LIETUVOS RESPUBLIKOS ŠVIETIMO IR MOKSLO MINISTERIJA
NACIONALINIS EGZAMINŲ CENTRAS

2006 METŲ INFORMACINIŲ TECHNOLOGIJŲ VALSTYBINIO BRANDOS EGZAMINO REZULTATŲ

KOKYBINĖ ANALIZĖ

Jonas Blonskis, Povilas Leonavičius

TESTAS

1. KOKYBINĖS ANALIZĖS TIKSLAI, UŽDAVINIAI, ŠALTINIAI

Informacinių technologijų valstybinio brandos egzamino rezultatų kokybinės analizės tikslai:

- išsiaiškinti, kaip egzamino turinys dera su išsilavinimo standartais ir egzamino programa;
- išanalizuoti, kokios mokinių žinios ir gebėjimai buvo tikrinami;
- įvertinti informacinių technologijų testo dalies svarbą;
- išsiaiškinti programavimo testo dalies sudėtingumą;
- įvertinti programavimo užduočių sudėtingumą, kokybę ir svarbą;
- apžvelgti valstybinio egzamino rezultatus ir aptarti, kaip mokiniai geba parodyti savo turimas informacinių technologijų ir programavimo žinias bei supratimą ir kaip geba pritaikyti praktinio pobūdžio užduotims spręsti;
- atkreipti mokinių ir jų informacinių technologijų mokytojų dėmesį į sudėtingiausias ir problemiškiausias ugdymo turinio temas;
- pateikti rekomendacijų 2007 m. informacinių technologijų valstybinio brandos egzamino užduočių rengėjams, kaip reikėtų tobulinti egzamino testą, praktines užduotis ir jų vertinimą;
- palaikyti grįžtamąjį ryšį tarp užduočių rengėjų ir informacinių technologijų dalyko mokytojų bei mokinių;
- tirti vertintojų kvalifikaciją ir inicijuoti valstybinio egzamino vertintojų kvalifikacijos tobulinimą.

Šios analizės uždaviniai:

- nustatyti, kurie klausimai ir praktinės užduotys buvo sunkiausi ir kurie lengviausi valstybinį brandos egzaminą laikiusiems mokiniams;
- kokie klausimai geriausiai diferencijavo mokinius;
- išryškinti tipiškas, dažniausiai mokinių daromas klaidas ir aptarti galimas jų priežastis;
- išsiaiškinti, kokių žinių ir gebėjimų mokiniams trūksta labiausiai;
- pateikti rekomendacijas užduočių rengėjams, mokytojams ir mokiniams, kurie ruošiasi laikyti informacinių technologijų valstybinį brandos egzaminą;
- įvertinti, kaip egzamino turinys atitinka egzamino tikslus.

Atliekant kokybinę informacinių technologijų valstybinio brandos egzamino rezultatų analizę remtasi šiais šaltiniais:

- Lietuvos bendrojo lavinimo mokyklos bendrosios programos ir išsilavinimo standartai XI–XII klasėms. /Patvirtinta Lietuvos Respublikos švietimo ir mokslo ministro 2002 m. rugpjūčio 21 d. įsakymu Nr. ISAK-1465.
- Informacinių technologijų brandos egzaminų programa. /Patvirtinta Lietuvos Respublikos švietimo ir mokslo ministro 2005 m. liepos 21 d. įsakymu Nr. ISAK-1524.
- 2006 m. Informacinių technologijų valstybinio brandos egzamino rezultatų statistinė analizė. Nacionalinis egzaminų centras, 2006.
- Informacinės technologijos. 2006 m. valstybinio brandos egzamino užduotis. Nacionalinis egzaminų centras, 2006.
- Informacinės technologijos. Valstybinio brandos egzamino vertinimo instrukcija. Nacionalinis egzaminų centras, 2006.
- Atsitiktinės atrankos principu atrinktų 2006 metų informacinių technologijų valstybinio brandos egzamino mokinių testų atsakymų lapai.
- 2006 metų informacinių technologijų valstybinio brandos egzamino mokinių darbų praktinės užduotys.

Egzamino užduotys parengtos remiantis informacinių technologijų valstybinio brandos egzamino

© Nacionalinis egzaminų centras, 2007

programos reikalavimais ir sudarytos laikantis programoje pateiktos egzamino matricos. Egzamino matricos paskirtis – užtikrinti, kad egzamino užduoties taškų skaičius pagal atskiras temas proporcingai atitiktų tų temų apimtį. Taip pat egzamino matrica reglamentuoja, kiek taškų mokinys gali gauti už žinias bei supratimą ir kiek taškų už gebėjimą spręsti problemas. Toliau pateikiame užpildytą 2006 m. informacinių technologijų valstybinio brandos egzamino pagrindinės sesijos užduoties matricą.

1 lentelė. Informacinių technologijų valstybinio brandos egzamino matrica

GEBĖJIMAI / SRITYS		ŽINIOS IR SUPRATIMAS	PRAKTINIAI ĮGŪDŽIAI	PROBLEMŲ SPRENDIMAS	Iš viso (%)	
					egzamino užduotyje	numatyta programoje
Teksto tvarkymas Naudojimasis skaičiuokle Žiniatinklis ir elektroninis paštas Pateikčių rengimas Socialiniai ir etiniai aspektai		T1(2)*, T2(1), T3(2), T4(2), T7(2), T9(2), T10(1), T11(2), T12(2)	T2(1), T5(2), T6(2), T8(3), T10(1)		25	**25**
Programavimas		T13(1), T14(1), T15(1), T16(1), T17(1), T18(1), T19(1), T20(2), T21(1), T22(1), P1(2), P2(2)	T13(1), T14(1), T15(1), T16(1), T17(1), T18(1), T19(1), T20(2), T21(3), T22(2), P1(1), P2(4)	P1(14), P2(27)	75	**75**
Iš viso (%)	egzamino užduotyje	31	28	41	100	
	numatyta programoje	**30**	**30**	**40**		**100**

*Pastaba. Žymėjimas **T1(2)** reiškia testo pirmąjį klausimą ir jo įvertinimą taškais, **P1(2)** reiškia pirmąją praktinę užduotį ir jos įvertinimą taškais.

Konkrečiose egzamino užduotyse galimi ne didesni kaip 5 proc. nukrypimai nuo egzamino matricoje pateikiamų skaičių.

Pateiktoje matricoje matyti, kad taškų pasiskirstymas pagal atskiras temas 2006 m. informacinių technologijų valstybinio brandos egzamino pagrindinės sesijos užduotyje visiškai atitiko numatytąjį informacinių technologijų brandos egzaminų programoje. Taškų pasiskirstymas pagal gebėjimus 2006 m. informacinių technologijų valstybinio brandos egzamino pagrindinės sesijos užduotyje atitiko numatytąjį informacinių technologijų brandos egzaminų programoje.

2. TESTO REZULTATŲ KOKYBINĖ ANALIZĖ

Užduoties testo dalį sudarė 22 klausimai: 15 klausimų su pasirenkamaisiais atsakymais ir 7 atvirojo tipo klausimai.

Į užduoties testo dalį buvo įtraukta 12 klausimų iš penkių informacinių technologijų sričių: *teksto tvarkymas, naudojimasis skaičiuokle, žiniatinklis ir elektroninis paštas, pateikčių rengimas, socialiniai ir etiniai aspektai*, t. y. iš pagrindinių informacinių technologijų dalyko sričių, ir 10 klausimų iš programavimo srities temų: *priskyrimo operacija, sakinys; paprasti kintamieji, loginis kintamasis; ciklai* `for`, `while`; *blokas* `begin ... end`; *sąlygos sakinys* `if`; *loginės operacijos* (`and`, `or`, `not`), *spausdinimas ekrane* (`Write`, `WriteLn` *procedūros*); *skaitymas iš tekstinio failo; masyvo, įrašo* (`record`) *duomenų tipai; masyvo tipo kintamasis; procedūra, parametrų sąrašas, kreipinys; funkcija, parametrų sąrašas, kreipinys; skaičių sumos, sandaugos algoritmai, programos struktūra*.

Informacinių technologijų valstybinį brandos egzaminą 2006 metais laikė 1164 kandidatai. Pateikiame pagrindinės sesijos valstybinio brandos egzamino užduoties **informacinių technologijų srities temų** testo klausimų pasiskirstymą 2 lentelėje, kur spalvoti stulpeliai žymi atviro tipo klausimus.

2 lentelė. Informacinių technologijų srities temų testo klausimų pasiskirstymas.

Testo klausimo numeris	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Testo klausimo vertė taškais	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**3**	**2**	**2**	**2**	**2**
Teksto tvarkymas	+	+	+	+								
Naudojimasis skaičiuokle					+	+	+	+				
Žiniatinklis ir el. paštas											+	
Pateikčių rengimas										+		
Socialiniai ir etiniai aspektai									+			+

Pateikiame pagrindinės sesijos valstybinio brandos egzamino užduoties **programavimo srities temų** testo klausimų pasiskirstymą 3 lentelėje, kur spalvoti stulpeliai žymi atviro tipo klausimus.

3 lentelė. Programavimo srities temų testo klausimų pasiskirstymas.

Testo klausimo numeris	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
Testo klausimo vertė taškais	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**4**	**4**	**3**
Priskyrimo operacija, sakinys, paprasti kintamieji	+	+	+	+	+	+	+	+	+	+
Loginis kintamasis			+			+				
Ciklas `for`			+	+					+	+
Ciklas `while`	+	+								
Blokas `begin ... end;`	+	+	+		+					
Sąlygos sakinys `if`			+	+		+	+			
Loginės operacijos (`and`, `or`, `not`)		+	+				+			
Programos struktūra	+	+	+	+	+	+	+	+	+	
Spausdinimas ekrane (`Write`, `WriteLn` procedūros)	+				+	+	+	+	+	+
Skaitymas iš tekstinio failo						+	+			
Masyvo duomenų tipas, masyvo tipo kintamasis				+					+	+
Įrašo (`record`) duomenų tipas										+
Procedūra, parametrų sąrašas, kreipinys					+					
Funkcija, parametrų sąrašas, kreipinys								+	+	
Skaičių suma, sandauga (algoritmas)	+	+	+			+			+	
Rašymas į failą										

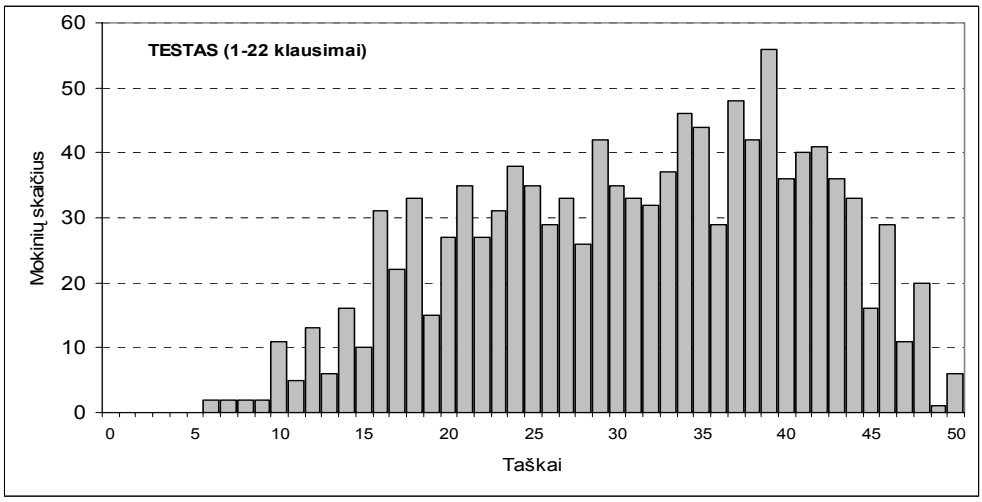
Programavimo testo 13, 14, 15, 16, 18 ir 19 klausimai skirti išsiaiškinti, kaip mokiniai suvokia, supranta ir žino elementarias Paskalio kalbos konstrukcijas (ciklus, sąlygos sakinius, bazinius duomenų tipus, masyvą).

17, 20 ir 21 klausimai yra sudėtingesni. Jų tikslas yra įvertinti esminių programavimo priemonių – funkcijų ir procedūrų – suvokimą, jų praktinio taikymo supratimą. Rezultatai rodo, kad ugdymo procese mokytojai tam skiria nepakankamai dėmesio.

Paskutinis 22 klausimas nesudėtingas. Juo tikrinamas įrašo duomenų tipo supratimas.

Remiantis 2006 m. informacinių technologijų valstybinio brandos egzamino rezultatų statistine analize galima teigti, kad tarp dvidešimt dviejų testo klausimų keturi buvo sunkūs (2, 4, 17 ir 21), penkiolika klausimų buvo optimalaus sunkumo ir trys klausimai labai lengvi. Taigi 2006 m. teste vyravo optimalaus sunkumo klausimai, nors maksimalų 50-ties taškų įvertinimą gavo tik 0,52 proc. mokinių, laikiusių egzaminą. Visas testas buvo labai gero sunkumo (62,32 proc.).

Kaip klausimas padeda diferencijuoti mokinius, rodo jo skiriamoji geba. Teste vyravo geros skiriamosios gebos klausimai: nuo 34,1 iki 78,22 proc. Tik trys testo klausimai (2, 10 ir 12) buvo ne itin geros skiriamosios gebos, vienas iš jų – atvirojo tipo klausimas. Visas testas buvo geros skiriamosios gebos (42,26 proc.).



1 pav. 2006 m. informacinių technologijų valstybinį brandos egzaminą laikiusių kandidatų 1–22 klausimų taškų pasiskirstymas

Aptarsime geriausius 2006 m. informacinių technologijų valstybinio brandos egzamino testo klausimus.

8. Kokį rezultatą suskaičiuos skaičiuoklės langelyje C1 įrašyta formulė?

	A	B	C
1	2	0	=IF(NOT(SUM(A1:A5)>15);SUM(A1:A5);AVERAGE(B1:B5))
2	5	6	
3	6	-5	
4	7	2	
5	8	7	
6			

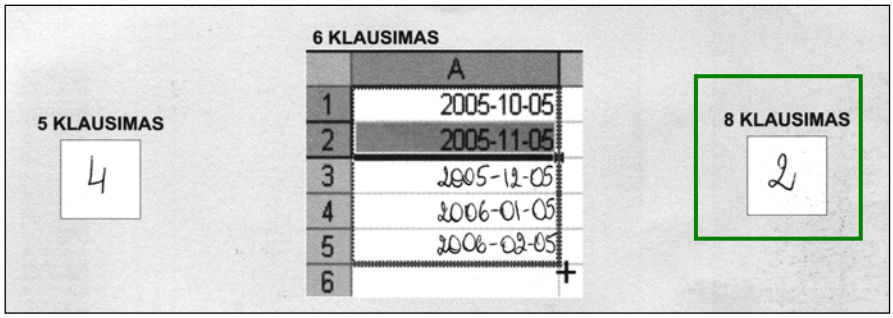
Atsakymas

(3 taškai)

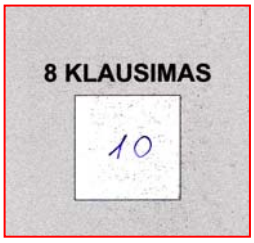
Taškų pasiskirstymas (%)				*Sunkumas*	*Skiriamoji geba*	*Koreliacija*
0	1	2	3			
36,17	0,00	0,00	63,83	63,83	57,88	0,50

Tai atvirojo tipo klausimas, apibendrinantis kelių skaičiuoklės temų žinias. Juo tikrinama, kaip mokiniai supranta formulių ir funkcijų paskirtį, moka apskaičiuoti langelių bloko skaitinių reikšmių sumą, vidurkį, žino loginės funkcijos NOT paskirtį formulėje; žino, kaip taikoma sąlygos funkcija. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 2.4. Formulės ir funkcijos. Paaiškinti formulių ir funkcijų paskirtį, žinoti, kur jas taikyti. Mokėti naudoti funkcijas: SUM, AVERAGE, MIN, MAX, IF, NOT, AND, OR.)

Dauguma mokinių (63,66 proc.) teisingai atsakė į šį klausimą. Galima teigti, kad mokiniai turi pakankamai šios temos žinių.



12,5 proc. mokinių įrašė klaidingą atsakymą 28, nes nesuvokė ar nepastebėjo loginės funkcijos NOT; maždaug kas dvidešimtas mokinys klydo (5,25 proc.), nes skaičiavo bloko B1:B5 skaičių sumą vietoj aritmetinio vidurkio (neteisingas rezultatas 10).



20. Nustatykite, **ką** spausdins programa `Te8`. Atsakymą įrašykite į pateiktą lentelę.

```
program Te8;
{----------------------------------------}
function FF(a, b, c : integer): integer;
begin
  a := a + b + c;
  FF := (a + c) div 2;
end;
{----------------------------------------}
var x, y, z, s : integer;
begin
  x := 5;  y := 6;  z := 8;
  s := FF(x, y, z);
  WriteLn(s);
  s := s + FF(y, x, 5);
  WriteLn(s);
  WriteLn(FF(x, y, FF(x, y, z)));
  WriteLn(x + y + z);
end.
```

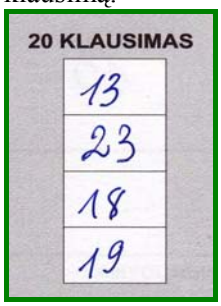

(4 taškai)

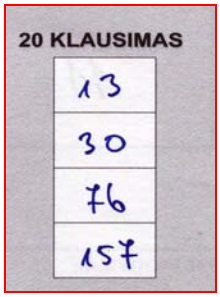
Kiekvienos teisingos reikšmės įrašymas vertinamas 1 tašku.

Taškų pasiskirstymas (%)					*Sunkumas*	*Skiriamoji geba*	*Koreliacija*
0	1	2	3	4			
13,06	22,25	5,07	14,26	45,36	64,15	59,10	0,62

Tai atvirojo tipo klausimas. Juo buvo tikrinami darbo su funkcija įgūdžiai: mokėjimas dirbti su funkcijos parametrais ir argumentais, taisyklingai kreiptis į funkciją. Taip pat reikėjo apskaičiuoti operacijos `div` rezultatus ir pateikti juos naudojant procedūrą `WriteLn`. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.5. Procedūros ir funkcijos. Mokėti skaidyti uždavinį į dalis, jas užrašyti procedūromis ir funkcijomis. Skirti procedūros ir funkcijos sąvokas. Paaiškinti jų vartojimo būdus, gebėti taikyti uždaviniuose.)

Atsakymų analizė parodė, kad į šį klausimą teisingai atsakė dauguma mokinių, tačiau nemaža mokinių dalis (apie 30 proc.) suklydo skaičiuodami, netaisyklingai kreipėsi į funkciją arba netinkamai naudojo `WriteLn` procedūrą rezultatams spausdinti. 13 proc. mokinių visiškai nesugebėjo atsakyti į šį klausimą.





22. Pažymėkite raide **T** teisingus programos fragmentus, neteisingų programos fragmentų klaidas trumpai paaiškinkite.

Duoti aprašai:

```
type Tvienas = record
                 x, y : integer;
               end;
     Tmas = array[1..50] of Tvienas;
var A, B, D : Tmas; n : integer;
    x, y : Tvienas;
    c : integer;
```

	Programos fragmentai	Atsakymas: T arba neteisingų programos fragmentų klaidų paaiškinimai

22.1	`...`
	`x := A[5];`
	`y := A[6];`
	`...`	

22.2	`...`
	`x := 9;`
	`y := x;`
	`...`	

22.3	`...`
	`A.x := 15;`
	`...`	

22.4	`...`
	`for c := 1 to n do`
	`  WriteLn(A[c].x + A[c].y);`
	`...`	

(3 taškai)

Taškų pasiskirstymas (%)				*Sunkumas*	*Skiriamoji geba*	*Koreliacija*
0	1	2	3			
7,30	42,53	25,52	24,66	55,84	55,49	0,75

Tai taip pat atvirojo tipo klausimas. Šis klausimas buvo nesudėtingas, nors visus tris taškus už jį surinko tik apie ketvirtadalis mokinių.

Klausimas reikalavo įrašo duomenų tipo supratimo ir mokėjimo apdoroti duomenų masyvą su įrašo tipo elementais. Mokiniai turėjo iš pateiktų programos fragmentų pasirinkti fragmentus su klaidomis ir tas klaidas pakomentuoti. Klaidos buvo sintaksės (priskyrimo sakinyje nenurodytas masyvo elemento indeksas, 22.3 fragmentas) ir loginio pobūdžio (priskyrimo operacijos kintamasis x yra įrašo tipo, o dešinėje priskyrimo sakinio pusėje – sveikoji skaitinė reikšmė, 22.2 fragmentas).

22 KLAUSIMAS	
22.1 T	22.2 Kintamasis x yra record (*įrašo*), o ne integer tipo, todėl toks priskyrimas yra neįmanomas
22.3 Nėra nurodytas masyvo indeksas (*elemento*)	22.4 T

arba

22 KLAUSIMAS	
22.1 T	22.2 x := 9 - blogai, nes įrašo tipo kintamasis negali būti lygus skaičiui.
22.3 Neurodytas masyvo elemento indeksas	22.4 T

Nemaža dalis „pusiau teisingų“ atsakymų, įvertintų 1 tašku, rodo, kad įrašo temai mokykloje nepakanka dėmesio ar/ir laiko, nes programavimo modulio programoje įrašo duomenų tipas yra viena iš paskutinių temų ir galbūt mokytojai daug daugiau dėmesio skiria darbui su skaičių masyvu.

22 KLAUSIMAS	
22.1 T	22.2 T
22.3 Netinkamai parašytas masyvo vardas, A, turi buti su laužtiniais skliausteliais pvz.: A[i].x	22.4 T

22 KLAUSIMAS	
22.1 Nenurodyta kuriam masyvo įrašui priskiriamos reikšmės (A[5].x; A[5].y).	22.2 T
22.3 Nenurodytas masyvo įrašo indeksas atliekant priskiriamo sakinį.	22.4 T

22 KLAUSIMAS	
22.1 T	22.2 T
22.3 Nėra laužtinių skliaustų prieš ir po x	22.4 Nereikalingi taškai prieš x ir y

Atsakydami į šį klausimą kai kurie mokiniai buvo labai originalūs.

22.1 Skaičiai nerašomi laužtiniuose skliaustuose.	22.2 negali būti taip ~~nes~~ (x=y), nes programa įrašoma.
22.3 vargu ar kintamojo varde gali būti taškas. ~~taip~~ Be to A yra masyvo elementas, todėl turėtų būti skliaustelis. (pirmas sakinys nubrauktas)	22.4 Na taškas gal ir gali būti, ~~todėl~~ T

22.4 taškas rašomas tik po programos pabaigos „end", kitur būti negali

Aptarsime kitus testo klausimus.

Teksto tvarkymo temos klausimai

1. Tekstų rengykle rengiamas dokumentas. Dokumento puslapio antraštėje (anglų k. *Header*) užrašyti žodžiai „Kalbos testas“. Ar galima antraštėje papildomai įterpti puslapio numerį?

A Ne, nes puslapio numeris gali būti nurodomas tik puslapio poraštėje (anglų k. *Footer*).
B Galima įterpti bet kurioje antraštės vietoje, net ir teksto viduje.
C Galima įterpti tik antraštės dešinėje pusėje.
D Galima įterpti tik antraštės kairėje pusėje.

(2 taškai)

Šiuo klausimu buvo tikrinama, kaip mokiniai suvokia teksto maketavimą tekstų rengykle. Mokiniai turėjo mokėti įdėti dokumento antraštę (anglų k. *Header*) ir žinoti, kaip joje galima nurodyti tekstą ir puslapių numerius. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 1.7. Teksto maketas, antraštės ir poraštės. Mokėti įdėti dokumentui antraštes (anglų k. *header*) ir poraštes (anglų k. *footer*), jose nurodyti pavadinimą, datą, puslapių numerius.)

Remiantis statistine egzamino rezultatų analize galima teigti, kad tai optimalaus sunkumo ir geros skiriamosios gebos klausimas. Nors į šį klausimą teisingai atsakė 58,59 proc. mokinių, tačiau nemažai mokinių suklydo atsakydami, kad įterpti puslapio numerį galima tik puslapio poraštėje (anglų k. *Footer*).

2. Tekstų rengykle parengta tokia lentelė:

I. Pramogos		II. Pastabos
III. Filmai	IV. Koncertai	

Kiek daugiausia šios lentelės langelių vienu metu galima sulieti į vieną, neprarandant langeliuose surinktų žodžių?

A Nė vieno langelio.
B Du langelius – III ir IV.
C Tris langelius – I, II ir IV.
D Visus keturis langelius.

(2 taškai)

Remiantis statistine egzamino rezultatų analize galima teigti, kad šis klausimas mokiniams buvo vienas iš sunkiausių – jo sunkumas 29,73 proc. Taip pat statistika rodo, kad šio klausimo skiriamoji geba yra mažiausiu iš visų testo klausimų – tik 1,15 proc.

Dauguma mokinių atsakė, kad vienu metu galima sulieti į vieną daugiausia tik du šios lentelės langelius – III ir IV (apie 41 proc.) arba nė vieno langelio (25 proc.). Toks mokinių pasirinkimas rodo, kad mokykloje darbui su lentelės langeliais ir lentelių pertvarkymui skiriama nepakankamai dėmesio, nors tokie veiksmai numatyti egzamino programoje. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 1.4. Lentelės. Mokėti sujungti ir perskelti lentelės langelius, keisti eilučių ir stulpelių požymius: dydį, spalvą, horizontalų bei vertikalų lygiavimą.)

7. Tekstų rengykle kuriamas dokumentas. Kurioje dokumento vietoje gali būti išnašos aiškinamasis tekstas?

A Tik puslapio pabaigoje arba žemiau teksto.
B Puslapio arba dokumento pabaigoje.
C Tik dokumento pabaigoje.
D Tik sekcijos pabaigoje.

(2 taškai)

Šiuo klausimu buvo tikrinama, kaip mokiniai supranta išnašos sąvoką, žino, kur dokumente galima ją įterpti. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 1.5. Automatinės teksto tvarkymo galimybės. Mokėti įdėti išnašas.)

Statistinė egzamino rezultatų analizė rodo, kad tai optimalaus sunkumo, bet ne itin geros skiriamosios gebos klausimas. Nors į šį klausimą teisingai atsakė 67,53 proc. mokinių, nemaža dalis mokinių nežinojo, kad išnašos aiškinamasis tekstas gali būti ne tik puslapio apačioje (puslapio išnaša), bet ir viso dokumento pabaigoje (dokumento išnaša). Maža skiriamoji geba rodo, kad klausimas blogai diferencijavo mokinius, t. y. kad į šį klausimą nesisekė teisingai atsakyti tiek geriau, tiek prasčiau egzaminą išlaikiusiems mokiniams.

4. Tekstų rengykle renkamas knygos rankraštis ir automatiškai sudaromas jos turinys. Pagal ką nustatoma, kad knygos skyrių pavadinimai turi būti įtraukti į turinį?

A Pagal pastraipos stilių.
B Pagal skaičių, surinktą prieš pavadinimą.
C Pagal šrifto dydį.
D Pagal šrifto stilių.

(2 taškai)

Šis klausimas egzaminą laikiusiems mokiniams buvo gana sudėtingas, į jį teisingai atsakė 38,49 proc. mokinių.

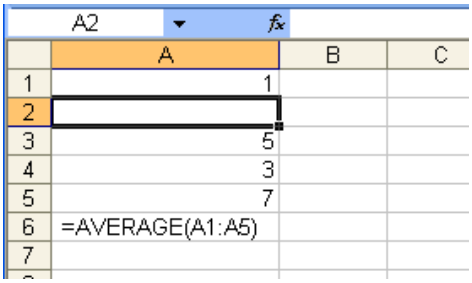
Klausimas reikalavo žinoti dokumento turinio sudarymo algoritmą ir suprasti, kaip tekstų rengyklė automatiškai sudaro rengiamo dokumento turinį. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 1.5. Automatinės teksto tvarkymo galimybės. Mokėti automatiškai sudaryti dokumento turinį.)

Mokiniai turėjo žinoti, kad į turinį įtraukiamas antraštes pirmiausia reikia sužymėti. Tekstus galima žymėti keliais būdais: pritaikyti antraščių stilius, sukurti vartotojo stilius (dar vienas būdas, kuris neįtrauktas į egzaminų programą, – sukurti dokumento struktūrą). Turbūt pats paprasčiausias antraščių tekstų žymėjimo būdas – formatuoti dokumento tekstą pritaikant rengyklėse įtaisytus antraščių stilius.

Daugiau nei trečdalis mokinių neatidžiai perskaitė klausimo formuluotę ir pasirinko klaidingą B atsakymą, maždaug kas penktas mokinys neskiria pastraipos ir šrifto stiliaus.

Naudojimosi skaičiuokle temos klausimai

5. Kokį rezultatą suskaičiuos skaičiuoklės langelyje A6 įrašyta formulė?



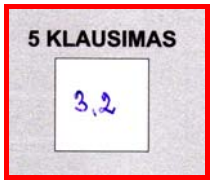
Atsakymas

(2 taškai)

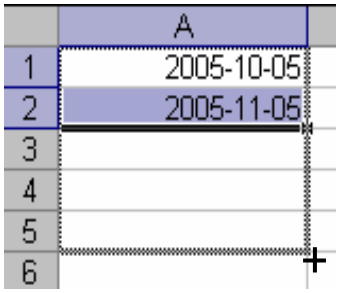
Tai atvirojo tipo klausimas, kuriuo tikrinamas skaičiuoklės aritmetinio vidurkio skaičiavimo funkcijos AVERAGE praktinis taikymas. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 2.4. Formulės ir funkcijos. Mokėti naudoti funkcijas: SUM, AVERAGE, MIN, MAX, IF, NOT, AND, OR.)

Į langelių bloką, kurio vidurkį turėjo suskaičiuoti funkcija, buvo įtrauktas tuščias langelis. Teisingai atsakiusieji į šį klausimą mokiniai turėjo žinoti, kad funkcija AVERAGE sudeda skaitines reikšmes ir sumą padalija iš reikšmių skaičiaus, tačiau ignoruoja tekstą, logines reikšmes ir tuščius langelius.

Remiantis statistine egzamino rezultatų analize galima teigti, kad didesnė dalis mokinių žino, kaip skaičiuojamas argumentų aritmetinis vidurkis taikant skaičiuoklės funkciją AVERAGE. Tačiau daugiau nei trečdalis mokinių nemoka pritaikyti šios funkcijos, iš jų penktadalis nemato skirtumo tarp tuščio langelio bloke ir langelio su nuline reikšme.

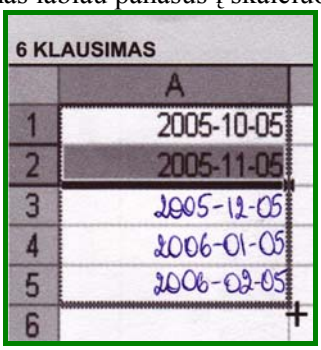


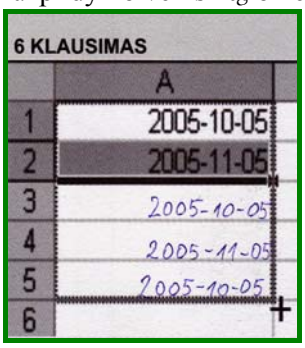
6. Skaičiuoklės stulpelio langeliuose surinktos dvi datos taisyklingu lietuvišku formatu. Pažymėjus šiuos du langelius, reikšmės nukopijuotos dar per tris langelius žemyn, kaip pavaizduota paveikslėlyje. Į skaičiuoklės langelių A3:A5 bloką įrašykite reikšmes, kurias matysime po kopijavimo.



(2 taškai)

Tai taip pat atvirojo tipo klausimas, kuriuo tikrinami mokinių gebėjimai dirbti su datos skaičių formatais, atlikti kopijavimo veiksmą skaičiuoklėje. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 2.2. Darbas su narveliais, eilutėmis, stulpeliais. Mokėti dirbti su narveliais: pažymėti, kopijuoti, perkelti, pašalinti.) Klausimo formuluotė nėra tiksli, nes aprašomas veiksmas labiau panašus į skaičiuoklės automatinio užpildymo veiksmą, o ne į kopijavimą.





Tačiau statistinė egzamino rezultatų analizė rodo, kad šis klausimas mokiniams nebuvo sunkus ir nesukėlė neaiškumų. Į jį teisingai atsakė daugiau nei 82 proc. mokinių.

Ateityje teste reikėtų vengti tokių dviprasmių formuluočių.

7. Skaičiuoklės langelyje įrašyta formulė =SUM($A$3:$D$14). Kokio tipo koordinatės naudojamos šioje formulėje?

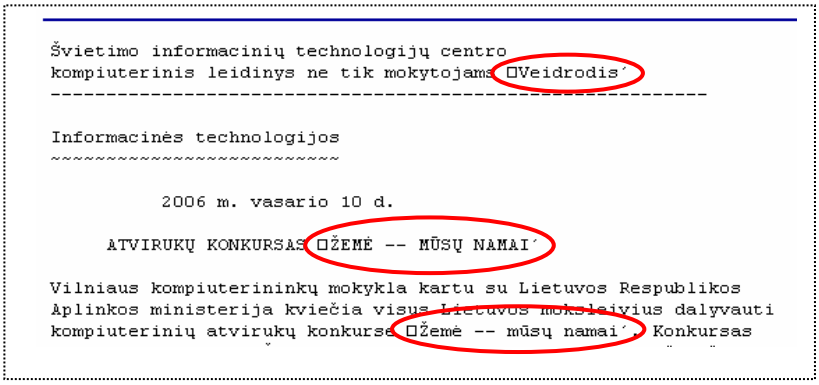
A Absoliučiosios koordinatės.
B Santykinės koordinatės.
C Tai nėra langelių bloko koordinatės.
D Absoliučiosios ir santykinės koordinatės. *(2 taškai)*

Šis klausimas reikalavo žinoti, kaip skaičiuoklėje vaizduojamos absoliučiosios koordinatės. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 2.4. Formulės ir funkcijos. Paaiškinti narvelių santykines ir absoliučiąsias koordinates.)

Dauguma mokinių į šį klausimą atsakė teisingai. Tai rodo klausimo sunkumas (65,38 proc.). Tačiau daugiau nei trečdalis mokinių į šį klausimą atsakė neteisingai: vieni mano, kad formulėje naudojamos santykinės koordinatės, kiti – kad ir absoliučiosios ir santykinės koordinatės, o kas dvidešimtas neturi jokio supratimo apie koordinates.

Žiniatinklio ir elektroninio pašto temos klausimai

11. Atvėrę žiniatinklio paštu gautą laišką matome tokį vaizdą:


Švietimo informacinių technologijų centro
kompiuterinis leidinys ne tik mokytojams ▯Veidrodis´

Informacinės technologijos
~~~~~~~~~~~~~~~~~~~~~~~~~~

2006 m. vasario 10 d.

ATVIRUKŲ KONKURSAS ▯ŽEMĖ -- MŪSŲ NAMAI´

Vilniaus kompiuterininkų mokykla kartu su Lietuvos Respublikos
Aplinkos ministerija kviečia visus Lietuvos moksleivius dalyvauti
kompiuterinių atvirukų konkurse ▯Žemė -- mūsų namai´. Konkursas

Žinoma, kad laiškas buvo parašytas tvarkingai, be klaidų. Laiško tekste visos raidės matomos taisyklingos, o vietoj kabučių matomi kitokie simboliai (pažymėtos vietos). Kokia to priežastis?

A Siunčiant laišką pasirinkta nelietuviška koduotė.
B Siunčiant laišką pasirinkta viena lietuviško standarto koduotė, o atveriant gautą laišką – kita lietuviška koduotė.
C Laiške negalima rašyti kabučių – jų niekada nenusiųs teisingai.
D Laiškas siųstas nekoduotas, dėl to kai kurie simboliai „nutrupėjo“.

(2 taškai)

Šiuo klausimu buvo tikrinama, kaip mokiniai žino lietuviškų rašmenų perdavimo elektroniniu paštu problemas. (Informacinių technologijų brandos egzaminų programos 1 priedas, Dalykiniai turinio reikalavimai: 3.5. Pagrindinės elektroninio pašto ypatybės ir nuostatos. Žinoti dviejų pagrindinių elektroninio pašto sistemų (elektroninio pašto programų ir žiniatinklio elektroninio pašto tarnybų) pranašumus. Įvardyti lietuviškų rašmenų perdavimo elektroniniu paštu problemas, žinoti jų sprendimo principus. Mokėti pasirinkti tinkamą elektroninio pašto programą ir nustatyti tinkamą lietuvių kalbai koduotę.)

Beveik kas dešimtas mokinys pasirinko klaidingą A variantą, nes nepastebėjo, kad laiško tekste yra matomų ir teisingų lietuviškų rašmenų Ž, Ė, Ų, Ū. Todėl teiginys, kad siunčiant laišką pasirinkta nelietuviška koduotė, nėra teisingas. Taip pat neteisūs yra ir mokiniai, kurie teigia, kad laiške išvis negalima rašyti kabučių arba, kad šis laiškas siųstas nekoduotas.

Gana didelis teisingai atsakiusių mokinių skaičius (daugiau kaip 78 proc.) rodo, kad jie yra susidūrę su lietuviškų rašmenų perdavimo elektroniniu paštu problemomis ir moka jas spręsti.

Pateikčių rengimo temos klausimas

10. Lina ruošia skaidres pranešimui pateikčių rengimo programa. Ji pasirinko skaidrės dizainą, tačiau tekstą nori rašyti kita, nei numatyta dizaine, spalva. Kuris teiginys teisingas?

A Pasirinkus dizainą teksto spalvų keisti nebegalima – reikia tenkintis numatytomis.
B Kiekviename dizaine tekstui spalvinti skirta tik tam tikra suderinta spalvų paletė – galima rinktis tik tos paletės spalvą.
C Lina gali laisvai pasirinkti bet kurią spalvą tekstui spalvinti.
D Lina gali laisvai rinktis tik fono spalvą – teksto spalvos keisti negalima.

(2 taškai)

Šiuo klausimu tikrinamos mokinių žinios apie skaidrių elementų formatavimo galimybes pateikčių rengyklėje. (Informacinių technologijų brandos egzaminų programos 1 priedas, Dalykiniai turinio reikalavimai: 4.2. Pateikties kūrimas ir formatavimas. Mokėti rinkti tekstą pateiktyje; pakartoti, perkelti bei pašalinti tekstą. Mokėti pakeisti teksto šriftą, simbolių dydį, stilių, lygiuoti tekstą, keisti pastraipų
~~~~~~~~~~~~~~~~~~~~~~~~~~

(išvardijimų, sąrašų) ženklinimą).

Mokiniai turi žinoti, kad pateikties skaidrei sukurti pasirinkus vieną iš daugiau nei trisdešimties siūlomų skaidrių dizaino šablonų su numatyta suderinta spalvų palete, vėliau galima laisvai keisti ne tik teksto spalvą ir kitus parametrus, bet ir kitų skaidrės elementų parametrus.

Remiantis statistine egzamino rezultatų analize galima teigti, kad tai labai lengvas, nedidelės skiriamosios gebos klausimas. Į jį teisingai atsakė daigiau kaip 92 proc. mokinių. Didelį teisingų atsakymų skaičių lėmė dažnas praktinis įvairių tekstų rašymas ir formatavimas ne tik pateikčių rengykle, bet ir kitomis rengyklėmis.

Į šį klausimą teisingai atsakė mokiniai, kurie turėjo ne tik teorinių žinių, bet ir praktinių darbo pateikčių rengykle įgūdžių. Tokių klausimų, į kuriuos atsakydamas mokinys turėtų parodyti ne tik teorines žinias, bet ir praktinius įgūdžius, kuriuos jis įgijo dirbdamas kompiuteriu, testo užduotyje turėtų būti daugiau.

Socialinių ir etinių aspektų temos klausimas

9. Internete sukurtos valstybės valdymo institucijų priemonės, skirtos viešosioms paslaugoms, kultūrai, turizmui, transportui ir kelionėms, aplinkos apsaugos paslaugoms ir paslaugoms, išplečiančioms piliečių dalyvavimą demokratiniuose procesuose, valdyti, vadinamos:

A elektroniniu verslu;
B elektroniniu mokymu;
C elektronine valdžia;
D elektronine prekyba.

(2 taškai)

Šiuo klausimu tikrinama, ar mokiniai supranta elektroninės valdžios sąvoką. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 5.1. Informacinė visuomenė, kompiuterio įtaka visuomenei, žmogui. Žinoti elektroninės demokratijos ir elektroninės valdžios sąvokas, gebėti naudotis jos teikiamomis galimybėmis.)

Mokinių atsakymų analizė parodė, kad į šį klausimą teisingai atsakė dauguma mokinių (70,70 proc.), tačiau apie 17 proc. mokinių supainiojo šią sąvoką su elektroniniu verslu. Taip yra todėl, kad socialinių, teisinių ir etinių aspektų temoms mokykloje skiriama per mažai dėmesio. Šis skyrius apima daug teorinės medžiagos, kurią mokiniai sunkiai suvokia.

12. Kompiuterių virusai veikia tam tikrais būdais, pavyzdžiui, šalina kompiuteryje esančius failus. Įvardykite dar du kompiuterių virusų veikimo būdus.

1.

2.

(2 taškai)

Tai atvirojo tipo klausimas, kuriuo tikrinamos mokinių žinios apie kompiuterių virusus, jų veikimo būdus. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 5.3. Duomenų apsauga ir autorių teisės. Išvardyti keletą virusų veiklos ir jų plitimo scenarijų.)

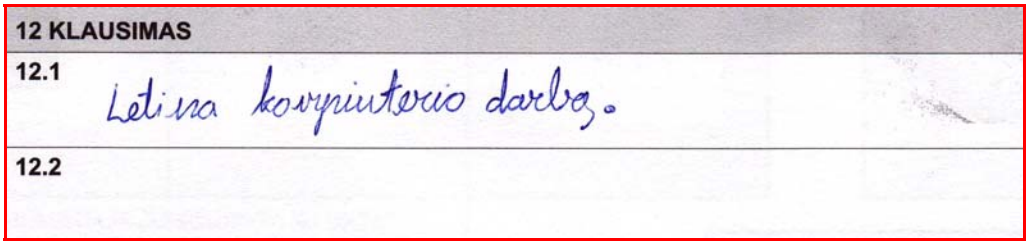
12 KLAUSIMAS

12.1 Virusas gali rodyti įvairius pranešimus, prot. gaisus.

12.2 Virusai gali daugyti savo kopijas ir išsiųsti jas el. pašto adresais, nustatais, pvz., „Outlook" adresų knygelėje.

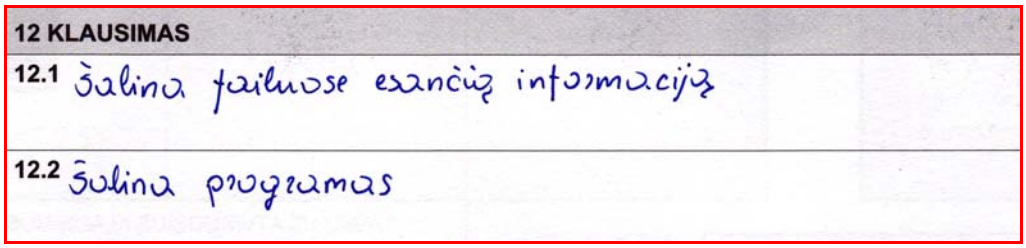
Mokiniams šis klausimas nebuvo sunkus. Į jį teisingai atsakė daugiau kaip 92 proc. mokinių. Tačiau

beveik kas dešimtas mokinys sugebėjo aprašyti tik vieną virusų veikimo būdą arba tą patį skirtingai aprašė du kartus. Kai kurie mokiniai pakartojo klausimo formuluotėje paminėtą virusų veikimo būdą.


12 KLAUSIMAS

12.1 Letina kompiuterio darbą.

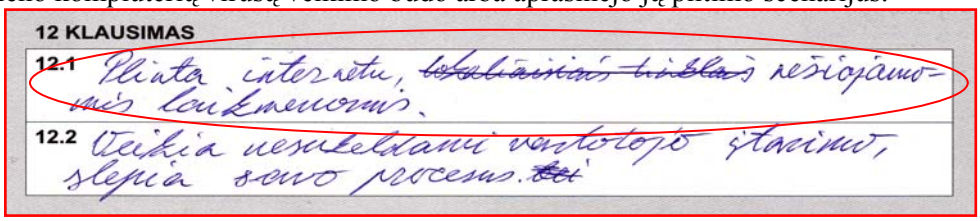
12.2


12 KLAUSIMAS

12.1 Šalina failuose esančią informaciją

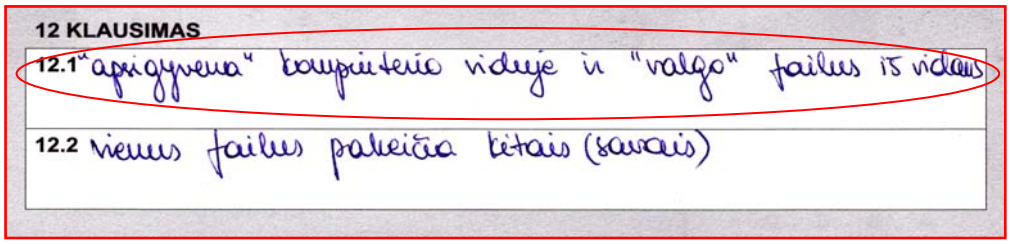
12.2 Šalina programas

Apie 3 proc. mokinių, laikiusių informacinių technologijų valstybinį brandos egzaminą, negalėjo paminėti nė vieno kompiuterių virusų veikimo būdo arba aprašinėjo jų plitimo scenarijus.


12 KLAUSIMAS

12.1 Plinta internetu, ~~lokaliniais tinklais~~ nešiojamosios laikmenomis.

12.2 Veikia nesankcionuotai vartotojo įtaisimo, slepia savo procesus. ~~Ši~~


12 KLAUSIMAS

12.1 "apgyvena" kompiuterio viduje ir "valgo" failus iš vidaus

12.2 viemus failus pakeičia kitais (savais)

Buvo ir labai originalių atsakymų.

Programavimo temų klausimai

13. Kuriai ciklo sąlygai esant bus spausdinamas skaičius 8? Pažymėkite teisingą atsakymą.

```
program Te1;
var  n, k : integer;
begin
  n := 2; k := 3;
  while ??? do
    begin
      k := k + 2;
      n := 2 * n;
    end;
  Write (n);
end.
```

A	k < 10
B	n < k
C	k < n
D	n < 10

(2 taškai)

Šiuo klausimu tikrinama, ar mokiniai supranta, kaip Paskalio kalboje realizuojamas ciklas `while`, sumos/sandaugos kaupimo algoritmas. (Informacinių technologijų brandos egzaminų programos 1 priedas, Dalykiniai turinio reikalavimai: 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą).)

Mokinių atsakymų analizė parodė, kad į šį klausimą teisingai atsakė dauguma mokinių (74,66 proc.), tačiau apie 25 proc. mokinių nesugebėjo parinkti teisingos sąlygos.

14. Kokias reikšmes įgis kintamieji `n` ir `k` įvykdžius programą `Te2`? Pažymėkite teisingą atsakymą.

```
program Te2;
var n, k : integer;
begin
  n := 4; k := 2;
  while (n < 10) or (k < n) do
    begin
      n := 2 * k;
      k := 2 * n;
    end;
end.
```

	n	k
A	4	8
B	32	16
C	16	32
D	8	4

(2 taškai)

Šiuo klausimu taip pat tikrinamas ciklo `while` ir sumos/sandaugos kaupimo algoritmo Paskalio kalboje realizavimo supratimas, bet ciklo sąlygą yra sudėtingesnė, joje panaudota loginė operacija `or`. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą); 6.3. Kintamųjų sąvoka. Mokėti sudaryti aritmetinius ir loginius reiškinius.)

Mokinių atsakymų analizė parodė, kad į šį klausimą teisingai atsakė daugiau nei pusė mokinių (58,93 proc.). Maždaug kas trečias mokinys (33,55 proc.) supainiojo loginės operacijos `or` ir `and` rezultatus ir pasirinko klaidingą atsakymo A variantą, kiti mokiniai taip pat nesugebėjo teisingai atsakyti į šį klausimą.

15. Kokias reikšmes įgis kintamieji `t` ir `n` įvykdžius programą `Te3`? Pažymėkite teisingą atsakymą.

```
program Te3;
var n, k : integer;
    t : boolean;
begin
  t := TRUE; n := 13;
  for k := 5 to 8 do
    begin
      if t then n := n + k
           else n := n - k;
      t := not t;
    end;
end.
```

	t	n
A	TRUE	19
B	TRUE	11
C	FALSE	19
D	FALSE	11

(2 taškai)

Šis klausimas reikalavo žinoti loginio kintamojo sąvoką, suprasti žinomo kartojimo ciklo `for` veikimo principą ir paprasto sąlyginio sakinio struktūrą. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.3. Kintamųjų sąvoka. Apibūdinti sveikuosius, realiuosius, simbolinius, loginius kintamuosius ir veiksmus su jais; 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo

sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą). Skirti nežinomo ir žinomo kartojimų skaičiaus ciklus, žinoti jų taikymo ypatybes.)

Daugiau nei pusė mokinių (51,63 proc.) į šį klausimą atsakė teisingai. Nemaža dalis mokinių (22,16 proc.) suklydo, nes nepakeitė loginio kintamojo `t` reikšmės, nors kintamojo `n` reikšmę suskaičiavo teisingai. 14, 69 proc. mokinių nevykdė ciklo `for` paskutinį kartą ir pasirinko klaidingą atsakymo D variantą.

16. Pažymėkite atsakymą, kuris būtų gautas įvykdžius programą `Te4`.

```
program Te4;
type TMas = array[1..3] of integer;
var A : TMas; n : integer;
    i : integer;
begin
  n := 3;  A[1] := 13;
  for i := 2 to n do
     if i mod 2 = 0
          then A[i] := A[i-1] + i
          else A[i] := 2 * A[i-1];
end.
```

	A[1]	A[2]	A[3]
A	13	26	28
B	13	28	30
C	13	14	28
D	13	15	30

(2 taškai)

Į šį klausimą teisingai atsakė mokiniai, kurie žino masyvo duomenų tipą, moka atlikti veiksmus su masyvo elementų reikšmėmis, supranta žinomo kartojimo ciklo `for` veikimo principą, paprasto sąlyginio sakinio struktūrą ir moka atlikti sveikųjų skaičių dalybą. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą); 6.6. Masyvo duomenų tipas ir masyvo duomenų tipo kintamieji. Apibūdinti masyvo sąvoką. Mokėti sukurti masyvo duomenų tipą. Gebėti atlikti veiksmus su masyvo elementais: įvesti ir išvesti duomenis, atlikti paiešką, vartoti reiškiniuose.)

Mokinių atsakymų analizė parodė, kad į šį klausimą teisingai atsakė dauguma mokinių (76,46 proc.). Apytiksliai kas dešimtas mokinys (12,11 proc.) suklydo skaičiuodamas antro ir trečio masyvo elementų reikšmes, kiti mokiniai neteisingai suskaičiavo sąlyginio sakinio sąlygos reikšmę ir todėl supainiojo veiksmų atlikimo tvarką.

17. Pažymėkite atsakymą, kuris būtų gautas įvykdžius programą `Te5`.

```
program Te5;
{-------------------------------------------------}
procedure Pakeisti(var x : integer; y : integer);
begin
  y := (x + y) * 2;
  x := x + y;
end;
{-------------------------------------------------}
var a, b : integer;
begin
  a := 4;  b := 2;
  Pakeisti(a, b);
  Pakeisti(b, a);
  Pakeisti(a, a + b);
  Write(a, '   ', b);
end.
```

A	864 748
B	416 124
C	156 38
D	4 2

(2 taškai)

Šiuo klausimu tikrinama, ar mokiniai supranta procedūros konstrukciją, skiria procedūros parametrus. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.5. Procedūros ir funkcijos. Suprasti uždavinio skaidymo į dalis pranašumus. Mokėti skaidyti uždavinį į dalis, jas užrašyti procedūromis ir funkcijomis. Skirti procedūros ir funkcijos sąvokas.

Paaiškinti jų vartojimo būdus, gebėti taikyti uždaviniuose. Suprasti parametrų sąvoką. Žinoti skirtumą tarp parametrų kintamųjų ir parametrų reikšmių. Žinoti vardų galiojimo sritis.)

Mokinių atsakymų analizė parodė, kad šis klausimas mokiniams buvo gana sunkus. Į jį teisingai atsakė tik 33,93 proc. mokinių. Galbūt mokiniams renkantis atsakymus turėjo įtakos procedūros parametrų užrašymo būdas, kai rezultatai aprašyti prieš argumentus.

18. Pažymėkite atsakymą, kuris būtų gautas įvykdžius programą `Te6`, jeigu tekstiniame faile `Te6.txt` užrašyta tokia skaičių eilutė: `13 16 -5 9 4`

```
program Te6;
var a, b, c, d : integer;
    t : boolean;
    F : text;
begin
  Assign(F, 'Te6.txt'); Reset(F);
  Read(F, a, b);
  t := a > b;
  if t then Read(F, c, d, a)
       else Read(F, a, c, d);
  Write(a:4, b:4, c:4, d:4, t:8);
  Close(F);
end.
```

A	-5	16	9	4	FALSE
B	13	16	-9	9	TRUE
C	-5	16	9	4	TRUE
D	13	16	-5	9	FALSE

(2 taškai)

Šis klausimas apibendrina kelių programavimo temų žinias ir gebėjimus. Jis skirtas mokinių darbo su tekstiniais failais įgūdžiams tikrinti. Mokiniai turėjo žinoti, kaip skaitomi duomenys iš tekstinio failo. Taip pat klausimo sąlygoje yra loginis kintamasis, sąlyginis sakinys ir spausdinimo procedūrą `Write` su duomenų išvedimo formatais. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai:, 6.3. Kintamųjų sąvoka. Apibūdinti sveikuosius, realiuosius, simbolinius, loginius kintamuosius ir veiksmus su jais; 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą); 6.8. Tekstinės bylos. Suprasti tekstinių bylų paskirtį. Suprasti ryšį tarp duomenų surašymo byloje tvarkos ir duomenų skaitymo iš bylos programos sakinių. Mokėti skaityti duomenis iš tekstinių bylų ir rašyti juos į bylas, kai kintamieji yra paprastieji, masyvo bei įrašo tipo.)

Į šį klausimą teisingai atsakė daugiau kaip 60 proc. mokinių. Statistinė analizė rodo, kad klausimo skiriamoji geba yra didelė (daugiau nei 78 proc.), todėl galima teigti, kad šis klausimas labai gerai diferencijavo mokinius. Maždaug kas trečias mokinys (30,33 proc.) suklydo priskirdamas tekstiniame faile užrašytus skaičius konkretiems kintamiesiems.

19. Pažymėkite atsakymą, kuris būtų gautas įvykdžius programą `Te7`, jeigu tekstiniame faile `Te7.txt` užrašyta tokia skaičių eilutė: `7 8 9 1 -5 8`

```
program Te7;
var a, b, c, d : integer;
    F : text;
begin
  Assign(F, 'Te7.txt'); Reset(F);
  Read(F, a, b, c);
  d := b + c;
  Read(F, b, c);
  if (a > b) and (a > c)
    then if d < c
         then d := d + b + c
         else d := d - b - c
    else d := 2 * (a + b + c);
  Write(a:5, b:5, c:5, d:5);
  Close(F);
end.
```

A	7	1	-5	11
B	7	8	1	20
C	7	9	-5	21
D	7	1	-5	21

(2 taškai)

Šis klausimas yra 18-ojo klausimo sudėtingesnis variantas, juo tikrinamos taip pat kelių programavimo temų žinios ir gebėjimai. Šiuo klausimu siekiama įvertinti mokinių darbo su tekstiniais failais įgūdžius. Sąlyginio sakinio sąlyga yra sudėtingesnė, joje panaudota loginė operacija `and`. Pateiktoje programoje naudojama sudėtingo sąlyginio sakinio konstrukcija `if`-as `if`-e ir spausdinimo procedūra `Write` su duomenų išvedimo formatais. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.3. Kintamųjų sąvoka. Apibūdinti sveikuosius, realiuosius, simbolinius, loginius kintamuosius ir veiksmus su jais. Mokėti sudaryti aritmetinius ir loginius reiškinius; 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą); 6.8. Tekstinės bylos. Suprasti tekstinių bylų paskirtį. Suprasti ryšį tarp duomenų surašymo byloje tvarkos ir duomenų skaitymo iš bylos programos sakinių. Mokėti skaityti duomenis iš tekstinių bylų ir rašyti juos į bylas, kai kintamieji yra paprastieji, masyvo bei įrašo tipo.)

Į šį klausimą teisingai atsakė apie 70 proc. mokinių. Statistinė analizė rodo, kad šio klausimo skiriamoji geba yra didelė (daugiau kaip 69 proc.), todėl galima teigti, kad jis labai gerai diferencijavo mokinius. Kiti mokiniai klydo priskirdami tekstiniame faile užrašytus skaičius konkretiems kintamiesiems, neteisingai atliko skaičiavimus arba supainiojo loginės operacijos `and` ir `or` rezultatus.

21. Nustatykite, **ką** ir **kaip** spausdins programos `Te9` fragmentas, jeigu:

- pirmieji masyvo `A` elementai yra skaičiai `13 -8 6 7`
- pirmieji masyvo `B` elementai yra skaičiai `9 -5 -8 -10`

```
program Te9;
  ...
type TMas = array[1..50] of integer;
  ...
{-------------------------------------------------------}
{Funkcija Suma skaičiuoja masyvo A elementų reikšmių sumą.}
{n - masyvo A elementų skaičius.}
function Suma(var A : TMas; n : integer): integer;
var s, i : integer;
begin
  s := 0;
  for i := 1 to n do
    s := s + A[i];
  Suma := s;
end;
{-------------------------------------------------------}
var A, B : TMas;  na, nb : integer;
begin
  ...
  na := 4; nb = 4;
  WriteLn(Suma(A, na):3);
  na := na + 1;  A[na] := Suma(B, nb);
  nb := nb + 1;  B[nb] := Suma(A, na);
  WriteLn(A[na]:5, B[nb]:5);
  WriteLn(Suma(A, na) + Suma(B, nb));
end.
```

Atsakymą įrašykite į pateiktą lentelę, kurios vienas langelis atitinka vieną simbolio poziciją.

Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė												
2 eilutė												
3 eilutė												

(4 taškai)

Po 1 tašką skiriama už kiekvienoje eilutėje įrašytas teisingas reikšmes ir 1 taškas skiriamas už teisingą reikšmių surašymą pozicijose.

Šis klausimas apibendrina kelių programavimo temų žinias ir gebėjimus. Jis skirtas tikrinti, kaip mokiniai dirba su masyvais, kaip supranta funkcijos konstrukciją ir ją naudoja programoje, kaip supranta žinomo kartojimo ciklo `for` veikimo principą, kaip mokiniai geba spausdinti reikšmes su formatais ir kaip supranta reikšmių sumos skaičiavimo algoritmą. (Informacinių technologijų brandos egzaminų programos 1 priedas. Dalykiniai turinio reikalavimai: 6.4. Pagrindiniai programavimo kalbos sakiniai. Mokėti taikyti uždaviniuose pagrindinius valdymo sakinius bei struktūras: priskyrimo sakinį, nuoseklią veiksmų seką (sudėtinį sakinį), veiksmų šakojimą (sąlyginį sakinį), veiksmų kartojimą (ciklą); 6.5. Procedūros ir funkcijos. Suprasti uždavinio skaidymo į dalis pranašumus. Mokėti skaidyti uždavinį į dalis, jas užrašyti procedūromis ir funkcijomis. Skirti procedūros ir funkcijos sąvokas. Paaiškinti jų vartojimo būdus, gebėti taikyti uždaviniuose. Suprasti parametrų sąvoką. Žinoti skirtumą tarp parametrų kintamųjų ir parametrų reikšmių. Žinoti vardų galiojimo sritis; 6.6. Masyvo duomenų tipas ir masyvo duomenų tipo kintamieji. Apibūdinti masyvo sąvoką. Mokėti sukurti masyvo duomenų tipą. Mokėti aprašyti masyvo tipo kintamuosius. Gebėti atlikti veiksmus su masyvo elementais: įvesti ir išvesti duomenis, atlikti paiešką, vartoti reiškiniuose.)

Norėdami teisingai atsakyti į šį klausimą, mokiniai turėjo ne tik rasti programos fragmento kintamųjų teisingas reikšmes, bet ir surašyti jas tinkamose menamo ekrano pozicijose. Visus galimus 4 taškus už šį klausimą surinko tik kas penktas egzaminą laikęs mokinys.

21 KLAUSIMAS

Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė		1	8									
2 eilutė			-	1	4					4		
3 eilutė	-	6										

15,38 proc. mokinių teisingai atliko skaičiavimus, bet suklydo surašydami reikšmes pozicijose. Šie mokiniai surinko po tris taškus.

21 KLAUSIMAS

Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė	1	8										
2 eilutė	-	1	4			4						
3 eilutė	-	6										

Beveik 12,5 proc. mokinių tik iš dalies teisingai atliko skaičiavimus ir/ar suklydo surašydami reikšmes pozicijose. Šie mokiniai surinko po du taškus.

21 KLAUSIMAS

Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė				1	8							
2 eilutė					4						-1	0
3 eilutė	-	6										

21 KLAUSIMAS

Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė				18								
2 eilutė						-14						4
3 eilutė	-10											

Apie 12 proc. mokinių teisingai atliko tik vienos reikšmės skaičiavimus, jie gavo po vieną tašką.

21 KLAUSIMAS												
Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė	18											
2 eilutė	18	-14										
3 eilutė	4											

Visai neatlikinėjo arba pateikė originalius sprendimus daugiau kaip 41 proc. mokinių.

21 KLAUSIMAS												
Pozicijos	1	2	3	4	5	6	7	8	9	10	11	12
1 eilutė	12	11	10	9	8	7	6	5	4	3	2	1
2 eilutė	7	8	9	10	11	12	6	5	4	3	2	1
3 eilutė	12	11	10	9	8	7	1	2	3	4	5	6

Nors šio klausimo sunkumas ne itin didelis (39,67 proc.), bet didelė skiriamoji geba (76,15 proc.). Taigi galima teigti, kad šis klausimas labai gerai diferencijavo mokinius.

PRAKTINIŲ UŽDUOČIŲ REZULTATŲ KOKYBINĖ ANALIZĖ

Praktinių užduočių tikslai

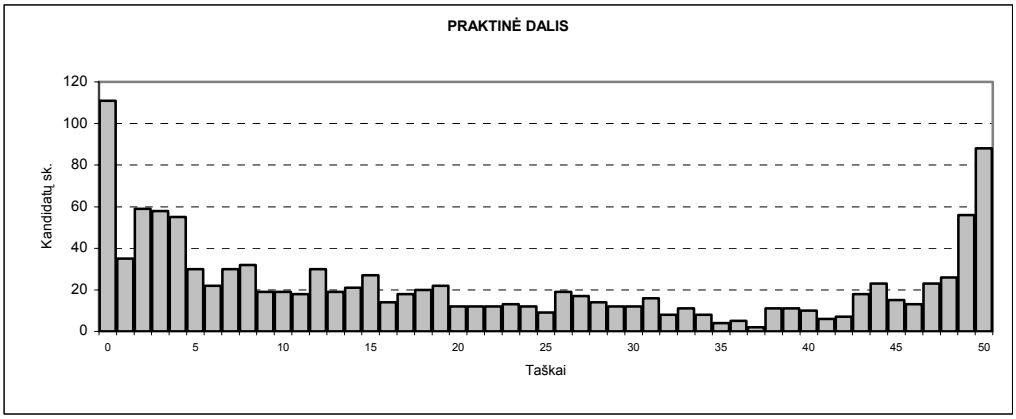
Pirmosios užduoties (U1) tikslai. Patikrinti, kaip mokiniai dirba su tekstiniais failais (skaitymas, rašymas), ar moka sudaryti sumos kaupimo algoritmą, ar geba parašyti paprastą programą darbui su skaičiais.

Antrosios užduoties (U2) tikslai. Patikrinti, kaip mokiniai dirba su tekstiniais failais, kai duomenys įrašo tipo, ar geba praktiškai panaudoti masyvą su įrašo tipo duomenimis. Įvertinti, kaip mokiniai geba naudoti programose funkcijas ir procedūras, ar moka sudaryti sumos kaupimo algoritmą.

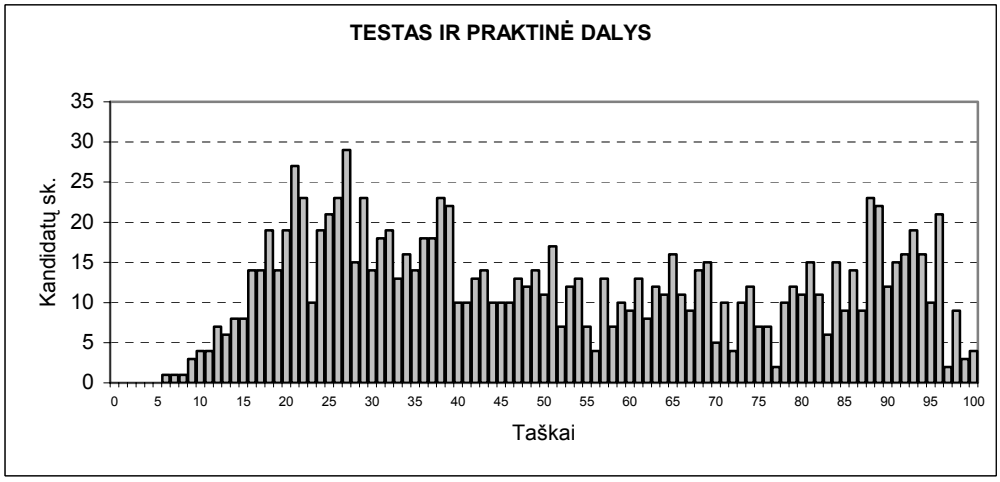
1 lentelė. Programavimo mokymo turinys.

Algoritmai	Duomenų struktūros	Programavimo kalba (Paskalis)
Sumos (sandaugos, kiekio, aritmetinio vidurkio) skaičiavimai. Didžiausios (mažiausios) reikšmės paieška. Duomenų skaitymas iš failo ir rašymas į failą (duomenų keltis). Duomenų rikiavimas. Gebėjimas pritaikyti algoritmus konkrečioms duomenų struktūroms.	Sveikieji ir realieji skaičiai, simbolio tipas, loginis tipas, eilutė (`integer`, `real`, `char`, `boolean`, `string`). Tekstinis failas (`Text`). Vienmatis masyvas (`array`). Įrašas (`record`). Gebėjimas kurti nesudėtingas duomenų struktūras.	Programos struktūra. Komentarai. Kintamieji. Priskyrimo operacija, sakinys. Santykio ir loginės operacijos, sąlygos ir ciklo sakiniai. Blokas `begin ... end`; Procedūra ir funkcija. Parametrų ir argumentų sąrašai. Standartinės matematinės procedūros ir funkcijos. Procedūros ir funkcijos darbui su failais.
Programavimo aplinka. Struktūrinio (procedūrinio) programavimo technologija. Testavimas. Programų dokumentavimas. Dialogo organizavimo būdai. Programų rašymas (kultūra, stilius).		

Žinių sritis	Dėmesio paskirstymas(%)
1. Duomenų struktūrų analizė ir kūrimas	20
2. Duomenų apdorojimo algoritmai	40
3. Programavimo kalbos konstrukcijos	10
4. Programavimo kalbos priemonės ir aplinkos	10
5. Programavimo technologijos	10
6. Programų derinimas, testavimas	5
7. Programavimo stilius, kultūra	5



1 pav. 2006 m. informacinių technologijų valstybinį brandos egzaminą laikiusių kandidatų praktinių užduočių taškų pasiskirstymas



2 pav. 2006 m. informacinių technologijų valstybinį brandos egzaminą laikiusių kandidatų gautų taškų pasiskirstymas

Pateikiame pagrindinės sesijos valstybinio brandos egzamino praktinės užduoties **programavimo srities temų** pasiskirstymą.

2 lentelė. Programavimo srities praktinės užduoties temų pasiskirstymas.

Užduoties numeris	**Programos**	
	pirma	**antra**
Užduoties vertė taškais	**17**	**33**
Priskyrimo operacija, sakinys, paprasti kintamieji	+	+
Loginis kintamasis		
Ciklas `for`	+	+
Ciklas `while`		
Blokas `begin ... end;`	+	+
Sąlygos sakinys `if`		
Loginės operacijos (`and`, `or`, `not`)		
Programos struktūra	+	+
Spausdinimas ekrane (`Write`, `WriteLn` procedūros)		
Skaitymas iš tekstinio failo	+	+
Masyvo duomenų tipas, masyvo tipo kintamasis	+	+
Įrašo (`record`) duomenų tipas		+
Procedūra, parametrų sąrašas, kreipinys	+	+
Funkcija, parametrų sąrašas, kreipinys	+	+
Skaičių suma, sandauga (algoritmas)	+	+
Rašymas į failą	+	+

Pastaba. Patamsintais langeliais pažymėta tai, kas gali būti panaudota programoje, bet nebūtinai.

Pirmosios praktinės užduoties analizė

Elektros grandinės varžos skaičiavimas

Maksimali taškų suma – 17 taškų.

Iš fizikos kurso žinome, kad lygiagrečiai sujungtų laidininkų bendra varža skaičiuojama pagal formulę $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+...+\frac{1}{R_n}$, čia R – lygiagrečiai sujungtų laidininkų varža, R_1, R_2, …, R_n – atskirų laidininkų varžos.

Nuosekliai sujungtų laidininkų bendra varža skaičiuojama pagal formulę $R = R_1 + R_2 + \ldots + R_n$, čia R – nuosekliai sujungtų laidininkų bendra varža, R_1, R_2, …, R_n – atskirų laidininkų varžos.

Parašykite programą, kuri apskaičiuotų grandinės bendrą varžą, kai grandinę sudaro viena ar daugiau nuosekliai sujungtų grandinės dalių; kiekviena grandinės dalis sudaryta iš dviejų ar daugiau lygiagrečiai sujungtų žinomos varžos laidininkų.

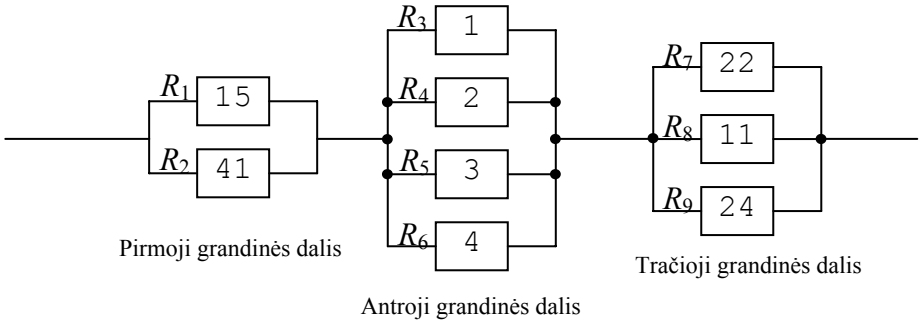
Programa turi skaityti duomenis iš tekstinio `Duom1.txt` failo. Pirmoje failo eilutėje įrašytas nuosekliai sujungtų grandinės dalių skaičius (ne daugiau kaip 100). Po to atskirose eilutėse surašyti grandinę sudarančių dalių duomenys: lygiagrečiai sujungtų laidininkų skaičius (ne daugiau kaip 50) ir jų varžų reikšmės.

Rezultatą – apskaičiuotą **grandinės bendrą varžą** – programa turi įrašyti į `Rez1.txt` failą dviejų ženklų po kablelio tikslumu.

Pavyzdys

Duomenų failas	Duomenų paaiškinimas	Rezultatų failas

`3`
`2 15 41`
`4 1 2 3 4`
`3 22 11 24`
	Nuosekliai sujungtų grandinės dalių skaičius:
	pirmoji grandinės dalis (dvi varžų R_1 ir R_2 reikšmės),
	antroji grandinės dalis (keturios varžų R_3, R_4, R_5 ir R_6 reikšmės),
	trečioji grandinės dalis (trys varžų R_7, R_8 ir R_9 reikšmės)
		`17.08`



Skaičiavimas:

$L_1=\frac{1}{R_1}+\frac{1}{R_2}$; $L_2=\frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_5}+\frac{1}{R_6}$; $L_3=\frac{1}{R_7}+\frac{1}{R_8}+\frac{1}{R_9}$ – grandinės dalių laidumų skaičiavimas,

$R=\frac{1}{L_1}+\frac{1}{L_2}+\frac{1}{L_3}$ – grandinės bendra varža.

Programą įrašykite į kietojo disko katalogą C:\Egzaminas, suteikdami failui vardą pagal šabloną *R01_1.pas* (raidė ir pirmieji du skaitmenys (*01*) – jūsų darbo vietos žymė, atskiras skaitmuo (*1*) – praktinės užduoties eilės numeris). Kitaip įvardyti failai nebus vertinami.

Gavę savo darbo vietos žyme pažymėtą diskelį, nepamirškite į jį nukopijuoti programos failo.

Programos vertinimas

Kas vertinama	Taškai
Tinkami programos darbo rezultatai	6
Duomenų skaitymas iš failo	2
Lygiagrečiai sujungtų varžų grandies bendros varžos reikšmės skaičiavimas	3
Visos grandinės varžos skaičiavimas	2
Programos struktūra	1
Rezultatų surašymas į failą	1
Programavimo stilius	2
Iš viso	17

Mokinių programų pavyzdžiai pateikti priede.

Pateiktų vertinimui programų analizėje daugiausia dėmesio skiriama panaudotoms duomenų struktūroms, nes nuo jų sudėtingumo priklauso programos apimtis, sudėtingumas, iš dalies algoritminė bazė. Nagrinėtos ne visos programos. Vertinimo metu vertintojai sudarė sąrašą programų, kurias verta nagrinėti kaip tipiškas (iš 997 programų, už kurias skirta bent po vieną tašką):

3 lentelė. Mokinių programose naudotos duomenų struktūros.

Mokinių programose naudotos duomenų struktūros	Vertintojų atrinktų mokinių programų	
	skaičius	apimtis baitais
Paprasti kintamieji	32	473–1675
Vienmatis masyvas	129	574–3123
Dvimatis masyvas	48	600–2767
Įrašo duomenų tipas	64	632–2883
„Abrakadabra“ tipo programos	38	86–1882
Iš viso	311	

Užduotį sudaro trys dalys:
- duomenų skaitymas;
- skaičiavimai;
- rezultato spausdinimas.

Duomenų skaitymas. Duomenims saugoti panaudoti duomenų tipai:
- paprastų kintamųjų rinkinys;
- vienmačiai masyvai;
- dvimatis masyvas;
- masyvai su įrašo tipo elementais.

Skaičiavimai:
- duomenų skaitymas ir skaičiavimai sujungti;
- visos trys dalys programuojamos atskirai.

Programos struktūra:
- visi veiksmai surašyti pagrindinėje programoje;
- procedūros rašomos kiekvienai iš dalių;
- mišri.

Užduoties sąlygoje nebuvo jokių programos struktūros, duomenų tipų ir užduoties algoritmo (sprendimo būdo) apribojimų. Rašydami programas, mokiniai galėjo pasirinkti tokią realizaciją, kuri jiems buvo priimtiniausia.

Šios užduoties programas vertinti buvo sunkiausia, nes jose buvo panaudota viskas, kas tik įmanoma, išskyrus dinamines duomenų struktūras, t. y. neapsiribota vien modulio turinio medžiaga. Vienai iš programų buvo panaudota rekursinė funkcija laidininko dalies varžai skaičiuoti. Programų apimtis nuo 40 eilučių iki kelių šimtų.

Tokios užduoties pranašumas tas, kad mokinys galėjo laisvai rinktis, galėjo rašyti programą pasinaudodamas tuo, ką geriausiai žino ir moka. Bloga yra tai, kad turėdami pasirinkimo laisvę mažiau įgudę mokiniai rinkosi sudėtingesnius ir painesnius sprendimo kelius. Jie nesugebėjo įvertinti pasirinktų

sprendimo kelių, suvokti jų sudėtingumo. Nedidelė dalis mokinių neturėjo elementaraus matematinio pasiruošimo, nesuvokė paprastų trupmenų.

Vertintojams šios užduoties programas buvo labai sudėtinga vertinti, ypač jeigu jose buvo nedidelių, nelogiškų klaidų. Labai sudėtinga buvo suvokti, kokį sprendimo kelią pasirinko mokinys, kodėl ir kaip vartoja pasirinktas duomenų struktūras ir atitinkamus kintamuosius. Gerai, kad vertintojai buvo įgudę programuotojai, žinantys sudėtingas duomenų struktūras ir jų panaudojimo subtilybes.

Ateityje siūlome nurodyti, kokias priemones galima naudoti užduoties programoje, ir neleisti naudoti priemonių, kurios nėra numatytos modulio turinyje. Tuomet dauguma mokinių rašys paprastesnes programas, jose liks mažiau klaidų, o vertintojai greičiau jas įvertins.

Šios užduoties sąlygoje reikėjo nurodyti, kad mokiniai nenaudotų jokių masyvų bei įrašų.

Lentelėje pateikiama statistika programų, kurios buvo įvertintos nuo 0 iki 8 taškų. Tuos taškus buvo galima surinkti, jeigu programa ir visiškai neteisinga.

4 lentelė. Mokinių programų įvertintų nuo 0 iki 8 taškų statistika.

Gauti taškai	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Programos tekstas (kiekis)	167	25	26	47	37	58	67	45	38
IT testas (maks. taškų)	23	19	20	23	20	23	21	23	23
Programavimo testas (kai IT maks. taškų)	8	8	7	12	3	10	14	12	11
Iš viso (IT + PrT + U1)	31	28	29	38	27	38	41	42	42

Neteisingos programos

Tai logiškai menkai susietų sakinių rinkinys (žr. programas priede P1, P2, P3, P4, P5): matematinė užduoties dalis (formulės) buvo nesuprasta; keletas mokinių keitė formules savomis, kurios buvo neteisingos.

Nesuprasta uždavinio esmė. Programuotas konkretus užduoties variantas, kuris buvo pateiktas užduotyje kaip kontrolinis pavyzdys. Tai trys lygiagrečiai sujungtų varžų laidininko dalys: 2 varžos, 4 varžos ir 3 varžos. Varžų reikšmėms saugoti buvo skelbiami 9 kintamieji. Šiems kintamiesiems reikšmės buvo:

– suteikiamos priskyrimo sakiniais – konkrečios iš užduoties pavyzdžio;
– skaitomos dialogu ir įvedamos klaviatūra;
– skaitomos iš failo. Jeigu faile būtų užduoties pavyzdžio kontroliniai duomenys, tuomet rezultatas būtų sutapęs su pateiktu užduotyje kontrolinės užduoties atsakymu.

Taškai buvo skiriami už atskirus punktus, kurie atitiko vertinimo kriterijus: failų paruošimas darbui, kintamųjų aprašymas, programavimo kultūra ir pan.

Programos, kuriose panaudoti tik paprasti kintamieji

Tai trumpiausios ir racionaliausios programos. Dauguma jų parašytos kaip pagrindinė programa. Programų pavyzdžiai pateikiami prieduose P6, P7.

Dalis programų yra paprastesnės struktūros. Siekiant išvengti ciklas cikle situacijos, vienos laidininko dalies laidumui arba varžai skaičiuoti parašoma funkcija arba procedūra. Programų pavyzdžiai pateikti prieduose P8 ir P9.

Nemokėta spausdinti duomenų su formatais. Pavyzdžiui, vietoje WriteLn(rez, R:8:2); rašo sudėtingą programos fragmentą:

```
temp2 := Trunc(Round(R * 100);   // Čia temp2 ir sot yra sveikojo tipo kintamieji
sot := temp2 mod 100;
Write(rez, temp2 div 100, '.'); // Čia rez – rezultatų failo kintamasis
if sot < 10 then Write(rez, 0);
WriteLn(rez, sot);
```

Keista situacija. Toks atskirų dalių (sveikosios, taško, trupmeninės) išskaidymas ir išvedimas atskirai rodo, kad mokinys pakankamai kvalifikuotas. Tad kodėl nemoka parašyti išvedamos reikšmės formato? Užduočiai skirtas laikas ribotas ir jį reikia taupyti.

Tipiška klaida yra sumos kaupimo kintamiesiems veiksmų pradžioje nepriskirti nulio. Dauguma Paskalio kalbos kompiliatorių valo `integer` ir `real` tipo kintamųjų atminties sritį, priskirdami nulį, tačiau tai nėra visiems kompiliatoriams būtina taisyklė. Tai gali būti nepataisoma klaida, jeigu kintamasis jau buvo panaudotas. Visuomet būtina pačiam programuotojui pasirūpinti kintamųjų reikšmių pradine būsena.

Panašių situacijų programose yra ne viena.

Mokiniai, kurie tik susipažinę su programavimu, savo programose tarpinius ir galutinius rezultatus spausdina ekrane. Yra ir tokių, kurie spausdina pradinius duomenis. Vertinant programas, tokie sakiniai buvo ignoruojami, nors struktūriniu požiūriu tai nereikalingas balastas ir vertinant programos struktūrą turėtų būti skiriama mažiau taškų.

Yra programų, kuriose tikrinama, ar teisingi duomenys. Labai dažnai buvo tikrinama, ar duomenų faile nėra daugiau duomenų, nei užduotyje nurodytas maksimalus galimas duomenų skaičius, nors užduotyje buvo pasakyta, kad duomenys yra teisingi ir jų tikrinti nereikia.

Programose pasitaikė labai neracionalių sakinių grupių, kaip pvz., tokių:

```
sum1:= 0;
sum1:= 1/s;
sum:= sum + sum1;
```

Per ilgi kintamųjų vardai. Dėl to nepasidaro aiškiau, tik rašant programą ilgiau gaištama. Pavadinimuose nerekomenduotina vartoti pabraukimo brūkšnelį, nes jis sunkiau pastebimas. Tikslinga vardus klijuoti, vardus, esančius kintamojo vardo viduje pradėti didžiąja raide, pvz.:

```
in_file, out_file    : text;
   count, i             : integer;
   resistance_full,
   resistance_block,
   resistance           : real;
```

Kodėl varžai žymėti netinka fizikoje vartojama raidė R? Ji ir užduotyje yra. Netikslinga užduotyje esančių vardų keisti kitais. Tai klaidų šaltinis.

Ne viena buvo programa, kurioje visi veiksmai apiforminami procedūra, o pagrindinėje programoje lieka vienintelis kreipinio į tą procedūrą sakinys, kurį sudaro vienas žodis (procedūros vardas). Kodėl taip daryta, neaišku.

Labai įdomus vienos programos algoritmas. Užduotyje pateiktos formulės mokiniui nepatiko, todėl jis skaičiavimus organizavo pagal principą „visuomet yra laidininko dalis, sudaryta iš dviejų lygiagrečiai sujungtų varžų“. Skaičiavimuose autorius laiko, kad visuomet lygiagrečiai sujungtos tik dvi varžos: pirmoji ir antroji, po to jų atstojamoji ir trečioji, vėl šių atstojamoji ir ketvirtoji, ir t. t. Tai visiškai teisingas algoritmas, tačiau sudėtingesnis. Tai rodo, kad užduotyje pateikiamas matematinis uždavinio sprendimo būdas daliai mokinių nepriimtinas, nors jis ir paprastesnis. Programos pavyzdys pateiktas priede P10.

Yra programų, kuriose nurodytas dalių skaičius pirmoje failo eilutėje nuskaitomas, bet nenaudojamas. Skaitoma iki failo pabaigos, vartojant funkciją:

`Eof(fd); // fd –` duomenų failo kintamasis

Analogiškai yra su duomenimis, skaitomais iš eilučių. Pirmasis skaičius ignoruojamas (perskaitomas, bet nenaudojamas) arba priimamas kaip varža (o tai jau klaida). Skaičiai skaitomi iki eilutės galo, vartojant funkciją:

`EoLn(fd); // fd –` duomenų failo kintamasis

Programos, kuriose panaudoti vienmačiai masyvai

Daugumoje tokių programų yra sukuriamas vienmačio masyvo tipas ir jis panaudojamas kintamiesiems bei procedūrų (funkcijų) parametrams skelbti. Yra programų, kuriose vartojamos lokalios masyvo duomenų struktūros. Esant nesudėtingos struktūros vienmačiams masyvams, taip daryti nėra klaidinga, tačiau nerekomenduotina.

Daugumoje programų masyvai skelbiami globaliais. Skirtingose programose jų yra nuo 1 iki 5.

Programos, kuriose vartojami daugiau kaip 4 masyvai, pagrindinė klaida yra nemokėjimas susieti juose saugomų reikšmių tiek pagal prasmę, tiek pagal sąrašų indeksinį atitikimą.

Tipiškas veiksmų su masyvais pavyzdys:

```
ReadLn( F, n);              // Skaitomas laidininko dalių skaičius
for i := 1 to n do begin
   Read(F, k);              // Skaitomas dalies varžų skaičius
   for j := 1 to k do       // Skaitomos vienos dalies varžų reikšmės
        Read(F, A[j]);
   ReadLn(F);
   L[i] := DaliesLaidumas(A, k); // Funkcija grąžina dalies laidumo reikšmę
end;
```

Dalis mokinių savo programose perskaitytų reikšmių n ir k nenaudoja, ciklus keičia į tokius:

```
 while not Eof(F) . . .
 while not EoLn(F) . . .
```

Nesuprantama, kodėl nemaža dalis mokinių vietoje vieno sakinio

```
Read(F, A[i]);
```

vartoja du:

```
Read(F, x);
A[i] := x;
```

Arba, skaičiuodami dalies laidumą, du sakinius

```
Read(F, x);
L := L + 1 / x;
```

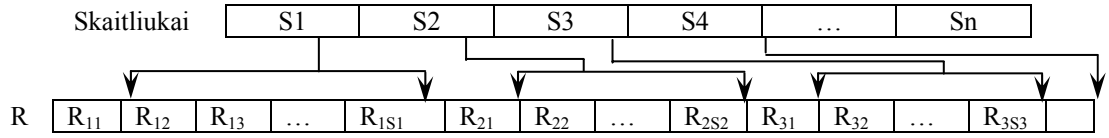
keičia į tris:

```
Read(F, x);
x := 1 / x;
L := L + x;
```

Laidumo nereikėtų žymėti mažąją raide l, kurią po to sunku atskirti nuo vieneto 1 ir dėl to galima padaryti klaidų. Mokiniai turi atsakingiau rinktis simbolius kintamiesiems žymėti.

Masyvo naudojimo variantai

Variantas A. Programa pateikta priede P11.



Čia S1, S2, …, Sn yra laidininko lygiagrečių dalių varžų skaičiai. Masyve R saugomos visų laidininko varžų reikšmės: pirmos dalies, po to antros, trečios ir t. t.

Variantas B. Programa pateikta priede P12.

R	R_1	R_2	R_3	…	Rn									
L	L_1	L_2	L_3	…	Lm									

Varžų masyve saugomos tik vienos lygiagrečios dalies varžų reikšmės. Veiksmai organizuojami taip: vienos dalies varžų reikšmės skaitomos į masyvą R, skaičiuojamas tos dalies laidumas (arba skaičiuojama atstojamoji varža). Po to į masyvą R skaitomos antros dalies varžų reikšmės ir skaičiuojamas laidumas (varža). Baigus nagrinėti visus duomenis, masyve R lieka paskutinės dalies varžų reikšmės, tačiau masyve L yra visos laidumo reikšmės (arba atstojamosios varžos). Lieka pagal duomenis masyve L rasti laidininko varžą. Taigi, tokio tipo programose skaičiavimai perskirti į dvi dalis: skaitymą duomenis laikant daliniais rezultatais ir galutinio rezultato skaičiavimą.

Variantas C. Programa pateikta priede P13.

A	S_1	S_2	S_3	…	Sn									
Rd	R_1	R_2	R_3	…	Rk									
R_L	R_1	R_2	R_3	…	Rn									

Masyve A saugomi laidininko dalių varžų skaičiai (kiek kurioje dalyje yra varžų). Vietoje masyvo galima parašyti paprastą kintamąjį ir programa dirbs be jokių kitų keitimų. Masyve Rd saugomos vienos dalies varžų reikšmės (kaip ir var. B). Masyve R_L surašomos dalių varžos. Atskiru ciklu susumuojami masyve R_L saugomi skaičiai. Tai ir yra laidininko varža.

Iš failo skaitomas pirmasis skaičius *N* (laidininko lygiagrečių dalių skaičius)
i := 1, *N*
i-os dalies varžų reikšmės skaitomos iš failo į masyvą *A(k)*. Čia *k* – varžų skaičius dalyje.
Dalies laidumo skaičiavimas $L[i] = \sum_{j=1}^{k} \frac{1}{A[j]}$
Skaičiuojama laidininko varža $R = \sum_{i=1}^{N} \frac{1}{L[i]}$
Spausdinamas rezultatas *R*.

Variantas D. Programos pateiktos prieduose P14, P15, P16.
Masyve saugomos laidininko atskirų dalių laidumų reikšmės.

L	L_1	L_2	L_3	…	Lm							

Arba masyve laikinai gali būti saugomos vienos laidininko dalies varžų reikšmės. Labai dažnai tokiu atveju masyvas visiškai nereikalingas, nes varžų reikšmės iš karto panaudojamos skaičiavimuose.

Variantas E. Programa pateikta priede P17.
Viename masyve saugomos visų laidininko dalių varžų reikšmės, to paties masyvo gale surašomos tų dalių laidumų reikšmės. Atskirame masyve surašomi kiekvienos laidininko dalies varžų skaičiai (skaitliukai). Gražios programos pavyzdys yra priede P18.

Programos, kuriose panaudoti dvimačiai masyvai

A. Dvimatis masyvas naudojamas lygiagrečių dalių varžų reikšmėms, paverstomis laidumu, saugoti: pirma eilutė – pirmos dalies, antra eilutė – antros dalies ir t. t. Kiek kurioje dalyje yra varžų, neišsaugoma. Veiksmai atliekami iš karto, todėl masyvai yra fiktyvūs (momentiniai), tolesniems veiksmams nereikalingi. Tai paprastų kintamųjų pakaitalas, be reikalo darantis programą sudėtingesnę. Vienmačiame masyve saugomos laidininko dalies laidumo reikšmės, kurios skaičiuojamos duomenų įvedimo metu. Tuo pačiu metu tie laidumai perskaičiuojami į dalies varžas, kurios sumuojamos. Programos pavyzdys pateiktas priede P19.

0.026	2.08	0.045		

1/15	1/41					
1/1	1/2	1/3	1/4			
1/22	1/11	1/24				

B. Duomenų skaitymas ir skaičiavimai atskirti ir dažnai apiforminti atskiromis procedūromis. Dvimačio masyvo eilutė skirta vienos dalies varžų reikšmėms saugoti, o varžų skaičiai (skaitliukai) surašomi vienmačiame masyve. Po to skaičiuojamas kiekvienos dalies laidumas ir išsaugomas atskirame vienmačiame masyve. Jo reikšmės iš karto panaudojamos laidininko varžai rasti, taigi tas masyvas nebūtinas. Vietoje jo gali būti panaudotas paprastas kintamasis. Programos pavyzdys yra priede P20.

2	4	3		

15	41							
1	2	3	4					
22	11	24						

0.026	2.08	0.045		

C. Dvimačio masyvo eilutės skiriamos, kaip ir ankstesniu atveju, dalių varžų reikšmėms saugoti. Tik jos surašomos nuo antro stulpelio, o pirmame stulpelyje yra skaitliukas, rodantis kiek toje dalyje yra varžų. Kitaip tariant, masyvo eilutėje saugomi failo vienos eilutės duomenys. Kitu veiksmu yra skaičiuojami kiekvienos dalies laidumai ir jie surašomi vienmačiame masyve. Paskutiniu ciklu yra skaičiuojama laidininko varža. Kadangi visi veiksmai atskirti, tai masyvai išnaudojami pagal paskirtį. Visi duomenys ir tarpiniai rezultatai išsaugomi ir gali būti panaudoti. Tai profesionalus požiūris, sudarantis galimybę programą modifikuoti, papildyti naujais veiksmais. Žinoma, per egzaminą taip elgtis netikslinga, nes nėra taupomas laikas. Programos pavyzdys pateiktas priede P21.

0.026	2.08	0.045		

2	15	41						
4	1	2	3	4				
3	22	11	24					

Pasitaikė programų, kuriose dvimatis masyvas naudojamas truputį kitaip, tačiau tai atskiri atvejai. Dauguma šių programų buvo klaidingos. Nežinant, kiek kurioje dalyje yra varžų reikšmių (neišsaugo), toliau mokiniai nagrinėja masyve kiekvieno elemento reikšmę: jeigu reikšmė nulis, tai varžos nėra, jeigu ne, – yra. Tuomet pamirštama skaičiuoti, o kiek tų varžų yra. Pasitaikė programų, kuriose panaudoti du dvimačiai masyvai. Pirmame surašytos varžų reikšmės, antrame – laidumai.

Programos, kuriose panaudoti masyvai su įrašo tipo elementais

Tipinė duomenų tipo struktūra tokia, kaip parodyta priede P22 pateiktame programos pavyzdyje:

```
type
 Tvarzos = array [ 1..50 ] of real;
 TLaidininkai = record
                 Lskaicius : integer;    // Varžų skaičius vienoje dalyje
                 Varzos : Tvarzos;       // Vienos dalies varžų reikšmės
                 Bvarza : real;          // Dalies laidumas
                end;
 Tdalys = array [ 1..100 ] of TLaidininkai;
 var Dskaicius : integer;  // Lygiagrečių dalių skaičius laidininke
     Vvarza : real;        // Laidininko varža (rezultatas)
     Dalys : Tdalys;       // Duomenys apie laidininką
```

Jeigu programoje duomenų skaitymas yra atskira procedūra, tai skaičiavimai atliekami visi iš karto.

Yra programų, kuriose skelbiami vienmačiai skaičių masyvai, tačiau arba jie nenaudojami, arba neteisingai naudojami. Tokių programų nedaug. Dauguma programų teisingos, tačiau pasitaiko ir neesminių klaidų. Panašios teisingos programos pavyzdys pateiktas priede P23. Beje, šioje programoje autorius komentarais išreiškė nepasitenkinimą, kad užduotyje pateiktos formulės yra neteisingos. Parašė savo nuomonę, tačiau programavo pagal užduotį (tai pagirtina, nes vykdo užduotį; liūdna, kad fizikos žinios silpnokos).

Tipinė profesinė (ne programinė) klaida programose yra duomenų tipo lokalus skelbimas. Tai priimtina, kai tokio tipo yra tik vienas vienintelis kintamasis. Dauguma kompiliatorių tokius tipus priima kaip unikalius ir tik esant vienmačiam masyvui struktūros yra palyginamos. Visais kitais atvejais priimama, kad tai yra skirtingo tipo kintamieji. Pavyzdžiui:

```
var  A : array[1..100] of TLaidininkai;
     B : array[1..100] of TLaidininkai;
```

Kintamieji A ir B yra skirtingo tipo (jų elementai – vienodo tipo). Visuomet patartina sukurti duomenų tipą, po to jį naudoti aprašant kintamuosius.

Yra programų, kuriose įrašą sudaro vienas masyvas, pvz.:

```
type mas_dalis = array[1..100] of real;
       dalis = record
                    mas : mas_dalis;
                end;
       masyvas = array[1..100] of dalis;
```

Čia iš esmės yra paslėptas dvimatis masyvas. Įrašas – tik papildomas nereikalingas apgaubimas.

Antrosios praktinės užduoties analizė

Kelionė

Maksimali taškų suma – 33 taškai.

Tarp Vilniaus ir Klaipėdos kursuojančiam autobusui reikia sudaryti grafiką. Žinomi atstumai tarp stotelių, autobuso išvykimo iš Vilniaus laikas (valandos ir minutės) ir vidutinis autobuso greitis. Autobusas į Klaipėdą atvyksta tą pačią parą.

Greitis, laikas ir atstumas yra susieti formule `v = s / t`; čia `v` – vidutinis greitis, `s` – atstumas, `t` – laikas, sugaištas nuvažiuoti tam atstumui.

Parašykite programą autobuso atvykimo į stoteles laikui apskaičiuoti. Skaičiavimus atlikite vienos minutės tikslumu. Laikykite, kad autobusas stotelėse nesugaišta laiko.

Programa turi skaityti duomenis iš tekstinio `Duom2.txt` failo. Pirmoje failo eilutėje yra 4 skaičiai: maršruto stotelių skaičius, autobuso vidutinis greitis, išvykimo iš Vilniaus valanda ir minutės. Stotelių yra ne daugiau kaip 100.

Tolesnėse eilutėse surašyti duomenys apie stoteles. Kiekvienoje eilutėje yra stotelės pavadinimas, užrašytas nuo eilutės pradžios, ir atstumas nuo ankstesnės stotelės. Greitis skaičiuojamas kilometrais per valandą, atstumai – kilometrais. Pavadinimui skirtos pirmos 15 eilutės pozicijų.

Pavyzdžiui, Elektrėnai nuo Vilniaus yra per 50,5 km, Žiežmariai nuo Elektrėnų – per 20 km.

Rezultatus programa turi įrašyti į failą `Rez2.txt`: kiekvienoje eilutėje turi būti pateiktas stotelės pavadinimas, kuriam skiriamos pirmos 15 eilutės pozicijų, ir autobuso atvykimo į stotelę laikas kaip nurodyta rezultatų failo pavyzdyje (po valandų skaičiaus paliekamas vienas tarpas ir rašomas sutrumpinimas `val.`, vėl vienas tarpas, minučių skaičius ir dar po vieno tarpo – sutrumpinimas `min.`).

Duomenų failo `Duom2.txt` pavyzdys

Duom2

```
6 70 10 15
Elektrėnai      50.5
Žiežmariai      20
Kaunas          22.35
Raseiniai       80
Kryžkalnis      20
Klaipėda        100.8
```

Rezultatų failo `Rez2.txt` pavyzdys

Rez2

```
Elektrėnai      10 val. 58 min.
Žiežmariai      11 val. 15 min.
Kaunas          11 val. 34 min.
Raseiniai       12 val. 43 min.
Kryžkalnis      13 val. 0 min.
Klaipėda        14 val. 26 min.
```

Privalomi reikalavimai:

- Duomenis ir rezultatus saugoti masyve (masyvuose) su įrašo tipo elementais.
- Sukurti ir panaudoti procedūrą duomenims skaityti į masyvą su įrašo tipo elementais.
- Sukurti ir panaudoti procedūrą skaičiavimams.
- Sukurti ir panaudoti procedūrą rezultatams įrašyti į failą.
- Programoje panaudoti funkciją, skaičiuojančią laiką (minutėmis), per kurį autobusas nuvažiuoja nurodytą atstumą:

```
function Laikas (atstumas, greitis : real) : integer;
begin
   Laikas := Trunc (atstumas / greitis * 60);
end;
```

čia `atstumas` – autobuso nuvažiuotas atstumas kilometrais, `greitis` – autobuso vidutinis greitis kilometrais per valandą.

Programą įrašykite į kietojo disko katalogą C:\Egzaminas, suteikdami failui vardą pagal šabloną *R01_2.pas* (raidė ir pirmieji du skaitmenys (*01*) – jūsų darbo vietos žymė, atskiras skaitmuo (*2*) – praktinės užduoties eilės numeris). Kitaip įvardyti failai nebus vertinami.

Gavę savo darbo vietos žyme pažymėtą diskelį, nepamirškite į jį nukopijuoti programos failo.

Programos vertinimas

Kas vertinama		Taškai
Tinkami programos darbo rezultatai		12
Programos tekstas	Pagrindinė programa	3
	Duomenų skaitymo iš failo į masyvą procedūra	4
	Rezultatų skaičiavimo procedūra	4
	Rezultatų rašymo į failą procedūra	4
Duomenų tipai ir struktūros, kintamųjų aprašymas		4
Programavimo stilius (rašybos taisyklės, komentarai, prasmingi vardai, programos struktūriškumas)		2
	Iš viso	33

Šios užduoties programa turi apribojimų. Jų tikslas – išsiaiškinti, kaip mokiniai geba panaudoti procedūras ir funkcijas, kaip moka struktūrizuoti programas, kurti duomenų struktūras. Užduotis nereikalauja specialių matematikos žinių. Skaičiavimuose yra perėjimas nuo realiųjų skaičių prie sveikųjų. Realiuosius skaičius paversti sveikaisiais galima dviem būdais (funkcijos `Round`, `Trunc`). Kadangi jų darbo rezultatai skirtingi, užduoties autoriai pateikė funkciją Laikas, siekdami suvienodinti gaunamas reikšmes. Čia panaudota funkcija `Trunc`. Viskas būtų gerai, jeigu funkcija `Trunc` dirbtų korektiškai. Pagal aprašymą ji turi atmesti realiojo skaičiaus trupmeninę dalį. Pasirodo, kad taip ji dirba Turbo Paskalio versijose. FreePascal, Delphi (konsolinis ir komponentinis darbo variantai), Lazarus sistemose ši funkcija dirba nekorektiškai. Tai paaiškėjo pasibaigus egzaminui. Pavyzdžiui:

a	Trunc(a)	a	Trunc(a)	a	Trunc(a)	a	Trunc(a)		
0.500	0	23.990	23	50.010	50	150.010	150	159.990	159
0.510	0	**24.000**	**24**	50.990	50	150.990	150	**160.000**	**159**
0.990	0	24.010	24	**51.000**	**50**	**151.000**	**150**	160.010	160
1.000	**1**	24.990	24	51.010	51	151.010	151	160.020	160
1.990	1	**25.000**	**25**	51.990	51	151.990	151	160.990	160
2.000	**2**	25.010	25	**52.000**	**51**	**152.000**	**151**	**161.000**	**160**
2.010	2	25.990	25	52.010	52	152.010	152	161.010	161

Čia raudonai pažymėtos eilutės, kuriuose funkcijos darbo rezultatas neteisingas. Tačiau ne visuomet, pereinant prie naujos sveikojo skaičiaus reikšmės, rezultatas būna klaidingas. Tai pažymėta mėlynai.

Eksperimentui panaudotas toks tekstas:

```
program TruncDarbas;
var a : real;  fr : text;
begin
  Assign(fr, 'TruncDarbas.txt'); Rewrite(fr);
  a := 0.5;
  while a < 170 do begin
    WriteLn(fr, a : 7:3, '    ', Trunc(a));
    a := a + 0.01;
  end;
  Close(fr);
end.
```

Vertindami U2 programas, vertintojai visur mokinių panaudotas funkcijas `Round` pakeitė į `Trunc`. Nors užduotyje buvo nurodymas vartoti `Trunc`, tačiau dalis mokinių tai ignoravo. Pasirinkę kitą skaičiavimo algoritmą, jie su `Round` gavo teisingus rezultatus pavyzdiniam duomenų rinkiniui, o su `Trunc` – neteisingus. Rezultatų įvairovė priklausė nuo paklaidų, kurios kaupėsi nevienodai, priklausomai nuo pasirinkto algoritmo. Vertintojai dėl šios situacijos taškų, skiriamų už programą, nemažino.

Mokinių programose vyrauja du visiškai skirtingi algoritmai ir jų modifikacijos.

Pirmasis algoritmas. Skaičiuojamas stotelės atstumas nuo Vilniaus. Tai paprastas skaičių sumos kaupimo algoritmas. Po to, panaudojant užduotyje nurodytą funkciją

```
function Laikas (atstumas, greitis : real) : integer; ,
```

buvo suskaičiuojamas laikas minutėmis, per kurį automobilis įveikė kelią nuo Vilniaus iki konkrečios stotelės. Pasivertus automobilio išvažiavimo laiką minutėmis, gaunamas atvykimo į stotelę laikas. Lieka gautą rezultatą pakeisti valandomis ir minutėmis, pavyzdžiui:

```
      startasMin := valStartas * 60 + minStartas; // Išvažiavimo laikas iš Vilniaus minutėmis.
laikas := Laikas(s, v);                           // Čia s – atstumas nuo Vilniaus, v – vidutinis greitis.
atvaziavo := startasMin + laikas;                 // Atvažiavimo laikas į stotelę minutėmis
valandos :=  atvaziavo div 60;                    // Atvažiavimo į stotelę valanda
minutės  :=  atvaziavo mod 60;                    // Atvažiavimo į stotelę minutė
```

Antrasis algoritmas. Skaičiuojamas laikas, per kurį buvo nuvažiuotas atstumas tarp gretimų stotelių. Po to šis laikas pridedamas prie atvažiavimo laiko į ankstesnę stotelę. Šiuo būdu skaičiuojant kaupėsi paklaidos, nes kiekvienas atvykimo į stotelę laikas buvo skaičiuojamas, verčiant gautas minutes sveikuoju skaičiumi. Be to, šiuo būdu parašytos programos buvo didesnės apimties ir jose buvo daug daugiau klaidų. Ypač tose, kuriose valandos ir minutės buvo skaičiuojamos atskirai ir sumuojamos atskirai. Šis būdas neracionalus. Tai rodo, kad rinkdamiesi sprendimo būdą mokiniai nepagalvoja, kad yra ir kitų, paprastesnių.

Mokiniai, kurių pirmos užduoties programos buvo neteisingos, tik iš dalies parašytos, antros užduoties taip pat buvo neteisingos, egzaminą išlaikė surinkę reikiamą skaičių taškų tik iš testo.

Pagal duomenų tipą programas, kurias vertintojai atrinko vertinimo metu, galima suskirstyti į tokias grupes:

- visiškai klaidingos programos;
- nenaudojamas įrašo duomenų tipas;
- su vienu įrašo duomenų tipu;
- su dviem įrašo duomenų tipais;
- su trimis ir daugiau įrašo duomenų tipais.

Programose minutės perskaičiuojamos į valandas ir minutes dviem būdais: panaudojant operacijas `div` ir `mod` ir jų nenaudojant. Tuo atveju yra rašomi papildomi ciklai, kurių metu atiminėjama iš bendro minučių skaičiaus po 60 (vertimas į valandas). Tais atvejais, kai skaičiavimai vykdomi valandomis ir minutėmis, tokie vertimai kelis kartus reikalingi, todėl programos apimtis labai padidėja.

Visiškai klaidingos programos

Programų, kurios iš esmės yra klaidingos nenagrinėsime. Pagrindines klaidas kitose programų grupėse aptarsime toliau (ne visos tokios programos buvo teisingos). Ne visos šios grupės programos buvo įvertintos nuliu taškų. Šiek tiek taškų buvo skirta už programos stilių, rašybos taisyklių išmanymą ir pan. Kadangi iš klaidų mokomasi, kelios programos pateikiamos, kaip pavyzdžiai priede P24 – P28. Keletas „perliukų“ iš tų programų:

```
type  stotel = record [1..100] of real;
type      km = 1..200 of real;
type  duomenys : record of integer;
type  pav : string array [0..15] of char;
type  mas : array [1..duom];  // Programoje duom skelbiamas failo kintamuoju
const D : 'Duom2.txt' : text;
const array : [1..100];
```

Nenaudojamas įrašo duomenų tipas

Nemokantys įrašo duomenų tipo rašė programą kurdami daug vienmačių masyvų, mažiausiai tiek, kiek yra skirtingo tipo informacijos. Pavadinimus saugojo eilučių (`string`) tipo masyve (pvz., P29), atstumus, laikus – skaičių masyvuose. Paprastai, tokiose programose nėra procedūrų ir funkcijų arba rašomos ne tos, kurios yra užduotyje. Jeigu nekreiptume dėmesio į tai, kad rezultatai spausdinami ekrane, failai neuždaromi, programos nestruktūrizuotos ir labai netvarkingos, dalis programų visus veiksmus atlieka teisingai. Tokios programos – tai mokymo pasekmė, kai programavimo kultūrai, struktūrizacijai neskiriama dėmesio, kai pamokose tenkinamasi tik galutiniu rezultatu: nesvarbu kaip programa parašyta, svarbu kad ji veikia teisingai. Užduotis gali būti sėkmingai sprendžiama ir nenaudojant jokių masyvų. Tokiu pavyzdžiu gali būti programa P30. Čia labai sudėtingai minutės verčiamos valandomis ir minutėmis, tačiau visi veiksmai iš esmės teisingi. Keista, kad nemokėta iš failo eilutės skaityti žodžio ir skaičiaus. Skaičius skaitomas kaip simbolių eilutė, po to ji verčiama skaičiumi panaudojant procedūrą `Val`. Pats mokinys nemoka rašyti procedūras ir funkcijas, tačiau pagirtina, kad užduotyje nurodytą funkciją teisingai naudoja. Programa P31 parašyta analogiškai. Joje yra viena klaida: kintamajam `s` (sumuojami atstumai) pradžioje nesuteikiama nulinė reikšmė. Galima įtarti, kad ją parašė mokantis programuoti mokinys, tačiau jam neliko laiko sukurti reikalaujamas procedūras ir funkcijas.

Su vienu įrašo duomenų tipu

Kaip ir tikėtasi, daugumoje programų buvo kuriamas vienas įrašo duomenų tipas, kuriame buvo numatyti laukai duomenims ir rezultatams saugoti. Dauguma tam sukūrė vieną masyvo tipą su įrašo tipo

elementais. Pirmos duomenų failo eilutės duomenims saugoti skyrė keturis paprastus kintamuosius. Kintamųjų vardai prasmingi, tačiau pasitaikė abstrakčių pavadinimų, kurie ne visuomet paaiškinami komentarais. Kai kuriems kintamiesiems žymėti vartojami angliški žodžiai, kurių prasmė neaiškinta (pvz.: City, h, H, FindTime, readFile, Writefile). Tai rodo žemą programavimo kultūrą.

Yra programų, kuriose kuriamas įrašo duomenų tipas tik masyvui, kuriame saugomi pradiniai duomenys. Rezultatai spausdinami skaičiavimo metu. Tie veiksmai ne visuomet surašomi procedūroje. Atskirais atvejais skaičiavimams skirta procedūra gautą eilinį rezultatą siunčia procedūrai, kuri jį spausdina faile. Matyti, kad mokiniai moka programose rašyti ir naudoti procedūras bei funkcijas, tačiau visiškai neaišku, kodėl jie jas kuria savo nuožiūra. Užduotyje nurodyta, kokias procedūras ir funkcijas reikia parašyti ir panaudoti. Gali būti tik viena priežastis – mokiniai neatidžiai skaito užduotį. Vertintojai mažino taškų skaičių už tai, kad nėra užduotyje nurodytos procedūros ar funkcijos, tačiau skyrė taškus už veiksmus, kurie teisingai surašyti, nors ir ne ten, kur turėtų būti.

Pasitaikė programų, kuriose stotelės pavadinimui saugoti buvo kuriamas simbolių masyvas:

```
type  zodis = array[1..15] of char;
```

Tai daug sudėtingiau nei apriboto ilgio simbolių eilutė:

```
type  zodis = string[15];
```

Yra programų, kuriose labai keistas duomenų eilutės skaitymas:

```
procedure Nuskaitymas(var A: TMas; var n, h, min: integer; v: real);
var
  DF: TextFile;           // Duomenų failo kintamasis
  m: integer;             // Laikinas kintamasis indeksui nurodyti
  tmp: string;            // Laikina eilutė nuskaitymui
begin
  m := 0;
  Assign(DF, Duomenys);
  Reset(DF);
  ReadLn(DF, n, h, min, v);
  while not Eof(DF) do
  begin
    Inc(m);
    ReadLn(DF, tmp);
    A[m].Vieta := Copy(tmp, 1, 15);
    A[m].S := val(Copy(tmp, 16, 5)));
  end;
  Close(DF);
end;
```

Neatrodo, kad programos autorius nežinojo arba nesuprato duomenų tipo `string`. Kadangi tokių programų ne viena, galima pagalvoti, kad čia kalti mokytojai.

Dažniausiai pasitaikę duomenų tipai išvardyti lentelėje.

`type stotis = record`
`pav : string [15];`
`atstumas : real;`
`val,`
`min : integer;`
`end;`
`mas = array [1 .. ilgis] of stotis;`
	Įrašas, skirtas saugoti duomenis ir rezultatus:
	stotelės pavadinimas,
	atstumas nuo ankstesnės stotelės,
	autobuso atvykimo į stotelę valanda,
	autobuso atvykimo į stotelę minutė.

Tai geras pasirinkimas. Visa informacija apie stotelę yra viename įraše. Tokios teisingai veikiančios programos pavyzdys yra priede P32.	

`Type eil =string[15];`
`stot = record`
`miest : eil;`
`s : real;`
`end;`
`mas = array [1..100] of stot;`
	Įrašas, skirtas saugoti duomenis:
	stotelės pavadinimas,
	atstumas nuo ankstesnės stotelės.

Saugomi tik duomenys. Jiems skaityti iš failo parašyta procedūra. Rezultatai nesaugomi. Jie spausdinami faile skaičiavimų metu.	

`type mas = record`
`Pav:string[15];`
`km:real;`
`end;`
	Įrašas, skirtas saugoti duomenis:
	stotelės pavadinimas,
	atstumas nuo ankstesnės stotelės.

`type mas2 = array[1..100] of mas;` `type mas3 = array[1..100] of integer;`	Duomenų masyvo tipas. Rezultatų masyvo tipas (saugomas sugaištas laikas minutėmis).
Yra programų, kuriose skaičiavimų rezultatai (atvykimo į stotelę minutės) saugomos atskirame sveikųjų skaičių masyve. Tuo atveju rezultatų spausdinimo procedūroje stotelės pavadinimas spausdinamas iš įrašo, o atvykimo laikas – iš to papildomo masyvo. Spausdinimo metu minutės perskaičiuojamos į atvykimo valandą ir minutę. Čia pažeidžiama programavimo technologijos taisyklė: negalima informacijos elementų išbarstyti skirtingose vietose.	

`type duomenys = record`
`pav : string[15];`
`atvh : integer;`
`atvm : integer;`
`end;`
`type mas = array[1..100] of duomenys;`
`type mas2 = array[1..100] of real;`
	Įrašas, skirtas rezultatams saugoti:
	stotelės pavadinimas,
	atvykimo į stotelę valanda,
	atvykimo į stotelę minutė.
	
	Rezultatų masyvo tipas.
	Duomenų (atstumai) masyvo tipas.

Duomenys skaitomi į skirtingas vietas: pavadinimas į masyvą su įrašo tipo elementais, o atstumai – į atskirą realiųjų skaičių masyvą. Įrašo duomenų tipo pavadinimas neatspindi realios situacijos: aprašo ne duomenis, bet rezultatus. Duomenys išbarstyti skirtingose vietose, tačiau rezultatai – tinkamai.	

`type autobusai = record`
`miestas:string[15];`
`atstumas:real;`
`laikass:integer;`
`end;`
`type masyvas=array[1..100] of autobusai;`
	Įrašas, skirtas saugoti duomenis ir rezultatus:
	stotelės pavadinimas,
	atstumas nuo Vilniaus,
	autobuso atvykimo į stotelę laikas minutėmis.

Duomenų skaitymo procedūroje vykdomas atstumo nuo ankstesnės stotelės vertimas į atstumą nuo Vilniaus. Tokiu būdu pradiniai duomenys prarandami, galimybė modifikuoti programą veiksmais, kuriems reikalingas atstumas tarp stotelių, neįmanomas. Skaičiavimų procedūra skaičiuoja atvažiavimo į stotelę minutes. Spausdinimo procedūra verčia tas minutes į atvykimo valandą ir minutę. Tokioje programoje veiksmai išbarstyti net per tris procedūras.	
Yra keletas programų, kuriose masyvas turi nulinę vietą, į kurią įrašo iš pirmos duomenų eilutės išvykimo iš Vilniaus valandą ir minutę. Tai supaprastina tolimesnius veiksmus. Atvykimo laikas į stotelę yra suma dviejų laikų: atvykimo į ankstesnę stotelę ir trukmė tarpstotyje. Taigi, pirmai stotelei ankstesnioji yra Vilnius. Tokios programos pavyzdys P33.	

`type Miest = record`
`pav : String[15];`
`ats : real;`
`end;`
`mas = array [1..NN] of Miest;`
`mas1 = array [1..NN,1..2] of integer;`
`var A : mas;`
`B : mas1;`
	Įrašas, skirtas saugoti duomenis:
	stotelės pavadinimas,
	atstumas nuo ankstesnės stotelės.
	
	Duomenų masyvo tipas
	Rezultatų masyvo tipas
	Duomenų masyvas
	Rezultatų dvimatis masyvas

Rezultatams saugoti kuriamas dvimatis masyvas, kurio pirmame stulpelyje saugomos atvykimo į stotelę valandos, o antrame – minutės. Stotelės pavadinimas, aišku, yra duomenų masyve. Tokių programų yra tik kelios.	

Su dviem įrašo duomenų tipais

Programų, kuriose du įrašo duomenų tipai, yra nemažai. Daugumoje jų kuriami atskiri įrašai duomenims ir rezultatams saugoti. Išvardysime dažniausiai pasitaikiusias situacijas.

`type TVienas = record`
`val,min : integer;`
`end;`
`TDu = record`
`pav : string[15];`
`ats : real;`
`end;`
`Duom = array[1..100] of TDu;`
`Rez = array[1..100] of TVienas;`
`var D : Duom;`
`R : Rez;`
	Atvykimo į stotelę laikas.
	
	Duomenys:
	stotelės pavadinimas ir
	atstumas nuo ankstesnės stotelės
	
	Duomenų masyvas
	Rezultatų masyvas

Masyvų `D` ir `R` duomenys susieti tuo pačiu indeksu – stotelės pavadinimas, spausdinant rezultatus, imamas iš duomenų masyvo. Tai negerai. Negalimi, pavyzdžiui, duomenų rikiavimo veiksmai viename iš masyvų. Visuose veiksmuose būtina stebėti, kad informacijos seka ir jos skaičius abiejuose masyvuose išliktų tas pats.

`type marsrutai = record`
`miestas : string[15];`
`atstum : real;`
`end;`
`mas = array[1..100] of marsrutai;`
`rezultatai = record`
`miestas1 : string[15];`
`valandos : integer;`
`minutes : integer;`
`end;`
`mas1 = array[1..100] of rezultatai;`
	Duomenys:
	stotelės pavadinimas ir
	atstumas nuo ankstesnės stotelės
	
	Rezultatai:
	stotelės pavadinimas,
	atvykimo į stotelę valanda,
	atvykimo į stotelę minutė

Šis variantas neturi to trūkumo, kuris buvo ankstesniame variante. Programoje reikalingas stotelės pavadinimo perrašymo iš duomenų masyvo į rezultatų masyvą veiksmas.

`type valmin = record`
`val: integer;`
`min: integer;`
`end;`
 	
`duomenys = record`
`pavadinimas: string[15];`
`atstumas: real;`
`laik: valmin;`
`end;`
 	
`mas = array [1..100] of duomenys;`
`var A : mas;`
	Laiko duomenys
	valandomis ir
	minutėmis
	
	Įrašas skirtas saugoti duomenis ir rezultatus.

Pirmasis įrašas panaudojamas ne tik antrojo įrašo viduje aprašant lauką `laik`, kuriame saugomas atvykimo į stotelę laikas, bet naudojamas ir kintamajam, skirtam išvažiavimo iš Vilniaus laikui saugoti, aprašyti. Tai labai puikus variantas.
Labai gerai pastebėta, kad išvykimo valandų versti minutėmis nėra tikslo. Prie išvykimo minučių pridėjus kelyje sugaištą laiką minutėmis, reikia iš tos sumos išskirti valandas ir pridėti prie išvykimo valandos. Šios programos tekstas yra pateiktas priede P34.

`type nn = record`
`st : string [ 15 ];`
`kl : real ;`
`end;`
`nnn = record`
`st : string [ 15 ] ;`
`la : integer ;`
`end ;`
	Duomenų masyvo įrašas:
	stotelės pavadinimas,
	atstumas nuo ankstesnės stotelės.
	
	Rezultatų masyvo įrašas:
	stotelės pavadinimas,
	atvykimo į stotelę laikas minutėmis.

Rezultatų masyve nesaugomas galutinis rezultatas: atvykimo į stotelę valanda ir minutė. Laiko minutėmis vertimas į valandas ir minutes vyksta rezultatų spausdinimo faile metu.

`const MAX = 100;`
`type`
`duom1 = array[0..MAX] of record`
`miestas: string[15];`
`atstumas: real;`
`end;`
`duom2 = array[0..MAX] of record`
`miestas: string[15];`
`val, min: integer;`
`end;`
	Tokių variantų, kai įrašo tipas yra lokalus, nedaug.
	Tai programavimo technologijos požiūriu yra negerai.

`type DuomTipas`
`= record`
`Kiekis: integer;`
`Skaicius: integer;`
`Greitis: real;`
`iValanda: integer;`
`iMinute: integer;`
	Duomenų sąrašo kintamojo tipas
	
	Stotelių skaičius
	Neaišku. Programoje nepanaudotas laukas
	Autobuso vidutinis greitis
	Išvykimo iš Vilniaus valanda
	Išvykimo iš Vilniaus minutė

`Duom: array[1..100] of record`
`Pav: string[15];`
`Kelias: real;`
`end;`
`end;`
	Stotelių sąrašas:
	pavadinimas,
	atstumas nuo ankstesnės stotelės

`AtsTipas` `= record`	Rezultatų sąrašo kintamojo tipas
`Kiekis: integer;`	Stotelių skaičius

`Duom: array[1..100] of record`
`Pav: string[15];`
`Valanda: integer;`
`Minute: integer;`
`end;`
`end;`
	Autobuso važiavimo grafikas:
	stotelės pavadinimas,
	atvykimo į stotelę valanda,
	atvykimo į stotelę minutė.

`var Duom : DuomTipas;`	Duomenų kintamasis
`Ats : AtsTipas;`	Rezultatų kintamasis
Struktūriniu požiūriu yra nerekomenduotina skelbti lokalius masyvų ir lokalius įrašo duomenų tipus. Juos tikslinga parašyti atskirai ir po to panaudoti. Šis duomenų aprašymo variantas yra tikrai įdomus ir vartotinas.	

Su trimis ir daugiau įrašo duomenų tipais

Tokių programų yra nedaug. Jose kuriami masyvai su įrašo tipo elementais atskirai duomenims, rezultatams ir tarpinių veiksmų duomenims saugoti. Tai nėra racionalu, tačiau reikalingas rezultatas gaunamas. Tokios programos pavyzdys yra pateiktas priede P35.

Yra programų, kuriose pirmos duomenų failo eilutės duomenims saugoti sukuriamas įrašo tipo kintamasis (keturi laukai). Dar įdomesnė situacija, kai įrašo viduje yra kitų įrašų. Pavyzdžiui:

`type duomenys = record`
`n: integer;`
`greitis: real;`
`isvykimas: record`
`val: integer;`
`min: integer;`
`end;`
`end;`
	Įrašas, skirtas pirmos duomenų failo eilutės duomenims saugoti.
	
	Vidinis įrašas skirtas autobuso iš Vilniaus išvykimo laikui saugoti.

`stotis = record`
`pavadinimas: string[15];`
`atstumas: real;`
`laikaskelyje: integer;`
	Kuriamas masyvas su šio įrašo tipo elementais. Čia saugomi duomenys ir skaičiavimų rezultatai: stotelės pavadinimas, atstumas nuo ankstesnės stotelės ir laikas, reikalingas nuvažiuoti tarpstotę.

`laikaskelyvalmin: record`
`val: integer;`
`min: integer;`
`end;`
	Čia laikas, per kurį autobusas nuvažiavo atstumą nuo ankstesnės stotelės, išreiškiamas valandomis ir minutėmis.

`atvykimas: record`
`val: integer;`
`min: integer;`
`end;`
`end;`
	Čia saugomas atvykimo į stotelę laikas (skaičiavimų rezultatas). Jis gaunamas sumuojant išvykimo iš ankstesnės stotelės valandas ir minutes, po to jas sudėtingais veiksmais koreguojant į teisingas valandas ir minutes.

Iš visų informacinių technologijų valstybinį brandos egzaminą laikiusių mokinių yra penki mokiniai, kurie iš programavimo testo negavo nė vieno taško, trys iš jų negavo nė vieno taško ir už programavimo užduotis. Lentelėje pateikiama informacija apie šių mokinių viso egzamino įvertinimą.

Mokiniai	Taškai, surinkti už IT testo klausimus	Taškai, surinkti už programavimo testo klausimus	Pirmosios praktinės užduoties ir antrosios praktinės užduoties įvertinimai	Taškų už visą egzamino užduotį skaičius
1	6	0	0	6
2	10	0	4 + 6 =10	20
3	14	0	0	14
4	8	0	0	8
5	10	0	2 + 0 = 2	12

IT testas egzamino esmės neatspindi. Tai bendrojo išsilavinimo žinios, reikalingos visų studijų krypčių studentams. Egzamino paskirtis – suskirstyti mokinius, pasirinkusius universitetuose informatikos ir jai gretimas studijų kryptis. Konkursinę egzamino dalį sudaro programavimo testas ir praktinės programavimo užduotys.

Labai gerai ir puikiai atsakiusieji į IT testo klausimus išlaikė informacinių technologijų valstybinį brandos egzaminą, negavę iš programavimo dalies nė vieno taško.

IŠVADOS IR REKOMENDACIJOS

Informacinių technologijų valstybinio brandos egzamino rezultatų analizė atlikta pirmą kartą. Tai leido ne tik įvertinti 2006 m. informacinių technologijų valstybinio brandos egzamino užduoties kokybę, šį egzaminą laikiusių mokinių pasiekimus, bet ir išryškino valstybinio egzamino problemas.

2006 m. informacinių technologijų valstybinio brandos egzamino pagrindinės sesijos užduoties taškų pasiskirstymas pagal tematiką ir mokinių gebėjimus visiškai atitiko numatytąjį informacinių technologijų brandos egzaminų programoje.

Mokinių darbų analizė parodė, kad geriausia jiems sekėsi atsakyti į testo klausimus, sunkiau sekėsi spręsti praktinius uždavinius.

Gera skiriamąja geba ir optimaliu sunkumu pasižymėjo ne tik testo atvirojo tipo klausimai, bet ir klausimai su pasirenkamaisiais atsakymais. 2006 m. informacinių technologijų valstybinio brandos egzamino pagrindinės sesijos užduoties testas sudarytas iš 7 atvirojo tipo klausimų ir 15 klausimų su pasirenkamaisiais atsakymais. Manytume, kad ir ateityje reikėtų išlaikyti tokią proporciją. Teste būtinai turi būti klausimų, į kuriuos atsakydamas mokinys parodytų ne tik teorines žinias, bet mokėjimą jas taikyti praktikoje, remdamasis įgyta darbo kompiuteriu patirtimi. Šie klausimai geriausiai atspindi mokinių žinias ir praktinius gebėjimus.

Rengiant egzamino užduotis:

- klausimų formuluotėse reikėtų vengti žodžių, suteikiančių asmeniškumo atsakymams;
- būtina, kad sąlygos nurodymai mokiniams būtų pateikiami vienareikšmiškai;
- pageidautina, kad testo klausimų formuluotės ir pateikimo formos (ypač programavimo) būtų įvairesnės;
- reikėtų atsisakyti praktinių užduočių su realiaisiais skaičiais, nes atliekant veiksmus kaupiasi skaičiavimo paklaidos, o pereinant prie sveikųjų skaičių nekorektiškai dirba tam skirtos *Free Pascal* funkcijos;
- reikėtų taip parinkti praktines užduotis, kad sprendimo rezultatai nepriklausytų nuo pasirinkto sprendimo būdo;
- reikėtų praktinių užduočių sąlygose įrašyti tam tikrus apribojimus naudoti sprendimui konstrukcijas ir priemones, kurios daro programas neracionalias, didina jų apimtis, bet nedaro įtakos rezultatams; taip pagerėtų rezultatai ir palengvėtų vertintojų darbas;
- pageidautina įtraukti į praktinę dalį vieną užduotį, kuriai reikėtų parašyti dvi programas: pirmoji tikrintų tik duomenų skaitymo ir rezultatų pateikimo mokinių žinias ir gebėjimus, antroji užduotis būtų pirmosios modifikacija, papildyta veiksmais ir rezultatų spausdinimu (užduočių pavyzdys pateikiamas priede).

Rengiant vertinimo instrukciją ir numatant vertinimo kriterijus, reikėtų vengti kriterijų, kurie iš esmės dubliuoja vienas kitą. Rekomenduojama praktinių užduočių vertinimo sistema turi būti kiek

įmanoma diskretiškesnė ir vienareikšmiškai aiški.

Rengdamiesi valstybiniam brandos egzaminui mokiniai turėtų žinoti, kad:

- egzamino rezultatą lemia ne vien praktiniai darbo kompiuteriu gebėjimai ir teorinės žinios, bet ir mokėjimas susikaupti;
- būtina atidžiai skaityti užduočių formuluotes, užduotis atlikti kruopščiai.

Mokytojai turėtų atkreipti mokinių dėmesį į:

- lentelių formatavimo, dokumentų maketavimo tekstų rengykle įgūdžių formavimą;
- gebėjimų tvarkyti dokumentą tekstų rengyklės teikiamomis dokumento automatinio tvarkymo priemonėmis (turinio, rodyklių sudarymas; išnašų įterpimas; puslapių numeravimas) tobulinimą;
- praktinių gebėjimų formavimą skaičiuoklėje rašant formules, jose naudojant įvairias funkcijas ir tinkamai parenkant langelių absoliučiąsias ir santykines koordinates;
- tinkamų duomenų formatų parinkimą ir langelių formatavimą skaičiuoklėje;
- lietuviškų rašmenų vartojimo elektroniniame laiške problemas;
- socialinius, teisinius ir etinius informacijos aspektus;
- įgūdžių formavimą taisyklingai taikyti realiųjų ir sveikųjų skaičių ir loginį duomenų tipą;
- žinomo ir nežinomo kartojimų skaičiaus ciklų skirtumus ir taikymo ypatybes;
- veiksmų šakojimo algoritmų taikymo ypatybes;
- loginių operacijų `or` ir `and` naudojimą reiškiniuose, jų rezultatų skirtumą;
- procedūros ir funkcijos sąvokas, jų vartojimo būdus;
- reikšmių su formatais spausdinimą.

Tai, kad yra dvi praktinės užduotys, kurių viena paprastesnė ir orientuota tik į veiksmus su skaičiais, o antroji skirta darbui su skaitmenine – tekstine informacija, gerai.

Mokytojams reikėtų daugiau dėmesio skirti duomenų struktūroms.

- Programų sudėtingumą ir apimtį iš esmės apibrėžia pasirinktų duomenų tipų sudėtingumas. Mokiniai, kurdami duomenų tipus, apie tai retai susimąsto.
- Mokiniai, rinkdamiesi praktinių užduočių sprendimo būdą, neapgalvoja, kuris iš tų būdų yra paprastesnis.
- Silpnai mokantys rašo programą naudodami užduotyje pateiktus duomenis, imdami tiek paprastų kintamųjų, kiek jų reikia tiems duomenims saugoti (dažnai įvesdami klaviatūra arba priskyrimo sakiniais suteikdami pradines reikšmes). Nesuvokia, kad duomenų yra daug, užduotyje esantis pavyzdys – tik vienas iš jų.
- Pirmoji užduotis buvo be apribojimų programai. Tai gerai, nes mokiniui palikta visa laisvė kurti. Blogai, nes silpnesni kūrė tokias sudėtingas ir ilgas programas, kad jiems nepakako laiko jas užbaigti. Tikslinga būtų apriboti veiksmų laisvę, pavyzdžiui, uždraudžiant naudoti dvimačius masyvus, įrašo duomenų tipus ir pan. Galima pirmai užduočiai pareikalauti sukurti bent vieną procedūrą.

IT dalis egzamine nėra esminė ir neatspindi egzamino paskirties. Neapribojus minimalaus būtino taškų skaičius iš programavimo dalies, yra tikimybė, kad daugės mokinių, kurie išlaikys egzaminą visiškai nemokėdami programuoti. Blogai, jeigu jie tuo pasinaudos rinkdamiesi informatikos krypties studijas. Jeigu tokių mokinių skaičius ženkliai išaugs ir kelis kartus viršys mokančiųjų programuoti skaičių, tuomet egzamino rezultatai bus iškreipti: dauguma gavusių taškus neviršys 25 proc. taškų ribos.

Priedas

PASIŪLYMAS RENGĖJAMS

Pirmoji užduotis U1

Metų bėgyje N dienų buvo matuojama oro temperatūra. Gauti duomenys surašyti tekstiniame faile 'Duom1.txt'. Pirmoje eilutėje užrašytas skaičius N. Toliau kitose N eilučių surašyti kiekvieno matavimo duomenys: mėnuo, diena, temperatūra. Duomenys faile surašyti atsitiktine tvarka.

Duomenų failo pavyzdys:

```
15
1 15 -25.32
4 2 4.12
4 12 6.4
8 15 32
2 25 -1.1
3 1 4
6 2 5
3 3 -1
7 17 23.14
8 8 27
8 9 18
10 10 5
11 11 -12
12 15 -13
12 13 -24
```

Programos versija U1-A

Parašykite programą, kuri suskaičiuotų vidutinę matavimų temperatūrą (visų temperatūrų reikšmių aritmetinį vidurkį). Gautą rezultatą spausdinkite tekstiniame faile 'Rez1.txt' dviejų ženklų po kablelio tikslumu.

Rezultatų failo pavyzdys:

```
3.22
```

Vertinimo kriterijai	Taškai	Pastaba
Pirmasis testas, kai 1 <= n <= 10	4	Jeigu testų rezultatai teisingi, programos dedamosios dalys nevertinamos
Antrasis testas, kai 1 < n <= 365	5	
Kintamųjų aprašymas	0,5	Vertinama, jeigu bent vieno testo rezultatai klaidingi.
Failo paruošimas skaitymui, uždarymas baigus skaityti	1	
Failo paruošimas rašymui, uždarymas baigus rašyti	1	
Pirmojo skaičiaus skaitymas	0,5	
Skaičių sumavimo algoritmas: ▪ Sumos pradinė reikšmė, ▪ Ciklo sakinys, ▪ Skaičių skaitymas iš failo ▪ Skaičiaus pridėjimas prie sumos	 1 1 1 1	
Vidurkio skaičiavimas	1	
Gauto rezultato spausdinimas faile	1	
Rašybos taisyklės	0,5	Visada vertinama
Programos rašymo stilius	0,5	
Iš viso taškų	10	

```
program U1_A;
var f : text;
    suma, sk : real;
    menuo, diena : integer;
    n, i : integer;
begin
  Assign(F, 'Duom1.txt');  Reset(F);
  suma := 0;
  ReadLn(F, n);
  for i := 1 to n do begin
    Read(F, menuo, diena, sk);
    suma := suma + sk;
  end;
  Close(F);
  Assign(F, 'Rez1.txt');  Rewrite(F);
  Write(F, suma / n :8:2);
  Close(F);
end.
```

Pastaba

2006 m. vertintojai išsakė nuomonę, kad jeigu testų rezultatai teisingi, tada už programą mokinys gauna dvigubai taškų: už testus ir už tas dalis, kurios buvo būtinos, kad programa veiktų teisingai.

Buvo pasiūlyta, kad jeigu testai yra geri, tai papildomai vertinti reikia tik tai, kas buvo privalu daryti (privaloma procedūra, funkcija, masyvas, įrašas ir pan.). Jeigu testai negeri, tuomet reikia vertinti tik tiek, kiek yra teisinga programa (t. y. kokia dalis programos veikia teisingai.

```
Rezultatų failo
pavyzdys:
1   -25.32
2     5.26
5    15.45
6     2.67
7    23.14
8    27.00
9    18.00
10     5.00
11   -12.00
12   -18.50
```

Programos versija U1-B

Parašykite programą, kuri suskaičiuotų kiekvieno mėnesio, kurį buvo bent vieną kartą matuota, vidutinę temperatūrą.

Rezultatus spausdinkite mėnesių numerių didėjimo tvarka tekstiniame faile 'Rez2.txt'. Kiekvienoje eilutėje po du skaičius: mėnesio numeris ir to mėnesio vidutinė temperatūra dviejų ženklų po kablelio tikslumu.

Vertinimo kriterijai	Taškai	Pastaba
Pirmasis testas, kai vienas mėnuo ir vienas matavimas	4	Jeigu testų rezultatai teisingi, programos dedamosios dalys nevertinamos
Antrasis testas, kai vienas mėnuo ir keli matavimai	5	
Trečiasis testas, kai keli mėnesiai ir matavimų skaičius skirtingas	5	
Duomenų tipų, kintamųjų aprašymas	1,5	Vertinama, jeigu bent vieno testo rezultatai klaidingi.
Failo paruošimas skaitymui, uždarymas baigus skaityti	1	
Failo paruošimas rašymui, uždarymas baigus rašyti	1	
Pirmojo skaičiaus skaitymas	0,5	
Sumos kaupimo kintamųjų pradinės reikšmės	2	
Duomenų skaitymas iš failo: ▪ ciklo sakinys, ▪ skaičių eilutės skaitymas iš failo ▪ mėnesio matavimų skaičiaus didinimas ▪ mėnesio temperatūrų sumos didinimas	 1 1 1 1	
Vidurkio skaičiavimo veiksmai	2	
Gautų rezultatų spausdinimas faile	2	
Rašybos taisyklės	0,5	Visada vertinama
Programos rašymo stilius	0,5	
Iš viso taškų	15	

```
program U1_B;
const Cn = 12;
type TM1 = array[1..Cn] of integer;
     TM2 = array[1..Cn] of real;
var f : text;
    sk : real;
    menuo, diena : integer;
    n, i : integer;
    Kiek : TM1;      // Matavimų skaičius
    Suma : TM2;      // Temperatūrų sumos
begin
  for i := 1 to Cn do begin
    Kiek[i] := 0;  Suma[i] := 0;
  end;
  Assign(F, 'Duom1.txt');  Reset(F);
  ReadLn(F, n);
  for i := 1 to n do begin
    Read(F, menuo, diena, sk);
    Kiek[menuo] := Kiek[menuo] + 1;  // Matavimų skaičius per mėnesį
    Suma[menuo] := Suma[menuo] + sk; // Mėnesio temperatūrų suma
  end;
  Close(F);
  Assign(F, 'Rez2.txt');  Rewrite(F);
  for i := 1 to Cn do
    if Kiek[i] > 0
    then  WriteLn(F, i, ' ', Suma[i] / Kiek[i]:8:2);
  Close(F);
end.
```

Antroji užduotis

Prekybos kioske turimų prekių duomenys surašyti tekstiniame faile ‘Duom2txt’. Pirmoje eilutėje parašytas prekių pavadinimų skaičius N. Kitose N eilučių yra duomenys apie prekes. Vienoje eilutėje – vienos prekės duomenys: pavadinimas (pirmosios 20 pozicijų), kiekis ir kaina (realūs skaičiai).

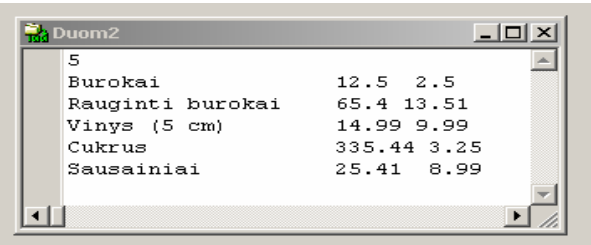
Parašykite programą, kuri surastų prekę, kurios yra už didžiausią pinigų sumą ir suskaičiuotų, už kokią pinigų sumą yra prekių kioske.

Tekstiniame faile ‘Rez3.txt’ pirmoje eilutėje reikia spausdinti duomenis apie surastą prekę: pavadinimas, kiekis, kaina ir pinigų suma. Antroje eilutėje spausdinti vieną skaičių: už kokią pinigų sumą yra prekių kioske. Visi skaičiai realūs ir spausdinami dviejų ženklų po kablelio tikslumu.

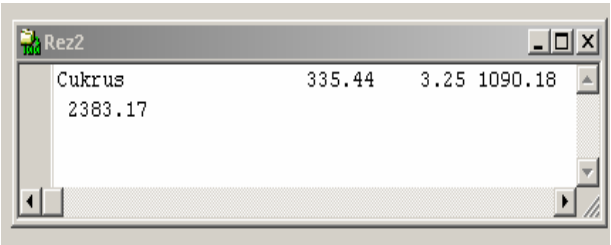
Privalomi reikalavimai programai:

- Duomenis saugoti masyve su įrašo tipo elementais. Tam reikia sukurti įrašų duomenų tipą.
- Parašyti duomenų skaitymo iš tekstinio failo į masyvą su įrašo tipo elementais procedūrą.
- Parašyti procedūrą, kuri suranda prekės, kurios yra už didžiausią pinigų sumą, eilės numerį masyve.
- Parašyti funkciją, kuri suskaičiuoja, už kiek pinigų yra prekių kioske.

Duom2.txt



Rez3.txt



Vertinimo kriterijai	Taškai	Pastaba
Pirmasis testas, kai viena prekė	5	Jeigu testų rezultatai teisingi, programos dedamosios dalys nevertinamos
Antrasis testas, kai daug prekių	10	
Duomenų tipų, kintamųjų aprašymas	2	Jeigu testai geri, tai tikrinama, ar yra reikalingos duomenų struktūros, procedūros ir funkcijos (2 + 2 + 2 + 2) Vertinama, jeigu bent vieno testo rezultatai blogi.
Duomenų skaitymo iš failo procedūra	5	
Didžiausios sumos reikšmės paieškos procedūra	5	
Piniginės prekių sumos skaičiavimo funkcija	4	
Rezultatų spausdinimas	5	
Pagrindinė programa (struktūra, kreipiniai į procedūras, lokalūs kintamieji)	2	
Rašybos taisyklės	0,5	Visada vertinama
Programos rašymo stilius	0,5	
Komentavimas	0,5	
Prasmingi kintamieji	0,5	
Iš viso taškų	25	

```
program U2;
const Cn = 100;
type  TPreke = record
                 pav : string[20];
                 kie, kai, sum : real;
               end;
       TMas = array[1..Cn] of Tpreke;
var A : TMas; n : integer;
//-----------------------------------------------
procedure Duomenys;
var F : text;
    i : integer;
begin
   Assign(F, 'Duom2.txt');  Reset(F);
   ReadLn(F, n);
   for i := 1 to n do  begin
     ReadLn(F, A[i].pav, A[i].kie, A[i].kai);
     A[i].sum := A[i].kie * A[i].kai;
     WriteLn(A[i].sum:8:3);
   end;
   Close(F);
end;
//-----------------------------------------------
procedure DidSuma(var k : integer);
```

```
      var i : integer;
begin
  k := 1;
  for i := 1  to n do
    if A[i].sum > A[k].sum then k := i;
end;
//-----------------------------------------------
function Suma : real;
var i : integer;
    s : real;
begin
   s := 0;
   for i := 1 to n do
     s := s + A[i].sum;
   Suma := s;
end;
//-----------------------------------------------
var k : integer;
    F : text;
begin
  Duomenys;
  DidSuma(k);
  Assign(F, 'Rez2.txt'); Rewrite(F);
  WriteLn(F, A[k].pav, A[k].kie:8:2, A[k].kai:8:2, A[k].sum:8:2);
  WriteLn(F, Suma:8:2);
  Close(F);
end.
```

Užduoties rengėjų pirmosios praktinės užduoties programų pavyzdžiai

Pirmasis pavyzdys. Panaudota funkcija vienos grandies varžai rasti.

```
// Elektros grandinės varžos skaičiavimas.
program Varza;
const CDuom = 'Duom1.txt';
      CRez  = 'Rez1.txt';
//----------------------------------------
// Vienos grandies varža
function RGrandies(var F : text) : real;
var n, i : integer;
    sk : real;
    L  : real;
begin
    L := 0;                      // Laidumas
    Read(F, n);                  // Grandinės dalyje esančių varžų skaičius
     for i := 1 to n do begin
        Read(F, sk);             // Skaitoma viena varža
        L := L + 1 / sk;         // Laidumų sumavimas
     end;
     ReadLn(F);                  // Vienos grandinės dalies varžų sąrašo pabaiga
     RGrandies :=  1 / L;        // Grandinės dalies varža
end;
//----------------------------------------
var n, i : integer;
       R : real;                 // Laidininko varža
       F : text;                 // Kintamasis darbui su failais
begin
  R := 0;                        //   Laidininko varža
  Assign(F, CDuom); Reset(F);
  ReadLn(F, n);                  // Laidininką sudarančių lygiagrečių dalių skaičius
  for i := 1 to n do
      R := R + RGrandies(F);    // Dalių varžų suma
  Close(F);
  Assign(F, CRez); Rewrite(F);
  WriteLn(F, R:8:2);             // Laidininko varža
  Close(F);
end.
```

Antrasis pavyzdys. Panaudota procedūra vienos grandies varžai rasti.

```
// Elektros grandinės varžos skaičiavimas.
program Du;
const CDuom = 'Duom1a.txt';
```

```
            CRez  = 'Rez1a.txt';
//----------------------------------------
// Vienos grandies varža
procedure RGrandies (var F : text; var R : real);
var n, i : integer;
    sk, L : real;
begin
    L := 0;          // Laidumas
    Read(F, n);      // Grandyje esančių varžų skaičius
    for i := 1 to n do
      begin
        Read(F, sk);         // Skaitoma viena grandies varža
        L := L + 1 / sk;     // Laidumų sumavimas
      end;
    ReadLn(F);               // Vienos grandies sąrašo pabaiga
    R := 1 / L;
end;
//----------------------------------------
var n, i : integer;
       F : text;       //   Kintamasis darbui su failais
       R, Gr : real;
begin
  R := 0;              //   Laidininko varža
  Assign(F, CDuom); Reset(F);
  ReadLn(F, n);        // Laidininką sudarančių dalių skaičius
  for i := 1 to n do
    begin
      RGrandies(F, Gr);
      R := R + Gr;    // Laidininko dalių varžų sumavimas
    end;
  Close(F);
  Assign(F, CRez); Rewrite(F);
  WriteLn(F, R:8:2);         // Laidininko varža
  Close(F);
end.
```

Trečiasis pavyzdys. Visi veiksmai atliekami pagrindinėje programoje. Struktūra ciklas cikle panaudota.

```
// Elektros grandinės varžos skaičiavimas
program Du;
const CDuom = 'Duom1a.txt';
      CRez  = 'Rez1a.txt';
//----------------------------------------
var n, k, i, j : integer;
    R, L, rj : real;
    fd, fr : text;
begin
  R := 0;              //   Laidininko varža
  Assign(fd, CDuom); Reset(fd);
  ReadLn(fd, n);        // Laidininką sudarančių dalių skaičius
  for i := 1 to n do
    begin
      L := 0;           // Vienos dalies laidumas
      Read(fd, k);      // Vienos dalies varžų skaičius
      for j := 1 to k do
        begin
          Read(fd, rj);      // Skaitoma varža
          L := L + 1 / rj;   // Laidumų sumavimas
        end;
      ReadLn(fd);            // Vienos dalies varžų sąrašo pabaiga
      R := R + 1 / L;        // Laidininko bendra varža
    end;
  Close(fd);
  Assign(fr, CRez); Rewrite(fr);
  WriteLn(fr, R:8:2);         // Laidininko bendra varža
  Close(fr);
end.
```

Ketvirtasis pavyzdys. Duomenų skaitymas ir skaičiavimai atskirti. Tokia situacija galima. Bandomojo egzamino metu tokių programų buvo (su masyvais, su įrašų masyvais ir pan.).

```
// Elektros grandinės varžos skaičiavimas
program Varza;
const CDuom = 'Duom1c.txt';
```

```
            CRez  = 'Rez1c.txt';
      Cn    = 100;              // Lygiagrečių dalių skaičius
      Ck    =  50;              // Varžų skaičius dalyje
type  TMas  = array[1..Ck] of real;
      TDalis = record
                 k : integer;
                 D : TMas;
               end;
      TLaidininkas = array[1..Cn] of TDalis;
var   A : TLaidininkas;  n : integer;  // Duomenys
//----------------------------------------
procedure Duomenys;
var  fd : text;
     i, j : integer;
begin
  Assign(fd, CDuom); Reset(fd);
  ReadLn(fd, n);                // Laidininką sudarančių dalių skaičius
  for i := 1 to n do
    begin
      Read(fd, A[i].k);         // Vienos dalies varžų skaičius
      for j := 1 to A[i].k do
        Read(fd, A[i].D[j]);    // Skaitoma varža
      ReadLn(fd);               // Vienos dalies varžų sąrašo pabaiga
    end;
  Close(fd);
end;
//----------------------------------------
function BendraVarza : real;
var i, j : integer;
    R, L : real;
begin
  R := 0;                         //   Laidininko varža
  for i := 1 to n do
    begin
      L := 0;                     // Vienos dalies laidumas
      for j := 1 to A[i].k do
         L := L + 1 / A[i].D[j];  // Laidumų sumavimas
      R := R + 1 / L;             // Laidininko bendra varža
    end;
  BendraVarza := R;
end;
//----------------------------------------
var fr : text;
begin
  Duomenys;
  Assign(fr, CRez); Rewrite(fr);
  WriteLn(fr, BendraVarza:8:2);     // Laidininko bendra varža
  Close(fr);
end.
```

EGZAMINE MOKINIŲ PARAŠYTŲ PROGRAMŲ PAVYZDŽIAI

Pirmoji užduotis

P1 programa.

```
program Varza;
 var a, b, sum : real;{a- grandines dalių kiekis, b-bendra varžų suma, sum-grandines
                       bendra varža}
begin
  WriteLn('Iveskite grandine daliu kieki');
  Readln(a);
  WriteLn('Iveskite visu varzu bendra suma'); {pvz. R1=15, R2=13, R3=14, tai sudeti ju
                                               visu skaiciusR1+R2+R3=15+13+14= 42}
  Read(b);
  sum := 1 / b ;
  writeLn('bendra varza yra', sum);
  Read(sum);
end.
```

P2 programa.

```
procedure
```

```
        accros := Duom1.txt;
  end
program Bevardл26;
  Duom1.txt : integer
  var R, R1, R2, R3, R4, R5, R6, R7, R8, R9 : integer;
      L1, L2, L3 : real;
begin
  Readln(R1, R2);
  Readln(R3, R4, R5, R6,);
  Readln(R7, R8, R9);
  L1 := (1 div R1) + (1 div R2);
  L2 := (1 div R3) + (1 div R4) + (1 div R5) + (1 div R6);
  L3 := (1 div R7) + (1 div R8) + (1 div R9);
  R := (1 div L1) + (1 div L2) + (1 div L3);
  writeln (R);
end.
```

P3 programa.

```
program Bevardė0;
Const NN = 500;
      Duom = 'Duom1.txt';
      Rez = 'Rez1.txt';
Type Mas = array[1..NN] of integer;
     Ma = array[1..NN] of real;
var a, b, n, j, i : integer;
    Duomenys, Rezultatai : text;
    Ats : real;
    R: Mas;
    L1, L2, L3: Ma;
    K1, K2, K3 : real;
begin
  Assign(Duomenys, Duom);
  Reset(Duomenys);
  Read(Duomenys, a);
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 2 do begin
                    read(Duomenys, n);
                    L1[i]:=1/n;
                   end;
  K1:=0;
  for i:=1 to 2 do begin K1:=l1[i]+k1; end;
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 4 do begin
                    read(Duomenys, n);
                    L2[i]:=1/n;
                   end;
  K2:=0;
  for i:=1 to 4 do begin K2:=l2[i]+k2; end;
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 3 do begin
                    read(Duomenys, n);
                    L3[i]:=1/n;
                   end;
  K3:=0;
  for i:=1 to 4 do begin K3:=l3[i]+k3; end;
  Close(Duomenys);
  Ats:=(1/k1)+(1/k2)+(1/k3);
  Assign(Rezultatai, Rez);
  Rewrite(Rezultatai);
  Write(Rezultatai, Ats:2:2);
  Close(Rezultatai);
  Readln;
end.
```

P4 programa.

```
program C03_1;
  var n, i, a : integer;
  type mas = array [n...100]
begin
  procedure lygiagretus (A : Mas; R:real);
```

```
        essign (Duom1.txt); reset (duom1.txt);
  read(n;i);
  for  1 to n do
    while i<=50 do
      R[n]:=1/R[a]+...+1/R[a+i];
      i := i + 1;

 procedure bendra(A : Mas; L : real);
  L:= 1/r[n]+...+1/r[n+1];
 write (Rez1.txt) rewrite (Rez1.txt);
end
```

P5 programa.

```
program Bevardл0;
Const NN = 500;
      Duom = 'Duom1.txt';
      Rez = 'Rez1.txt';
Type Mas = array[1..NN] of integer;
     Ma = array[1..NN] of real;
var a, b, n, j, i : integer;
    Duomenys, Rezultatai : text;
    Ats : real;
    R: Mas;
    L1, L2, L3: Ma;
    K1, K2, K3 : real;
begin
  Assign(Duomenys, Duom);
  Reset(Duomenys);
  Read(Duomenys, a);
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 2 do begin
                    read(Duomenys, n);
                    L1[i]:=1/n;
                   end;
  K1:=0;
  for i:=1 to 2 do begin K1:=l1[i]+k1; end;
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 4 do begin
                    read(Duomenys, n);
                    L2[i]:=1/n;
                   end;
  K2:=0;
  for i:=1 to 4 do begin K2:=l2[i]+k2; end;
  Readln(Duomenys);
  Read(Duomenys, j);
  for i:=1 to 3 do begin
                    read(Duomenys, n);
                    L3[i]:=1/n;
                   end;
  K3:=0;
  for i:=1 to 4 do begin K3:=l3[i]+k3; end;
  Close(Duomenys);
  Ats:=(1/k1)+(1/k2)+(1/k3);
  Assign(Rezultatai, Rez);
  Rewrite(Rezultatai);
  Write(Rezultatai, Ats:2:2);
  Close(Rezultatai);
  Readln;
end.
```

P6 programa.

```
program Varza;
var duom, rez : text;                    {tekstiniai failai}
    i, j,
    n, d : integer;
    R, L, LR : real;
begin
 assign (duom, 'duom1.txt');             {priskiriami failai}
 assign (rez, 'rez1.txt');
 reset (duom);
```

```
      rewrite (rez);
  readln (duom, n);
  for i := 1 to n do begin              {skaičiuojama varža}
   read (duom, d);
   L := 0;
   for j := 1 to d do begin
    read (duom, LR);
    L := L + (1 / LR);
    end; readln(duom);
   R := R + (1 / L);
  end;
 writeln(rez, R:5:2);                   {rezultatai rašomi į faila}
 close(duom);
 close(rez);
end.
```

P7 programa.

```
program A04_1;
  var i, j, n, m : integer;
      R,
      Rn,                         { atskirų laidininkų varžos }
      L : real;
      f, g : text;
begin
  assign(f, 'Duom1.txt');
  reset(f);
  readln(f, n);            { nuskaitomas nuosekliai sujungtų dalių skaičius}
  R := 0;
  for i := 1 to n do
    begin
      read(f, m);            { nuskaitomas lygiagrečiai sujungtų varžų kiekis }
      L := 0;
      for j := 1 to m do
        begin
          read(f, Rn);       { nuskaitomos atskiros laidininkų varžos }
          L := L + 1/Rn;   { lygiagrečiai sujungtų varžų varžos skaičiavimas }
        end;
      R := R + 1/L;         { visos grandinės varžos skaičiavimas}
    end;
  close(f);
  assign(g, 'Rez1.txt');
  rewrite(g);
  writeln(g, R:8:2);        { į byla įrašomas atsakymas }
  close(g);
end.
```

P8 programa.

```
program RA04_1;
  var f, f1 : text;
      n, k : integer;
      rv : real;

function resist:real; //Funkcija skirta varzos skaičiavimui grandies dalyje
  var i, m : integer;
      r : real;
begin
  read(f, m);     //Laidinikų skaičius
  resist := 0;
  for i := 1 to m do      //Skaičiuoja varža grandies dalyje
    begin
      read(f, r);
      resist := resist + 1/r;  //pagal formulę 1/R = 1/R1+1/R1+...
    end;
  resist := 1/resist;
end;

begin
  assign(f, 'Duom1.txt');
  reset(f);
  assign(f1, 'Rez1.txt');
  rewrite(f1);
  readln(f, n);
```

```
        rv := 0;
  for k := 1 to n do     //Ciklas skaičiuoja bendra važa
    rv := rv + resist;
  writeln(f1, rv:4:2);
  close(f);
  close(f1);
end.
```

P9 programa.

```
program varžos;
    var x, y: integer;
        suma: real;
        duom, rez: text;     {kintamieji byloms}
function skaitsum: real;     {funkcija, nuskaitanti viena duomenų eilutę}
    var a, c, d: integer;    {ir ja pagal formulę susumuojanti}
begin
    skaitsum := 0;
    read(duom, a);
    for c := 1 to a do
        begin
            read(duom, d);
            skaitsum := skaitsum + 1/d;
        end;
    readln(duom);
end;
{---programa "Varžos"---}
begin
    assign (duom, 'Duom1.txt'); reset (duom);         {bylos, priskiriamos}
    assign (rez, 'Rez1.txt'); rewrite (rez);          {kintamiesiems}
    readln(duom, x);           {nuskaitomas nuosekliai sujungtų dalių skaičius}
    suma := 0;
    for y := 1 to x do
        begin
            suma := suma + 1/skaitsum;
        end;
    writeln(rez, suma:3:2);       {rezultatas įrašomas į byla}
    close (duom); close (rez);    {uždaromos bylos}
end.
```

P10 programa.

```
program A06_1;
var duom, rezl:text;
    NSDS , LSDS, i, j:integer;
    pirvar, antvar, GAV:real;
function LSVAV (r1, r2:real):real; {Skaiciujam dvieju gretimu lygreciai sujungtu varzu atstojamaja}
begin
 LSVAV := (r1*r2)/(r1+r2);
end;
begin
 {failu priskyrimas}
 Assign (duom, 'Duom1.txt');
 Assign (rezl, 'Rez1.txt');
 {Failu prauosimas}
 Reset (duom);
 ReWrite (rezl);
 {Duomenu skaitymas}
 ReadLn(duom,NSDS);
 {Duomenu paruosimas}
 GAV:=0;
 {Uzdavinio sprendimas}
 for i:=1 to NSDS do begin
                  Read (duom,LSDS, pirvar); {Skaitom lygegreciai sujungtu varzu kiekiir pirmaja varza}
                  for j:=2 to LSDS do begin {sukam kol turim varzu}
                          Read (duom,antvar); {skaitom antraja varza}
                          pirvar:=LSVAV(pirvar, antvar); {Pirmoji varza bus dviju varzu atstojamoji}
                        end;
                  GAV:=GAV+pirvar; {Sumuojam nuosekliai sujungtu daliu varza}
                  end;
 WriteLn (rezl,GAV:0:2); {Spausdinam atsakyma (mus domina 2 skaiciai po kablelio)}
 {uzdarome failus}
 close (rezl);
 close (duom);
```

```
        {Pabaiga}
end.
```

P11 programa.

```
program B04_1;
  const Dfail = 'Duom1.txt';       {Pradinių duomenų failas}
        Rfail = 'Rez1.txt';        {Rezultatų faila}
  type Tdalys = array[1..100] of integer;
       Tvarzos = array[1..50000] of integer;
{---------------------------------------}
procedure skaitymas(var dalys : Tdalys; var varzos : Tvarzos; var n : integer);
  var i, i2, i3 : integer;         {Ciklo kintamieji}
      f : text;
begin
  i3 := 1;
  Assign(f, Dfail);
  Reset(f);
  ReadLn(f, n);                    {Nuskaitomas grandinių skaičius}
  for i := 1 to n do
   begin
     Read(f, dalys[i]);            {Nuskaitomas varžų kiekis vienoje grandinėje}
     for i2 := 1 to dalys[i] do
       begin
         Read(f, varzos[i3]);      {Nuskaitomos varžų reikšmės}
         i3 := i3 + 1;
       end;
     ReadLn(f);
   end;
  close(f);
end;
{---------------------------------------}
procedure skaiciavimas(dalys : Tdalys; varzos : Tvarzos; n : integer);
  var i, i2, i3 : integer;
      R,                           {Bendra grandinių varža}
      L : Real;                    {Vienos grandinės varža}
      F2 : text;
begin
  i3 := 0;
  R := 0;
  for i := 1 to n do
    begin
      L := 0;
      for i2 := 1 to dalys[i] do
        begin
          i3 := i3 + 1;
          L := L + 1 / varzos[i3]; {Skaičiuojama vienos grandinės varža}
        end;
      R := R + 1 / L;              {Skaičiuojama bendra grandinės varža}
    end;
  Assign(F2, Rfail);
  Rewrite(F2);
  WriteLn(f2, R :0:2);             {Įrašoma į faila rezultatai}
  close(F2);
end;
{---------------------------------------}
  var  dalys : Tdalys;             {Dalies varžų skaičius}
      varzos : Tvarzos;            {Varžų reikšmės}
           n : integer;            {Dalių skaičius}
begin
  skaitymas(dalys, varzos, n);
  skaiciavimas(dalys, varzos, n);
end.
```

P12 programa.

```
program A05_1;
uses crt;
var  F,Rez:text;
     a:array[1..100] of real;
     b:array[1..100] of real;
     x,i,k,t,v:integer;
     L,R:real;
```

```
      begin
clrscr;
writeln('* Programa veikia *');
writeln;
Assign(F,'Duom1.txt'); //Failo priskyrimas
Reset(F);
Assign(Rez,'Rez1.txt'); //Priskiriamas atsakymų failas
Rewrite(Rez);
v:=1;
   readLn(F, k);
while not Eof(F) do          //Duomeų skaitymas
  begin
  i:=1;
   while not Eoln(F) do
  begin
  Read(F,x);
  a[i]:=x;         //Suomenų kėlimas įmasyva
  i:=i+1;
  end;
  L := 0;
  for k:=2 to i-1 do begin    //Skaičiuojama grandinės varža
  if a[k]>0 then
  L:=L+(1/a[k]);
  end;
  b[v]:=L;                  //Varža keliama į masyva
  v:=v+1;
  Readln(F);
  end;
        R := 0;
for t:=1 to v-1 do begin     //Skaičiuojama bendra varža
//if b[t]>0 then
R:=R+(1/b[t]);
end;
//writeln('Atsakymas',R:6:2);
writeln(Rez,R:5:2,' ');         //Atsakymo rašymas į faila
//writeln;
//writeln('* Programa darba baigė *');
close(F);
close(Rez);
//Readkey;
end.
```

P13 programa.

```
program A09_1;
  var D:text;
     x,y,dalys:integer;
     R,A:array [1..100] of integer;
     RB:array [1..100] of real;
     RBL,RG:real;{RB - atskiros dalies, visų lygegrčиiai sujungtų laidinikų bendra
varža}
begin                          {RG - visos grandinės varža}
  assign(D,'Duom1.txt');
  reset(D);
  readln(D,dalys);
  RG:=0;
  for x:=1 to dalys do
    begin
      read(D,A[x]);
      RBL:=0;
      for y:=1 to A[x] do
        begin
          read(D,R[y]);
          RBL:=RBL+1/R[y]
        end;
      RB[x]:=1/RBL;          {suskaičiuoju lygiagrečios dalies varža}
    end;
  close(D);
  for x:=1 to dalys do
    RG:=RG+{1/}RB[x];
  {RG:=1/RG;}
  assign(D,'rez1.txt');
  rewrite(D);
```

```
        writeln(D,RG:2:2);
  close(D)
end.
```

P14 programa.

```
program A02_1;
  var f : text;
      sum : array [1..100] of real;   {masivas kur įrašisime sumas lygagrečiai sujungtų rezistorių}
      n1, n2, i, j : integer;  {n1 - eilutės skaičiai}
      Rvidut, r : real;     {Rvidut - bendra varža}
begin
  assign (f, 'Duom1.txt');
  reset (f);
  for i := 1 to 100 do
    sum[i] := 0;
  readln(f, n1);
  for i := 1 to n1 do
    begin
      read (f, n2);
      for j := 1 to n2 do
        begin
          read (f, r);
          sum[i] := sum[i] + 1/r;   {skaičiojame sumas lygagrečiai sujungtų rezistorių}
        end;
      readln (f)
    end;
  Rvidut := 0;
  for i := 1 to n1 do
    Rvidut := Rvidut + 1/sum[i];    {skaičiojame bendra varža}
  close (f);
  assign (f, 'Rez1.txt');
  rewrite (f);
  write (f, Rvidut:4:2);
  close(f);
end.
```

P15 programa.

```
program Varza;

type varz = array[1..50] of integer;

function Lvarza(n:integer; mas:varz) : real;
  var i : integer;              { funkcija lygegrečiai sujungtų varžų grandies }
begin                           { varžai rasti }
      Lvarza:=0;
      for i:=1 to n do
      begin
           Lvarza:= Lvarza + 1 / mas[i];

      end;
end;

var a, i, n, j: integer;
    duom, duom2 : text;
    mas : varz;
    R : real;
begin
     assign(duom, 'Duom1.txt');
     reset(duom);                { skaitymas ip bylos }
     readln(duom, a);
     R:=0;
     for i:=1 to a do
     begin
          read(duom, n);
          for j:=1 to n do read(duom, mas[j]); { nuskaitomi visi duomenys }
          R:=R + 1 / Lvarza(n, mas);   { randama visos grandies varža }
     end;
     assign(duom2, 'Rez1.txt');
     rewrite(duom2);             { įrašymas i byla }
     writeln(duom2, R:2:2);
     close(duom2);
end.
```

P16 programa.

```
program varzos;
  const duom = 'duom1.txt';
        rez = 'rez1.txt';
  type mas = array [1..100] of integer;
       masiv = array [1..15000] of real;
    var M : mas;
        L : masiv;
     x, n : integer;
    visas : real;
{----------------------ivedimas i masyva----------------------------}
 procedure ivedimas;
   var i, j, s : integer;
         F1 : text;
 begin
   assign (f1,duom);
   reset (f1);
   readln (f1,n);
   s := 0;
   for i := 1 to n do
   begin
     read (f1,M[i]);
     for j := 1 to M[i] do
     begin
       s := s + 1;
       read (f1,L[s]);
     end;
   end;
   close (f1);
 end;
{-------------------------suskaiciuojama suma varzu----------------}
procedure suma;
  var i : integer;

begin
  x := 0;
  for i := 1 to n do
    x := x + M[i];
end;
{----------------------visu grandiniu sumu ivedimas i masyva--------}
procedure grandines;
  var i, j, s : integer;
        d, a : real;
begin
  s := 0;
  for j := 1 to n do
  begin
    d := 0;
    for i := 1 to M[j] do
    begin
      s := s + 1;
      a := 1 / L[s];
      d := d + a;
    end;
    L[j+x] := d;
  end;
end;

{---------------------------bendra grandiniu varza------------------}
procedure visa;
  var i : integer;
begin
  visas := 0;
  for i := 1 to n do
    visas := visas + 1/L[i+x];
end;

{--------------------------isvedimas i faila------------------------}
procedure isvedimas;
  var f2 : text;
begin
```

```
        assign (f2,rez);
  rewrite (f2);
  writeln (f2,visas:0:2);
  close (f2);
end;

{-----------------------------pagrindine programa------------------}
begin
  ivedimas;
  suma;
  grandines;
  visa;
  isvedimas;
end.
```

P17 programa.

```
program varzos;
  const duom = 'duom1.txt';
        rez = 'rez1.txt';
  type mas = array [1..100] of integer;
       masiv = array [1..15000] of real;
    var M : mas;
        L : masiv;
     x, n : integer;
    visas : real;
{----------------------ivedimas i masyva----------------------------}
 procedure ivedimas;
   var i, j, s : integer;
         F1 : text;
 begin
   assign (f1,duom);
   reset (f1);
   readln (f1,n);
   s := 0;
   for i := 1 to n do
   begin
     read (f1,M[i]);
     for j := 1 to M[i] do
     begin
       s := s + 1;
       read (f1,L[s]);
     end;
   end;
   close (f1);
 end;
{-------------------------suskaiciuojama suma varzu----------------}
procedure suma;
  var i : integer;
begin
  x := 0;
  for i := 1 to n do
    x := x + M[i];
end;
{----------------------visu grandiniu sumu ivedimas i masyva--------}
procedure grandines;
  var i, j, s : integer;
        d, a : real;
begin
  s := 0;
  for j := 1 to n do
  begin
    d := 0;
    for i := 1 to M[j] do
    begin
      s := s + 1;
      a := 1 / L[s];
      d := d + a;
    end;
    L[j+x] := d;
  end;
end;
```

```
{---------------------------bendra grandiniu varza------------------}
procedure visa;
  var i : integer;
begin
  visas := 0;
  for i := 1 to n do
    visas := visas + 1/L[i+x];
end;
{-------------------------isvedimas i faila-----------------------}
procedure isvedimas;
  var f2 : text;
begin
  assign (f2,rez);
  rewrite (f2);
  writeln (f2,visas:0:2);
  close (f2);
end;
{-----------------------------pagrindine programa------------------}
begin
  ivedimas;
  suma;
  grandines;
  visa;
  isvedimas;
end.
```

P18 programa. **Gražios programos su masyvu pavyzdys.**

```
program rezistoriai;
    type Nmas = array [1..100] of real;
    var a, b: text;{skaitymo rasymo bylos}
        n, k: integer;{grandines dalys ir rez.dalyje sk.}
        r, varza: real;{vieno r ir bendra varza}
        D: Nmas; {grandines daliu varzu saugojimo masyvas}
{-----------------------------------------------------------------}
procedure skaiciuot; {skaiciuoja grandines daliu varzas}
    var i, t: integer;
        L: real;
begin
    assign(a, 'Duom1.txt'); reset(a);
    readln(a, n);
    for i := 1 to n do
        begin
            read(a, k);
            for t := 1 to k do
                begin
                    read(a, r);
                    L := L + 1/r;
                end;
            D[i] := 1/L;
            L := 0;
            readln(a);
        end;
    close(a);
end;
{-----------------------------------------------------------------}
procedure bvarza;{skaiciuojama bendra varza}
    var i: integer;
begin
    varza := 0;
    for i := 1 to n do
        varza := varza + D[i]
end;
{-----------------------------------------------------------------}
begin
    assign(b, 'Rez1.txt'); rewrite(b);
    skaiciuot;
    bvarza;
    writeln(b, varza:2:2);
    close(b)
end.
```

P19 programa.

```
program A09_1;
    type ar1 = array[1..100] of real;
         ar2 = array[1..100, 1..50] of real;
    var duom, rez: text;
        R: real;    {bendra varza}
        Rn : ar1;   {nuosekliai sujungtu grandines daliu varzos}
        Rl : ar2;   {lygiagreciai sujungtu laidininku varzos}
        i, j, n, m: integer;
begin
    assign(duom, 'Duom1.txt');
    reset(duom);
    assign(rez, 'Rez1.txt');
    rewrite(rez);
    readln(duom, n); {nuosekliai sujungtu grandines daliu skaicius}
    R:= 0;
    for i:= 1 to n do begin
        read(duom, m);
        Rn[i]:= 0;
        for j:= 1 to m do begin
            read(duom, Rl[i][j]);
            Rl[i][j]:= 1 / Rl[i][j];
            Rn[i]:= Rn[i] + Rl[i][j];
        end;
        R:= R + 1/Rn[i];
        readln(duom);
    end;
    Write(rez, R:8:2);
    close(duom);
    close(rez);
end.
```

P20 programa.

```
program A02_1;
  var t,g: text;  {t - duomenų failo kintamasis, g - rezultatų failo kintamasis}
      a,b,sk: integer;
      n: array [1..100] of integer;
      l: array [1..50] of real;
      r: array [1..100,1..50] of real;
      visar: real;
begin
{----------------------- dumenų nuskaitymas iš bylos -------------------------}
  assign(t,'Duom1.txt');
  reset(t);
  read(t,sk);     {sk - nurodo kiek grandinės dalių yra sujungta}
  readln(t);
  for a:=1 to sk do
    begin
      read(t,n[a]);          {n[a] - keikvienos grandines dalies varžų skaičius}
      for b:=1 to n[a] do
        read(t,r[a,b]);
      readln(t)
    end;
{---------------------- skaičiavimas ----------------------------------------}
  for a:=1 to sk do
    begin
      for b:=1 to n[a] do
        l[a]:=l[a]+1/r[a,b];
      visar:=visar+1/l[a]
    end;
{--------------------- rezultatų įrašymas į byla ----------------------------}
  assign(g,'Rez1.txt');
  rewrite(g);
  write(g,visar:0:2);
  close(t);
  close(g);
  Readln
end.
```

P21 programa.

```
program A05_1;
var mas : array[1..100, 1..50] of integer;
    L : array[1..100] of real;
    t : text;
    x, y, i, j : integer;
    R : real;
begin
  assign(t, 'Duom1.txt');
  reset(t);
  readln(t, x); {nuskaitome kiek yra grandines daliu}
  for i := 1 to x do
    begin
      read(t, y);   {nuskaitome kiek yra varzu reiksmiu}
      mas[i, 1] := y; {varzu reiksmiu skaiciu irasome kaip pirma masyvo elementa}
      for j := 2 to y+1 do
        read(t, mas[i, j])
    end;
  close(t);
  for i := 1 to x do {skaiciujoame atskiru grandiniu L ir rasome i masyva}
    begin
      y := mas[i, 1];
      for j := 2 to y+1 do
        L[i] := L[i] + 1/mas[i, j]
    end;
  for i := 1 to x do {skaiciuojame bendra varza}
    R := R + 1/L[i];
  assign(t, 'Rez1.txt');
  rewrite(t);
  writeln(t, R:5:2);
  close(t);
  Readln
end.
```

P22 programa.

```
program Varza;
type
 TVarzos = array [ 1..50 ] of real;
 TLaidininkai = record
                 Lskaicius : integer;
                 Varzos : TVarzos;
                 BVarza : real;
                end;
 TDalys = array [ 1..100 ] of TLaidininkai;

 var DSkaicius : integer;
     VVarza : real;
     Dalys : TDalys;

{-----------------Procedūra duomenų nuskaitymui iš tekstinio failo-------------}
Procedure Nuskaitymas;
 var F : Text;
     i, j : integer;
begin
 Assign (F, 'Duom1.txt');
 Reset (F);
 ReadLn (F, Dskaicius);
 for i := 1 to Dskaicius do
 begin
  Read (F, Dalys[i].Lskaicius);
  for j := 1 to Dalys[i].Lskaicius do
   Read (F, Dalys[i].Varzos[j]);
  ReadLn (F);
 end;
 Close (F);
end;

{-----------------Procedūra bendrai varžai skaičiuoti-------------------------}
```

```
      Procedure BendraVarza;
 var i, j : integer;
begin
 for i := 1 to Dskaicius do
 begin
  for j := 1 to Dalys[i].Lskaicius do
   Dalys[i].BVarza := Dalys[i].BVarza + (1 / Dalys[i].Varzos[j]);
  VVarza := VVarza + (1 / Dalys[i].BVarza);
 end;
end;
{-----------------Procedūra duomenų saugojimui rezultatų faile-----------------}
Procedure Rezultatai;
 var F : Text;
begin
 Assign (F, 'Rez1.txt');
 Rewrite (F);
 Write (F, VVarza:5:2);
 Close (F);
end;
{-----------------------------Pagrindinė programa-----------------------------}
begin
  Nuskaitymas;
  BendraVarza;
  Rezultatai;
end.
```

P23 programa.

```
program Varza;
const duom = 'Duom1.txt';
      rez = 'Rez1.txt';
type  masyvas = array[1..50] of integer;
      informacija = record
            A : masyvas;
            k : integer;
      end;
      mas = array[1..100] of informacija;
var n : integer;          {kiek yra nuosekliai sujungtų grandinės dalių}
    A : masyvas;
    M : mas;
    r : real;
{nuskaitomi duomenys ip failo}
procedure nuskaitymas;
var  f1 : text;
     i,
     j : integer;
begin
  assign (f1, duom);
  reset (f1);
  readln (f1, n);
  for i := 1 to n do
  with M[i] do
  begin
    read (f1, k);
    for j := 1 to k do
      read (f1, A[j]);
    readln (f1);
  end;
  close (f1);
end;
{funkcija skaičiuoja grandinės dalies varža}
function dalies_varza (j, v : integer) : real;
var i : integer;
    l : real;
begin
  l := 0;
  for i := 1 to v do
     l := 1/M[j].A[i] + l;
  dalies_varza := l;
end;
{duomenys įrašomi į faila}
procedure irasymas;
var f2 : text;
    i , j : integer;
```

```
      begin
  assign (f2, rez);
  rewrite (f2);
  write (f2, r:5:2);
  close (f2);
end;
{manau, kad salygoje pateikta neteisinga visos grandinės varžos skaičiavimo
formulė, nes grandinės dalys(kurių viduje grandinės dalys sujungtos lygiagrčiai)
yra sujungtos nuosekliai, todėl formulė turėtų būti: R = L1 + L2 + L3 ir t.t.}
var i : integer;
    l : real;
begin
  nuskaitymas;
  r := 0;
  for i := 1 to n do
  begin
    l := dalies_varza (i, M[i].k);
    r := r + 1/l;      {pagal salygoje duota formulė būtų taip, o jei
                        skaičiuotume nuosekliai, tai : r := r + l}
  end;
  irasymas;
end.
```

Užduoties rengėjų pateikti vertintojams antrosios praktinės užduoties programų pavyzdžiai

Pirmasis pavyzdys. Duomenys skaitomi iš failo ir saugomi masyve su įrašo tipo elementais be pakeitimų.

```
program Kelione;
const CDuom = 'Duom2.txt';
      CRez  = 'Rez2.txt';
      Cn = 100;
type  zodis = string[15];         // Žodžio ilgis iki 15 simbolių
      stotele = record
                  pav : zodis;    // Stotelės pavadinimas
                  km  : real;     // Atstumas iki stotelės
                  val,            // Autobuso atvykimo į stotelę valanda
                  min : integer;  // Autobuso atvykimo į stotelę minutė
                end;
      Mas = array[1..Cn] of stotele;
var A : Mas;  n : integer;
    v : real;                     // Autobuso vidutinis greitis
    val, min : integer;           // Išvykimo iš Vilniaus laikas
//-------------------------------------
procedure Duomenys;               // Duomenų skaitymas iš failo
var fd : text;  i : integer;
begin
  Assign(fd, CDuom);  Reset(fd);
  ReadLn(fd, n, v, val, min);     // Stotelių skaičius, greitis, išvykimo val. ir min.
  for i := 1 to n do with A[i] do begin
     ReadLn(fd, pav, km);         // Stotelės pav. ir atstumas nuo ankstesnės stotelės
     val := 0;                    // Rezultatas: atvykimo į stotelę valanda (rekomenduojama
nepalikti „šiukšlės")
     min := 0;                    // Rezultatas: atvykimo į stotelę minutė (rekomenduojama
nepalikti „šiukšlės")
  end;
  Close(fd);
end;
//-------------------------------------
// Laikas minutėmis, per kurį autobusas nuvažiuos atstumą atstumas greičiu greitis
function Laikas (atstumas, greitis : real) : integer;
begin
   Laikas := Trunc (atstumas / greitis * 60);
end;
//-------------------------------------
procedure Rasti;
var i : integer;  s : real;
    atvLaikas : integer;
begin
  s := 0;                      // Atstumui nuo Vilniaus saugoti
  for i := 1 to n do begin
    s := s + A[i].km;          // Atstumas nuo Vilniaus iki stotelės nr. i
    atvLaikas := Laikas(s, v) + val * 60 + min;  // Atvykimo į stotelę laikas minutėmis
    A[i].val := atvLaikas div 60;                // Atvykimo į stotelę valanda
    A[i].min := atvLaikas mod 60;                // Atvykimo į stotelę minutė
  end;
end;
//-------------------------------------
procedure Rezultatai;
var i : integer;  fr : text;
begin
  Assign(fr, CRez); Rewrite(fr);
  for i := 1 to n do with A[i] do
    WriteLn(fr, pav, val, ' val. ', min, ' min.');
  Close(fr);
end;
//-------------------------------------
begin
  Duomenys;
  Rasti;
  Rezultatai;
end.
```

Antrasis pavyzdys. Duomenys skaitomi iš failo ir saugomi masyve su įrašo tipo elementais įrašant atstumą nuo pradinės stotelės Vilnius.

```
program Kelione;
const CDuom = 'Duom2b.txt';
      CRez  = 'Rez2b.txt';
      Cn = 100;
type  zodis = string[15];         // Žodžio ilgis iki 15 simbolių
      stotele = record
                  pav : zodis;   // Stotelės pavadinimas
                  km  : real;    // Atstumas iki stotelės
                  val,           // Autobuso atvykimo į stotelę valanda
                  min : integer; // Autobuso atvykimo į stotelę minutė
                end;
      Mas = array[1..Cn] of stotele;
var A : Mas;  n : integer;
    v : real;                    // Autobuso vidutinis greitis
    val, min : integer;          // Išvykimo iš Vilniaus laikas
//-------------------------------------
procedure Duomenys;              // Duomenų skaitymas iš failo
var fd : text;  i : integer;   s, k : real;
begin
  s := 0;                        // Atstumui nuo Vilniaus saugoti
  Assign(fd, CDuom);  Reset(fd);
  ReadLn(fd, n, v, val, min);    // Stotelių skaičius, greitis, išvykimo val. ir min.
  for i := 1 to n do with A[i] do begin
     ReadLn(fd, pav, k);         // Stotelės pavadinimas ir atstumas nuo ankstesnės
stotelės
     s := s + k;                 // Atstumas nuo Vilniaus
     km := s;
     val := 0;                   // Rezultatas: atvykimo į stotelę valanda (rekomenduojama
nepalikti „šiukšlės“)
     min := 0;                   // Rezultatas: atvykimo į stotelę minutė (rekomenduojama
nepalikti „šiukšlės“)
  end;
  Close(fd);
end;
//-------------------------------------
// Laikas minutėmis, per kurį autobusas nuvažiuos atstumą atstumas greičiu greitis
function Laikas (atstumas, greitis : real) : integer;
begin
   Laikas := Trunc (atstumas / greitis * 60);
end;
//-------------------------------------
procedure Rasti;
var i : integer;
    atvLaikas : integer;
begin
  for i := 1 to n do
    begin
      atvLaikas := Laikas(A[i].km, v) + val * 60 + min;  // Atvykimo į stotelę laikas
minutėmis
      A[i].val := atvLaikas div 60;                      // Atvykimo į stotelę valanda
      A[i].min := atvLaikas mod 60;                      // Atvykimo į stotelę minutė
  end;
end;
//-------------------------------------
procedure Rezultatai;
var i : integer;  fr : text;
begin
  Assign(fr, CRez); Rewrite(fr);
  for i := 1 to n do with A[i] do
    WriteLn(fr, pav, val, ' val. ', min, ' min.');
  Close(fr);
end;
//-------------------------------------
begin
  Duomenys;
  Rasti;
  Rezultatai;
end.
```

Trečiasis pavyzdys. Duomenys skaitomi iš failo ir saugomi masyve su įrašo tipo elementais įrašant atstumą nuo pradinės stotelės Vilnius. Rezultatams saugoti sukuriama kita duomenų struktūra.

```
program Kelione;
const CDuom = 'Duom2b.txt';
      CRez  = 'Rez2b.txt';
      Cn = 100;
type  TZodis = string[15];
//------------------------------------ Duomenims saugoti
      TStotele = record
                   pav : TZodis;
                   km  : real;
                end;
      TMas = array[1..Cn] of TStotele;
//------------------------------------ Rezultatams saugoti
      TGrafikas = record
                     pav : TZodis;
                     val,
                     min : integer;
                  end;
      TAtvykimai = array[1..Cn] of TGrafikas;
var A : TMas;  n : integer;// Duomenų masyvas
    v : real;              // Greitis
    val, min : integer;    // Išvykimo laikas
    B : TAtvykimai;        // Rezultatų masyvas
//-------------------------------------
procedure Duomenys;
var fd : text;  i : integer;   s, k : real;
begin
  s := 0;
  Assign(fd, CDuom);  Reset(fd);
  ReadLn(fd, n, v, val, min);
  for i := 1 to n do with A[i] do
    begin
      ReadLn(fd, pav, k);
      s := s + k;             // Atstumas nuo Vilniaus
      km := s;
    end;
  Close(fd);
end;
//-------------------------------------
function Laikas (atstumas, greitis : real) : integer;
begin
   Laikas := Trunc (atstumas / greitis * 60);
end;
//-------------------------------------
procedure Rasti;
var i : integer;
    atvLaikas : integer;
begin
  for i := 1 to n do
    begin
     B[i].pav := A[i].pav;
                                     // Atvykimo į stotelę laikas minutėmis
     atvLaikas := Laikas(A[i].km, v) + val * 60 + min;
     B[i].val := atvLaikas div 60; // Atvykimo į stotelę valanda
     B[i].min := atvLaikas mod 60; // Atvykimo į stotelę minutė
    end;
end;
//-------------------------------------
procedure Rezultatai;
var i : integer;  fr : text;
begin
  Assign(fr, CRez); Rewrite(fr);
  for i := 1 to n do with B[i] do
    WriteLn(fr, pav, val, ' val. ', min, ' min.');
  Close(fr);
end;
//-------------------------------------
begin
  Duomenys;
  Rasti;
  Rezultatai;
end.
```

Egzamine mokinių parašytų programų pavyzdžiai

Antroji užduotis

P24 programa.

```
program Kelionė;
  var  v1, v2, v3, v4, v5, v6,
       s1, s2, s3, s4, s5, s6,
       t, t1, t2, t3, t4, t5, t6 : real;
begin
  WriteLn (' Įveskite atstumą nuo Vilniaus iki Elektrėnų ');
  ReadLn (s1);
  WriteLn (' Įveskite greitį ');
  ReadLn (v1);
  t1 := s1 / v1;
  WriteLn (' Įveskite atstumą nuo Elektrėnų iki Žiežmarių ');
  ReadLn (s2);
  WriteLn (' Įveskite greitį ');
  ReadLn (v2);
  t2 := s2 / v2;
  WriteLn (' Įveskite atstumą nuo Žiežmarių iki Kauno ');
  ReadLn (s3);
  WriteLn (' Įveskite greitį ');
  ReadLn (v3);
  t3 := s3 / v3;
  WriteLn (' Įveskite atstumą nuo Kauno iki Raseinių ');
  ReadLn (s4);
  WriteLn (' Įveskite greitį ');
  ReadLn (v4);
  t4 := s4 / v4;
  WriteLn (' Įveskite atstumą nuo Raseinių iki Kryžkalnio ');
  ReadLn (s5);
  WriteLn (' Įveskite greitį ');
  ReadLn (v5);
  t5 := s5 / v5;
  WriteLn (' Įveskite atstumą nuo Kryžkalnio iki Klaipėdos ');
  ReadLn (s6);
  WriteLn (' Įveskite greitį ');
  ReadLn (v6);
  t6 := s6 / v6;
  t := t1 + t2 + t3 + t4 + t5 + t6;
  WriteLn (' Bendras laikas ', t);
end.
```

P25 programa.

```
program kelione;
{-----------------------------------------------------}
funcion t (s, v : real) : integer;
funcion s (t, v : real) : integer;
funcion v (s, t : real) : integer;
begin
  var a, b, c, d, e, f : integer;
  Elektrenai (a);
  Ziezmariai (b);
  Kaunas (c);
  Raseiniai (d);
  Kryzkalnis (e);
  Klaipeda (f);
begin
   a := 50.5;
   b :=20;
   c :=22.35;
   d := 80;
   e := 20;
   f := 100.8
 Laikas := Trunc (atstumas / greitis * 60);
 Atstumas := Trunc (greitis * 60);
 Greitis := Trunc (atstumas * 60);
end;
{-----------------------------------------------------}
```

```
 var s, v, t : real;
  Read (a, b, c, d, e, f);
  then v := s / t
  WriteLn (a);
  WriteLn (b);
  WriteLn (c);
  WriteLn (d);
  WriteLn (e);
  WriteLn (f);
end.
```

P26 programa.

```
program Bevardė27;
   Mas [a] := [1,..., 100];
   var SEl, SЮi, SKa, SRas, SKr, SKl, atstumas : real;
       Duom2.txt (text) : real;
begin
   accros := Duom2.txt;
   readln (SEl, SЮi, SKa, SRas, SKr, SKl);
   atstumas := 0;
   atstumas := atstumas + SEl + SЮ + SKa + SRas + SKr + SKl;
   write (atstumas)
   end;
 procedure Laikas (atstumas, greitis : real) :integer;
   begin
     readln (greitis)
     Laikas := Trunc (atstumas / greitis * 60);
     t1 := Trunc (SEl / greitis * 60);
     t2 := Trunc (SЮi / geritis * 60);
     t3 := Trunc (SKa / greitis * 60);
     t4 := Trunc (SRas / greitis * 60);
     t5 := Trunc (SKr / greitis * 60);
     t6 := Trunc (SKl /greitis * 60);
   end;
 procedure rez2.txt
  var rez2.txt : text;
   begin
     accros := rez2.txt;
     wrileln ('Elektrлnai', ' ', t1);
     writwln ('Юieюmariai', ' ', t2);
     writeln ('Kaunas', ' ', t3);
     writeln ('Raseiniai', ' ', t4);
     writeln ('Kryюkalnis', ' ', t5);
     writeln ('Klaipлda', ' ', t6);
   end;
```

P27 programa.

```
program Kelione;
var s1, s2, s3, s4, s5, s6, v, t1, t2, t3, t4, t5, t6 : integer;
    x : mas
function t (s / v : real) : integer;
type mas = array[1..100] of integer;
begin
readln (s1, v)
s1 := 50.5
writeln('t1=');
t1 := s1 / v;
readln(t1);
writeln('t2=');
t2 := s2 / v;
readln(t2);
writeln('t3=');
t3 := s3 / v;
readln(t3);
writeln('t4=');
t4 := s4 / v;
```

```
readln(t4);
writeln('t5=');
t5 := s5 / v;
readln(t5);
writeln('t6=');
t6 := s6 / v;
readln(t6);
writeln('t1=');
t1:=
end.
```

P28 programa.

```
program Kelione;
    var v,s,t:real;
        Elektrenai,Ziezmariai,Kaunas,Raseiniai,Kryzkalnis,Klaipeda:real;
        Function Laikas(atstumas,greitis:real):integer;
begin
    Laikas:=trunc(atstumas/greitis * 60);
end;
begin
    Elektrenai:=50.5;
    Ziezmariai:=70.5;
    Kaunas:=92.85;
    Raseiniai:=172.85;
    kryzkalnis:=192.85;
    klaipeda:=293.65;
end.
```

P29 programa.

```
program Kelionė;
  var spav:array[1..100] of string[15];  //stotelių pavadinimai//
      Laikas:array[1..100] of integer;
      laikas2:array[1..100] of integer;   //laikas išvykimo iš stotelių//
      duom:text;              //duomenų saugojimo failas//
      ss,iivv,iivm,ilm:integer;  //stotelių skaičius,išvykimo iš Vilniaus valanda, išvykimo
iš Vilniaus minutės, išvykimo laikas minutėmis//
      x,y,z:integer;         //tarpiniai kintamieji//
      greitis:real;          //vidutinis autobuso greitis//
      atstumas:array[1..100] of real;      //atstumas tarp stotelių//
{-------------------------------------------------------------}
begin
  assign(duom,'Duom2.txt');
  reset(duom);
  read(duom,ss,greitis,iivv,iivm);
  readln(duom);
  for x:=1 to ss do
      begin
        read(duom,spav[x],atstumas[x]);
        readln(duom)
      end;
  for x:=1 to ss do
      Laikas[x]:=Trunc(atstumas[x]/greitis*60);
      ilm:=iivm+iivv*60;
  for x:=1 to ss do
      begin
        y:=y+Laikas[x];
        laikas2[x]:=ilm+y;
      end;
  for x:=1 to ss do
      begin
        write(spav[x],laikas2[x],'min.');
        writeln
      end;
  readln;
end
```

P30 programa.

```
program A05_2;
var st_sk, v, h, min, laik1, i : integer;
    prad, gal : text;
    miestas : string[15];
    atstum : string[5];
    atstum1 : real;
function Laikas(atstumas, greitis : real) : integer;
begin
  Laikas := Trunc(atstumas / greitis * 60);
end;
begin
  Assign(prad, 'Duom2.txt'); Reset(prad);
  Assign(gal, 'Rez2.txt'); Rewrite(gal);
  Read(prad, st_sk, v, h);
  Readln(prad, min);
{-----------------------------------------------------------------------}
  for i := 1 to st_sk do
    begin
      Read(prad, miestas);
      Readln(prad, atstum);
      val(atstum, atstum1);
      laik1 := Laikas(atstum1, v);
      writeln(atstum1:2:2);
      writeln(laik1);
      if laik1 + min < 60 then min := min + laik1
        else if (laik1 + min) mod 60 = 0 then h := h + (laik1 + min) div 60
          else if laik1 + min > 60 then
            begin
              h := h + (laik1 + min) div 60;
             if (min + laik1) mod 60 + min < 60 then min := min + (min + laik1) mod 60
                 else if (min + laik1) mod 60 + min = 60 then
                      begin
{skaičiuojamas laikas}
                        h := h + 1;
                        min := 0;
                      end
                   else if (min + laik1) mod 60 + min > 60 then
                     begin
                       min :=(min + laik1) mod 60;
                     end;
            end;
{-------------------------------------------------------------------------------}
      Writeln(gal, miestas, h, ' val. ', min, ' min.');
    end;
  Close(prad);
  Close(gal);
  readln;
end.
```

P31 programa.

```
program A10_2;
 var f1, f2: text;
   mst: string[15];
   x, v, h, m, i, t: integer;
   s, s2: real;

 function Laikas (atstumas, greitis : real) : integer;
 begin
   Laikas := trunc (atstumas / greitis * 60);
 end;

 begin
   assign (f1, 'Duom2.txt');
   assign (f2, 'Rez2.txt');
   reset (f1);
   rewrite (f2);
   readln (f1, x, v, h, m);
   for i := 1 to x do
```

```
   begin
     read (f1, mst);
     readln (f1, s2);
     s := s + s2;
     t := Laikas (s, v) + h * 60 + m;;
     writeln (f2, mst, t div 60, ' val. ', t mod 60, ' min.');
   end;
   close (f1);
   close (f2);
end.
```

P32 programa.

```
Program a07_2;
  const il = 100;
  type stotele = record
                  stot : string [15];                    {stotelės pavadinimas}
                  atst : real;                           {atstumas}
                  valan,                                 {valandų skaičius}
                  minut : integer;                       {minučių skaičius}
                end;
       marsr = array [1..il] of stotele;
{------------------------skaitymas------------------------}
  procedure skaityti (var A : marsr; var v : real; var n, val, min : integer);
     var d : text;
         i : integer;
  begin
    Assign (d, 'Duom2.txt');
    Reset (d);
    ReadLn (d, n, v, val, min);
    For i := 1 to n do
      ReadLn (d, A[i].stot, A[i].atst);
    Close (d);
  end;
{------------------------laikas---------------------------}
  function Laikas (atstumas, greitis : real) : integer;
  begin
    Laikas := Trunc (atstumas / greitis * 60);            {sveikoji skaičiaus dalis}
  end;
{----------------------skaičiavimas----------------------}
  procedure skaiciuoti (var A : marsr; v : real; n, tpr: integer);
    var i, tkel : integer;
  begin

    For i := 1 to n do
      begin
       tkel := Laikas (A[i].atst, v);                     {kelionėje sugaištas laikas}
       tkel := tpr + tkel;                                {atvykino i stotelę laikas}
       tpr := tkel;
       A[i].valan := tkel div 60;                         {valandų skaičius}
       A[i].minut := tkel mod 60;                         {minučių skaičius}
      end;
  end;
{------------------------rašymas--------------------------}
  procedure rasyti (A : marsr; n : integer);
     var r : text;
         i : integer;
  begin
    Assign (r, 'Rez2.txt');
    Rewrite (r);
    For i := 1 to n do
      WriteLn (r, A[i].stot, A[i].valan, ' val. ', A[i].minut, ' min.');
    Close (r);
  end;
{----------------PAGRINDINĖ PROGRAMA---------------------}
  var A : marsr;
      v : real;
      n, val, min, tpr : integer;
Begin
  skaityti (A, v, n, val, min);
  tpr := val * 60 + min;                                  {pradinis laikas minutemis}
  skaiciuoti (A, v, n, tpr);
```

```
  rasyti (A, n);
End.
```

P33 programa.

```
program Kelionė;

type stotele = record
            stotel   : string[15];
            atstum   : real;
            val, min : byte;
          end;
        masyvas = array [0..100] of stotele;

var  mas: masyvas;
     i, n, greitis, min : integer;
     F : text;

{----------------------------------}

procedure duomenys (var n, greitis: integer);
begin
  readln (F, n, greitis, mas[0].val ,mas[0].min);
  for i:=1 to n do
     readln (F, mas[i].stotel, mas[i].atstum);
end;

{----------------------------------}

function Laikas (atstumas, greitis : real) : integer;
begin
  Laikas := Trunc (atstumas / greitis * 60);
end;

{----------------------------------}

procedure skaiciavimas (i: integer);
begin
  min:=laikas(mas[i].atstum, greitis);
  mas[i].min:=mas[i-1].min + min;
  mas[i].val:=mas[i-1].val + mas[i].min div 60;
  mas[i].min:=mas[i].min mod 60;
end;

{----------------------------------}

procedure rezultatai;
begin
  for i:=1 to n do
     writeln (F, mas[i].stotel,mas[i].val, ' val. ' ,mas[i].min, ' min.');
end;

{----------------------------------}

begin
  assign( F, 'Duom2.txt');  reset( F );
  duomenys (n, greitis);
  close (F);

  for i:= 1 to n do
    skaiciavimas(i);

  assign( F, 'rez2.txt');  rewrite( F );
  rezultatai;
  close (F);

end.
```

P34 programa.

```
program kelione;

   type valmin = record
                 val: integer;     {laiko duomenys valandomis ir minutemis}
                 min: integer;
                 end;

      duomenys = record
                 pavadinimas: string[15];     {kiekvienos stoteles duomenys}
                 atstumas: real;
                 laik: valmin;
                 end;

      mas = array [1..100] of duomenys;

{==============================================================================}
 {ivedimo procedura, nuskaitomas pavadiniams, kiekis, greitis, isvykimo laikas}

 procedure Ivesti (var DD : text; var A : mas; var n : integer;
                  var greitis : real; var isvykimas : valmin);

var i : integer;
 begin
 Assign (DD, 'duom2.txt'); Reset(DD);
 Readln (DD, n, greitis, isvykimas.val, isvykimas.min);
 for i:= 1 to n do
   Readln (DD, A[i].pavadinimas, A[i].atstumas);
 Close (DD);
 end;

{==============================================================================}
  {sugasito laiko minutemis skaiciavimas}

 function Laikas (atstumas, greitis: real) : integer;
  begin
   Laikas := Trunc (atstumas / greitis * 60);
 end;

{==============================================================================}
 {kiekvienos stoteles laiko skaiciavimas valandomis ir minutemis}

procedure Skaiciuoti (var A : mas; isvykimas : valmin; n : integer; greitis : real);

 var i, minutes : integer;
 begin
 minutes := 0;
 for i := 1 to n  do begin
                      minutes := Laikas(A[i].atstumas, greitis) + isvykimas.min;
                      A[i].laik.val := minutes div 60 + isvykimas.val;
                      A[i].laik.min := minutes mod 60;
                     isvykimas.min := A[i].laik.min; isvykimas.val := A[i].laik.val;
                     end;
end;

{==============================================================================}
  {rasymas i byla}
Procedure Spauzdinti(var FF : text; n : integer; A: mas);

 var i : integer;

begin
 Assign (FF, 'Rez2.txt'); Rewrite (FF);
 for i := 1 to n do
    Writeln (FF, A[i].pavadinimas, A[i].laik.val,' val. ', A[i].laik.min,' min. ');
 Close (FF);
end;

{==============================================================================}
```

```
var Duom, Rezultatai : text; Stoteles : mas; kiekis : integer;
    greitis : real; isvykimas : valmin;

{==============================================================================}

begin
 Ivesti (Duom, Stoteles, kiekis, greitis, isvykimas);
 Skaiciuoti (Stoteles, isvykimas, kiekis, greitis);
 Spauzdinti (Rezultatai, kiekis, Stoteles);
 Readln;
end.
```

P35 programa.

```
program A09_2;
type
    TDuomData = record
        Kur: string[15];
        Atstumas: real;
    end;

    TRezData = record
        Kur: string[15];
        Min: integer;
    end;
    TPirmiDuom = record
        stot, min: integer;
        speed: real;
    end;
    TDuom = array[1..100] of TDuomData;
    TRez = array[1..100] of TRezData;

var
    Duom: TDuom;
    Rez: TRez;
    Pradinfo: TPirmiDuom;
    failas: text;

    { laiko skaiciavimas }
function Laikas(atstumas, greitis : real) : integer;
    begin
        Laikas := Trunc (atstumas / greitis * 60);
    end;

    { skaitom duomenis is failo }
procedure MyReadData(fil: string; var Dm: TDuom; var Prad: TPirmiDuom);
var
    k, val, min: integer;
    begin
        assign(failas, fil);
        reset(failas);

        readln(failas, Prad.stot, Prad.speed, val, min);
        Prad.min := val * 60 + min;

        for k := 1 to Prad.stot do
            begin
                readln(failas, Dm[k].Kur, Dm[k].Atstumas);
            end;
        close(failas);
    end;

    { rasom duomenis i faila }
procedure MyWriteData(fil: string; Rz: TRez; Prad: TPirmiDuom);
var
    k: integer;
    begin
        assign(failas, fil);
        rewrite(failas);
        for k := 1 to Prad.stot do
         begin
```

```
         writeln(failas,Rz[k].Kur:15,Rz[k].Min div 60,' val. ',Rz[k].Min mod 60,' min.');
         end;
        close(failas);
    end;
    { skaiciuojam kiek uztruks minuciu iki kiekvienos stoties }
procedure MyCalc(Dm: TDuom; var Rz: TRez; Prad: TPirmiDuom);
var
    laik, k: integer;
    begin
        laik := Prad.min;
        for k := 1 to Prad.stot do begin
            laik := laik + Laikas(Dm[k].Atstumas, Prad.speed);
            Rz[k].Kur := Dm[k].Kur;
            Rz[k].Min := laik;
        end;
    end;
begin
    MyReadData('Duom2.txt', Duom, Pradinfo);
    MyCalc(Duom, Rez, Pradinfo);
    MyWriteData('Rez2.txt', Rez, Pradinfo);
end.
```