# Informaatika õpetamisest mitteinformaatikutele

## Informaatika põhivaldkonnad

Kõigepealt üritaks täpsustame vaadeldava teemaga seotud mõningaid mõisteid. Selles vallas valitseb üsna suur mitmekesisus erinevates keeltes ja süsteemides. Seda mida paljudes Euroopa riikides nimetatakse informaatikaks (informatics) nimetatakse USAs arvutiteaduseks (computer science). Kusjuures ka olemuse ja sisu tõlgendused on sageli üsna erinevad. Ei ole üheseid määratlusi näiteks mitmel siin kasutataval mõistel nagu infotöö ja arendustöö jm. Tegemist ei ole siin definitsioonidega või määratlustega vaid selgitustega, mis ehk aitavad paremini mõista järgnevat. Õpetamise seisukohast võiks eristada kolme tihedalt seotud valdkonda.

**Programmjuhtimisega süsteemide** (PJS) **arhitektuur**: riist- ja süsteemitarkvara, töökor­ral­dus, protokollid, failisüsteemid, andmete esitusviisid jm. Kunagi oli siin tegemist peamiselt arvutitega. Praegusel ajal on programmjuhti­misega süsteemide valik väga lai: arvutid, arvutivõrgud, robotid, telefonid jm. Õppijate seisukohast on esmaseks ja peamiseks arvutid ja arvutivõrgud.

**Infotöö**: informatsiooni kasutamine, loomine, säilitamine ja edastamine programm-juhtimisega süsteemide abil: dokumentide lugemine, koostamine ja redigeerimine; arvutused ja andmeanalüüs; joonistamine, joonestamine, pilditöötlus jmt; informatsiooni hankimine ja korrastamine, kommunikatsioon, Infotöö jaoks on tänapäeval olem väga lai valik üldotstarbelist ja spetsialiseeritud tarkvara: tekstitöötluse ja tabeliprogrammid, brauserid, graafikaprogrammid, matemaatika- ja statistikaprogrammid, andmebaasisüsteemid, infosüsteemide kasutajaliidesed jm.

## Arendustöö: rakenduste, infosüsteemide ja tarkvara loomine ja arendamine. Peamised tegevused on siin andmete ja protsesside ning süsteemide modelleerimine, süsteemianalüüs, disain, algoritmimine, program­meerimine (kodeerimine). Seoses rakenduste ja infosüsteemide mahu ja keerukuse kasvuga üha suuremat osakaalu saab modelleerimine. Viimase vahendeid kasutatakse süsteemide struktuuri ja käitumise formaalseks kirjeldamiseks ja projekteerimiseks (disainiks).

## Väikene tagasivaade ja taust

Informaatika õpetamine mitteinformaatikute on kogu aeg olnud üheks oluliseks tegevuseks instituudi jaoks. Informaatika üldkursust on Informaatikainstituut õpetanud ja õpetab ülikooli kõikidele mittemitteinformaatika erialadele. Viimasel ajal on iga aasta asunud õppima esimesele kursusele üle 1200 mitteinformaatika eriala üliõpilase.

Üldinformaatika õpetamist alates algusest kuni aastani 1993 käsitleti raamatutes [1, 2]. Viimases on toodud ka „Informaatika õpetamise kontseptsioon mitteinformaatikule“. See oli loodud teaduskondade vahelise töögrupi poolt. Kontseptsioonis fikseeritud üldised põhimõtted ja eesmärgid ning kursuse struktuur ja maht on jäänud kehtima praktiliselt käesoleva ajani. Loomulikult neid ja eriti kursuse sisu on järk järgult täiendatud ja uuendatud vastavalt infotehnoloogia arenguga. Tegemist on ju kiiresti areneva valdkonnaga. Põhjalik ülevaade ja analüüs informaatika õpetamisest mitteinformaatikutele läbi aegade ja eriti viimasel ajal ning ettepanekud arenguks on toodud aastal 2008 koostatud Kersti **Antoi** magistritöös „Visioon mitteinformaatika erialade informaatikakursuse arendamiseks“.

### Infotöö tulek. Suurte segaduste aeg

Toll ajal (80-te lõpp, 90-te algus) aga oli tegemist olulise ülemineku perioodiga: suurarvutite asemele hakkasid tulema kasutusse personaalarvutid. Kui varem oli kursuse sisuks programmeerimine ja algoritmimine, käsitlemist leidis arvuti riistvara ja süsteemitarkvara (kasutusel oleva opsüsteemi praktiline kasutamine), siis nüüd tekkis ja kiiresi arenes uus valdkond: **infotöö**. Selle mõiste (termini) võttis meil kasutusele Toomas Mikli: täienduskoolituse dekaan, infotöö rakendamise suur entusiast. Infotöö põhisisuks sai tekstitöötlus, tabelarvutused, esitlused, arvutigraafika ja personaalarvuti andmebaasid. Need tegevused on infotöös keskel kohal ka käesoleval ajal, kuigi vahendid on oluliselt täiustunud ja lihtsustunud. Olgu märgitud, et näiteks tekstide redigeerimine ja vormindamine tolleaja peamiste tekstitöötluse programmidega: WordStar ja WordPerfekt, meenutab, eriti tagantjärele, veidi programmeerimist.

Samal ajal oli õppetööks mõeldud personaalarvutite hulk ülikoolis üsna väike ning need olid erinevat tüüpi ja varustatud üsna erineva infotööd võimaldava tarkvaraga. Sellepärast oli õppetöö läbiviimine üsna keeruline: tuli kasutada neid vahendeid, mis olid parajagu kättesaadavad. Näiteks mõnikord isegi ühe teaduskonna üliõpilased kasutasid samal semestril samas aines erinevaid arvuteid ja tarkvara.

Pikemat aega programmeerimise osa jaoks ei saanud kasutada personaalarvuteid vaid siin kasutati suurarvuteid (IBM/360 koopiad: ES arvutid olid suured mõõtmetelt aga mitte võimsuselt ja nad ei viitsinud korralikult töötada), miniarvuteid (PDP/8 koopiad: SM arvutid, olid väiksemad ja veidi võimsamad ning isegi töötasid), Sun tööjaamu: Rootsis vist mahakantud ja meile kingitud – võimsad ja töökindlad. Mitteinformaatikute programmeerimise algkursuse jaoks olid, võiks öelda, isegi ülemäära luksuslikud.

Suur segadus õpetöös kasutatava riistvara ja tarkvaraga oli praktiliselt aastatuhande vahetuseni. Veidi enne seda (aasta?) toimus nö „tehnoloogiline revolutsioon“, Infotehnoloogia teaduskonna ettepanekul ja korraldusel, võttis ülikool laenu ja liisis (?) korraga ligi sada (?) võiks öelda, kõige moodsamat personaalarvutid ja vastava tarkvara,

Alates siit tekkis võimalus kavandada õppetöö sisu, vahendeid ja metoodikat mitte olude sunnil, vaid arvestades püstitatud eesmärke ja vajadusi. Samal ajal pidi siis ja peab ka praegu arvestama informaatika õpetamisega koolides. Alguses koolides informaatikat praktiliselt ei olnud ja õpetamine algas nö puhtalt lehelt. Hiljem ja ka praegu peab arvestama, et koolides ühtset ainekava selles osa ei ole ning informaatika ei ole kohustuslikuks aineks (tegemist on nn läbiva ainega: õpetavad kõik, st mittekeegi?) on ettevalmistus tase nö seinast seina.

### Maha programmeerimine või üleminek arendustööle

Nagu on tüüpiline „revolutsioonidele“, uue tekkimisel tekkib paljudele soov hävitada varem olnu. Vastukaaluks toll ajal käibel olnud akadeemik A. Jeršovi (koolide arvutiseerimise peaideoloog NSV Liidus) loosungile „Programmeerimine – teine kirjaoskus!“, tekkis ka loosung „Maha programmeerimine!“. Paljud siis ja ka praegu arvavad, et programmeerimist peab õpetama ainult informaatikutele ning näiteks meie koolides on programmeerimise õpetamine oluliselt vähenenud. Ka TTÜs on mõnede erialade instituudid taotlenud programmeerimise ärajätmist. Ülalnimetatud kontseptsioonis on selgelt fikseeritud aine jagunemist kaheks osaks: infotöö ja programmeerimine, ning viimase õpetamist TTÜs on õnnestunud säilitada.

Tegelikult on viimasel ajal märgata programmeerimise taassündi arendustöö ühe olulise osana. Programmeerimise olemus ja sisu üsna oluliselt ja kiiresti muutunud. Rakenduste ja süsteemide mahu ja keerukuse kasvamine ning programmeerimistehnoloogiate ja ‑vahendite areng, suurendab üha analüüsi ja disaini etappide osakaalu arendusprotsessis. Sellepärast on tekkinud ja kiiresti arenevad ka vastavad tehnoloogiad ja vahendid: objektorienteeritud (OO) analüüs ja disain, modelleerimiskeeled, CASE‑süsteemid jm. Samal ajal on tekkinud ja arenenud uued tehnoloogiad ja vahendid realisatsioonietapi (ehk kitsamas mõttes programmeerimise) jaoks: OO-programmeerimine, komponenttehnoloogia, uue põlvkonna rakendusprogrammide arendusvahendid, mille tüüpiliseks esindajaks näiteks Visual Basicul põhinev arendussüsteem VBA.

Programmeerimise mõistet käsitletakse laiemas ja kitsamas mõttes. Laiemas tähenduses see hõlmab kõiki rakenduste (programmide) loomisega ja arendamisega seotud faase ehk etappe: ülesande püstitus, probleemi analüüs, projekteerimine ehk disain (andmete organisatsiooni ja programmi struktuuri määramine, algoritmide koostamine), programmide koostamine ehk kodeerimine, silumine ja testimine, dokumenteerimine. Võiks öelda, et selles tähenduses programmeerimise mõiste langeb peaaegu kokku arendustöö mõistega. Kitsamas tähenduses mõistetakse programmeerimise all tavaliselt järgmisi tegevusi: algoritmimine, programmide koostamine, silumine ja testimine. Õpetamisel aga sageli taandatakse programmeerimine ka konkreetse programmeerimiskeele

## Üldinformaatika kursuse eesmärgid ja põhisisu

Tegemist on üldainega (nagu matemaatika, füüsika jmt). Eesmärgid jagunevad kahte rühma:

* **üldarendavad ja üldharivad eesmärgid**:
arendada loogilist, analüütilist ja algoritmilist mõtlemist ning süsteemset lähenemis­viisi probleemide ja ülesannete lahendamisel,
anda ettekujutus andmete ja protsesside modelleerimisest ning kirjeldamisest,
teadvustada ja tunnetada programmjuhtimisega süsteemide töö ja informatsiooni töötlemise põhimõtteid
* **pragmaatilised eesmärgid**:
saadud oskuste kasutamine õppetöös: paralleelsed ja järgnevad ained,
kasutamine tulevas kutsetöös: ülesannete püstitamine, osalemine IT projektides jm,
luua eeldused täiendavate IT ainete õppimiseks

Viimastes õppekavades on moodustunud kolm üldinformaatika kursust: keemia, tehnika ning majandus ja humanitaar, mis arvestavad valdkondade ja õppesuundade iseärasusi. Piltlikult öeldes tegemist on sinimustvalge jaotusega, arvestades infotöö ja arendustöö suhet kursustes. Tehnika erialadel on arendustöö osakaal kõige suurem, majanduses ja eriti humanitaarerialadel moodustab infotöö kursuses arvestatava osa. Keemiaerialadel on kursuse maht 6APd (üks semester 4 t/n), tehnikal ja majanduses 8 APd (kaks semestrit 3t/n)

Kõikides kursustes laiendatakse ja süvendatakse infotööga seotud teadmisi ja oskusi, kuid oluline on ka arendustööl, eriti tehnikaerialadel. Käsitletakse andmete ja protsesside model­leerimist, meetodeid ja vahendeid rakenduste loomiseks, algoritmimise ja programmeeri­mise aluseid. Vaadeldakse rakenduste loomise üldiseid põhimõtteid, meetodeid, vahendeid ja põhifaase: ülesande püstitus, analüüs, disain, realisatsioon. Tutvustatakse objektorienteeritud modelleeri­mist ja unifitseeritud modelleerimiskeelt UML. Rakenduste realisatsioon toimub peamiselt tabeliprogrammi Excel keskkonnas. Kasutatakse nii tabeliprogrammi valemeid, funktsioone ja käske, kui ka arendussüsteemi Visual Basic. Viimasel ajal kasutatakse programmeerimise ja algoritmimise algteadmiste ja oskuste kiiremaks omandamiseks graafilist programmeerimis­süsteemi Scratch. Majanduse ja keemia kursustes tegeletakse teatud määral ka dokumentide koostamisega ja esitluste loomisega.

## Rakenduste loomine tabeliprogrammide vahenditega

Oluline osa informaatikakursustes on kuulunud ja kuulub rakenduste loomisele tabeli­program­mide abil. Sellega süvendatakse ja laiendatakse infotöö oskusi ja tutvustatakse arendustöö aluseid. Juba pikemat aega kasutatakse meil selleks MS Excelit. Tahaks rõhutada, et kursuses ei õpetada Excelit vaid rakenduste loomist tabeliprogrammide abil. Excel on lihtsalt üks võimalikest vahenditest. Olgu aga märgitud, et enimkasutatavad tabeliprogrammid on üsna sarnased ja omavad analoogilist funktsionaalsust. Omandades oskused töötamiseks ühe programmiga ei tekki erilisi teise kasutamisega. Olulisemad on erinevused aredusvahendite kasutamisel.

Tänapäeva tabeliprogrammid omavad rikkaliku hulka lihtsalt kasutatavaid vahendeid, mis võimaldavad ka ilma program­meerimi­seta luua mitteinformaati­kutel paindlike ja laialdase võimalustega rakendusi. Saab kasutada erinevat liiki andmeid (arvud, tekstid, aja- ja tõeväärtused, graafikaandmed: diagrammid, graafikud, joonised ja pildid) ja valemeid, suurt hulk väga erineva otstarbega funktsioone, vahendeid andmete korrastamiseks ning analüüsiks ja visualiseerimiseks. Saab realiseerida andmebaasidega sarnaseid seotud tabelitega . Siin kasutatakse teatud määral ka andmete modelleerimist klassi­diagrammidena. Tabeliprog­rammid ise kujutavad endast omavahel seotud objektide kogumeid ning nende struktuuri, omaduste ja tegevuste kaudu saab selgitada objektorienteerituid mo­del­lee­rimise ja mudelite esituse põhimõtteid. Näiteks on toodud, UMLi diagrammi abil esitatud, Exceli klassimudeli fragment.

Kursuse selles osas saadud teadmised vastavad põhilises Euroopa arvuti­kasutaja oskustunnistuse (ECDL) edasijõudnute tasemele tabeltöötluses.

Võttes aga kasutusele VBA, saab Exceli rakendustele lisada veel täiendavaid võimalusi.

## VBA kasutamine, programmeerimine objektorienteeritud keskkonnas

Arendustöö ja programmeerimise õpetamiseks kasutatakse kursuses Visual Basic’ul põhinevat arendussüsteemi VBA (Visual Basic for Application) MS Exceli keskkonnas. Olgu märgitud, et eesmärgiks ei ole Exceli programmeerimine vaid programmeerimise ja dokumendipõhiste rakenduste loomise õppimine Exceli keskkonnas. Sellise kombinatsiooni kasutamiseks võiks nimetada järgmisi põhjusi:

* VBA keelevahendite lihtsus, süsteemsus ja struktuursus ning nende ühilduvus üld-otstarbelise programmeerimissüsteemiga Visual Basic ja võrgupõhiste rakenduste arendussüsteemiga VBScript.
* VBA-d kasutatakse arendusvahendina paljudes teistes üldotstarbelistes rakendusprogrammides: Microsofti põhitooted, Corel Office, AutoCAD, Imagineer, Corel Draw jm. Mitmetes teistes programmides näiteks Open Office, Lotus Office kasutatakse VBAga sarnaseid vahendeid.
* Exceli kasutamisel tekkinud teadmised, kogemused ja oskused: avaldiste ja funktsioonide kasutamine, tingimuslike valemite loomine jm, aitavad oluliselt kaasa programmeerimise põhimõistete omandamisel.
* Excel pakub mugavat keskkonda VBA programmi poolt kasutatavate andmete salvestamiseks, kujundamiseks ja vormindamiseks ning kasutajaliidese loomiseks, mis vähendab oluliselt vajadust tegeleda sellega loodavas programmis.
* Võimalus kasutada Excelit ülesande lahendusprotsessi analüüsiks, algoritmide kavandamiseks ja selgitamiseks ning tulemuste kontrollimiseks ja analüüsiks.
* Võimalus luua kombineeritud rakendusi, kus üks osa töötlust tehakse Exceli vahenditega, teine VBA abil. See lubab õppimise käigus realiseerida tõsisemaid ja sisukamaid rakendusi ning praktiliselt käsitleda ka analüüsi ja disainiga seotud küsimusi.
* Võimalus käsitleda VBAs objektorienteeritud lähenemisviisi ja komponent­tehnoloogiat, ilma et õppija peaks ise tegelema objektide ja komponentide loomisega.

Kursuse selles osas on vaatluse all:

* objektide (Exceli töölehed, lahtrid ja lahtriplokid, graafilised kujundid, ohjurid jm) ning nende omaduste, meetodite ja sündmuste kasutamine
* andmete: konstandid, muutujad, massiivid ja tabelid
* väärtuste leidmine ja omistamine, avaldised ja funktsioonid
* protsesside juhtimine (jada, valik, kordused),
* mitmeprotseduurilised programmid, parameetrite ja argumentide ning ühisandmete kasutamine andmevahetuseks protseduuride vahel,
* algoritmide koostamine ja esitamine,

Algoritmide esitamiseks kasutatakse UMLi tegevusdiagramme ning nn pseudokoodi ehk algoritmikeelt. Kasutatavate vahendite näiteks on toodud nö „klassika“: ruutvõrrandi lahendamise algoritm.

|  |  |
| --- | --- |
|  |  |

### Mängude kasutamine õpetamisel

Juba pikemat aega on kursuses pööratud olulist tähelepanu graafikaobjektide kasutamisele ja nende abil mängutaoliste raken­duste, animatsioonid ja nö „multikate“ loomisele. Excel ja VBA pakuvad siin laialdasi ja lihtsaid võimalusi. Selle läbi saab kiiremini teha algajatele selgeks objektorienteeritud lähe­ne­mis­viisi olemuse ja program­meerimise põhitõed, sh protsesside juhtimine: valikud ja kordused jm.

Näiteks on toodud rakenduse „Labürint“ ehk „Juku koolis“ kasutajaliides Exceli töölehel. Liigutades Jukut nooleklahvidega peab võimalikult kiiresti läbima labürindi: rakendus mõõdab aega ja peab edetabelit. Teel on mitmeid takistusi: peab vastama erinevatele küsimustele ja lahendama mitmesuguseid ülesandeid, näiteks aritmeetika, tõlkimine ühest keellest teise ning muud viktoriini ja testi taolised ülesanded. Valede vastuste ja „keelatud“ aladele (tähistatud erinevate värvidega) sattumisel saab trahvi või lisaküsimusi. Õppija, koostades rakendust, saab suurel määral ise valida labürindi skeemi ja stsenaariumi ning küsimused ja ülesanded. Realisatsioonis võivad on kasutusel paljud programmeerimise elemendid: objektid ning nende omadused, meetodid ja sündmused, kordused ja valikud, erineva organisatsiooniga andmed, sh tabelid ja massiivid näiteks küsimuste ja vastuste hoidmiseks.

Ja veel üks mängutaolise rakenduse näide, mis ei ole eriti tüüpiline Exceli jaoks. Töölehel käib „jalgpallilahing“. Kraps (tegelane graafilisest programmeerimissüsteemist Scratch) imiteerib pealelööke: pall liigub juhusliku nurga all väärava suunas. Liigutades Jukut nooleklahvide abil, saab mängija kaitsta väravat). Rakendus peab statistikat ja kontrollib aega.

|  |  |
| --- | --- |
|  | **algoritm** Jalka **Algseaded** **korda kuni** aeg > 45 **Liigu** Kraps, pall  pall.suund = juhuarv (240..300) lööke = lööke + 1 **kordus** ' palli lendamine **Liigu** pall, hp ' pall teeb sammu  **paus** pp **kui** **On\_Puude**(pall, Juku) **siis**  käes = käes +1 **katkesta** kordus **kui** **On\_Sees**(pall, värav) **siis**  tabas = tabas + 1 **katkesta** kordus  **kui** mitte **On\_Sees**(pall, väljak) **siis**  **katkesta** kordus  **Liigu\_XY** pall, x, y, xp/2  **paus** 1...2 |

Näiteks on toodud rakenduse põhiosa algoritm. Sellest on näha kasutusel on kordused ja valikud ja muud programmide tüüpelemendid. Programmis on kasutusel mitmed protseduurid: Liigu( ), Liigu\_XY(), paus(), On\_Puude( ), On\_Sees( ). Tüüpiliselt on tegemist valmisprotseduuridega, mida õppijad kasutavad alguses „mustade kastidena“, teades ainult otstarvet ja argumentide olemust. Soovi korral saavad nad ka uurida nende sisu. Taoliste valmisprotseduuride kasutamine võimaldavad õppijatel kiiremini asuda sisukamate rakenduste koostamisel.

Taoliste meetodite ja vahendite kasutamine mitteinformaatikutele programmeerimise õpetamisel, igati õigustab end, tõstes õppijate huvi ja motivatsiooni. Alustades taolistest mängutaolistest asjadest, jõutakse ka oluliselt sisukamate rakendusteni: algoritmid tabelite, massiivide ja ankeetide töötlemiseks, funktsioonide uurimine jm. Näiteks on toodud kasutajaliides rakenduse jaoks, mis võimaldab teha antud funktsioonide ette­anta­val lõigul ning leida teatud karakteristi­kud: aritmeetili­sed keskmi­sed, integraalid jm.

|  |  |
| --- | --- |
|  |  |

## Uue põlvkonna vahendid programmeerimise õpetamiseks. Scratch

Viimastel aastatel on tekkinud ning kiiresi arenevad ja levivad uue põlvkonna programmeerimise õpetamise/õppimise keeled ja keskkonnad. Traditsiooniliselt neid nimetatakse programmeerimiskeelteks, kuid pigem võiks taolisi vahendeid nimetada programmeerimise keskkondadeks, sest neis on palju enamat, kui lihtsalt keel. Konkureerimiseks uute vahendi**t**ega, arendatakse ja modifi**t**seeritakse ka mitmeid olemasolevaid keeli ja süsteeme. Wikipeedias, jaotises „Educational programming language“, on nimetatud juba ligi 60 uut ja vanad taolist keelt.

Tegemist on suhteliselt lihtsa süntaksiga keeltega ning mugavate kasutajaliidestega keskkondadega, mis on eeskätt mõeldud programmeerimise õppimiseks/õpetamiseks algajatele, sh lastele ja noortele. Taolistes süsteemides on osutatud suurt tähelepanu atraktiivsusele ja multimeedia kasutamisele, võimalusele lihtsalt ja kiirelt luua mänge, animatsioone jmt. See võimaldab omandada programmeerimise olemuse, põhimõisted ja meetodid märgatavalt kiiremini sellest, kui alustada kohe mingi "suure", proffidele mõeldud, keelega (Pascal, C, PHP, Java jmt). Ka autoga sõitmise õppimist ei pea alustama Vormel I-ga, võib alustada ka kardist või vanast Fordist ning järjest liikuda võimsamate masinate poole.

Programmeerimise õpetamiseks mõeldud uutest keeltest ja keskkondadest võiks nimetada järgmisi: Alice, AgentSheets, Baltie, Greenfoot, Karel++, Phrogram ja Scratch. Nö renoveeritud keeltest võib märkida Logo ja Basicu viimaseid versioone.

Uue põlvkonna vahendite tekke põhjuseid ja arendamise eesmärke on analüüsitud näiteks Scratch’i arendusmeeskonna (MIT, Media Lab) poolt. Üheks peamiseks ajendiks oli neile see, et märgatavalt hakkas vähenema noorte huvi programmeerimise vastu. Traditsioonilised programmeerimisvahendid on vähe atraktiivsed ja laiemale kasutajate ringile üldiselt rasked (keeruline süntaks, andmetüübid jm), samal ajal Internet pakub noortele hulgaliselt huvitavat tegevust.

Uued vahendid ja keeled aga sobivad igas vanuses algajatele programmeerimise õpetamiseks. Neid kasutatakse koolide algklassidest kuni kolledžide ja ülikoolide programmeerimise sissejuhatavate kursusteni välja. Näiteks meil kasutusel oleva Scratchi, „avastasime“ Harvardi ülikooli (üks kõrgeima reitinguga ülikool maailmas) kursuses „Computer Science 50“, milles alustatakse Scratchiga. Berkeley ülikolis (kah esikümne ülikool) on terve semestri pikkune Scratchile põhinev kursus mitteinformaatikutele „Computer Science 10“: „The Beauty and Joy of Computing“. On veel mitmeid ülikoole, kus kasutatakse Scratchi, rääkimata koolidest.

Scratch on loodud Massachusetts'i Tehnoloogia Instituudis (MIT) spetsiaalselt programmeerimise õpetamise jaoks. Scratchi rakenduste keskseteks elementideks on graafikaobjektid ehk spraidid. Programmi käskude abil saab nendele rakendada mitmesuguseid tegevusi: muuta asukohta, suurust, värvust jms, panna need kõndima, tantsima, tekitama helisid, arvutama, joonistama jms. Süsteemiga tuleb kaasa suur valik spraite, aga võib kasutada ka oma loodud graafikaobjekte. Spraidil võib olla mitu kostüümi, mida programselt vahetades tekitatakse animatsiooniefekte: kõndimine, tantsimine vms. Spraidid tegutsevad ekraanil piirkonnas (aknas), mida nimetatakse lavaks.

Scratchi projektis võib olla kasutusel mitu programmiüksust – skripti. Skriptid moodustatakse hiire abil graafilistest käsuplokkidest. Skript kuulub alati ühele kindlale spraidile või lavale. Ühel spraidil võib olla suvaline hulk skripte. Skriptid saavad teha omavahel koostööd teadete abil.

Pildil toodud näites on kasutusel kolm spraiti: Juku (1 kostüüm, 1 skript), Kraps (2 kostüümi, 2 skripti) ja piiga (4 kostüümi, 1 skript). Programmis on viis muutujat: punkte, aeg, piir, a ja b. Kraps „jalutab“ edasi tagasi mööda lava, annab korrutamisülesandeid, hindab kasutaja vastuseid ja loendab õigeid vastuseid. Krapsu skriptid (algavad plokiga Läks) töötavad paralleelselt. Need käivitab (käsuga teavita Läks) Juku skript, millest algab programmi töö. Juku skripti lõpuosa töötab paralleelselt Krapsu skriptidega, kuvades taimeri abil Laval muutujas aeg jooksvat aega ja lõpetab programmi, kui aeg ületab etteantud piiri. Õige vastuse korral teeb piiga väikese tantsu, vahetades kostüüme (pildil on neist näidatud kaks); jõudnud viimaseni, alustatakse jälle esimesest.

Kirjeldatud näide kujutab endast ühte lihtsamat alguses tehtavatest harjutustest. Näiteks on toodud rakenduse kasutajaliides, mis loob funktsiooni graafiku, leiab ekstreemumite väärtused (min ja max), nende asukoha ja nullkohad. On toodud ka skript, mis leiab maksimaalse väärtuse ühemõõtmelises massiivis (vektoris).