gildi;

   public Tala(int i)
   {
      gildi = i;
   }

   public String tuga2tvi()
   {
      String s = "";
      int x = gildi;
      int t;
      while(x > 0)
      {
         t = x % 2;
         x = x / 2;
         s = t + s;
      }
      return s;
   }
}
```

Breytan *gildi* er af tegundinni *int* (þ.e. heiltala) og hún geymir töluna. Hún er *lokuð* (*private*) sem merkir að aðferðirnar sem tilheyra klasanum hafa einkarétt á að breyta gildi hennar. Auk þessarar breytu hefur klasinn tvær aðferðir. Önnur er smiður sem er notaður til að búa til hluti af tegundinni *Tala* hin, *tuga2tvi()*, skilar streng sem er talan á tvíundaformi.

Þegar þessi klasi hefur verið skilgreindur er hægt að gefa skipun eins og

```
Tala t = new Tala(25)
```

til að búa til nýja tölu með nafninu *t* og gefa henni gildið 25. Svo er hægt að nota skipunina

```
String tvi = t.tuga2tvi()
```

til að gefa strengnum *tvi* gildið sem fæst með því að láta hlutinn *t* framkvæma á sér aðferðina *tuga2tvi()*. Með skipuninni

```
t.tuga2tvi()
```

er hlutnum sem heitir *t* send skilaboð um að framkvæma aðferðina *tuga2tvi()*.

Meginmunurinn á mótuðum málum og hlutbundnum er sá að í mótuðu málunum eru gögn aðskilin frá aðferðum. Heilt forrit á mótuðu máli eins og *Pascal* eða *C* samanstendur af gögnum og aðferðum. Aðferðirnar eru undirforrit og föll sem gera eitthvað við gögnin. Gögnin eru geymd í breytum (sem geta verið einfaldar eins og heiltölubreytur, allflóknar eins og spilastokkur eða mjög flóknar eins og ættartré, líkan af gatnakerfi í stórborg eða persóna í tölvuleik) og breyturnar gera ekki neitt.

Forrit sem skrifuð eru á hlutbundnum málum innihalda yfirleitt marga hluti sem geta sent hver öðrum skilaboð. Ein mikilvæg tegund skilaboða eru svokallaðir *atburðir* (á ensku *event*). Atburður er eitthvað sem gerist í tölvunni, t.d. þegar smellt er á hnapp á lyklaborðinu, músin færð eða smellt með henni. Atburðir gegna mikilvægu hlutverki við forritun notendaskila. Með hlutbundnum málum er auðvelt að láta hnapp, valmynd eða annan hlut sem er sýnilegur á skjánum senda öðrum hlutum atburði t.d. þegar smellt er á þá með músinni. Þetta er orðað svo að með hlutbundnum málum sé auðvelt að búa til *atburðarekin forrit*.

*

Hugsum okkur að forrita eigi tölvuleik þar sem karlar og kerlingar hoppa og hlaupa um. Á hlutbundnu máli væri eðlilegast að byrja á að skilgreina klasa sem inniheldur eiginleika og aðferðir sem eru sameiginleg öllum körlum og kerlingum í leiknum. Hann gæti t.d. heitið *Persona*[2] og innihaldið *notfrjálsu* (á ensku *public*) aðferðirnar *hoppa* og *hlaupa* og *skjóta*.

Trúlega þyrfti að nota *ofansækna* hönnun til að búa til *notfrjálsu* aðferðirnar *hoppa* og *hlaupa*. En undirforritin sem lýsa einstökum pörtum þessara verka og gögnin sem þau nota yrðu algerlega *lokuð* (á ensku *private*) innan í klasanum. Að aðferð eða breyta sé *notfrjáls* þýðir að hægt að nota hana utan klasans sem hún er skilgreind í. Sé hún *lokuð* er þetta hins vegar ekki hægt.

Hugsum okkur að klasinn *Persona* innihaldi *notfrjálsa* aðferð sem heitir *hlaupa*. Gerum ráð fyrir að breytan p geymi *hlut* af tegundinni *Persona*. Við getum þá sent p skilaboð um að nota aðferðina *hlaupa* svona:

```
p.hlaupa();
```

Ef aðferðin *hlaupa* notar *lokað* undiforrit sem heitir t.d. *beygjaHnje* þá getum við ekki skipað:

```
p.beygjaHnje();
```

því *beygjaHnje* er ekki notfrjáls aðferð. Með því að hafa sumar aðferðir og sumar breytur *notfrjálsar* og aðrar *lokaðar* geta klasar skammtað aðgang að innviðum sínum. Það er því eins og hver hlutur segi við hina hlutina: Ég leyfi ykkur ekki að fikta í mínu dóti því þá getið þið klúðrað málunum. Þið megið nota aðferðir og breytur sem ég skilgreini sem *notfrjálsar* og þar sem ég fæ að hafa allt annað í friði get ég tryggt að þær virki rétt. Þessi eiginleiki hlutbundinnar forritunar, að geta *lokað* gögn og aðferðir inni í sínum klösum kallast ýmist *hjúpun* eða *upplýsingahuld* (á ensku *encapsulation*).

Við ætluðum að hafa sumar persónurnar í leiknum karla og sumar áttu að vera kerlingar. Klasinn *Persona* inniheldur allt sem karlar og kerlingar eiga sameiginlegt. Klasarnir *Karl* og *Kerling* þurfa ekki að innihalda þessi sameiginlegu atriði. Þeir geta erft þau frá *Persona*. Ásamt möguleikum á að *hjúpa* aðferðir og gögn eru *erfðir* (á

[2] Hér er fylgt þeirri venju að láta klasa hafa nöfn sem byrja á stórum staf.

ensku *inheritance*) með mikilvægustu einkennum hlutbundinnar forritunar. Með því að láta einn klasa erfa eiginleika annars er hægt að endurnýta forritskóta sem er búið að fullvinna og þaulprófa án þess að hræra neitt í honum.

Enn eru ótaldir tveir mikilvægir eiginleikar hlutbundinna forritunarmála sem eru er *fjölbreytni* og *kvikleg binding*.

Kvikleg binding (á ensku *dynamic binding*) þýðir að það getur ráðist í keyrslu forrits af hvaða tegund breyta er, það þarf ekki að negla það fast um leið og forritið er þýtt. Við getum t.d. hugsað okkur að breytan k sé skilgreind sem *Persona* og ef notandi smellir með músinni á einn hlut á skjánum þá fái hún gildi af tegundinni *Karl* en ef smellt er á annan stað þá fái hún gildi af tegundinni *Kerling*.

Fjölbreytni (á ensku *polymorphism*) þýðir að margar aðferðir geta heitið sama nafni og þegar nafnið er notað ræðst það af samhengi hver af aðferðunum er framkvæmd. Við getum t.d. hugsað okkur að klasinn *Karl* hafi aðferð sem heitir *dansa* og lætur karlinn dansa ólman stríðsdans og klasinn *Kerling* hafi líka aðferð sem heitir *dansa* og hún láti kerlinguna kjaga kringum mannætupott. Ef forrit inniheldur skipunina

```
k.dansa();
```

þá er útkoman stríðsdans ef gildi k er af tegundinni *Karl* en öðru vísi dans ef það er *Kerling* og það getur ráðist eftir að keyrsla forrit hefst hvort tegund breytunnar er skilgreind af klasanum *Karl* eða af klasanum *Kerling*.

Við getum nú dregið saman í stutt mál helstu einkenni hlutbundinna forritunarmála:

- Í hlutbundnum málum er gögnum og aðferðum pakkað saman í einingar sem kallast *klasar*.
- Breyta af tegund einhvers klasa er *hlutur* sem inniheldur eiginleika og aðferðir klasans. Hlutir geta sent hver öðrum *skilaboð* og stórt forrit er yfirleitt safn margra hluta sem skiptast á *skilaboðum* og bregðast við *atburðum*.
- Hlutbundin mál bjóða upp á *hjúpun*, *erfðir*, *kviklega bindingu* og *fjölbreytni*.

Rétt eins og söfn undirforrita fylgja mótuðum málum fylgja hlutbundnum málum söfn af klösum. Flestir forritarar nýta sér slík söfn og bæta við sínum eigin klösum sem oft erfa eiginleika og aðferðir frá klösum sem fylgja með málinu. Einnig ganga klasar milli forritara og sumir eiga stór söfn til viðbótar þeim sem fylgja forritunarmálinu. Þegar forritari lýkur við forrit hefur hann því kannski aðeins skrifað brot af öllum þeim kóta sem forritið er gert úr.

Þótt hver klasi sé ef til vill ekki mjög frábrugðinn forriti á mótuðu máli auðvelda þeir eiginleikar hlutbundinna mál sem hér hefur verið fjallað um forriturum mjög mikið að glíma við flókin verkefni, skipta með sér verkum og nýta vinnu hver annars. Á móti kemur að hlutbundin mál eru yfirleitt fremur flókin svo það tekur lengri tíma að læra þau almennilega heldur en einfaldari mál eins og *Pascal* eða *BASIC*.

Aðrar málaættir

Flest algengustu forritunarmálin tilheyra flokki *gildingarmála*. Mál af öðrum ættum gegna þó mikilvægu hlutverki á ýmsum sviðum tölvufræðinnar. Hér verður fjallað stuttlega um tvenns konar öðru vísi forritunarmál *fallamál* og *rökfræðimál*.

Elsta *fallamálið* (á ensku *functional language*) heitir *Lisp*. Það kom fram þegar í árdaga tölvanna fyrir 1960 og var byggt á *lambda calculus* sem er eins konar forritunarmál sem rökfræðingurin *Alonso Church* bjó til í þeim tilgangi að skilgreina stærðfræði-

lega aðferð á 4. áratug 20. aldar. Nokkur önnur fallamál eins og *Scheme* og *Logo* eru náskyld *Lisp*. Þessi mál eru afar öflug að því leyti að á þeim er hægt að orða flóknar aðferðir í stuttu máli. Málskipan þeirra og merkingarfræði er líka einfaldari en í flestum öðrum forritunarmálum. Á móti kemur að fallamálin gera yfirleitt miklar kröfur til vélabúnaðar, forritin eru frek á minni og yfirleitt keyrð af túlkum sem eru heldur svifaseinir. Þótt þessi mál henti vel til ýmiss konar tilraunastarfsemi eru þau af þessum sökum illa til þess fallin að smíða notendaforrit sem þurfa að keyra hratt á ódýrum vélum.

Logo er mest notað til að kenna byrjendum forritun. *Lisp* og *Scheme* gegna mikilvægu hlutverki við rannsóknir í ýmsum greinum tölvufræði m.a. gervigreindarfræðum (þ.e. fræðum sem snúast um að fá tölvur til að líkja eftir mannlegu hugarstarfi og skynjun).

Meðal einkenna fallamála eru:

- Listavinnsla. Gögn eru geymd í listum sem eru kviklegar og afar sveigjanlegar gagnagrindur.
- Forrit og gögn eru ekki aðgreind. Forrit eru listar af táknum og gögnin sem þau vinna með eru líka listar af táknum. Þetta þýðir að forrit geta breytt sjálfum sér og auðvelt er að búa til forrit sem bæta aðferðum við sjálf sig.
- Tiltekið er hvað forrit á að gera með því að skilgreina föll og láta þau kalla hvert á annað fremur en með því að telja upp í röð það sem á að gera.

Hér er ekki rúm til að gera fallamálum frekari skil. Forritið sem hér fer á eftir er skrifað á *Scheme*. Það tekur við tölu og skrifar hana á formi tvíundakerfis.

```
(define (tuga2tvi x t)
 (if (> x 0)
  (tuga2tvi (quotient x 2) (append (list (remainder x 2)) t ))
   t
 )
)

(define (tvi x)  (tuga2tvi x '()))
```

Eins og önnur mál á *Lisp-ættinni* er *Scheme* gagnvirkt þannig að um leið og búið er að slá inn eitt fall er hægt að láta túlkinn keyra það. Það er t.d. hægt að keyra fallið *tvi* (sem kallar á *tuga2tvi*) og fá að sjá hvernig 25 lítur út í tvíundakerfi með því að skrifa:

```
(tvi 25)
```

og *Scheme* túlkurinn svarar með:

```
(1 1 0 0 1)
```

*

Af *rökfræðimálum* er *Prolog* einna þekktast. Líkt og *Lisp* og *Scheme* er það enn sem komið er mest notað við rannsóknir og tilraunastarfsemi. Það hefur þó verið notað til að smíða sérfræðikerfi (þ.e. forrit sem draga rökréttar ályktanir af upplýsingum í gagnasafni, yfirleitt á einhverju sérfræðisviði).

Prolog er forritunarmál af 4. kynslóð sem þýðir að hver skipun samsvarar yfirleitt mjög löngum vélamálskóta og forritari getur í mörgum tilvikum látið duga að lýsa því hvað forrit á að gera í stað þess að skrifa algrím sem tiltekur skref fyrir skref hvernig á að gera það. Forrit á *Prolog* eru skrifuð sem runur af staðreyndum og skilgreiningum á eiginleikum og venslum. Eigi til dæmis að skrifa þær staðreyndir að Úlfur sé faðir

Gríms og Grímur sé faðir Þórólfs og Egils og Egill sé faðir Böðvars þá er hægt að gera það svona:

```
faðir(úlfur, grímur).
faðir(grímur, egill).
faðir(grímur, þórólfur).
faðir(egill, böðvar).
```

Það er svo hægt að skilgreina venslin afi og bróðir með

```
afi(A, B) :- faðir(A, X), faðir(X, B).
bróðir(A, B) :- faðir(X, A), faðir(X, B).
```

Fyrri skilgreiningin segir að A sé afi B ef A er faðir einhvers, X, og sá sami X er faðir B. (Hér er öllu kvenfólki sleppt og skilgreiningin yrði dálítið flóknari ef það væri haft með.)

Þessar fjórar staðreyndir og tvær skilgreiningar mynda saman ofurlítið forrit. Þegar *Prolog* túlkur hefur verið mataður á því er hægt að leggja fyrir hann spurningar eins og

```
afi(X, böðvar).
```

og túlkurinn svarar

```
X = grímur
```

Prolog er að því leyti líkt málum af *Lisp-ættinni* að listavinnsla gegnir mikilvægu hlutverki. Hér fer á eftir forrit á *Prolog* sem skrifar tölu á formi tvíundakerfis.

```
dec2bin(0, [0]).
dec2bin(1, [1]).
dec2bin(X, [Haus|Hali]) :- Haus is X mod 2,
                           Y is X // 2,
                           dec2bin(Y, Hali).
conc([], L, L).
conc([X|L1], L2, [X|L3]) :- conc(L1, L2, L3).

snua([],[]).
snua([Haus|Hali], X) :- snua(Hali, Y),
                        conc(Y, [Haus], X).

tuga2tvi(X, S) :- dec2bin(X, S1), snua(S1, S).
```

Þegar *Prolog* túlkur hefur verið mataður á þessum skilgreiningum er hægt að gefa fyrirmæli eins og

```
tuga2tvi(25, S).
```

og túlkurinn svarar

S = [1, 1, 0, 0, 1]

Skilgreiningin og staðreyndirnar um venslin *dec2bin* duga raunar til að breyta tölu í lista af tölustöfum í tvíundakerfi en sá listi er öfugur. Venslin *conc* og *snua* snúa honum við.

8.3 Hugbúnaðargerð, villur og áreiðanleiki

Hugbúnaður – tvöfalt stærri, margfalt dýrari

Þegar tölvur hófu innreið sína í atvinnulíf á árunum eftir 1960 voru vélarnar dýrar og kostnaðurinn við kaup og viðhald þeirra margfalt meiri en það sem greitt var fyrir hugbúnað. Nú hefur þetta hlutfall fyrir löngu snúist við og fyrirtæki sem nota tölvur greiða að jafnaði miklu meira fyrir hugbúnað heldur en fyrir vélbúnað.

Fyrir þá sem aðeins hafa skrifað smáforrit, eins og verkefni í byrjendabók í forritun, getur verið erfitt að gera sér grein fyrir því hve mikil vinna liggur að baki hugbúnaði eins og bókhaldskerfi, töflureikni eða tölvuleik. Það er freistandi að hugsa sem svo að ef það tekur einn mann viku að skrifa smáforrit þá geti hann skrifað 50 sinnum stærra forrit á einu ári. En reynslan sýnir að þetta er fjarri sanni. Stór hugbúnaðarverkefni eru yfirleitt mun dýrari og mun tímafrekari en þeir sem aðeins hafa reynslu af smíði lítilla forrita geta með góðu móti gert sér grein fyrir. Fyrir þessu eru ýmsar ástæður:

- Lítil forrit eins og fjölvar eða smáforrit á vefsíðum eru yfirleitt skrifuð sem ein heild. Stórir hugbúnaðarpakkar eru hins vegar samsettir úr mörgum hlutum sem þurfa að spila saman og það er tímafrek vinna að skilgreina viðmót og samskeyti og tryggja snurðulausa samverkun.
- Smáforrit þarfnast ekki *skjalbúnaðar* (þ.e. leiðarvísa, handbóka, upplýsinga um viðhald o.s.fr.) en með stórum hugbúnaðapökkum þarf að fylgja verulega mikið af gögnum og upplýsingum svo hægt sé setja þá upp, nota þá, viðhalda og endurbæta þótt höfundar þeirra séu ekki við.
- Stóra hugbúnaðarpakka þarf að vera hægt að keyra í fjölbreytilegu umhverfi (t.d. á öllum PC tölvum sem keyra *Microsoft Windows* hvort sem þær nota Windows 95, 98 eða NT og hvort sem þær eru stakar, eða hluti af staðarneti og hvort sem þær vista gögn á eigin diski eða á diski netþjóns o.s.fr.).
- Stórir hugbúnaðarpakkar þurfa yfirleitt að mæta fjölbreytilegum þörfum sem er erfitt að hafa yfirsýn yfir og skilgreina nákvæmlega. Sá sem skrifar smáforrit veit oftast nákvæmlega hvað forritið á að gera. En þeir sem vinna stór verkefni þurfa oft og iðulega að verja miklum tíma til að átta sig á vinnubrögðum, þörfum og hugsunarhætti væntanlegra notenda.

Þetta þrennt sem talið hefur verið veldur því að vinna við hverja línu af kóta í stórum hugbúnaðarpakka er margfalt meiri en í smáforriti. Enn er þó ótalið það sem mestu skiptir:

- Stórir hugbúnaðarpakkar eru iðulega samvinnuverkefni margra forritara, enda tæki vinna við þá mörg ár og jafnvel áratugi ef hún væri unnin af einum manni. En tveir forritarar sem vinna saman afkasta alls ekki tvöfalt meiru en einn, því þótt hlutbundin forritun hafi greitt mjög fyrir samvinnu forritara fer samt verulega mikill hluti af tíma þeirra í að samstilla kraftana og koma sér saman um verkaskiptingu og tilhögun eða skilgreiningu einstakra verkþátta. Þegar margir forritarar vinna saman getur þetta þýtt að hver þeirra skrifi tvöfalt, fjórfalt eða áttfalt færri línur af kóta á dag en hann gerði ef hann ynni einn. Ef einn maður yrði tvö ár að ljúka verkefni, en það þarf að ljúka því á einu ári, þá getur því þurft að fjölga forriturum í fjóra, átta eða jafnvel sextán fremur en tvo.

Frá hönnun til útgáfu

Vinnu við gerð hugbúnaðar er hægt að skipta í nokkra áfanga. Alls fyrst þarf auðvitað að ákveða nokkurn veginn hvað á að gera og safna saman forriturum og öðru sérfróðu fólki til að vinna verkið. En eftir að hin eiginlega vinna er hafin eru helstu áfangar á leið frá hönnun til útgáfu þessir:

- Lýsing á því hvað hugbúnaðurinn á að gera og hvernig hann á að vera. Slík lýsing getur verið ansi langt mál. Hún þarf m.a. að tiltaka hvaða þörfum forritið á að þjóna og lýsa notendaskilum þess.
- Lýsing á innviðum forritsins: Hvernig skiptist það í klasa; Hvers konar gagnagrindur eru notaðar og hvaða algrím eru notuð til að vinna einstaka verkþætti.
- Verkaskipting milli einstaklinga og vinnuhópa ákveðin.
- Forritunin sjálf. Forritarar skrifa kóta á einhverju forritunarmáli. Á þessu stigi koma iðulega í ljós vankantar á þeirri hönnunar- og skipulagsvinnu sem unnin var á fyrri stigum svo það verður að endurtaka hana að einhverju leyti.
- Prófun einstakra klasa og forritshluta. Leiði prófanir í ljós galla í hönnun eða forritun getur þurft að endurtaka einhvern hluta þeirrar vinnu.
- Samtenging allra hluta hugbúnaðarins í eina heild og prófanir á samspili þeirra.
- Samning skjalbúnaðar (leiðarvísa, handbóka o. þ. u. l.)
- Einhver hópur væntanlegra notenda fenginn til að reyna bráðabirgðaútgáfu af hugbúnaðinum. Slík bráðabirgðaútgáfa er oft kölluð *beta-útgáfa*.
- Hugbúnaðurinn er lagaður í ljós reynslunnar af *beta útgáfunni*.
- Útgáfa og markaðssetning: Þegar hingað er komið er vinna við gerð annarrar útgáfu ef til vill hafin.

Villur

Stór hluti af vinnunni við hugbúnaðargerð snýst um prófanir, leit að villum, lagfæringar og leiðréttingar. Samt eru villur í flestum forritum og sumar þessar villur leyna á sér, forrit getur jafnvel verið í notkun árum saman án þess að þær komi í ljós.

Hinn svokallaði 2000-vandi var t.d. í því fólginn að forrit geymdu ártöl sem tveggja stafa tölu. Þannig var ártalið 1998 geymt sem talan 98. Oft þurfa forrit að finna bil milli tveggja dagsetninga (t.d. til að reikna vexti) og hluti af þeim útreikningum er iðulega í því fólginn að draga eitt ártal frá öðru. Þannig er bilið milli áranna 1991 og 1998 reiknað sem 98-91=7 ár. En ef ártalið 2000 er geymt sem talan 00 þá reiknast bilið milli 1998 og 2000 vera 00-98=-98 ár. Þessi villa leyndist árum saman í fjölmörgum forritum og gerði fyrst vart við sig um áramótin 1999 til 2000.

Á ensku eru villur í forritum stundum kallaðar *bugs* (þ.e. pöddur) og forrit sem leitar uppi villur í kóta kallað *debugger*. Á bak við þetta orðalag er sú saga að *Grace Hooper* (einn af höfundum forritunarmálsins *COBOL*) hafi lent í því einhvern tíma í árdaga tölvutækninnar að vélin sem hún vann við hagaði sér undarlega. Grace leitaði án árangurs að villu í forritinu en fann enga. Þá stóð hún upp, opnaði tölvuna (sem var á stærð við heilt herbergi) gekk inn í hana og fann hvað var að: Skorkvikindi hafði troðið sér milli raflagna með þeim afleiðingum að vélin vann ekki sem skyldi. Þetta dýr var víst ekki lús en á íslensku er samt talað um að forrit sem innihalda mikið af villum séu

lúsug og forrit sem leita uppi villur í kóta kölluð *kembiforrit* (enda eru lúsakambar notaðir til að greiða lýs úr hári manna).

Villum í forritum er hægt að skipta í tvo meginflokka: *Málvillur* og *hugsunarvillur*.

Það er *málvilla* þegar forritskótinn samrýmist ekki málskipun forritunarmálsins. Þýðendur eiga að bregðast við slíkum villum með því að neita að þýða forritið en þýðendur geta innihaldið villur rétt eins og önnur forrit svo stundum valda málfræðivillur því að til verður gallaður vélamálskóti. Flestum þýðendum fylgja kembiforrit sem finna málfræðivillur og aðstoða forritara við að leiðrétta þær.

Alvarlegar villur í hugbúnaði koma þó oftast til af öðru en því að forritskóti samræmist ekki málfræðireglum. Þetta eru *hugsunarvillur* af ýmsu tagi. Sumar valda *inningarvillum* (á ensku *run-time error*) þ.e. villum sem koma í ljós þegar forrit er keyrt og valda því oft að keyrslan stöðvast. Sem dæmi um villur af þessu tagi má nefna deilingu með núlli eða yfirflæði í reikningi (á ensku *arithmetic overflow*) sem verður þegar útkoma úr reikniaðgerð fer út fyrir þau mörk sem gisti eða breyta getur rúmað. Aðrar hugsunarvillur birtast ekki með jafnaugljósum hætti. Þær eru *rökvillur* sem valda því að forrit gerir eitthvað annað en því er ætlað að gera eða skilar rangri niðurstöðu. Það getur verið mjög erfitt að sannreyna að forrit sé laust við *rökvillur*.

Rökvillur, forskilyrði og eftirskilyrði

Það er hægt að fullyrða án fyrirvara að forritskóti innihaldi málvillu eða að inningarvilla verði við keyrslu þess en það er ekki hægt að tala um rökvillu nema með hliðsjón af einhverri lýsingu á hvað forriti er ætlað að gera. Ef forriti er t.d. aðeins ætlað að meðhöndla dagsetningar frá 1.1.1900 til 31.12.1999 þá er varla hægt að tala um *rökvillu* þó það fari rangt með dagsetninguna 1.1.2000. Það er heldur ekkert undan því að kvarta þó forrit skrifi á skjáinn 2+2=5 ef því er aðeins ætlað að vera skjáhvílir. Sé því hins vegar ætlað að vera reiknivél hlýtur þetta að teljast *rökvilla*.

Eitt af þeim verkum sem ekki er hægt að leysa með algrími er að skera úr um hvort forrit hagar sér í samræmi við lýsingu á verki eða greinargerð fyrir því hvað það á að gera. En þótt ekki sé til pottþétt aðferð til að komast að þessu í öllum tilvikum er stundum hægt að leiða í ljós, og jafnvel að sanna með stærðfræðilegum hætti, að forrit eða forritshluti vinni sitt verk eins og til er ætlast.

Eitt af mikilvægustu hjálpartækjum sem forritarar nota til að staðfesta réttmæti og áreiðanleika forrita er nákvæmar lýsingar á *forskilyrðum* og *eftirskilyrðum*. Flest forrit skiptast í nokkra klasa og hver klasi inniheldur nokkrar aðferðir. Sé ofansækin aðferðafræði notuð er hver aðferð mynduð úr nokkrum stuttum undirforritum. Til að staðfesta að hvert undirforrit (eða hver forritshluti) vinni rétt þarf að lýsa því hvað það á að gera. Þetta er yfirleitt gert með því að tiltaka *forskilyrði* og *eftirskilyrði*.

- *Forskilyrði* segja í hvaða ástandi gögn mega vera og hvaða gildi breytur mega hafa áður en tiltekinn hluti forrits er framkvæmdur.
- *Eftirskilyrði* segja hvaða áhrif tiltekinn hluti forrits hefur á gögn og breytur, á hvern hátt staða þeirra eftir er fall af stöðu þeirra fyrir keyrslu þessa hluta.

Til upprifjunar

1. Hvað eru:
   Backus-Naur ritháttur, BASIC, beta útgáfa, Bytecode, C, C++, eftirskilyrði, erfðir, fallamál, fjölbreytni, flæðirit, forskilyrði, gildingarmál, inningarvilla, Java, Javascript, kembiforrit, klasi, kótasmíð, kvikleg binding, lágtæknimál, lesgreining, Lisp, Logo, lokatákn, málskipan, merkingarfræði, notfrjáls aðferð, ofansækin hönnun, Pascal, Perl, Prolog, Scheme, smalamál, túlkur, vélamál, þýðandi, æðri forritunarmál.

2. Nefndu dæmi um:
   Fallamál, gildingarmál, hlutbundin mál, lágtæknimál, mótuð mál.