ISSN 1392-0561. INFORMACIJOS MOKSLAI. 2007 42–43

Integruoto programavimo koncepcijų mokymo C++ kalba principai

Antanas Vidžiūnas

Vytauto Didžiojo universiteto docentas, daktaras
Vytautas Magnus University,
Assoc. Professor, PhD
Vileikos g. 8, LT-44404 Kaunas
Tel. (+370 37) 327 897
El. paštas: iianvi@vdu.lt

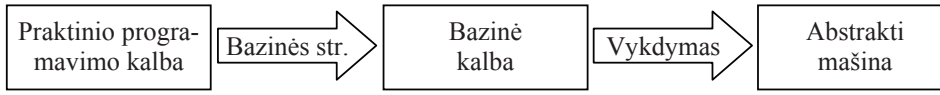
Artūras Mickus

Vytauto Didžiojo universiteto docentas, daktaras
Vytautas Magnus University,
Assoc. Professor, PhD
Vileikos g. 8, LT-44404 Kaunas
Tel. (+370 37) 327 897
El. paštas: arturas.mickus@if.vdu.lt

Straipsnyje aptariami klasikiniai istoriškai susiformavę programavimo mokymo organizavimo aukštosiose mokyklose metodai, analizuojami jų trūkumai ir pateikiami siūlymai, kaip šių trūkumų būtų galima išvengti įdiegus integruoto pagrindinių koncepcijų mokymo programą. Dažniausiai tokias mokymo programas siūloma realizuoti specialiose mokymo aplinkose, sudarytose funkcinio programavimo kalbų pagrindu. Rengiant taikomosios informatikos specialistus, toks mokymo būdas nėra geras, nes per daug formalus ir pagrįstas specialiomis matematinėmis specifikacijomis. Remiantis autorių patirtimi, analizuojama, kaip integruoto programavimo koncepcijų mokymo principai gali būti įdiegti naudojant mokymui universalios paskirties C++ kalbą. Aptariama atskirų mokymo dalykų struktūra. Pabrėžiama tikslaus pagrindinių sąvokų kūrimo ir įvairias koncepcijas apimančių teorinių pagrindų kūrimo svarba įvadiniuose programavimo dalykuose, nes juose įgytos žinios turi didelę įtaką informatikos studentų požiūriui į savo specialybę ir tolesnių studijų sėkmei. Glaustai aptariama racionalios mokymo aplinkos parinkimo problema.

Programavimo dalykai sudaro informatikos specialybių studentų mokymo programų pagrindą, tačiau vienodos nuomonės, kaip šie dalykai turi būti mokomi ir kokia turi būti jų struktūra, nėra. Kaip ir kiekvienoje taikomojo pobūdžio žinių srityje, skiriamos dvi pagrindinės programavimo dalys: praktinio taikymo technologija ir teorinis šios technologijos pagrindimas. Todėl natūralu tikėtis, kad abiem šioms dalims mokymo programose būtų skiriama vienodai dėmesio. Tačiau iš tiesų taip nėra. Dažniausiai naudojamos trys skirtingos programavimo mokymo metodikos: matematinė, pragmatinė ir koncepcinė.

Seniausias tradicijas turi 1976 metais suformuota matematinė programavimo mokymo metodika (Dijkstra, 1997), kurioje siūloma į programavimą žiūrėti kaip į matematikos šaką. Naudojant tokį mokymo būdą, sukuriamas griežtas teorinis pagrindas, tačiau jis neapima daugelio naujų programavimo koncepcijų, yra siaurai orientuotas į atskiras programavimo kalbas ir reikalauja gerų matematikos žinių. Be to, sudaromas įspūdis, kad programavimas nėra savarankiška žinių sritis, o tiktai diskrečiosios matematikos priedas, taisyklių rinkinys, kaip rašyti įvairių taikomųjų sričių uždavinių algoritmus. Todėl taikomosios informatikos specialistų rengimo programose populiaresnis pragmatinis mokymas (Hume, Barnard, 2006), kuris pagrįstas atskirų programavimo paradigmų (procedūrinės, objektinės, funkcinės, loginės ir kitų) arba jas realizuojančių programavimo kalbų praktinio naudojimo galimybių studijoms. Paradigmos suprantamos kaip rinkiniai pagrindinių koncepcijų, kuriomis



1 pav. ***Specializuotos integruoto programavimo koncepcijų mokymo aplinkos struktūra***

vadovaujantis gali būti parengiamos įvairių uždavinių sprendimo programos. Toks mokymo būdas leidžia sparčiau suteikti praktinio programavimo įgūdžius populiariomis programavimo kalbomis ir nuosekliai pereiti nuo paprastesnių prie sudėtingesnių paradigmų mokymo, tačiau jis turi daug trūkumų, nes nesukuriamas sisteminis požiūris į programavimą kaip į žinių sritį su aiškiais teoriniais pagrindais bei išsamia jos sprendžiamų problemų analize. Daug laiko sugaištama įvairių programavimo kalbų sintaksinių struktūrų, semantikos, skirtingai įvairiose paradigmose interpretuojamų fundamentalių koncepcijų pakartotinėms studijoms. Dargi nesutariama, kokia seka šių koncepcijų turi būti mokoma. Net ACM asociacijos rekomenduojamoje tipinėje informatikos specialistų rengimo programoje (Computing Curricula 2001, 2001) siūlomos įvairios programavimo paradigmų mokymo alternatyvos – pradedant nuo procedūrinio, objektinio, funkcinio arba loginio programavimo.

Nesunku pastebėti, kad daugelis įvairiose paradigmose vartojamų fundamentalių koncepcijų turi daug bendrų bruožų, todėl pastaruoju metu aiškėja tendencija organizuoti integruotą šių koncepcijų mokymą, pereinant nuo išsamių atskirų programavimo paradigmų studijų prie joms bendrų koncepcijų studijų. Integruotu būdu formuojami pagrindiniai programavimo įgūdžiai jau pirmųjų kursų studentams įdiegia platesnį požiūrį į specialybę ir sudaro puikias sąlygas tolesnei dalykinių žinių plėtrai specialiuose dalykuose, kurie skiriami programų inžinerijos, diskrečiosios matematikos, kompiuterinės grafikos, dirbtinio intelekto ir kitoms aktualioms problemoms.

Integruoto programavimo koncepcijų mokymo būdai

Integruotas programavimo koncepcijų mokymas organizuojamas dviem būdais: kuriant specialias programavimo mokymo aplinkas ir pritaikant šiam tikslui populiarias praktinio programavimo kalbas. Pačią programavimo sąvoką rekomenduojama apibrėžti kaip kompiuterio įrangos funkcionalumo išplėtimą, jos pritaikymą konkrečioms problemoms spręsti. Toks platus šios sąvokos apibrėžimas tinka visoms programavimo paradigmoms ir apima visus programų rengimo etapus: nuo sprendžiamų uždavinių specifikacijų sudarymo iki vykdymui konkrečioje aplinkoje tinkamos programos kodo sukūrimo.

Literatūroje plačiau aprašytos specialios koncepcijų programavimo mokymo aplinkos (Reinfelds, 2002; Van Roy, Haridi, 2004), kurių pagrindas yra įvairiose paradigmose naudojamų koncepcijų aprašymo bazinės kalbos ir šių kalbų struktūras realizuojančios abstrakčios mašinos (1 pav.). Tokiose aplinkose praktinio programavimo kalbos apibrėžiamos kaip bazinės kalbos išplėtimai, gaunami papildant šią kalbą naujomis sintaksinėmis struktūromis.

Bazinė kalba šiose mokymo aplinkose paprastai būna funkcinė arba loginė, nes jose tikslesnis naudojamų paradigmų teorinis pagrindimas ir paprastesnė sintaksė. Todėl toks mokymo būdas, kuriam būdingos specialios sprendžiamų uždavinių matematinio specifikavimo priemonės, populiarus matematikos studijose. Rengiant taikomosios informatikos specialistus, programavimo studijas pradėti nuo funkciniam ir loginiam programavimui būdingų formalių matematinių specifikacijų sudarymo vargu ar tikslinga, nes praktiniam programavimui šios priemonės naudojamos retai. Be to, toks abstraktus požiūris, kuris yra matematinio mokymo metodo modifikacija, nepritaikytas mokymui populiariose praktinio programavimo aplinkose ir neturintiems specialaus matematinio pasirengimo pirmųjų kursų studentams yra per sunkus. Tačiau patys integruoto koncepcijų mokymo principai ir specialiose mokymo aplinkose parengta jų

studijų metodika yra puikios priemonės, kurios leidžia racionaliau organizuoti programavimo mokymą ir padeda formuoti aiškesnius teorinius šios žinių srities pagrindus.

Įvairių autorių (Felleisen ir kiti, 2001; Deitel, 2000) patirtis rodo, kad integruotas koncepcijų studijas galima sėkmingai organizuoti ir tradicinėmis programavimo kalbomis, jeigu nuo pat pradžių studijų tikslas bus ne tik taikomasis programavimas, bet ir įvairias koncepcijas realizuojančių programavimo priemonių kūrimas. Geriausiai šiam tikslui tinka C++ kalba, kuri pritaikyta įvairioms programavimo technologijoms organizuoti, pasižymi nepaprastai gausiu įvairių priemonių rinkiniu ir yra viena plačiausiai naudojamų praktinio programavimo kalbų. Atskirose programavimo paradigmose naudojamoms koncepcijoms integruoti siūlomos tokios priemonės:

- bendrų sąvokų ir koncepcijų išskyrimas;
- pagrindinių sąvokų semantikos išplėtimas;
- įvairių koncepcijų naudojimo sprendžiant praktinius uždavinius analizė;
- specialių programavimo priemonių modeliavimas pasirinktos kalbos priemonėmis.

Ypač svarbus išsamus sąvokų apibrėžimas. Jas būtina apibrėžti taip, kad atspindėtų įvairių koncepcijų poreikius. Pavyzdžiui, integruoto mokymo požiūriu nepakankamas tradicinis net ir tokios paprastos sąvokos kaip reikšmė aiškinimas. Dažniausiai apsiribojama intuityviu supaprastintu požiūriu, kad tai yra skaičius, loginė reikšmė arba simbolių eilutė (teksto fragmentas). Šis požiūris neatitinka kompiuterinio duomenų tvarkymo principų, nes technologiniu požiūriu visi duomenys yra bitų arba baitų rinkiniai, o duomenų interpretavimas yra jų apdorojimo problema. Vėliau, kai tenka kalbėti apie sudėtingos sandaros objektus ir funkcijas bei šiuos objektus perduodančius parametrus, supaprastintą požiūrį sunku pakeisti ir pritaikyti prie naujo konteksto poreikių. Todėl siūlomas išplėstas duomenų reikšmės apibrėžimas: reikšmė yra bet kuris duomenų struktūrinis elementas, kurio vidinė struktūra paslėpta nuo vartotojo ir kuriam galima taikyti įvairius veiksmus. Šis apibrėžimas išplečia ir abstraktaus tipo, kuris suprantamas kaip duomenų ir jiems taikomų veiksmų junginys, sąvokos taikymo galimybes. Išplėtus reikšmės sąvoką, duomenų tipą galima suvokti ne tik kaip tradicinių duomenų aprašymo priemonių atmainą, bet ir kaip tam tikras savybes turinčių funkcijų grupę arba objektų klasę, o tai atitinka objektinio ir loginio programavimo požiūrį.

Pradinių programavimo įgūdžių ugdymas

Rengiant pradinių programavimo įgūdžių ugdymo programas, dažniausiai vadovaujamasi pragmatiniu požiūriu – programos turinys ir problematika pasirenkama atsižvelgiant į mokymo tradicijas ir į tai, kokių galimybių suteikia turima programinė įranga. Toks požiūris nėra blogas, tačiau būtina turėti galvoje, kad programavimo įrangos suteikiamos priemonės yra ne studijų tikslas, o studijoms parinktų koncepcijų praktinio taikymo galimybių iliustravimo priemonė.

Formuojant pradinius įgūdžius, siūloma apsiriboti keliomis paprastomis koncepcijomis, kurios leidžia parengti ir išbandyti nesudėtingas programas. Tokios yra duomenų tipų, funkcijų, rekursinių procesų, duomenų srautų ir modulinės programos struktūros formavimo koncepcijos. Jomis vadovaujantis, uždavinio sprendimas aprašomas kaip duomenų srautų formavimo, jų pertvarkymo ir pateikimo vartotojui veiksmų seka. Svarbu mokymo neperkrauti įvairių tipų uždavinių sprendimo algoritmų analize, nes tai daugiau atskirų taikomųjų sričių, bet ne programavimo technologijų problema. Be to, ryškus pradinio mokymo orientavimas į algoritmų analizę daugumoje Lietuvos universitetų yra pradinių informatikos specialybių kursų studentų, kurie neturi pakankamai matematikos žinių, prasto pažangumo pagrindinė priežastis.

Įvadiniame dalyke rekomenduojama daugiau dėmesio skirti pagrindinių programavimo sąvokų ir uždavinių sprendimo aprašymo priemonių įsisavinimui bei koncepcijų raidos suvokimui. Taip pat būtinos bent minimalios žinios apie programų rengimui naudojamos įrangos struktūrą bei jos naudojimo principus.

Tokių žinių formavimui ypač gerai tinka C++ kalba, kurioje pateikiamos priemonės leidžia ne tik manipuliuoti kalboje apibrėžtomis loginėmis duomenų struktūromis, bet ir analizuoti jų fizinio realizavimo kompiuterio atmintinėje principus. Tokiai analizei patogu naudoti sveikųjų skaičių reikšmių interpretavimą dvejetainiais vektoriais, duomenų suglaudinimo ir jų išskleidimo principų iliustravimą, masyvų kaip vientisų struktūrizuotų atmintinės laukų pateikimą ir jų dinaminio tvarkymo principų aiškinimą.

Visų pradiniams įgūdžiams formuoti siūlomų koncepcijų studijoms pakanka procedūrinio programavimo priemonių, tačiau tokiomis priemonėmis apsiriboti nepatariama, nes taip gali susiformuoti įspūdis, kad jų pakanka praktiniam programavimui. Be to, dauguma pagalbinių priemonių bibliotekų, be kurių praktinis programavimas neįmanomas, yra objektinės. Todėl jau įvadiniame dalyke būtina pabrėžti procedūrinio programavimo problemas, paaiškinti, kaip jos sprendžiamos vadovaujantis kitomis programavimo koncepcijomis, ir supažindinti su pagalbinėse bibliotekose pateikiamų objektų rinkinių naudojimo principais. Geriausiai tam tinka srautų ir tekstinių duomenų tvarkymo priemonių bibliotekos.

Sudėtingesnių programavimo įgūdžių ugdymas

Tradiciškai pragmatinio programavimo pagrindų mokymo programose šiam tikslui skiriami du dalykai („Programavimo įvadas“ ir „Duomenų tipai ir struktūros“), o vėliau pereinama prie išsamaus objektinio programavimo ir kitų paradigmų bei programavimo praktinio taikymo įvairiose studijų srityse. Organizuojant integruotą mokymą, objektinio programavimo koncepcijų studijos įtraukiamos į visus pagrindinių įgūdžių formavimo dalykus (1 lentelė), todėl šių koncepcijų studijoms skirti atskirą dalyką netikslinga. Jį siūloma pakeisti specialių programavimo priemonių studijomis, kurių tikslas būtų aiškinti objektinių tokių priemonių bibliotekų kūrimo ir jų panaudojimo praktiniam

1 lentelė. ***Pagrindinių programavimo įgūdžių formavimo dalykų struktūra***

Dalyko pavadinimas	**Pagrindinės koncepcijos**
Programavimo įvadas	Duomenų tipų ir funkcijų abstrakcijos, rekursija, duomenų srautų tvarkymas ir modulinė programų struktūra
Duomenų tipai ir struktūros	Abstraktūs tipai, klasės ir objektai, vidinės struktūros paslėpimas, polimorfiškumas, paveldimumas, klasių kompozicijos ir konteineriai
Specialios programavimo priemonės	Šabloninis ir komponentinis programavimas, procesų sinchronizavimas, kritinių situacijų apdorojimas

programavimui principus remiantis šabloninio programavimo, komponentinio ir konkurencinio programavimo koncepcijomis.

Visos tokioms studijoms reikalingos priemonės taip pat yra C++ kalboje arba populiariose jos pagalbinių priemonių bibliotekose. Dar viena svarbi koncepcija, kuri turi būti aptarta ugdant pagrindinius programavimo įgūdžius, yra kritinių situacijų išskyrimas ir apdorojimas. Būtina pabrėžti, kad kiekviena programa privalo būti parengta taip, kad ji pati gebėtų tokias situacijas aptikti, priimti pageidaujamus sprendimus dėl reakcijų į jas, ir iliustruoti, kaip tai daroma sukuriant centralizuoto pranešimų apie kritines situacijas apdorojimo klases.

Išsamias specialių programavimo priemonių studijoms reikalingas žinias turi suteikti „Duomenų tipų ir struktūrų“ dalykas, kurį siūloma paskirti abstrakčių duomenų tipų koncepcijai, jos realizavimui objektinio programavimo priemonėmis ir išsamioms objektinio programavimo koncepcijų studijoms. Pradėdami šio dalyko studijas, studentai jau įvadiniame programavimo dalyke turėtų būti supažindinti su pagrindinėmis objektinio programavimo sąvokomis ir mokėti naudotis tipinėmis srautų ir tekstų tvarkymo klasėmis. Paankstinant išsamias objektinio programavimo koncepcijų studijas, svarbu parinkti tokias jų iliustravimo priemones, kurias būtų galima naudoti sprendžiant tipinius prakti-

nio programavimo uždavinius. Daugelyje vadovėlių pateikti eklektiški klasių šeimų pavyzdžiai aprašant grafikos objektų, gyvūnų, automobilių ir kitų objektų šeimas tokiam tikslui netinka.

Nuoseklioms objektinio programavimo koncepcijų studijoms puikiai tinka dinaminių masyvų realizavimo priemonių analizė (Vidžiūnas, 2003). Visų pirma, siūloma atkreipti dėmesį į tai, kad net ir tipinis statinis masyvas iš esmės yra objektas, kurio vidinė struktūra ir fiziniai elementų atrankos veiksmai paslepiami nuo vartotojo. Kuriant dinaminį masyvą, vartotojui pačiam tenka formuoti jo vidinę struktūrą aprašančius parametrus, kurie informuotų apie masyvui skirtos atmintinės lauko dydį, jo pripildymą, ir apibrėžti masyvo tvarkymo veiksmus, kurie kontroliuotų jo pripildymą duomenimis, naujos atmintinės srities skyrimą ir duomenų perrašymą į šią sritį. Procedūrinio programavimo priemonėmis parengti tokių veiksmų aprašymai vartotojams, kurie užsiima tik taikomuoju programavimu ir suinteresuoti tai daryti kuo paprasčiau, būtų nepriimtini. Todėl vidinę dinaminių masyvų struktūrą bei pagalbinių veiksmų realizavimo priemones pageidaujama paslėpti objektinio programavimo priemonėmis ir sudaryti sąlygas dinaminius masyvus naudoti tokiu pat būdu, kaip ir statinius.

Sudėtingesnes objektinio programavimo sąvokas (paveldėjimą, dinaminį polimorfiškumą, abstrakčias ir šablonines klases, konteinerius, specialius klasių metodus, operatorių perdengimą ir kitas) patogu nuosekliai aptarti analizuojant dinaminio masyvo klasės pritaikymą aibių, tekstinių duomenų ir sudėtingos sandaros įrašų rinkinių tvarkymo poreikiams. Tradiciškai „Duomenų tipų ir struktūrų“ dalykas daug dėmesio skiria sąrašų tvarkymo priemonėms. Jos taip pat yra puikūs specializuotų konteinerių klasių praktinio pritaikymo pavyzdžiai.

Mokymo aplinkos parinkimo problema

Dar vienas svarbus veiksnys, į kurį tenka atsižvelgti formuojant pagrindinių programavimo įgūdžių ugdymo programas, yra tinkamos mokymo aplinkos parinkimas. Specialios programavimo mokymo aplinkos patrauklios tuo, kad jos pačios gal būti naudojamos kaip studijų objektas sudominant studentus tokių aplinkų palaikymu ir tobulinimu. Tačiau tai padaryti pajėgūs tiktai didieji universitetai, kurie jas kuria. Organizuoti mokymą kurios nors vienos profesinio programavimo aplinkos, pavyzdžiui, C++ Builder arba Visual C++, pagrindu taip pat nepatariama, nes tada mokymas siaurai orientuojamas į vienos firmos siūlomas programavimo priemones ir technologijas. Pirmųjų dviejų programavimo dalykų studijoms visiškai pakanka laisvai platinamų atviro kodo kompiliatorių, kuriuos nesunku papildyti įvairių pagalbinių priemonių bibliotekomis ir pritaikyti mokymo poreikiams, o trečiajam dalykui geriau tinka vaizdinės aplinkos, kurios papildomai leidžia susipažinti su našiomis automatizuoto programų kūrimo technologijomis, programų išorinės sąsajos kūrimo priemonėmis ir principais.

Išvados

Istoriškai susiformavę programavimo pagrindų mokymo principai, kai pagrindinėms programavimo paradigmoms skiriami atskiri mokymo programos dalykai, netenkina nūdienos poreikių, nes tokiose programose neišvengiama dubliavimo, skirtingo tų pačių sąvokų interpretavimo, neracionaliai išnaudojamas programavimo žinių ugdymui skiriamas laikas. Daugelio šių problemų galima išvengti įdiegus integruoto įvairių koncepcijų mokymo principus, kuriuos būtina pasitelkti jau įvadiniuose programavimo dalykuose, išplečiant pagrindines sąvokas ir modeliuojant įvairių koncepcijų elementus universalios paskirties programavimo kalbų priemonėmis.

LITERATŪRA

VAN ROY, P.; SEIF HARIDI, S. (2004). *Concepts, Techniques and Models of Computer Programming*. MIT Press.

DIJKSTRA, E. (1997). *A discipline of programming*. Prentice Hall.

Computing Curricula 2001. (2001). Prieiga per internetą: http://www.acm.org/education/curricula.html [žiūrėta 2005-09-12].

HUME, J.; BARNARD, D. (2006). *Programming: Concepts and Paradigms*. Holt Software Associates Inc.

REINFELDS, J. (2002). Teaching of programming with a programmer's theory of programming. In *Informatics Curricula, Teaching Methods, and Best Practice (ICTEM 2002)*. Kluwer Academic Publishers.

DEITEL, H.; DEITEL, P. (2000). *C++, How to programm*. Third edition. New York: Prentice Hall.

FELLEISEN, M.; FINDLER, R.; FLATT, M.; KRISHNAMURTHI, S. (2001). *How to Design Programs: An Introduction to Computing and Programming*. MIT Press.

VIDŽIŪNAS, A. (2003). *C++ ir C++ Builder pradmenys*. Kaunas: Smaltija.

PRINCIPLES OF INTEGRATED TEACHING PROGRAMMING CONCEPTS USING C++

Antanas Vidžiūnas, Artūras Mickus

S u m m a r y

The basic methods of teaching programming are presented and discussed paying attention to their disadvantages and possibilities to avoid them using integrated teaching of the main programming concepts. Usually the concept-based programming teaching is associated with using of special teaching environment on logical programming language basis. This approach isn't accessible in curricula for specialties in applied informatics because it is too formal and needs using of additional formal mathematical specifications. Experience of the authors shows, that it is possible to organize concept-based programming teaching using popular C++ language and structures of subjects for such studies are presented. It is emphasized that the main categories and concepts must be defined according the needs of all basic programming paradigms, because it has a great influence to the success in the further studies. Also the problem of using rational teaching environment is discussed.