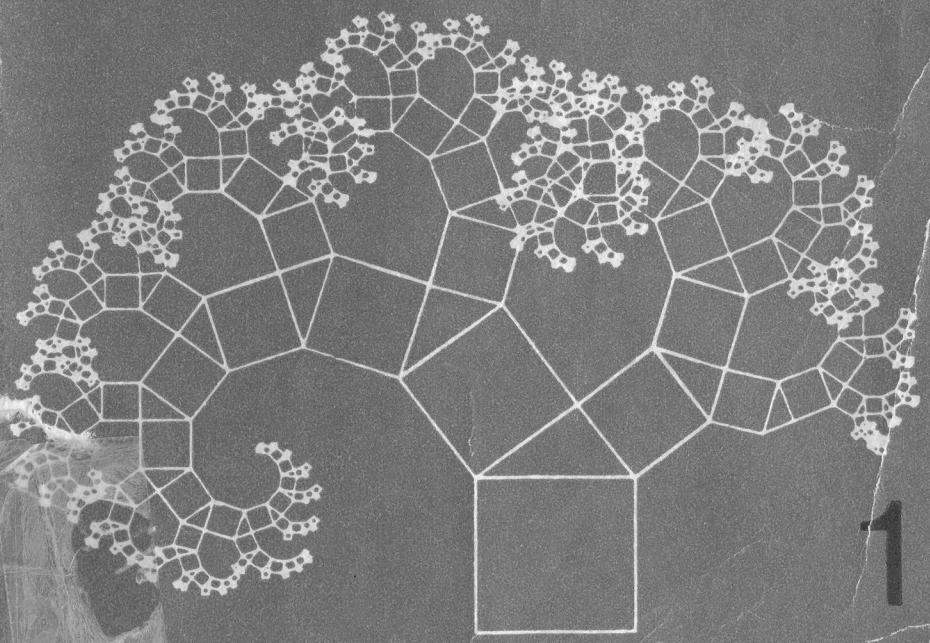
10 kp

ISBN 5—430—00111—2

PROGRAMAVIMO MOKYKLA

Valentina Dagienė

MOKOMĖS PROGRAMUOTI

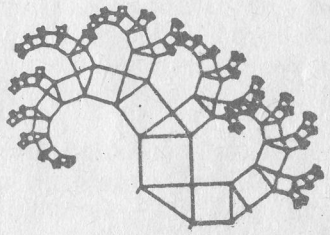


1

Serija „PROGRAMAVIMO MOKYKLA"

Valentina
Dagienė

MOKOMĖS PROGRAMUOTI



KAUNAS
„ŠVIESA"
1989

BBK 74.202.4
Da 77

Knygą recenzavo Varnių vidurinės mokyklos mokytojas
TOMAS KINCIUS

Serija leidžiama nuo 1989 m.

Dagienė V.
Da77 Mokomės programuoti / [Pratarmė, p. 3—4 ir Pabaigos žodis, p. 53—54, aut.] — K.: Šviesa, 1989.— 56 p.—(Programavimo mokykla).

Bibliogr. kn. gale.

ISBN 5—430—00111—2

Pradedamos leisti mokyklai skirtų leidinių serijos „Programavimo mokykla" pirmoji knygelė parengta „Komjaunimo tiesoje" skelbtos medžiagos pagrindu. Ji supažindina su bendraisiais programavimo principais, pataria, kaip pačiam pradėti programuoti, moko nagrinėti kitų sudarytas programas.

BBK 74.202.4
51

D $\frac{4802030000-120}{M\,853(10)-89}$ 33—89

ISBN 5—430—00111—2

PRATARMĖ

Elektroninė skaičiavimo mašina, vis dažniau vadinama kompiuteriu,— duomenų apdorojimo mašina. Norint gauti reikiamus rezultatus, pateikus duomenis, reikia nurodyti, ką ir kaip turi daryti kompiuteris, kitaip sakant, parengti jam suprantamą užduotį — parašyti programą. Tas pats kompiuteris gali spręsti įvairiausius uždavinius: ieškoti kelio labirinte, versti iš vienos kalbos į kitą, nurodyti gramatines klaidas rašinyje, piešti ornamentus, žaisti šachmatais, konsultuoti mokinius prieš egzaminą ar net egzaminuoti juos. Tam reikia tik sudaryti atitinkamą programą ir įrašyti ją į kompiuterį. Visos jo gudrybės, kurios kartais atrodo paslaptingos, ir slypi programose.

Programa turi būti griežta, vienareikšmiška — užrašyta kuria nors programavimo kalba. Šių kalbų yra daug. Kai kurios skirtos specialiems uždaviniams spręsti (pavyzdžiui, Fortranas — techniniams), kitos kalbos (PL/1, Algol-68, Ada) universalesnės, bet ir sudėtingesnės.

Šioje knygelėje programas užrašysime Paskalio (Pascal) kalba. Ji universali, tačiau gana paprasta, ją lengva išmokti.

Kai kas mano, kad išmokti programuoti — tai visų pirma išnagrinėti kompiuterį ir jo programavimo kalbas. Tačiau kur kas svarbiau suprasti bendrus programavimo principus, suvokti pagrindines programavimo kalbų konstrukcijas.

Knygelės tikslas — supažindinti skaitytojus su bendraisiais programavimo principais, išmokyti juos nagrinėti kitų sudarytas programas ir programuoti. Ji parengta remiantis „Komjaunimo tiesoje" skelbtomis pamokomis. Skiriama Jaunųjų programuotojų neakivaizdinės mokyklos pradinio kurso klausytojams ir pradedantiems studijuoti programavimą.

Leidinys suskirstytas į 13 maždaug vienodos apimties pamokų. Kiekvienoje gausu pavyzdžių. Gerai išnagrinėkite juos ir pabandykite patys sugalvoti ir išspręsti panašių uždavinių. Po kiekvienos pamokos pateikiami kartojimo uždaviniai. Pirmiausia pabandykite savarankiškai juos išspręsti. Jei nepavyks, pakartokite atitinkamą pamoką.

Kai kurių uždavinių (jie pažymėti žvaigždute) pateikti sprendimai. Tam pačiam uždaviniui spręsti galima sudaryti daug

įvairių programų. Duotas sprendimo variantas tėra vienas iš daugelio galimų. Galima parašyti kitokį, netgi geresnį.

Knygelėje pateiktai medžiagai suprasti pakanka elementarių matematikos žinių, įgytų vidurinės mokyklos žemesnėse klasėse. Išnagrinėjus ją, galima sėkmingai sudaryti visiškai užbaigtas nedidelių uždavinių programas.

Nuoširdžiai dėkoju technikos mokslų kandidatui G. Grigui, taip pat K. Augučiui, V. Dagiui, Z. Kuralavičiūtei už dalykiškas pastabas, pasiūlymus, kitokią pagalbą rengiant spaudai šį leidinį.

Autorė

Pirmoji pamoka

SVEIKIEJI SKAIČIAI

Kompiuteris atlieka veiksmus su duomenimis. Jie įrašomi į kompiuterio atmintį, kurią galima įsivaizduoti kaip popieriaus lapą, padalytą į laukelius. Tuose laukeliuose ir surašomi duomenys — vienas skaičius kiekviename. Jeigu į tą patį laukelį reikia įrašyti naują duomenį, anksčiau ten buvęs duomuo ištrinamas.

Kompiuteris skaito atmintyje (laukeliuose) esančius duomenis, atlieka su jais reikiamus veiksmus, po to rezultatus vėl įrašo į atmintį. Kokius veiksmus ir su kuriais duomenimis atlikti, kur įrašyti rezultatą, nurodoma programoje.

Duomenys gali būti įvairūs: skaičiai (sveikieji ir trupmeniniai), raidės, tekstas. Tai pareina nuo sprendžiamo uždavinio. Šioje knygelėje vartosime tik sveikuosius skaičius. Todėl dažnai sakysime tiesiog „skaičius".

Sveikieji skaičiai suprantami taip, kaip matematikoje. Paskalio kalba su sveikaisiais skaičiais galima atlikti tik šias aritmetines operacijas (veiksmus): sudėtį, atimtį, daugybą ir dalybą. Sudėtis ir atimtis žymima įprastai: + ir —. Daugybos ženklas yra * (žvaigždutė). Atkreipiame dėmesį, kad daugybos ženklo (*) jokiu būdu negalima praleisti, kaip kartais daroma matematikoje.

Dalijant skaičius sveikųjų skaičių tikslumu, gaunami du rezultatai: dalmuo ir liekana. Pavyzdžiui, 14 padalijus iš 5, gaunamas dalmuo 2 ir liekana 4. Sprendžiant uždavinius, kai kada reikia tik dalmens, o kartais — tik liekanos. Paskalio kalba turi dvi dalybos rūšis: **div** ir **mod**. *Jeigu reikia gauti dalmenį, tai operacija žymima* **div**, *jeigu liekaną* — **mod**.

Pavyzdžiui:

13 **div** 5=2,	3 **div** 8=0,
13 **mod** 5=3,	3 **mod** 8=3,
5 **div** 5=1,	0 **div** 7=0,
5 **mod** 5=0,	0 **mod** 7=0.

Kaip ir matematikoje, dalyti iš nulio negalima. Dalijant neigiamus skaičius, operacija **div** atliekama taip: skaičiai dalijami

be ženklų (kaip teigiami), gauto rezultato — dalmens ženklas nustatomas kaip ir matematikoje dauginant ar dalijant skaičius, pavyzdžiui:

-8 **div** $(-3) = 2$,
8 **div** $(-3) = -2$.

Neigiamų skaičių liekaną įsivaizduoti sunkiau, todėl operaciją **mod** atliksime tik su neneigiamais skaičiais.

Reiškiniai sudaromi taip, kaip ir matematikoje. Vartojami tik lenktiniai skliaustai. Operacijų atlikimo tvarka niekuo nesiskiria nuo įprastos: pirmiausia atliekami veiksmai skliaustuose, po to — dauginama ir dalijama, o galiausiai — sudedama ir atimama. Vieno rango operacijos (pavyzdžiui, daugyba ir abi dalybos rūšys) atliekamos paeiliui iš kairės į dešinę. Pavyzdžiui, reiškinys

10 **div** 6 **mod** 3+4 ∗ 3 **mod** 5

apskaičiuojamas taip, lyg būtų šitaip suskliaustas:

((10 **div** 6) **mod** 3) + ((4 ∗ 3) **mod** 5).

Jo reikšmė lygi 3.

Pateiksime keletą reiškinių pavyzdžių:

a) 8 **div** 6 **mod** 4 ∗ 2;
b) 8 **div** (6 **mod** 4) ∗ 2;
c) 8 **div** 6 **mod** (4 ∗ 2);
d) 8 **mod** (6 **mod** 4 ∗ 2);
e) 18 **mod** 7 ∗ 2 − 3 **mod** 7.

Apskaičiuokite šių reiškinių reikšmes. Turite gauti tokius rezultatus: a) 2; b) 8; c) 1; d) 0; e) 5.

KARTOJIMO UŽDAVINIAI

1. Apskaičiuokite šių uždavinių reikšmes:

a)* 6 **mod** 7+5 **div** 4 ∗ 2;
b)* 6 **mod** (7+5) **div** 4 ∗ 2;
c)* 6 **mod** ((7+5) **div** 4) ∗ 2;
d) 7 **div** 5 **mod** 3 ∗ 2;
e) 7 **div** (5 **mod** 3) ∗ 2;
f) 7 **div** 3 **mod** 5 ∗ 2;
g) 2 **div** 3+24 **mod** 6;
h) 3 **div** 2+6 **mod** 24;
i) (3 **div** 2+6) **mod** 24;
j) 1+49 **mod** 5+7 **div** 4;
k) (1+49) **mod** 5−7 **div** 4;
l) 1+49 **mod** (5+7) **div** 4;
m) (1+15 **div** 3) **mod** 3;
n) 1+(15 **div** 3) **mod** 3;
o) 1+15 **div** 3 **mod** 3;
p) 6 **mod** 7+5 **div** 3;
r) 6 **mod** (7+5 **div** 3);
s) 6 **mod** ((7+5) **div** 3).

2. Su kuriomis x reikšmėmis teisingos šios lygybės?

a) x **div** $5 = x$ **mod** 5;
b) 20 **div** $x = 20$ **mod** x;
c)* x **div** $5 = 8$;
d)* 50 **div** $x = 7$.

3. Su kuriomis a ir b reikšmėmis teisingos šios lygybės?

a) a **div** $b * b = a$;
b) $a * b$ **div** $a = b$;
c) $a * (b$ **div** $a) = b$;
d) $a * b$ **div** $b = a$;
e) $a * (b$ **div** $b) = a$;
f) a **div** $b * b + a$ **mod** $b = a$.

4. Suraskite tokią k reikšmę, su kuria lygtis

k **div** $x = k$ **mod** x

turėtų:

a) tris sprendinius;
b) penkis sprendinius.

99 div x = 99 mod x

10
32
38

Antroji pamoka

KINTAMIEJI IR SAKINIAI

Programoje gali būti daug duomenų. Kad galėtume nurodyti, su kuriais jų reikia atlikti veiksmus, duomenys žymimi *vardais*.

Vardai sudaromi iš lotyniškosios abėcėlės raidžių ir skaitmenų. Svarbu, kad vardai prasidėtų raide ir juose nebūtų tarpų. Kai kada vardas gali būti užrašytas ir viena raide. Vardų pavyzdžiai: *a*, *x*1, *x*2, *pi*, *suma*.

Vardu pažymėtas duomuo vadinamas *kintamuoju*. Programa lengviau suprantama, jei vardai parenkami taip, kad atspindėtų duomenų prasmę. Pavyzdžiui, skaičių sumas reikėtų žymėti vardu *suma* arba *s*, laipsnio rodiklį — *rod* ir panašiai.

Atliekant programą, kiekvienam kintamajam paskiriamas laukelis atmintyje. Vadinasi, kintamojo vardą galima laikyti ir laukelio vardu. Laukeliuose įrašomos kintamųjų reikšmės — konkretūs skaičiai (ar kiti duomenys). Atliekant programą, šios reikšmės gali būti keičiamos.

Reikšmių įrašymas į laukelius vadinamas *priskyrimu* ir žymimas priskyrimo simboliu := (šio simbolio nereikia painioti su panašių į jį lygybės simboliu =). Kairėje priskyrimo simbolio pusėje rašomas vardas to kintamojo, kuriam reikia priskirti reikšmę, o dešinėje — reikšmė, kurią reikia priskirti kintamajam. Pavyzdžiui,

$a := 3$

Čia dešinėje pusėje parašytas skaičius. Tačiau dažniausiai rašomas reiškinys, kurio reikšmę reikia apskaičiuoti ir priskirti kairėje priskyrimo simbolio pusėje esančiam kintamajam, pavyzdžiui,

$s := x + y$

Šiuo atveju į reiškinį $x+y$ reikia įrašyti kintamųjų x ir y reikšmes, apskaičiuoti jo reikšmę ir gautą rezultatą priskirti kintama-

jam s. Jeigu kintamajam x buvo priskirta reikšmė 4, kintamajam y — reikšmė 5, tai kintamajam s kompiuteris priskirs reikšmę 9.

Veiksmai, kuriuos turi atlikti kompiuteris, nurodomi *sakiniais*. Sakinių yra įvairių rūšių, ir jų vartojimas pareina nuo to, kokį veiksmą reikia nusakyti. Reikšmės priskyrimas kintamajam nusakomas *priskyrimo sakiniu*. Taigi $a:=3$ ir $s:=x+y$ yra priskyrimo sakiniai.

Kai tam pačiam kintamajam priskiriama nauja reikšmė, ankstesnė (prieš tai buvusi) panaikinama. Pavyzdžiui, atlikus sakinius

$k:=3;$
$k:=k+2;$
$k:=2*k$

kintamojo k reikšmė bus lygi 10 (pirmuoju sakiniu kintamajam k priskyrėme reikšmę 3; atlikus antrąjį sakinį, k reikšmė padidėjo dviem, t. y. pasidarė lygi 5; trečiuoju sakiniu prieš tai buvusią k reikšmę padvigubinome — ji pasidarė lygi 10).

Užrašyti sakiniai sudaro sakinių seką. *Vienas sakinys nuo kito skiriamas kabliataškiu.* Sakiniai atliekami iš eilės, kaip užrašyti sekoje. Pavyzdžiui, atlikus kiek pakeistą sakinių seką

$k:=3;$
$k:=2*k;$
$k:=k+2$

kintamojo k reikšmė bus lygi 8.

1 p a v y z d y s. Parašykime priskyrimo sakinį kintamojo y reikšmei apskaičiuoti pagal formulę

$y=a^2+b^3.$

Programoje galima vartoti tik tas operacijas ir tik tuos sakinius, kurie yra Paskalio kalboje. Paskalio kalba neturi kėlimo laipsnių operacijos, todėl šį veiksmą reikia išreikšti daugyba:

$y:=a*a+b*b*b$

2 p a v y z d y s. Parašykime sakinių seką kintamųjų x ir y reikšmėms sukeisti vietomis:

$p:=x;$
$x:=y;$
$y:=p$

Papildomo kintamojo p reikšmės reikėjo tam, kad būtų išsaugota kintamojo x reikšmė (atlikus sakinį $x:=y$, kintamojo x reikšmė tampa lygi y reikšmei, o prieš tai buvusi x reikšmė dingsta).

3 p a v y z d y s. Kintamojo *trio* reikšmė — triženklis natūrinis skaičius. Parašykime sakinių seką šio skaičiaus skaitmenų sumai apskaičiuoti.

Paskalio kalba neturi operacijos, kuri išskirtų skaičiaus skaitmenis. Tai reikia išreikšti arimetinėmis operacijomis.

Pirmąjį triženklio skaičiaus skaitmenį gauname, dalydami šį skaičių iš 100:

$s1:=$ *trio* **div** 100

Antrąjį skaitmenį gauti sudėtingiau. Pirmiau reikia atmesti pirmąjį skaičiaus skaitmenį, po to likusį dviženklį skaičių dalyti iš 10:

$s2:=$ *trio* **mod** 100 **div** 10

Galima ir kitaip — pirmiau atmesti trečiąjį skaitmenį, po to — pirmąjį:

$s2:=$ *trio* **div** 10 **mod** 10

Trečiasis skaitmuo lygus liekanai, gautai padalijus skaičių iš 10:

$s3:=$ *trio* **mod** 10

Taigi skaičiaus *trio* skaitmenų sumą galima apskaičiuoti šitokia sakinių seka:

$s1:=$ *trio* **div** 100;
$s2:=$ *trio* **div** 10 **mod** 10;
$s3:=$ *trio* **mod** 10;
$suma:=s1+s2+s3$

KARTOJIMO UŽDAVINIAI

5*. Duotos tokios kintamųjų reikšmės: $a=1$, $b=5$. Kokios bus jų reikšmės, atlikus šią sakinių seką?

$a:=b;$
$b:=a$

6*. Kokios bus kintamųjų x ir y reikšmės, atlikus šitokias sakinių sekas?

a) $x:=15$ **div** (8 **mod** 3);
$y:=17$ **mod** $x*5-19$ **mod** $5*2$;
b) $x:=2*5$ **div** 3 **mod** 2;
$y:=2*5$ **div** (3 **mod** 2);
$x:=x*y;$
$y:=y*y$

7. Kokia bus kintamojo sk reikšmė, atlikus šią sakinių seką?

$sk:=5;$
$sk:=sk+sk;$
$sk:=sk+sk;$
$sk:=sk$ **mod** 7

8. Kokia bus kintamojo *galutinis* reikšmė, atlikus šią sakinių seką?

pradinis := 7;
tarpinis := *pradinis* + *pradinis* **mod** 50;
galutinis := *tarpinis* **div** (*tarpinis* − 30)

9. Kokia bus kintamojo *trečias* reikšmė, atlikus šią sakinių seką?

pirmas := 20 **div** 3 **mod** 5;
antras := *pirmas* **mod** 5 + 5 * 2;
trečias := *pirmas* **mod** 2 + *antras* **div** 5

10. Pakelkite reiškinius laipsniu, vartodami kuo mažiau sudėties bei daugybos operacijų ženklų:

a)* x^7;
b) $(x+y)^8$;
c) x^9;
d) $(a+1)^{21}$;
e) $(a+b+c)^{32}$;
f) $(a+b)^{100}$.

11. Parašykite sakinių seką, pagal kurią apskaičiuotume, kiek *n* parų turi valandų, minučių ir sekundžių.

12*. Kintamojo *a* reikšmė — triženklis natūrinis skaičius. Rezultatas — dviženklis skaičius *b*, gautas iš skaičiaus *a* pašalinus vidurinįjį skaitmenį. Parašykite sakinį (ar sakinių seką) skaičiui *b* gauti.

13. Parašykite sakinių seką kintamųjų *a*, *b* ir *c* reikšmėms sukeisti vietomis (kintamasis *a* turi įgyti kintamojo *b* reikšmę, kintamasis *b* — kintamojo *c*, o kintamasis *c* — kintamojo *a*).

14. Parašykite sakinių seką, kintamojo *x* reikšmei padidinti du kartus, o kintamojo *y* reikšmei — tris kartus.

15. Kintamojo *dv* reikšmė — dviženklis natūrinis skaičius. Parašykite sakinių seką kintamojo *atv* reikšmei, gautai sukeitus vietomis skaičiaus *dv* skaitmenis, rasti.

16. Kintamųjų *pirmas* ir *antras* reikšmės — natūriniai triženkliai skaičiai. Parašykite sakinį, pagal kurį kintamajam *sandauga* būtų priskiriama kintamųjų *pirmas* ir *antras* vidurinių skaitmenų sandauga.

17. Kintamojo *penk* reikšmė — penkiaženklis natūrinis skaičius. Parašykite sakinį (arba sakinių seką):

a) antrajam šio skaičiaus skaitmeniui su paskutiniuoju sukeisti,
b) viduriniajam skaitmeniui pašalinti,
c) trim viduriniesiems skaitmenims pašalinti,
d) šio skaičiaus skaitmenų kvadratų sumai apskaičiuoti,

Trečioji pamoka

SĄLYGINIS SAKINYS

Matematikoje, kaip ir gyvenime, dažnai tenka pasirinkti vieną veiksmą iš kelių galimų. Šioje pamokoje paaiškinsime, kaip užrašyti vieno veiksmo parinkimą iš dviejų galimų.

Išnagrinėkime pavyzdį. Tarkim, kintamojo *f* reikšmė priklauso nuo kintamojo *a* reikšmės. Kai ši teigiama, kintamajam *f* reikia priskirti 1, o kai neigiama arba lygi nuliui — nulį. Taigi turi būti atliekamas vienas iš dviejų sakinių: $f:=1$ arba $f:=0$, priklausomai nuo to, ar $a>0$, ar $a\leqslant 0$. Matematiškai tai galima užrašyti šitaip:

$$f=\begin{cases}1, & \text{kai } a>0,\\ 0, & \text{kai } a\leqslant 0.\end{cases}$$

O kaip tokį veiksmą nurodyti kompiuteriui? Kadangi sudarant programą, kintamojo *a* reikšmė nežinoma (be to, ji gali būti parenkama vis kitokia, o programa turi tikti ta pati), tai reikia iš anksto numatyti visus galimus atvejus (sprendimo variantus) ir programoje nurodyti, kuriuos veiksmus atlikti kiekvienu konkrečiu atveju. Vadinasi, reikiamą variantą turi parinkti pats kompiuteris, spręsdamas uždavinį (t. y. atlikdamas programą), kai kintamojo *a* reikšmė jau yra jo atmintyje. Suformuluokime nurodymą minėtam uždaviniui spręsti: "patikrinti, ar $a>0$; jeigu taip, tai atlikti sakinį $f:=1$, priešingu atveju — sakinį $f:=0$".

Programavimo kalbose vartojami trumpesni, lakoniškesni būdai varianto parinkimui užrašyti — *sąlyginiai sakiniai*. Veiksmai nurodomi sutartiniais ženklais bei žodžiais. Sąlyginis sakinys nagrinėtiems veiksmams atlikti Paskalio kalba užrašomas šitaip:

if $a>0$
  then $f:=1$
  else $f:=0$

Šiuo sąlyginiu sakiniu nurodoma atlikti vieną veiksmą iš dviejų: $f:=1$ arba $f:=0$, priklausomai nuo kintamojo *a* reikšmės. Čia vartojami sutartiniai angliški žodžiai. Į lietuvių kalbą jie verčiami šitaip:

if — jeigu,
then — tai,
else — priešingu atveju (kitaip).

Iš kitos pamokos paaiškės, kad po žodžių **then** ir **else** galima rašyti ne tik priskyrimo, bet ir bet kurį kitą sakinį, netgi vėl sąlyginį.

Sąlyginį sakinį sudaro sąlyga, einanti po žodžio **if**, *ir dar du sakiniai, einantys po žodžių* **then** *ir* **else**. *Atliekamas tik vienas iš*

jų. Jei sąlyga tenkinama, tai atliekamas sakinys, einantis po žodžio **then**, *jei ne — sakinys, einantis po žodžio* **else**.

Sąlygoje skaičiams lyginti vartojamos šitokios operacijos:

$<$ mažiau,
$\leqslant$ mažiau arba lygu,
$>$ daugiau,
$\geqslant$ daugiau arba lygu,
$=$ lygu,
$\neq$ nelygu.

Lyginamos algebrinės skaičių reikšmės, t. y. atsižvelgiama ir į jų ženklus. Pavyzdžiui, sąlyga $1<2$ tenkinama, o sąlyga $-1<-2$ — netenkinama.

Gali būti lyginamos ne tik kintamųjų, bet ir reiškinių reikšmės, pavyzdžiui, $a+b$ **div** $2\geqslant a$ **mod** 10. Įsidėmėkite, kad lyginama paskiausiai — pirmiausia atliekamos visos aritmetinės operacijos.

1 p a v y z d y s. Tradicinis programavimo uždavinys — mažiausios (ar didžiausios) iš kelių duotų kintamųjų reikšmių radimas. Parašykime sąlyginį sakinį, pagal kurį kintamajam *min* būtų priskiriama viena (mažesnioji) iš dviejų kintamųjų a ir b reikšmių. Kai kintamųjų a ir b reikšmės lygios, kintamajam *min* galima priskirti bet kurią reikšmę. Užrašome tokį sąlyginį sakinį:

```
if a<b
   then min:=a
   else min:=b
```

arba šitokį:

```
if a≤b
   then min:=a
   else min:=b
```

Pirmuoju sakiniu, kai a ir b reikšmės lygios, kintamajam *min* priskiriama b reikšmė, antruoju — a reikšmė. Tai visiškai nesvarbu, juk $a=b$. Taigi abiem sakiniais gaunamas tas pats rezultatas.

2 p a v y z d y s. Parašykime sąlyginį sakinį, kuriuo kintamajam k būtų priskiriama kintamojo a kvadrato reikšmė, kai skaičius a nelyginis, arba kubo reikšmė, kai a lyginis.

```
if a mod 2=1
   then k:=a*a
   else k:=a*a*a
```

Kartais veiksmą reikia atlikti tik tuo atveju, kai sąlyga tenkinama. Tam vartojamas sutrumpintas sąlyginis sakinys (be žodžio **else**):

```
if ...
   then ...
```


Toks sakinys vadinamas suprastintu sąlyginiu sakiniu.

3 p a v y z d y s. Kintamojo a reikšmę pakeiskime jos moduliu. Jei $a<0$, tai skaičiaus a ženklą reikia pakeisti priešingu, t. y. atlikti sakinį $a:=-a$; kai $a\geqslant 0$, nereikia atlikti jokio veiksmo. Visa tai galima užrašyti suprastintu sąlyginiu sakiniu:

```
if a<0
   then a:=-a
```

4 p a v y z d y s. Pakeiskime antrojo pavyzdžio sąlyginį sakinį dviem — priskyrimo ir suprastintu sąlyginiu — sakiniais:

```
k:=a*a;
if a mod 2=0
   then k:=k*a
```

Sakinys $k:=k*a$ atliekamas tik tada, kai skaičius a yra lyginis. Tuomet kintamojo k reikšmė (o ji buvo lygi skaičiaus a kvadratui) dar kartą padauginama iš a, ir gaunamas skaičiaus a kubas.

Matome, kad tam pačiam uždaviniui galima sudaryti įvairias programas.

KARTOJIMO UŽDAVINIAI

18. Kokios bus kintamųjų x ir y reikšmės, atlikus šias sakinių sekas?

```
a) x:=10; y:=5;
   if x<y
      then y:=y-x
      else x:=x-y;
   if x≤y
      then x:=x-5;
   if x≥y
      then y:=y+5
b) x:=9; y:=8;
   if x>y
      then x:=y-3;
   if x<y
      then y:=y-3
      else x:=x+1
```

19. Parašykite sąlyginį sakinį kintamojo z reikšmei apskaičiuoti pagal formulę:

$$z=\begin{cases}(x+y)^2, & \text{jei } x \text{ — lyginis natūrinis skaičius,}\\ (x-y)^2, & \text{jei } x \text{ — nelyginis natūrinis skaičius.}\end{cases}$$

20. Duotas sąlyginis sakinys:

```
if x>y
   then x:=x-y
   else y:=y-x
```

Kokios galimos pradinės x ir y reikšmės, jei, atlikus šį sakinį, buvo gauta $x=5$ ir $y=5$?

21. Parašykite sakinį kintamųjų *pirmas* ir *antras* reikšmėms palyginti. Jeigu jos nelygios, reikia:

a) mažesniąją pakelti kvadratu;
b) iš didesniosios atimti mažesniąją;
c) mažesniąją padidinti vienetu;
d) didesniąją sumažinti vienetu.

22. Kintamojo *ktž* reikšmė — keturženklis natūrinis skaičius. Parašykite sakinių seką kintamojo *ktž* skaitmenų, lygių nuliui, skaičiui rasti.

Ketvirtoji pamoka

SĄLYGINIS SAKINYS (tęsinys)

Praeitoje pamokoje sužinojome, kaip parinkti vieną sprendimo variantą iš dviejų galimų. Tačiau dažnai tenka rinktis vieną iš kelių variantų. Tokį parinkimą galima išreikšti keliais sąlyginiais sakiniais, įterptais vienas į kitą.

1 p a v y z d y s. Parašykime sakinius y reikšmei, priklausančiai nuo kintamojo x reikšmės, apskaičiuoti pagal tokią formulę:

$$y=\begin{cases} 0, \text{ jei } x<0, \\ x, \text{ jei } 0\leqslant x<5, \\ 2x, \text{ jei } x\geqslant 5. \end{cases}$$

Rezultato reikšmę galima gauti atliekant vieną iš priskyrimo sakinių: $y:=0$, $y:=x$ arba $y:=2*x$. Vadinasi, yra trys variantai, priklausantys nuo kintamojo x reikšmės. Pirmiausia sujunkime juos į du variantus: $x<0$ ir $x\geqslant 0$. Tuomet sąlyginį sakinį pradėkime šitaip:

```
if x<0
   then y:=0
   else ...
```

Po to suskaidykime antrąjį variantą į du, t. y. vietoj daugtaškio įrašykime naują sąlyginį sakinį:

```
if x<0
   then y:=0
   else if x<5
           then y:=x
           else y:=2*x
```


Štai ir išspręstas uždavinys.

Po žodžio **else** einantis sąlyginis sakinys atliekamas tik tada, kai $x\geqslant 0$, todėl jame esanti sąlyga, palyginti su atitinkama formulėje, suprastinta — pakanka patikrinti, ar $x<5$.

Nagrinėtą pavyzdį sudaro vienas ilgas sąlyginis sakinys, į kurį po žodžio **else** įterptas kitas sąlyginis sakinys. Šį sakinį galima įterpti ne tik po žodžio **else**, bet ir po **then**.

2 p a v y z d y s. Kintamųjų a, b ir c reikšmės — sveikieji skaičiai. Parašykime sakinį didžiausiai duotų kintamųjų reikšmei rasti.

```
if a>b
   then if a>c
           then max:=a
           else max:=c
   else if b>c
           then max:=b
           else max:=c
```

Kai $a>b$, tai didesnio skaičiaus ieškome iš a ir c, priešingu atveju — iš b ir c.

Tuos pačius veiksmus galima užrašyti ir keliais paprastesniais sakiniais:

```
if a>b
   then max:=a
   else max:=b;
if c>max
   then max:=c
```

arba

```
max:=a;
if b>max
   then max:=b;
if c>max
   then max:=c
```

3 p a v y z d y s. Keliamieji metai turi 366 dienas, paprastieji — 365. Keliamaisiais vadinami tie metai, kurių skaičius dalus iš 4. Šimtmečių metai keliamaisiais laikomi tuomet, kai jų šimtų skaičius dalus iš 4 (pavyzdžiui, 1600 metai yra keliamieji, o 1700 — paprastieji). Parašykime sakinį metų m dienų skaičiui rasti.

```
if m mod 100=0
   then if m mod 400=0
           then m:=366
           else m:=365
   else if m mod 4=0
           then m:=366
           else m:=365
```

Sąlyginiai sakiniai gali sudaryti sudėtingos struktūros programą. Kad ją būtų lengviau skaityti, tekstą reikia išdėstyti kaip galima vaizdžiau. Žodį **then** įprasta rašyti kitoje eilutėje, šiek tiek patrauktą į dešinę, o žodį **else** lygiuoti vertikaliai su jį atitinkančiu žodžiu **then**.

Norint teisingai perskaityti sudėtingesnį sąlyginį sakinį, reikia jame rasti žodžių **if**, **then** ir **else** trejetus (arba žodžių **if** ir **then** poras, jei sakinys suprastintas), priklausančius tam pačiam sąlyginiam sakiniui. Kai sąlyginiame sakinyje yra suprastintų sąlyginių sakinių, gali būti neaišku, kuris sakinys sutrumpintas. Norint teisingai suprasti tokį sudėtingos struktūros sąlyginį sakinį, reikia įsivaizduoti, kad visi trūkstami **else** parašyti sakinio pabaigoje.

KARTOJIMO UŽDAVINIAI

23*. Kokia bus kintamojo x reikšmė, atlikus šias sakinių sekas?

a)
```
x:=5;
if x≥5
   then x:=x*2
   else if x≤10
           then x:=-x;
x:=x*5
```

b)
```
x:=5;
if x≥5
   then x:=x*2
   else if x≤10
           then x:=-x
           else x:=x*5
```

24*. Kokios bus kintamųjų a ir b reikšmės, atlikus šią sakinių seką?

```
a:=1; b:=2;
if a<b
   then a:=a+1
   else if a=b
           then a:=a+2
           else if a>b
                   then b:=b+2
```

25. Parašykite sąlyginį sakinį kintamojo f reikšmei apskaičiuoti pagal šias formules (x — sveikasis skaičius):

a)*

$$y=\begin{cases} -x^2, \text{ jei } x<0, \\ 0, \text{ jei } x=0, \\ x, \text{ jei } x>0 \text{ ir lyginis}, \\ x+1, \text{ jei } x>0 \text{ ir nelyginis}. \end{cases}$$

b)

$$y=\begin{cases} x^2, \text{ jei } x<-5, \\ x, \text{ jei } -5\leqslant x\leqslant 5, \\ 5, \text{ jei } x>5 \text{ ir baigiasi skaitmeniu 5}, \\ x^3, \text{ jei } x>5 \text{ ir nesibaigia skaitmeniu 5}. \end{cases}$$

c)

$$y=\begin{cases} a^3+b^3, \text{ jei } a \text{ ir } b \text{ — lyginiai}, \\ a^3+b^2, \text{ jei } a \text{ — lyginis, o } b \text{ — nelyginis}, \\ a^2+b^3, \text{ jei } a \text{ — nelyginis, o } b \text{ — lyginis}, \\ a^2+b^2, \text{ jei } a \text{ ir } b \text{ — nelyginiai}. \end{cases}$$

26.* Pirmosios olimpinės žaidynės įvyko 1896 metais ir organizuojamos kas ketveri metai. Jei žaidynės neįvyksta, tie metai vis tiek laikomi olimpiniais, o žaidynėms skiriamas eilės numeris. Parašykite sakinių seką m-ųjų metų olimpinių žaidynių numeriui rasti (jeigu m-ieji metai neolimpiniai, žaidynių numerį laikykite lygiu nuliui).

27. Kintamojo tr reikšmė — triženklis natūrinis skaičius. Parašykite sakinių seką kintamojo tr nelyginių skaitmenų skaičiui rasti.

28. Kintamojo n reikšmė — triženklis natūrinis skaičius. Parašykite sakinių seką kintamojo n skaitmenų, iš kurių jis dalijasi be liekanos, skaičiui rasti.

29. Kintamojo nnn reikšmė — triženklis natūrinis skaičius. Parašykite sakinių seką, pagal kurią šį skaičių galima būtų padalyti (jei dalosi be liekanos) iš jo skaitmenų sandaugos. Rezultatą priskirkite kintamajam *dalmuo*.

SUDĖTINIS SAKINYS

Penktoji pamoka

Pagal Paskalio kalbos taisykles po žodžių **then** arba **else** galima rašyti tik vieną sakinį. Jei reikia rašyti keletą sakinių, jie jungiami į vieną sudėtinį sakinį. *Sudėtinio sakinio pradžioje rašomas žodis* **begin** (pradžia), *o pabaigoje* — **end** (pabaiga).

Išnagrinėkime pavyzdį:

```
if a<0
   then begin
           b:=b+1;
           c:=c+1
        end
```

Jeigu $a<0$, tai vienetu padidinama ir kintamojo b, ir kintamojo c reikšmė.

Žodžiai **begin** ir **end** primena lenktinius skliaustus aritmetiniuose reiškiniuose. Taigi bet kurių sakinių seką galime paversti vienu sudėtiniu sakiniu, ją suskliausdami sakinių skliaustais **begin** ir

end. Kad būtų geriau matomos sudėtinio sakinio ribos, žodžiai **begin** ir **end** lygiuojami vertikaliai. Sudėtinį sakinį gali sudaryti daug sąlyginių bei sudėtinių sakinių.

1 p a v y z d y s. Kintamųjų *h*1 ir *min*1 reikšmės — laikas, išreikštas valandomis ir minutėmis. Parašykime sakinių seką laikui po vienos minutės rasti.

```
min2 := min1 + 1;
if min2 = 60
   then begin
           min2 := 0;
           h2 := h1 + 1;
           if h2 = 24
              then h2 := 0
        end
   else h2 := h1
```

Šia sakinių seka paprastai, be jokių gudrybių, išreikštas uždavinio sprendimas. Galima sudaryti ir kitokių programų. Pateiksime dar vieną, painesnį, sprendimo variantą:

```
min2 := min1 + 1;
h2 := (h1 + min2 div 60) mod 24;
min2 := min2 mod 60
```

2 p a v y z d y s. Kintamųjų *a*, *b* ir *c* reikšmės — sveikieji skaičiai. Parašykime sakinių seką didžiausiai ir mažiausiai šių kintamųjų reikšmei rasti.

```
if a > b
   then begin
           max := a;
           min := b
        end
   else begin
           max := b;
           min := a
        end;
if c > max
   then max := c;
if c < min
   then min := c
```

Pirmuoju, ilgesniu, sąlyginiu sakiniu randama didžiausia ir mažiausia iš dviejų kintamųjų *a* ir *b* reikšmių. Antruoju ir trečiuoju sąlyginiais sakiniais patikrinama, ar kintamojo *c* reikšmė nėra didesnė arba mažesnė už jau rastą maksimumą bei minimumą. Jei *c* reikšmė didesnė už rastą maksimumą arba mažesnė už rastą minimumą, tai atliekami atitinkami pakeitimai.

30. Kokia bus kintamojo *rez* reikšmė, atlikus šias sakinių sekas?

a)
```
sk := -7;
if sk < 0
   then begin
           sk := -sk;
           rez := sk div 2
        end
```

b)
```
sk := 7;
if sk < 0
   then rez := -sk
   else begin
           if sk < 10
              then rez := sk * 10;
           if sk < 100
              then rez := sk * 100
        end
```

c)
```
sk := 5;
if sk mod 3 = 0
   then rez := sk * 5
   else if sk > 0
           then begin
                   sk := sk * 5;
                   rez := sk * 3
                end
           else rez := sk * 2
```

31. Raskite visas kintamojo *a* reikšmes, kurias jis gali įgyti atlikus šį sakinį:

```
if a < -5
   then a := 0
   else if a < 0
           then a := -a
           else if a < 5
                   then a := a + 1
                   else a := 5
```

32. Kintamojo *tr* reikšmė — triženklis natūrinis skaičius. Parašykite sakinių seką, pagal kurią šis skaičius būtų dalijamas sveikųjų skaičių tikslumu iš jį sudarančių skaitmenų sandaugos (jei nėra lygių nuliui skaitmenų) arba skaitmenų sumos (jei yra lygių nuliui skaitmenų). Dalyti sveikųjų skaičių tikslumu — t. y. rasti dalmenį ir liekaną.

33. Elektroninis laikrodis rodo *h* valandų, *min* minučių ir *s* sekundžių. Parašykite sakinių seką, pagal kurią būtų galima rasti

prieš vieną sekundę laikrodžio rodytą laiką. Pavyzdžiui, jei $h=0$, $min=0$, $s=0$, tai laikrodis rodė 23 valandas 59 minutes ir 59 sekundes.

34.* Mokinys pradėjo spręsti JPM kontrolinę užduotį, kai elektroninis laikrodis rodė $h1$ valandų ir $min1$ minučių, o baigė, kai buvo $h2$ valandų ir $min2$ minučių. Parašykite sakinių seką, pagal kurią būtų randama, kiek laiko (valandų ir minučių) mokinys sprendė uždavinius. (Laikykite, kad mokinys uždavinius sprendė mažiau kaip parą.)

Šeštoji pamoka

CIKLAS while

Iki šiol nagrinėtiems uždaviniams spręsti kompiuteris buvo ne itin reikalingas. Dabar susipažinsime su tokiais uždaviniais, kuriuos be skaičiavimo mašinos išspręsti būtų gana sunku. Mat juose vieną ar kelis sakinius reikia pakartoti daugelį kartų. Šias kartojamas programos dalis nurodome *ciklo sakiniais* (dažnai sakome trumpiau — *ciklais*). Kaip atliekami tokie sakiniai, suprasime išnagrinėję keletą paprastų pavyzdžių.

1 p a v y z d y s. Parašykime sakinių seką natūrinio skaičiaus n pirmajam skaitmeniui rasti.

Kadangi nežinome, kiek skaitmenų turi duotasis skaičius, tai, norėdami rasti pirmąjį jo skaitmenį, turime šį skaičių dalyti iš 10 tol, kol gausime vienaženklį skaičių (jis ir bus rezultatas). Šį kartojimą išreikšime ciklo sakiniu:

```
p:=n;
while p>9 do
    p:=p div 10
```

Ciklo sakinį sudaro antraštė, prasidedanti žodžiu **while**, ir po jos (žodžio **do**) esantis sakinys, kuris kartojamas tol, kol tenkinama antraštėje nurodyta sąlyga. Antraštės sutartiniai žodžiai verčiami šitaip:

while — kol,
do — atlikti.

Kaip atliekamas ciklas? *Pirmiausia tikrinama sąlyga. Jeigu ji netenkinama, tai sakinys, einantis po žodžio* **do**, *neatliekamas nė karto* (pateiktame pavyzdyje taip būtų, kai n — vienaženklis skaičius). *Jeigu sąlyga tenkinama, tai atliekamas po antraštės einantis sakinys ir vėl grįžtama prie sąlygos tikrinimo. Veiksmai kartojami tol, kol tenkinama sąlyga. Kai sąlyga netenkinama, ciklas baigiamas ir vykdomas po ciklo einantis sakinys.*

Panagrinėsime, kaip atliekamas pavyzdyje užrašytas ciklas, kai duota konkreti kintamojo n reikšmė, pavyzdžiui $n=325$. Iš

pradžių sąlyga tenkinama, nes $325>9$. Todėl atliekamas priskyrimo sakinys $p:=p$ **div** 10. Dabar $p=32$. Antrą kartą tikrinama sąlyga vėl tenkinama, nes $32>9$. Dar kartą atliekamas sakinys $p:=p$ **div** 10. Dabar $p=3$. Sąlyga nebetenkinama, nes $3<9$. Todėl priskyrimo sakinys neatliekamas ir ciklas baigiamas. Taigi kintamojo p reikšmė ir yra rezultatas — pirmasis duoto skaičiaus skaitmuo.

Ciklo antraštė valdo tik vieno, po jos einančio, sakinio kartojimą. Jeigu reikia kartoti kelis sakinius, tai jie jungiami į sudėtinį sakinį (suskliaudžiami žodžiais **begin** ir **end**).

2 p a v y z d y s. Parašykime sakinių seką skaičių nuo 1 iki 100 kvadratų sumai s apskaičiuoti.

```
s:=0; i:=1;
while i≤100 do
  begin
    s:=s+i*i;
    i:=i+1
  end
```

Ciklo suma didėja pridedant prie jos kintamojo i reikšmės kvadratus. Kokios yra šio kintamojo reikšmės? Sumuojant pirmą kartą, i reikšmė tebėra tokia, kokia ji buvo prieš ciklą, t. y. 1. Ciklą kartojant, i reikšmė padidinama vienetu. Taigi sumuojant antrą kartą, i reikšmė bus 2, trečią — 3 ir t. t. Atlikus ciklą šimtąjį kartą, kintamojo i reikšmė bus 101, sąlyga netenkinama ir ciklas baigiamas.

Atkreipkime dėmesį į vieną svarbią detalę. Kaskart atliekant ciklą, kintamojo i reikšmės kvadratai pridedami prie buvusios kintamojo s reikšmės. Vadinasi, kintamasis s dar prieš ciklą turi įgyti kokią nors reikšmę, nes kitaip, atlikdami ciklą pirmą kartą, negalėtume pradėti sumuoti — kintamasis s neturėtų jokios reikšmės (tokiais atvejais sakoma, kad *kintamojo reikšmė neapibrėžta* arba *neapibrėžtas kintamasis*). Todėl pateiktame pavyzdyje prieš ciklo sakinį kintamajam s (būsimai sumai) priskiriamas nulis.

Sudarydami ciklus, galime suklysti ir užrašyti ne tuos veiksmus, kuriuos turėjome galvoje. Todėl svarbu programą gerai patikrinti, rasti ir ištaisyti joje esančias klaidas. Viena dažniau pasitaikančių klaidų yra niekad nesibaigiantis ciklas (vadinamas amžinuoju).

3 p a v y z d y s. Programuotojas, sudarydamas skaičių kvadratų sumavimo programos fragmentą, pamiršo kintamojo i reikšmę padidinti vienetu:

```
s:=0; i:=1;
while i≤100 do
  s:=s+i*i
```

Kintamojo i reikšmė cikle nekeičiama. Vadinasi, visą laiką ji lygi vienetui. Todėl ciklo antraštėje esanti sąlyga visada bus tenkinama, ir kompiuteris negalės užbaigti ciklo.

Pasitaiko ir kitokių klaidų. Daugelį jų galime rasti patys tikrindami programą. Tam parenkame paprastesnes pradines kintamųjų reikšmes ir atliekame programoje užrašytus veiksmus taip, kaip juos atliktų kompiuteris. Popieriaus lape parašome programoje vartojamus kintamuosius ir jų pradines reikšmes. Kintamajam priskyrę naują reikšmę, ankstesnę jo reikšmę užbraukiame, o jos vietoje rašome naują. Jeigu taip apskaičiuoti rezultatai nesutampa su reikiamais, tai programoje yra klaida.

4 p a v y z d y s. Natūrinio skaičiaus n skaitmenų, nelygių nuliui, sandaugai s rasti parašytas šitoks programos fragmentas:

```
p:=n; s:=0;
while p>0 do
  begin
    if p mod 10≠0
      then s:=s*(p mod 10);
    p:=p div 10
  end
```

Kai $n=23$, rezultatas turi būti 6. Patikrinsime, koks jis pagal pateiktą programą. Prieš ciklą kintamųjų reikšmės yra $p=23$, $s=0$. Taigi ciklo antraštėje esanti sąlyga tenkinama. Atlikę ciklą, gauname $p=2$, $s=0$. Dar kartą atlikę ciklą, gauname $p=0$, $s=0$. Dabar sąlyga netenkinama, todėl ciklo nebeatliekame. Gautoji sandaugos reikšmė s nelygi 6. Vadinasi, yra klaida. Išnagrinėję programos fragmentą, galime pastebėti, kad vietoj sakinio $s:=0$ turi būti sakinys $s:=1$.

Tikrinimui parinktos pradinės kintamųjų reikšmės vadinamos *kontroliniais duomenimis*. Jie turi būti tinkami, būdingi duotai programai. Pavyzdžiui, kai programoje yra sąlyginių sakinių, kontrolinius duomenis reikia parinkti taip, kad bent po kartą būtų atliekama kiekvieno sąlyginio sakinio dalis (šaka). Tikrinant programas, kuriose yra ciklų, kontrolinius duomenis pravartu imti tokius, kad ciklo nereikėtų atlikti nė karto, kad jį reikėtų atlikti vieną kartą, keletą kartų.

KARTOJIMO UŽDAVINIAI

35. Kokia bus kintamojo x reikšmė, atlikus šias sakinių sekas?

a)
```
k:=1;
while k<5 do
  k:=k+1;
x:=k
```

b)
```
x:=1;
while x≤5 do
  x:=x+1
```


36. Kokios bus kintamųjų x ir y reikšmės, atlikus šias sakinių sekas?

a)
```
x:=0; y:=0;
while x≤2 do
  x:=x+1;
y:=y+1
```

b)
```
x:=0; y:=0;
while x≤2 do
  begin
    x:=x+1;
    y:=y+1
  end
```

37. Kiek kartų bus atliekamas šis ciklas?

```
while x≤k do
  z:=k+x
```

38.* Duota sakinių seka:

```
a:=n; kiek:=0;
while a≥0 do
  begin
    kiek:=kiek+1
    a:=a div 10
  end
```

Atlikus ją, nustatoma, kiek skaitmenų turi natūrinis skaičius n. Tačiau sekoje yra klaidų. Ištaisykite jas.

39.* Kintamojo n reikšmė — natūrinis skaičius. Reikia rasti skaičių a, kurio skaitmenys išdėstyti atvirkščiai negu duotojo skaičiaus n (pavyzdžiui, kai $n=351$, tai $a=153$). Tam buvo sudarytas šitoks programos fragmentas:

```
p=n;
while p≥0 do
  begin
    a:=a+p mod 10
    p:=p div 10
  end
```

Tačiau jame yra klaidų. Ištaisykite jas.

40. Duota sakinių seka:

```
rez:=a; i:=1;
while i≤n do
  begin
    rez:=rez*a;
    i:=i+1
  end
```

Atlikus ją, duotasis sveikasis skaičius a turėtų būti pakeltas n-uoju laipsniu (n — neneigiamas sveikasis skaičius). Tačiau yra klaidų. Ištaisykite jas.

41. Duota sakinių seka (kintamojo s reikšmė — natūrinis skaičius):

```
max := 0; min := 0;
while s ≥ 0 do
   begin
      s := s mod 10;
      if s > max
         then max := s
         else if s < min
                then min := s;
      s := s div 10
   end
```

Pagal šią seką turėtume rasti mažiausią ir didžiausią skaičiaus s skaitmenį. Tačiau yra klaidų. Ištaisykite jas.

42. Parašykite sakinių seką, pagal kurią būtų galima rasti, kiek skaitmenų, lygių nuliui, turi natūrinis skaičius n.

43. Turime dvi lėkšteles. Pirmoje yra riešutai, o antra — tuščia. Riešutus perkeliame iš pirmos lėkštelės į antrąją pagal šitokį algoritmą:

1) jei riešutų skaičius pirmoje lėkštelėje lyginis, perkeliame pusę jų,
2) jei riešutų skaičius pirmoje lėkštelėje nelyginis, perkeliame vieną riešutą,
3) pirmąjį ir antrąjį veiksmą atliekame tol, kol visi riešutai bus perdėti iš pirmos lėkštelės į antrąją.

Parašykite sakinių seką perkėlimų skaičiui rasti, jei iš pradžių buvo n riešutų.

44. Duotas natūrinis skaičius n. Parašykite sakinių seką, pagal kurią galima sužinoti jo skaitmenų skaičių, iš kurių šis skaičius dalijasi be liekanos.

45. Parašykite sakinių seką, pagal kurią būtų suskaičiuojama, kiek duotasis natūrinis skaičius turi lyginių ir nelyginių skaitmenų atskirai.

Septintoji pamoka

CIKLAS for

Dažnai programose pasitaiko ciklų, turinčių kintamąjį, kurio reikšmė kiekvieną kartą atliekant ciklą padidinama vienetu, ir ciklas kartojamas tol, kol kintamasis įgyja tam tikrą reikšmę. Todėl vartojama dar viena ciklo rūšis — ciklas **for**.

1 p a v y z d y s. Parašykime sakinius skaičių nuo 1 iki 100 kvadratų sumai rasti, vartodami ciklą **for**:

```
s := 0;
for i := 1 to 100 do
   s := s + i * i
```


Ciklo antraštė prasideda žodžiu **for** (dėl), po kurio rašomas ciklo kintamojo vardas ir nurodoma, su kokiomis šio kintamojo reikšmėmis turi būti atliekamas ciklas. *Pirmą kartą atliekant ciklą, jo kintamajam priskiriama antraštėje nurodyta reikšmė* (šiame pavyzdyje $i := 1$), *ir atliekamas sakinys, esantis po žodžio* **do** (atlikti). *Tada ciklo kintamojo reikšmė padidinama vienetu, vėl atliekamas sakinys, esantis po žodžio* **do**, *ir t. t. Kai ciklo kintamasis įgyja reikšmę, nurodytą po žodžio* **to** (iki), *ciklas atliekamas paskutinį kartą.* Vadinasi, šis ciklas bus atliekamas 100 kartų (ciklo kintamojo reikšmės bus $i = 1, 2, 3, \ldots, 99, 100$).

Ciklo kintamojo reikšmės padidinamos vienetu automatiškai, todėl keisti jų ciklo viduje negalima.

Pirmoji ir paskutinioji ciklo kintamojo reikšmė gali būti ne tik sveikasis skaičius, bet ir reiškinys ar kintamasis.

2 p a v y z d y s.

```
s := 0;
for i := m to n do
   s := s + i * i
```

Čia skaičiuojama sveikųjų skaičių nuo m iki n kvadratų suma. Priskiriant kintamiesiems m ir n įvairias reikšmes (sveikuosius skaičius), apskaičiuojama vis kito intervalo sveikųjų skaičių kvadratų suma. Jeigu $m = 1$ ir $n = 100$, rezultatas gaunamas toks pat, kaip ir ankstesniame pavyzdyje. Taigi šis programos fragmentas universalesnis.

Ciklas atliekamas $n - m + 1$ kartų (pavyzdžiui, kai $m = -5$, $n = 0$, šešis kartus).

Jei pirmoji ciklo kintamojo reikšmė didesnė už paskutiniąją, ciklas neatliekamas nė karto.

3 p a v y z d y s. Parašykime sakinius natūrinio skaičiaus n faktorialui rasti. Skaičiaus n faktorialas (žymimas $n!$) lygus skaičių nuo 1 iki n sandaugai, pavyzdžiui $3! = 1 \cdot 2 \cdot 3$.

```
f := 1;
for k := 1 to n do
   f := f * k
```

Ciklas atliekamas n kartų, t. y. tiek kartų, kokio skaičiaus faktorialas skaičiuojamas. Kiekvieną kartą atliekant ciklą, rezultatui f priskiriama vis nauja reikšmė, lygi ankstesnei f reikšmei, padaugintai iš kintamojo k reikšmės.

Ciklui priklauso tiktai vienas sakinys, esantis po antraštės. Kai reikia kartoti kelis sakinius, jie jungiami į vieną sudėtinį sakinį.

4 p a v y z d y s. Parašykime sakinius sumai

$$1 \cdot 1! + 2 \cdot 2! + \ldots + n \cdot n!$$

apskaičiuoti.

```
s:=0; f:=1;
for i:=1 to n do
    begin
        f:=f*i;
        s:=s+i*f
    end
```

Į ciklą gali įeiti kitas ciklo ar sąlyginis sakinys, o ciklo sakinys įsiterpti į sudėtinį ar sąlyginį sakinį. Tai neribojama.

5 p a v y z d y s. Parašykime sakinius didžiausiam natūrinio skaičiaus n dalikliui, nelygiam skaičiui n, rasti:

```
for i:=1 to n-1 do
    if n mod i=0
        then daliklis:=i
```

Šį programos fragmentą galima patobulinti sumažinant ciklo kartojimų skaičių: pakanka tikrinti intervalą nuo 1 iki n **div** 2 (įsitikinkite). Be to, lyginių skaičių daliklį galima rasti iš karto, neatliekant ciklo (tada pradžioje reikėtų tokio sąlyginio sakinio: jei skaičius lyginis, daliklis turi būti n **div** 2, jei nelyginis — ieškomas atliekant ciklą).

Kada kokį ciklą vartoti? Jeigu kartojimų skaičius žinomas iš anksto, tai vartotinas ciklas **for**, nes jis užrašomas trumpiau ir vaizdžiau. Jeigu kartojimų skaičiaus prieš atliekant ciklą negalima nustatyti, reikia vartoti ciklą **while**.

KARTOJIMO UŽDAVINIAI

46. Kokia bus kintamojo x reikšmė, atlikus šias sakinių sekas?

a)
```
x:=0;
for k:=1 to 10 do
    x:=x+1
```

b)
```
for k:=1 to 5 do
    x:=k
```

c)
```
x:=0;
for k:=1 to 10 do
    x:=x+k
```

47. Kiek kartų atliekami šie ciklai?

a) **for** $k:=70$ **to** 91 **do** ...
b) **for** $k:=1$ **to** 1 **do** ...
c) **for** $k:=-1$ **to** 0 **do** ...
d) **for** $k:=0$ **to** -1 **do** ...
e)
```
a:=3; b:=7;
for k:=a to b do
    b:=b-2
```

48. Duota sakinių seka (kintamojo x reikšmė — natūrinis skaičius iš intervalo [1; 9]):


```
sum:=0; sk:=1;
for i:=1 to x do
    begin
        sk:=sk+10*x;
        sum:=sum+sk
    end
```

Kam lygi kintamojo *sum* reikšmė, jei prieš atliekant seką kintamojo x reikšmė buvo 3? Paaiškinkite, ką skaičiuojame šia seka.

49.* Parašykite sakinių seką sumai

$$1+11+111+\ldots+111\ldots1$$

apskaičiuoti, kai paskutinis dėmuo turi n skaitmenų.

50.* Parašykite sakinių seką sumai

$$1\cdot3+3\cdot5+\ldots+n(n+2)$$

apskaičiuoti (n — bet kuris nelyginis natūrinis skaičius).

51. Parašykite sakinių seką sandaugai

$$n(n+1)\ldots(2n-1)2n$$

apskaičiuoti (n — bet kuris natūrinis skaičius).

52. Troleibuso bilietai numeruojami nuo 000000 iki 999999. Laimingu laikomas tas bilietas, kurio numerio pirmųjų trijų skaitmenų suma yra lygi paskutiniųjų trijų skaitmenų sumai. Parašykite sakinių seką, pagal kurią galima rasti laimingų bilietų skaičių.

53. Parašykite sakinių seką, pagal kurią galima sužinoti, kiek yra laimingų bilietų (žr. **52** uždavinį), kurių numerių visi skaitmenys lyginiai.

54. Parašykite sakinių seką, pagal kurią galima sužinoti, kiek yra laimingų bilietų (žr. **52** uždavinį), kurių numeriai dalijasi be liekanos iš visų savo skaitmenų.

Aštuntoji pamoka

CIKLAS CIKLE

Kaip jau minėjome, į ciklą gali įeiti kitas ciklas, o tame vidiniame cikle vėl gali būti naujų ciklų ir t. t.

1 p a v y z d y s. Parašykime sakinius sumai

$$1^k+2^k+\ldots+n^k$$

apskaičiuoti.

```
suma:=0;
for i:=1 to n do
    begin
        p:=1;
```

```
    for j:=1 to k do
        p:=p*i;
    suma:=suma+p
end
```

Išorinis ciklas (jo antraštė **for** $i:=1$ **to** n **do**) atliekamas n kartų. Šio ciklo sakinys sudėtinis. Jame yra kitas, vidinis ciklas i-tajam nariui pakelti k-tuoju laipsniu. Kaip atliekami šie ciklai? Iš pradžių išorinio ciklo kintamajam i priskiriamas vienetas ir su šia reikšme atliekamas vidinis ciklas, kuris kartojamas k kartų (kintamojo j reikšmė keičiama nuo 1 iki k). Tuomet vėl grįžtama prie išorinio ciklo, kintamojo i reikšmė padidinama vienetu (t. y. i tampa lygi 2), ir vėl k kartų atliekamas vidinis ciklas. Tada vėl grįžtama prie išorinio ciklo ir t. t. Tai kartojama tol, kol išorinio ciklo kintamojo i reikšmė pasidarys lygi n. Pavyzdžiui, kai $n=20$, $k=3$, išorinis ciklas atliekamas 20, o vidinis $20\times3=60$ kartų.

Įsitikinkite, kad, norint apskaičiuoti sumą $1^1+1^2+\ldots+n^n$, užtenka nagrinėtoje programoje pakeisti tik vidinio ciklo antraštę. Ji turėtų būti tokia: **for** $j:=1$ **to** i **do**.

2 p a v y z d y s. Parašykime sakinių seką, pagal kurią sužinotume, kiek yra dviženklių skaičių, kurie dalijasi iš juos sudarančių skaitmenų sumos.

Uždavinį galima spręsti dviem būdais: 1) surasti kiekvieno dviženklio skaičiaus skaitmenis ir patikrinti, ar jis dalijasi iš šių skaitmenų sumos; 2) imti visas galimas skaitmenų poras, iš jų sudaryti dviženklį skaičių ir patikrinti, ar šis skaičius dalijasi iš jį sudarančių skaitmenų sumos. Spręsime antruoju būdu.

```
kiek:=0;
for a:=1 to 9 do
    for b:=0 to 9 do
        begin
            dv:=a*10+b;
            s:=a+b;
            if dv mod s=0
                then kiek:=kiek+1
        end
```

Dviženklį skaičių pažymėkime vardu dv, jo skaitmenis — vardais a ir b, skaitmenų sumą — s, o rezultatą — vardu *kiek*. Išoriniu ciklu perrenkami visi skaitmenys nuo 1 iki 9. Tai pirmasis dviženklio skaičiaus skaitmuo. Vidiniu ciklu perrenkami visi skaitmenys nuo 0 iki 9 (antrasis dviženklio skaičiaus skaitmuo gali būti ir nulis). Taigi išorinis ciklas atliekamas 9 kartus, o vidinis $9\times10=90$ kartų.

Išnagrinėjome keturių rūšių sakinius: priskyrimo, sudėtinį, sąlyginį ir ciklo. Priskyrimo sakinys yra elementarus, juo kintamajam priskiriama reikšmė. Sudėtinis, sąlyginis ir ciklo sakiniai

vadinami valdymo struktūromis. Jais nurodoma, kaip valdyti kitų sakinių atlikimo tvarką. Visos nagrinėtos valdymo struktūros yra lygiateisės: jos gali įeiti viena į kitą be jokių ribojimų. Šių valdymo struktūrų pakanka, kad galėtume sudaryti bet kokio uždavinio programą.

KARTOJIMO UŽDAVINIAI

55. Kiek kartų atliekamas išorinis ir vidinis ciklas šiuose programos fragmentuose?

a)* $a:=0$;
```
    while a≤5 do
        while a≤10 do
            a:=a+1
```
b) $a:=0$;
```
    while a≤10 do
        while a≤20 do
            a:=a+1
```
c) $a:=0$;
```
    while a≤20 do
        while a≤10 do
            a:=a+1
```

56. Duota sakinių seka:

```
s:=0;
for i:=0 to a do
    for j:=7 to b do
        for k:=1 to c do
            s:=s+1
```

Kokios galimos a, b ir c reikšmės, jei, atlikus šią seką, buvo gauta $s=4$?

57. Duota sakinių seka:

```
kiek:=0;
for x:=1 to v do
    for y:=1 to x do
        if v mod (x*y)=0
            then kiek:=kiek+1
```

Pagal ją suskaičiuojami visi skirtingi stačiakampiai gretasieniai, kurių briaunų ilgiai išreiškiami natūriniais skaičiais, o kiekvieno stačiakampio gretasienio tūris lygus natūriniam skaičiui v. Tačiau ši seka ne visuomet teisinga. Nurodykite tokią kintamojo v reikšmę, kad būtų gaunamas neteisingas rezultatas. Pataisykite klaidas.

58. Duota sakinių seka (kintamojo n reikšmė — bet kuris natūrinis skaičius):

```
sum := 0; fak := 1;
for i := 1 to n do
    begin
        fak := fak * i;
        k := 1;
        for j := 1 to 3 do
            k := k * i;
        sum := sum + k * fak
    end
```

Kam bus lygi kintamojo *sum* reikšmė, jei, prieš atliekant seką, kintamojo n reikšmė buvo 3? Kokį uždavinį sprendžia kompiuteris, atlikdamas šiuos sakinius?

59. Parašykite sakinių seką sumai

$$1 \cdot 2 + 2 \cdot 3 \cdot 4 + \ldots + n(n+1) \cdot \ldots \cdot 2n$$

apskaičiuoti (n — bet kuris natūrinis skaičius).

60.* Parašykite sakinių seką, pagal kurią galima rasti, kiek yra triženklių natūrinių skaičių, kurių vidurinysis skaitmuo lygus pirmojo ir trečiojo skaitmenų sumai.

61. Parašykite sakinių seką duoto natūrinio skaičiaus skaitmenims padvigubinti. Pavyzdžiui, jei $n = 1986$, tai rezultatas turi būti 11998866.

62. Parašykite sakinių seką trikampių, kurių kraštinių ilgiai a, b ir c yra natūriniai skaičiai ir tenkina šias nelygybes:

$$min \leqslant a,\ a \leqslant b,\ b \leqslant c,\ c \leqslant max,$$

skaičiui rasti (kintamųjų *min* ir *max* reikšmės — duoti natūriniai skaičiai).

63. Skaičiai, vienodai skaitomi iš kairės į dešinę ir atgal, vadinami palindromais. Pavyzdžiui, 22, 121, 42924. Parašykite sakinių seką, pagal kurią galima būtų suskaičiuoti, kiek palindromų yra intervale $[m; n]$ (m, n — natūriniai skaičiai).

64. Parašykite sakinių seką, pagal kurią galima rasti mažiausią intervalo [1; 100] skaičių, turintį daugiausia daliklių.

Devintoji pamoka

DUOMENŲ SKAITYMAS IR RAŠYMAS

Kaip jau minėjome, kompiuteris atlieka veiksmus su duomenimis. Duomenys, žinomi iš anksto, vadinami pradiniais. Atlikę programą, gauname *rezultatus* arba *galutinius duomenis*. Pavyzdžiui, praeitoje pamokoje sumuojant eilutę

$$1^k + 2^k + \ldots + n^k,$$

pradiniais duomenimis buvo skaičiai n ir k, o rezultatu — eilutės suma.

Iki šiol, rašydami programas, nesidomėjome, kaip pradiniai duomenys patenka į kompiuterio atmintį ir kaip rezultatas paimamas iš jos. Juk žmogus negali pats įrašyti duomenų į kompiuterio atmintį (kaip skaičių klasės lentoje) arba perskaityti atmintyje gautų rezultatų. Sakoma, kad pradinius duomenis reikia *įvesti* į kompiuterio atmintį, o rezultatus *išvesti* iš jos.

Dažniausiai pradinių duomenų reikšmės įvedamos į kompiuterį paspaudus atitinkamos paskirties klaviatūros klavišus, o rezultatai parodomi displėjaus ekrane arba spausdinami popieriuje.

Programoje duomenų įvedimas ir išvedimas nurodomas skaitymo ir rašymo (spausdinimo) sakiniais.

Skaitymo sakinys prasideda žodžiu „read“ (skaityti). *Po jo skliaustuose rašomi vardai tų kintamųjų, kuriems turi būti priskirtos įvestų* (perskaitytų) *duomenų reikšmės.*

1 p a v y z d y s. Parašykime sakinių seką dviem skaičiams perskaityti ir sudėti:

```
read (a);
read (b);
s := a + b
```

Kompiuteriui duomenis reikia pateikti tokia tvarka, kokia jie surašyti skaitymo sakiniuose. Šiuo atveju pirmiausia įvedama kintamojo a, po to kintamojo b reikšmė.

Skaitymo sakinius galima sujungti į vieną stambesnį sakinį ir juo įvesti keletą duomenų. Skaitymo sakinyje kintamieji vienas nuo kito skiriami kableliu. Pavyzdžiui, vietoj ankstesnių dviejų skaitymo sakinių galime rašyti vieną:

```
read (a, b);
s := a + b
```

Rašymo (spausdinimo) *sakinys prasideda žodžiu „write“* (rašyti). *Po jo skliaustuose nurodomi tie duomenys, kuriuos reikia spausdinti.*

Rašymo sakinyje duomenys atskiriami kableliu.

2 p a v y z d y s. Kompiuteris turi perskaityti du pradinius duomenis, po to išspausdinti juos atvirkščia tvarka, t. y. pirma spausdinti antrąjį skaičių, tuomet — pirmąjį. Tai galima nurodyti tokiais sakiniais:

```
read (sk1, sk2);
write (sk2, sk1)
```

3 p a v y z d y s. Pradiniai duomenys — 100 skaičių. Reikia apskaičiuoti ir išspausdinti jų sumą.

Jeigu skaičiuotume taip pat, kaip 1 pavyzdyje, tai reikėtų vartoti skaitymo sakinį su 100 kintamųjų, o tai nepatogu. Todėl spręsime kitokiu būdu.

```
s:=0;
for k:=1 to 100 do
    begin
        read (a);
        s:=s+a
    end;
write (s)
```

Ciklas kartojamas 100 kartų — tiek, kiek yra pradinių duomenų. Jame perskaitomas pradinis duomuo ir pridedamas prie sumos s. Ciklą kartojant, naujas pradinis duomuo užrašomas į ankstesniojo vietą, nes pastarasis jau nebereikalingas. Todėl tą patį kintamąjį (ir tą pačią vietą kompiuterio atmintyje) galima panaudoti kitoms pradinių duomenų reikšmėms saugoti.

Prieš atliekant ciklą, kintamajam s priskiriamas nulis. Toks veiksmas būtinas, antraip negalėtume pradėti sumuoti — nebūtų prie ko pridėti, t. y. būtų neapibrėžta s reikšmė.

4 pavyzdys. Pradiniai duomenys — skaičių, nelygių nuliui, seka. Kiek skaičių yra sekoje, nežinome. Tačiau žinome, kad seka baigiasi nuliu, kitaip sakant, sekos pabaigos požymis — nulis. Parašykime programos fragmentą šios sekos skaičių sumai apskaičiuoti bei spausdinti:

```
s:=0;
read (a);
while a≠0 do
    begin
        s:=s+a;
        read (a)
    end;
write (s)
```

Ši programa universalesnė už 3-iojo pavyzdžio programą, nes pagal ją galima apskaičiuoti bet kurio pradinių duomenų skaičiaus sumą, išskyrus tuos atvejus, kai duomenyse yra nulių.

Rašymo sakinyje galima vartoti ne tik kintamųjų vardus, bet ir skaičius bei reiškinius. Tuomet bus spausdinami tie skaičiai arba reiškinių reikšmės. Pavyzdžiui, kompiuteris, atlikęs sakinius

```
mn:=11;
d:=15;
write (1985, mn, d+5)
```

išspausdina tris skaičius:

1985 11 20

5 pavyzdys. Parašykite sakinių seką visų intervalo $[m; n]$ (m, n — sveikieji skaičiai) sveikųjų skaičių kvadratams ir kubams rasti. Kiekvieną kartą turi būti spausdinamas skaičius, jo kvadratas ir kubas.

```
read (m, n);
for k:=m to n do
    write (k, k*k, k*k*k)
```

Pavyzdžiui, kai pradiniai duomenys yra 2 ir 5, tai kompiuteris išspausdina tokius skaičius:

2 4 8 3 9 27 4 16 64 5 25 125

KARTOJIMO UŽDAVINIAI

65. Pradiniai duomenys: a) 5, 10; b) 10, 5, c) 5, 5; d) 0, 1; e) 2, 0; f) 0, 5. Ką spausdins kompiuteris, atlikęs šią sakinių seką?

```
read (a, b);
if a<b
  then for k:=a to b do
          write (a*k)
  else for k:=b to a-1 do
          write (a+b)
```

66. Pradiniai duomenys — sveikųjų skaičių, nelygių nuliui, seka. Sekos pabaigos požymis — nulis. Parašykite sakinių seką, kurią atlikęs kompiuteris išspausdintų:

a) kiekvieno sekos nario kvadratą bei visų kvadratų sumą;

b) atskirai lyginių ir nelyginių sekos narių maksimumą ir minimumą;

c) sekos narius, lygius dvejetui, pakeltam kuriuo nors laipsniu.

67. Parašykite sakinių seką triženkliams skaičiams, kurie dalijasi iš visų savo skaitmenų, spausdinti.

68. Parašykite sakinių seką intervalo $[m, n]$ skaičiams, kurie dalijasi iš visų savo skaitmenų, spausdinti (m, n — pradiniai duomenys, natūriniai skaičiai).

69.* Atlikęs sakinių seką

```
read (a);
s:=0;
for i:=a to 5 do
    if i>0 then s:=s-1
           else s:=s+1;
write (s)
```

kompiuteris išspausdino skaičių 1. Kokia buvo pradinio duomens reikšmė?

70. Supersudėtiniu skaičiumi laikomas toks natūrinis skaičius, kuris turi daugiau daliklių negu bet kuris už jį mažesnis natūrinis skaičius. Parašykite sakinių seką visiems supersudėtiniams skaičiams iš intervalo $[2, n]$ rasti (n — pradinis duomuo). Pavyzdžiui, kai $n=12$, tai supersudėtiniai skaičiai yra 4, 6 ir 12.

Dešimtoji pamoka

REZULTATŲ SPAUSDINIMAS

Spausdinti galima ne tik skaičius, bet ir tekstą, kitokius simbolius. Jie nurodomi rašymo sakinyje tarp apostrofų. Pavyzdžiui:

```
write ('***** GIMIMO METAI 1954 *****')
```

Kompiuteris, atlikęs šį sakinį, išspausdintų tokį pat tekstą, koks užrašytas spausdinimo sakinyje, tik be apostrofų:

```
***** GIMIMO METAI 1954 *****
```

Įsidėmėkite, kad tarpas taip pat laikomas simboliu. Kompiuteris spausdindamas palieka tiek tarpų, kiek jų yra tarp apostrofų.

Derinant įvairius simbolius (žvaigždutes, brūkšnelius, taškus, raides), sudaromos programos, pagal kurias spausdinami rezultatai. Jie būna pateikti lentelėse, su antraštėmis, paaiškinti formulėmis, žodžiais ir panašiai. Programavimo mėgėjai dažnai sukuria programas, pagal kurias kompiuteriai piešia iš simbolių sudarytus įmantrius piešinius, spausdina ornamentais apipavidalintus kalendorius.

Atsiminkite, kad kai kurie kompiuteriai spausdina tekstus vien didžiosiomis raidėmis. Mes programas galime rašyti taip, kaip patogiau bei vaizdžiau. Paskalio kalba programas priimta rašyti mažosiomis raidėmis, o spausdinamą tekstą — didžiosiomis.

Kompiuteris spausdina duomenis popieriaus lape vieną po kito. Kai eilutė baigiasi, automatiškai pereinama į naują. *Norint pereiti į naują eilutę, neužpildžius iki galo ankstesnės, rašomas sakinys „writeln“* (angliškai „write line“ — rašyti eilutę). Pavyzdžiui, kompiuteris, atlikęs sakinius

```
write ('INFORMATIKA');
write ('IR SKAIČIAVIMO TECHNIKA')
```

viską išspausdins vienoje eilutėje:

INFORMATIKA IR SKAIČIAVIMO TECHNIKA

o, atlikęs sakinius

```
write ('INFORMATIKA');
writeln;
write ('IR SKAIČIAVIMO TECHNIKA')
```

— dviem eilutėmis:

INFORMATIKA
IR SKAIČIAVIMO TECHNIKA

Sakinyje *writeln* galima rašyti ir reikiamus duomenis. Tada pirmiausia išspausdinami duomenys, po to pereinama į naują eilutę. Taigi anksčiau pateiktus tris sakinius galima pakeisti tokiais dviem sakiniais:

```
writeln ('INFORMATIKA');
write ('IR SKAIČIAVIMO TECHNIKA')
```


1 p a v y z d y s. Pradiniai duomenys — skaičių, nelygių nuliui, seka. Sekos pabaigos požymis — nulis. Parašykime sakinių seką, kurią atlikęs kompiuteris kiekvieną skaičių spausdintų iš naujos eilutės, o šalia jo — žodį TAIP, jei tas skaičius dalijasi iš 11, arba žodį NE — jei nesidalija.

Pateikiame šiam uždaviniui sudarytą programos fragmentą:

```
read (s);
while s≠0 do
   begin
      write (s); write (' ');
      if s mod 11=0
         then writeln ('TAIP')
         else writeln ('NE');
      read (s)
   end
```

Sakiniu *write* (' ') nurodėme, kad po skaičiaus prieš žodį TAIP arba NE būtų paliekamas tarpas (tarp skaičių tarpai paliekami automatiškai, o tekstas spausdinamas be tarpų).

Pateiksime ir kitaip užrašytą sakinių seką tam pačiam uždaviniui spręsti:

```
read (s);
while s≠0 do
   begin
      if s mod 11=0
         then writeln (s,' TAIP')
         else writeln (s,' NE');
      read (s)
   end
```

2 p a v y z d y s. Parašykime programos fragmentą, kurį atlikęs kompiuteris išspausdintų stačiakampį, sudarytą iš $n*m$ žvaigždučių (n, m — pradiniai duomenys, natūriniai skaičiai). Pavyzdžiui, kai pradiniai duomenys yra skaičiai 3 ir 5, spausdinama taip:

```
* * * * *
* * * * *
* * * * *
```

Pateikiame sakinių seką, pagal kurią spausdinamas šis stačiakampis:

```
read (n, m);
for h:=1 to n do
   begin
      for i:=1 to m do
         write ('*');
```

```
    writeln
  end
```

Programoje du ciklai: išorinis skaičiuoja stačiakampį sudarančių žvaigždučių eilutes, vidinis — žvaigždučių skaičių eilutėse. Kiekvieną kartą, išspausdinus *m* žvaigždučių, peršokama į naują eilutę.

3 p a v y z d y s. Pradiniai duomenys — natūrinių skaičių seka. Sekos pabaigos požymis — nulis. Sudarykime programos fragmentą, kurį atlikęs kompiuteris sekos skaičius spausdintų po tris eilutėje (trimis stulpeliais). Pavyzdžiui, jei duota seka

7 8 4 15 20 3 8 7 9 25 50 0,

tai turi būti spausdinama

```
 7   8   4
15  20   3
 8   7   9
25  50
```

Kaip nurodyti programoje skaičių išdėstymą po tris eilutėje? Paprasčiausiai galime numeruoti ir spausdinti sekos narius. Kai išspausdinamas sekos narys, kurio numeris dalijasi iš 3, reikia peršokti į naują eilutę. Pateikiame tokiu būdu sudarytą programą:

```
nr := 0;
read (x);
while x≠0 do
  begin
    nr := nr+1;
    write (x);
    if nr mod 3=0 then
      writeln;
    read (x)
  end
```

Sekos elemento numerį nusako kintamasis *nr*. Išspausdinus narį, kurio numeris dalijasi iš 3, peršokama į naują eilutę.

KARTOJIMO UŽDAVINIAI

71.* Duota sakinių seka:

```
read (n);
for i := 1 to n do
  begin
    for j := 1 to n do
      if i≥j then write ('A')
             else write ('B');
    writeln
  end
```


Ką spausdins kompiuteris, atlikęs šią seką, jei pradinis duomuo: a) 4; b) 7?

72.* Duota sakinių seka:

```
read (n, m);
for a := 1 to n do
  begin
    write ('U');
    for b := 2 to m-1 do
      if a=n
        then write ('U')
        else write ('=');
    writeln ('U')
  end
```

Ką spausdins kompiuteris, atlikęs šią seką, jei pradiniai duomenys yra skaičiai 4 ir 5?

73. Parašykite sakinių seką visiems lygties

$$i^3+j^3+k^3=z^3$$

sveikiesiems sprendiniams iš intervalo [1; 20] rasti. Kiekvieną lygties sprendinį (keturis skaičius) reikia spausdinti iš naujos eilutės.

74. Parašykite sakinių seką visiems sveikiesiems skaičiams, kurie yra lygties

$$x^3-10x^2-27x+36=0$$

sprendiniai intervale [−10; 0], rasti.

75. Pradiniai duomenys — natūrinių skaičių seka. Sekos pabaigoje — nulis. Parašykite sakinių seką, pagal kurią galima skaityti šiuos skaičius ir nustatyti, kurie jų pirminiai. Kompiuteris turi spausdinti po vieną skaičių eilutėje ir greta jo žodį PIRMINIS (jei skaičius pirminis) arba žodį NE (jei skaičius sudėtinis).

Vienuoliktoji pamoka

PROGRAMOS STRUKTŪRA

Ankstesnėse pamokose nagrinėtas sakinių sekas kartais vadindavome programomis, nors iš tikrųjų tai dar nebuvo užbaigtos programos. Tačiau jas nesunku papildyti iki kompiuteriui suprantamų programų.

Paskalio kalba užrašytą programą sudaro šios dalys: 1) programos antraštė; 2) aprašai; 3) sudėtinis sakinys; 4) programos pabaigos simbolis — taškas.

1 p a v y z d y s. Sudarykime programą dviejų skaičių sumai apskaičiuoti ir spausdinti:

```
program sum (input, output);
```

```
    var a, b, s : integer;
begin
    read (a, b);
    s:=a+b;
    write (s)
end.
```

Pirmoji eilutė yra programos antraštė. Ji visada prasideda žodžiu **program**. Po jo rašomas programos vardas, kurį parenka programuotojas. Mes šią programą pavadinome vardu *sum*. Galėjome ją pavadinti kitaip. Tačiau paprastai vardas parenkamas pagal uždavinio prasmę.

Skliaustuose po vardo rašomas žodis *input* (įvedimas), jeigu programoje yra sakinių duomenims įvesti, ir žodis *output* (išvedimas), jeigu yra sakinių duomenims spausdinti. (Kai kuriems kompiuteriams skirtose programose skliaustus ir juose esančią informaciją galima praleisti. Taip mes toliau ir darysime.)

Antroji programos eilutė — kintamųjų aprašas. Jis prasideda žodžiu **var** (kilęs iš anglų k. žodžio „variable"— kintamasis), po kurio išvardijami kintamųjų vardai (jie vienas nuo kito skiriami kableliu). Žodžiu *integer* (sveikasis) pasakoma, kad šiais vardais pažymėti kintamieji gali įgyti tiktai sveikųjų skaičių reikšmes. Jų reikšmių absoliutinį dydį riboja ESM konstrukcija. Dauguma kompiuterių gali atlikti programą su sveikaisiais skaičiais, susidedančiais iš 10—13 skaitmenų.

Visi programoje vartojami kintamųjų vardai turi būti aprašyti.

Po aprašų eina sudėtinis sakinys. Taigi galima sakyti, kad visi programos veiksmai užrašomi vienu sudėtiniu sakiniu. Žinoma, toks sakinys labai sudėtingas ir jam užrašyti gali prireikti daug puslapių.

Atkreipkime dėmesį į skyrybą. Po antraštės ir po aprašų rašomas kabliataškis, o visos programos pabaigoje — taškas. Sakiniai vienas nuo kito skiriami kabliataškiu.

2 p a v y z d y s. Sudarykime programą visiems natūrinio skaičiaus n dalikliams rasti.

Pats paprasčiausias sprendimo būdas — tikrinti, ar skaičius n dalijasi paeiliui iš visų skaičių 1, 2, 3, ..., n. Sudarome programą:

```
program dalikliai;
    var n, d : integer;
begin
    read (n);
    for d:=1 to n do
        if n mod d=0
            then write (d)
end.
```

Pavyzdžiui, kai pradinis duomuo 30, kompiuteris turi spausdinti

1 2 3 5 6 10 15 30

Šią programą galime patobulinti — sumažinti veiksmų skaičių jojė. Lengva įsitikinti, kad intervale [n **div** 2+1; n−1] skaičiaus n daliklių nėra. Todėl neverta nurodyti, kad kompiuteris jų ten ieškotų. Pateikiame tobulesnį programos variantą:

```
program dalikliai;
    var n, d : integer;
begin
    read (n);
    for d:=1 to n div 2 do
        if n mod d=0
            then write (d);
    write (n)
end.
```

Užbaigta programa įrašoma (panašiai kaip ir pradiniai duomenys) į kompiuterio atmintį. Kompiuteris, atlikdamas programą, išspausdina jos *listingą*. Tai programos tekstas, įvairios antraštės, paaiškinimai bei skaičiavimų rezultatai.

KARTOJIMO UŽDAVINIAI

76. Duota programa:

```
program iksas;
    var plotis,
        eil, st : integer;
begin
    for eil:=1 to plotis do
        begin
            for st:=1 to plotis do
                if eil=st
                    then write ('+')
                    else if eil+st=plotis
                            then write ('+')
                            else write (' ');
            writeln
        end
end.
```

Atlikus ją, turėtų būti spausdinama x formos figūra, sudaryta iš pliusų. Tačiau programoje yra klaidų. Ištaisykite jas.

77. Duota programa:

```
program suma;
    var n,
        x, i : integer;
begin
```

```
    read (n);
    i:=1; sum:=n;
    while i*i≤n do
      begin
        if n mod i=0
          then if i+n div i<sum
                 then x:=i;
                      i:=i+1
      end;
    write (n, '=', 'x', '*', n div x)
end.
```

Atlikus ją, natūrinis skaičius n turėtų būti išreikštas dviejų natūrinių skaičių sandauga taip, kad tų skaičių suma būtų mažiausia. Tačiau programoje yra klaidų. Ištaisykite jas.

78. Trys natūriniai skaičiai x, yx ir zyx sudaro geometrinę progresiją. (Raidės x, y, z žymi skaičiaus skaitmenis.)

Duota programa:

```
program trys;
  var x, y, z : integer
begin
  for x:=1 to 9 do
    for y:=1 to 9 do
      for z:=1 to 9 do
        begin
          yx:=10*y+x;
          zyx:=100*z+yx;
          if yx div x*yx=zyx
            then if yx mod x=0
                   then write (x, yx, zyx)
        end
end.
```

Atlikus ją, šie skaičiai turėtų būti spausdinami. Tačiau programoje yra klaidų. Ištaisykite jas.

79.* Automorfiniu vadinamas toks skaičius, kurio skaitmenys sutampa su jo kvadrato paskutiniaisiais skaitmenimis. Pavyzdžiui,

$6^2=36$
$25^2=625$
$76^2=5776$

Automorfiniams intervalo $[m; n]$ skaičiams rasti sudaryta tokia programa:

```
program automorf;
  var m, n, x, d : integer;
begin
  read (m, n);
  d:=10;
  for x:=m to n do
    begin
      while ... do
        d:=d*10;
      if ... then
        writeln (x, x*x)
    end
end.
```

Daugtaškių vietoje įrašykite praleistas sąlygas.

80. „Kubiniu“ automorfiniu skaičiumi vadinamas toks skaičius, kurio skaitmenys sutampa su jo kubo paskutiniaisiais skaitmenimis. Pavyzdžiui,

$6^3=\ \ 216$
$51^3=132651$

Sudarykite programą visiems nuo 1 iki 1000 „kubiniams“ automorfiniams skaičiams rasti.

81.* Pradinis duomuo — natūrinis skaičius p, reiškiantis plotą. Parašykite programą visiems tokio ploto stačiakampiams, kurių kraštinės išreiškiamos natūriniais skaičiais, rasti. Kiekvieno stačiakampio kraštinių ilgius spausdinkite iš naujos eilutės. Pavyzdžiui, jei $p=20$, tai rezultatus reikia spausdinti taip:

```
1  20
2  10
4   5
```

82.* Sudarykite programą skaičių nuo 1 iki n (n — pradinis duomuo) dalumui grafiškai pavaizduoti. Kiekvienoje eilutėje reikia spausdinti skaičių ir tiek pliusų, kiek jis turi dalikliu. Pavyzdžiui, kai pradinis duomuo yra 4, turi būti išspausdinta:

```
1+
2++
3++
4+++
```

83. Parašykite programą kiekvienam duoto natūrinio skaičiaus skaitmeniui spausdinti iš naujos eilutės, šalia rašant tiek pliusų, kiek skaičius turi vienetų. Pavyzdžiui, jei pradinis duomuo 1702, tai kompiuteris turi išspausdinti:

```
1+
7+++++++
0
2++
```

84. Pradinis duomuo — natūrinis skaičius a. Parašykite programą natūriniam skaičiui n, tenkinančiam šias dvi nelygybes:

$n!\leqslant a$,
$(n+1)!>a$,

rasti ir išspausdinti.

Pavyzdžiui, jeigu $a=30$, tai $n=4$ tenkins abi nelygybes, nes $4!=24$ ir $(4+1)!=120$.

85.* Duota programa:

```
program xxx;
   var a, b : integer;
begin
   read (a);
   b := 0;
   while a≠0 do
      begin
         b := b * 10 + a mod 10;
         a := a div 10
      end;
   write (b)
end.
```

Ką spausdins kompiuteris, atlikęs šią programą, kai $a=13305$? Kokio uždavinio sprendimas užrašytas šia programa?

86.* Duota programa:

```
program atspėk;
   var a, b, c, i, j : integer;
begin
   read (n);
   a := 1; b := 1;
   for i := 0 to n do
      begin
         for j := 1 to b do
            write ('*');
         writeln;
         c := a + b;
         a := b; b := c
      end
end.
```

Ką išspausdins kompiuteris, atlikęs šią programą, jei pradinis duomuo yra 7? Kokio uždavinio sprendimas užrašytas šia programa?

Dvyliktoji pamoka

PROGRAMAVIMO METODIKA

Išnagrinėtų programavimo konstrukcijų pakanka daugelio uždavinių programoms sudaryti. Bet sudėtingam uždaviniui parašyti programą nėra paprasta. Programuoti lengviau, jei laikomasi tam tikrų taisyklių — programavimo metodikos.

Uždavinio programavimą galima suskirstyti į šias dalis:

1. Uždavinio formulavimas.
2. Sprendimo algoritmo parinkimas ar sudarymas.
3. Programos sudarymas.
4. Programos tikrinimas.
5. Programos tobulinimas.
6. Programos derinimas.

Trumpai aptarsime kiekvieną jų:

Uždavinio formulavimas. Prieš programuodami turime išsiaiškinti visus reikalavimus būsimai programai, t. y. sudaryti tikslią programavimo užduotį. Todėl formuojant uždavinį, reikia aiškiai nurodyti, ką turi atlikti programa, kokie turi būti pradiniai duomenys ir rezultatai. Svarbu išsiaiškinti, kaip reikės pateikti rezultatus — išdėstyti lentelėse, pavaizduoti grafiškai, išspausdinti su antraštėmis, eilutėmis ir t. t. Kaip gauti rezultatus, t. y. kokius veiksmus reikia atlikti, nustatoma vėliau, sudarant programą.

Sprendimo algoritmo parinkimas ar sudarymas. Uždavinio sprendimo algoritmą turi sugalvoti programuotojas. Juk programa ir yra tikslus uždavinio sprendimo aprašymas, kurį vienodai supranta ir žmogus, ir kompiuteris. Tiktai kompiuteris, spręsdamas uždavinį, atlieka dar skaičiavimo darbą, kuris nurodomas programoje. Taigi programuotojas visų pirma turi išsiaiškinti, kaip spręsti duotą uždavinį. Jis gali pritaikyti jau žinomus metodus, pavyzdžiui, tiesinių lygčių sistemą spręsti sudėties būdu, pirminius skaičius ieškoti Eratosteno rėčio metodu ir panašiai. Tačiau kartais programuotojui tenka pačiam sukurti uždavinio sprendimo metodą.

Programos sudarymas. Kiekvieno uždavinio programa yra savita. Be to, tam pačiam uždaviniui spręsti galima sudaryti daug skirtingų programų. Programavimas yra kūrybinis procesas, ir universalių receptų, kaip sudaryti vieno ar kito uždavinio programą, nėra. Tačiau vadovaujantis bendrais dėsniais, šį darbą būtų galima paspartinti, atlikti geriau. Iš karto sudaryti sudėtingo uždavinio programą sunku arba iš viso neįmanoma. Tokiu atveju uždavinys skaidomas į keletą smulkesnių dalių, kurių kiekviena sprendžiama (programuojama) atskirai. Jeigu toji dalis yra dar per stambi, tai ji skaidoma į smulkesnes dalis tol, kol jų programos pasidaro pakankamai trumpos ir vaizdžios.

Programos tikrinimas. Sudarant programą, labai lengva suklysti. Todėl prieš pateikiant ją kompiuteriui, reikia kruopščiai patikrinti.

Tikrinant programą, reikėtų įsitikinti: 1) ar nėra sintaksės klaidų (netaisyklingai užrašytas kuris nors sakinys, praleistas skyrybos ženklas ir panašiai; 2) ar aprašyti visų kintamųjų vardai; 3) ar apibrėžtos visų kintamųjų reikšmės; 4) ar programos veiksmai baigtiniai; 5) ar programa duos teisingus rezultatus.

Be abejo, svarbiausia gauti teisingus rezultatus. Tačiau juos patikrinti sunku. Vienas paprasčiausių tikrinimo būdų — pasirinkus kokius nors pradinius duomenis, pačiam programuotojui (be kompiuterio) atlikti programoje užrašytus veiksmus. Veiksmai atliekami taip, kaip juos atliktų kompiuteris, t. y. paraidžiui, mechaniškai, nesigilinant į jų prasmę. Tada gaunami tokie patys rezultatai, kaip ir kompiuteriu.

Tikrinimui parinkti pradiniai duomenys vadinami *kontroliniais*. Labai *svarbu parinkti tinkamus kontrolinius duomenis*. Jie turi būti būdingi tikrinamai programai — tokie, kad būtų patikrinamos visos programos dalys. Be to, kontrolinius duomenis reikia parinkti taip, kad galėtume nesunkiai apskaičiuoti rezultatą.

Programos tobulinimas. Ne iš karto pavyksta sudaryti tobulą ir ekonomišką programą. Dažnai gimsta naujų idėjų programai patobulinti — padaryti ją trumpesnę, lakoniškesnę, aiškesnę ar ekonomiškesnę (pavyzdžiui, greičiau atliekamą). Programos tobulinimo pavyzdį pateikėme praeitoje pamokoje, nagrinėdami duoto skaičiaus daliklių radimo programą.

Programos derinimas. Programuotojas, sudaręs programą ir pateikęs ją kompiuteriui, paprastai tikisi teisingų rezultatų. Tačiau net ir gerai patikrintoje programoje pasitaiko klaidų. Kompiuteris spausdina pranešimus apie visas aptiktas klaidas. Programuotojas turi išnagrinėti kiekvieną klaidą ir ištaisytą programą vėl pateikti kompiuteriui. Šis procesas kartojamas tol, kol programoje nelieka klaidų. Šitaip ištaisomos visos sintaksės klaidos.

Prasmines klaidas rasti sunkiau. Jų ieškoma pateikiant kompiuteriui pradinius duomenis ir lyginant jo spausdinamus rezultatus su iš anksto žinomais rezultatais. Tinka tie patys kontroliniai duomenys, kurie buvo vartojami tikrinant programą be kompiuterio. Tiktai dabar galime imti ir kitus, sudėtingesnius duomenis, nes skaičiuojame ne mes, o kompiuteris.

Ne visas prasmines klaidas pavyksta greitai ir lengvai surasti. Sudėtingose programose pasitaiko sunkiai aptinkamų klaidų. Kaip jų ieškoti — bendrų receptų nėra. Dažnai praverčia įžvalgumas, logika, programavimo išmanymas. Kai programos vaizdžiai, suprantamai parašytos, jas lengviau skaityti, o kartu ir surasti klaidas.

Taigi programos paruošimas yra ilgas ir kruopštus darbas. Tačiau kai programa sudaryta ir suderinta, ją galima atlikti daug kartų, įvedant vis kitus pradinius duomenis. Tokia programa gali pasinaudoti ir kiti, ne tik jos autorius.

KARTOJIMO UŽDAVINIAI

87. Parašykite programą duotam natūriniam skaičiui spausdinti taip, kad kiekvienoje eilutėje būtų po vieną jo skaitmenį,

pakartotą tiek kartų, kokia jo reikšmė. Pavyzdžiui, jei duota 471, tai turėtų būti spausdinama taip:

```
4444
7777777
1
```

Patikslinkite uždavinio formulavimą. Į kokius paprastesnius uždavinius galima suskaidyti šį uždavinį? Kokius kontrolinius duomenis reikėtų parinkti?

88. Duota natūrinių skaičių seka. Sekos pabaigos požymis — nulis. Parašykite programą kiekvieno skaičiaus skaitmenų sumai ir sandaugai apskaičiuoti bei patikrinti ar skaičius yra kurio nors natūrinio skaičiaus kvadratas. Sugalvokite, kaip spausdinsite rezultatus. Kokius kontrolinius duomenis reikėtų parinkti?

89. Parašykite programą tokiems *n*-ženkliams natūriniams skaičiams rasti, kurių pirmąjį skaitmenį perkėlus į galą, būtų gaunamas skaičius, kartotinis arba dalus duotajam. Į kokius paprastesnius uždavinius galima suskaidyti šį uždavinį? Pasirinkite kontrolinius duomenis.

90. Natūrinį skaičių užrašius atvirkščia tvarka, gali atsitikti taip, kad mažesnis iš jų yra didesniojo daliklis. Pavyzdžiui,

$$8712 = 2178 \times 4$$

Parašykite programą visiems tokiems intervalo [1; 1000] skaičiams rasti. Uždavinį suskaidykite į paprastesnius uždavinius.

Tryliktoji pamoka

PROGRAMAVIMO KULTŪRA

Programuotojai, rašantys neaiškias, griozdiškas programas, mėgsta teisintis, kad programa skiriama kompiuteriui, o ne žmogui. Be abejo, kompiuteriui programos aiškumas ir vaizdumas nesvarbus — jis mechaniškai atlieka nurodytus veiksmus. Tačiau kad ir labai keista, pagrindinis programų skaitytojas vis dėlto yra žmogus, o ne kompiuteris. Skaitydamas programas, žmogus susipažįsta su kitų programuotojų idėjomis ir patirtimi, mokosi pats sudarinėti programas. Dažnai tenka tobulinti ir pačių sukurtas programas. Visais šiais atvejais reikia gilintis į programos esmę. Ką tik parašytą, dar tebesančią atmintyje programą skaityti ne taip sunku. Tačiau ilgainiui ji pamirštama. Todėl *programos turi būti rašomos aiškiai, vaizdžiai, suprantamai.*

Programavimo vadovėliuose vis daugiau dėmesio skiriama gero programavimo stiliaus ugdymui, sukurta nemažai taisyklių, kaip rašyti aiškias programas.

Ankstesnėse pamokose buvo kalbėta apie prasmingų vardų parinkimą. Tai kaip tik vienas programavimo kultūros elementų, gero stiliaus požymis. *Prasmingi vardai* leidžia greičiau suprasti programą. Tačiau kai vardai pernelyg ilgi, programa tampa griozdiška, o sutrumpinus juos, dažnai pasidaro nebeaiški prasmė. Tada į programą patogu įterpti papildomą tekstą, kuris programos atlikimui neturėtų jokios įtakos, tačiau padėtų ją skaityti. Toks tekstas vadinamas komentarais. Jie paaiškina ne tik kintamųjų vardus, bet ir atskiras programos dalis, nurodo, kas atliekama vienu ar kitu sakiniu ir panašiai. *Komentarus galima įterpti visur: tarp atskirų simbolių, žodžių, skaičių, vardų. Jie rašomi tarp* (* *ir* *) *simbolių.*

Komentarai padeda greitai ir lengvai skaityti programas. Tačiau jais nereikia piktnaudžiauti. Jie turi būti lakoniški, trumpai nusakantys pagrindinius dalykus, neužgriozdinantys programos teksto.

Paminėsime dar vieną programavimo kultūros elementą — programų redagavimą. *Redagavimu vadinamas programos teksto išdėstymas popieriaus lape.* Nekyla abejonių, kad žmogui kur kas lengviau skaityti vaizdžiai išdėstytą programą. Be to, tokioje programoje būna mažiau klaidų, lengviau jas surasti ir pataisyti (pavyzdžiui, nepamiršime parašyti žodžio **end**, jei jį visada rašysime vertikaliai po jį atitinkančiu žodžiu **begin**).

Visose pamokose stengėmės pateikti suredaguotas programas. Laikėmės kai kurių bendrų redagavimo taisyklių: sakinius, esančius kitame sakinyje, rašydavome patraukę į dešinę per keletą pozicijų, vertikaliai lygiavome žodžius **begin** ir **end** ir panašiai.

P a v y z d y s. Sudarykime programą populiariam matematikos uždaviniui, kurį 1202 metais suformulavo italų matematikas Fibonačis, spręsti.

Triušių pora kas mėnesį atsiveda po du triušiukus (patelę ir patinėlį), o iš jų po dviejų mėnesių jau gaunamas prieauglis. Kiek triušiukų bus po metų, jei pradžioje turėjome vieną triušių porą?

Iš sąlygos aišku, kad pirmojo mėnesio pabaigoje turėsime dvi triušių poras. Antrojo mėnesio pabaigoje prieauglį duos tik pirmoji pora, todėl turėsime tris poras, o dar po mėnesio prieauglį duos pradinė pora ir pora, gimusi prieš du mėnesius. Todėl iš viso bus 5 poros.

Simboliu $F(n)$ pažymėkime triušių porų skaičių po n mėnesių. Matome, kad n-ojo mėnesio pabaigoje turėsime tiek porų, kiek jų buvo prieš mėnesį, t. y. $F(n-1)$, ir dar tiek naujų porų, kiek jų buvo prieš du mėnesius, t. y. $(n-2)$-ojo mėnesio pabaigoje. Kitaip sakant, gausime tokią priklausomybę:

$$F(n)=F(n-1)+F(n-2)$$

Pateiksime programą triušių porų skaičiui po n mėnesių rasti ir spausdinti.


```
program fibonacci;
   var fn, (* F(n) *)
       fn1, (* F(n-1) *)
       fn2, (* F(n-2) *)
       n, (* mėnesių skaičius *)
       k : integer;
begin
  fn1 := 1;
  fn := 1; (* F(0) *)
   read (n);
   for k := 1 to n do
      begin
         fn2 := fn1; (* praėjo *)
         fn1 := fn; (* vienas *)
         fn := fn1 + fn2 (* mėnuo *)
      end;
   write (fn)
end.
```

Kai pradinis duomuo 12, kompiuteris išspausdina skaičių 377. Vadinasi, po 12 mėnesių turėsime 377 triušių poras.

KARTOJIMO UŽDAVINIAI

91. Duota programa:

```
program žvaigždė:
   var n, (* žvaigždės spindulio ilgis *)
       i, j, s : integer;
begin
   read (n);
for i := -n to n do
   for j := -n to n do
      begin
         x := i * j;
         if x<0
            then x := -x;
         if x<n
            then write ('*')
            else write (' ')
      end
end.
```

Atlikus ją, turėtų būti spausdinama keturkampė žvaigždė. Tačiau programoje yra klaidų. Ištaisykite jas.

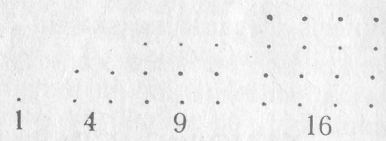
92.* Autobuso bilietas laikomas laimingu, jeigu jo numerio pirmųjų trijų skaitmenų suma lygi paskutiniųjų trijų sumai. Parašykite programą visiems intervalo $[m, n]$ laimingų bilietų numeriams rasti ir spausdinti (m, n — pradiniai duomenys).

93.* Pradiniai duomenys — natūrinių skaičių seka. Sekos pabaigos požymis — nulis. Parašykite programą mažiausiam ir didžiausiam sekos nariui rasti ir spausdinti. Nulis sekos nariu nelaikomas.

94.* Pradiniai duomenys — du natūriniai skaičiai a ir b, reiškiantys stačiakampio kraštinių ilgius. Parašykite programą, pagal kurią būtų suskaičiuojama, į kiek kvadratų, kurių kraštinės išreiškiamos natūriniais skaičiais, galima suskirstyti duotąjį stačiakampį, jei nuo jo kaskart atskiriamas didžiausio ploto kvadratas. Pavyzdžiui, kai pradiniai duomenys 5 ir 2, tai rezultatas turi būti 4.

95. Pradinis duomuo — natūrinis skaičius n. Parašykite programą iš eilės einančių natūrinių skaičių sekai, kurios narių suma būtų lygi duotajam skaičiui, spausdinti. Jei tokios sekos nėra, išspausdinamas pranešimas SPRENDINIO NĖRA.

96. Kvadratišku skaičiumi vadinamas toks skaičius, kurį galima pavaizduoti kvadratu išdėstytais taškais, pavyzdžiui:



Trikampišku skaičiumi vadinamas toks skaičius, kurį galima pavaizduoti trikampiu išdėstytais taškais, pavyzdžiui:

36 yra ir kvadratiškas, ir trikampiškas skaičius.

Sudarykite programą visiems intervalo [1; n] skaičiams, kurie yra ir kvadratiški, ir trikampiški, rasti.

97.* Parašykite programą visiems natūriniams skaičiams, kurie lygūs savo skaitmenų kubų sumai, rasti ir spausdinti.

98. Parašykite programą visiems natūriniams skaičiams, kurie lygūs savo skaitmenų sumos kubui, rasti ir spausdinti.

99. Parašykite programą visiems pirminiams skaičiams, mažesniems už skaičių *nmax*, rasti ir spausdinti.

100. Parašykite programą, kuri nustatytų, ar galima duotąjį natūrinį skaičių išreikšti dviejų natūrinių skaičių kvadratų suma. Jei galima, spausdiname duotąjį skaičių, žodį TAIP bei tuos du natūrinius skaičius, priešingu atveju — skaičių ir žodį NE. Jei galimi keli sprendimo variantai (pavyzdžiui, $50^2=1^2+7^2=5^2+5^2$), reikia rasti ir spausdinti visus.

Uždavinių sprendimai ir atsakymai

1. a) 8; b) 2; c) 0. **2.** c) 40; 41; 42; 43; 44; d) 7. **5.** $a=5$, $b=5$. **6.** a) $x=7$, $y=7$; b) $x=10$, $y=100$. **10.** a)

```
x2 := x * x;
x4 := x2 * x2;
x7 := x4 * x2 * x
```

12. Galimi įvairūs sprendimai. Pateikiame du:

```
1) b := a div 100 * 10+a mod 10
2) x := a div 100;
   y := a mod 10;
   b := x * 10+y
```

23. a) 50; b) 10. **24.** $a=2$, $b=2$.

25. a)

```
if x<0
  then y := -x * x
  else if x mod 2=0
         then y := x
         else y := x+1
```

26.

```
olimp := 0;
if m≥1896
  then if (m-1896) mod 4=0
         then olimp := (m-1896) div 4+1
```

34.

```
if min2<min1
  then begin
         min2 := min2+60;
         h2 := h2-1
       end;
min := min2-min1;
if h2<h1
  then h2 := h2+24;
h := h2-h1
```

38.

```
a := n; kiek := 0;
while a>0 do
  begin
    kiek := kiek+1;
    a := a div 10
  end
```

39.

```
p := n; a := 0;
while p>0 do
  begin
    a := a * 10+p mod 10;
    p := p div 10
  end
```

49.

```
sum := 0;
vnt := 1;
for i := 1 to n do
  begin
```

```
        sum := sum+vnt;
        vnt := vnt * 10+1
    end
```

50.
```
s := 0;
for i := 1 to n do
  if i mod 2 = 1
    then s := s+i*(i+2)
```

55. a) išorinis ciklas atliekamas 1 kartą, vidinis — 11 kartų.

60. Pateikiame du variantus.

1)
```
k := 0;
for a := 1 to 9 do
  for b := 0 to 9 do
    for c := 0 to 9 do
      if b = a+c
        then k := k+1
```

2)
```
k := 0;
for a := 1 to 9 do
  for c := 0 to 8 do
    if a+c < 10
      then k := k+1
```

69. $a = -5$

71. a)
```
ABBB
AABB
AAAB
AAAA
```

b)
```
ABBBBBB
AABBBBB
AAABBBB
AAAABBB
AAAAABB
AAAAAAB
AAAAAAA
```

72.
```
U===U
U===U
U===U
UUUUU
```

79. Praleistos sąlygos: $d \leqslant x$ ir $x * x$ **mod** $d = x$.

81.
```
program stplotas;
  var a, b, p : integer;
begin
  read (p);
  a := 1; b := p;
  while a ≤ b do
    begin
      if a * b = p
        then writeln (a, b);
      a := a+1;
      b := p div a
    end
end.
```

82. Vienas paprastesnių sprendimų yra toks:
```
program dalumas;
  var n, s, d : integer;
begin
  read (n);
  for s := 1 to n do
    begin
      write (s); write (' ');
      for d := 1 to s do
        if s mod d = 0
          then write ('+');
      writeln
    end
end.
```

Šis sprendimas nėra efektyvus, tačiau kadangi iš uždavinio sąlygos aišku, kad n negali būti labai didelis, tai skaičiavimų bus nedaug ir šiuo aspektu nėra reikalo tobulinti programos.

85. 50331

Pagal šią programą duotasis skaičius spausdinamas atvirkščiai (pradedant nuo paskutiniojo skaitmens).

86.
```
*
**
***
*****
*********
*************
*********************
*****************************
```

Pagal šią programą žvaigždutėmis spausdinami Fibonačio sekos pirmieji $n+1$ nariai.

92.
```
program laimingi;
  var m, n, (* intervalas *)
      nr : integer;
begin
  read (m, n);
  for nr := m to n do
   if nr div 100000+
      nr div 10000 mod 10+
      nr div 1000 mod 10=
      nr mod 1000 div 100+
      nr mod 100 div 10+
      nr mod 10
      then write (nr)
end.
```

93.
```
program minmax;
  var min, max, n : integer;
begin
  read (n);
  min := n; max := n;
  while n ≠ 0 do
    begin
      if n < min
        then min := n
        else if n > max
               then max := n;
      read (n)
    end;
  write (min, max)
end.
```

94.
```
program kvadratai;
  var a, b, (* kraštinių ilgiai *)
      n, (* kvadratų skaičius *)
      x : integer;
begin
  read (a, b);
  n := 0;
  while b ≠ 0 do
    begin
      n := n+a div b;
      x := b;
      b := a mod b;
      a := x
    end;
  write (n)
end.
```

97. Kompiuteris visų natūrinių skaičių patikrinti negali. Vadinasi, pirmiausia reikia pamėginti nustatyti intervalą, kuriame galima tikėtis ieškomų skaičių. Gal jis baigtinis?

Didžiausias skaitmuo yra 9. Taigi didžiausia vienaženklių skaičių skaitmenų kubų suma yra $9^3=729$, dviženklių — $9^3+9^3=1458$, triženklių — $9^3+9^3+9^3=2281$, keturženklių — $9^3+9^3+9^3+9^3=2916$. Didžiausia penkiaženklių skaičių skaitmenų kubų suma yra 3645, t. y. tik keturženklis skaičius. Vadinasi, penkiaženkliai ir didesni skaičiai netenkina nurodytos sąlygos ir todėl jų netikrinsime. Taip pat galima netikrinti ir keturženklių skaičių, didesnių už 2916. Dar labiau įsigilinę, pamatysime, kad skaičius, kurio skaitmenų kubų suma didžiausia ir kuris mažesnis už 2916, bet didesnis už 2000, yra 2899. Šio skaičiaus skaitmenų kubų suma lygi $2^3+8^3+9^3+9^3=1978$, t. y. mažesnė už 2000. Vadinasi, skaičiai, kurie yra didesni už 2000, mūsų nedomina. Pakanka tikrinti intervalo [1; 2000] skaičius.

Dabar pateikiame sudarytą programą:

```
program kubai (output);
   var x, (* tiriamasis skaičius *)
       n, (* paskutinysis skaitmuo *)
       p, (* skaičius be paskutiniojo skaitmens *)
       s : integer; (* skaitmenų kubų suma *)
begin
   for x := 1 to 2000 do
      begin
         s := 0;
         p := x;
         while p>0 do (* skaičiaus x skaitmenų kubų *)
            begin (* sumos radimas *)
               n := p mod 10;
               p := p div 10;
               s := s+n * n * n
            end;
         if x=s
            then writeln (x)
      end
end.
```

Įdomumo dėlei pateiksime ir kompiuteriò, atlikusio šią programą, išspausdintus rezultatus:

```
1
153
370
371
407
```

Atkreipkite dėmesį dar į tai, kad programa, skirtingai nuo daugumos čia pateiktų, yra vienkartinė. T. y. ji nepriklauso nuo pradinių duomenų, ją pakanka atlikti vieną kartą ir gauti rezultatus. Vienkartinių programų efektyvumas nėra itin svarbus. Todėl mes ir nebandėme dar labiau susiaurinti tiriamo intervalo.

Pabaigos žodis

Perskaitę šią knygelę, susipažinote su svarbiausiomis programavimo konstrukcijomis: kintamojo sąvoka, reikšmių priskyrimo, sąlyginiu, sudėtiniu, ciklo sakiniais. Šiek tiek buvo paliesti įvedimo bei išvedimo, programavimo metodikos klausimai. Nemaža dėmesio skyrėme programavimo kultūros elementams.

Šio leidinio medžiagos visiškai pakanka norint sudaryti užbaigtas paprastų uždavinių programas. Tai dažniausiai galvosūkių, žaidimų uždaviniai, kuriems išspręsti užtenka sveikųjų skaičių, kadangi knygelėje vartojama tik ši duomenų rūšis. Norint sudaryti sudėtingesnių uždavinių programas, pirmiausia reikės išmokti vartoti kitų rūšių duomenis: loginius, simbolinius, vardinius, realiuosius (trupmeninius) skaičius, taip pat duomenis, kurių reikšmės yra sudėtinės, sudarytos iš kitų duomenų reikšmių — masyvus, įrašus, aibes.

Sudaryti sudėtingesnio uždavinio programą iš karto gana sunku. Todėl uždavinys skaidomas į smulkesnes dalis ir kiekviena jų programuojama atskirai. Apie tai tik trumpai užsiminėme vienoje iš paskutiniųjų pamokų. Atskiroms programų dalims apiforminti yra vartojamos specialios programavimo konstrukcijos — funkcijos ir procedūros. Jas reikės išmokti tiems, kas toliau domėsis programavimu.

Išmokę vartoti minėtas duomenų rūšis bei funkcijas ir procedūras, būsite susipažinę su visomis pagrindinėmis programavimo konstrukcijomis. Toliau plėsti savo žinias galite besimokydami neakivaizdinėje Jaunųjų programuotojų mokykloje arba savarankiškai skaitydami knygas, išvardytas literatūros sąraše.

Programoms užrašyti vartojome Paskalio kalbą, tačiau mokėmės ne jos, o tik programavimo konstrukcijų pagrindų. Gerai supratus pagrindines programavimo konstrukcijas, nesunku išmokti ir visą Paskalio arba kurią nors kitą programavimo kalbą. Tam reikia tik perskaityti tą kalbą aprašantį leidinį.

Pateikdami teorinę medžiagą, pratimus, pavyzdžius, stengėmės ugdyti skaitytojo gerą programavimo stilių, kuris išliktų ir programuojant kitomis, net žemo lygio (pavyzdžiui, Fortranas, Beisikas) kalbomis.

Jei skaitant šią knygelę, kils klausimų, pasiūlymų, neaiškumų arba patys sudarysite naujų uždavinių bei programų, patobulinsite čia pateiktas programas, rašykite mums adresu: 232600 Vilnius, Akademijos g. 4, Matematikos ir kibernetikos institutas.

Literatūra

1. Grigas G. Programavimo pradmenys.— V., 1982.
2. Dagienė V., Grigas G., Petrauskienė A. Paskalio programavimo kalba.— V., 1983.
3. Dagienė V., Grigas G. Programavimo pradmenų uždavinynas.— V., 1983.
4. Grigas G. Duomenų tipai.— V., 1984.
5. Augutis K., Dagienė V., Grigas G. Duomenų tipų uždavinynas.— V., 1984.
6. Dagienė V., Grigas G., Augutis K. Šimtas programavimo uždavinių.— K., 1986.