Liet. matem. rink., T. 42, spec. nr., 229–234

Programavimo mokymas išplėstiniame informatikos kurse

Valentina DAGIENĖ (MII), Jonas BLONSKIS (KTU)
el. paštas: dagiene@ktl.mii.lt, jonasb@soften.ktu.lt

1. Įvadas

Bendrojo lavinimo mokyklos XI–XII klasių informatikos išplėstinio kurso mokymo programą sudaro trys moduliai, vienas kurių – „Programavimo pagrindai" [6, 3]. Jam skiriamos 68 valandos.

Informatika bendrojo lavinimo mokyklose dėstoma daugiau kaip penkiolika metų. Pradžioje beveik visą kursą sudarė algoritmavimo mokymas. Atsiradus mokykloje kompiuteriams, algoritmavimo dalis informatikoje mažėjo. Programavimo mokymas nebuvo aktyvus, nes tai sunkus, daug jėgų ir išmanymo reikalautis procesas. Tobulėjant kompiuteriams ir jų programinei įrangai, atsiradus galimybėms daug ką lengviau daryti su kompiuteriu, programavimui imta skirti vis mažiau dėmesio.

Dabar algoritmavimo mokoma privalomai tik pagrindinės mokyklos informatikos kurse [2]. Tam skiriama maždaug ketvirtadalis privalomo informatikos kurso valandų. Šiuo kursu siūloma supažindinti su pagrindinėmis algoritmavimo konstrukcijomis remiantis Paskalio arba Logo kalba.

Programavimo mokymą lemia informatikos mokytojų pasirengimas ir tinkamai sudaryta mokymo metodika. Pedagogus ruošiančios aukštosios mokyklos pernelyg mažai dėmesio skiria programavimo didaktiniams klausimams, todėl nuolat jaučiama jų nepakankama kvalifikacija.

Ar reikia mokyti moksleivius programuoti? Kaip ir kiek mokyti programuoti? Kam skirtas programavimo pagrindų modulis?

2. Programavimo mokymo problemos

7–8 dešimtmečiuose niekam nekilo klausimo, kas yra programuotojas, kas yra programavimas. Pirmosios ir antrosios kartos kompiuteriams programos buvo rašomos aštuntainiais–dvejetainiais kodais. Programavimo procesas buvo lėtas, programų apimtys nedidelės. Apie 90% laiko buvo gaištama programoms rašyti ir derinti. Algoritmai buvo nesudėtingi, taip pat ir duomenų struktūros. Programuotojo dėmesys buvo kreipiamas vien programos tekstui.

Šiuo metu programos teksto rašymas ir derinimas programuotojo darbe užima nedidelę darbo laiko sąnaudų dalį. Pagrindinis dėmesys skiriamas duomenų struktūroms

projektuoti ir algoritmams pritaikyti. Tačiau programuotojai nėra įvardijami algoritmuotojais. Taip pat taikiai baigiasi ginčai, kur yra riba tarp algoritmo ir programos.

Tinkamai suprojektuotos duomenų struktūros nulemia programos sudėtingumą, apimtį, patikimumą, mobilumą, darbo greitį, poreikį kompiuterio ištekliams. Norint suprojektuoti patogias duomenų struktūras, nepakanka žinoti, kaip jos kuriamos, bet būtina išmanyti algoritmus, skirtus darbui su duomenų struktūromis (kelties, paieškos, rūšiavimo, suliejimo, įterpimo, šalinimo algoritmai ir kt.).

Uždavinių sprendimo algoritmus (būdus, metodus) nagrinėja ir kuria atitinkamos mokslo sritys. Tačiau tie algoritmai pagal nuo seno esamą tvarką tik aprašo sprendimo eigą ir nurodo reikalingus atlikti veiksmus, jie skirti žmogui. Reikalinga tų algoritmų realizacija, įvertinant duomenų struktūras ir duomenų apimtį, kompiuterio galimybes, būsimos programos vartotojo poreikius. Tai labai sudėtingas ir kruopštus kūrybinis darbas, reikalaujantis daug žinių (ypač matematikos), nuovokumo, kantrybės. Nemažai programuotojų programų nerašo, o tik kuria algoritmus. Programų tekstus už juos rašo žemesnės kvalifikacijos programuotojai.

Dabartiniu metu egzistuoja programavimo priemonės, kurios automatizuoja, daro patogų ir greitą programų kūrimą. Programuotojas vis mažiau laiko skiria programų rašymui ir derinimui.

Procedūrinio programavimo technologija – jau praeitis. Objektinio programavimo technologija, nesuspėjusi sukurti jai tinkamą mokymo metodiką, persipina su komponentiniu programavimu. Čia tampa opus klausimas: kaip, kiek, ką, kokia seka pateikti pradedančiajam mokytis programuoti [7].

Istoriškai perėjimą iš procedūrinio į objektinį ir komponentinį programavimą atkartoti mokyme daugeliui yra priimtiniausias ir lengviausias kelias, tačiau besimokančiam tai dažniausiai netinkamas. Tikslinga mokyti tai, kas aktualu, negaišti laiko senoms technologijoms įsisavinti, tačiau tam dar nėra sukurtos geros mokymo metodikos. Kiekvienas programavimui skirtos mokomosios medžiagos autorius pristato, jo nuomone, tinkamą mokymo būdą. Dalis autorių siūlo atskirti programavimo kalbos mokymą nuo algoritmavimo ir duomenų struktūrų projektavimo. Tik įsisavinus kiekvienos dalies pradmenis, siūloma mokytis programuoti.

Yra siūlymų programavimo elementus nagrinėti visapusiškai, kai jungiančioji grandis būtų nedidelių realių uždavinių programų kūrimas [1]. Gal ir sudėtingesnis kelias, bet juo besimokančiam greičiau ir pilniau perteikiamos visos kuriamos programos problemos, mokinys greičius pajunta kuriamos programos galią kompiuteriui.

Programų kūrimo procese išskiriami tokie momentai:

- Duomenų analizė, t. y. programos vartotojo poreikio pradiniams duomenims ir rezultatams analizė. Sukuriamos duomenų struktūros įvertinant būsimos programos struktūrą. Modeliuojant uždavinio sprendimą įsitikinama duomenų struktūrų efektyvumu.
- Parenkami uždavinio sprendimo būdai, sukuriami algoritmai. Modeliuojant situacijas įsitikinama jų tinkamumu.
- Projektuojamas programos dizainas. Tai ne tik programos vartotojui matomi kompiuterio ekrane vaizdai, dialogas, pranešimai, bet ir programos struktūra. Objek-

tiniame programavime tai yra vieninga visuma, nes objektus aprašančios klasės jungia tarpusavyje duomenis ir veiksmus su tais duomenimis.

3. Ar reikia mokyti mokinius programuoti?

Galimi du pasirinkimai.

NE, nes:

- masinis kompiuterio vartotojas neprogramuoja;
- naujų programų įmonės, organizacijos nerašo. Jas kuria atitinkami „programų fabrikai" bei programavimo kolektyvai;
- Lietuvai profesionalių programuotojų reikia nedaug. TAIP, nes:
- Įvairių įstaigų kompiuterius ir jų tinklus aptarnaujantiems žmonėms programavimo žinios reikalingos;
- dažnai reikia modifikuoti taikomuosius programinius paketus pritaikant juos konkretiems organizacijos poreikiams;
- kiekvienas programinis paketas, sudėtingesnė taikomoji programa turi savo programavimo priemones, kurios būtinos norint profesionaliai pritaikyti paketą;
- masinis kompiuterių vartotojas sparčiau, lengviau, giliau suvokia vartojamą programą;.
- komponentinio programavimo priemonės leidžia sukurti greitai nedideliems poreikiams reikalingas programas.

Tikslinga informacinių technologijų vartotojui turėti sampratą apie duomenų struktūras, algoritmus, kompiuterio mastymo principus.

Todėl kiekvienas moksleivis turi gauti programavimo pradmenis, o galvojantis apie informatiko profesiją, turi stengtis kaip galint anksčiau išmokti programuoti.

4. Kiek ir kaip mokyti programuoti?

Programuoti galima išmokti tik rašant programas. Reikalingas specifinis mąstymo būdas, kuris pasireiškia rašant veikiančias, realias programas. Kokios jos turi būti? Kiek programų reikia parašyti, kad išmoktum? Praktika rodo, kad programavimo pagrindų mokymas yra sudėtingas procesas. Jis labai priklauso nuo pedagogo pasiruošimo, gebėjimo sudominti ir tinkamai parinkti užduotis mokinimas. Programavimo mokymas – individualus darbas, todėl mokytojas turi gebėti pagal mokinio galimybes paruošti užduotis.

Kompiuterio galimybės ir programavimo priemonės labai svarbios. Mokiniai greitai atskiria, ar jis mokomas senomis, ar šiuolaikinėmis priemonėmis. Tai formuoja mokinio požiūrį į programavimą.

Tobula procedūrinio programavimo mokymo priemonė buvo Paskalio kalba. *Turbo Pascal 7.0* buvo sukurtas DOS'ui ir jau visą dešimtmetį vis mažiau tenkinantis poreikius, nors ir galima kurti dinamines duomenų struktūras, susipažinti su objektiniu programavimu, grafika. Lyginant su dabartinėmis programavimo priemonėmis ir reikalavimais

programoms, *Turbo Pascal 7.0* tampa labai „nerangus". Algoritmams mokyti naudojamas *Free Pascal* [4].

Object Pascal realizacija *Delphi* terpėje iš esmės keičia ne tik programų kūrimo metodiką, bet ir požiūrį į patį programavimą [5]. Kadangi programa surenkama iš terpėje esančių komponentų, suteikiant jiems reikalingas savybes ir užrašant veiksmus, generuojami programos „rėmai", tai programuotojui lieka intelektualus darbas: algoritmai, duomenų struktūros, dizainas.

Senoji procedūrinio programavimo mokymo metodika šiuo atveju darosi netinkama. Objektinio programavimo mokymo būdai dar nebuvo nusistovėję, kuomet pasirodė komponentinio programavimo priemonės (*Delphi, C++Builder* ir kt.). Iš esmės tai objektinis programavimas, tačiau programų kūrimo eiga jau kita.

Reikia rasti minimaliai reikalingą teorinių ir praktinių veiksmų visumą, kurią įsisavinus, galima toliau tobulėti einant savo individualiu keliu.

Programavimo mokymui būtinas pasiruošimas:

- darbo kompiuteriu įgūdžiai;
- pažintis su programavimo terpe;
- programavimo kalbos elementai.

Darbo kompiuteriu įgūdžiai gaunami pradedant IX klase (jei yra galimybių – ir anksčiau). Kiti du punktai yra „Programavimo pagrindai" modulio sudėtinė dalis. Tam rekomenduojama skirti 5–8 val. Šios pamokos turi būti scenarijaus tipo, pagal kurį mokinys individualiu tempu konsultuojant mokytojui atlieka nurodytus žingsnius. Pagrindinis dėmesys skiriamas Delphi terpei „prisijaukinti", suprasti, kas yra programos projektas, kiek ir kokios bylos sudaro programą. Čia mokinys pažinčiai panaudoja tik kelis komponentus, išmoksta nustatyti jų savybes, parašo kelis veiksmus programavimo kalba.

Object Pascal mažai skiriasi nuo Turbo Paskalio kalbos (pagrindinės programavimo konstrukcijos pateikiamos 9–10 klasėje). Šiame etape mokinys prisimena *Window* lango struktūrą, susipažįsta su *Delphi* terpės kuriamos programos struktūra, programos rašymo elementais, komponentais ir jų savybėmis, komponento įvykiais, klase *TForm1* ir objektu *Form1*, priskyrimo sakiniu, sąlygos sakiniu, prisimena duomenų tipus (integer, real, boolean) ir susipažįsta su naujais (char, string).

Programų kūrimui reikalingas pasiruošimas:

- duomenų struktūros ir jų kūrimo būdai;
- algoritmai darbui su duomenimis tose struktūrose (keltis, paieška);
- uždavinių sprendimo metodai, nagrinėjami tiksliųjų mokslų dalykuose.

Šio mokymo etapo pagrindiniu tikslu yra programuotojui reikalingo mastymo būdo formavimas. Pradiniam apmokymui pakanka išmokti kurti pagrindines duomenų struktūras: masyvą ir įrašą (derinant šias dvi struktūras galima gauti norimą sudėtingą duomenų struktūrą), gebėti pasinaudoti tekstinėmis bylomis, žinoti klasės struktūrą (remiantis *Delphi TForm1* klase).

Pradžioje pakanka išmokti užpildyti savo veiksmais *Delphi* komponentų įvykių metodus (pagal užsakymą generuojamas procedūras). Algoritmai neturi būti sudėtingi. Reikia išmokyti duomenis skaityti iš bylos, rašyti į bylą, rezultatus pateikti ekrane. Skaičiavimai nesudėtingi. Jiems organizuoti pakanka sumavimo (sandaugos), didžiausios (mažiausios) reikšmių radimo, paieškos algoritmų.

Tolesnis etapas būtų – išmokti kurti savus metodus (procedūras ir funkcijas): susipažinti su formaliaisiais ir faktiniais parametrais, kreipiniais, vardų galiojimo sritimis.

Nereikia imti daug naujų komponentų, nes jiems įsisavinti gaištamas laikas. Vėliau, kai mokinys jau perpranta programavimo esmę, jis bus pajėgus savarankiškai panaudoti jam reikalingus naujus komponentus. Būtina patarti, kaip sukomponuoti programos langą, parenkant komponentų vietą, dydį, užrašus, spalvas.

Rekomenduotinas teorinių ir praktinių užsiėmimų santykis: 1:5. Praktiniai užsiėmimai privalo būti individualūs, atitinkantys mokinio galimybes.

Tam tikslinga skirti beveik pusę likusio moduliui laiko (apie 30 val.). Jeigu reikia, tai silpnesniems mokiniams gali būti skiriamas visas likęs laikas. Šiuo etapu baigiamas minimalus būtinas pasiruošimas.

Programų kūrimas – tai

- duomenų struktūrų pasirinkimas ir projektavimas;
- sudėtingesni darbo su duomenimis algoritmai (rūšiavimas, įterpimas, šalinimas);
- matematinių uždavinio sprendimo metodų realizacija.

Šiame etape svarbu suvokti duomenų struktūrinimo esmę, programos struktūrą, grupinio programavimo galimybę. Kiekvienas mokinys turėtų spręsdamas realų uždavinį atlikti duomenų analizę, sukurti reikalingas duomenų struktūras, struktūrizuoti uždavinio sprendimą, parinkti tinkamą algoritmą, suprojektuoti programos langus, suderinti reikalingus komponentus. Gabiems mokiniams galima pasiūlyti kurti savo klases ir objektus, pasidomėti kitomis duomenų struktūromis, panaudoti grafikos elementus gautų rezultatų pristatymui ekrane.

5. Išvados

1. Mokant programavimo pagrindų išskiriami trys etapai: pasiruošimo, bazinių žinių kaupimo ir programos kūrimo pagrindų mokymasis.
2. Programavimo mokymas yra individualus. Mokiniai pereina į kitą etapą, sukaupę tam reikalingas žinias ir įgūdžius.
3. Informatikos mokytojas dirba su kiekvienu mokiniu individualiai pateikdamas užduotis ir konsultuodamas.
4. Programavimo elementai (programavimo kalba, duomenų struktūros, algoritmavimas, programos struktūrizavimas, programos dizainas, programavimo terpė) mokomi kompleksiškai juos taikant sprendžiant kiek galima realesnius uždaviniuose.
5. Pradinės programavimo žinios, pagrindinės konstrukcijos išmokomos naudojantis kuo paprastesne programavimo terpe (Logo, Free Paskaliu).

6. Programoms kurti patariama naudotis modernesne technologija (pavyzdžiui, Delphi). Įpratus dirbti su Delphi klasėmis ir objektais, galima pradėti kurti savo klases ir objektus.

Literatūra

[1] J. Blonskis, V. Dagienė, *Programavimo pradmenys*, Vilnius, TEV (2001).
[2] V. Dagienė, Privalomas informatikos kursas bendrojo lavinimo mokykloje, *Informacijos mokslai*, **8**, 20–36 (1998).
[3] V. Dagienė, Profilinės vidurinės mokyklos informatikos mokymo problematika, *Informacijos mokslai*, **14**, 36–42 (2000).
[4] *Free Pascal* interneto svetainė, `http://www.freepascal.org`
[5] D. Intersimone, *Object-Oriented Extensions to Pascal*, Kompiuterinis leidinys (1993), `http://aldona.mii.lt/pms/fps/download/oopascal.txt`
[6] *Lietuvos bendrojo lavinimo mokyklos bendrosios programos ir bendrojo išsilavinimo standartai, XI-XII klasės*, Patvirtinta Lietuvos Respublikos Švietimo ir mokslo ministro 2002 m. rugpjūčio 21 d. įsakymu Nr. 1465, `http://www.pedagogika.lt/bps.htm`
[7] C. Sculte, Towards a pedagogical framework for teaching programming and object-oriented modelling in secondary education, *Open IFIP-GI-Conference on Social, Ethical and Cognitive Issues of Informatics and ICT*, Dortmund (2002), pp. 64–65.

Teaching of programming in advanced course of informatics

V. Dagienė, J. Blonskis

Teaching of programming in secondary school raises some problems: how to teach programming, which language and environment to choose, what kinds of tasks to prepare, etc. The paper deals with these questions. The objective and component programming are briefly described as the most suitable for developing programming skills of students. The stages in teaching of programming are analyzed: basic computer skills, basic knowledge in programming, and developing of programs.