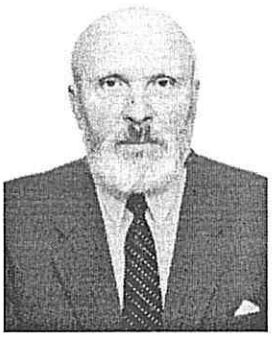
Lietuvos jaunųjų programuotojų mokykla

$\alpha + \omega$, 2000, Nr. 2, 77–79

Uždavinio sprendimo idėjos aprašas

Gintautas Grigas

grigas@ktl.mii.lt



Informatikos olimpiadų bei Jaunųjų programuotojų mokyklos užduotyse dažnai prašoma pateikti uždavinio sprendimo idėjos aprašą. Kam toks aprašas reikalingas ir kas jame turi būti parašyta?

Docentas G. Grigas — Matematikos ir informatikos instituto Programavimo metodologijos skyriaus vedėjas, daugelio knygų apie programavimą ir mokslo darbų autorius.

Nuo idėjos prie darbų

Norint išspręsti uždavinį, reikia turėti idėją, kaip jį spręsti. Ir gyvenime, ir mokykloje pasitaiko įvairių uždavinių. Vieni būna paprastesni, kiti sudėtingesni. Tiek Jaunųjų programuotojų mokykloje, tiek informatikos olimpiadose dėmesys koncentruojamas į intelektualiausią programavimo dalį — algoritmus. Pirmiausia reikia „perkąsti riešutą" — suvokti, kaip rašyti algoritmą. Ypač „kieti riešutai" būna olimpiadiniuose uždaviniuose. Tiktai sugalvojus uždavinio sprendimo idėją, ją visapusiškai apmąsčius ir įsitikinus, kad ji teisinga ir racionali, galima pradėti rašyti algoritmą. Vienas iš svarbiausių programuotojo priesakų yra: „Pirma galvok, po to programuok".

Idėja realizuojama algoritmu. Kam reikalingas atskiras sprendimo idėjos aprašas, jeigu ją perteikia algoritmas?

Darbo pristatymas visuomenei

Šiuolaikinėje visuomenėje nepakanka padaryti gerą darbą. Tam, kad darbas būtų deramai įvertintas, reikia mokėti jį pateikti. Sprendimo idėja ir yra svarbiausias intelektualaus darbo rezultatas, kurį galima trumpai suformuluoti ir pateikti straipsnyje, konferencijoje, seminare. Todėl reikia nuolat mokytis trumpai ir aiškiai formuluoti savo mintis.

Idėja realizuojama algoritmu. Tačiau algoritmas tėra tik pasyvus veiksmų, kuriuos reikia atlikti, kad būtų išspręstas uždavinys, aprašas. Už borto lieka veiksmų apibūdinimas gyvąja arba matematine kalba, pagrindimas, kodėl iš tikrųjų tokius veiksmus reikia atlikti, parodantis, kad uždavinys sprendžiamas teisingai ir racionaliai. Be to, užrašytas programavimo kalba algoritmas, nors ir skirtas žmogui, tačiau turi per daug techninių (kompiuterinių) detalių, todėl nėra skaitomas taip lengvai kaip romanas.

Programa ir jos dokumentacija

Uždavinio sprendimo idėjos aprašas yra programos dokumentacijos dalis. Profesionalus programuotojas rašo programą ne sau, o kitiems. Jų programos platinamos. Kiekviena platinimui skirta programa turi būti dokumentuota, t. y. turi būti pateikiamas ne tik programos tekstas kompiuteriui, bet ir įvairūs aprašai, paaiškinimai žmogui.

Vieni aprašai skirti programos vartotojui, kad jis žinotų, kuo ta programa gali būti naudinga ir kaip tą naudą išpešti. Tai dažniausiai būna elektroniniai žinynai, kuriuose aprašoma, kaip

pateikti pradinius duomenis, valdyti programos darbą ir suprasti jos rezultatus.

Kiti aprašai skirti programuotojams — programos autoriaus bendraminčiams bei bendradarbiams, kad jie galėtų suprasti, kodėl uždavinys buvo sprendžiamas taip, o ne kitaip, kodėl pasirinktas sprendimas iš tikrųjų yra geras. Tokie aprašai padeda tobulinti programą, joje priimtais sprendimais galima pasinaudoti rašant kitas programas. Uždavinio sprendimo idėjos aprašas irgi skiriamas programuotojams.

Štai todėl į daugelį Jaunųjų programuotojų mokyklos užduočių įtraukiama algoritmui artimiausia dokumentacijos dalis — uždavinio sprendimo idėjos aprašas. Tikimasi, kad tokie aprašai bent šiek tiek padės susipažinti su profesionaliam programavimui būdingu programų dokumentavimu.

Ką reikia ir ko nereikia rašyti?

Įsivaizduokime, kad gimė uždavinio sprendimo idėja ir ją norime pasufleruoti draugui, kuris programuoti moka, bet jam dar neatėjo į galvą mintis, kaip šį uždavinį spręsti. Trumpai ir aiškiai užrašę gimusią idėją jau būsime atlikę nemažą dalį darbo. Jeigu dar įsivaizduosime, kad draugas yra „neviernas Tamošius“ ir įtikinamai paaiškinsime, kad ta idėja iš tikrųjų yra teisinga ir pati geriausia, kokią tik galima sugalvoti, būsime sėkmingai įveikę visą uždavinį.

Idėjos aprašas — tai ne algoritmo ar programos aprašas. Jį rašant reikia įsivaizduoti, kad algoritmo dar nėra. Todėl apraše nereikia minėti algoritme vartojamų programavimo konstrukcijų, duomenų tipų ir pan. Idėją reikia aprašyti taip, kad ją galima būtų nagrinėti žiūrint į algoritmą. Jeigu reikia kokių nors žymenų, tai galima juos čia pat aprašyti (įvesti). Jei reikia duomenų struktūrų, reikia jas aprašyti. Tačiau geriau operuoti matematiniais, o ne programavimo kalbos terminais. Pavyzdžiui, jeigu veiksmai atliekami su duomenimis, surašytais į lentelę, tai reikia ir kalbėti apie lentelę, jos stulpelius bei eilutes, neverčiant lentelės masyvu (masyvas — tai jau lentelės realizacija).

Kada aprašyti idėją?

Geriausiai aprašyti idėją sekasi, kol ji nauja. Ką tik gimusias mintis lengviau užfiksuoti, nes vėliau jos blanksta, darosi lyg ir savaime aiškios. Be to, raštu išreikšta mintis yra tikslesnė, negu galvoje turima. Rašant tobulėja ne tik išraiškos forma, bet ir turinys, nes išryškėja netikslumai, prieštaravimai, klaidos. Kuo anksčiau klaidos pastebimos, tuo lengviau jas ištaisyti, tuo mažiau laiko sugaištama. Taigi idėją aprašyti geriau dar prieš jos realizavimą, t. y. prieš rašant algoritmą. Naujų minčių, keičiančių sprendimo idėją, gali atsirasti ir rašant algoritmą. Tada reikia koreguoti ir idėjos aprašą. Tačiau taisyti trumpą aprašą beveik visada paprasčiau negu algoritmą, kai gerai neapgalvojus (ir neaprašius) idėjos per toli nueinama klaidingu keliu.

Lietuvos moksleivių informatikos olimpiadų dalyviams paprastai už idėjos aprašą skiriami taškai nepriklausomai nuo to, ar ji realizuota ir kaip realizuota. Parašyti ir suderinti algoritmą dažnai užtrunka ilgiau, negu aprašyti jau subrandintą idėją. Pergalė (gerai veikianti programa) atrodo čia pat, o laikas vis bėga dėl kokios nors pasislėpusios apmaudžios klaidelės. Todėl iš karto aprašius ką tik gimusią idėją, jau galima užsitikrinti tam tikrą skaičių taškų.

Kur dėti aprašą?

Idėjos aprašas yra nepriklausomas nuo algoritmo. Todėl jį galima pateikti kaip atskirą savarankišką dokumentą, iliustruotą brėžiniais, lentelėmis. Taip siūloma daryti Jaunųjų programuotojų mokyklos klausytojams, kai jie sprendimus siunčia išspausdintus popieriuje.

Kadangi idėjos aprašas dažniausiai būna trumpas, tai jį galima įdėti į algoritmo (programos) pradžią kaip komentarą — algoritmo jis neužgoš. Šitaip reikalaujama olimpiadose, nes jose laiko nedaug ir nėra kada rūpintis iliustracijų piešimu bei teksto maketavimu. Be to, kai viskas surašyta į vieną bylą, paprasčiau tvarkyti darbus.

Pavyzdžiai

Pateiksime keletą uždavinių ir jų idėjų aprašų.

1 pavyzdys. Išskaidykite natūralųjį skaičių N pirminiais dauginamaisiais.

Tarkime, kad mokinys pateikė tokią šio uždavinio sprendimo idėją.

1 sprendimo idėja. *Tam tikrą skaičių dalijame iš visų už jį mažesnių pirminių skaičių. Jeigu dalijasi, tai ir bus pirminis dauginamasis.*

Tai nepakankamas ir netikslus aprašas, nes neparašyta, kaip gauti pirminius skaičius. Be to, skaičius gali turėti kelis vienodus dauginamuosius, pavyzdžiui, $12 = 2 \times 2 \times 3$. Todėl iš tokio idėjos aprašo mažai naudos.

Pateikiame kitą idėjos aprašą.

2 sprendimo idėja. *Skaičių N reikia dalyti paeiliui iš skaičių $n_i = 2, 3, 5, 7, 9, \ldots, (2m+1)$. Jeigu dalijasi, tai skaičius n_i, bus pirminis dauginamasis ir jį reikia įtraukti į rezultatus (pvz., išspausdinti), o skaičių N pakeisti dalmeniu. Skaičius N, gali turėti kelis vienodus dauginamuosius. Todėl minėtus veiksmus reikia kartoti su tuo pačiu dalikliu tol, kol dalijasi (be liekanos). Kai nebesidalija, reikia pereiti prie dalybos iš kito skaičiaus n_i. Skaičiavimai baigiami, kai dalmuo tampa lygus vienetui.*

Iš aprašo aišku, kad skaičius N dalijamas iš visų nelyginių skaičių, nors jie ne visi pirminiai. Tačiau nelyginiai sudėtiniai skaičiai į rezultatų sąrašą nepateks, kadangi prieš tai skaičius N jau buvo padalytas iš jį sudarančių pirminių daugiklių.

2 pavyzdys. Parašykite algoritmą, kuris nustatytų, keliais pirminiais dauginamaisiais galima išskaidyti skaičiaus faktorialą [1, p. 14].

Sprendimo idėja. *Skaičiaus faktorialo ieškoti nereikia. Pakanka išskaidyti pirminiais dauginamaisiais kiekvieną faktorialą sudarantį dauginamąjį: 2, 3, 4, 5, 6, ... Šitaip išvengiama perpildymo, kuris atsirastų skaičiuojant didesnių skaičių faktorialus.*

Apie tai, kaip skaidyti pirminiais dauginamaisiais faktorialą sudarančius skaičius, galima ir nerašyti, nes šio uždavinio esmė kaip tik čia ir yra, kad nereikia ieškoti faktorialo. Tai ir yra tas riešutas, kurį reikėjo perkąsti.

3 pavyzdys. *Atsiskaitymas skolomis.* Įmonės įsiskolinusios viena kitai. Jeigu įmonė turėtų pinigų ji galėtų skolas sumokėti. Tačiau jų neturi. 0 neturi todėl, kad jai negrąžina skolų kitos įmonės. Kitos įmonės negali sumokėti savo skolų dėl tos pačios priežasties — jos negauna pinigų iš joms skolingų įmonių. Susidaro uždaras ratas. Tačiau jį galima pralaužti. Pasirodo, skolas galima sumokėti ... skolomis.

Parašykite programą, kuri išanalizuotų įmonių skolas ir jas pertvarkytų taip, kad visų įmonių įsiskolinimų suma būtų mažiausia [1, p. 36].

Sprendimo idėja. *Sprendimas paprastas: sudaromas fiktyvus bankas su visų įmonių sąskaitomis. Iš pradžių visos sąskaitos tuščios — lygios nuliui. Jeigu įmonė A skolinga įmonei B, tarkime,* 100 *litų, tai iš įmonės A sąskaitos atimama* 100 *litų, o prie įmonės B sąskaitos pridedama* 100 *litų. Šitaip sudėjus visas skolas, kiekvienos įmonės balansas bus teigiamas arba neigiamas ir įsiskolinimai mažiausi. Bet bankas fiktyvus, todėl jį reikia panaikinti — sąskaitose esančius pinigus išreikšti įmonių skolomis. Tą galima padaryti išskolinus įmonių, turinčių teigiamas sąskaitas, pinigus įmonėms, turinčioms neigiamas sąskaitas. Kadangi uždavinio sąlygoje neribojamos nei naujų skolų santykių įmonių poros, nei skolinimų skaičius, tai skolų perskirstymą galima atlikti bet kokia tvarka, geriausia kuo paprasčiau — iš eilės.*

Iš idėjos aprašo išryškėja svarbiausias dalykas — nereikia analizuoti painaus skolų grafo, o pakanka visas skolas sukrauti į krūvą. Šią idėją suvokus, jau galima rašyti algoritmą. Jį detalizuojant, gali prireikti naujų sprendimų. Tačiau tai jau smulkesni uždaviniai ir jų sprendimo galima nebeaprašyti. Svarbiausia yra pati idėja, kaip susidoroti su esmine uždavinio dalimi.

1. V. Dagienė, J. Skūpienė, *Moksleivių informatikos olimpiadų uždaviniai.* I–VII olimpiados, TEV, Vilnius, 1999.