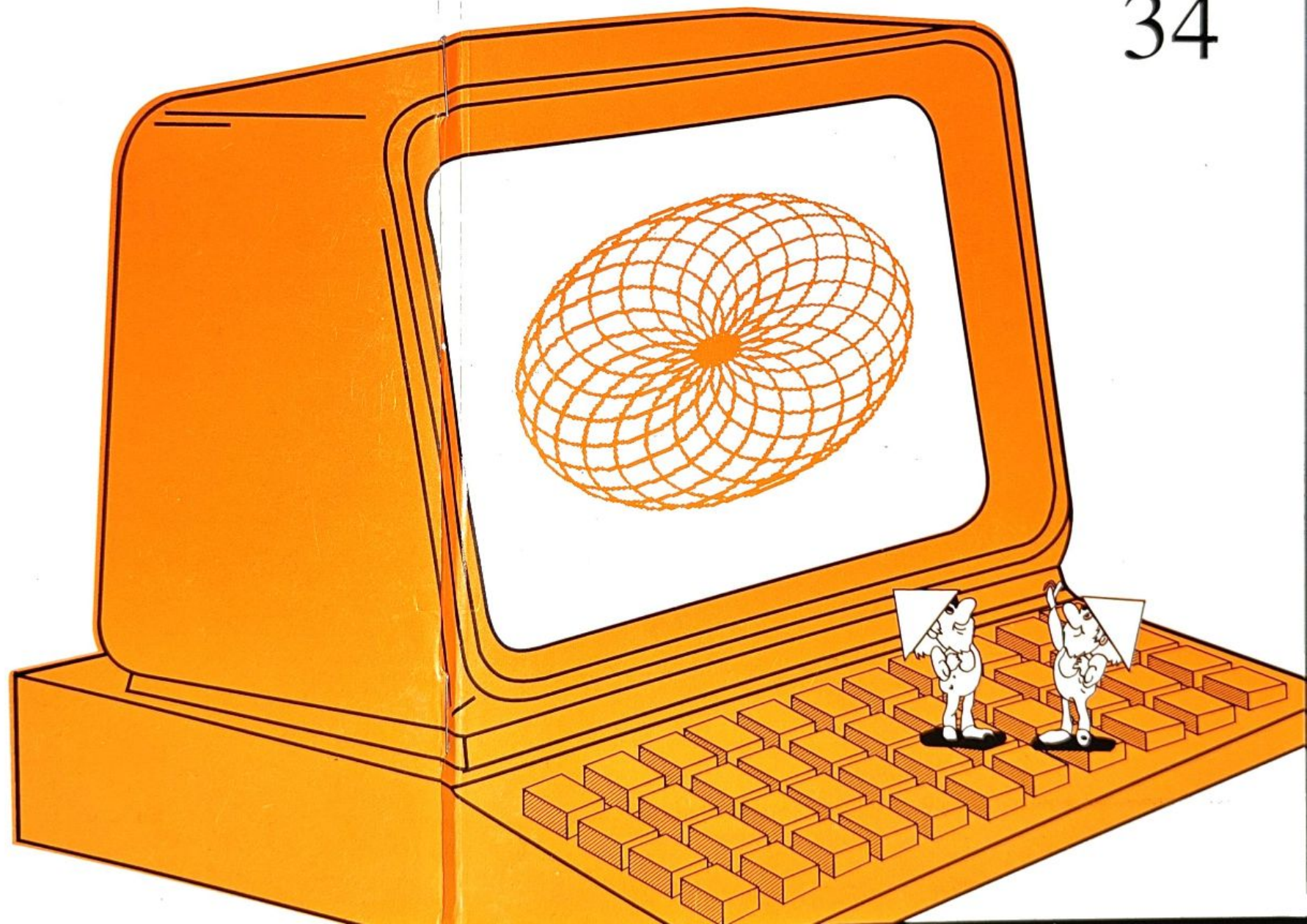


INFORMATIKA

INFORMACINIS-METODINIS LEIDINYS

34



MATEMATIKOS IR INFORMATIKOS INSTITUTAS

L. M. Markuševič

INFORMATIKA

Nr. 34

Vilnius – 1998

Šį numerį parengė

Irena Baradinskienė
Valentina Dagienė
Viktoras Dagys
Gintautas Grigas
Ramutė Žandarienė
Aidas Žandaris

Mūsų adresas:

„Informatika“
Matematikos ir informatikos institutas,
Akademijos g. 4,
2600 Vilnius

© „Informatika“, 1998

KOMPIUTERIZUOTAS MOKYMAS IR INTELEKTUALIOS MOKYMO SISTEMOS

Lina Markauskaitė

Kalbant apie kompiuterio taikymo mokymui kryptis, dažniausiai paminimos dvi svarbiausios: informatikos mokymas ir kompiuterizuotas mokymas [1, 12]. Ir viena, ir kita atsirado siekiant skirtingų tikslų.

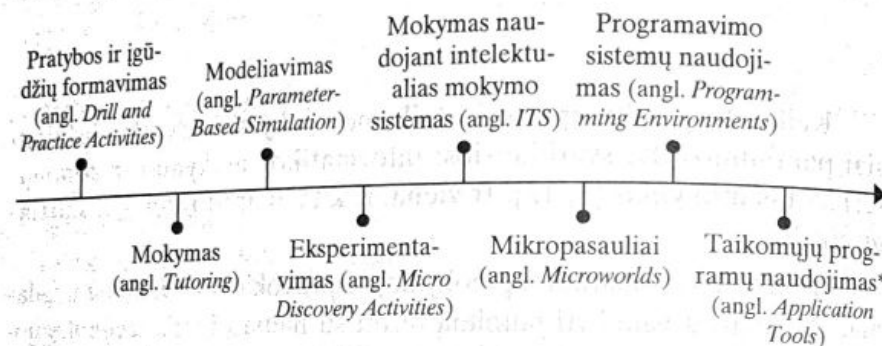
Informatikos atsiradimą mokykloje išprovokavo visuomenė teigdama, kad svarbu išmokyti jaunimą dirbti su naujausiomis technologijomis, ugdyti informacinę kultūrą ir panašiai: t. y. suformuoti informacinės veiklos įgūdžius, kurių prireiks darbinėje ir visuomeninėje veikloje [2].

Naudoti kompiuterius mokymui skatina svarbūs pedagoginiai tikslai [18, 87]. XX a. švietimui yra aktualu:

- individualizuoti mokymą: mokymosi ir mokymo turinys bei tempas turi atitikti individualius besimokančiojo poreikius (ši nuostata vylauja daugelyje šiuolaikinių psichologijos bei pedagogikos mokymo teorijų [14; 21]);
- sudaryti eksperimentinio mokymosi (angl. *learning by doing*) galimybes: konstruojant bei išradinėjant įgyti kritinio mąstymo patirties (ši nuostata ryški humanistinės krypties [14, 293–336] bei pragmatizmo [14, 158–207; 16] filosofijoje, kognityvinėje psichologijoje [21, 56–65]);
- formuoti besimokančiojo individualaus pažinimo, konceptualaus mąstymo įgūdžius, skatinti jo aktyvumą ir savarankiško pažinimo gebėjimą (technologijų naudojimo privalumus siekiant šių tikslų pagrindė S. Papertas) [15].

Kompiuterizuoto mokymo būdai ir mokomųjų kompiuterinių programų tipai

Naudoti kompiuterį mokymui galima skirtingiems tikslams ir skirtingais būdais. D. L. Fergusonas išsamiai pateikia kompiuterizuoto mokymo būdų įvairovę ir jų klasifikaciją pagal besimokančiojo aktyvumo lygmenį (1 pav.) [6, 34–40].



1 pav. Kompiuterizuoto mokymo būdai pagal besimokančiojo aktyvumo lygmenį

Kiekvienas kompiuterizuoto mokymo būdas pasižymi tik jam būdingomis savybėmis. Be to, kiekvieną mokymo būdą atitinka vis kitoks kompiuterinės mokomosios programos tipas. Ne visuomet įmanoma išvėlgti įvairių tipų programų akivaizdžius skirtumus, nes daugelyje jų iš karto taikomi keli mokymo būdai. Pagal tai, kas kontroliuoja mokymosi eigą (programa ar moksleivis), programos skirstomos į vadovaujančias (angl. *directed*) bei tyrinėjimo (angl. *exploratory*).

Vadovaujančiosios programos – pratybų ir įgūdžių formavimas bei mokymas – konkrečiai apibrėžia tikslą, parenka mokymo būdą, pateikia reikiamų žinių ir įtvirtina įgūdžius. Tokio tipo mokymo programų šerdis – kartojimas. Čia du didaktiniai principai: siekiama išmokyti reikiamų taisyklių bei jų taikymo; daug kartų atliekant panašaus tipo ne-

* Prie taikomojo tipo programų priskirčiau ir įvairias informacijos programas – kompiuterines knygas, žinytus, žodynus, enciklopedijas ir t. t. – bei „Internet“ pateiktą medžiagą. Mokymui ypatingai svarbios daugialypės informacijos sistemos, hipertekstai ir hipermedija.

sudėtingas užduotis, suformuojami reikiami įgūdžiai, perprantami bendrieji principai.

Moksleivis, naudodamas tyrinėjimo programas – nuo modeliavimo iki taikomųjų (žr. 1 pav.) – pats gali pasirinkti mokymosi tikslus bei eigą. Šių programų branduolys – efektyvus grįžtamasis ryšys. Įvairūs mokymosi sistemoje esantys objektai (dažnai tai būna įvairių daiktų ar reiškinių modeliai) pasižymi tam tikromis, žinomomis savybėmis ir valdomi iš anksto nusakytais būdais. Atlikus bet kokius veiksmus (pavyzdžiui, pakeitus jų charakteristikas), objektai tuoj pat reaguoja į tai. Pagal gautus rezultatus pats moksleivis nusprendžia, kaip tęsti studijas. Šios programos pasikliauja moksleivio aktyvumu, sugebėjimu pačiam kelti ir siekti mokymosi tikslų.

Kiekvienas kompiuterizuoto mokymo būdas grindžiamas pasirinktąja ugdymo filosofija bei mokymo psichologija ir yra efektyvus siekiant tik kai kurių mokymosi tikslų. O mokymo būdą atitinkanti programa vertinama pagal jai keliamus reikalavimus, yra jai būdingos sandaros, kuriama naudojant tinkamus programavimo metodus.

Kompiuterizuoto mokymo sąsajos su XX amžiaus švietimo filosofijomis

Kompiuterizuotas mokymas pagrįstas įvairių ugdymo filosofijų bei psichologijų teorijomis. Jose atsispindi mokymo samprata, turinys, tikslai, jų siekimo būdai. Ugdymo programos, vadovėliai, kompiuteris, mokomosios programos – tai tik priemonės, padedančios įgyvendinti šiuos švietimo filosofijoje keliamus tikslus. Kompiuterio ir mokomųjų programų taikymo efektyvumą galima įvertinti tik pagal tai, kaip jos padeda įgyvendinti įvairius pedagoginius sumanymus, t. y., kaip atitinka pasirinktus ugdymo tikslus. Kad lengviau būtų vertinti mokymo priemonių efektyvumą, apžvelgsiu šiuo metu vyraujančių švietimo filosofijų pagrindinius bruožus. Po to pagal šiuo metu švietime vyraujančius požiūrius į mokymą pamėginsiu apibrėžti efektyvaus mokymo ir efektyvios mokymo aplinkos sąvokas.

Pagal svarbiausius švietimo objektus (žinios, visuomenė ar vaikas) visas XX amžiaus švietimo filosofijas galima suskirstyti į tris grupes [16].

Žinių ugdymui daugiausia dėmesio skiriančių filosofijų (perenializmo ir esencializmo) pagrindinis tikslas – išmokyti vieno ar kito dalyko kartojant mokytojo pateiktą informaciją, pateikti svarbiausių žinių, mokėjimų ir įgūdžių, būtinų sėkmingam buvimui visuomenėje. Mokykla turi duoti darnią žinių sistemą, mokytojui pateikiant svarbiausias idėjas ir vertybes. Šiose filosofijose atsispindi švietimo lygybė: visiems – tas pats turinys, vienoda mokymo kokybė. Vienoje iš jų – esencializme – gamtos, tikslieji mokslai bei informatika yra prioritetiniai dalykai. Su ugdymo filosofijomis, žinioms teikiančiomis ypatingą reikšmę, susijęs ir ugdymo filosofijoje sukurtas programuotas mokymas. Kompiuteriu mokiniui pateikiama dalyko medžiaga. Naudodamas pratybų ir įgūdžių formavimo programas, mokinys įvaldo išmoktą teoriją, testų programomis patikrinamos jo žinios. Beveik visose vadovaujančio tipo mokymo programose yra realizuojamos programuoto mokymo idėjos. Nemažai jų galima įžvelgti ir intelektualiose mokymo sistemose.

Kita švietimo filosofijų grupė – progresyvizmas bei rekonstruktyvizmas – atskaitos tašku laiko *visuomenę*. Šios pedagoginės krypties atstovai mano, kad pasaulis nuolat keičiasi. Taigi ir mokymasis – tai nuolatinis procesas. Mokyklos paskirtis – išmokyti mokyti ir ugdyti gebėjimą spręsti problemas, o ne suteikti kurio nors dalyko žinių. Pripažįstamos konstruktyvios pedagogikos ir kognityvinės psichologijos idėjos: atsižvelgiama į mokinių intelektualines skirtybes, propaguojamas integruotas mokymas, aktyvieji mokymo metodai. Formuojant mokymo turinį, daug dėmesio skiriama socialiniams bei technologijos pokyčiams. Informacijos valdymo įgūdžių bei atsakomybės už informaciją ugdymas yra vienas iš svarbiausių švietimo uždavinių. Kompiuteris, be abejo, yra pagrindinė priemonė siekiant šių tikslų. Be to, jis gali talkinti įgyvendinant konstruktyvaus pažinimo bei mokymo integralumo idėjas. Čia puikiai tinka daug mokomųjų programų: taikomieji paketai, programavimo sistemos, mikropasauliai bei kai kurios intelektualiosios mokymo sistemos, modeliavimo ir eksperimentavimo programos. Konstruktyvaus mokymosi idėjos geriausiai realizuotos mikropasaulyuose bei hipertekstinės struktūros informacijos šaltiniuose.

Trečiosios ugdymo filosofijų grupės – egzistencialistų bei humanistų – dėmesio centre yra *vaikas*. Jų pagrindinis tikslas – ugdyti saviraišką, savivoką, o ne akademinius gebėjimus. Šios krypties atstovai atsisako dalykinės mokymo sistemos, propaguoja individualų mokymąsi. Deja, jų idėjas sunku suderinti su dabartiniu visuomenės gyvenimu.

Nauja švietimo filosofijos kryptis – holizmas – atsirado humanistinių ir visuomeninių idėjų sanglaudoje. Jos tikslas – vaiko kritiško ir kūrybiško mąstymo ugdymas, jo unikalio gebėjimų atskleidimas, pasaulėžiūros vientisumo formavimas. Integralumas yra vienas iš svarbiausių mokymo turinio bruožų. Nors holizmas išskirtinio dėmesio informacijos technologijai ir neskiria, bet kompiuterio, kaip efektyvios priemonės konstruktyviam mokymuisi, svarba abejoti netenka.

Taigi galima manyti, kad nė vienos šalies švietimo sistema nėra pagrįsta kuria nors viena ugdymo filosofija. Šiuo metu švietime vyrauja konstruktyvaus mokymosi idėjos, labiausiai būdingos švietimo filosofijoms, kurių pagrindas yra visuomenė, bet šių nuostatų atspindžių galima įžvelgti ir kituose požiūriuose į ugdymą.

Efektyvaus mokymo ir efektyvios kompiuterinės mokymo priemonės samprata

Efektyvaus mokymo sampratą apibūdinsime pagal šiuo metu vyraujančias ugdymo nuostatas. Nekelia abejonių teiginys, kad gera (efektyvi) kompiuterinė mokymo programa yra ta, kuri padeda mokyti efektyviau. Vertinant būtina prisiminti, kad ji yra tik pagalbinė priemonė efektyvesniam mokymui ir visiškai nebūtina, kad ji būtų lemiamas faktorius. Taigi gera mokymo programa turėtų padėti sukurti aplinką, kurioje mokymas būtų efektyvus.

Efektyviam mokymui, paremtam konstruktyvizmo idėjomis, būdingi šeši bruožai [4, 98–100].

1. Mokymasis yra konstruktyvus. Besimokantis asmuo nėra pasyvus informacijos gavėjas, jis pats formuoja savo supratimą, žinias ir įgūdžius.
2. Mokymasis nuolatos plečiasi. Žmogus konstruoja savo žinias: nauja informacija bei patyrimu jis papildo seną žinių sistemą, perpranta jų esmę, suteikia visoms savo žinioms naują prasmę.

3. Mokymasis yra metapažintinis procesas: nuolatos turi vykti savęs, kaip besimokančiojo, vertinimas. Moksleivis turi sugebėti pats vertinti savo žinias, rasti jų spragas, tinkamai pakreipti mokymąsi.
4. Mokymasis tuomet yra prasmingas, jeigu jo tikslą suvokia pats besimokantysis.
5. Mokomasi natūralioje (įvairialypėje) aplinkoje, t. y. įvairių socialinių ir fizinių procesų aplinkoje, kurioje ateityje reikės taikyti savo žinias.
6. Mokomasi bendraujant. Besimokant formuojami ir bendruomeninio gyvenimo įgūdžiai.

Kompiuterinių mokymo programų vertinimas

Kompiuterinės mokymo priemonės vertinimas – aktuali ir sudėtinga problema. Bet kokio tipo mokomojoje programoje svarbūs du jos aspektai: techninis ir ugdymo.

Mokomąją kompiuterinę programą visų pirma reikia įvertinti kaip programavimo produktą. Ji turi būti korektiška, be dalykinių bei programavimo klaidų, tvarkingai naudojanti kompiuterio resursus, būti ergonomiška, patikima, suderinta su kitomis programomis, lengvai įdiegiama, generiška – t. y. turi teisingai veikti (išorinis teisingumas) ir būti korektiškai parašyta (vidinis teisingumas).

Ar programa tinkama ugdyti moksleivio intelektą, įvertinti daug sudėtingiau. Priemonės autoriai ir jos vartotojai labai dažnai skirtingai suvokia programos paskirtį, todėl skirtingai vertina jos efektyvumą. Kiekvienoje mokymo priemonėje visų pirma atsispindi požiūris į šį ugdymą (jo tikslai, uždaviniai, mokymo turinys ir metodai) bei efektyvaus mokymo sampratą. Formaliai įvertinti šias priemonės savybes yra sudėtinga: vertinimo kriterijai nėra aiškūs. Norint tai išsiaiškinti, dažnai naudojamos kompiuterinių mokymo priemonių vertinimo anketos (angl. *checklists*) [20]. Čia pateikiama klausimų apie mokomosios priemonės savybes: programos struktūrą, dėstomąją medžiagą bei mokymo metodiką (pritaikomumą dirbti pamokoje ir mokytis savarankiškai) [10]. Aišku, toks vertinimo būdas atskleidžia tik kai kurias moky-

mo priemonės savybes ir neatspindi subjektyvaus požiūrio į ugdymą, kuris dažnai yra itin svarbus.

Intelektualių mokymo sistemų samprata, tikslai ir savybės

Maždaug 1970 metais dirbtinio intelekto (angl. *artificial intelligence*) idėjos buvo pradėtos taikyti kuriant kompiuterines mokymo programas – intelektualias mokymo sistemas. Pagrindinis šių sistemų tikslas – individualizuoti mokymą. Jos turi pasižymėti geromis dėstomojo dalyko žiniomis, efektyvia didaktika, našumu, intelektualumu. Intelektualios mokymo sistemos kuriamos pagal „gero mokytojo“ modelį, kuris [6, 36]:

- gerai išmano dėstomą dalyką,
- tinkamai įvertina besimokančiojo žinias bei gebėjimus,
- naudoja įvairius mokymo metodus, juos parenka pagal besimokančiojo sugebėjimus, žinias bei poreikius,
- įdomiai pateikia dėstomą dalyką bei užduotis, sudomina besimokantįjį, skatina mokytis toliau.

Mokymo sistemos, sukurtos pagal „gero mokytojo“ modelį (nepriklausomai nuo to, ar tiksliai jį atspindi), yra vadinamos intelektualiomis. Intelektuali mokymo sistema turi kurias nors iš toliau pateiktų savybių [6, 36–37; 3, 104–105].

1. Sistema moko spręsti netrivialias problemas (jų sprendimo algoritmas nėra paprastas). Pavyzdžiui, siūloma spręsti žodinius matematikos uždavinius, įrodyti geometrijos teoremas, prastinti logikos reiškinius, taikyti skyrybos taisykles.
2. Bendravimo su sistema kalba yra artimesnė įprastai žmonių kalbai. Kitoms mokymo programoms, neintelektualioms mokymo sistemoms būdingas labiau formalizuotas ir trafaretiškas bendravimas (įvedimas bei informacijos pateikimas).
3. Intelektuali sistema mokymosi eigą valdo dinamiškai, t. y. pritaikius taisykles, gaunami rezultatai, galintys turėti įtakos tolimesniems sistemos veiksmams.

4. Intelektuali mokymo sistema gali pati papildyti savo žinių bei taisyklių bazę.
5. Intelektuali sistema naudoja mokymo metodus, imituojančius „gerą mokytoją“. Pavyzdžiui, ji nustato, ko yra nesuvokęs ar ką neteisingai supratęs besimokantysis, ji tinkamai nurodo tolesnį mokymosi etapą. Apskritai šiose mokymo sistemose daug dėmesio skiriama „mokinio modeliui“ sukurti: įvertinti besimokančiojo žinias bei gebėjimus, pagal juos parinkti mokymo metodą, vertinti atsakymus. Intelektuali mokymo sistema turi suprasti kiekvieną moksleivio pateiktą teisingą, nors ir ne optimaliausią, sprendimą.
6. Intelektuali mokymo sistema naudoja labai daug įvairių grafinių (grafikai, diagramos, schemas ir pan.) bei simbolinių (natūralios bei formalios kalbos) informacijos pateikimo būdų, pagrindžia bei pakoментuoja pateiktą informaciją.
7. Intelektuali mokymo sistema gali paaiškinti ir argumentuoti pateiktą informaciją. Besimokantysis gali paklausti apie kiekvieną sistemos žingsnį, atliktą siekiant rezultato. Mokymo sistema turi atsakyti tiksliai ir aiškiai.
8. Intelektualios mokymo sistemos naudojami problemų sprendimo metodai bei mokymo didaktika lengvai pakeičiama.

Sistema, pasižyminti visomis šiomis savybėmis, yra idealus intelektualios mokymo sistemos modelis. Daugelyje sistemų yra realizuoti tik keli iš išvardintų reikalavimų. Apie svarbiausias intelektualių sistemų savybes jų kūrėjų nuomonės yra skirtingos [3, 104; 6, 36–37; 17, 200]: kiekvienas jų savaip interpretuoja „gero mokytojo“ sąvoką. Visi sutinka, kad intelektualiai mokymo sistema turi: (1) suteikti žinių, (2) įvertinti moksleivio gebėjimus, (3) parinkti tinkamiausią mokymo metodą.

Vieną iš intelektualių mokymo sistemų savybių sąlygojo jų eksperimentinė prigimtis: ji turi *suteikti moksleiviui žinių* (šis tikslas daugelio tokių sistemų kūrėjų buvo laikomas pagrindiniu). Moksleivis, naudodamas tokią mokymo sistemą, turi tapti tam tikros srities ekspertu (tokio paties lygio, kaip ir jo mokytojas) [3, 104]. Aišku, diskutuojama dėl beato-dairiško šio tikslo siekimo, tai dažnai ir kritikuojama: moksleivių gebė-

jimai bei žinios yra nevienodi, ne kiekvienas gali suprasti visus mokytojo sprendimus. Gera mokymo sistema turi *įvertinti moksleivio gebėjimus*, pateikti suprantamai informaciją (teoriją, konkretūs ar abstraktūs pavyzdžiai, analogijos ir pan.), t. y. pagal moksleivio gebėjimus *parinkti tinkamą mokymo metodą*.

Intelektualių mokymo sistemų įvairovė

Intelektualias mokymo sistemas (angl. *Intelligent Tutoring System*) daugelis autorių apibūdina vienodai [3; 8; 11]). Jų apibrėžimuose minimos ankstesniame skyrelyje išvardintos savybės.

Intelektualioms mokymo sistemoms apibūdinti literatūroje vartojami ir kiti terminai: angl. *Intelligent Instruction System* [13], *Explanation-Based Environment* [3], *Artificial Intelligence-Based System* [8].

Visi šie terminai nusako intelektualių mokymo sistemų poaibį, t. y. programas, kuriose kažkurios intelektualios mokymo sistemos savybės yra ryškesnės nei kitos.

Nuo įprastųjų intelektualių mokymo programų skiriasi intelektualios mokymo terpės (angl. *Intelligent Learning Environments* – ILE [12], *Intelligent Cooperative Systems* – ICS [19], *Computer-Supported Intentional Learning Environments* – CSILE [19]). Tai mišraus tipo mokomosios programos, kuriose mėginama sujungti geriausias mikropasaulių ir intelektualių mokymo sistemų savybes. Jose ir moksleivis, ir programa turėtų elgtis kaip lygiaverčiai, vienas kitam padedantys partneriai: drauge pasirinkti mokymosi eigą, įgyti naujų žinių, spręsti problemas. Kurio tokio tipo mokymosi terpes pradėta apie 1990 metus [9, 187] ir konkrečių rezultatų kol kas yra vos keletas.

Intelektualių mokymo sistemų sandara

Intelektualios mokymo sistemos susideda iš keturių pagrindinių modulių [3, 98–99; 18, 88–89]: žinių, moksleivio, mokytojo bei sistemos ir moksleivio sąsajos (1 lentelė).

1 lentelė. Intelektualių mokymo sistemų pagrindiniai moduliai ir jų funkcijos

Modulis	Funkcijos
Žinių (angl. domain arba expert domain-knowledge)	Mokomojo dalyko žinios Sistemos žinių bazė: faktai, apibrėžimai, aksiomos, taisyklės, užduotys ir pan. Ekspertinė programa.
Moksleivio (angl. learner arba student)	Moksleivio modelio sukūrimas Moksleivio suvokimo, mąstymo pobūdžio, žinių bei jų trūkumų diagnozė ir vertinimas. Naujai įgytų žinių bei įgūdžių dinamiškas nustatymas.
Mokytojo (pedagogo) (angl. tutoring arba pedagogue)	Didaktinių programos aspektų kontrolė ir valdymas Moksleivio ir mokytojo (programos) sąveikos modeliavimas. Tinkamų užduočių parinkimas ir pateikimas. Moksleivio žinių pedagoginis vertinimas, reikiamos mokytojo (programos) pagalbos pateikimas. Mokymas: paaiškinimai, patarimai, žinių patikrinimas ir pan.
Sąsajos (angl. user-interface)	Sistemos sąsaja su mokiniu Programos sąsaja: valdymas, kalba, grafika, informacijos pateikimas ir pan.

Intelektualios mokymo sistemos sudėtis atspindi esmines jos savybes. Žinių, moksleivio ir mokytojo dalys atitinka kompiuterizuoto mokymo priemonių kūrimo paradigmą „ko, kam, kaip“ (angl. „What, Whom, and How“) [3, 98].

Žinių modulyje (ko yra mokoma) pateikiamas visas mokymosi turinys: faktai, taisyklės ir pan. Kuriant šį modulį daug dėmesio skiriama efektyviam žinių vaizdavimui kompiuteryje. Jame taip pat turi būti taisyklės, aprašančios tos pačios problemos alternatyvius sprendimo būdus, generuojančios skirtingo tipo pavyzdžius ir pan.

Moksleivio modulis (*kam* skirta sistema) – tai tikslus besimokančiojo žinių įvertinimas: ką jis žino bei sugeba ir ko ne, ką žino netiksliai ar neteisingai. Moksleivio žinių vertinimas sistemoje saugomas vienu iš dviejų būdų: kaip statinė duomenų struktūra, kurioje yra informacija apie besimokančiojo įsisavintus dalykus (angl. *an overlay model*), arba kaip procedūrų rinkinys, kuris atspindi moksleivio darbą su programa ir žinias (angl. *a buggy model*).

Mokytojo modulis (*kaip* vyksta mokymas) pagal moksleivio moduli pateiktą žinių bei įgūdžių nustatymą kiekvienu metu dinamiškai parenka tokį mokymo metodą, informaciją, jos pateikimo būdą, pavyzdžius bei užduotis, kad mokymasis būtų kuo efektyvesnis. Mokytojo modulį dažniausiai sudaro metažinių ir taisyklių rinkinys, realizuojantis kompiuteryje bendrosios ir specialiosios didaktikos principus.

Intelektualios mokymo sistemos kūrimas susideda iš keturių etapų. Pirmiausia sukuriamas ekspertinis žinių modulis, vėliau – moksleivio. Antrasis etapas labai priklauso nuo pirmojo. Moksleivis dažnai yra modeliuojamas kaip žinių moduli poaibis. Trečiajame etape, remiantis sukurta sistemos žinių baze ir mokinio vertinimo principais, kuriamas mokytojo modulis. Jame įgyvendinami pedagoginiai sumanymai. Pabaigoje kuriama ergonomiška sistemos sąsaja su moksleiviu.

Intelektualių mokymo sistemų pedagoginis ir psichologinis pagrindimas

Daugelis specialistų mano, kad sudėtingiausia yra tinkamai parengti intelektualios mokymo sistemos mokytojo modulį [8, 3]. Nėra vieningos nuomonės, kokius įgūdžius šios sistemos turi išugdyti ir kokie mokymo metodai tam turi būti naudojami. Vienose intelektualiose mokymo sistemose akcentuojamas tikslas išmokyti konkretaus dalyko žinių, kitose – ugdyti intelekto gebėjimus, išmokyti konceptualiai mąstyti, priimti sprendimus. Be to, intelektualių sistemų kūrėjai pabrėžia skirtumą tarp mokymo (angl. *instructional theory*) ir mokymosi teorijų (angl. *learning theory*). Mokymasis turi vykti tam tikroje įvairialypėje aplinkoje, kurioje priklausomai nuo daugelio mokymosi veiksnių turi būti naudojami tinkami mo-

kymo metodai. Mokymas – mokymosi sudėtinė dalis. Tinkamas mokymo metodas yra parenkamas atsižvelgus į tikslą [8, 4].

Tuo, kad intelektualiai mokymo sistema yra tobula programuoto mokymo forma, abejoti netenka, nes ji visiškai atitinka B. F. Skinnerio aprašytą programuoto mokymo sistemą [7, 379]: kiekvienam mokiniui yra pateikiamas (a) mokomojo dalyko turinio elementas, (b) reikalaujama, kad jis aktyviai į tai reaguotų (atsakytų į klausimą ar išspręstų užduotį), (c) mokinys tuoj pat yra informuojamas, ar teisingai atsakė. Programuotas mokymas vyksta pakopomis – nuo pradinio lygio iki pakeidaujamo lygio (išmokimo), kiekvienoje naujoje pakopoje atsižvelgiama į ankstesnę. Dabar programuoto mokymo idėjos ir metodai kritikuojami visuotinai [1, 17–18], nes visiškai neatitinka šiuolaikinių švietimo filosofijų: „<...> nuobodus, <...> nėra naudingas, kai dėmesys kreipiamas į nuostatus ir vertybes“ [7, 380] ir pan. Tačiau pedagogai pripažįsta, kad programuotas mokymas gali būti naudingas „mokantis kai kurių įgūdžių <...>, teikiant mokiniui pagalbą, papildomai mokantis“ [7, 380], bet visam ugdymui jis nėra tinkamas.

Nors daugelyje intelektualių mokymo sistemų yra realizuotos programuoto mokymo idėjos, bet čia dažnai naudojami ir pažangesni mokymo metodai: prasminga kiekvieno mokinio atsakymo analizė, nuolatinis ryšys su besimokančiuoju – atsakymai į jo klausimus, patarimai [8, 383]. Šių sistemų savybės atspindi ir humanistinio ugdymo tikslus: paskatinti mokinių savimoką ir nepriklausomybę, padėti jiems imtis didesnės atsakomybės, apsisprendžiant, ko mokytis, žadinti žingeidumą.

Intelektualių mokymo sistemų privalumai

Intelektualios mokymo sistemos neabejotinai yra pažangesnės už vadovaujančias kompiuterines mokymo programas [3, 98; 7, 383]. Jų privalumai:

- mokymas yra savarankiškas,
- mokymosi eiga ir tempas bei mokymo metodai parenkami pagal moksleivio gebėjimus,

- nėra griežtų terminų, mokymosi etapų, dėl kurių dažniausiai nukenčia greičiausiai ir lėčiausiai besimokantys,
- nėra mokinių konkurencijos,
- nereikia iš anksto parengti užduočių, kurios bus pateiktos moksleiviui,
- užduočių klausimai ir atsakymai konstruojami dinamiškai sistemos veikimo metu.

Tokio tipo sistemos yra naudingos mokantis savarankiškai konkrečių dalyko žinių bei formuojant įgūdžius.

Intelektualių mokymo sistemų kritika

Pastaruoju metu intelektualios mokymo sistemos dažniau yra vertinamos kritiškai nei palankiai. Nors įvairias tokio tipo sistemas bando kurti beveik trisdešimt metų, jos taip ir nesugebėjo įsitvirtinti mokykloje. Mokslininkai nevienodai interpretuoja šį nepopuliarumą [3; 11]. G. I. McCallas pamini tokias nesėkmės priežastis [3].

1. Mažai gerų intelektualių mokymo sistemų pavyzdžių. Daugelis sukurtų sistemų skirtos labai konkrečioms tikslams, jos neturi komercinių perspektyvų. Be to, yra sukurtos sėkmingesnės mokomosios programos (mikropasauliai, hipertekstai).
2. Neaiškios perspektyvos. Nėra specialistų, išmanančių ir dirbtinio intelekto, ir ugdymo problemas. Intelektualių mokymo sistemų kūrimas yra iš esmės eksperimentinė dirbtinio intelekto sritis, kurios tyrimų tikslas – moksliniai rezultatai, o ne pragmatiškas jų pritaikymas.
3. Gauti rezultatai bei naudojami metodai sudėtingai pritaikomi kuriant naujas sistemas.
4. Menkas teorinis pagrindas: neaišku, kaip aukšto lygio dirbtinio intelekto ir ugdymo teorijas perkelti į konkrečias mokymo sistemas.
5. Sudėtingas sistemų vertinimas: neaišku, kaip iširti sukurtosios sistemos efektyvumą ir tinkamumą.

Intelektualios mokymo sistemos ypač kritikuojamos konstruktyvaus mokymosi šalininkų [4, 102–104; 5, 132–134; 18, 89–90]. Neįmano-

ma kompiuteriu realizuoti visų konstruktyvaus mokymosi idėjų, nes žinių inžinerijos ir kognityvinės psichologijos teorijos kompiuterizuotame mokyme yra ribotos (neformalizuojama žmogaus kalba ir mokymo psichologijos teorijos kol kas yra neįveikiama kliūtis). Be to, mokant be kompiuterio neskiriama tiek daug dėmesio besimokančiojo žinioms diagnozuoti. Taigi kažin ar tai taip svarbu, kaip yra pabrėžiama daugelyje intelektualių mokymo sistemų. Apskritai mokinio modelio sukūrimą jie laiko tolimos ateities tikslu. Konstruktyvaus mokymo šalininkai dar labiau kritikuoja pačią „intelektualaus mokytojo“ idėją: moksleivis stumiamas „iš anksto numatyto, galbūt efektyviu sprendimų keliu, bet jam neleidžiama suklysti ar nesuprasti, jis neskatinamas pasirinkti, susiformuluoti tikslą, vertinti“ [4, 104]. Efektyvi mokymo programa turi skatinti tai atlikti patį moksleivį, o ne daryti pati. Apskritai intelektualios turi būti ir mokytojo pedagoginės bei dalyko žinios, ir besimokantysis, ir mokymosi kontekstas, ir kompiuterinės priemonės. Pagrindinis dalykas, kurį turi sugebėti atlikti kompiuterinė mokymo programa, – suteikti besimokančiajam pagalbą, kurios jam reikia konkrečiu metu atliekant vieną ar kitą veiksmą.

Į šias konstruktyvistų pastabas atsižvelgia intelektualių mokymo terpių kūrėjai.

Intelektualių mokymo sistemų vertinimas

Vertinant intelektualias mokymo sistemas galima naudotis metodais bei kriterijais, kurie buvo minėti skyrelyje „Kompiuterinių mokymo programų vertinimas“. Taigi reikėtų atskirai kalbėti apie sistemos ugdymo ir techninius aspektus.

Didaktiniai intelektualių mokymo sistemų principai yra kritikuojami daugelio kompiuterizuoto mokymo specialistų. Tai dažnai lemiamas aspektas vertinant šių produktų tinkamumą ugdymui. Atsisakius išankstinių nuostatų apie šių sistemų tinkamumą mokymui, jas galima įvertinti pagal tai, kokiomis iš aštuonių savybių (žr. skyrelį „Intelektualių mokymo sistemų samprata, tikslai ir savybės“) jos pasižymi ir ar pakankamai sėkmingai šios savybės realizuotos sistemoje.

Atsižvelgiant į griežtą modulinę intelektualių mokymo sistemų struktūrą, galima naudoti kitą vertinimo būdą: žinių, moksleivio, mokytojo ir sąsajos dalis vertinti atskirai. Pagrindinis vertinimo kriterijus – ar modulis pakankamai gerai atlieka savo funkcijas. Be to, reikia atsižvelgti ir į tai, kaip visi programos moduliai dera vienas su kitu.

Kitas vertinimo būdas – panagrinėti, kaip sukurtoji intelektualė sistema atitinka efektyvios mokymo aplinkos sampratą.

Ne mažiau sudėtingas uždavinys – įvertinti sistemos techninę dalį (vidinį bei išorinį teisingumą) [11, 115].

Vidinį sistemos teisingumą – ar programa yra suderinta, ar ji teisinga, ar atitinka jai keliamus techninius reikalavimus, – galima nustatyti išanalizavus programos kodą. Kartais tam naudojami matematiniai algoritmų analizės bei grafų teorijos metodai.

Dar sunkiau įrodyti išorinį sistemos teisingumą: žinių, didaktinių ir psichologinių aspektų korektiškumą ir pilnumą. Pastariesiems dviem tikslams dažniausiai naudojami tie patys metodai kaip ir didaktikoje bei psichologijoje (beje, ne visuomet pavyksta juos pritaikyti vertinant intelektualias mokymo sistemas). Bet koks vertinimas nėra visiškai patikimas jau vien todėl, kad kiekvienas moksleivis turi savo išankstinių nuostatų, požiūrių. Be to, mokymuisi daro įtaką ir kiti išoriniai (laikini bei nuolatiniai) veiksniai (pavyzdžiui, triukšmas klasėje ar bloga nuotaika). Neįmanoma iš anksto numatyti bei atsižvelgti į visas šias aplinkybes. Intelektualių mokymo sistemų išorinis teisingumas paprastai patvirtinamas tik eksperimentais.

Intelektualios mokymo sistemos per trisdešimt metų taip ir netapo efektyvia ir visuotinai taikoma mokymo priemone. Pagrindinės nesėkmės priežastys – neįmanoma sukurti idealios intelektualios mokymo sistemos, pedagoginiai ir psichologiniai šios sistemos pagrindai neatitinka šiuolaikinių švietimo filosofijų tikslų. Šiuo metu vis daugiau dėmesio mokymo priemonių kūrėjai skiria konstruktyvaus mokymosi terpėms, kuriose yra naudojamos ir kai kurios intelektualių mokymo sistemų bei dirbtinio intelekto idėjos, – intelektualioms mokymo aplinkoms. Jų efektyvumas taip pat dar nepagrįstas.

Literatūra

1. Ališauskas R. Kompiuterinių technologijų ir švietimo reformos sąveika. – *Kompiuterių naudojimas mokykloje: Paskaitų konspektai mokytojams*. – Vilnius: Baltic Amadeus, 1997, 5–25.
2. Bendrojo lavinimo mokyklos programos: Informatika. – Vilnius: Leidybos centras, 1994.
3. Costa E. The Present and Future of Intelligent Tutoring Systems. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 97–106.
4. De Corte E. Learning Theory and Instructional Science. – Reimann P., Spanda H. (Eds.) *Learning in Humans and Machines. Towards an Interdisciplinary Learning Science*. – ESF: Pergamon, 1996, 97–108.
5. De Corte E. Changing Views of Computer-Supported Learning Environments for the Acquisition of Knowledge and Thinking Skills. – Vosniadou S., De Corte E., Glaser R., Mandl H. (Eds.) *International Perspectives on the Design of Technology-Supported Learning Environments*. – New Jersey: Lawrence Erlbaum Associates, 1996, 129–145.
6. Ferguson D. L. Computers in Teaching and Learning: An Interpretation of current Practices and Suggestions for Further Directions. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 33–50.
7. Gage N. L., Berliner D. C. *Pedagoginė psichologija*. – Vilnius: Alma litera, 1994.
8. Jones M. Instructional Systems Need Instructional Theory: Comments on Truism. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 1–13.
9. Kanselaar G., Erkens G. Interactivity in Cooperative Problem Solving with Computer. – Vosniadou S., De Corte E., Glaser R., Mandl H. (Eds.) *International Perspectives on the Design of Technology-Supported Learning Environments*. – New Jersey: Lawrence Erlbaum Associates, 1996, 185–202.
10. Markauskaitė L. Kaip įvertinti kompiuterinę mokymo priemonę. – *Kompiuterizuotas mokymas Lietuvoje: Konferencijos darbai*. – Vilnius: Solertija, 1997, 65–74.
11. McCalla G. I. Intelligent Tutoring Systems: Navigating the Rocky Road to Success. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 107–122.
12. Mendelsohn P. Mapping Models of Cognitive Development of Design Principles of Learning Environments. – Vosniadou S., De Corte E., Glaser R., Mandl H. (Eds.) *International Perspectives on the Design of Technology-Supported Learning Environments*. – New Jersey: Lawrence Erlbaum Associates, 1996, 323–344.
13. Newman D. Formative experiments on the Coevolution of Technology and the Educational Environments. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 61–70.
14. Ozmon H. A., Carver S. M. *Filosofiniai ugdymo pagrindai*. – Vilnius: Leidybos centras, 1996.
15. Papertas S. *Minčių audros: Vaikai, kompiuteriai ir veiksmingos idėjos*. – Vilnius: Žara, 1995.
16. Paulauskaitė V. XX amžiaus švietimo filosofijų apžvalga. – *Mokykla*, 1997, Nr. 4, 1–5.
17. Reiser B. Problem Solving and Explanation in Intelligent Tutoring Systems: Issues for Future Research. – Scanlon E., O'Shea T. (Eds.) *New Directions of Educational Technology*. – NATO ASI Series F, Vol. 96. Springer, 1992, 199–210.
18. Reusser K. From Cognitive modelling to the Design of Pedagogical Tools. – Vosniadou S., De Corte E., Glaser R., Mandl H. (Eds.) *International Perspectives on the Design of Technology-Supported Learning Environments*. – New Jersey: Lawrence Erlbaum Associates, 1996, 81–103.
19. Scardamalia M., Bereiter C. Adaptation and Understanding: A Case for New Cultures. – Vosniadou S., De Corte E., Glaser R., Mandl H. (Eds.) *International Perspectives on the Design of Technology-Supported Learning Environments*. – New Jersey: Lawrence Erlbaum Associates, 1996, 149–163.
20. Squires D., McDougall A. *Choosing and Using Educational Software*. – London: The Falmer Press, 1994.
21. Žukauskienė R. *Raidos psichologija*. – Vilnius: Valstybinis leidybos centras, 1996.

JAVA IŠ N. WIRTHO MOKYKLOS POZICIJŲ

Michael Franz

Apie Javą, „Interneto“ programavimo kalbą, diskutuojama dabar labai daug. Deja, dažnai nutylima, kad dauguma Javos galimybių nėra naujos: daug kas seniai įtvirtinta kitose, ypač artimose Paskaliui, programavimo kalbose. Šiame straipsnyje kalbėsime apie didėjančią susidomėjimą Java, atskleisime šiuo metu dar mažai tyrinėtus ryšius tarp Javos ir Paskalio, pagaliau pabandysime nuspėti Javos ateitį.

Kaip pastebėjo Niklausas Wirthas [17], informatikos pasaulis irgi neatsispiria madai. Galima drąsiai pasakyti, kad Java dabar tikrai madinga, bet anksčiau ar vėliau, madai pasibaigus, domėjimasis ja irgi atslūgs. Šio straipsnio tikslas – išanalizuoti Javos fenomeną šiuolaikinio, ir ypač europinio, informatikos mokslo kontekste, išsklaidyti mitą apie spontanišką naujos technologijos atsiradimą „iš nieko“, įvertinant realų Javos indėlį į šį mokslą.

Javos koncepcija – tai objektinio programavimo kalba ir šiai kalbai sukonstruotos virtualios mašinos architektūra, pateikta 1995 metų rugpjūtį per išpūdingą marketingo akciją. Dėl daugelio faktorių, apie kuriuos išsamiau kalbėsime vėliau, gan greitai Java tapo neįtikėtinai populiari. Kartais netgi pačiam autoriui sunku patikėti, kad Java, gyvuojanti tik dvejus metus, jau spėjo giliai įleisti šaknis į informatikos pasaulį.

Greitas Javos plitimas

Javos koncepcijos, t. y. kalbos ir virtualios mašinos kombinacijos, šaknys – 1990 metų firmos „Sun“ projektas „Green(Oak)“. Pradinis šio projekto tikslas buvo būtinių elektronikos programinės įrangos tobulinimas moderniuose elektros prietaisuose: pavyzdžiui, tosteriuose, skal-

bimo mašinose, mobiliuose telefonuose, yra sudėtingesnių programų, kurias atlieka į tuos prietaisus įmontuoti kompiuteriai, vadinami mikrokontrolieriais. Firmos „Sun“ tikslas buvo rasti daugybei mikrokontrolierių vienintelę subalansuotą architektūrą – šiandieninę Javos virtualią mašiną, turinčią daug galimybių. Kaip mikroschemų gamintojas, firma „Sun“, žinoma, planavo pagaminti ir atitinkamų mikrokontrolierių bei prekiauti jais.

Kaip teigia firma „Sun“ [10], idėja perorientuoti Javą į WWW pirmausia kilo vienam iš firmos įkūrėjų Billui Joy'ui 1994 metais. 1994 metų rudenį prasidėjo vadinamųjų „HotJava“ naršyklių realizacija su mažų programėlių funkcionavimu. 1995 m. kovo mėn. naršyklė buvo pateikta parinktai strateginei „Sun“ partnerių grupei. Tarp jų buvo ir „Netscape Communications Corporation“ vadovų. Šios firmos parama vėliau iš esmės prisidėjo prie Javos sėkmės. 1995 m. rugpjūtį buvo pateiktas galutinis Javos variantas. Pristatymas buvo gan neįprastas – per daugybę spaudos konferencijų.

Stebėtinai greitą Javos idėjos plitimą nulėmė beveik ideali daugelio faktorių darna.

- Vis labiau įsigalint „Internetui“ plito ir su juo susijusios idėjos. Įsivaizduokime, kad idėjos – tai skruzdėlės, šliaužiojančios oro balionu. Jei balionas bus pučiamas, skruzdėlės galės judėti greičiau, negu sugebėtų pačios.
- „Microsoft“ firmos dominavimas programinės įrangos rinkoje skatino jos konkurentus vienytis, dėl to Javos technologiją parėmė daug firmų, kurių nuomone, Java – tinkama priemonė kovai su monopoline „Microsoft“.
- Vartotojai nuolat ieško „vaistų“, t. y. būdų išspręsti jų visas su programine įranga susijusias problemas, o prekiautojai – naujų šūkių ir laikmetį atitinkančių naujovių, kurių dėka būtų galima lengviau parduoti produktus. Galima prisiminti ankstesnių madų kaitą ir jų šūkius „struktūrinis programavimas“, „objektinis programavimas“ ir „klientas – tarnybinė stotis“, į kuriuos buvo dedama didelių, bet vos vos pasiteisinusių vilčių.

- Dėl visuotinio nepasitenkinimo Si [11] ir Si++ [15] kalbomis Javos panaudojimą mokymui labai entuziastingai priėmė Amerikos universitetų dėstytojai. Šis faktas, beje, neprieštaravo ir JAV universitetuose vis labiau išsiviraujančiai nuomonei keisti fundamentalų išsilavinimą trumpalaikiais, rinkos poreikius atitinkančiais kursais. Esant tokiai situacijai tik elitinės aukštosios mokyklos, tokios kaip Masačiūsetso technologijos institutas, tegalėjo vartoti paskaitose šiuolaikines, bet ne „pramonines“ programavimo kalbas, pavyzdžiui, *Scheme* [1]. Atsiradus Javai aukštosios mokyklos pagaliau galėjo atsikratyti joms primestos Si arba Si++, galėjo nekreipti dėmesio į „nepalankius pramonei“ vertinimus. Kalifornijos universitete (Irvine, JAV), kur dėsto šių eilučių autorius, mokymo planas jau 1996 m. rudenį buvo sudarytas praktiškai laikantis Javos koncepcijos. Stebėtina, bet diskusijų beveik nebuvo, nors iki tol dėstytojų kolektyvas ilgą laiką buvo susiskaldęs ir neturėjo vienos nuomonės dėl „Paskalio arba Si“ pasirinkimo.

Java, žinoma, tokio milžiniško pasisekimo neturėtų, jei būtų bloga programavimo kalba. Java ne tik nėra prasta programavimo kalba, bet iš tikrųjų tai viena geriausių kalbų, kur kas geresnė už anksčiau dominavusias Si ir Si++. Būtų neteisinga manyti, kad Java – tik viena iš daugelio modernių, gerų programavimo kalbų, kad ji už savo išskirtinį, svarbų vaidmenį turi būti dėkinga tik rinkai.

Akivaizdu, kad firma „Sun“ jau turėjo patirties operacijų sistemų rinkoje (pavyzdžiui, *VHS* nurungė *Betamax*, *DOS* – *Macintosh*) ir žinojo: didesnę pasisekimą turi tas produktas, kuris kuo anksčiau užkariauja rinką. Taigi, pavyzdžiui, buvo priimtas protingas firmos „Sun“ sprendimas: reklamuoti Javą, nelaukti, kol dėl kiekvienos šios kalbos ir bibliotekų apibrėžimų smulkmenos bus iki galo apsispręsta. Ir nors šiek tiek vėliau, kai pagerinta versija *Java 1.1* buvo galutinai įvertinta, Java susilaukė kritikos, ji vis tiek jau funkcionavo greta daugybės varžovų kaip dominuojanti aktyvaus teksto „Internet“ technologija. Javos ankstyvo ir aktyvaus pardavimo priežastys yra konkuruojančios technologijos, pavyzdžiui, „Bell“ laboratorijos produktas „Inferno“ [13], kurį ruošė tik mažas būrys specialistų.

Kas tikrai nauja Javos kalboje?

Išsamiau panagrinėjus pasidaro aišku, kad esminių naujovių Javoje nedaug. Java sujungė daugybę seniai žinomų idėjų, kurios, tariamai nepriklausomos viena nuo kitos, jau ne vieną kartą buvo atrastos įvairiose pasaulio vietose. Stulbinančiai didelė Javos branduolio dalis – tai pirmiausia ankstesnių įžymaus mokslininko Niklausio Wirtho ir jo grupės iš Šveicarijos federalinio technologijos instituto (ETH) Ciuriche darbų vaisius.

- Javos suderinamumas su skirtingomis architektūromis – tai pirmiausia jos virtualios mašinos [12], leidžiančios lengvai imituoti daugelį architektūrų, galimybes. Virtualios mašinos idėja buvo labai populiari jau daugiau kaip prieš 20 metų, tačiau vėliau ji buvo primiršta. Tada buvo kalbama apie Šveicarijos federaliniame technologijos institute sukurtą *Pascal-P* [14], kuris turėjo lemiamos įtakos, kad paplito Paskalis. Įdomu tai, kad Paskalio ir Javos virtualių mašinų architektūra nepaprastai sudėtinga: abiejose naudojamos vieno baido komandos be adreso (operandai talpinami į dėklą).
- Nenuja buvo ir griežtos duomenų tipų kontrolės idėja, kurios esmė – dinaminiai duomenų tipai ir „šiukšlių rinkimo“ sistema. Čia vėlgi pradininkas buvo Šveicarijos federalinis technologijos institutas. Daugelis argumentų, kuriais remdamasi Javos grupė atsisakė rodyklių, buvo suformuluoti penkeriais metais anksčiau, negu Java pasirodė tarp kitų modulinių kalbų, – tai laikas, kai buvo kuriama programavimo kalba *Oberon* [19]. Nuo to laiko Wirtho kuriamose programavimo kalbose [18, 19, 20] imamasi ir kitų priemonių patikimui didinti: pavyzdžiui, tikrinama, ar masyvo elementas patenka į indeksų režius prieš atliekant operacijas su juo. Deja, iki šiol programuotojai laikė tai greičiau nemaloniu ribojimu negu nauda. Šiaip visuomenės nuomonė firmai „Sun“ turėjo nemažą įtaką.
- Mintis apie tai, kad griežtos duomenų tipų kontrolės kalba įgalina kurti programas, perkeliamas iš vienos architektūros į kitą, gimė 1990–1993 metais, visose Šveicarijos federalinio technologijos instituto kuriamose programavimo šeimose [3] realizavus *Oberon-Sys-*

tems [21, 22]. Jos išsiskyrė ne tik vientisa taikomųjų programų sąsaja su visomis programinės įrangos terpėmis [7], bet ir vientisa nuo mašinių nepriklausoma dokumentų architektūra [16], jau tada numačiusia slankių aktyvių objektų įtraukimą į tekstą. Plėtojant šią idėją buvo sukurtas programų paketas *Oberon System 3* [9] – taigi buvo pažengta dar toliau, negu Java grindžiama šiuolaikinė naršyklė. Konceptualių reikalavimų netenkinantis teksto ir objektų atskyrimas buvo pakeistas vientisu, integruotu objekto modeliu.

- Javos programa gali būti sudaryta iš kelių viena nuo kitos nepriklausomai kompiliuojamų dalių, kurios dinamiškai sujungiamos į visumą tik paleidžiant programą [6]. Javos sistemoje iš tikrųjų vyksta kompiliavimas, patikrinant ryšius tarp modulių. Programuotojams, anksčiau naudojusiems Si arba Si++ kalbas, gali pasirodyti, kad tai reikšmingas patobulinimas, kadangi Si ir Si++ programavimo mechanizmas labai primityvus, įtraukiantis į tekstą išorines bylas. Tai, ką gali Java, modulinio programavimo pasaulyje jau daugiau kaip dešimt metų yra tapę standartais, o Wirthas, sukūręs Modula-2 kalbą [18], ir čia padėjo kertinį akmenį.
- Pagrindinė Javos idėja – nuo terpės nepriklausomos mobilios programos ir kodo generavimas tų programų interpretavimo (vykdymo) metu – gerokai anksčiau atsirado Šveicarijos federaliniame technologijos institute. Šių eilučių autoriaus disertacijoje [5] (jos vadovas N. Wirthas) yra aprašyta dinamiškai išplečiama sistema, kurioje objektinis kodas nepastebimai generuojamas iš nuo kompiuterio nepriklausomo tarpinio kodo. Šioje disertacijoje, kuri anglų kalba buvo paskelbta 1994 m. vasarį, t. y. daugiau kaip metai prieš patį pirmąjį Javos pristatymą, jau buvo žodis „*applet*“ (maža programėlė) ir keli scenarijai, labai panašūs į šiandieninius Javos scenarijus. Šiuo metu autorius dirba toliau plėtodamas šią sistemą, kuri, palyginus ją su Java, turi keletą pranašumų [4, 8].

Ko gero, būtų įdomu ir tai, kad 1994 m. kovo mėnesį Kalifornijoje autorius skaitė nemažai pranešimų savo disertacijos tema. Aišku, skaitytas pranešimas ir firmoje „Sun Laboratories, Inc“. Be to, minėtas Bil-

lis Joy, perorientavęs Javos projektą į WWW, vienas iš pirmųjų įsigijo iš Šveicarijos federalinio technologijos instituto *Oberon-Systems* licenciją ir 1994–1995 metais palaikė ryšius su šiuo institutu: iš čia ir paaiškėjo, kad jis taip pat yra skaitęs minėtą disertaciją [5].

Kokia Javos ateitis?

Jau minėjome, kad domėjimasis Java neišvengiamai turėtų sumažėti. Dabar vis daugiau pasiūlymų perrašyti Javos programavimo kalbą „seną programinių palikimą“, tačiau šis uždavinys dėl to, kad Si arba Si++ yra šio palikimo pagrindas, nėra visai paprastas, kadangi Java neturi rodyklių. Kyla pavojus, kad priėmus tokį projektą kam nors gali gimi idėja sukurti specialią Javos versiją su rodyklėmis, kuri, žinoma, lyg viesulas nušluotų visus Javos pranašumus.

Be to, Javos pasaulio susiskaldymas – tai grėsmė ir kalbai subirti į keletą nesuderintų dialektų: taip, deja, jau ne kartą buvo nutikę „pramoniniams standartams“. „Microsoft“ korporacija, prieš kurią buvo iš pradžių nusistatęs „Java-Allianz“, būdama Javos technologijos savininkė, tarsį kilimą iš po kojų ištraukė konkurentams iš „Sun“. „Microsoft“ dabar kuria ne tik Javos programavimo kalbą, kurią klastingai vadina J++ ir kuri galbūt ateityje nebus visiškai suderinta su Java, bet taip pat tobulina ir Javos virtualiąją mašiną.

Javos, kaip mokymui skirtos programavimo kalbos, gyvavimas aukštosiose mokyklose garantuotas, ir tai labai sveikintina. Ir autoriaus universitete, ir daugelyje aukštųjų mokytojų įstaigų praeitais metais buvo pasinaudota palankiu momentu, ir Java mokymo planuose pakeitė skandalingą Si ir Si++. Atgal kelio nėra, netgi jei pramoniniai projektai ir toliau būtų realizuojami Si arba Si++ kalba. Profesoriai švaria sąžine galės pareikšti, kad vartoja modernią ir į ateitį orientuotą kalbą, o Java ir Si sintaksinis panašumas turėtų patenkinti ir praktikus.

Gali pasirodyti šiek tiek ironiška, tačiau „Sun“ marketingo mašina padarė tai, ko mes, informatikos dėstytojai, ne vieną dešimtmetį bergždžiai siekėme: netgi tikrieji hakeriai dabar pripažįsta griežtos duomenų tipų kontrolės ir abstrahavimo naudą.

Apibendrinimas

Kaip JAV aukštosios mokyklos profesoriui man Java priimtina jau vien dėl to, kad leido be skausmo atsisakyti nepasiteisinusių Si ir Si++ kalbų. Java, kaip moderni programavimo kalba, artima Paskalio, Modula-2 ir Oberono tradicijai, pasiūlė gan priimtina kompromisą, užsimaskuodama Si kalbos sintakse. Todėl aš kaip praktikas palaikau Java visur, kur ji žengia kaip Si arba Si++ alternatyva. Tačiau tie, kurie jau yra įvaldę galingas, bet estetiškai šiuolaikiškesnes programavimo kalbas (Oberoną arba Ada95 [2]), gali ramia sąžine dirbti toliau, nebūgštaudami dėl savo perspektyvų.

Literatūra

1. Abelson H., Sussman G. J., Sussman J. *Structure and Interpretation of Computer Programs*. – The MIT Press / McGraw-Hill, 1985.
2. *Ada Reference Manual: Language and Standard Libraries*, Version 6.0. – ISO/IEC 8652:1995, 1995.
3. Brandis M., Crelier R., Franz M., Templ J. The Oberon System Family. – *Software-Practice and Experience*, 1995, v. 25, Nr. 12, 1331–1366.
4. Franz M. Beyond Java: An Infrastructure for High-Performance Mobile Code on the World Wide Web, – *Proceedings of WebNet 97, World Conference of the WWW, Internet, and Intranet*. – Association for the Advancement of Computing in Education, 1997, 33–38.
5. Franz M. *Code-Generation On-the-Fly: Key to Portable Software*. – Dissertation Nr. 10497, ETH Zürich, 1994.
6. Franz M. Dynamic Linking of Software Components. – *IEEE Computer*, 1997, v. 30, Nr.3, 74–81.
7. Franz M. Emulating an Operating System on Top of Another. – *Software – Practice and Experience*, 1993, v. 23, Nr. 6, 677–692.
8. Franz M., Kistler T. Slim Binaries. – *Communications of the ACM*, 1997, v. 40, Nr. 12, 87–94.
9. Gutknecht J. Oberon System 3: Vision of a Future Software Technology. – *Software – Concepts and Tools*, 1994, v. 15, Nr. 1, 26–33.
10. JavaSoft, Inc. What Is Java? – <http://www.javasoft.com/nav/whatis> – 1998.
11. Kernighan B. W., Ritchie D. M. *The C Programming Language*. – Prentice-Hall, 1978.

12. Lindholm T., Yellin F., Joy B., Walrath K. *The Java Virtual Machine Specification*. – Addison-Wesley, 1996.
13. Lucent Technologies Inc. Inferno. – <http://plan9.bell-labs.com/inferno/>.
14. Nori K. V., Amman U., Jensen K., Nägeli H. H., Jacobi Ch. Pascal-P Implementation Notes. – *Pascal: The Language and its Implementation*. – Chichester: Wiley, 1981.
15. Stroustrup B. *The C++ Programming Language*. – Addison-Wesley, Reading, 1987.
16. Szyperski C. A. Writing Applications: Designing an Extensible Text Editor as an Application Framework. – *Proceedings of the Seventh International Conference on Technology of Object-Oriented Languages and Systems (TOOLS'92)*. – Dortmund, 1992, 247–261.
17. Wirth N. Modula-2 and Object-Oriented Programming. – *Proceedings of the First International Modula-2 Conference*. – Bled, 1989, 7–13.
18. Wirth N. *Programming in Modula-2*. – Heidelberg: Springer-Verlag, 1982.
19. Wirth N. The Programming Language Oberon. – *Software – Practice and Experience*, 1988, v. 18, Nr. 7, 671–690.
20. Wirth N. The Programming Language Pascal. – *Acta Informatica*, 1971, v. 1, Nr. 1, p. 35–63.
21. Wirth N., Gutknecht J. *Project Oberon: The Design of Operating System and Compiler*. – Addison-Wesley, 1992.
22. Wirth N., Gutknecht J. The Oberon System. – *Software-Practice and Experience*, 1989, v. 19, Nr. 9, p. 857–893.

Versta iš: M. Franz. Java – Anmerkungen eines Wirt-Schülers. – *Informatik-Spektrum*, Februar 1998, B. 21, H. 1, S. 23-26

Iš vokiečių k. vertė Kristina Palaimaitė

PELĖS PANAUDOJIMO GALIMYBĖS „LOGOWRITER“ SISTEMOJE

Tatjana Balvočienė

Turintys darbo su „LogoWriter“ sistema įgūdžių jau žino, kad galima piešti ne tik Vėžliukui duodant komandas, bet ir naudojantis pele. Apie tai galima paskaityti T. Balvočienės, V. Dagienės ir A. Klupšaitės knygoje „Logo: „LogoWriter“ komandų žinynas mokytojams ir mokiniams“ (p. 210) arba tų pačių autorių knygoje „Aš mokausi Logo: Projektų knyga mokiniams“ (p. 48–49). Neturintiems šių knygų ir nemokantiems naudotis pele Vėžliukui kilnoti ir piešti, trumpai paaiškinsiu, kaip tai daroma.

Norėdami pele perkelti Vėžliuką į kitą ekrano vietą, nuvedame pelės žymeklį ties Vėžliuku, nuspausdami kairįjį pelės klavišą ir neatleisdami jo stumiame pelę (kartu ir Vėžliuką) į norimą ekrano vietą. Jei tuo metu kartu nuspausime klavišą <Shift>, Vėžliukas brės liniją. Šią „LogoWriter“ sistemos galimybę galime panaudoti įvairiai, o ypač toks piešimo būdas praverčia, kai turime nupiešti ekrane netaisyklingos formos didelius kontūrus: pavyzdžiui, debesis, upę, kelią, dūmus, kylančius iš kamino, jūros krantą ir panašiai. Apie tai galite paskaityti ir Logo projektų knygos II skyriaus 4 skyrelyje „Jūra“ (p. 77–78).

Tačiau taip piešti ne visiems yra lengva (viena ranka reikia laikyti nuspaustą klavišą <Shift>, kita – stumdyti pelę), todėl yra labai patogų turėti kai kurias procedūras, palengvinančias piešimą ranka (naudojant pelę). Taigi apie tai ir pakalbėsime.

„LogoWriter“ sistema turi darbo su pele komandas **pelėsvieta** ir **pelėsmygtukai**¹. Abi šios komandos funkcinės, jas galima naudoti kitoje komandoje. Komandos **pelėsvieta** rezultatas – pelės žymeklio koordinatės *x* ir *y*, sujungtos į sąrašą. Komandos **pelėsmygtukai** rezultatas –

sąrašas iš trijų loginių reiškinių, kurios gali būti *taip* arba *ne*, priklauso-
mai nuo to, ar buvo paspaustas kuris nors iš pelės klavišų, pavyzdžiui:

[*taip ne ne*] – buvo paspaustas pirmas pelės klavišas;

[*ne ne ne*] – pelės klavišai nebuvo spaudžiami;

[*ne ne taip*] – buvo paspaustas trečias pelės klavišas².

Sukurkime tokią rekursinę procedūrą.

tai pelytė_piešia

jei pirmas pelėsmygtukai [piešim eikxy pelėsvieta]

pelytė_piešia

taškas

Ši procedūra yra rekursinė. Kaip žinome, tokios procedūros baigiamos vienu metu paspaudus klavišus <Ctrl> ir <Break>. Liepkime Vėžliukui vykdyti procedūrą *pelytė_piešia* – ir galime ekrane piešti ranka (naudodami pelę), spausdami tik pirmąjį pelės klavišą (klavišo <Shift> klaviatūroje spausti nereikia). Taip piešti yra daug patogiau – pabandykite! Galite nuspaudę pirmąjį pelės klavišą tempti Vėžliuką ir piešti liniją ekrane, o galite reikiamoje ekrano vietoje tik spustelėti pirmąjį pelės klavišą – Vėžliukas atkeliaus į tą vietą nubrėžęs liniją.

Perkelti Vėžliuką ekrane galima ir be specialios procedūros, tačiau yra patogiau naudoti tam tikslui parašytą procedūrą – Vėžliuką bus galima iš karto perkelti į nurodytą ekrano vietą (Vėžliuką perkėlinėsime spausdami paskutinįjį pelės klavišą).

tai pelytė_vaikšto

jei paskutinis pelėsmygtukai [eiksim eikxy pelėsvieta]

pelytė_vaikšto

taškas

Šias procedūras galima sujungti į vieną.

tai pelytė

jei pirmas pelėsmygtukai [piešim eikxy pelėsvieta]

jei paskutinis pelėsmygtukai [eiksim eikxy pelėsvieta]

pelytė

taškas

Kai spausite pirmąjį pelės klavišą, Vėžliukas pieš, o kai spausite paskutinįjį pelės klavišą, Vėžliukas tik vaikščios ekrane.

¹ Komanda **pelėsmygtukai** nėra aprašyta „LogoWriter“ komandų žinyne.

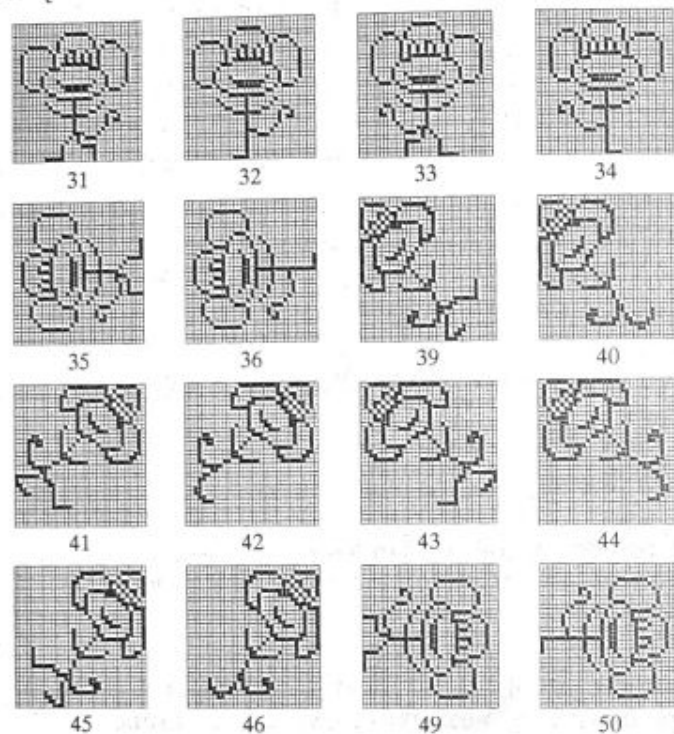
² Jei Jūsų pelė turi tik du klavišus, sąrašo antrojo elemento reikšmė visuomet bus *ne*.

Manau, jūs sugalvosite ir daugiau būdų, kaip panaudoti komandą **pelėsmygtukai**.

Dar noriu pasiūlyti jums keletą procedūrų, kurias naudodami vaikai greičiau išmoks valdyti pelę. Galite sukurti keletą panašių procedūrų kartu su vyresniųjų klasių moksleiviais ir šias procedūras naudoti per pradedančiųjų pamokas.

Mūsų pavyzdyje Vėžliukas užsidės beždžionėlės kaukę ir beždžionėlė lakstys po ekraną bandydama sugauti pelytę (rodyklę ekrane). Jei beždžionėlė sugaus pelytę (priartės prie jos per kelis žingsnelius) – gaus kokį nors prizą: pavyzdžiui, ananasą ar bananą.

Kad beždžionėlė galėtų judėti skirtingomis kryptimis, reikia paruošti keletą beždžionėlės kaukių.



1 pav. 16 beždžionėlės kaukių – beždžionėlė galės lakstyti 8 kryptimis

Procedūra *kur_eina* liepia Vėžliukui paimti nurodytas kaukes (vietiniai kintamieji *kaukė1* ir *kaukė2*) ir paeiti keletą žingsnelių nurodyta kryptimi (visuotinis kintamasis *kur_žiūri*).

```

tai kur_eina :kaukė1 :kaukė2
  eisim greitai
  žvelk :kur_žiūri
  imkkaukę :kaukė1 priekin 7 lauk 2
  imkkaukę :kaukė2 priekin 7 lauk 2

```

taškas

Procedūra *gaudynės* nustato, į kurią pusę žiūri beždžionėlė, ir kiekvienai kryptiai nurodo atitinkamas kaukes, kad vaizdas būtų tikroviškesnis.

```

tai gaudynės
  keisksp 6
  tebus "koks_atstumas kiekiki pelėsvieta
  jei :koks_atstumas < 8 [baik]
  tebus "kur_žiūri kieklink pelėsvieta
  jei (kiekvienas :kur_žiūri > 20 :kur_žiūri < 71) [kur_eina 39 40]
  jei (kiekvienas :kur_žiūri > 70 :kur_žiūri < 111) [kur_eina 33 34]
  jei (kiekvienas :kur_žiūri > 110 :kur_žiūri < 161) [kur_eina 41 42]
  jei (kiekvienas :kur_žiūri > 160 :kur_žiūri < 201) [kur_eina 49 50]
  jei (kiekvienas :kur_žiūri > 200 :kur_žiūri < 251) [kur_eina 43 44]
  jei (kiekvienas :kur_žiūri > 250 :kur_žiūri < 291) [kur_eina 31 32]
  jei (kiekvienas :kur_žiūri > 290 :kur_žiūri < 341) [kur_eina 45 46]
  jei (kiekvienas :kur_žiūri > 340 :kur_žiūri < 361) [kur_eina 35 36]
  jei (kiekvienas :kur_žiūri > 0 :kur_žiūri < 21) [kur_eina 35 36]

```

gaudynės

taškas

Galima pasirinkti ir daugiau krypčių nei yra procedūroje *gaudynės*. Tada reikės sukurti daugiau kaukių ir šiek tiek patobulinti šią procedūrą.

Procedūra *gaudynės* vykdoma tol, kol atstumas nuo beždžionėlės iki pelės žymeklio (rodyklės) ekrane yra didesnis nei 7 žingsneliai. Jei šis atstumas mažesnis, manoma, kad beždžionėlė sugavo pelę. Tada vykdoma procedūra *dovana*, kurioje beždžionėlei įteikiamas ananasas.

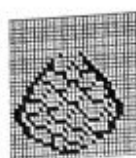
tai dovana

```

  imkkaukę 31 piešim trept
  žvelk 90 eisim priekin 28
  piešim keisksp 14 imkkaukę 52 trept
  keisksp 15 imkkaukę 53 trept slėpkis
  Valio! 25

```

taškas



52



53

2 pav. Beždžionėlė, pagavusi pelę, gauna prizą – ananasą, nupieštą iš šių dviejų kaukių

Procedūroje *dovana* naudojama procedūra *Valio!*, kuri atsitiktinai parinktose ekrano vietose piešia skirtingos spalvos žodžius „Valio!!!“.

```

tai Valio! :kiek_džiaugsmo
kartok :kiek_džiaugsmo
[eisim betkur
keisksp (1 + betkoks 15)
piešim pieštekstą "Valio!!! lauk 3]
taškas

```



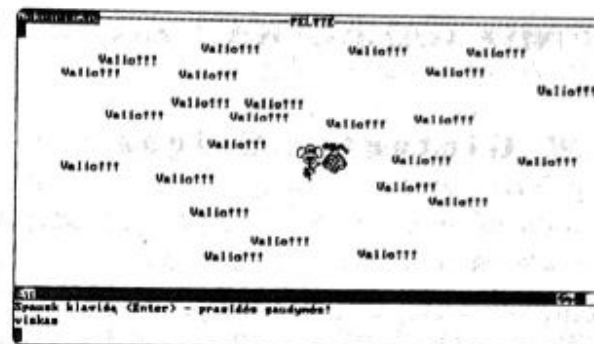
3 pav. Beždžionėlė su ananasu

Procedūra *betkur* atsitiktinai parenka koordinates x ir y – panaši procedūra aprašyta Logo projektų knygos V skyriaus 4 skyrelyje „Ramonės“ (p. 226). Šioje procedūroje matote, kaip komandoje *eikxy* galima naudoti kintamuosius – reikia gautas kintamųjų x ir y reikšmes sujungti į sąrašą komanda *gauksarašą*.

```

tai betkur
tebus "x betkoks 520
tebus "y betkoks 320
jei :x > 260 [tebus "x minus (:x - 260)]
jei :y > 160 [tebus "y minus (:y - 160)]
tebus "koordinatės gauksarašą :x :y
eikxy :koordinatės
taškas

```



4 pav. Taip atrodo kompiuterio ekranas po to, kai beždžionėlė pagauna pelę

Belieka sukurti procedūras *viskas* ir *startas*.

```

tai viskas
valykpš
rodykis
gaudynės
dovana
taškas

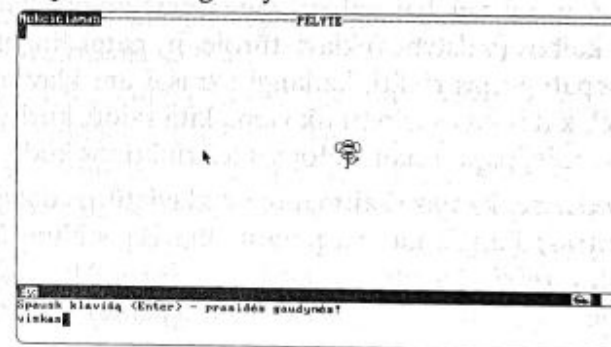
```

```

tai startas
jei priešingai geroji? [versk]
valykkl
parodyk "[Spausk klavišą <Enter> – prasidės gaudynės!]
rodyk "viskas
taškas

```

O dabar – pradėkime gaudynes.



5 pav. Gaudynių pradžia

UŽSIENIO KALBŲ RAIDŽIŲ KODAI

Gintautas Grigas

Rašant lietuvišką tekstą kompiuteriu problemų nėra. Tačiau, kai tenka rašyti kuria nors kita kalba, prireikia vienos ar kitos raidės, kurios nėra kompiuterio klaviatūroje.

Laiškus į užsienį dažniausiai rašome angliškai – ir ne vien tik anglams. Bet, kai reikia parašyti užsieniečio vardą, pavardę, įstaigos pavadinimą ar adresą, prireikia vienos kitos raidės, kurios nėra klaviatūroje. Kaip jas parašyti?

Kad kompiuteris galėtų parodyti raidę ekrane arba išspausdinti, reikia, kad ta raidė būtų įtraukta į kodų lentelę, o šriftas turėtų jos „paveikslą“. Kodų lentelės ir jas atitinkantys šriftai turi daugiau raidžių, negu galima surinkti klaviatūra. Pavyzdžiui, šriftas „Times New Roman Baltic“ turi ne tik lietuviškas, bet ir latviškas, estiškas, lenkiškas, vokiškas ir dar kelių kitų kalbų raides. O lietuviška klaviatūra galima surinkti visas lietuviškas, bet ne visas kitų kalbų raides.

Kitos kalbos raides galima surinkti persijungus į reikiamos kalbos klaviatūrą. Tačiau tai nelabai patogiu, nes reikia žinoti, kaip raidės išdėstytos kitos kalbos (valstybės) klaviatūroje, ir, netgi žinant išdėstymą, neįprasta ir nepatogu jas rinkti, kadangi užrašai ant klavišų neatitinka tikrovės. Todėl, kai reikia surinkti tik vieną kitą raidę, kurios nėra lietuviškoje klaviatūroje, paprasčiau vietoj raidės rinkti jos kodą.

Raidės kodas renkamas skaitmeninėje klaviatūros dalyje. Skaitmenų lygis (registas) įjungiamas paspaudus klavišą <Num Lock>. Kol renkamas kodas, reikia laikyti paspaustą klavišą <Alt>. Kai surenkamas kodas, atleidžiamas klavišas <Alt> ir tada ekrane atsiranda raidės vaizdas.

„Windows“ terpėje kodo pradžioje reikia surinkti nulį. Pavyzdžiui, vokiškos raidės Ä kodas yra 196. Todėl, norėdami parašyti šią raidę, turėtume surinkti 0196.

Visų kalbų raidės netelpa į vieną kodų lentelę. „Windows“ terpėje vartojamos net keturios kodų lentelės skirtos lotynišką abėcėlę vartojančioms kalboms. Šių lentelių numeriai, lentelės atitinkantys žodžiai, vartojami kai kurių (vadinamųjų unikodinių) „Windows“ terpės šriftų pavadinimuose, šriftų kodai, reikalingi užregistruojant šriftus, ir kalbos, kurioms tos kodų lentelės yra skiriamos, yra pateikti 1 lentelėje.

1 lentelė. Kodų lentelės „Windows“ terpėje

Kodų lentelė	Lentelės santrumpa šrifto pavadinime	Šrifto kodas	Kalbos (pagal „Windows95 Pan Euro“)
1250	CE	238	albanių, čekų, kroatų, lenkų, slovakių, slovėnų, vengrų
1252	–	–	afrikanių, anglų, baskų, danų, indoneziečių, islandų, ispanų, italų, katalonietų, norvegų, olandų, portugalų, prancūzų, suomių, švedų, vokiečių
1254	Tur	162	turkų
1257	Baltic	186	estų, latvių, lietuvių

Lotyniškosios abėcėlės raidžių, kurių nėra angliškoje abėcėlėje, kodus pateikiame 2 lentelėje. Tikimės, kad ja bus galima pasinaudoti rašant tekstus įvairiomis kalbomis.

Plačiau vartojamų kalbų (pvz., anglų, vokiečių) raides galima rasti keliose kodų lentelėse. Tačiau tik viena kodų lentelė laikoma pagrindine tos kalbos lentele. Pavyzdžiui, anglai naudoja 1252 kodų lentelę, nors angliškos raidės yra ir visose kitose lentelėse. Mat anglai linkę daugiau bendrauti su kaimynais, kurių kalbų raidės yra toje kodų lentelėje (prancūzais, vokiečiais). Pagaliau ir šiuolaikinė anglų kalba nėra tokia gryna, kaip mes manome. Joje vartojamos ne tik „tikros“ angliškos 26 raidės, bet ir keletas kitų kalbų (dažniausia prancūzų) raidžių,

kurios lieka iš tų kalbų perimtuose žodžiuose, įgijusiuose angliškas formas, bet išlaikiusiuose originalo abėcėlės raides, pavyzdžiui, „façade“, „belovèd“.

Vienų raidžių kodai yra tie patys visose ar keliose kodų lentelėse, kitų – skirtingi. Kai dokumentas popierinis, tai nesvarbu, iš kurios kodų lentelės raidė paimta. Svarbu raidės atvaizdas, o ne jos kodas. Kai naudojamas elektroninis dokumentas (pavyzdžiui, persiunčiamas elektroniniu paštu), tai reikia pasirinkti tos kalbos (valstybės) vartojamą kodų lentelę bei ją atitinkantį šriftą.

Raidės Đđ ir Đđ yra skirtingos, nors didžiųjų raidžių vaizdai sutampa. Raidę Đđ vartoja islandai ir fareriečiai, raidę Đđ – kroatai.

2 lentelėje pateiktos ir specifinės lietuviškos raidės, nes kai kurios iš jų (Ą, Č, ę, Š, Ž) vartojamos ir kitose kalbose (jų kodai yra kitose kodų lentelėse).

Paskutiniame 2 lentelės stulpelyje pateikti raidžių kodai Unikode. Šiame kode kiekviena raidė turi skirtingą kodą ir pagal jį galima viena-reikšmiškai identifikuoti kiekvieną raidę.

2 lentelė. Raidžių, kurių nėra anglų kalbos abėcėlėje, kodai

Pastaba. A – didžiųjų raidžių kodai, a – mažųjų raidžių kodai.

Raidė	1250 Vidurio Europa		1252 Vakarų Europa		1254 Pietų Europa		1257 Baltijos valstybės		Unikodas	
	A	a	A	a	A	a	A	a	A	a
Ąą	193	225	193	225	193	225	–	–	U+00C1	U+00E1
Åå	–	–	192	224	192	224	–	–	U+00C0	U+00E0
Ăă	195	227	–	–	–	–	–	–	U+0102	U+0103
ȂȂ	194	226	194	226	194	226	–	–	U+00C2	U+00E2
ȦȦ	–	–	197	229	197	229	197	229	U+00C5	U+00E5
ȦȦ	196	228	196	228	196	228	196	228	U+00C4	U+00E4
ȦȦ	–	–	195	227	195	227	–	–	U+00C3	U+00E3
ȦȦ	165	185	–	–	–	–	192	224	U+0104	U+0105

Raidė	1250 Vidurio Europa		1252 Vakarų Europa		1254 Pietų Europa		1257 Baltijos valstybės		Unikodas	
	A	a	A	a	A	a	A	a	A	a
ȦȦ	–	–	–	–	–	–	194	226	U+0100	U+0101
Ćć	198	230	–	–	–	–	195	227	U+0106	U+0107
Čč	200	232	–	–	–	–	200	232	U+010C	U+010D
Çç	199	231	199	231	199	231	–	–	U+00C7	U+00E7
Đđ	207	239	–	–	–	–	–	–	U+010E	U+010F
Đđ	208	240	–	–	–	–	–	–	U+0110	U+0111
Đđ	–	–	208	240	–	–	–	–	U+00D0	U+00F0
Éé	201	233	201	233	201	233	–	–	U+00C9	U+00E9
Èè	–	–	200	232	200	232	–	–	U+00C8	U+00E8
Êê	–	–	202	234	202	234	–	–	U+00CA	U+00EA
Ëë	204	236	–	–	–	–	–	–	U+011A	U+011B
Ėė	203	235	203	235	203	235	–	–	U+00CB	U+00EB
Êê	–	–	–	–	–	–	203	235	U+0116	U+0117
Ęę	202	234	–	–	–	–	198	230	U+0118	U+0119
Ėė	–	–	–	–	–	–	199	231	U+0112	U+0113
Ĝĝ	–	–	–	–	208	240	–	–	U+011E	U+011F
Ĝĝ	–	–	–	–	–	–	204	236	U+0122	U+0123
Ĭĭ	205	237	205	237	205	237	–	–	U+00CD	U+00ED
Ĭĭ	–	–	204	236	204	236	–	–	U+00CC	U+00EC
Ĭĭ	206	238	206	238	206	238	–	–	U+00CE	U+00EE
Ĭĭ	–	–	207	239	207	239	–	–	U+00CF	U+00EF
Ĭĭ	–	–	–	–	221	253	–	–	U+0130	U+0131
Ĭĭ	–	–	–	–	–	–	193	225	U+012E	U+012F
Ĭĭ	–	–	–	–	–	–	206	238	U+012A	U+012B
ŸŸ	221	253	221	253	–	–	–	–	U+00DD	U+00FD
ŸŸ	–	–	159	255	159	255	–	–	U+0178	U+00FF
ŸŸ	–	–	–	–	–	–	205	237	U+0136	U+0137
ŸŸ	197	229	–	–	–	–	–	–	U+0139	U+013A

Raidė	1250 Vidurio Europa		1252 Vakarų Europa		1254 Pietų Europa		1257 Baltijos valstybės		Unikodas	
	A	a	A	a	A	a	A	a	A	a
Łł	188	019	-	-	-	-	-	-	U+013D	U+013E
Łł	163	179	-	-	-	-	217	249	U+0141	U+0142
Łł	-	-	-	-	-	-	207	239	U+013B	U+013C
Ńń	209	241	-	-	-	-	209	241	U+0143	U+0144
Ńń	210	242	-	-	-	-	-	-	U+0147	U+0148
Ńń	-	-	209	241	209	241	-	-	U+00D1	U+00F1
Ńń	-	-	-	-	-	-	210	242	U+0145	U+0146
Ńń	211	243	211	243	211	243	211	243	U+00D3	U+00F3
Ńń	213	245	-	-	-	-	-	-	U+0150	U+0151
Ńń	-	-	210	242	210	242	-	-	U+00D2	U+00F2
Ńń	212	244	212	244	212	244	-	-	U+00D4	U+00F4
Ńń	214	246	214	246	214	246	214	246	U+00D6	U+00F6
Ńń	-	-	213	245	213	245	213	245	U+00D5	U+00F5
Ńń	-	-	216	248	216	248	-	-	U+00D8	U+00F8
Ńń	192	-	-	-	-	-	212	244	U+014C	U+014D
Ńń	216	224	-	-	-	-	-	-	U+0154	U+0155
Ńń	-	248	-	-	-	-	-	-	U+0158	U+0159
Ńń	140	-	-	-	-	-	170	186	U+0156	U+0157
Ńń	138	156	-	-	-	-	218	250	U+015A	U+015B
Ńń	170	154	138	154	138	154	208	240	U+0160	U+0161
Ńń	141	186	-	-	222	254	-	-	U+015E	U+015F
Ńń	222	157	-	-	-	-	-	-	U+0164	U+0165
Ńń	218	254	-	-	-	-	-	-	U+0162	U+0163
Ńń	219	250	218	250	218	250	-	-	U+00DA	U+00FA
Ńń	-	251	-	-	-	-	-	-	U+0170	U+0171
Ńń	-	-	217	249	217	249	-	-	U+00D9	U+00F9
Ńń	217	-	219	251	219	251	-	-	U+00DB	U+00FB
Ńń	220	249	-	-	-	-	-	-	U+016E	U+016F

Raidė	1250 Vidurio Europa		1252 Vakarų Europa		1254 Pietų Europa		1257 Baltijos valstybės		Unikodas	
	A	a	A	a	A	a	A	a	A	a
Ůů	-	252	220	252	220	252	220	252	U+00DC	U+00FC
Ůů	-	-	-	-	-	-	216	248	U+0172	U+0173
Ůů	143	-	-	-	-	-	219	251	U+016A	U+016B
Ůů	175	159	-	-	-	-	202	234	U+0179	U+017A
Ůů	142	191	-	-	-	-	221	253	U+017B	U+017C
Ůů	-	158	-	-	-	-	222	254	U+017D	U+017E
Ææ	-	-	198	230	198	230	175	191	U+00C6	U+00E6
Ůœ	-	-	140	156	140	156	-	-	U+0152	U+0153
Þþ	-	-	222	254	-	-	-	-	U+00DE	U+00FE
ß	-	223	-	223	-	223	-	223	-	U+00DF

DEVINTOJI LIETUVOS MOKSLEIVIŲ INFORMATIKOS OLIMPIADA

Valentina Dagienė, Viktoras Dagys

Devintosios Lietuvos moksleivių informatikos olimpiados, vykusios 1997–1998 mokslo metais, organizaciniais klausimais rūpinosi Lietuvos Respublikos švietimo ir mokslo ministerijos Informatikos ir prognozavimo centras, dalykiniais bei metodiniais klausimais – Matematikos ir informatikos institutas.

Informatikos ir prognozavimo centras (IPC) siuntė pranešimus švietimo skyriams, palaikė ryšius su rajonų bei miestų olimpiados vertinimo komisijomis, rūpinosi trečiojo etapo pirmosios ir antrosios dalies rengimu, ieškojo lėšų olimpiadai, vėliau informavo mokytojus apie olimpiados eigą bei rezultatus.

Matematikos ir informatikos instituto (MII) darbuotojai sukūrė algoritmavimo uždavinių, parengė testų, patikrino bei įvertino moksleivių trečiojo etapo (abiejų dalių) darbus, stovykloje mokė moksleivius pasirengti pasaulinei olimpiadai. Žurnale „Kompiuterija“ bei kompiuteriniame leidinyje „Veidrodis“ išspausdino dalykinių straipsnių apie olimpiados uždavinius bei jų sprendimus.

Devintoji Lietuvos moksleivių informatikos olimpiada buvo rengiama laikantis 1996 m. spalio 17 d. Švietimo ir mokslo ministerijos patvirtintų moksleivių informatikos olimpiados rengimo nuostatų.

Olimpiada organizuojama trimis etapais. Pirmasis rengiamas bendrojo lavinimo mokyklose, antrasis – kiekviename rajone. Trečiasis etapas susideda iš dviejų dalių. Pirmoji (atrankinė) dalis vyksta elektroni-

niu paštu, antroji (baigiamoji) dalis yra stacionarinė (ši kartą ji buvo surengta Panevėžyje).

Olimpiados dalyviai skirstomi į dvi amžiaus grupes: jaunesniųjų grupei priklauso mokiniai iki devintos klasės, vyresniųjų – dešimtokai, vienuoliktokai ir dvyliktokai.

Devintosios moksleivių informatikos olimpiados organizavimo grupę sudarė: Rima Čėsniene (IPC), Benas Budvytis (Panevėžio J. Balčikonio gimnazija), Geminas Grėblikas (Panevėžio Senvagės vid. m-kla), Ingrida Gurklytė (Panevėžio 20-oji vid. m-kla), Bronislovas Jakutonis (Panevėžio m. švietimo skyrius), Sonata Likienė (Panevėžio „Vyturio“ vid. m-kla), Birutė Savickienė (Panevėžio A. Lipniūno vid. m-kla), Tatjana Skusevičienė (IPC), Jolita Stankevičienė (Panevėžio 5-oji vid. m-kla).

Olimpiados uždavinius sudarė ir trečiojo etapo sprendimus tikrino Valentinos Dagienės (MII) vadovaujama vertinimo komisija: Jonas Blonskis (Kauno technologijos universitetas), Jūratė Bulotaitė (MII), Mikas Grigaliūnas (Vilniaus universitetas), Gintautas Grigas (MII), Tomas Laurinavičius (Vilniaus universitetas), Kristina Palaimaitė (MII), Jolita Stasiukaitytė (Vilniaus pedagoginis universitetas), Vladas Tumasonis (Vilniaus universitetas).

PIRMASIS ETAPAS

Pirmasis olimpiados etapas vyko mokyklose 1997 metų lapkričio mėnesį. Jį rengė mokyklų informatikos mokytojai, kurie parinko uždavinių, nustatė vertinimo kriterijus, patikrino moksleivių darbus, iš kurių geriausius atrinko tolesniam etapui. Nacionalinė olimpiados vertinimo komisija per rajonų ir miestų švietimo skyrius pasiūlė mokykloms pirmojo etapo uždavinių, tačiau jie buvo tik rekomendaciniai – mokytojai galėjo parinkti ir kitokių.

Pateikiame uždavinių, kuriuos nacionalinė olimpiados vertinimo komisija rekomendavo pirmajam etapui, sąlygas. (Uždavinių komentarai bei sprendimo idėjos buvo pateiktos kompiuterinio leidinio „Veidrodis“ 1997 metų aštuntame numeryje.)

Jaunesniųjų grupė

1.J1 uždavinys. Kaip pasiekti lemputę. Turime n dėžių, kurių briaunų $a \times b \times c$ ilgiai yra žinomi. Dėžes galima guldyti ant bet kurio šono. Kambaryje perdegė elektros lemputė. Atstumas nuo grindų iki jos yra t . Žmogus gali pasiekti tik h aukštyje kabančią lemputę.

Reikia surasti pirmąją iš duotų dėžių, kurią panaudojus galima pakesti lemputę nauja. Kiek iš viso yra tam tinkamų dėžių?

Pradiniai duomenys teisingi: visi skaičiai teigiami, $h \leq t$, dėžių briaunų ilgiai a , b , ir c neviršija $h/2$ (priešingu atveju žmogus negalės užlipti ant dėžės).

1.J2 uždavinys. Sklypas. Ūkininkas gavo žemės. Tačiau vietoj konkretaus sklypo jam buvo duota tvora, susidedanti iš n dalių (atkarpų), kiekviena – 10 metrų ilgio. Ūkininkui buvo pasakyta, kad sklypas, kurį jis apsitvers šia tvora, bus jo. Sklypo tvora turi būti laužtė, kurią sudarančios atkarpos eina dviem kryptimis: *šiaurė-pietūs* (arba *pietūs-šiaurė*) ir *rytai-vakarai* (arba *vakarai-rytai*). Kiekviena laužtės atkarpa sudaro ma iš sveikąjo tvoros dalių skaičiaus.

Parašykite programą, kuri nustatytų, kokio didžiausio ploto sklypą gali apsitverti ūkininkas. Sklypo dydį reikia išreikšti hektarais ir arais ($1 \text{ ha} = 100 \text{ a} = 10000 \text{ m}^2$).

Pavyzdys

Pradinis duomuo: 80

Rezultatas: 4 ha 0 a

1.J3 uždavinys. Elektroninis laikrodis. Skaitmenys elektroniniame laikrodyje yra sudaromi iš horizontalių ir vertikalų pagaliukų:

1234567890

Duotas horizontalių ir vertikalų pagaliukų skaičius. Parašykite algoritmą, kuris nustatytų, kokią laiką galėjo rodyti elektroninis laikrodis. Pradiniai duomenys tokie, kad sprendinys būtinai egzistuoja.

Vyresniųjų grupė

1.V1 uždavinys. Skaičių kriptogramos. Toliau pateiktoje kriptogramoje dauginami skaitmenys pažymėti žvaigždutėmis:

```
****
***
----
****
*****
*****
-----
*****
```

Parašykite programą, kuri rastų visus duotos kriptogramos sprendinius. Vieną sprendinį turi sudaryti trys skaičiai: dauginamasis, daugiklis ir sandauga.

1.V2 uždavinys. Dešimtainės trupmenos keitimas paprastąja. Parašykite programą, kuri dešimtainę trupmeną paverstų paprastąja ir išspausdintų formatu S/V , čia S – skaitiklis, V – vardiklis. Be abejo, trupmena turi būti supaprastinta.

Pavyzdžiai

$$0,25 = 1/4$$

$$0,375 = 3/8$$

$$4,4 = 22/5$$

$$0,3 = 3/10$$

1.V3 uždavinys. Kubelis su skaičiais. Ant stalo stovi kubelis, ant kurio kiekvienos sienelės užrašyta po vieną sveikąjį skaičių. Ant viršutinės sienelės užrašytas skaičius a , ant apatinės – b , ant priekinės – c , ant užpakalinės – d , ant kairiosios – e , ant dešinėsios – f .

Yra keturios kubelio vartymo komandos: „K“ – kubelis apverčiamas ant kairiojo šono, „D“ – apverčiamas ant dešiniojo šono, „P“ – apverčiamas priekine puse, „A“ – apverčiamas galine sienele.

Duota kubelio vartymo komandos (raidžių be tarpų seka). Parašykite algoritmą, nustatantį, koks skaičius bus matomas ant kubelio viršutinės sienelės, kai atliksime duotąją komandų seką.

ANTRASIS ETAPAS

Antrąją olimpiados etapą 1997 metų gruodžio mėnesį rengė rajonų bei miestų švietimo skyriai. Jie paskyrė olimpiados organizavimo ir vertinimo komisijas, sudarytas iš vietinių mokytojų. Rajonų (miestų) olimpiados organizavimo ir vertinimo komisijos turėjo parinkti uždavinių, nustatyti algoritmų pateikimo būdus (algoritmines kalbas) bei vertinimo kriterijus, patikrinti ir įvertinti moksleivių darbus, o taip pat išsiųsti nacionalinei olimpiados organizavimo grupei pirmojo ir antrojo etapų ataskaitą bei nugalėtojų sąrašus. Šios komisijos, atsižvelgdamos į olimpiados rezultatus, atrinko ir pasiūlė moksleivius į trečiojo etapo pirmąją dalį, kuri vyksta elektroniniu paštu.

Antrojo etapo vyresniųjų grupėje iš viso dalyvavo 858 moksleiviai, jaunesniųjų – 130. Iš miestų, kaip paprastai, gausumu pasižymėjo Vilnius (168), Kaunas (113), Panevėžys (55), o iš rajonų: Šilalės (22), Šiaulių (21), Ignalinos, Molėtų, Rokiškio (po 16), Radviliškio, Šakių, Telšių, Vilniaus, Zarasų (po 15). Rajonų (miestų) olimpiados organizavimo ir vertinimo komisijos trečiajame etape dalyvauti pasiūlė 191 vyresniųjų grupės ir 43 jaunesniųjų grupės moksleiviams.

Rajonų bei miestų olimpiados organizavimo ir vertinimo komisijos galėjo pačios sudaryti bei parinkti uždavinius. Tačiau toms, kurios pagirdavo, nacionalinė vertinimo komisija, kaip ir pirmajame etape, pasiūlė uždavinių rinkinį. Pateikiame abiejų grupių (jaunesniųjų ir vyresniųjų) uždavinių sąlygas. (Kai kurių uždavinių sprendimo idėjos buvo pateiktos kompiuterinio leidinio „Veidrodis“ 1998 metų pirmame numeryje.)

Jaunesniųjų grupė

2.J1 uždavinys. *Sultis gerti sveika.* Kiekvienos savaitės pirmadienio rytą Jonas gauna kišenpinigių (k centų). Vienas butelis sulčių kainuoja s centų, o tušti buteliai superkami po b centų.

Kartą (tai buvo i -toji savaitės diena), iš viso turėdamas n centų, Jonas nusprendė kasdien (pradedant i -tąja diena) pirkti sulčių už visus tu-

rimus pinigus. Pinigai, gauti pardavus butelius, bus panaudojami sultims pirkti kitą dieną. Šitaip Jonas darys tol, kol jis įstengs nusipirkti bent vieną sulčių butelį. Kiek butelių sulčių išgers Jonas?

Parašykite programą šiam uždaviniui išspręsti. Pradiniai duomenys – natūralieji skaičiai k, s, b, n, i ($1 \leq i \leq 7$). Jei nusprendžiama sultis pirkti pirmadienį, tai iš karto gaunami kišenpinigiai (pasipildo turima pinigų suma). Už pirmadienį gautus kišenpinigius sultys perkamos pirmadienį.

Pavyzdys

Pradiniai duomenys: $k=10, s=3, b=1, i=6, n=12$.

Rezultatas: 10.

2.J2 uždavinys. *Skaitmuo sekoje.* Sakykime, kad turime seką, sudarytą iš paciliui einančių Fibonačio skaičių (be tarpų):

1123581321345589144233...

Parašykite programą šios sekos k -ajam skaitmeniui rasti.

Fibonačio skaičių seka – tai tokia seka, kurios pirmasis ir antrasis nariai vienetai, o kiekvienas tolesnis yra lygus prieš jį einančių dviejų narių sumai.

Pavyzdys

Pradinis duomuo: 15.

Rezultatas: 8.

2.J3 uždavinys. *Skaičiaus spėjimas.* Yra toks žaidimas, kai pirmasis žaidėjas sugalvoja sveikąjį skaičių, didesnę už nulį, bet mažesnę už $maxint$, o antrasis, sakydamas po vieną skaitmenį, bando atspėti sugalvotąjį skaičių.

Duotas sugalvotas skaičius ir šešių skaitmenų seka. Toji seka – tai antrojo žaidėjo spėti skaitmenys. Visi sekos skaitmenys skirtingi. Parašykite programą nustatyti, ar antrasis žaidėjas, pasakęs duotus šešis skaitmenis, atspėjo sugalvotąjį skaičių.

Jei skaičius atspėtas, tai rezultatas turi būti žodis „TAIP“ ir skaičius, kuris rodo, kelių bandymų (skaitmenų) pakako duotajam skaičiui

atspėti. Jei skaičius nebuvo atspėtas, tai rezultatas – žodis „NE“ ir skaičius, reiškiantis, kiek liko skirtingų neatspėtų skaitmenų.

Pavyzdžiai

Pradiniai duomenys: 15456
0 4 5 6 1 3

Rezultatas: TAIP 5
{ nes pasakius skaitmenis 0, 4, 5, 6, 1 skaičius buvo atspėtas }

Pradiniai duomenys: 3341
9 8 4 7 1 5

Rezultatas: NE 1
{ nes neatspėti liko du trejetai, tačiau jie atitinka vieną skaitmenį }

Pradiniai duomenys: 381
9 4 2 3 5 7

Rezultatas: NE 2
{ nes neatspėti liko du skirtingi skaitmenys: 8 ir 1 }

Vyresniųjų grupė

2.V1 uždavinys. Abėcėlė. Laikant informatikos brandos egzaminą būna uždavinių, kuriuose reikia surasti visus galimus duotos abėcėlės žodžius. Pavyzdžiui: „Abėcėlė turi keturis simbolius: A, B, C, D. Parašykite visus galimus trijų ženklų žodžius, sudarytus iš šios abėcėlės simbolių.“ Parašykite programą apibendrintam uždaviniui spręsti.

Duota n simbolių abėcėlė, kurią sudaro tik didžiosios lietuvių kalbos abėcėlės raidės. Reikia rasti visus galimus k ženklų žodžius, sudarytus iš šios abėcėlės simbolių.

Pradiniai duomenys turi būti skaitomi iš tekstinės bylos. Pirmoje bylos eilutėje įrašytas abėcėlę sudarančių simbolių skaičius n , tolesnėse n eilučių – patys abėcėlės simboliai. Paskutinėje eilutėje pateikiamas žodžių ilgis k .

Rezultatus įrašykite į tekstinę bylą.

2.V2 uždavinys. Nuliai skaičiaus gale. Duotas dešimtainis natūralusis skaičius n . Parašykite programą rasti tokiai skaičiavimo sistemai, kuria užrašius duotąjį skaičių, jo gale gautume kuo daugiau nulių. Jei gali būti keli variantai, užtenka pateikti vieną.

Pavyzdžiai

Pradinis duomuo: 4
Rezultatas: 2 { nes $4_{10} = 100_2$ }

Pradinis duomuo: 5
Rezultatas: 5 { nes $5_{10} = 10_5$ }

2.V3 uždavinys. Žodynas. Kaip visi žinome, žodžiai žodynuose surikiuoti pagal abėcėlę. Pateiktame sąraše lietuviškosios abėcėlės raidės surašytos prioriteto mažėjimo tvarka (vienoje eilutėje esančios raidės turi tą patį prioritetą):

A, Ą	I, ĩ, Y	S
B	J	Š
C	K	T
Č	L	U, ū, Ű
D	M	V
E, ė, Ė	N	Z
F	O	Ž
G	P	
H	R	

Du žodžiai lyginami tokiu būdu: imamos jų pirmosios raidės ir žiūrimi prioritetai. Jei jie nesutampa, tai žodyne pirmiau eis žodis, turintis didesnio prioriteto raidę. Jei prioritetai lygūs, tai pereiname prie antroje pozicijoje esančių simbolių ir kartojame tuos pačius veiksmus. Jei vienas žodis yra kito žodžio pradžia, tai pirmesnis turi būti trumpesnis žodis.

Reikia parašyti programą, kuri iš tekstinės bylos perskaitytų žodžius, juos surikiuotų abėcėlės tvarka ir sutvarkytą žodžių sąrašą įrašytų į tekstinę bylą. Kiekvienoje pradinių duomenų bylos eilutėje yra po vieną žodį, kurio ilgis didesnis nei dvidešimt simbolių. Žodžiai sudaryti tik

iš didžiųjų lietuvių kalbos abėcėlės raidžių. Byloje įrašyta ne daugiau kaip šimtas žodžių (vienodų nėra). Rezultatų byloje kiekvienas žodis rašomas į atskirą eilutę.

Pavyzdys

Pradiniai duomenys:

ASTA
AŠMENYS
GYVASIS
AŽUOLYNAS
GYVAS
AŠA
ATŽALYNAS

Rezultatai:

AŠA
ASTA
AŠMENYS
ATŽALYNAS
AŽUOLYNAS
GYVAS
GYVASIS

2.V4 uždavinys. Šešiolyktainių skaičių kodavimas (teorinis uždavinys, sprendžiamas be kompiuterio). Šešiolyktainiai skaitmenys yra 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Visi skaitmenys šešiolyktainio skaičiaus užrašė vienodai tikėtini. Nuliai skaičiaus pradžioje nerašomi.

Sveikojo teigiamo šešiolyktainio skaičiaus užrašą reikia užkoduoti tik dešimtainiais skaitmenimis: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Kodavimo taisyklės:

- 1) kiekvieną šešiolyktainio skaičiaus skaitmenį reikia koduoti atskirai, t. y. viso šešiolyktainio skaičiaus negalima paversti dešimtainiu;
- 2) vieną šešiolyktainio skaičiaus skaitmenį galima koduoti dviem dešimtainiais skaitmenimis.

Kodavimo reikalavimai:

- 1) koduotas skaičiaus užrašas turi būti vienareikšmiškai dekoduojamas;
- 2) koduoto skaičiaus ilgis turi būti minimalus.

Sugalvokite kodavimo būdą ir įrodykite, kad jis tikrai tenkina keliamus reikalavimus.

TREČIOJO ETAPO PIRMOJI DALIS

1998 m. kovo 14 dieną nacionalinės olimpiados organizavimo grupės paskirtuose Lietuvos mokyklų ar kitų švietimo įstaigų kompiuterių centruose buvo surengta Devintosios moksleivių informatikos olimpiados trečiojo etapo pirmoji dalis.

Olimpiados trečiojo etapo pirmosios dalies uždavinius sprendė apie 200 moksleivių (tikslų skaičių gana sunku nustatyti, nes mokytojai, pristatydami sąrašus, ne visada nurodė, ar mokinys neatvyko į olimpiadą, ar dalyvavo, bet nepateikė nė vieno uždavinio sprendimo).

Ir jaunesniųjų, ir vyresniųjų moksleivių grupės gavo po du skirtingo sudėtingumo uždavinius, kuriems išspręsti buvo skirtos penkios valandos. Baigus darbą, mokytojai, atsakingi už olimpiadą, suarchyvuotus mokinių sprendimus išsiuntė elektroniniu paštu į Informatikos ir prognozavimo centrą.

Uždaviniai buvo vertinami įprastai: didžiausias taškų skaičius skiriamas už programos teisingumą, dalį taškų buvo galima gauti už uždavinio sprendimo idėjos aprašymą bei algoritmavimo kultūrą. Vertinimo schema buvo vienoda abiems grupėms. Kiekvienas uždavinys buvo vertinamas 100 taškų. Vienam uždaviniui šie taškai buvo skiriami už programos teisingumą (90 taškų) ir algoritmavimo kultūrą (10 taškų), kitam – už programos teisingumą (70 taškų) ir už sprendimo idėjos aprašą (30 taškų).

Pateikiame trečiojo etapo pirmosios dalies uždavinių sąlygas.

Jaunesniųjų grupė

3.1.J1 uždavinys. Siųstuvai ir imtuvas (vykdymo laikas – ne daugiau kaip 5 sek. kompiuteriu su *Pentium* 133 MHz procesoriumi).

Į vieną imtuvą signalus siunčia n siųstuvų ($1 \leq n \leq 20$). Siųstuvas k skleidžia signalus kas T_k ($1 \leq T_k \leq 100$) sekundžių, o šie imtuvą pasiekia po L_k ($1 \leq L_k \leq 50$) sekundžių. Taigi kiekvieno siųstuvo darbą aprašo du skaičiai. Pavyzdžiui, $T_k = 3$, $L_k = 2$. Jei siųstuvas išsiųs signalus laiko momentais 1 ir 4, tai imtuvas gaus signalus laiko momentais 3 ir 6.

Gali atsitikti taip, kad tuo pačiu metu į imtuvą ateina keli signalai iš karto. Gautasis signalas (kelių signalų mišinys) vadinamas blogu signalu.

Tarkime, kad visi siųstuvai pirmąjį signalą pasiuntė vienu metu ir veikia toliau.

Parašykite algoritmą, kuris nustatytų, kiek procentų visų signalų, kuriuos gauna imtuvas, yra blogi.

Pradiniai duomenys skaitomi iš bylos RYSYS.DAT. Pirmoje eilutėje įrašytas siųstuvų skaičius n . Tolesnėse n eilučių yra po du skaičius, aprašančius kiekvieno siųstuvo darbą. Pirmasis skaičius eilutėje yra T_k , antrasis – L_k .

Rezultatą apvalinkite iki sveikojo skaičiaus, jį spausdinkite displejaus ekrane.

Pastaba: jokie veiksmai ekrane (pavyzdžiui, ekrano valymas) negali būti atliekami, taip pat iš vartotojo negali būti reikalaujama atlikti kokių nors veiksmų (pavyzdžiui, paspausti klavišą).

Pavyzdys

RYSYS.DAT	Rezultatas
3	40
2 2	
3 5	
3 4	

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 70 taškų; už sprendimo idėjos aprašymą – iki 30 taškų;

3.1.J2 uždavinys. *Pokeris* (vykdymo laikas – ne daugiau kaip 5 sek. kompiuteriu su *Pentium* 133 MHz procesoriumi).

Šis kortų žaidimas dažnai sutinkamas lošimų automatuose. Nagrinėsime supaprastintą pokerio variantą. Automatas iš 24 kortų kaladės (nuo devynakės iki tūzo) atsitiktinai parenka lošėjui penkias kortas. Žaidėjas nusprendžia, kurias keturias iš penkių kortų jis norėtų pasilikti, o kurią vieną pakeisti (kortą keisti privaloma). Nurodžius keičiamą kortą, automatas įdeda ją į kaladę ir iš jos atsitiktinai išrenka vieną kortą.

Nuo turimų penkių kortų priklauso išlošimas. Išlošiama, jei iš turimų kortų galima sudaryti kurią nors iš žemiau išvardintų kombinacijų.

1. *Iš eilės* (pavyzdžiui, devynakė, dešimtakė, valetas, dama ir karalius; kortų išsidėstymo tvarka nesvarbi).
2. *Pora* (dvi vienodo rango kortos, pavyzdžiui, dvi damos).
3. *Dvi poros* (pavyzdžiui, dvi dešimtakės ir du tūzai).
4. *Trejetas* (trys vienodo rango kortos, pavyzdžiui, trys devynakės).
5. *Ketvertas* (keturios vienodo rango kortos, pavyzdžiui, keturi karaliai).
6. *Pilni namai* (juos sudaro anksčiau minėti *Pora* ir *Trejetas*).
7. *Vienos spalvos kortos* (pavyzdžiui, visi kryžiai).
8. *Visos kortos* (visos kortos išdėstomos iš eilės ir yra vienos spalvos).

Daugeliu atvejų galima sudaryti kelias kombinacijas. Pavyzdžiui, jei turime *Ketvertą*, galime sudaryti ir *Porą*, ir *Trejetą*. Tokiu atveju laikykite, kad susidaro tik viena išlošiama situacija – ta, kurios numeris (prioritetas) didžiausias.

Kortų kombinacijos pasitaikymo *tikimybė* yra galimų skirtingų kortų kombinacijų, kuriose yra norima kombinacija, skaičiaus santykis su visų galimų kombinacijų skaičiumi.

Pavyzdys. Tarkime, jūs pasilikote dvi damas ir du karalius. Atsitiktinai ištraukus penktą kortą galite gauti arba tą pačią situaciją (*Dvi poros*) arba *Pilnus namus*.

Kaip sužinoti, kokia tikimybė susidaryti *Pilniems namams* atsitiktinai ištraukus penktą kortą? Kadangi iš 24 kortų kaladės išimtos dvi damos ir du karaliai (t. y. keturios kortos), tai penktą kortą galime ištraukti $24 - 4 = 20$ būdų, taigi iš viso yra 20 galimų skirtingų kombinacijų. Tačiau *Pilni namai* susidarys tik tada, jei ištraukta korta bus arba dama, arba karalius. Kadangi tarp kaladėje likusių 20 kortų yra dvi damos ir du karaliai, tai yra $2 + 2 = 4$ galimos skirtingos kombinacijos, kada gausime *Pilnus namus*. Taigi tikimybė, kad susidarys *Pilni namai* bus $4 / 20 = 0,2$.

O štai tikimybė, kad pasilikus dvi damas ir du karalius ir atsitiktinai ištraukus penktą kortą turėsime tas pačias *Dvi poros* lygi 0,8. Tą ne-

sunku apskaičiuoti. Mat visų galimų situacijų tikimybių suma visada lygi 1. O šiuo atveju galimos tik dvi situacijos.

Parašykite programą, kuri pagal duotas penkias kortas nuspręstų, kurią kortą reikia pakeisti, kad tikimybė sudaryti kurią nors išlošiamą kombinaciją būtų didžiausia bei rastų išlošiamos kombinacijos numerį ir tą tikimybę.

Jei galimi keli sprendimai, pateikite vieną.

Pradiniai duomenys įrašyti byloje POKERIS.DAT. Pirmose penkiose eilutėse įrašyti automato išrinktų kortų kodai. Kortos koduojamos viena didžiąja raide ir vienu skaičiumi:

- raidė nurodo kortos spalvą: Č – čirvai, B – būgnai, K – kryžiai, V – vynai;
- skaičius nurodo kortos rangą: 0 – devynakė, 1 – dešimtakė, 2 – valetas, 3 – dama, 4 – karalius, 5 – tūzas.

Eilutėse nėra tarpų, visos kortos yra skirtingos.

Rezultatai rašomi į bylą POKERIS.REZ. Pirmoje eilutėje turėtų būti įrašyta keičiama korta, antroje – išlošiamos kombinacijos numeris, trečiojoje – tikimybė sudaryti tą kombinaciją.

Pastaba: jokie veiksmai ekrane (pavyzdžiui, ekrano valymas) negali būti atliekami, taip pat iš vartotojo negali būti reikalaujama atlikti kokių nors veiksmų (pavyzdžiui, paspausti klavišą).

Duomenų pavyzdys

POKERIS.DAT	POKERIS.REZ
K4	V1
V4	2
V1	0.8
B3	
Č3	

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų; už algoritmavimo kultūrą – iki 10 taškų.

Vyresniųjų grupė

3.1.V1 uždavinys. Siųstuvas ir imtuvas (vykdymo laikas – ne daugiau kaip 60 sek. kompiuteriu su Pentium 133 MHz procesoriumi).

Į vieną imtuvą signalus siunčia n siųstuvų ($1 \leq n \leq 20$). Siųstuvas k skleidžia signalus kas T_k sekundžių, o šie imtuvą pasiekia po L_k sekundžių. Kiekvienas siųstuvas turi pasiųsti pranešimą iš P_k signalų. Taigi kiekvieno siųstuvo darbą apibūdina trys skaičiai. Pavyzdžiui, $T_k = 3, L_k = 2, P_k = 2$. Jei siųstuvas įjungiamas laiko momentu 0, jis išsiųs du signalus laiko momentais 0 ir 3, o imtuvas gaus signalus laiko momentais 2 ir 5.

Gali atsitikti taip, kad tuo pačiu metu į imtuvą ateina keli signalai iš karto. Gautasis signalas (kelių signalų mišinys) vadinamas blogu signalu.

Siųstuvas gali būti įjungti arba išjungti. Kai siųstuvas pradeda veikti (t. y. įjungiamas), jis tuoj pat išsiunčia savo pranešimą ir tada išjungiamas.

Momentas, kada paleidžiamas(i) dirbti pirmasis(ieji) siųstuvas(i), vadinamas *nulinio* laiko momentu (nuo tada pradedamas skaičiuoti laikas). Darbo pabaiga laikomas momentas, kai imtuvas gauna paskutinįjį signalą.

Parašykite algoritmą, kuris nustatytų, kuriuo laiko momentu reikia įjungti kiekvieną siųstuvą, kad imtuvas negautų blogų signalų, o visi pranešimai būtų perduoti per trumpiausią laiką.

Pastaba. Laikas, per kurį siųstuvas perduoda pranešimą imtuvui, ne ilgesnis nei du šimtai sekundžių. Pradedama skaičiuoti nuo siųstuvo įjungimo momento ir baigiama, kai imtuvas gauna paskutinį pranešimo signalą.

Pradiniai duomenys skaitomi iš bylos RYSYS.DAT. Pirmoje eilutėje įrašytas siųstuvų skaičius n . Tolesnėse n eilučių yra po tris skaičius, aprašančius kiekvieno siųstuvo darbą. Pirmas skaičius eilutėje yra T_k , antrasis – L_k , o trečiasis – P_k .

Rezultatą reikia įrašyti į bylą RYSYS.REZ. Joje turi būti n eilučių, kiekvienoje jų – po vieną skaičių. Šis skaičius rodo, kuriuo laiko momentu (kurią sekundę) reikia įjungti kiekvieną siųstuvą. Pirmą rezultatų bylos eilutę atitiks pirmąjį siųstuvą, įrašytą antroje pradinių duome-

nų bylos eilutėje, antra – antrąjį siustuvą, ir t. t. Jei galimi keli atsakymo variantai, užtenka pateikti bet kurį vieną.

Pavyzdys

RYSYS.DAT	RYSYS.REZ
3	4
2 4 3	0
5 1 3	0
3 2 2	

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų; už algoritmovimo kultūrą – iki 10 taškų.

3.1.V2 uždavinys. Trys vienodi skaitmenys (vykdymo laikas – ne daugiau kaip 50 sek. kompiuteriu su Pentium 133 MHz procesoriumi).

Yra tokių įdomių skaičių, kurie gali būti išreikšti dviejose skirtingose skaičiavimo sistemose trimis vienodais skaitmenimis, pavyzdžiui:

$$273 = 333_9 = 111_{16}.$$

Parašykite algoritmą visiems intervalo $[m, n]$ skaičiams, pasižymintiems šia savybe, rasti ($1 \leq m \leq n \leq 10000$).

Pradiniai duomenys – natūralieji skaičiai m ir n pirmoje bylos VIENODI.DAT eilutėje.

Rezultatai rašomi į bylą VIENODI.REZ. Pirmoje eilutėje rašomas dešimtainis skaičius, kuris tenkina uždavinio sąlygą.

Kitose dviejose eilutėse – šio skaičiaus užrašas dviem skaičiavimo sistemomis.

Skaičius kuria nors skaičiavimo sistema (nedešimtaine) užrašomas griežtai laikantis šių taisyklių: pradžioje – sistemos pagrindas, po to palikus vieną tarpą – skaičiaus užrašas šia sistema (tarp skaitmenų paliekama po vieną tarpą). Jei skaičiavimo sistema didesnė už dešimtainę, dalis skaitmenų užrašomi dviem dešimtainiais skaitmenimis.

Ketvirtoje bylos eilutėje rašomas tolesnis sąlygas tenkinantis skaičius ir t. t.

Jei duotame intervale nėra nė vieno skaičiaus, pasižymintio nurodyta savybe, į rezultatų bylą rašomas nulis.

Pavyzdys

VIENODI.DAT	VIENODI.REZ	Komentaras
152 543	273	$273_{10} = 333_9 = 111_{16}$
	9 3 3 3	
	16 1 1 1	

Sprendimas vertinamas 100 taškų: už sprendimo idėjos aprašymą – iki 30 taškų; už programos teisingumą – iki 70 taškų.

TREČIOJO ETAPO ANTROJI DALIS

1998 m. balandžio 14–18 d. Panevėžyje vyko Devintosios moksleivių informatikos olimpiados finalas – trečiojo etapo antroji dalis. Čia varžėsi trisdešimt vienas vyresniųjų grupės (dešimtų – dvyliktų klasių) ir penkiolika jaunesniųjų grupės moksleivių.

Į baigiamąsias informatikos varžybas susirinko moksleivių iš įvairių Lietuvos vietovių. Daugiausia (net keturiolika) buvo vilniečių (po 7 jaunesniųjų ir vyresniųjų grupėje), po septynis atvyko iš Kauno ir Panevėžio, trys – iš Visagino, po du – iš Marijampolės miesto bei Telšių ir Šakių rajonų, o iš visur kitur (Druskininkų m., Ignalinos, Kretingos, Rokiškio, Skuodo, Šiaulių, Šilalės, Širvintų, Švenčionių raj.) – po vieną moksleivį.

Moksleiviai uždavinius sprendė Panevėžio J. Balčikonio gimnazijoje ir Panevėžio „Vyturio“ vidurinėje mokykloje. Laisvalaikio jiems buvo surengtos ekskursijos po miestą ir jo apylinkes, į Panevėžio stiklo fabriką ir Krašto apsaugos ministerijos Karinių oro pajėgų antrąją aviabazę.

Olimpiados organizavimu rūpinosi Panevėžio miesto švietimo skyrius bei kitos vietos valdžios įstaigos. Olimpiadą parėmė, skirdamos prizus moksleiviams ar prisidėdamos prie laisvalaikio renginių, Vilniaus „Ogmios sistemos“, „Sigmanta“ Panevėžio „Ciklas“, „Kompiuterių centras“, „Odris“, „Elsis“, „Skorpiono takas“, kompiuterių bendrovės

bei kitos įmonės („Biznio kontaktai“, „Coca-Cola“, „Comliet“, „Kalnapilis“, „Sema“), Taupomasis bankas, J. Masiulio knygynas, parduotuvės „Pas Juozapą“, „Sąvaržėlė“.

Devintosios informatikos olimpiados baigiamoji dalis savo struktūra buvo visiškai panaši į praeitų metų: iš ryto penkias valandas sprendžia vyresnieji, po pietų – jaunesnieji. Atitinkamai vyko ir jų sprendimų testavimas.

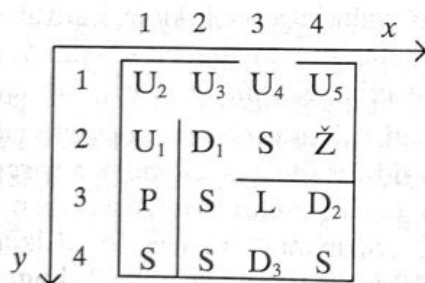
Dalis uždavinių buvo klasikinio stiliaus, reikėjo mokėti apdoroti grafus. Keletas uždavinių buvo įdomesni, negirdėto pas mus pobūdžio. Štai kodavimo uždavinys. Buvo duota programos vykdomoji (.exe) byla. Pagal jos darbo rezultatus reikėjo nustatyti, kaip ši programa užkoduoja ir iškoduoja tekstą. Tas pats uždavinys buvo duotas abiejų grupių moksleiviams, koduojančios programos skyrėsi tik sudėtingumu.

Pateikiame trečiojo etapo antrosios dalies uždavinių sąlygas.

Jaunesniųjų grupė

3.2.J1 uždavinys. Lobiai ieškojimas (vykdymo laikas – ne daugiau kaip 5 sek.). Tarkime, turime 4×4 laukelių labirintą. Keliautojo tikslas – kuo greičiau rasti labirinte paslėptą lobį ir iš labirinto išeiti. Į labirintą galima patekti tik pro vieną kurios nors išorinės sienos plyšį – tai ir įėjimas, ir išėjimas. Būdamas labirinte (t. y. kuriame nors laukelyje) keliautojas gali eiti bet kuria iš keturių kryptų: į viršų, į apačią, į kairę, į dešinę, žinoma, jei tik netrukdo sienos. Įstrižai eiti negalima.

Labirinto pavyzdys



Kiekvienas labirinto laukelis yra tam tikros rūšies. Iš viso esama šešių rūšių laukelių.

1. Sausuma S. Tokių laukelių labirinte yra penki.
2. Trys duobės D₁, D₂, D₃. Jei žaidėjas ateina į laukelį D₁, tai jis permestas į laukelį D₂ ir iš ten atlieka tolesnį ėjimą. Analogiškai patekęs į laukelį D₂, žaidėjas permetamas į D₃, o atsідūręs laukelyje D₃, patenka į laukelį D₁.
3. Pelkė P. Ji viena. Pelkė prasideda upė.
4. Žiotys Ž. Jos vienos – jomis baigiasi upė.
5. Upė. Ją sudaro penki laukeliai U₁, U₂, U₃, U₄, U₅. Upėje atsідūrusį keliautoją neša srovė. Iš laukelio U₁ jis patenka į U₄, iš U₂ – į U₅. Jei keliautojas patenka į likusius tris upės laukelius, jis nunešamas į žiotis Ž. Toliau keliautojas turės tęsti kelionę iš laukelio, į kurį jį atnešė srovė. Visi upės laukeliai susisiekiantys, t. y. U₁ turi bendrą kraštinę su U₂, U₂ su U₃, ir t. t. Pelkė P susisiekia su U₁, o žiotys Ž – su U₅.
6. Lobis L. Laikoma, jog lobis surastas, kai žaidėjas atsistoja laukelyje L.

Labirinte yra dvi tiesios sienos: viena trijų laukelių ilgio, kita – dviejų (jos eina laukelių pakraščiu). Sieną niekuomet nekerta upės, neatiskiria pelkės arba žiočių nuo upės. Be to, abi sienos negali viena su kita liestis. Vienas sienos galas visada liečiasi su labirinto išorine siena.

Keliautojas turi du užtaisus sienoms sprogdinti. Sienos sprogdinimas prilygsta vienam ėjimui. Išsprogdinęs skylę sienoje, keliautojas gali pro ją pralįsti. Sprogdinant sieną, sugriaunamas tik vienas sienos laukelis, o ne visa siena. Susprogdintos sienos neatstatomos. Draudžiama sprogdinti išorines labirinto sienas, t. y. pasidaryti išėjimą. Keliautojas gali ir nepanaudoti sprogdinimo arba panaudoti tik vieną.

Sprogdinti galima iš bet kurių laukelių. Jei sprogdinama iš laukelių, į kuriuos keliautojas buvo permestas ar jį į ten atnešė srovė, po sprogdinimo žaidėjas pasilieka tame pačiame laukelyje. Pirmu ėjimu po sprogdinimo keliautojas gali lįsti pro skylę arba eiti į kitą gretimą laukelį (t. y. jis nebus iš karto permestas į kitą duobę bei jo nuneš srovė). Beje, tai yra vienintelis atvejis, kada žaidėjas pasilieka tame pačiame laukelyje.

Iėjimas į labirintą bei išėjimas iš labirinto laikomas ėjimu. Permetimas iš vienos duobės į kitą arba srovės nunešimas nelaikomas ėjimu.

Sudarykite programą, kuri pagal duotą labirintą nustatytų, kiek mažiausiai ėjimų reikia, kad keliautojas rastų lobį ir išeitų iš labirinto.

Pradiniai duomenys skaitomi iš bylos LABIR.DAT. Pirmoje eilutėje pateikti du skaičiai – laukelio, į kurį patenkama įėjus į labirintą, koordinatės (x, y). Tolesnėse 16 eilučių įrašyta po du skaičius – visų labirinto laukelių koordinatės (x, y). Koordinatės išdėstytos tokia tvarka: P, U₁, U₂, U₃, U₄, U₅, Ž, penkių S laukelių koordinatės, D₁, D₂, D₃, L.

Paskutinėse dviejose bylos eilutėse yra po tris skaičius. Pirmoje iš jų aprašyta dviejų laukelių ilgio siena, antroje – trijų. Pirmasis skaičius eilutėje parodo, su kuria labirinto išorine siena liečiasi vidinė siena: skaičius 1 nurodo lietimąsi su viršutine siena, 2 – su dešiniąja, 3 – su apatine, 4 – su kairiąja. Kiti du skaičiai nusako, tarp kurių eilučių ar stulpelių yra siena (žr. toliau pateiktą pavyzdį).

Pradiniai duomenys tokie, kad sprendinys visuomet egzistuoja.

Rezultatas – minimalus ėjimų skaičius – spausdinamas ekrane.

Pavyzdys (atitinka paveikslėlį)

LABIR.DAT	Rezultatas
3 1	8
1 3	
1 2	
1 1	
2 1	
3 1	
4 1	
4 2	
3 2	
2 3	
1 4	
2 4	
4 4	

2 2	
4 3	
3 4	
3 3	
2 2 3	
3 1 2	

Komentaras

Rezultatas atspindi vieną iš galimų ėjimų variantų:

- 1) pro įėjimą patenkama į U₄ ir upė nuneša į žiotis Ž,
- 2) sprogdinama siena tarp Ž ir D₂,
- 3) iš Ž pro skylę sienoje einama į D₂, keliautojas permetamas į D₃,
- 4) iš D₃ einama į L; lobis surastas,
- 5) iš L einama į D₃, patenkama į D₁,
- 6) sprogdinama siena iš D₁ į U₁,
- 7) einama iš D₁ į U₁, upė nuneša į U₄,
- 8) išeinama iš labirinto.

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų; už algoritmavimo kultūrą – iki 10 taškų.

3.2.J2 uždavinys. Netvarkingas bibliotekininkas (vykdymo laikas – ne daugiau kaip 5 sek.). Bibliotekoje yra leidinių, kurie susideda iš ke-
lių tomų. Tačiau šie tomai lentynose sumaišyti su kitų leidinių tomiais, pavyzdžiui:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	1	1	2	2	3	2	3	4			
1	3	1	2	4	2	3	2	4	1	1			

Skaičiai virš lentynos žymi vietą lentynoje, skaičiai pirmoje eilutėje rodo leidinio numerį, antroje – leidinio tomo numerį.

Knygų bibliotekininkas sutvarkyti nenori, tačiau nori, kad būtų lengviau jas rasti. Susiradęs kiekvieno leidinio pirmą tomą, pasižiūrėjo, kiek tomų sudaro leidinį, ir įsidėmėjo pirmojo tomo vietą lentynoje, t. y. susidarė lentelę. Anksčiau pateiktam pavyzdžiui lentelė atrodytų šitaip.

Pirmoji lentelė

Leidiny	Pirmojo tomo vieta	Tomų skaičius
1	1	4
2	3	4
3	10	2
4	11	1

Po to bibliotekininkas sudarė dar vieną lentelę, turinčią tiek laukelių, kiek yra vietų lentynoje. Į kiekvieną lentelės laukelį jis įrašė tam tikrą skaičių. Jei vieta lentynoje tuščia, į lentelės laukelį įrašė nulį. Jei toje vietoje padėtas kurio nors leidinio tomas, į lentelę įrašoma vieta, kurioje padėtas tolesnis to paties leidinio tomas. Jei ten padėtas paskutinis kurio nors leidinio tomas, į lentelę įrašomas skaičius -1.

Mūsų pateiktam pavyzdžiui antroji lentelė atrodo šitaip:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	5	6	2	-1	7	9	-1	-1	8	-1	0	0	0

Tokiu būdu bibliotekininkas galėjo lengviau rasti skaitytojo pageidaujamą tomą. Pavyzdžiui, skaitytojas pageidauja gauti antrojo leidinio trečiąjį tomą. Iš pirmos lentelės bibliotekininkas mato, kad pirmasis tomas lentynoje yra trečias. Tada iš antros lentelės trečio stulpelio aišku, kad antrasis tomas bus šeštoje lentynos vietoje. O šeštasis stulpelis parodo, kad trečiasis tomas yra septintoje lentynos vietoje.

Kartą prisireikė perrašyti antrąją lentelę. Bibliotekininkas netyčia galėjo padaryti vieną klaidą – įrašyti ne tokį skaičių antroje eilutėje. Bibliotekininkas tikrai žino, kad perrašydamas nulius jokios klaidos nepadarė: kur senoje lentelėje buvo įrašyti nuliai, ten ir perrašytoje lentelėje tikrai yra nuliai.

Parašykite algoritmą, kuris nustatytų, ar bibliotekininkas padarė klaidą, ir jei taip – ištaisytų ją, jeigu tai įmanoma.

Pradiniai duomenys skaitomi iš bylos KNYGOS.DAT. Pirmoje eilutėje įrašytas leidinių skaičius n ($1 \leq n \leq 100$). Vieną leidinį sudaro ne daugiau kaip 20 tomų.

Tolesnėse n eilučių pateikta pirmoji lentelė. Į kiekvieną eilutę įrašyta po du skaičius: leidinio pirmojo tomo vieta ir leidinio tomų skaičius.

Kitoje eilutėje įrašytas vietų skaičius lentynoje k ($1 \leq k \leq 2000$). Toliau eina k eilučių, kuriose pateikta antroji lentelė. Kiekvienoje eilutėje įrašyta po vieną skaičių.

Rezultatai spausdinami ekrane. Jeigu klaidos nebuvo, tai spausdinamas pranešimas GERALI. Jei klaida vis dėlto buvo padaryta ir jos ištaisyti nepavyksta, spausdinamas pranešimas BLOGAI.

Jei klaidą ištaisyti galima, tai spausdinami du skaičiai (vienoje eilutėje, atskirti tarpu). Pirmasis skaičius – tai perrašytos lentelės stulpelis, kur buvo klaida, antrasis – atstatyta reikšmė.

Pavyzdžiai

1 pavyzdys

KNYGOS.DAT	Rezultatai
4	GERALI
1 4	
3 4	
10 2	
11 1	
14	
4	
5	
6	
2	
-1	
7	
9	
-1	
-1	
8	
-1	
0	
0	
0	

2 pavyzdys

KNYGOS.DAT	Rezultatai
4	2 5
1 4	
3 4	
10 2	
11 1	
14	
4	
1	
6	
2	
-1	
7	
9	
-1	
-1	
8	
-1	
0	
0	
0	

2-ojo pavyzdžio komentaras. Buvo padaryta klaida antrame lentelės stulpelyje: penketas pakeistas į vienetą. Todėl susidarė ciklas: pirmas tomas stovi pirmoje vietoje, antras – ketvirtoje, trečias – antroje, o ketvirtas – vėl pirmoje.

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 85 taškų; už sprendimo idėjos aprašymą – iki 15 taškų.

3.2.J3 uždavinys. Nežinomas kodavimas (vykdymo laikas – ne daugiau kaip 30 sek.). Diskelyje pateikta programa KOD.EXE. Ji dvejetainėje byloje A esančius duomenis užkoduoja ir surašo į bylą B.

Kaip koduoja programa KOD.EXE, galite sužinoti eksperimentuodami – pateikdami jai savus pradinis duomenis ir analizuodami iš jos gautus rezultatus.

Yra žinomi tik kodavimo principai.

1. Į pradinių duomenų bylą A žiūrima kaip į ištisinę bitų eilę. Bitai skirstomi į vienodo ilgio grupes po 7, 6, 5, 4, 3 arba 2 bitus. Jei paskutinė grupė nepilna, tai jos pabaiga užpildoma dvejetainiais nuliais.
2. Kiekviena grupė koduojama atskiru baitu.

Parašykite programą, kuri iškoduotų duomenis, t. y. iš bylos B gautų bylą C, identišką bylai A.

Diskelyje įrašyta kodavimo programa KOD.EXE turi du parametrus. Programa pasinaudoti galima perduodant koduojamos bylos ir bylos, į kurią turi būti rašomi užkoduoti duomenys, vardus. Pavyzdžiui:

`kod.exe duom.dat duom.kod`

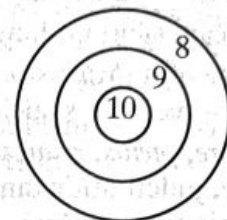
Testuojama bus naudojantis bylomis, kurių dydis (užkoduotų) ne daugiau kaip 10000 baitų.

Pradiniai duomenys ir rezultatai. Iškodavimo programa pradinis duomenis (užkoduotą tekstą) turi skaityti iš bylos KOD.DAT. Iškoduo- tą tekstą ji turi rašyti į bylą KOD.REZ.

Reikia pateikti programą ir aprašymą, iš kurio būtų aišku, kaip analizavote programą KOD.EXE ir kad Jūsų sudaryta programa tikrai atlieka tą darbą, kurio iš jos reikalaujama.

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 70 taškų; už sprendimo aprašymą – iki 30 taškų.

3.2.J4 uždavinys. Šaudymas (vykdymo laikas – ne daugiau kaip 5 sek.). Visiems gerai žinomas šaudymo taikiny: apskritimais ribojamas plotas.



Taikinio centras sutampa su koordinatų pradžia. Duotas pirmojo (mažiausio) apskritimo spindulys r . Į jį patekę šūviai vertinami 10 balų. Šūvis, patekęs ant apskritimo ribos, vertinamas taip, tarsi būtų pataikyta į apskritimo ribojamą plotą. Kiekvieno tolesnio apskritimo spindulys du kartus didesnis už prieš einančio, o šūviai vertinami vienu balu mažiau. Jei pataikyta už paskutiniojo (dešimtojo) apskritimo ribos, šūvis vertinamas 0 balų.

Parašykite algoritmą balų, surinktų už šūvius, sumai rasti.

Pradiniai duomenys įrašyti tekstinėje byloje SAUD.TXT. Pirmoje eilutėje nurodytas mažiausio apskritimo spindulys r bei šūvių skaičius. Antroje – pirmojo šūvio koordinatės, trečioje – antrojo ir t. t.

Visi pradiniai duomenys yra sveikieji skaičiai.

Rezultatą spausdinkite ekrane.

Pavyzdys

SAUD . TXT	Rezultatas
1 2	19
1 1	
0 0	

Sprendimas vertinamas 100 taškų – visi už programos teisingumą.

3.2.J5 uždavinys. Labirintas (vykdymo laikas – ne daugiau kaip 5 sek.). Duotas kvadratinis labirintas, sudarytas iš laukelių. Jūs stovite prie labirinto įėjimo ir žiūrite labirinto kryptimi. Įėjimą iš abiejų pusių juosia sienos, jis negali būti labirinto kampe.

Parašykite algoritmą, kuris padėtų Jums išeiti iš labirinto. Žinoma, kad išeiti visada galima. Ejimų skaičius labirinte turėtų būti kuo mažesnis, tačiau nebūtina iš labirinto išeiti pačiu trumpiausiu keliu.

Pradinių duomenų nėra. Vaikščioti labirintu Jums padės modulis *labir*.

Kreipdamiesi į modulio procedūrą *dydis*, sužinosite labirinto dydį n ($3 \leq n \leq 64000$). Labirinto dydis – sveikasis skaičius.

Kreipdamiesi į funkcijas *siaure*, *pietus*, *rytai*, *vakarai* sužinosite, ar galite iš laukelio, kuriame stovite, judėti atitinkama kryptimi. Visų šių procedūrų rezultatas – viena loginio tipo reikšmė.

Kreipdamiesi į procedūras *pirmyn*, *atgal*, *kairen*, *desinen*, paeisite labirintu per vieną laukelį nurodyta kryptimi. Šios procedūros neturi parametrų – rezultatų.

Nė viena iš anksčiau išvardytų funkcijų bei procedūrų neturi pradinių duomenų.

Jūsų programa neturi spausdinti jokių *rezultatų*. Pasiekus išėjimą iš labirinto, modulis sustabdys Jūsų programą. Modulis taip pat sustabdys Jūsų programą, jei bandysite atlikti klaidingą ėjimą, t. y. eiti į laukelį, kuriame yra labirinto sienos, arba jei Jūsų programa viršys tam tikrą ribinį ėjimų skaičių (taip gali atsitikti, jei, pavyzdžiui, imsite vaikščioti ratu).

Nebūtina rasti trumpiausią kelią iš labirinto.

Testavimui bus naudojama kita modulio *labir* versija.

Sprendimas vertinamas 100 taškų – visi už programos teisingumą.

3.2.J6 uždavinys. Nameliai (vykdymo laikas – ne daugiau kaip 5 sek.). Duota n pagaliukų, kurių ilgiai išreikšti sveikaisiais skaičiais. Iš pagaliukų reikėtų sudėti namelius: jo siena – kvadratas, stogas – bet koks trikampis, kurio viena kraštinė sutampa su sienos briauna.



Pagaliukų sudurti negalima. Kiekvieną pagaliuką galima panaudoti tik vieną kartą.

Parašykite programą, kuri suskaičiuotų, kiek daugiausia namelių galima sudaryti iš duotų pagaliukų.

Pradiniai duomenys įrašyti tekstinėje byloje NAMELIS.DAT. Pirmoje eilutėje įrašytas skaičius n ($0 \leq n \leq 1000$), tolesnėse n eilučių – atkarpų ilgiai, po vieną atkarpą eilutėje.

Rezultatas – namelių skaičius – pateikiamas pirmoje bylos NAMELIS.REZ eilutėje.

Pavyzdys

NAMELIS.DAT	NAMELIS.REZ
7	1
3	
4	
4	
2	
4	
8	
4	

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų; už algoritmavimo kultūrą – iki 10 taškų.

Vyresniųjų grupė

3.2.V1 uždavinys. Siųstuvas ir imtuvas (vykdymo laikas – ne daugiau kaip 90 sek.). Į vieną imtuvą signalus siunčia n siųstuvų ($1 \leq n \leq 20$). Siųstuvas k skleidžia signalus kas T_k sekundžių, o šie imtuvą pasiekia po L_k sekundžių. Kiekvienas siųstuvas turi pasiųsti pranešimą iš P_k signalų. Taigi kiekvieno siųstuvo darbą apibūdina trys skaičiai. Pavyzdžiui, $T_k = 3$, $L_k = 2$, $P_k = 2$. Jei siųstuvas įjungiamas laiko momentu 0, jis išsiųs du signalus laiko momentais 0 ir 3, o imtuvas gaus signalus laiko momentais 2 ir 5.

Gali atsitikti taip, kad tuo pačiu metu į imtuvą ateina keli signalai iš karto. Gautasis signalas (kelių signalų mišinys) vadinamas *blogu* signalu.

Siųstuvas gali būti įjungti arba išjungti. Kai siųstuvas pradeda veikti, t. y. įjungiamas, jis tuoj pat išsiunčia savo pranešimą ir tada išjungia-

mas. Siųstuvai turi pradėti darbą iš eilės: pirmasis, antrasis ir t. t. Pavyzdžiui, trečiasis siųstuvus negali būti įjungtas anksčiau nei antrasis, bet gali dirbti kartu su anksčiau įjungtais siųstuvais. Tuo pačiu momentu pradėti siųsti pranešimus keli siųstuvai negali.

Momentas, kada paleidžiamas dirbti pirmasis siųstuvus, vadinamas *nuliniu* laiko momentu (nuo tada imamas skaičiuoti laikas). Darbo pabaiga laikomas momentas, kai imtuvas gauna paskutinį signalą.

Parašykite algoritmą, kuris nustatytų, kuriuo laiko momentu reikia įjungti kiekvieną siųstuvą laikantis nustatytos tvarkos, kad imtuvas negautų blogų signalų, o visi pranešimai būtų perduoti per trumpiausią laiką.

Pastaba. Laikas, per kurį siųstuvus perduoda pranešimą imtuvui, ne didesnis nei du šimtai sekundžių. Pradedama skaičiuoti nuo siųstuvo įjungimo momento ir baigiama, kai imtuvas gauna paskutinį pranešimo signalą.

Pradiniai duomenys skaitomi iš bylos RYSYS.DAT. Pirmoje eilutėje įrašytas siųstuvų skaičius n . Tolesnėse n eilučių yra po tris skaičius, aprašančius kiekvieno siųstuvo darbą. Pirmas skaičius eilutėje yra T_k , antrasis – L_k , o trečiasis – P_k .

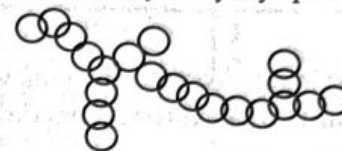
Rezultatą reikia įrašyti į bylą RYSYS.REZ. Joje turi būti n eilučių, kiekvienoje jų – po vieną skaičių. Šis skaičius rodo, kuriuo laiko momentu (kurią sekundę) reikia įjungti kiekvieną siųstuvą. Pirmą rezultatų bylos eilutę atitiks pirmąjį siųstuvą (joje visada įrašytas nulis), antra – antrąjį siųstuvą, ir t. t. Jei galimi keli atsakymo variantai, užtenka pateikti bet kurį vieną.

Pavyzdys

RYSYS.DAT	RYSYS.REZ
5	0
3 2 4	9
1 3 3	11
2 4 3	13
3 5 4	23
2 3 4	

Sprendimas vertinamas 100 taškų – visi už programos teisingumą.

3.2.V2 uždavinys. *Sukabinti žiedai* (vykdymo laikas – ne daugiau kaip 5 sek.). Sukabinta n žiedų. Pavyzdys pateiktas paveiksle.



Parašykite algoritmą, kuris rastų, kiek daugiausia žiedų galima pašalinti, kad tarp žiedų a ir b liktų vientisa, be išsišakojimų grandinė.

Pradiniai duomenys įrašyti tekstinėje byloje ZIEDAI.DAT. Pirmoje eilutėje įrašytas žiedų skaičius n ($1 \leq n \leq 300$), žiedų a ir b numeriai bei skaičius k ($1 \leq k \leq 1000$). Tolesnėse k eilučių įrašyta po du skaičius: sujungtų žiedų porų numeriai.

Rezultatą pateikite ekrane.

Pavyzdys

ZIEDAI.DAT	Rezultatas
10 1 10 9	2
1 2	
2 3	
3 4	
4 5	
4 7	
5 6	
7 8	
8 9	
9 10	

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų, už algoritmovimo kultūrą – iki 10 taškų.

3.2.V3 uždavinys. *Erdvėlaivis* (vykdymo laikas – ne daugiau kaip 20 sek.). Erdvėlaivis turi nugabenti svarbų krovinį iš vienos bazės į kitą, tačiau skrydžiui ne visada užtenka kuro, kurį jis gali pasiimti kelionės pradžioje. Todėl jis skrenda pro tarpines bazes, jose papildydamas kuro atsargas. Įvairiose bazėse kuro atsargų papildymas kainuoja nevienodai.

dai, todėl turėdamas ribotą kreditą, skirtą kurui, erdvėlaivio vadas ne visada pasirenka trumpiausią kelią.

Galioja šios sąlygos:

- kuro papildymas bazėje kainuoja fiksuotą piniginių vienetų skaičių (nesvarbu, kiek pilama kuro), todėl kuro rezervuaras užpildomas visas;
- erdvėlaivis skrieja vienodu greičiu;
- laikoma, kad bazėse erdvėlaivis užtrunka artimą nuliui laiko tarpą, todėl kelionės trukmė proporcinga nuskristam keliui.

Parašykite programą, kuri rastų tokį erdvėlaivio skridimo maršrutą, kad krovinyt būtų pristatytas kiek galima greičiau.

Pradiniai duomenys. Bylos ERDV.DAT pirmoje eilutėje įrašyti keturi sveikieji skaičiai:

- erdvėlaivio kuro rezervuaro talpa, nurodyta kuro tūrio vienetais ($1 \leq V \leq 100$),
- kuro sąnaudos vienam ilgio vienetui ($1 \leq S \leq 100$),
- turimas kreditas, išreikštas piniginių vienetais ($1 \leq K \leq 100$),
- bazių, kuriose erdvėlaivis gali papildyti kuro atsargas, skaičius ($2 \leq N \leq 100$).

Likusiose N eilučių įrašyta po keturis sveikuosius skaičius k, x, y, z , apibūdinančius atitinkamą kuro bazę:

- k – kuro rezervuaro užpildymo kaina bazėje, nurodyta piniginių vienetais ($0 \leq k \leq 100$),
- (x, y, z) – bazės padėtis erdvėje, visos koordinatės nurodytos ilgio vienetais ($0 \leq x, y, z \leq 100$).

Erdvėlaivis turi nugabenti krovinį iš pirmos bazės į N-ąją. Pradžioje erdvėlaivio bakas yra tuščias.

Rezultatai rašomi į bylą ERDV.REZ. Joje reikia įrašyti bazių, kuriose erdvėlaivis papildoma kuro atsargas skrisdamas optimaliu maršrutu, numerius. Numeriai turi būti rašomi ta tvarka, kuria bazės aplankomos: po vieną numerį į eilutę. Į paskutinįją eilutę įrašykite N-osios bazės numerį.

Pavyzdys

ERDV.DAT	ERDV.REZ
10 1 10 4	1
5 0 0 0	3
15 10 0 0	4
5 5 5 0	
1 10 2 0	

Sprendimas vertinamas 100 taškų – visi už programos teisingumą.

3.2.V4 uždavinys. Pokeris (vykdymo laikas – ne daugiau kaip 5 sek.). Šis kortų žaidimas dažnai yra lošimų automatuose. Mes nagrinėsime paprastesnį pokerio variantą. Automatas iš 24 kortų (nuo devynakės iki tūzo) kaladės atsitiktinai išrenka lošėjui penkias kortas. Lošejas nusprendžia, kurias kortas jis norėtų pasilikti, o kurias – pakeisti. Lošejo keičiamas kortas automatas vėl įdeda į kaladę ir vėl iš jos atsitiktinai išrenka tiek kortų, kiek jų buvo įdėta.

Nuo susidariusios kortų kombinacijos priklauso išlošimas. Išlošiama šiais atvejais.

1. *Pora* – dvi vienodo rango kortos (pavyzdžiui, dvi damos).
2. *Dvi poros* (pavyzdžiui, dvi dešimtakės ir du tūzai).
3. *Trejetas* – trys vienodo rango kortos.
4. *Iš eilės* – visos kortos galima išdėstyti iš eilės pagal rangą (pavyzdžiui, devynakė, dešimtakė, berniukas, dama ir karalius).
5. *Vienos spalvos* – visos kortos vienos rūšies (pavyzdžiui, visi kryžiai).
6. *Pilni namai* – trejetas ir pora.
7. *Ketvertas* – keturios vienodo rango kortos.
8. *Pokeris* – visos kortos išdėstomos iš eilės ir yra vienos rūšies.

Daugeliu atveju galima sudaryti kelias kombinacijas. Pavyzdžiui, jei turime *Ketvertą*, galime sudaryti ir *Porą*, ir *Trejetą*. Tokiu atveju laikykite, kad susidaro tik viena išlošiama situacija – ta, kurios numeris (prioritetas) didžiausias.

Vienas padėtis yra lengviau sudaryti, kitas – sunkiau. Todėl sudarius vienas padėtis išlošiama mažiau pinigų, kitas – daugiau. Yra suda-

ryta išlošimų lentelė, kurioje nurodyta, kiek pinigų išlošiama susidarius kiekvienai iš aštuonių padėčių.

Padėties pasitaikymo tikimybė yra galimų skirtingų kortų kombinacijų, kuriose yra norima padėtis, skaičiaus santykis su visų galimų kombinacijų skaičiumi.

Pavyzdys. Tarkime, Jūs pasilikote dvi damas ir du karalius ir norite sužinoti, kokia tikimybė susidaryti *Pilniems namams* atsitiktinai ištraukus penktą kortą. Kadangi iš 24 kortų kaladės išimtos dvi damos ir du karaliai, tai penktą kortą galime ištraukti $24 - 2 - 2 = 20$ būdų, taigi iš viso yra 20 galimų skirtingų kombinacijų. Tačiau *Pilni namai* susidarys, jei ištraukta korta bus arba dama, arba karalius. Kadangi kaladėje yra dvi damos ir du karaliai, tai bus $2 + 2 = 4$ galimos skirtingos kombinacijos, kurios gali sudaryti *Pilnus namus*. Taigi tikimybė, kad susidarys *Pilni namai*, bus $4 / 20 = 0,2$.

Išlošimo vidurkis apibrėžiamas kaip išlošiančių padėčių pasitaikymo tikimybų ir atitinkamų išlošimų sandaugų suma.

Pavyzdys. Tarkime, Jūs pasilikote dvi damas ir du karalius ir susidarius *Dviems poroms* išlošiate 5 Lt, o susidarius *Pilniems namams* išlošiate 30 Lt, kitos išlošiančios padėtys šiuo atveju susidaryti negali. Jūs jau paskaičiavote, kad *Pilnų namų* susidarymo tikimybė šiuo atveju lygi 0,2. Kadangi likusiose 16 iš 20 galimų kombinacijų yra *Dvi poros*, tai jų susidarymo tikimybė bus lygi $16 / 20 = 0,8$. Taigi tuo atveju, kai pasiliksite anksčiau minėtas keturias kortas, Jūsų išlošimo vidurkis bus lygus $0,8 \times 5 \text{ Lt} + 0,2 \times 30 \text{ Lt} = 10 \text{ Lt}$ (vidutiniškai galite išlošti 10 Lt).

Jūsų programa pagal duotas penkias kortas ir išlošimų lentelę turėtų nuspręsti, kurias kortas reikia pasilikti, o kurias pakeisti, kad išlošimo vidurkis būtų didžiausias.

Pradiniai duomenys. Bylos POKERIS.DAT pirmose penkiose eilutėse yra automato išrinktų kortų kodai. Kortos koduojamos viena didžiąja raide ir vienu skaičiumi. Raidė nurodo kortos rūšį (S – širdys, B – būgnai, K – kryžiai, V – vynai). Skaičius nurodo kortos rangą: 1 – devynakė, 2 – dešimtakė, 3 – berniukas, 4 – dama, 5 – karalius, 6 – tūzas. Eilutėse nėra tarpų. Visos kortos skirtingos.

Kitose aštuoniosiose eilutėse yra po vieną sveikąjį skaičių, kuris yra išlošimo suma, gaunama susidarius atitinkamai kortų padėčiai.

Pavyzdys

POKERIS.DAT	POKERIS.REZ
K1	TAIP
V1	TAIP
V3	NE
B2	NE
S6	NE
1	
5	
10	
20	
200	
100	
500	
1000	

Rezultatų byloje POKERIS.REZ turėtų būti penkios eilutės, kiekvienoje kurių yra arba žodis TAIP, jei atitinkamą kortą norima pasilikti, arba žodis NE, jei tą kortą norima pakeisti.

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 90 taškų, už algoritmavimo kultūrą – iki 10 taškų.

3.2.V5 uždavinys. *Didelė valstybė* (vykdymo laikas – ne daugiau kaip 20 sek.). *Terra incognita* (lot.) valstybę valdo diktatorius. Kaip dažniausiai būna tokiais atvejais, bet kokia tiksli informacija apie valstybę yra slepiama. Diktatorius teigia, kad jo valstybė yra labai didelė ir pateikia rimtų argumentų: *Terra incognita* valstybę galima sutalpinti į $n \times n$ ($1 \leq n \leq 2000$) dydžio kvadratinį žemės plotą ir valstybės kontūrai lies visus jo kraštus! Tačiau tikslaus ploto diktatorius pasakyti nenori. Žinoma, kad valstybės kontūras yra iškiliojo daugiakampio formos.

Parašykite algoritmą, kuris apskaičiuotų valstybės plotą.

Pradinių duomenų nėra. Duomenis apie valstybę programa gali gauti pasinaudodama modulio *terrainc* funkcija *valstybe*. Ši funkcija turi du pradinius duomenis – taško koordinatės (realiuosius skaičius) ir vieną rezultatą, nurodantį, ar duotas taškas priklauso valstybės teritorijai, ar ne. Jei taškas priklauso valstybės sienai, laikoma, kad jis priklauso valstybei.

Norint sužinoti kvadratinio žemės ploto koordinatės reikia kreiptis į procedūrą *dydis*. Jos rezultatai – trys sveikieji skaičiai: apatinio kairiojo kvadrato kampo koordinatės bei kvadrato dydis n .

Rezultatą (penkių skaitmenų tikslumu) spausdinkite ekrane.

Sprendimas vertinamas 100 taškų – visi už programos teisingumą.

3.2.V6 uždavinys. *Nežinomas kodavimas* (vykdymo laikas – ne daugiau kaip 30 sek.). Diskelyje pateikta programa KOD.EXE. Ji dvejetainės bylos A duomenis užkoduoja ir surašo į bylą B.

Kaip koduoja programa KOD.EXE, galite sužinoti eksperimentuodami – pateikdami jai savus pradinius duomenis ir analizuodami iš jos gautus rezultatus.

Yra žinomi tik kodavimo principai.

1. Į pradinių duomenų bylą A žiūrima kaip į ištisinę bitų eilę. Bitai skirstomi į vienodo ilgio grupes po 7, 6, 5, 4, 3 arba 2 bitus. Jei paskutinė grupė nepilna, tai jos pabaiga užpildoma dvejetainiais nuliais.
2. Kiekviena grupė koduojama atskiru baitu.

Parašykite programą, kuri iškoduotų duomenis, t. y. iš bylos B gautų bylą C, identišką bylai A.

Diskelyje įrašyta kodavimo programa KOD.EXE turi du parametrus. Programa pasinaudoti galima perduodant koduojamos bylos ir bylos, į kurią turi būti rašomi užkoduoti duomenys, vardus. Pavyzdžiui:

`kod.exe duom.dat duom.kod`

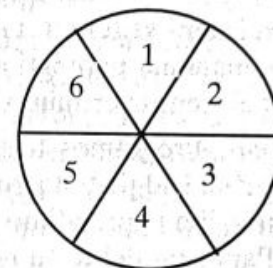
Testuojama bus naudojantis bylomis, kurių dydis (užkoduotų) ne daugiau kaip 10000 baitų.

Pradiniai duomenys ir rezultatai. Iškodavimo programa pradinius duomenis (užkoduotą tekstą) turi skaityti iš bylos KOD.DAT. Iškoduo- tą tekstą ji turi rašyti į bylą KOD.REZ.

Reikia pateikti programą ir aprašymą, iš kurio būtų aišku, kaip analizavote programą KOD.EXE ir kad Jūsų sudaryta programa tikrai atlieka tą darbą, kurio iš jos reikalaujama.

Sprendimas vertinamas 100 taškų: už programos teisingumą – iki 70 taškų; už sprendimo aprašymą – iki 30 taškų.

3.2.V7 uždavinys. *Lošimas disku* (vykdymo laikas – ne daugiau kaip 5 sek.). Lošiama dviese. Abu lošėjai sukioja į 6 segmentus suskirstytą žemiau nupieštą diską. Segmentai sužymėti skaičiais nuo 1 iki 6.



Lošėjai paeiliui sukioja diską per vieną segmentą į kairę arba į dešinę. Lošėjas, pasukęs diską, perskaito jo viršutiniame segmente atsiradusį skaičių n ir jį prideda prie sumos s .

Lošimo pradžioje $s = 0$. Lošimas baigiamas, kai s pasiekia sutartą skaičių m (pavyzdžiui, 1000). Jeigu $s = m$, tai laimi tas, kuris pasiekė šį skaičių. Jeigu $s > m$, tai pralaimi tas, kas buvo priverstas viršyti m (t. y. laimi tas, kuris priešininkui sudarė tokią situaciją, kai jis nebegalėjo pasirinkti tokio ėjimo, kad suma neviršytų sutarto skaičiaus).

Sudarykite algoritmą, kuris rastų geriausią ėjimą esant duotai situacijai. Algoritmą užrašykite funkcija, turinčia tokią antraštę:

`function diskas (n, s, m: integer): integer;`

Funkcijos reikšmė yra skaičius, kuris turi atsirasti viršutiniame disko segmente atlikus ėjimą.

Pastaba. Reikia ėjimą tik rasti, o ne atlikti. Ėjimą atliks lošimą valdanti programa, kurios sudaryti nereikia.

Reikia pateikti: 1) funkciją, sprendžiančią duotąjį uždavinį; 2) taisykles (instrukciją) lošėjui, kaip parinkti geriausią ėimą; 3) įrodymą, įtikinantį, kad ėjimas, parinktas pagal instrukciją, yra geriausias.

Sprendimas vertinamas 100 taškų: už algoritmo teisingumą – iki 50 taškų; už taisykles lošėjui – iki 20 taškų; už įrodymą – iki 30 taškų.

Sprendimų vertinimas

Olimpiados darbų vertinimas visuomet būna sunkiausias dalykas. Dėl to dažnai ginčijamasi, keičiami kriterijai, tačiau visiško objektyvumo neįmanoma pasiekti. Neįmanoma parengti absoliučiai teisingų kriterijų. Smulkiau aptarsime kiekvienos vertinimo dalies principus.

Programavimo teisingumas. Programos teisingumas vertinamas iš anksto parengtais testais. Svarbu, kad jie būtų kuo korektiškiau parengti, kad apręptų kuo daugiau galimų sprendimo atvejų. Čia daugiausia darbo vertinimo komisijai. Paprastai dėl testų nekyla ginčų: pasitikima kompiuteriu, kuris arba rodo sprendimą, arba ne.

Tačiau yra kitų su programų teisingumu susijusių dalykų. Pirmiausia, pasitaiko menkų sintaksės klaidų (dažniausiai jų – pavyzdžiui, komentarų skliausto nebuvimas – atsiranda, kai komentarai rašomi pasutinę minutę ir nebeįsitikinama programos teisingumu). Taisyti jas ar ne? Jei netaisome, lyg ir jaučiame kartėlį: tas mokinys, kuris gal gerai galvoja, dėl žioplos klaidos nepakliūva į tolesnį etapą... Jei taisome, iškyla problema, kiek taisyti: ar tik komentaro skliaustą, ar vieno kintamojo aprašą, ar masyvo režius, ar dar ką nors... Vertindama olimpiados trečiojo etapo pirmosios dalies sprendimus komisija dažniausiai dar toleruoja menkas sintaksės klaidas, tačiau už tai skiriama baudos taškų – apie 50 % programos teisingumui numatytų taškų.

Yra netikslumų pateikiant rezultatus. Testuojama automatiškai. Jei mokinio programa nepateikia sprendinio taip, kaip reikalaujama sąlygoje (pavyzdžiui, pridedama papildomų tarpų, duomenys neišrikiuojami prašoma tvarka), jis paprasčiausiai ignoruojamas. Vertindama trečiojo etapo pirmosios dalies sprendimus komisija tokiais atvejais „ran-

kiniu“ būdu peržiūri šias programas. Jei sprendimai teisingi, paprastai jų autorius gauna dalį taškų.

Kalbant apie rezultatų pateikimą yra dar viena opi problema: tai nereikalingi veiksmai ekrane – pavyzdžiui, ekranas nuspalvinamas kokia nors spalva arba išvalomas. Kartais mėginama palaikyti dialogą su vartotoju – prašoma paspausti kokį nors klavišą ir panašiai. Olimpiadoje tai nereikalingi, taisymą sunkinantys veiksmai, nes nepaisoma uždavinio sąlygos ir už tai mokinys netenka taškų.

Algoritmavimo kultūra. Ji paprastai vertinama tik tada, kai programa yra iš dalies teisinga, t. y., kai veikia bent su kai kuriais testais. Paprastai žiūrima į du svarbiausius dalykus: pirma, ar programa racionali (racionalumą iš dalies matuoja ir testai: paprastai nurodomas programos vykdymo laikas, bet čia dar žiūrima į tinkamą duomenų struktūrų bei valdymo konstrukcijų vartojimą, perėjimo, ciklo nutraukimo sakinį itin apgalvotą vartojimą ir pan.) ir, antra, ar programa tinkamai išdėstyta ir pakomentuota.

Sprendimo idėjos aprašymas. Prašoma aprašyti sprendimo idėją tik tų uždavinių, kurie paprastai turi kažką gražaus, neįprasto, dėl ko reikia gerokai pasukti galvą. Už idėjos suradimą paprastai galima gauti apie pusę idėjos aprašui skiriamų taškų. Kita pusė lieka už idėjos pateikimą. Tačiau bėda ta, kad mūsų tiksluosius mokslus išmanantys moksleiviai nepratinami aiškiai išreikšti minčių sakiniiais. Ko reikalaujama aprašant idėją, buvo nusakyta olimpiados trečiojo etapo pirmosios dalies darbų vertinimo rašte, įteiktame kartu su kitais olimpiados dokumentais kiekvienam švietimo skyriui. Gal kai kurie švietimo skyriai nepasirūpino, kad šie nurodymai pasiektų kiekvieną olimpiados dalyvį?

Sakoma, kad iš sprendimo idėjos aprašo turėtų būti vienareikšmiškai aišku, kaip sudaryti algoritmą. Sprendimo idėja turėtų būti aiški be pačios programos, todėl apraše neturi būti vartojama jokių programos žymenų: kintamųjų, procedūrų, funkcijų vardų ir panašiai. Aprašą reikia rašyti sklandžiai, taisyklinga lietuvių (rusų ar lenkų – priklausomai nuo to, kokioje mokykloje moksleivis mokosi) kalba, be žargono, vertalų ir panašiai, mintis dėstyti aiškiai ir glaustai.

Vertinant trečiojo etapo pirmosios dalies sprendimus kilo diskusija, ką daryti su tais darbais, kurie sprendimo idėjų aprašus rašo nevariodami lietuviškų žymenų. Pasirodo, buvo net tokių mokytojų, kurie rekomendavo savo mokiniams nevarioti lietuviškų raidžių... Nacionalinė vertinimo komisija jau daugelį metų reikalauja rašyti taisyklinga lietuvių kalba. Tačiau daliai mokinių (o gal mokytojų?) tai – nė motais. Todėl šį kartą buvo nutarta elgtis principingai: nelietuviškais rašmenimis parašytų darbų neskaityti ir juos vertinti nulių taškų.

Vertinant olimpiados trečiojo etapo antrosios dalies sprendimus jokių išimčių nebuvo daroma ir visi darbai buvo vertinami griežtai pagal uždavinio sąlygos reikalavimus – lygiai kaip ir pasaulinėje informatikos olimpiadoje.

Olimpiados rezultatai

Susumavus trečiojo etapo antrosios dalies visų uždavinių sprendimo rezultatus, paaiškėjo Lietuvos devintosios moksleivių informatikos olimpiados nugalėtojai. Vyresniųjų grupėje pirmąją vietą užėmė Kauno technologijos universiteto gimnazijos dešimtokas Gediminas Lukšys, surinkęs 415 taškų iš 700 galimų. Prieš metus (1996–1997 mokslo metais) jis pirmą kartą dalyvavo informatikos olimpiadoje ir jaunesniųjų grupėje buvo pirmasis, nepelnęs prizinės vietos. Jaunesniųjų grupėje pirmąją vietą, kaip ir prieš metus, užėmė Jurgis Pašukonis, Vilniaus tikslųjų, gamtos ir technikos mokslų licėjaus devintokas, surinkęs 295 taškus iš 600 galimų.

Visi olimpiados laimėtojai išvardyti toliau pateiktose lentelėse.

Jaunesniųjų grupė

Laimėtojas	Mokykla	Taškai	Vieta
Jurgis Pašukonis	Vilniaus TGTM licėjus	295	I
Vidmantas Maskoliūnas	Marijampolės 6-oji vid. m.	255	II
Antanas Kompanas	Vilniaus M. Biržiškos gimn.	249	III

Vyresniųjų grupė

Laimėtojas	Mokykla	Taškai	Vieta
Gediminas Lukšys	KTU gimnazija	415	I
Marius Gedminas	Vilniaus 64-oji vid. m.	408	II
Justas Kranauskas	Vilniaus TGTM licėjus	345	II
Vidas Pažusis	Širvintų „Atžalyno“ vid. m.	320	III
Remigijus Staniulis	KTU gimnazija	315	III
Vaidas Balys	Panevėžio J. Balčikonio gimn.	298	III

Pirmieji aštuoni vyresniųjų grupės olimpiadininkai pelnė apdovanojimą – 1998 m. balandžio 25–28 d. varžėsi Ketvirtojoje Baltijos šalių informatikos olimpiadoje, kuri vyko Estijoje, Tartu ir Talino miestuose. Be Estijos, Latvijos ir Lietuvos komandų, šį kartą Baltijos šalių olimpiadoje dalyvavo ir Suomijos bei Švedijos moksleiviai. (Šios olimpiados uždavinių sąlygos pateiktos kompiuterinio leidinio „Veidrodis“ 1998 m. ketvirtame numeryje.)

Jaunesniųjų grupės geriausiųjų ketvertukas 1998 m. gegužės 11–16 d. dalyvavo Mogiliove (Baltarusija) vykusiose Septintosiose atvirose moksleivių informatikos varžybose (individualių varžybų uždavinių sąlygos pateiktos kompiuterinio leidinio „Veidrodis“ 1998 m. penktame numeryje).

BŪDAS MOKYTI KITAIP

Valentina Dagienė



Auksės Balčytienės knyga „Būdas mokyti kitaip: hipertekstinė mokymo aplinka“ (Vilnius: Margi raštai, 1998) turėtų patraukti visų švietimo specialistų dėmesį jau vien pavadinimo aktualumu.

Išskirčiau tris svarbiausias knygos temas: apie šiuolaikines mokymo teorijas, apie hiperteksto naudojimą mokymui ir apie mokomųjų programų kūrimą panaudojant programą „Toolbook 3.0“. Mane ypač domino pirmosios dvi, todėl du trečdalius knygos perskaičiau labai atidžiai, ne tik perskaičiau, bet ir kai kurias temas išdėčiau Vilniaus pedagoginio universiteto magistrantams.

Autorė puikiai išmano įvairias mokymo teorijas, sugeba jas glaustai ir aiškiai pateikti, pagrindinį dėmesį skiria šiuo metu vyraujančiam bihevioristiniam ir kognityviajam požiūriui į mokymą. Panagrinėjusi B. Skinnerio pažinimo teoriją, autorė ima gvildinti klausimus, susijusius su mokomųjų programų kūrimu. Niekas neabejoja, kad šiandien tai itin aktualia tema, tačiau Lietuvoje apie tai rašoma labai mažai.

Panagrinėjusi mokomųjų programų kūrimo principus, autorė suformuluoja teiginį, jog hipertekstas – tai moderni šių laikų informaci-

jos technologija ir puiki priemonė mokomajai medžiagai pateikti. Hipertekstas – nauja informacijos šaltinio vidinės struktūros forma. Hiperteksto skaitymas labiau atitinka žmogaus psichologiją. Čia pasiūlyama laisvomis žmogaus asociacijomis ir interpretacijomis bei tolesnių teksto fragmentų laisvu pasirinkimu.

Džiugu, kad A. Balčytienė, aptardama kiekvieną klausimą, jį susieja su įvairiais mokymo teorijos požiūriais, argumentuoja pasirinktą kryptį, giliai ir mokliškai atskleidžia esmę. Antra vertus, knyga nėra vien mokslinių teorijų rinkinys, ji parašyta gana lengvai ir suprantamai.

Autorė pateikia hipertekstinio dokumento išsamią charakteristiką, aprašo darbo principus. Ypatinę dėmesį ji skiria skaitymo proceso analizei – nuosekliajam ir nenuosekliajam būdams.

Hipertekstas yra nauja technologija, todėl vartotojas, norintis naudotis hipertekstu, turi ugdyti įgūdžius – to reikia mokytis mokykloje. Skaitantysis turi mokėti suformuluoti tikslus, įvertinti jų sunkumą, sudaryti paieškos strategiją, gebėti palyginti gautą informacinę medžiagą, spręsti iškylančias problemas. Apie visa tai ir kalbama knygoje.

Dalis knygos – tai supažindinimas su „Toolbook“ sistemos taikymu. Čia nėra nagrinėjama pati sistema, o tik akcentuojamos jos ypatybės, taikymas kuriant mokomąsias programas. Pateikiama nemažai šia sistema sukurtų mokomųjų programų, jos aptariamos, išsamiu panagrinėjamos metodiškai svarbesnės vietos.

Knyga baigiama pamąstymų sukeliančiu skyreliu „Informacijos technologijos ir naujas požiūris į mokymą“. Čia autorė pateikia vieną svariausių savo teiginių: „Mokymo tikslas – sukurti naujomis informacijos technologijomis aprūpintą mokymo aplinką, kurioje būtų skatinami aktyvūs pažinimo procesai – supratimas, kritiškas nuomonių vertinimas, aptarimas ir kūrybiškumas“.

Knyga išleista parėmus Atviros Lietuvos fondui, todėl turėtų būti nesunku susirasti ją mokyklų bibliotekose. Šią knygą turėtų perskaityti visi mokytojai, besidomintys švietimo naujovėmis, ji turėtų nudžiuginti informatikus, ypač tuos, kurie žada kurti patys ar (dar geriau) su mokiniais mokomąsias programas.

APIE AUTORIUS

BALVOČIENĖ Tatjana, 1978 m. Vilniaus universitete įgijo matematikos specialybę. Informatikos mokytoja ekspertė. Dirba Šilutės 4-oje vidurinėje mokykloje. Domisi Logo pedagogika, „Intraneto“ taikymu mokykloje. Švietimo ir mokslo ministerijos Informatikos mokymo ekspertų komisijos narė.

DAGIENĖ Valentina, 1978 m. Vilniaus universitete įgijo taikomosios matematikos specialybę. Matematikos mokslų daktarė, docentė. Dirba Matematikos ir informatikos institute. Domisi programavimo kalbų teorija, algoritmavimo mokymo metodika, informatikos olimpiadinių uždavinių kūrimu. Dėsto informatikos didaktikos, mokyklinės informatikos, informacinės mokymo technologijos, programavimo kalbų kursus Vilniaus universitete bei Vilniaus pedagoginiame universitete. Švietimo ir mokslo ministerijos Informatikos mokymo ekspertų komisijos narė.

DAGYS Viktoras, 1976 m. Vilniaus universitete įgijo taikomosios matematikos specialybę. Dirba Matematikos ir informatikos institute. Domisi informatikos didaktika, algoritmavimo mokymo metodika, lietuviškų kompiuterijos terminų darybos bei vartojimo problemomis. Lietuvos kompiuterininkų sąjungos Tarybos pirmininko pavaduotojas. Švietimo ir mokslo ministerijos Informatikos mokytojų kvalifikacijos komisijos pirmininkas.

FRANZAS Michaelis, 1989 m. baigė Šveicarijos federalinį technologijos institutą (ETH) įgydamas inžinieriaus diplomą. Technikos mokslų daktaras. Nuo 1996 m. Kalifornijos universiteto (Irvine, JAV) informacijos ir informatikos fakulteto docentas. 1994–1995 m. dirbo Šveicarijos federalinio technologijos instituto Kompiuterinių sistemų institute, Niklausio Wirtho mokslinėje grupėje. Oberono kalbos realizacijos „Macintosh“ tipo kompiuteriams autorius. Domisi programavimo kalbomis, dinamiškai išplečiamomis programinės įrangos sistemomis.

GRIGAS Gintautas, 1959 m. Kauno politechnikos institute įgijo elektros mašinų ir prietaisų specialybę. Matematikos mokslų daktaras, docentas. Dirba Matematikos ir informatikos institute, Programavimo metodologijos skyriaus vadovas. Domisi programavimo kalbomis, programavimo mokymu, lietuvių kalbos vartojimo kompiuterijoje problemomis. Skaito programavimo kalbų ir duomenų tipų kursus Vilniaus universitete bei Vilniaus pedagoginiame universitete. Vadovauja keletui doktorantų bei magistrantų.

MARKAUSKAITĖ Lina, 1995 m. Vilniaus universitete įgijo taikomosios matematikos specialybę. Dirba Atviros Lietuvos fondo Naujųjų technologijų programos koordinuotoja. Nuo 1997 m. Matematikos ir informatikos instituto informatikos krypties doktorantė. Domisi informacijos ir komunikacijos technologijos taikymu mokymui, švietimo tyrimais.

SUMMARY

L. Markauskaitė. Computer Assisted Learning and Intelligent Tutoring Systems

There are two goals of this article. Firstly, the article discusses the conceptual framework for understanding computer assisted learning, i. e. underlying philosophy, types of computer based learning activities, concepts of *effective learning* and *effective educational software*, basic methods for evaluation of educational software. Secondly, different aspects of Intelligent Tutoring Systems (ITS) are described. The article overviews the framework and an internal structure of ITS. The strengths and weaknesses of ITS are summarised.

M. Franz. JAVA from the point of N. Wirth school

The paper investigates relationship between JAVA and Pascal programming languages. Most JAVA possibilities are not new, most of them were long time ago implemented into other programming languages, especially those close to Pascal. There are also discussions about the increasing interest in JAVA, coming innovations and some tries to forecast the future of JAVA.

T. Balvočienė. The possibilities of using mouse in LogoWriter system

The article explains how to apply LogoWriter commands **mousebuttons** and **mouseposition**. There are examples of procedures which can help children to learn how to use mouse successfully. The article is illustrated with turtle shape (used in the procedures) pictures.

G. Grigas. Codes of foreign letters

In the article it is explained how to input characters which are not found on computer keyboard but are present in 8-bit code table and font. Codes of Latin script letters which are in the second half of Windows code tables 1250, 1252, 1254, and 1257 are given.

V. Dagienė, V. Dagys. The Ninth Lithuanian Olympiad in Informatics

The 9th Lithuanian Olympiad in Informatics took place in the school year 1997-1998. Information about the structure and organisation of the Olympiad, the problems (tasks), and the final results can be found in the article.

Bibliography

Review of book "Būdas mokyti kitaip: Hipertekstinė mokymo aplinka" (New way to teach: Hypertext learning environment) by Aukšė Balčytienė published in Lithuania in 1998.

TURINYS

L. Markauskaitė. Kompiuterizuotas mokymas ir intelektualios mokymo sistemos	3
M. Franz. Java iš N. Wirtho mokyklos pozicijų	20
T. Balvočienė. Pelės panaudojimo galimybės „LogoWriter“ sistemoje	28
G. Grigas. Užsienio raidžių kodai	34
Programuotojų konkursai	
V. Dagienė, V. Dagys. Devintoji Lietuvos moksleivių informatikos olimpiada	40
Bibliografija	
V. Dagienė. Būdas mokyti kitaip	78
Apie autorius	80

Lietuvos Respublika
Matematikos ir informatikos institutas
INFORMATIKA Nr. 34
Vilnius, 1998

Institute of Mathematics and Informatics
INFORMATIKA Nr. 34
Vilnius, 1998

Atsakingas už leidinį V. Dagys.
Viršelį dailininkės L. Ausiejūtės.

Formatas 60x84/16. Apimtis 5,25 leidyb. apsk. l.
Tiražas 750 egz. Kaina sutartinė. Užs.39.

Leido Matematikos ir informatikos institutas
Akademijos g. 4, 2600 Vilnius.

Spausdino IĮ „Petro ofsetas“