Kauno technologijos universitetas

Jaunųjų kompiuterininkų mokykla

PRADINĖ PAŽINTIS SU TURBO PASKALIU

Jonas Blonskis,
Antanas Vidžiūnas,
Jolanda Ziberkienė.

Kaunas 1996.

Anotacija

Metodinė priemonė sudaro įvadinės teorinės ir praktinės žinios apie programavimo kalbą ir programavimo techniką. Kalbos elementai, algoritmavimo ir programavimo klausimai aprašomi nuosekliai po truputį įvedant sudėtingesnius elementus. Medžiaga suskirstyta į 12 temų. Tai nebūtinai vieno užsiėmimo medžiaga. Kiekvienas pagal savo sugebėjimus ir galimybes skyrelio medžiagą gali įsisavinti per jam tinkamą laikotarpį.

Kiekvienos temos tikslas yra paaiškinti tam tikrą kalbos elementų grupė, naują algoritmą. Temos apima bazinius duomenų tipus. Supažindinama su viena paprasčiausių struktūrinių duomenų forma masyvais. Duomenų ir skaičiavimo rezultatų išsaugojimiu naudojami tekstiniai failai. Analizuojamos Turbo Paskalio kalbos priemonės, skirtos ekrano valdymui (biblioteka Crt), klaviatūros apklausai, garsams. Susipažindinama su pagrindinėmis Turbo Paskalio aplinkos paprogramėmis, programų sudarymo pagrindiniu elementu - procedūra.

Teoriniai elementai iliustruojami programomis, kuriose siūloma besimokančiajam nedideliais pakeitimais bei papildymais patobulinti programą ir padaryti ją patrauklesnė.

Kiekvienos temos pabaigoje pateikiama: analizuojamos medžiagos santrauka, klausimai, kontrolinės užduotys. Smalsesniam ir darbštesniam mokiniui yra pasiūloma savarankiško darbo užduotis.

Atskirai pateikiama algoritmų išraiškos priemonių santrauka bei samprata apie programavimo kultūrą.

TURINYS

1 tema. Turbo Paskalio kalba ir aplinka

1.1. Programos struktūra

Programos tekstas Turbo Paskalio kalba:

program < Programos pavadinimas >;
    < Aprašai >
begin
    < Pagrindinė dalis >
end.

Simboliai < > aprašyme nurodo paaiškinimus ne Paskalio kalba.

Programoje galima rašyti veiksmus aiškinantį tekstą. Jis skiriamas žmogui ir vadinamas komentarais. Programos darbui komentarai neturi įtakos ir yra ignoruojami. Jie rašomi tarp specialių simbolių:

{ <Komentarai> }
(* <Komentarai> *)

1.2. Programos tekstas

Programos kompiuteriui nurodo, kaip reikia spręsti jų aprašomą uždavinį. Jos yra rašomos programavimo kalbos sakiniais. Surinkite Turbo Paskalio aplinkos redaktoriaus lange programą **Pirmoji**. Rinkdami programos tekstą komentarus galite praleisti.

Pradžia Pirmoji
x, y realūs skaičiai
Įvesti x reikšmę
y := x^2
Išvesti y reikšmę
Pabaiga

```
program Pirmoji;                    { Antraštė                          }
var     x, y : real;                { Kintamųjų aprašai                 }
begin                               { Pagrindinės dalies pradžia        }
                    { Duomenų išvedimo į ekraną sakinys                 }
    WriteLn('Jus sveikina Turbo Paskalio aplinka');
    WriteLn('Įvesk skaičių:');{   Užklausa įvedimui                     }
    ReadLn (x);                 {   Įvedimas                            }
    y:= x*x;                    {   Skaičiavimų aprašymas               }
    WriteLn('Kvadratas: ', y:10:2); {       Rezultatų išvedimas         }
    ReadLn;                 { Laukti kl. Enter paspaudimo               }
end.                                { Pabaigos sakinys                  }
```

1.3. Programos darbo patikrinimas

Norint išbandyti redaktoriaus lange esančios programos darbą, reikia kartu paspausti klavišų porą **Ctrl** ir **F9** (sutrumpintai tai yra žymima užrašu **Ctrl+F9**). Šiais klavišais parenkama aplinkos komanda **Run**, kuri nurodo, kad redaktoriaus lange esančią programą reikia transliuoti ir vykdyti. Įvykdžiusi programą, Turbo Paskalio aplinka iš karto grąžina valdymą savo redaktoriui, kuriam skirtas ekrano langas uždengia programos darbo rezultatus. Todėl programų gale dažnai yra įrašomas sakinys *ReadLn*, kuris stabdo programos darbą ir laukia, kol bus paspaustas klavišas **Enter**. Į rezultatų langą taip pat galima pereiti naudojant ekrano langų perjungimo komandą - **Alt+F5**. Atgal į redaktoriaus langą sugrįžtama paspaudus bet kurį klavišą.

1.4. Pagrindinės programų sudarymo taisyklės

Programos rašomos Turbo Paskalio kalbos sakiniais. Reikalavimai programų tekstui:

- kiekviena programa pradedama antraštės sakiniu, kurio sintaksė (rašymo taisyklės):

  program <Programos vardas>;

- visi veiksmai aprašomi pagrindinėje programos dalyje, kurios ribos nurodomos baziniais kalbos žodžiais *begin* ir *end* (operatoriniais skliaustais);
- bet kurioje programos vietoje, kur gali būti tarpai, galima įterpti jokios įtakos programos darbui neturinčius komentarus;
- komentarų ribos žymimos simbolių grupėmis (* *) arba skliaustais { };
- visi sakiniai, išskyrus komentarus, vienas nuo kito atskiriami kabliataškiais.

1.5. Skaičiavimų aprašymas

Skaičiavimai aprašomi priskyrimo sakiniais, kurių struktūra:

<Kintamasis>:=<Aritmetinė išraiška>;

Aritmetinėse išraiškose yra nurodoma, kokius veiksmus, kokia tvarka ir su kokiais argumentais reikia atlikti. Jų argumentai gali būti kintamieji, konstantos ir funkcijos. Kintamieji nurodomi vardais, kurie sudaromi tik iš lotyniškų raidžių ir skaitmenų, bet būtinai turi prasidėti raide (raidėms taip pat priklauso pabraukimo simbolis "_"). Vardų pavyzdžiai:

x, y, s1, prad_suma, PradSuma.

Visi programoje vartojami kintamieji privalo būti aprašyti baziniu žodžiu *var* pažymėtoje aprašų dalyje. Kintamųjų aprašų sintaksė:

<Vardų sąrašas>: <Tipas>.

Kintamųjų tipai nurodomi programavimo kalbos baziniais žodžiais. Pavyzdžiui, skaitmeniniams duomenims, kurie gali turėti trupmeninę dalį, yra skirtas tipas *real*. Trupmeninė dalis nuo sveikosios programavimo kalbose atskiriama dešimtainiu tašku. Vardų sąrašo elementai atskiriami kableliais.

Pagrindinių operacijų simboliai:

+ (sudėtis), – (atimtis), * (daugyba), / (dalyba).

Jie suskirstyti į dvi prioritetų (pirmumo) grupes:

pirma grupė	*	/,
antra grupė	+	–.

Aritmetinės operacijos vykdomos pirmumo tvarka, o vienodo pirmumo veiksmai - iš kairės. Veiksmų tvarką galima pakeisti skliaustais.

Priskyrimo sakinio pavyzdys: y:= x*(5/2) - 7.14*(x/2);

Išraiškų argumentais gali būti tik tokie kintamieji, kurių reikšmės jau yra žinomos, apibrėžtos anksčiau. Kintamųjų reikšmės apibrėžiamos priskyrimo arba įvedimo sakiniais. Skyrelio 1.2 programoje tai padaryta įvedimo sakiniu **ReadLn**.

1.6. Programos ryšys su išore

Kiekviena programa turi palaikyti ryšį su išore, išorine aplinka: gauti iš ten duomenis ir pateikti rezultatus. Personaliniuose kompiuteriuose labai populiari dialoginio ryšio su išore forma, kai programa rašo ekrane pranešimą apie savo pageidavimus, stabdo savo darbą ir laukia, kol vartotojas klaviatūroje surinks atsakymą. Ekrane rašomi programos pageidavimai yra vadinami užklausomis. Jos formuojamos rašymo sakiniais:

WriteLn ('<Užklausos tekstas>');

Sakiniu **WriteLn** į ekraną siunčiamus tekstus būtina rašyti tarp pavienių kabučių (apostrofų). Taip sudaryta struktūra vadinama tekstiniais duomenimis, simbolių eilutėmis arba tiesiog eilutėmis. Jose gali būti visi kompiuterio alfabeto simboliai. Sutarta, kad apostrofo simbolis tekste rašomas dvigubas. Kintamųjų, pačių programų ir kitų programos objektų vardus galima rašyti tik lotyniškomis raidėmis.

Už užklausos sakinio programoje turi būti rašomas skaitymo iš klaviatūros sakinys, kuris stabdo kompiuterio darbą ir laukia surinktų duomenų. Skaitymo sakinio sintaksė:

ReadLn(<Kintamųjų sąrašas>);

Sąrašas nurodo, kiek reikšmių klaviatūroje reikia surinkti ir kokiems kintamiesiems jas suteikti. Reikšmes galima surinkti vienoje ekrano eilutėje, atskiriant tarpais, arba galima jas rašyti atskirose eilutėse. Labai svarbi yra reikšmių surinkimo tvarka. Klaviatūroje surinktos eilutės pabaiga yra žymima klavišu **Enter**. Užklausos pavyzdys:

```
WriteLn('Įvesk x:');
ReadLn(x);                    { Programa lauks x reikšmės }
```

1.7. Rezultatų pateikimas

Programos darbo rezultatai siunčiami į ekraną rašymo sakiniu **WriteLn**, kurio atskiras atvejis buvo aprašytas anksčiau. Bendru atveju, jo sintaksė yra tokia:

WriteLn(<Į ekraną siunčiamų duomenų sąrašas>);

Sąrašo elementais gali būti kintamieji, išraiškos ir įvairių tipų konstantos. Sakinys nuosekliai rašo ekrane visų savo argumentų sąrašo elementų reikšmes. Pavyzdžiui, sakiniai

```
y:= 2;          WriteLn('y= ', y:10:2);
```

užrašys ekrane **y= 2.00**, nes sakinio **WriteLn** argumentų sąraše yra du elementai: eilutės tipo konstanta '*y*= ' ir kintamasis *y*. Eilutės tipo konstantų ir kintamųjų vardų komponavimas rašymo sąrašuose yra labai patogi ir vaizdi programos darbo rezultatų aprašymo priemonė.

Nurodant išvedimo sąrašuose *real* tipo reikšmes, rekomenduojama jas rašyti kartu su išvedimo šablonais, kurie aprašo reikšmėms skiriamus ekrano laukus:

<Duomenų elementas>: <Lauko ilgis>: <Trupmeninė dalis>

Šablonas aprašo, kokio dydžio ekrano laukas yra skiriamas visam skaičiui (Lauko ilgis) ir trupmeninės dalies vaizdavimo tikslumą (Trupmeninė dalis).

Pavyzdžiui, sakinys **WriteLn (x:10:2);** nurodo, kad *x* reikšmei ekrane skiriamas 10 simbolių laukas, kuriame turi būti rašoma dviejų ženklų po kablelio tikslumu.

1.8. Programos modifikavimas

Sudarytą programą visuomet galima modifikuoti, pritaikyti giminingų uždavinių sprendimui. Pavyzdžiui, 1.2 skyrelyje pateiktą programą galima lengvai pritaikyti skaičiavimams pagal įvairias formules. Tam tereikia pakeisti kintamojo *y* skaičiavimui skirtą išraišką ir pranešimo apie rezultatus paaiškinimus.

- Pakeiskite 1.1 skyrelio programą taip, kad ji galėtų skaičiuoti įvesto skaičiaus kubą ir įvairių 1.1 lentelėje pateiktų funkcijų (pavyzdžiui, *Sin(x)*, *Sqrt(x)*) reikšmes. x - klaviatūra įvedama reikšmė.

Kreipiniai į funkcijas gali būti išraiškų argumentais. Pačios išraiškos taip pat gali būti funkcijų argumentais.

- Pertvarkykite 1.1 skyrelio programą, kad ji skaičiuotų pagal formulę
  y:= Sqrt(1+x*x*x).

1.1 lentelė. Pagrindinės aritmetinės funkcijos

Tradicinė forma	Paskalyje	Pastaba
arctg x	Arctan(x)	Rezultatas radianais
cos x	Cos(x)	Argumentas radianais
sin x	Sin(x)	Argumentas radianais
e^x	Exp(x)	
$\sqrt{x}$	Sqrt(x)	
ln x	Ln(x)	
	Frac(x)	Trupmeninė argumento dalis
	Int(x)	Sveikoji argumento dalis
|x|	Abs(x)	

1.9. Programos su keliais argumentais

Jei programos darbui yra reikalinga grupė argumentų, galima rašyti užklausas kiekvienam argumentui atskirai arba vieną bendrą užklausą visiems argumentams. Pavyzdžiui, tarkime, kad reikia sudaryti programą trijų skaičių ***x, y, z*** aritmetinio vidurkio ***av*** ir tų skaičių sandaugos trečdalio ***gv*** skaičiavimui. Užklausa argumentams gali būti užrašoma taip:

```
WriteLn(' Įveskite tris skaičius:');        ReadLn(x, y, z);
```

Skaičiavimo sakiniai užrašomi atitinkamai:

```
av:= (x + y + z) / 3;        gv:= (x * y * z) / 3;
```

- Sudarykite *av* ir *gv* skaičiavimo programą su bendra visų argumentų užklausa ir jos modifikaciją su atskiromis užklausomis kiekvienam argumentui.

1.10. Užduotis savarankiškam darbui

Sudaryti stačiakampio gretasienio tūrio skaičiavimo programą. Argumentai - kraštinių ilgiai.

Programoje turi būti užklausa argumentams, skaičiavimus ir rezultatų pateikimą aprašantys sakiniai, išsamūs komentarai.

1.11. Ką turėjau išmokti?

Programos tekstą Turbo Paskalio kalboje sudaro trys dalys: programos antraštė, aprašai ir pagrindinė dalis.

Komentaras rašomas, bet kurioje programos vietoje, kur galima rašyti tarpo simbolį, specialiuose simboliuose { } arba (* *) ir neturi įtakos programos vykdymui.

Surinkus programos tekstą, programos darbą galima patikrinti, nuspaudus klavišų kombinaciją **Ctrl+F9**, arba komandą **RUN** ir nuspaudus **Enter** klavišą.

Norint patekti į TP rezultatų langą, spaudžiame klavišų kombinaciją **Alt+F5**.

Programos antraštės struktūra:

program <Programos vardas>;

Programos vardas gali būti sudarytas tik iš lotyniškų raidžių ir skaitmenų. Simbolis "_" (pabraukimo) priskiriamas prie raidžių. Pirmas simbolis būtinai turi būti raidė.

Pagrindinė dalis prasideda žodžiu *begin*, o baigiasi *end*., kur *begin* ir *end* vadinami operatoriniais skliaustais.

Visi sakiniai vienas nuo kito skiriami kabliataškiais.

Skaičiavimai programoje užrašomi priskyrimo sakiniais, kurių bendra struktūra yra:

<Kintamasis> := <Aritmetinė išraiška>; .

Kintamieji yra žymimi vardais, kurie sudaromi taip pat kaip programos vardai.

Aritmetinėse išraiškose gali būti kintamieji, konstantos, funkcijos, sujungtos aritmetinių operacijų ženklais bei skliaustais.

Visi programos pagrindinėje dalyje naudojami kintamieji, turi būti aprašyti aprašų dalyje *var* tokio tipo sakiniais:

<Vardų sąrašas> : <Tipas>;

Kintamųjų tipai nurodomi Paskalio kalbos baziniais žodžiais. Pvz.: realūs kintamieji - aprašomi žodžiu *real.* (Realūs kintamieji tai tokie, kurių reikšmės gali įgyti trupmenines reikšmes. Sveikoji dalis nuo trupmeninės skiriama dešimtainiu tašku.)

Skaitymo sakinio sintaksė: *ReadLn*(<Kintamųjų sąrašas>). Kintamųjų sąrašo elementus vieną nuo kito Paskalio kalboje atskiriame kableliais.

Rašymo sakinys naudojamas rezultatams siųsti į ekraną. Jo struktūra: *WriteLn*(<Siunčiamų duomenų sąrašas>). Sąrašo elementais gali būti kintamieji, tekstas, išraiškos, konstantos.

Rašymo sakinyje gali būti naudojami išvedimo šablonai, kurie aprašo reikšmėms skiriamų ekrano laukų dydžius. Šablonai rašomi prie duomenų elemento taip:

<Duomenų elementas>:<Bendras lauko ilgis>;

Atskiru atveju, kai į ekraną siunčiami realūs skaičiai, dar yra nurodomas ir trupmeninės dalies ilgis:

<Duomenų elementas>:<Bendras lauko ilgis>:<Trupmeninės dalies lauko dydis>;

Įvedant pradinius duomenis į kompiuterio atmintį, yra formuojamos užklausos. Joms sudaryti naudojamas rašymo ir skaitymo sakinys:

WriteLn(‘<Užklausos tekstas>‘);
ReadLn(<Kintamojo vardas>);.

Užklausos tekste gali būti panaudoti, bet kokie klaviatūros simboliai. Jeigu programos darbui reikalinga grupė argumentų, tai užklausos gali būti rašomos kiekvienam atskirai arba visiems bendrai.

1.12. Kontroliniai klausimai

1. Iš kokių dalių susideda programa Turbo Paskalio kalboje?
2. Kas būdinga programos aprašų daliai?
3. Kaip sudaromi vardai?
4. Kur ir kaip rašomi komentarai?
5. Kaip pamatyti skaičiavimo rezultatus, jei programos tekstas uždengia rezultatų langą?
6. Kaip vykdyti programą?
7. Ką vadiname operatoriniais skliaustais?
8. Kaip užrašomi skaičiavimo sakiniai?
9. Kaip aprašomi kintamieji?
10. Iš kokių elementų sudaromos aritmetinės išraiškos?
11. Kokia yra veiksmų tvarka aritmetinėje išraiškoje?
12. Kaip pakeisti veiksmų tvarką aritmetinėje išraiškoje?
13. Kokia yra rašymo sakinio bendra struktūra?
14. Kas yra duomenų išvedimo šablonas?
15. Kokia yra skaitymo sakinio struktūra?
16. Kokius žinote variantus įvesti į kompiuterio atmintį kelių kintamųjų reikšmes?
17. Kam reikalingos programose užklausos?

1.13. Kontrolinės užduotys

1. Kurie iš žemiau pateiktų programos antraštės sakinių yra teisingi?

a) *program* Suma.
b) *Program* suma_S1;
c) *Program* S1;
d) *program* a
e) *program* a 1;
f) *program* bendrdiddalik;

2. Rasti klaidas programos tekste.

program antroji { programos pradžia};

```
                                    (*  testo užduotis}
    var sk  :  real
     begin
       WriteLn(Jus sveikina kompiuteris!);
       WriteLn('Įveskite savo skaičių ')        ReadLn(sk);
       WriteLn(' Jūs įvedėte: ', sk:5:0');
       WriteLn('Ačiū, kad bendravote.')
     end.
```

3. Kurių aritmetinių išraiškų reikšmės bus vienodos?

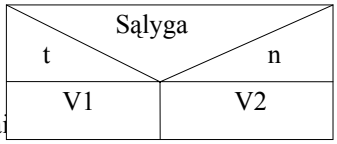
a) x := 3*a*b-4*a*c;
b) x := a*(4*c - 3*b);
c) x := 3*(a*b) - 4*(a*c);
d) x := 12*(a*b - a*c);
e) x := a*(3*b - 4*c);

2 tema. Sąlygos sakiniai ir ciklai (valdančios struktūros)

2.1. Sąlygos sakinys

Valdančios struktūros leidžia aprašyti pageidaujamą veiksmų tvarką, alternatyvų parinkimą ir veiksmų kartojimą. Populiariausia valdanti struktūra yra alternatyvų parinkimą aprašantis sąlygos sakinys, kurio sintaksė:

if <Sąlyga> *then* <V1>
[*else* <V2>]



Paprasčiausios sąlygos yra aprašomos santykiai

<i1> <Santykio operacija> <i2>

Čia **i1** ir **i2** - aritmetinės išraiškos, o santykio operacija gali būti žymima tokiais simboliais:

=	lygu	<	mažiau	<=	mažiau arba lygu
<>	nelygu	>	daugiau	>=	daugiau arba lygu

Jei išraiškų reikšmės tenkina santykio operaciją, santykiui suteikiama loginė reikšmė **True** (tiesa), o jei ne - **False** (melas).

Jeigu sąlygos reikšmė yra *true* (sąlyga tenkinama), sakinyje parenkama vykdymui alternatyva **V1**, jeigu reikšmė *false* - **V2**.

Taigi sakinys ***if*** aprašo, kaip parinkti vykdymui vieną iš dviejų galimų alternatyvų - **V1** arba **V2** (alternatyvos aprašomos sakiniais). Jeigu sakinyje ***if*** šaka *else* praleista, jis aprašo vieno sakinio (**V1**) vykdymo sąlygas.

2.2. Argumentų kontrolė

Dažnai tenka tikrinti argumentų priklausymo leistinai reikšmių atkarpai sąlygas. Pavyzdžiui, kvadratinė šaknis gali būti skaičiuojama tik teigiamoms argumentų reikšmėms. Todėl šaknis skaičiuojančiose programose reikėtų tikrinti, ar argumento reikšmė yra leistina. Kaip tai yra daroma, parodyta programoje **Trauk**.

```
program Trauk;
var   x, y: real;                    { Argumentas ir rezultatas }
begin
   WriteLn ('Įveskite argumento reikšmę');{ Įvedimo užklausa   }
   ReadLn(x);                        { Įvedimas                 }
   if (x>0)                          { Argumento kontrolė       }
    then WriteLn ('Šaknis: ', Sqrt(x):6:2)
    else      WriteLn ('Neleistinas argumentas');
end.
```

Jei šioje programoje argumento kontrolės nebūtų, įvedus neigiamą skaičių, programos darbas būtų nutraukiamas ir ekrane matytųsi pranešimas apie jos vykdymo klaidą.

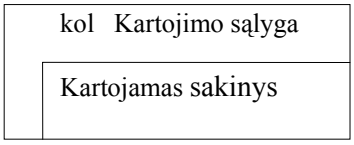
- Savarankiškai sudarykite programą, kuri iš trijų įvestų *skaičių x, y* ir *z* atrinktų didžiausią ir jo reikšmę suteiktų kintamajam *max*. Tokioje programoje turi būti du sakiniai ***if***. Pirmasis turi atrinkti didesniąją reikšmę iš dviejų, o antrasis turi lyginti atrinktąją reikšmę su trečiuoju skaičiumi:

```
if   x>y     then    max:= x
             else    max:= y;     { Didesnis iš dviejų      }
if   z>max   then    max:= z;     { Didžiausias iš trijų    }
```

2.3. Ciklai while

Programa **Trauk** yra nelabai gera, nes, įvedus blogą argumentą, nutraukiamas jos darbas. Geriau kai, įvedus neigiamą skaičių, argumento užklausa kartojama tol, kol įvedamas šaknies skaičiavimo argumentui leistina teigiama reikšmė. Tai galima padaryti su ciklo sakiniu ***while***, kurio sintaksė:

while <Kartojimo sąlyga> *do*
    <Kartojamas sakinys>



Šis sakinys nurodo, kad už žodelio *do* įrašytas sakinys privalo būti kartojamas tol, kol tenkinama kartojimo sąlyga. Jeigu norime kartoti grupę sakinių, juos reikia įrašyti tarp baziniais žodžiais ***begin*** ir ***end*** žymimų

operatorinių skliaustų (programa **Control**). Norint, kad ciklas netaptų begaliniu, būtina, kad kartojimo sąlyga priklausytų nuo kartojamo sakinio rezultatų.
Programoje **Control** ciklas sudarytas tam, kad užklausa argumentui būtų kartojama tol, kol įvedama neleistina neigiama argumento reikšmė. Surinkite ir patikrinkite šią programą:

```
program Control;
var     x, y: real;                 { Argumentas ir rezultatas    }
begin
                                    { Įvedimo užklausa            }
WriteLn ('Įveskite iš pradžių neigiamą, o po to teigiamą skaičių');
    ReadLn(x);                      { Įvedimas                    }
    while (x<0) do    begin         { Užklausos kartojimas      }
        WriteLn (' Kartokite, argumentas neigiamas');
        ReadLn(x);
    end;
    WriteLn ('Šaknis: ', Sqrt(x):6:2);
end.
```

2.4 Pratimas

Tarkime, kad reikia skaičiuoti aukštyn mesto kūno trajektoriją pagal išraiškas: $\mathbf{x = a*t;\ y = b*t - g*t^2/2}$, kur koeficiantai **a** ir **b** bei pagreitis **g** yra įvedami užklausos būdu iš klaviatūros. Trajektorijos taškų koordinates siūloma skaičiuoti laiko **t** reikšmėms 0, 1, 2, 3, ir t.t., kol kūnas nukris ant žemės.

Parašykite programą duotam algoritmui **Trajektorija**, suderinkite programą ir patikrinkite gaunamus rezultatus. Parinkite išvedamiems razultatams vaizdžią formą.

Pradžia. Trajektorija;
a, b, g, t, x, y realūs skaičiai
Įvesti a, b, g reikšmes
t := 0; x := a * t; y := b*t – g*t^2 / 2;
kol y > 0
Išvesti x, y, t reikšmes
t := t+1; x := a * t; y := b*t – g*t/2;
Pabaiga

2.5. Užduotis savarankiškam darbui

- Sudarykite programą kvadratinės lygties $\mathbf{ax^2 + bx + c = 0}$ skaičiavimui. Čia duomenys yra lygties argumentai, o rezultatai yra jos šaknys. Prieš ieškant šaknų reikia patikrinti, ar jos egzistuoja.
- Koordinačių plokštumoje yra padėta daug taškų. Jų koordinatės (x, y) yra žinomos. Bet kuriuos tris taškus jungiant atkarpomis galime gauti trikampį. Reikia nustatyti kuriuos taškus sujungus gaunami trikampiai ir kokie taškų rinkiniai negali būti panaudoti trikampio sudarymui. Reikia parašyti programą, kuri nustatytų, ar nurodyti trys taškai gali būti trikampio viršūnėmis, ar negali. Taškų koordinatės įvedamos iš klaviatūros šešių skaičių grupėmis (pirmo taško x, y, po to antro ir trečio). Įvedamų skaičių srauto pabaiga nurodoma šešiais skaičiais, lygiais nuliui.
- Papildykite programą veiksmais, kuriais būtų suskaičiuojamas galimo trikampio plotas ir ekrane išvedama informacija apie tą trikampį: kraštinių ilgiai ir plotas.

2.6. Ką turėjau išmokti?

Populiariausia valdanti struktūra, alternatyvų parinkimui, yra sąlygos sakinys, kurio sintaksė:

if <sąlyga> *then* <V1>
[*else* <V2>];

kur V1 ir V2 yra alternatyvos, aprašomos sakiniais.

Sąlygos sintaksė yra tokia:

<i1> <Santykio operacija> < i2>,

kur i1 ir i2 - aritmetinės išraiškos, o santykio operacija gali būti žymima simboliais:

= lygu,	< mažiau,	<= mažiau lygu,
<> nelygu,	> daugiau,	>= daugiau lygu.

Jeigu i1 ir i2 reikšmės tenkina santykio operaciją, tai santykiui suteikiama loginė reikšmė **True** (tiesa) ir vykdoma alternatyva V1, jei ne, santykiui suteikiama reikšmė **False** (melas) ir vykdoma alternatyva V2.

Šaka *else* šiame sakinyje gali būti praleista, ir tuomet alternatyva V1 bus vykdoma tik tada, kai santykis įgys reikšmę **True**.

Sąlygos sakinys dažnai yra vartojamas argumentų kontrolei, nes ne visuomet iš klaviatūros įvedami skaičiai tinka tolimesniems skaičiavimams. Pvz.: kvadratinės šaknies negalima ištraukti iš neigiamo skaičiaus, o dalinti negalima iš nulio.

Kita valdanti struktūra yra ciklo sakinys. Ciklai gali būti kelių skirtingų tipų. Plačiausiai vartojami ciklai, kurių aprašymo sakinio sintaksė yra tokia:

while <Kartojimo sąlyga> *do* <Kartojamas sakinys>;

Šis sakinys reiškia, kad už žodelio *do* nurodytas veiksmas bus kartojamas tol, kol tenkinama kartojimo sąlyga. Jeigu norime kartoti ne vieną, o daugiau sakinių, juos būtina rašyti tarp operatorinių skliaustų (žr. 1 pamoką). Taip pat reikia žiūrėti, kad ciklas netaptų begalinis, t.y., kad kartojimo sąlyga priklausytų nuo kartojimo sakinio rezultatų.

2.7. Kontroliniai klausimai

1. Kokius žinote valdančių struktūrų sakinius?
2. Kokia yra sąlygos sakinio sintaksė?
3. Kokia yra sąlygos sintaksė?
4. Ką reiškia V1 ir V2?
5. Kokie yra santykio operacijos ženklai?
6. Kokias reikšmes gali įgauti santykis?
7. Kaip pasirenkama, kurią alternatyvą vykdyti?
8. Kaip atrodo sutrumpintas sąlygos sakinys?
9. Kada vartojamas sutrumpintas sąlygos sakinys?
10. Kokia yra ciklo sakinio sintaksė?
11. Kaip vykdomas kartojimas?
12. Ką daryti, kad ciklas nebūtų begalinis?
13. Kaip užrašyti ciklo sakinį, kai norim kartoti daugiau sakinių?
14. Kam vartojami valdančių struktūrų sakiniai?

2.8. Kontrolinės užduotys

1. Suraskite klaidas sąlygos sakiniuose:

```
a)  if a<b          then s := s + a;
                    else      s := s + b;
b)  if s => 0 then d := Sqrt(s)          else  d := Sqr(s);
c)  if k<> 0  then a := k+1        else  a := k+1;
d)  if r=0    then WriteLn ( Tai nėra apskritimas);
```

2. Ką spausdins kompiuterio ekrane tokia programa?

```
program alternatyva;
   var a, b, c : real;
      begin
          a := 5;          b := 6;          c := 10;
          if a+b > c       then c := b
                           else      c := a;
          if a+c <b        then b:= 0
```

```
                    else        b:= a;
        WriteLn (‘ a=‘, a:3:1, ‘ b=‘, b:4:1, ‘ c=‘, c:2)
    end.
```

3. Kurie iš žemiau pateiktų ciklo sakinių, kai a = -3, yra begaliniai ir kodėl?

a)
```
while  a<0   do begin
                WriteLn (‘ argumentas neigiamas’);
                 a := a +1
end;
```

b)
```
while  a<0  do;
    begin
        WriteLn(‘ argumentas neigiamas’);
        a := a+1;
    end;
```

c)
```
while a<0 do
     WriteLn(‘ argumentas neigiamas’);
        a := a+1;
```

d)
```
while a>=0 do
    WriteLn(‘ argumentas neigiamas’);
    a := a+1;
```

4. Parašykite ciklo sakinį, kuris, įvedus kintamojo x reikšmes mažesnes už 10, spausdintų jas su komentarais ir prašytų naujų dviženklių skaičių. Įvedus dviženklį skaičių, turi būti spausdinama padėka.

3 tema. Sumos kaupimo uždaviniai. Skaičių srautai

Skaičių srautus apdorojančiose programose labai dažnai taikomi sumos, sandaugos ir kiekio skaičiavimo algoritmai.

3.1 pratimas

Pavyzdžiui tarkime, kad reikia sudaryti programą klaviatūra įvedamo skaičių srauto sumavimui, kai yra žinoma, kad duomenų pabaigą žymi nulinė reikšmė. Tokio uždavinio sprendimui yra reikalingas vienas kintamasis (**x**) įvedamos reikšmės saugojimui ir antras (**s**) - sumos kaupimui. Suma yra skaičiuojama suteikiant jai pradinę reikšmę **0** (nulis) ir kartojant įvedimo bei sumos kaupimo operacijas tol, kol

Pradžia. Suma
x, s realūs skaičiai
Išvesti: darbo pabaiga, kai x=0
s := 0;
Įvesti x reikšmę
kol x <> 0
s := s+x;
Įvesti x reikšmę
Išvesti rezultatą: s reikšmę
Pabaiga

įvedama reikšmė nelygi nuliui. Sumos kaupimo veiksmas yra aprašomas sakiniu

s:= s+x;

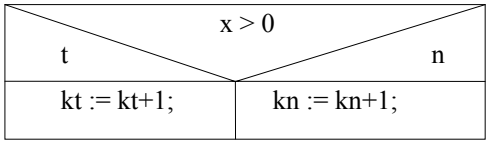
Priskyrimo sakinio kairėje ir dešinėje pusėje nurodytos kintamųjų reikšmės atitinka skirtingus laiko momentus, todėl sumos kaupimo sakinio prasmė yra

$$s_{nauja} := s_{sena} + x$$

Surinkite ir patikrinkite skaičių srauto sumavimo programą **Suma**.

```
program Suma;
var       x, s : real;                          { Įvedama reikšmė ir Suma    }
begin
WriteLn ('Sumavimo programa. Darbas nutraukiamas įvedus 0.');
     s:= 0;                                     {      Pradinė reikšmė       }
     WriteLn('Įveskite x: ');   ReadLn(x);
     while (x<>0) do  begin          { Sumos kaupimo ciklas                  }
               s:= s+ x;             { Sumos kaupimo veiksmas   }
     WriteLn('Įveskite x:');  ReadLn(x);
     end;
WriteLn('Sumos reikšmė: ', s:8:2); { Pranešimas apie rezultatus  }
ReadLn;                                 { Programos stabdymas       }
end.
```

- Pakeiskite šią programą taip, kad ji skaičiuotų ne įvedamų skaičių sumą, o sandaugą. Tam reikia sumos kaupimo sakinį pakeisti sandaugos kaupimo sakiniu **s := s * x**, sandaugai **s** suteikti pradinę reikšmę **1** ir pakeisti pranešimą apie rezultatus.
- Papildykite programą, kad ji suskaičiuotų, kiek buvo įvesta teigiamų ir kiek neigiamų skaičių. Tam reikia turėti papildomus du kintamuosius, pavyzdžiui **kt** teigiamų skaičių kiekiui saugoti ir **kn** neigiamų skaičių kiekiui saugoti. Kiekio skaičiavimas nuo sumos kaupimo skiriasi tik tuo, kad kiekio reikšmė didinama vienetu kiekvienu atveju, kada yra surandama tinkama reikšmė. Programą **Suma** ciklo viduje reikia papildyti sakiniu, kurio algoritmas toks:



3.2. Užduotis savarankiškam darbui

Klaviatūra įvedamos skaičių poros. Pirmasis skaičius reiškia prekės kainą, o antrasis tos prekės pirkėjo pasirinktą kiekį. Įvedamų skaičių srauto pabaiga nurodoma dviem skaičiais, lygiais nuliui. Reikia sudaryti programą, kuri suskaičiuotų, kiek pirkėjas privalo sumokėti pinigų už savo pirkinius. Išvesti į ekraną suskaičiuotą sumą ir kiek skirtingų prekių pasirinko pirkėjas (kiek skaičių porų buvo įvesta).

3.3. Ką turėjau išmokti?

Skaičių srauto sumavimui, sudauginimui, ar kiekio skaičiavimui reikalingi du kintamieji: įvedamos reikšmės saugojimui ir kaupimui. Jeigu skaičiuojama suma arba kiekis, tai kaupimo kintamajam suteikiama nulinė reikšmė, o jeigu sandauga - vienetinė reikšmė. Sumos kaupimo veiksmas aprašomas sakiniu

s := s + x;

o kiekio skaičiavimo k := k + 1;

Čia priskyrimo operatoriaus kairėje ir dešinėje pusėje nurodytos to paties kintamojo reikšmės skirtingais laiko momentais, todėl kaupimo veiksmo prasmė yra

nauja reikšmė := sena reikšmė + dydis.

3.4. Kontroliniai klausimai

1. Kaip galima nustatyti skaičių srauto pabaigą?
2. Kiek ir kokių kintamųjų dažniausiai reikia kaupimo uždaviniuose?
3. Kokią pradinę reikšmę reikia suteikti sumos kaupimo kintamajam?
4. Kuo skiriasi skaičių srauto sumavimo ir dauginimo algoritmai?
5. Kuo skiriasi sumos ir kiekio skaičiavimo algoritmai?
6. Kokiu operatoriumi aprašomas sumavimo veiksmas?
7. Kokia yra sumos kaupimo operatoriaus prasmė?
8. Kodėl, apdorojant skaičių srautus, kaupimo kintamiesiems reikia priskirti tam tikras pradines reikšmes?

3.5. Kontrolinės užduotys

1. Ką skaičiuoja ši programa?

```
program  Nul_ats;
    var   ats,   k : real;
     begin
        WriteLn(‘ Darbas nutraukiamas įvedus nulį’);
```

```
ats := 1;
WriteLn( 'Įveskite kintamojo reikšmę');
 ReadLn(k);
 ats := ats * k;
 while  k<>0  do      begin
       WriteLn( 'įveskite kintamojo reikšmę');
        ReadLn(k);
        ats := ats * k
                                  end;
  WriteLn(' Atsakymas bus:', ats : 3 : 0);
   ReadLn;
end.
```

2. Patikrinkite šią programą. Jos atsakymas neteisingas. Pataisykite šią programą.

3. Duota skaičių seka su nuliniais elementais, bet joje negali būti dviejų vienodų gretimų skaičių. Reikia suskaičiuoti, kiek teigiamų, neigiamų ir nulinių reikšmių įvedėme. Parašykite kartojimo sakinį, kuris užbaigtų duomenų įvedimą.

4 darbas. Didžiausios ir mažiausios reikšmės paieška skaičių sraute

4.1. Algoritmas

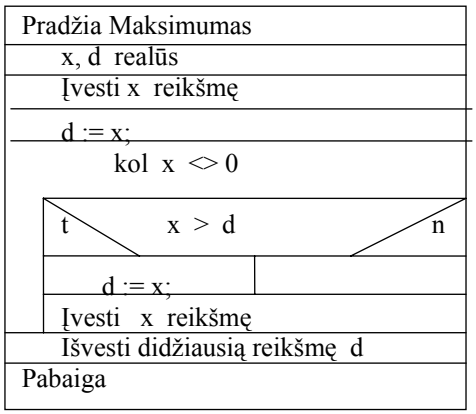
Tai dažnai sprendžiami uždaviniai. Populiariausias yra toks jų sprendimo būdas. Deklaruojami kintamieji įvedamai (**x**) ir didžiausiai (**d**) reikšmėms saugoti. Skaitoma pirmoji reikšmė ir padaroma prielaida, kad ji yra didžiausia (jei ieškoma didžiausios reikšmės):

d := x

Po to skaitomos kitos reikšmės ir lyginamos su **d**. Jei randama didesnė, kintamojo **d** saugoma reikšmė keičiama nauja:

if **x>d** *then* **d:= x**

Taip apdorojus visą įvedamą srautą, kintamasis **d** saugos didžiausią įvestą reikšmę.



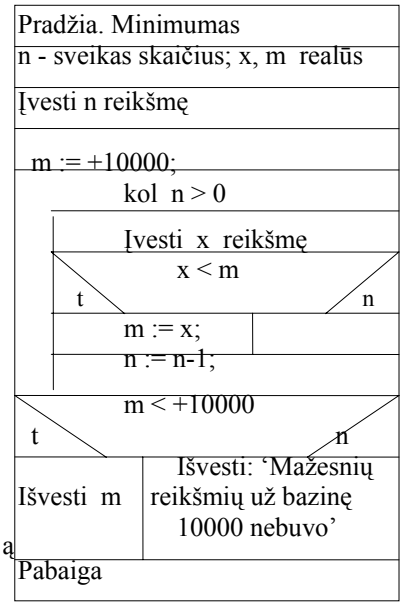
- Surinkite ir patikrinkite didžiausios reikšmės atrinkimo iš klaviatūra įvedamo srauto programą

Maksimumas.

```
program Maksimumas;
var       x, d : real;                  { Įvedama ir didžiausia reikšmės }
begin
      WriteLn('Maksimumo paieškos programa.',
                  'Darbas nutraukiamas įvedus 0. ');
      WriteLn('Įveskite x: ');   ReadLn(x);
      d:= x;                              { Pradinė reikšmė              }
      while (x<>0) do  begin        { Įvedimo ir analizės ciklas   }
            if  x>d  then  d:= x;       {  Maksimumo paieška         }
            WriteLn('Įveskite x:');          ReadLn(x);
      end;                          { Pranešimas apie rezultatus  }
      WriteLn('Didžiausia reikšmė: ', d:8:2);
      ReadLn;                       { Programos stabdymas         }
end.
```

- Pertvarkykite didžiausios reikšmės paieškos programą taip, kad ji ieškotų mažiausios reikšmės įvedimo sraute.

Ne visuomet patogu pirmąją reikšmę laikyti pradine didžiausia reikšme. Ypač tuomet, kai ta reikšmė turi būti skaičiuojama (pavyzdžiui, ieškant didžiausios funkcijos reikšmės nurodytame argumentų kitimo intervale). Tokiu atveju, galima pradine didžiausia reikšme laikyti pakankamai mažą reikšmę, tokią, kuri tikrai būtų mažesnė už visas galimas reikšmes. Ieškant mažiausios reikšmės, pradine reikšme galima laikyti pakankamai didelį skaičių.



Pavyzdys. Klaviatūra įvedami **n** realių skaičių. Reikia surasti mažiausią skaičių. Parašykite programą pagal duotą algoritmą **Minimumas**, suderinkite ją nurodytais 4.1 lentelėje.

4.1 lentelė

n reikšmė	x reikšmės	Rezultatas
4	8 3 –5 32	–5
3	15000 28000 39000	nebuvo
0		nebuvo

1	8	8
3	18 15230 −8	−8

4.2. Užduotis savarankiškam darbui

Sudarykite didžiausios reikšmės paieškos programos modifikaciją, kuri ne tik rastų didžiausią reikšmę, bet ir jos eilės numerį įvedimo sraute. Tam reikia papildomai įvesti įvedamų skaičių skaitiklį n, jam suteikti pradinę reikšmę 0 ir, įvedus kiekvieną skaičių, didinti šią reikšmę vienetu (n := n+1). Kiekių bei eilės numerių skaičiavimui paprastai yra naudojami sveiko tipo kintamieji, kurių tipas deklaravimo sakiniuose nurodomas baziniu žodžiu *integer*. Ieškant didžiausios reikšmės vietos, reikės įsiminti ne tik dalinius maksimumus, bet ir jų numerius. Taip pat turės būti pakeistas ir pranešimas apie rezultatus.

4.3. Ką turėjau išmokti?

Didžiausios ir mažiausios reikšmių (ekstremumų) ieškojimo uždaviniai skirstomi į dvi grupes pagal tai, ar reikia surasti tos maksimalios (minimalios) reikšmės eilės numerį arba prie kokios argumento reikšmės jis gautas, ar ne. Algoritmo struktūra taip pat priklauso nuo to, kaip bus pateikiami pradiniai duomenys: skaičių srautu ar skaičiuojami pagal formules. Jeigu reikia rasti didžiausią ar mažiausią skaičių srauto reikšmę, tai ***var*** srityje yra aprašomi tik du kintamieji: įvestam srauto skaičiui saugoti ir ekstremumui įsiminti. Kintamajam, kuris saugo didžiausią ar mažiausią reikšmę, skaičiavimų pradžioje reikia suteikti pradinę reikšmę. Dažniausiai tai būna pirmoji srauto reikšmė arba bet koks skaičius, už kurį sraute nėra didesnių arba mažesnių. Kartojant įvedimą, tikrinama, ar įvestoji reikšmė nedidesnė (nemažesnė) už įsimintąją. Todėl sąlygos sakinys, ieškant didžiausios reikšmės, bus toks:

```
if x > d then d := x;
```

Ieškant mažiausios reikšmės:

```
if x < m then m := x;
```

Jeigu reikia įsiminti didžiausios reikšmės eilės numerį, tuomet ***var*** srityje reikia aprašyti tris kintamuosius: du kaip ir ankstesniu atveju, o trečią - eilės numeriui įsiminti. Kadangi eilės numeris gali būti tik sveikas skaičius, šis kintamasis aprašomas baziniu žodžiu ***integer***.

4.4. Kontroliniai klausimai

1. Kaip dar vadinami didžiausios ir mažiausios reikšmės ieškojimo algoritmai?
2. Kaip skirstomi didžiausios (mažiausios) reikšmių ieškojimo uždaviniai?
3. Kiek ir kokių kintamųjų reikia aprašyti ***var*** srityje, norint surasti tik patį ekstremumą?

4. Kiek ir kokių kintamųjų reikia aprašyti ***var*** srityje, kai norim rasti kelintas elementas iš srauto yra mažiausias?
5. Koks yra didžiausios reikšmės ieškojimo sąlygos sakinys?
6. Koks yra mažiausios reikšmės ieškojimo sąlygos sakinys?
7. Kokias pradines reikšmes reikia suteikti tam kintamajam, kuris saugo ekstremumo reikšmę?
8. Kada kintamuosius ***var*** srityje aprašome baziniu žodžiu ***integer***?
9. Ar visuomet didžiausios reikšmės saugojamam kintamajam galima suteikti pirmą srauto reikšmę? Kodėl?

4.5. Kontrolinės užduotys

1. Mažiausiai reikšmei iš trijų skaičiuoti duotas toks sąlygos sakinys:

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

Ar jis teisingas? Jei ne, tai pateikite keletą tokių kintamųjų reikšmių, su kuriomis jis bus neteisingas.

2. Kaip ištaisyti aukščiau parašytą sakinį?

3. Reikia rasti skaičių sraute didžiausią neigiamą reikšmę. Duota tokia programa:

```
program  Klaida;
var      did, x : real;
begin
  WriteLn ('Įveskite x');      ReadLn(x);
   did := x;
   while (x<>0)  do      begin
      if  x < 0  then       if  x > did  then  did := x;
          WriteLn ('Įveskite x');       ReadLn(x);
    end;
    WriteLn('Didžiausia neigiama reikšmė:', did :5:2);
end.
```

Ar visuomet šios programos rezultatas bus teisingas? Jeigu ne, tai kada atsakymas bus neteisingas?

4. Pataisykite programą **Klaida**.

5 tema. Skaičių srautų analizės uždaviniai

5.1 Srautų analizė

Apdorojant duomenų srautus, gali būti sprendžiami įvairūs analizės uždaviniai. Pavyzdžiui, gali būti skaičiuojama, kiek skaičių telpa nurodytoje atkarpoje, kokia yra jų suma, koks yra šios sumos santykis su visų skaičių suma ir panašiai. Sprendžiant tokius uždavinius, kiekvienai skaitomai srauto reikšmei yra taikomi du veiksmai: filtravimas ir apdorojimas. Filtruojant yra nustatoma, ar reikšmė priklauso tiriamai duomenų grupei, o skaičiuojant yra nustatomos įvairios tiriamos grupės charakteristikos.

Paprastos filtravimo sąlygos yra aprašomos operatoriumi *if* ir santykiais. Sudėtingesnėms sąlygoms aprašyti tokių priemonių nepakanka, tenka iš santykių ir kitokių loginių dydžių formuoti logines išraiškas. Jose loginiams dydžiams gali būti taikomos šios operacijos: **not** (*neigimas*),

or (veiksmas *arba*),
and (veiksmas *ir*),
xor (griežtas *arba*).

Loginių išraiškų reikšmės gali būti suteikiamos loginiams kintamiesiems, kurie aprašomi baziniu žodžiu *boolean*. Logines operacijas apibūdina 5.1 lentelė. Sudarant logines išraiškas, reikia žinoti, kad loginio neigimo, kaip ir kitų unarinių operacijų, prioritetas yra aukščiausias, operacijos **and** prioritetas atitinka ***** ir **/** prioritetus, o operacijos **or** - veiksmų **+** ir **–** prioritetus. Natūralią veiksmų tvarką galima pakeisti skliaustais.
Pavyzdžiui, kintamojo x priklausymas atkarpai [5, 15] gali būti aprašomas taip: (x>=5) and (x<=15). Šios atkarpos išorę aprašo išraiška (x<5) or (x>15).

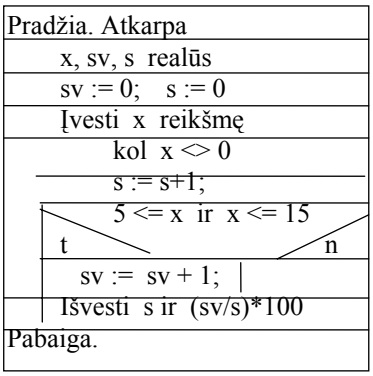
5.1 lentelė. Loginės operacijos

A	b	not a	a or b	a and b	a xor b
True	true	false	true	true	false
True	false	false	true	false	true
False	true	true	true	false	true
False	false	true	false	false	false

5.2 Pratimas

Išsiaiškinkite, kaip yra sudaryta ir kaip dirba programa, kuri skaičiuoja, koks klaviatūra įvedamų skaičių procentas patenka į atkarpą [5, 15].

Duomenų pabaigą nurodo nulinė reikšmė. Skaičių kiekių skaičiavimui yra skiriami *integer* tipo kintamieji (skaitikliai), kurie gali turėti tik sveikąsias reikšmes.




```
program Atkarpa;
var
    x,                          {   Įvedama reikšmė          }
    sv, s: integer;             {   Skaičių skaitikliai      }
begin
    WriteLn('Klaviatūra įvedamo skaičių srauto analizė');
    WriteLn('Srauto pabaiga - nulinė reikšmė');
    WriteLn('Įveskite skaičius (po vieną eilutėje);',
                ' pirmasis ne nulis:');
    sv:= 0;   s:= 0;            {   Pradinės skaitiklių reikšmės }
    ReadLn(x);                  {   Pirmoji srauto reikšmė       }
    while x<>0 do begin     {   Srauto pabaigos kontrolė         }
        s:= s+1;                {   Perskaitytų skaičių kiekis          }
        if (x>=5) and (x<=15) then  {   Srauto filtravimas  }
                sv:= sv+1;              {   Atrinktų skaičių kiekis }
        ReadLn(x);              {   Nauja srauto reikšmė             }
    end;
    WriteLn('Analizės rezultatai:');
    WriteLn('Įvesta skaičių: ', s);
    WriteLn('Atkarpos[5, 15] viduje yra ', (sv/s)*100:5:2,
                                    ' procentų įvestų skaičių.');
    ReadLn;
end.
```

- Pertvarkykite pratimo programą taip, kad ji skaičiuotų į atkarpos [5,15] vidų patenkančių skaičių sumą ir vidutinę reikšmę.
- Savarankiškai sudarykite programą, kuri įvedamo srauto duomenis pertvarkytų pagal formulę y:= Sqrt(100- x*x). Sudarykite filtrą, kuris skaičiavimams nukreiptų tik tas reikšmes, kurioms pertvarkymo formulė yra apibrėžta. Įvedus kitokias reikšmes turi būti formuojamas pranešimas, kad šioms reikšmėms skaičiavimų formulės negalima taikyti.

5.3. Užduotis savarankiškam darbui

Klaviatūra įvedamų sveikų skaičių sekos pabaiga yra nulis. Reikia parašyti programą, kuri suskaičiuotų, kiek buvo vienaženklių, dviženklių, triženklių, keturženklių ir kitokių skaičių. Vėliau papildykite programą veiksmais, kuriais būtų suskaičiuojamas įvestų skaičių aritmetinis vidurkis.

5.4. Ką turėjau išmokti?

Sprendžiant uždavinius, apdorojančius duomenų srautus, kiekvienai skaitomai srauto reikšmei yra taikomi du veiksmai: filtravimas ir apdorojimas. Filtruojant yra nustatoma, ar reikšmė priklauso tiriamai duomenų grupei. Skaičiuojant (apdorojant) yra nustatomos įvairios tiriamos grupės charakteristikos. Paprastos filtravimo sąlygos yra aprašomos operatoriumi *if* ir santykiais. Sudėtingesnėms sąlygoms aprašyti tenka iš santykių ir kitokių loginių dydžių formuoti logines išraiškas. Jose loginiams dydžiams gali būti taikomos šios operacijos:

not (neigimas),
or (veiksmas arba),
and (veiksmas ir),
xor (griežtas arba).

Loginiai kintamieji aprašomi baziniu žodžiu ***boolean***. Pirmiausia atliekamas neigimas, po to operacija ***and*** (ir) ir pabaigoje ***or*** (arba). Natūralią veiksmų tvarką galima pakeisti skliaustais.

5.5. Kontroliniai klausimai

1. Kokie veiksmai taikomi apdorojant duomenų srautus?
2. Kas atsitinka filtruojant ir skaičiuojant?
3. Kaip aprašoma filtravimo sąlyga?
4. Kokias žinote logines operacijas?
5. Kokiu baziniu žodžiu aprašomi loginiai kintamieji?
6. Kokie yra loginių operacijų prioritetai?

5.6. Kontrolinės užduotys

1. Kuris iš anksčiau nagrinėtos programos ***if*** sakinių atlieka filtravimą, o kuris apdorojimą?

```
program  Klaida;
    var         did, x : real;
    begin
        did := -1E + 06;
        WriteLn (‘Įveskite x’);          ReadLn(x);
```

```
        while (x<>0) do        begin
            if  (x < 0)   then   if  (x > did)  then   did := x;
            WriteLn ('Įveskite x');          ReadLn(x);
        end;
        Writeln('Didžiausia neigiama reikšmė:', did:5:2);
end.
```

2. Kaip, naudojant logines išraiškas, sutrumpinti sąlygos sakinį?

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

3. Duoti trijų trikampio kraštinių ilgiai a, b, c. Kuriam tikslui reikalingas toks sakinys:

lk := (a =b) *and* (b=c) *and* (a=c) ?

Ar jis teisingas? Ar šio priskyrimo sakinio negalima supaprastinti?

6 tema. Tekstiniai failai

6.1. Failo tipas ir failo kintamasis

Sprendžiant uždavinius, dažnai yra pageidaujama atskirti pradinių duomenų parengimo, jų apdorojimo ir rezultatų analizės procesus. Tai galima padaryti surašant pradinius duomenis į failus ir parengiant šių failų apdorojimo programas ir rezultatus vėl rašant į failus. Programos aprašų dalyje kiekvienam failui yra skiriamas failo kintamasis, kurio aprašo sintaksė:

<Vardas>: <Failo tipas>

Universaliausi yra tekstiniai failai, kurių tipas žymimas baziniu žodžiu *text*. Šie failai yra sudaromi iš į eilutes sugrupuotų simbolių. Eilučių pabaigos žymimos specialiais eilutės pabaigos simboliais, o failo pabaigą žymi failo pabaigos simbolis. Siunčiant parengtą failą į diską, šis simbolis įrašomas automatiškai. Ekrane eilučių ir failo pabaigos simboliai nerodomi.

Programos pagrindinėje dalyje kiekvienas failo kintamasis turi būti susiejamas su konkrečiu operacinės sistemos (OS) failu. Tai nurodoma procedūra:

Assign(<Failo kintamasis>, '<Failo vardas>')

Failo vardas yra sudaromas pagal aptarnaujančioje OS numatytas taisykles ir yra suteikiamas failui kuriant jį redaktoriumi. Programoje OS

failo vardas tiesiogiai nurodomas tik procedūroje *Assign*, o po to jis yra atstovaujamas failo kintamojo. Po to failas turi būti parengiamas konkrečiai apdorojimo operacijai:

skaitymui Reset (<Failo kintamasis>);
rašymui Rewrite (<Failo kintamasis>);

Duomenys iš failų yra skaitomi panašiai kaip iš klaviatūros:

Read(<Failo kintamasis>, <Kintamųjų sąrašas>);
ReadLn(<Failo kintamasis>, <Kintamųjų sąrašas>);

Iš tekstinių failų skaitomi skaičiai turi būti atskirti tarpais, tabuliacijos arba eilutės pabaigos simboliais. Procedūra *Read* iš tekstinio failo visuomet skaito tiek reikšmių, kiek yra elementų jo kintamųjų sąraše. Procedūra *ReadLn* skiriasi tuo, kad ji iš pradžių perskaito visą eilutę ir tik po to skirsto jos duomenis kintamiesiems. Jei eilutėje duomenų būna perdaug, pertekliniai duomenys atmetami. Jei jų trūksta, skaitoma kita failo eilutė ir reikšmių skirstymas kintamiesiems tęsiamas.

Išvedimui į failus naudojamos procedūros:

Write (<Failo kintamasis>, <Duomenų sąrašas>);
WriteLn(<Failo kintamasis>, <Duomenų sąrašas>);

Jos savo savybėmis yra analogiški atitinkamoms išvedimo į ekraną procedūroms. Procedūra WriteLn nuo Write skiriasi tuo, kad ji išvedamų duomenų sąrašą papildo specialiais eilutės pabaigos simboliais.

Kai failų apdorojimas baigiamas, juos būtina uždaryti:

Close(<Failo kintamasis>)

Skaitant failus, loginėmis funkcijomis

EoLn(<Failo kintamasis>)
ir Eof(<Failo kintamasis>)

galima tikrinti atitinkamai eilutės ir failo pabaigos žymes. Šioms funkcijoms yra suteikiama reikšmė *true* nuskaičius paskutinį duomenų elementą eilutėje arba faile atitinkamai.

6.2. Pratimas

Patikrinkite programą, kuri sumuoja atskirose tekstinio failo *duomenys.dat* eilutėse įrašytus skaičius. Rezultatai parodomi ekrane ir surašomi į failą. Prieš rašant programą, duomenų failas su laisvai pasirinktais skaičiais turi būti sukuriamas Turbo Paskalio aplinkos redaktoriumi ir įrašomas į darbinį katalogą.

```
program Sumos;
var     duomenys, rezultatai: text;   { Duomenų ir rezultatų failai }
        x,                            { Skaitoma reikšmė              }
        s, n: integer;                { Skaičių suma ir eilutės numeris }
```

```
begin
  WriteLn ('*** Failo duomenų skaitymas ir apdorojimas ***');
  Assign(duomenys, 'duomenys.dat');
  Reset(duomenys);                    { Failo parengimas skaitymui}
  Assign (rezultatai, 'result.dat');
  Rewrite(rezultatai);                { Failo parengimas rašymui      }
  n:= 0                               { Pradinė reikšmė           }
  while not Eof(duomenys) do begin    { Failo skaitymas               }
      s:= 0;      n:= n+1             { Suma eilutei ir numeris       }
      while not EoLn(duomenys) do     { Eilučių apdorojimas           }
      begin
            Read(duomenys, x);        { Skaičiaus skaitymas           }
            s:= s+x;                  { Sumos kaupimas            }
      end;
      ReadLn(duomenys);               { Perėjimas į naują eilutę      }
                                      { Rezultatų išvedimas į ekraną  }
      WriteLn('Eilutės nr. ', n, '  Suma: ', s:8:2);
                                      { Rezultatų rašymas į failą     }
      WriteLn(rezultatai, 'Eilutės nr. ', n, '  Suma: ', s:8:2);
  end;
  Close(duomenys);                { Failų uždarymas           }
  Close(resultatai);
  ReadLn;
end.
```

Atsiminkite, kad duomenų failas turi arba būti tame pačiame kataloge, kaip ir jį apdorojanti programa, arba jo vietą turi nurodyti aplinkos parametrai, arba kelią į jį turi aprašyti programoje sakinys *Assign*.
Rezultatų failo duomenis galima pamatyti ir po to, kai programa baigia darbą. Naudodami integruotos aplinkos priemones, patikrinkite tai savarankiškai.

- Pakeiskite programą taip, kad ji ekrane parodytų skaitomus iš pradinių duomenų failo skaičius.
- Pakeiskite programą, kad ji papildomai skaičiuotų kiekvienoje eilutėje esančių skaičių kiekį, ir visame faile esančių skaičių sumą bei kiekį.
- Savarankiškai sudarykite sveikųjų skaičių failo analizės programą, kuri nustatytų, kokį procentą faile sudaro lyginiai skaičiai. Nustatant, ar skaičius yra lyginis, patogu vartoti liekanos skaičiavimo operaciją, kuri dar yra vadinama dalyba moduliu. Jos aprašymo sintaksė: <dalinamasis> *mod* <daliklis>. Operacijos rezultatas yra lygus liekanai. Pavyzdžiui, sąlyga, kuri tikrina, ar skaičius x yra lyginis, gali būti aprašoma taip: (x mod 2) =0.

Modifikuokite sveikųjų skaičių failo analizės programą taip, kad ji skaičiuotų, kokį procentą sudaro skaičiai, kurie baigiasi skaitmeniu 5. Tokių skaičių filtravimo sąlyga yra (x mod 10) = 5.

6.3. Užduotis savarankiškam darbui

Sudaryti sveikųjų skaičių failo analizės programą, kuri nustatytų, kiek procentų skaičių yra mažesnių už didžiausios reikšmės pusę. Pradinių duomenų failą teks skaityti du kartus: ieškant didžiausios reikšmės ir filtruojant reikšmes pagal duotą sąlygą.

6.4. Ką turėjau išmokti?

Pradinių duomenų ir rezultatų saugojimui galima naudoti failus. Aprašų dalyje kiekvienam failui yra skiriamas failo kintamasis, kuris aprašomas *var* srityje taip:

<Vardas>:<Failo tipas>

Tekstinių failų tipas dažniausiai žymimas baziniu žodiu ***text.***

Norint susieti failo kintamąjį su operacinės sistemos konkrečiu failu, naudojama procedūra:

Assign(<Failo kintamasis>, '<Failo vardas>');

Failo vardas yra sudaromas pagal OS galiojančias taisykles. Jeigu failas yra ne tame pačiame kataloge, kur yra jį apdorojanti programa, tai prie vardo reikia nurodyti ir kelią.

Po susiejimo failas turi būti parengiamas konkrečiai apdorojimo operacijai:

a) skaitymui: Reset(<Failo kintamasis>);
b) rašymui: Rewrite(<Failo kintamasis>);

Duomenys iš failo yra skaitomi panašiai kaip ir iš klaviatūros:

Read(<Failo kintamasis>,<Kintamųjų sąrašas>);
ReadLn(<Failo kintamasis>,<Kintamųjų sąrašas>);

Tekstinis failas, iš kurio skaitomi duomenys turi būti paruoštas iš anksto. Skaičiai faile vienas nuo kito turi būti atskirti tarpais. Kiekvienos eilutės pabaigą ir failo pabaigą žymi specialūs simboliai, kurie ekrane nematomi, bet užrašomi diske. Skaitant su procedūra Read, yra skaitoma tiek reikšmių, kiek kintamųjų yra sąraše, o procedūra ReadLn skaito visas reikšmes iki eilutės pabaigos simbolio ir tik po to priskiria jas sąrašo kintamiesiems. Jeigu reikšmių yra per daug, jos atmetamos. Jeigu trūksta, tai padaroma su sekančia eilute.

Skaitant failą, loginėmis funkcijomis EoLn(<Failo kintamasis>) ir Eof(<Failo kintamasis>) galima tikrinti atitinkamai eilutės ir failo pabaigos

žymes. Šioms funkcijoms suteikiama reikšmė *true* skaitant paskutinį simbolį eilutėje ar faile atitinkamai.

Išvedimui į failus naudojamos procedūros:

Write(<Failo kintamasis>, <Duomenų sąrašas>);
WriteLn(<Failo kintamasis>, <Duomenų sąrašas>);

Šios procedūros skiriasi tik tuo, kad WriteLn papildo failą eilutės pabaigos simboliais.

Kai failų apdorojimas baigiamas, juos būtina uždaryti:

Close(<Failo kintamasis>)

Negalima vienu metu ir skaitymui ir rašymui naudoti to paties failo.

6.5. Kontroliniai klausimai

1. Kokia procedūra susieja failo kintamąjį su OS failu?
2. Kam reikalingos procedūros Reset ir Rewrite?
3. Kaip reikia aprašyti failo kintamąjį *var* srityje?
4. Ką reikia daryti, norint nuskaityti duomenis iš failo?
5. Kokia procedūra naudojama failų uždarymui?
6. Kuo skiriasi procedūros Read ir ReadLn, skaitant duomenis iš failo?
7. Kuo skiriasi procedūros Write ir WriteLn rašant rezultatus į failą?
8. Kuriam tikslui reikalingos funkcijos EoLn ir Eof?
9. Kaip sudaromi duomenų failai ?

6.6 Kontrolinės užduotys

1. Sudaryta programa įvedamų iš klaviatūros sveikų skaičių sekos įrašymui į diską ir jų sumavimui. Skaičiai įvedinėjami tol, kol jų suma neviršija 100. Galutinė suma taip pat rašoma į diską ir spausdinama ekrane. Ar teisinga programa? Pagrįskite savo nuomonę.

```
program  Sumavimas;
    var          x, sum : integer;
                 F : text;
     begin
        Assign(f, 'a:\duom.dat');              Rewrite(F);
        sum := 0;
        while   sum <= 100   do      begin
                WriteLn(' Įveskite sveiką skaičių');      ReadLn(x);
                sum := sum + x;
                WriteLn(F, x);
         end;
        WriteLn(F, sum);
        ReadLn(F, sum);
```

```
    WriteLn('Galutinė suma:', sum);
    Close(F);
end.
```

2. Duotas duomenų failas d.dat, kuriame skaičiai išdėstyti tokia tvarka:

```
–3   2   5
–4  –2   1
```

Programoje duotas toks sakinys:

```
while not Eof(F) do      ReadLn(F, a, b, c, d);
```

Čia F failo kintamasis. Kas bus atspausdinta ekrane, įvykdžius komandą:

```
WriteLn(' a=', a:2, ' b=', b:2, ' c=', c:2, ' d=', d:2);
```

Kodėl?

3. Kaip pasikeis antros užduoties atsakymas, jeigu procedūrą ReadLn pakeisime Read?

4. Sudarykite programą, kuri iš duomenų failo d.dat (žr. užduotį Nr. 2) sudarytų failą d1.dat, kuriame būtų tik neigiami skaičiai.

7 tema. Atsitiktinių situacijų modeliavimas

Dauguma programų modeliuoja įvairius mūsų aplinkos reiškinius, kur dažnai tenka turėti reikalų su atsitiktiniais dydžiais. Pavyzdžiui, žaidžiant kortomis, atsitiktinius skaičių šaltinis yra kortų kaladė. Tik siekiant, kad maloniau būtų žaisti, kortos papuošiamos paveiksliukais. Žaidėjai, imdami kortas ir darydami ėjimus, formuoja situacijas, kurios lemią laimėtoją.

7.1. Žaidimų modeliavimas

Panagrinėkime, kaip galima sumodeliuoti paprasčiausią žaidimą kortomis 'Akis'. Šiame žaidime žaidėjai ima pakaitomis kortas iš kaladės tol, kol surenka sumą pakankamai artimą skaičiui 21. Laimi tas žaidėjas, kurio suma būna artimesnė skaičiui 21. Vienas iš žaidėjų yra kompiuteris.

Žaidimo modeliavimui reikalingi kintamieji, kurie vaizduotų žaidėjų imamas kortas, jų surinktas taškų sumas, aktyvų žaidėją ir sprendimus apie žaidimo nutraukimą. Kortų kaladę galima imituoti atsitiktinių skaičių generavimo funkcija Random(n), kuri generuoja atkarpos [0, n-1] atsitiktinius skaičius. Prieš panaudojant šią funkciją, rekomenduojama iškviesti procedūrą Randomize, kuri susieja funkcijos Random generuojamos atsitiktinių skaičių sekos pradžią su kompiuterio laikrodžio parodymu.

Prieš pradedant rašyti programą, taip pat turi būti sudaromas jos scenarijus (algoritmas), kuris turi tiksliai numatyti visų programos valdomų

veiksmų seką. Tokio scenarijaus metmenis galima aprašyti paprastais lietuvių kalbos sakiniais.
Pavyzdžiui, kortų žaidimo Akis modeliavimui gali būti parenkamas toks scenarijus:

- Žaidėjų sumoms suteikiamos pradinės nulinės reikšmės;
- Formuojama užklausa apie pradedantį žaidėją;
- Kol bent vienas žaidėjas sutinka žaisti, kartojami veiksmai:
  - ◊ Modeliuojamas kortos ėmimas;
  - ◊ Kortos taškai pridedami prie aktyvaus žaidėjo sumos;
  - ◊ Keičiama aktyvaus žaidėjo žymė;
  - ◊ Informuojama apie žaidybinę situaciją;
  - ◊ Priimamas sprendimas apie aktyvaus žaidėjo norą tęsti žaidimą;
- Skelbiami žaidimo rezultatai.

Turint scenarijų, galima parinkti jo realizavimui skirtus kintamuosius ir numatyti atskirų veiksmų programinio realizavimo būdus. Pavyzdžiui, iš kaladės imamos kortos ir žaidėjų surinktų sumų modeliavimui tinka *integer* tipo kintamieji, sprendimams apie norą tęsti žaidimą - *boolean* tipas, aktyvaus žaidėjo nurodymui - *char* tipas.

Iš scenarijaus aprašymo struktūros matyti, kad jį galima aprašyti operatoriumi *while*, kuris turi tikrinti žaidėjų apsisprendimą aprašančias logines (boolean) žymes. Apie kortos ėmimo iš kaladės modeliavimą jau buvo minėta - šiam tikslui tinka atsitiktinių skaičių generavimo funkcija Random. Užklausos ir informaciniai pranešimai realizuojami įvedimo/išvedimo operatoriais, o žaidėjų apsisprendimas apie žaidimo nutraukimą - sąlyginiais operatoriais *if*.

- Išnagrinėkite žaidimą kortomis modeliuojančią programą Akis ir patikrinkite jos darbą.

```
                { Programa modeliuoja žaidimą kortomis Akis }
program Akis;
var
    x,                              { Žaidėjo imama korta   }
    n1, n2: integer;                { Sukauptos sumos       }
    e,                              { Pagalbinis kintamasis }
    c:  char;                       { Aktyvus žaidėjas      }
    dar1, dar2: boolean;            { Apsisprendimo žymės   }
begin
 WriteLn (' ****** Ž A I D I M A S   A K I S ****** ');
                        { Žaidžiant siekiama surinkti sumą,          }
                        { kuri būtų  galimai artimesnė skaičiui 21 }
                        { Kintamieji dar1 ir n1 modeliuoja           }
                        { kompiuterį, o dar2 ir n2 - žaidėją.        }
```

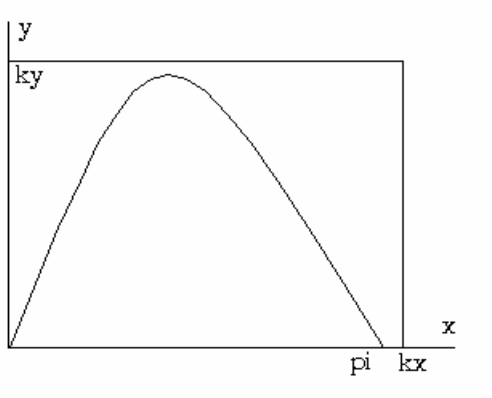
```
 dar1:= true;  dar2:= true;    { Pradinis apsisprendimas          }
 n1:= 0;       n2:=0;          { Pradinės sumos                   }
 WriteLn('Kas pradeda žaidimą (k- kompiuteris, ž- žaidėjas)');
 ReadLn(c);
 Randomize;                    { Generatoriaus nustatymas }
 while  dar1  or  dar2  do begin      { Žaidimo ciklai             }
    x:= Random(11)+1;                 { Imama korta                }
    if  dar1  and  (c='k')  then  begin  { Kompiuterio korta       }
      n1:= n1+x;                      { Kompiuterio suma           }
      c:='ž';                         { Aktyvaus žaidėjo keitimas }
      if (21-n1)<=2  then dar1:= false;{ Sprendimas, ar tęsti      }
    end
    else  begin                       { Žaidėjo korta              }
      WriteLn('Ar dar bus ėjimas (t/n)');   { Žaidėjas atsako    }
      ReadLn(e);
      if  e='t'  then begin           { Sprendimo analizė          }
                  n2:= n2+x;          c:='k';
                  WriteLn ('Jums teko: ', x, ' Jūsų suma: ', n2)
                     end
      else  begin  dar2:= false; c:= 'k'  end;  { Atsisakymas tęsti }
         end;
 end;
 WriteLn ('Rezultatai:');
 WriteLn('Kompiuterio: ', n1, ', Jūsų: ', n2);
    ReadLn;
end.
```

- Papildykite programą, kad jos pradžioje būtų pateikiamos žaidimo taisyklės;
- Pakeisti kompiuterio sumos kaupimo sąlygą taip, kad joje būtų atsitiktinis skaičius ir žaidėjui būtų sunkiau orientuotis, kada nutraukti žaidimą;
- Papildykite programą rezultatų analizės dalimi, kurioje būtų nustatoma, kas laimėjo ir būtų pranešama apie laimėtoją.
- Savarankiškai sudarykite programą žaidimo kauliukais modeliavimui. Šiame žaidime kiekvienas žaidėjas meta po tris kubo formos kauliukus, ant kurių šonų yra atkarpos [1..6] skaičiai. Todėl žaidėjui tenka trijų atsitiktinių šios atkarpos skaičių suma. Laimi tas žaidėjas, kurio suma didesnė.

7.2. Atsitiktinių skaičių panaudojimas skaičiavimuose

Atsitiktinių skaičių sekos taip pat sėkmingai yra vartojamos sprendžiant įvairius skaičiavimo uždavinius: ieškant daugelio kintamųjų funkcijų maksimalių ir minimalių reikšmių, skaičiuojant sudėtingų figūrų plotus,

tūrius ir daugeliu kitų atvejų. Pavyzdžiui, panagrinėkime, kaip gali būti apskaičiuotas x ašies ir linijos y=sin(x) pirmojo pusperiodžio ribojamos figūros plotas (7.1 pav.).



7.1 pav. Ploto skaičiavimo modelis

Figūra, kurios plotas yra skaičiuojamas, apibrėžiama stačiakampiu, kurio dvi kraštines sudaro koordinačių ašys, o kitos (kx ir ky) yra parenkamos taip, kad kraštinių ilgiai būtų sveiki skaičiai ir visa nagrinėjama figūra tilptų stačiakampio viduje. Po to turi būti generuojamos atsitiktinės taškų stačiakampio viduje koordinatės (x,y) ir skaičiuojama, kiek tokių taškų patenka į figūros vidų (y<=yl). Jeigu generuojamos koordinačių reikšmės bus tolygiai pasiskirsčiusios, šio skaičiaus ir visų generuojamų taškų skaičiaus santykis nurodys, kokią stačiakampio dalį sudaro nagrinėjamos figūros plotas.

Tokio skaičiavimo būdo tikslumas priklauso nuo generuojamų taškų skaičiaus. Skaičiavimų tikslumą galima įvertinti kartojant skaičiavimus ir sulyginant jų rezultatus. Realūs tolydinio pasiskirstymo atkaroje [0..1] skaičiai yra generuojami parametrų neturinčia funkcija Random.

- Išbandykite programą **Plotas**, kuri 10 kartų skaičiuoja funkcijos y=sinx pirmojo pusperiodžio ir y ašies ribojamą plotą ir įvertinkite skaičiavimo paklaidą (ji lygi didžiausios ir mažiausios reikšmių skirtumo pusei).

```
program  Plotas;
const n= 100;                  {    Tiriamų taškų skaičius   }
var
     x, y: real;               {    Taško koordinatės        }
     i,  j,                    {    Pagalbiniai kintamieji          }
     m:  integer;              {    Taškai figūros viduje           }
     kx, ky, s:  real;         { Stačiakampio kraštinės, plotas }
     ss:  real;                { Skaičiuojamas figūros plotas     }
begin
```

```
  kx:= 4; ky:= 1;                { Stačiakampio kraštinių dydžiai }
  s:= kx*ky;                     {     Stačiakampio plotas        }
  Randomize;
  for  j:=1  to  10  do begin    { Ploto skaičiavimo kartojimas   }
     m:= 0;
     for  i:=  1  to  n  do begin   { Taškų skaičiavimas         }
          x:= Random* kx;           { Atsitiktinės koordinatės   }
          y:= Random* ky;
          if y <= Sin(x) then m:= m+1;{ Taškai figūros viduje   }
     end;
  ss:= s*m/n;
  Write (ss: 6: 2 );             { Figūros plotas                 }
  end;
  ReadLn;
end.
```

- Papildykite programą informaciniais pranešimais taip, kad jos darbo rezultatai ekrane būtų pateikti vaizdžiau.
- Pakeiskite programos generuojamų taškų skaičių ir patikrinkite, kaip nuo to priklauso skaičiavimų paklaida.

7.3. Užduotis savarankiškam darbui

Maiše sumesti trijų skirtingų spalvų rutuliukai. Du žaidėjai paeiliui iš maišo traukia po vieną rutuliuką. Laimi tas, kuris pirmas surenka kurios nors vienos spalvos penkis rutuliukus. Reikia parašyti programą, kuri imituotų maiše esančius rutuliukus ir atsitiktinai išrinktų po vieną kiekvienam žaidėjui tol, kol kuris nors iš jų laimės. Programos algoritmas gali būti toks:

Pradžia. Rutuliukai
Kintamųjų aprašymas.
sa1 :=0; sa2 := 0; sa3 := 0; Pirmojo žaidėjo rutuliukų skaičiai sb1 :=0; sb2 := 0; sb3 := 0; Antrojo žaidėjo rutuliukų skaičiai
kol nei vienas iš rutuliukų skaitliukų dar nelygūs 5
Spalva := Random(3); Pirmojo žaidėjo rutuliukas Pridedamas vienetukas prie vieno iš skaitliukų
Spalva := Random(3); Antrojo žaidėjo rutuliukas Pridedamas vienetukas prie vieno iš skaitliukų
Skelbiamas nugalėtojas arba lygiosios
Pabaiga

7.4. Ką turėjau išmokti?

Modeliuojant įvairius aplinkos reiškinius, dažnai tenka susidurti su atsitiktiniais dydžiais. Dažniausiai tai būna, kai su kompiuteriu norime sumodeliuoti kokį nors žaidimą, pavyzdžiui žaidimus kortomis, kauliuku, ar domino. Atsitiktinius skaičius generuoja funkcija Random, kuri gali ir neturėti parametrų. Skaičiai, gauti šios funkcijos pagalba, tolydžiai pasiskirsto intervale [0..1]. Jeigu funkciją Random naudosime su parametru n, kuris gali būti tik *integer* tipo, tai skaičiai tolygiai pasiskirstys intervale [0..n-1]. Prieš naudojant funkciją Random, rekomenduojama atsitiktinių skaičių generatoriui suteikti pradinę reikšmę. Tai atlieka standartinė procedūra Randomize.

Programuojant žaidimus, kaip ir sprendžiant kitus uždavinius, turi būti sudarytas jo scenarijus (algoritmas), kuris tiksliai numatytų visą programos valdomų veiksmų seką. Turint scenarijų, galima numatyti, kokio tipo ir kokius vartosime kintamuosius, iš kur imsime duomenis, kokiais programavimo sakiniais užrašysime vieną ar kitą scenarijaus dalį.

Atsitiktinių skaičių sekas galima naudoti taip pat ir programų patikrinimui, nes kontrolinių duomenų įvedimas iš klaviatūros atima daug laiko.

7.5. Kontroliniai klausimai

1. Kokiems tikslams dažniausiai naudojami atsitiktiniai dydžiai?
2. Kokios funkcijos pagalba galima gauti atsitiktinius skaičius?
3. Kas yra svarbu programuojant žaidimus?
4. Kuo pasižymi funkcija *Random*?
5. Kodėl patogu sudaryti didesnes skaičių sekas iš atsitiktinių dydžių?
6. Ką daryti, kad naudojant atsitiktinius dydžius nenukentėtų skaičiavimų tikslumas?

7.6 Kontrolinės užduotys

1. Reikalingas didelis skaičių masyvas, kurio skaičiai būtų tolydžiai pasiskirstę intervale [-5..5]. Kaip parašyti funkcija *Random*?

2. Pradinių duomenų srautai sudaromi dviem būdais: įvedant atkarpos [1, 10] skaičius iš klaviatūros ir generuojant skaičių seką procedūra *Random(10)*. Pabaigos sąlyga - įvestas ar sugeneruotas skaičius 0. Reikia abiem atvejais rasti didžiausią ir mažiausią reikšmę sraute. Prie kokių sąlygų rezultatai bus vienodi? Kodėl? Pateikite pavyzdžių.

8 tema. Ekrano valdymo pradmenys.

8.1. Ekrano dažymas ir langai

Kiekviena programa turi ne tik realizuoti sprendžiamo uždavinio algoritmą, bet ir būti vaizdi, lengvai valdoma. Programos valdymo ir jos pateikiamų duomenų vaizdavimo priemonės bei būdai yra vadinami programos išoriniu interfeisu. Išorinio interfeiso sudarymo priemonės priklauso nuo kompiuterio įrangos savybių ir jos nuolat yra tobulinamos, kinta. Todėl šių priemonių aprašymui skirtos konstrukcijos dažniausiai priklauso ne pačioms programavimo kalboms, bet jų išplėtimams, papildymams, kurie pateikiami pagalbinių priemonių rinkiniuose - bibliotekose. Turbo Paskalio aplinkoje yra labai efektyvi ir paprasta IBM PC tipo kompiuterių ekranų vaizdo valdymo biblioteka **Crt**. Norint ja naudotis, reikia programos pradžioje įrašyti sakinį **uses Crt;**

Dažniausiai vartojamos bibliotekos procedūros ir funkcijos:

function ReadKey : char;
Klavišo simbolio apklausa.

procedure GoToXY(<x,y>: integer;)
Kursorius perkeliamas į koordinatėmis x,y nurodomą ekrano vietą.

procedure ClrScr;
Ekrano valymas.

procedure ClrEol;
Valoma aktyvios eilutės pabaiga.

procedure TextColor(<spalva>: byte);
Teksto spalva: spalvos kodas arba konstanta.

procedure TextBackground(<spalva >)
Fono spalva: spalvos kodas arba konstanta.

Prcedure Window(<x1,y1,x2,y2>: integer);
Aktyvaus lango aprašymas: (x1,y1) ir (x2,y2) atitinkamai kairiojo viršutinio ir dešinio apatinio lango kampų koordinatės.

procedure KeyPressed;
Loginė procedūra, kuri praneša, kad paspaustas bet koks klavišas.

var TextAttr: word;
Kintamasis, kuris saugo duomenis apie spalvinį ekrano režimą

Norint gauti išsamesnių žinių apie kurią nors iš išvardintų procedūrų ir funkcijų arba kitą Turbo Paskalio kalbos struktūrą, reikia redaktoriaus lange surinkti struktūros pavadinimą, prie jo perkelti kursorių ir paspausti klavišų pora **Ctrl+F1**. Po to ekrane bus matoma pagalbinė informacija apie kursoriaus nurodomą struktūrą.

Nurodant kursoriaus ir langų koordinates, yra laikoma, kad koordinačių sistemos pradžia yra kairiajame viršutiniame ekrano kampe. **X** ašis nukreipta į dešinę, o **Y** - žemyn. Ekranas skaidomas į simbolių rašymui skirtus langelius, kurių dydis priklauso nuo ekrano darbo režimo. Standartinis ekrano dydis **80x25** langelių. Koordinačių sistemos pradžios langelio koordinatės yra(1,1).

Ekrano darbo spalvos yra nurodomi skaitmeniniais kodais arba modulio **Crt** konstantomis:

1 lentelė. Spalvas aprašančios konstantos

Vardas	Reikšmė	Vardas	Reikšmė
black	0	darkgray	8
blue	1	lightblue	9
green	2	lightgreen	10
cyan	3	lightcyan	11
red	4	lightred	12
magenta	5	lightmagenta	13
brown	6	yellow	14
lightgray	7	white	15

Spalvų pulsavimą aprašo konstanta blink = 128.

Simbolių, fono spalvas ir ekrano režimą galima keisti tiktai tada, kai pageidaujamus režimus palaiko kompiuteryje esančio ekrano kontroleris (valdymo įtaisas). Yra ir kitų ribojimų. Pavyzdžiui, foną galima nurodyti tik pirmais 8 spalvų kodais: 0..7.

Jeigu prie teksto spalvos kodo pridėsime konstantą blink, pavyzdžiui TextColor (Red+Blink), ekrane bus rašomas pulsuojančio ryškumo tekstas.

Dažant ekraną, turi būti aprašoma fono spalva (TextBackground(<fonas>)) ir vykdoma ekrano valymo komanda ClrScr.

Ekrano spalvas keičiančiose programose reikia įsiminti kintamojo TextAttr saugomas spalvas ir programos darbo pabaigoje jas atstatyti.

- Patikrinkite programą, kuri dažo ekraną dialogo metu nurodomomis spalvomis

```
program Colors;
  uses Crt;                    {   Bibliotekos Crt aktyvizavimas  }
  var sb, se, st : integer;    {   Pagalbiniai kintamieji         }
begin
     se:= TextAttr;            {   Spalvinio režimo įsiminimas    }
     ClrScr;                   {   Ekrano valymas                 }
     sb:= 1;                   {   Pradinė reikšmė                }
     Writeln('Ekrano dažymas');
   while  sb<>0 do begin       {   Ekrano dažymo ciklai           }
     WriteLn ('Nurodykite spavos kodą (1..7). Pabaiga - 0:');
     ReadLn(sb);
     if  sb<>0 then begin    { Pranešimo apie pabaigą kontrolė  }
          WriteLn ('Nurodykite teksto spalvos kodą (0..15);
          ReadLn (st);
          if st = sb       { Ar teksto ir fono spalvos sutampa?   }
               then  Writeln(' Teksto ir fono spalva sutapo. Spalvų
nekeisiu. Lieka ankstesnės')
               else  begin         {   Spalvų keitimas            }
                    TextColor(st);          TextBackground(sb);
                    WriteLn ('Rašau jūsų pasirinktomis spalvomis');
                    end
     end;
   end;
     TextBackground(Blue);     {   Atsisveikinimo lango spalva    }
     Window(20, 2, 70, 4);
     Clrscr;                 { Lango apibrėžimas ir dažymas       }
     Window(22, 3, 68, 3);     {   Langas lange                   }
     TextAttr:= se;                {   Spalvų atstatymas          }
     GoToXY(7, 1);           {   Kursoriaus perkėlimas lange    }
     Write('Atsisveikinu. Paspausk klavišą Enter'^G);
     ReadLn;
     ClrScr;
end.
```

- Pakeiskite programą taip, kad, pakeitus spalvas, būtų dažomos visa eilutė, į kurią rašomas tekstas. Tai galima padaryti įterpiant prieš kiekvieną rašymo ir užklausos veiksmą eilučių valymo procedūrą ClrEol.

- Pakeiskite atsisveikinimo lango sudarymo veiksmus taip, kad šis langas būtų ekrano centre.

8.2. Užduotys savarankiškam darbui.

- Sudarykite sveikųjų skaičių failo duomenų analizės programą, kuri iš failo skaitomus teigiamus skaičius pavaizduotų ekrane mėlynomis, o neigiamus - raudonomis juostomis. Juostų ilgis turi būti proporcingas skaičių moduliams.
- Sudarykite programą, kuri kompiuterio ekrane pieštų atsitiktinio dydžio langus ir juos dažytų atsitiktine spalva. Numatykite darbo pabaigos sąlygą. Ekrane matysite įvairiaspalvę stačiakampių mozaiką.

8.4. Ką turėjau išmokti?

Programos valdymo ir jos pateikiamų duomenų vaizdavimo priemonės bei būdai yra vadinami programos išoriniu interfeisu. Išorinio interfeiso priemonės dažniausiai priklauso ne pačioms programavimo kalboms, bet jų išplėtimams, kurie yra surašomi specialiose bibliotekose. IBM tipo kompiuterių ekrano vaizdo valdymo biblioteka turi vardą **Crt**. Todėl programos, kurioje naudojami ekrano vaizdo valdymo priemonės, pradžioje būtina prijungti šią biblioteką naudojimui, užrašant sakinį: **uses Crt;**

Temos medžiagoje pateiktos 9 dažniausiai vartojamos standartinės funkcijos bei procedūros iš šios bibliotekos. Norint gauti išsamesnių žinių, apie kurią nors iš jų, pakanka TP redaktoriaus lange surinkti jos pavadinimą, prie jo perkelti kursorių ir paspausti klavišų porą Ctrl +F1. Tada ekrane bus matoma pagalbinė informacija apie tą struktūrą.

Procedūroje GoToXY(<x, y> : integer;), kur x, y yra kursoriaus perkėlimo koordinatės, yra laikoma, kad koordinačių pradžia(kurios koordinatės - 1,1) yra ekrano kairiajame viršutiniame kampe, absisių ašis nukripta į dešinę, o ordinačių žemyn. Ekranas yra skaidomas į simbolių rašymui skirtus langelius. Standartinis ekrano dydis 80x25 langelių.

Ekrano ir teksto spalvos nurodomos skaitmeniniais kodais arba modulio Crt suprantamais standartiniais žodžiais. Simbolių, fono spalvas ir ekrano režimą galima keisti tiktai tada, kai pageidaujamus režimus palaiko kompiuteryje esančio ekrano kontroleris (valdymo įtaisas). Fonui galima nurodyti tik pirmąsias aštuonias spalvas. Jei prie spalvos kodo pridėsime žodelį *blink*, gausime pulsuojančią spalvą.

Norint perdažyti ekraną pirmiausiai aprašoma fono spalva, o po to būtina nuvalyti ekraną, kad ji pasikeistų. Keičiant ekrano spalvas, būtina konstanta TextAttr, kuri pradžioje įsimena standartinę ekrano spalvą, o po darbo ją grąžina.

8.5. Kontroliniai klausimai.

1. Kokiu sakiniu yra prijungiama vartojimui spalvų valdymo procedūrų biblioteka?
2. Ką vadinam išoriniu programos interfeisu?
3. Kaip gauti papildomos informacijos apie bibliotekos Crt tam tikrą funkciją ar procedūrą?
4. Kur yra ekrane koordinačių pradžia ir kokios yra jos koordinatės?
5. Kuo skiriasi įprasta koordinačių sistema nuo naudojamos kompiuterio ekrane?
6. Kaip pakeisti ekrano spalvą?
7. Koks standartinis ekrano dydis pagal koordinates?
8. Koks specialūs reikalavimai keliami ekrano spalvoms?
9.Kaip nurodomos ekrano spalvos?
10. Kaip gauti pulsuojančią spalvą?
11. Kaip grąžinti ekranui standartinę spalvą?

8.6 Kontrolinės užduotys.

1. Duota tokia programa:

```
program      Spalvos;
    uses Crt;
     var j,  ek : integer;
    begin
        ek := TextAttr;
        ClrScr;
        j := 1;
        while   j <= 7   do            begin
            TextBackground(j);          WriteLn('fonas:', j);
            TextColor(j+1);                   WriteLn(' spalva: ', j+1);
            j := j+1;
                WriteLn(' Kai pasižiūrėsite, paspauskite ENTER');
                 ReadLn;
         end;
```

```
        TextAttr :=ek;
        ClrScr;
    end.
```

Ar teisinga ši programa? Jeigu ne, kas joje blogai?

2. Ką ji turėtų spausdinti ir kaip?
3. Pataisykite šią programą.

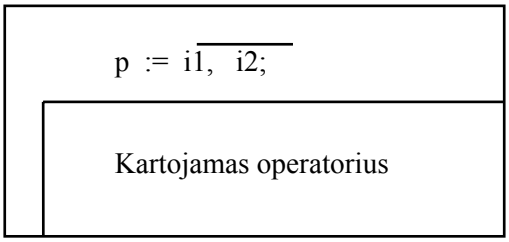
9 tema. Ciklas for. Simbolinis duomenų tipas

9.1. Ciklas for.

Ciklas for yra naudojamas tada, kai iš anksto žinomas ciklo kartojimų skaičius. Ciklo kartojimo metu specialiam diskretinio tipo kintamajam, vadinamam ciklo parametru, iš eilės yra suteikiamos ciklo apraše nurodytos atkarpos reikšmės. Ciklo kartojimų skaičius yra lygus jo parametro reikšmių skaičiui šioje atkarpoje. Operatoriaus sintaksė:

for <p>:= <i1> *to* <i2> *do* <Kartojamas operatorius>;

Apraše **p** žymi ciklo parametrą, o išraiškos **i1** ir **i2** nurodo jo reikšmių atkarpos ribas. Jei **i1>i2** (parametro reikšmių atkarpa yra mažėjanti),tai vietoje bazinio žodžio **to** rašomas žodis **downto**.



Jeigu kartojami veiksmai aprašomi ne vienu operatoriumi arba sakiniu, tai jų pradžioje parašomas žodis *begin* , o gale *end;* Tokia sakinių grupė yra vadinama sudėtiniu sakiniu.

9.2. Simbolinis duomenų tipas

Simbolinio tipo kintamiesiems turi būti nurodomas tipas ***char***, o simbolių reikšmės gali būti nurodomos tiesiogiai, rašant jas tarp kabučių, arba netiesiogiai, vartojant funkciją **Chr**, kurios sintaksė:

function **Chr(<Simbolio kodas>): *char*;**

Pavyzdžiui:

```
var  A, B, C, D : char;
A :='g';   B := Chr( 66 );  Read( C );  D := A;  Write( B );
```

Čia kintamojo A reikšmė bus raidė **g**, kintamojo B bus raidė **B**, kintamojo C bus klaviatūroje paspausto simbolinio klavišo reikšmė, o D reikšmė sutaps su kintamojo A, ekrane bus atpausdinta kintamojo B reikšmė: matysime raidę **B**.
Leistina kodų reikšmių atkarpa yra [0..255]. Dalis šių kodų [0..31] yra skirti duomenų perdavimo ir atvaizdavimo įrenginių valdymui. Kiti simboliai yra skaitmenys [48..57], raidės (didžiosios lotyniško alfabeto raidės [65..90] ir mažosios [97..122]), ir specialūs ženklai (skliaustai, operacijų ženklai, skirybos ženklai ir pan.).
Simbolio kodą galima sužinoti funkcijos **Ord** pagalba:

function **Ord(<Simbolis >): *integer*;**

Funkcijos argumentu gali būti konkretus simbolis, užrašomas tarp apostrofų, arba simbolinio tipo kintamasis, kurio reikšmė yra konkretus simbolis. Pavyzdžiui:

```
var n : integer;
n := Ord( 'A' );
```

Čia n reikšmė bus 65, t.y. nurodyto simbolio kodas (raidės A kodas).

Simbolinio tipo kintamųjų reikšmes galima palyginti. Čia simbolis keičiamas jo kodu. Simbolis yra didesnis už kitą simbolį, jeigu jo kodo reikšmė yra didesnė. Tuo būdu gauname, kad raidė **A** yra mažiausia, nes jos kodas yra 65 mažiausias, o raidė **z** yra didžiausia, nes jos kodas yra 122.
Pavyzdžiui:

```
if  A > B then  Write( A )   else  Write( B );
```

atspausdins mūsų atveju kintamojo A reikšmę raidę **g** , nes jos kodas didesnis už kintamojo B reikšmę (raidė **B**)

9.3. Pratimai.

- Programa **Simboliai** į ekraną išveda simbolių ir jų kodų lentelę. Patikrinkite simbolių ir jų kodų demonstravimo programos darbą.

```
program  Simboliai;
uses Crt;
var  i: integer;                    (* Ciklo parametras              *)
begin
  ClrScr;
  WriteLn(' Kompiuterio alfabeto simboliai ir jų kodai');
  for   i:= 1  to  255  do          (* Kodų generavimo ciklas        *)
  begin
      Write (chr(i): 3, ':', i: 5);    (* Simboliai ir jų kodai      *)
      if  i  mod 7  = 0 then  WriteLn;  (* Vienoje ekrano eilutėje *)
  end;                              (*po 7 simbolius rodome    *)
  WriteLn;
  ReadLn;
end.
```

Stebėdami programos Simboliai darbo rezultatus, pamatysite, kad kompiuterio alfabetą sudaro ne tik tradiciniai rašymui skirti simboliai bet ir įvairūs remelių bei kitokių grafinių struktūrų sudarymui skirti simboliai. Jie dar yra vadinami pseudografikos simboliais. Pavyzdžiui, ekrano langelių dažymui visomis 16 - spalvų galima naudoti simbolį '■', kurio kodas yra 178.

- Patikrinkite spalvų kodų demonstravimo programos Spalvos darbą.
-

```
program Spalvos;
uses Crt;
var  i, j, se : integer;
begin
  se:= TextAttr;
  WriteLn(' Demonstruojamos ekrano spalvos ir jų kodai:');
  WriteLn;
  for i:= 0 to 15 do begin          (* Spalvų kodų generavimas     *)
      TextColor(white);
      Write('Spalvos kodas ', i:3, '  ');
      TextColor(i);
      for j:=1 to 20 do             (* Juostos iš 20 simbolių dažymas *)
            Write(  Chr(178)  );
```

```
            WriteLn;
      end;
  ReadLn;     TextAttr:= se;      ClrScr;
end.
```

- Savarankiškai sudarykite programą, kuri ekranui suformuotų mėlynos spalvos rėmelį. Rėmelio simbolius parinkite iš kompiuterio simbolių alfabeto patys.

9.3. Užduotis savarankiškam darbui.

7.2. skyrelyje buvo nagrinėtas atsitiktinių skaičių generavimo būdas kreive ribojančiam plotui skaičiuoti. Papildykite programą veiksmais, kurie leistų vizualiai ekrane pavaizduoti skaičiuojamus taškus: jeigu kreivės viduje, tai žalias simbolis, o išorėje raudonas. Taškus vaizduokite simboliu '*', kurio fonas atitiktų nurodytas spalvas. rezultatus pavaizduokite balto fono lange, kurio dydis 20x10 (čia x=20, y=10). Tam reikia sinuso skaičiavimo reikšmes 10 kartų padidinti.

9.4. Ką turėjau išmokti?

Ciklas for yra naudojamas tik tada, kai iš anksto yra žinomas tikslus kartojimų skaičius. Operatoriaus sintaksė:

for <p> := <i1> *to* <i2> *do*
<Kreipimosi operatorius>;

čia **p** sveiko tipo kintamasis, kuris vadinamas ciklo parametru. **i1**, **i2** žymi ciklo parametro kitimo ribas. Jos gali būti išreikšos sveikais skaičiais, sveiko tipo kintamaisiais arba išraiškomis, kurių skaičiavimo rezultatas yra diskretinio dydžio ir tipas sutampa su parametro tipu. Ciklo parametras **p** keičiasi nuo **i1** iki **i2** tik kas vienetą. Jeigu **i1>i2**, tai tada parametro **p** reikšmė mažėja kas vienetą ir vietoje bazinio žodžio *to* reikia vartoti *downto*. Jeigu reikia kartoti ne vieną sakinį, o jų grupę, tai vartojame operatorinius skliaustus. Tada žinomo kartojimų skaičiaus sakinys atrodys taip:

```
for <p> := <i1> to <i2> do
begin
   <kartojami operatoriai>;
end;
```

Tokia sakinių grupė yraa vadinama sudėtiniu sakiniu.

Simbolinio tipo kintamieji yra tokie, kurie programos darbo metu įgyja simbolių reikšmes, t.y. raidžių, operacijos ženklų. Jie *var* srityje aprašomi naudojant bazinį žodį *char*. Simbolių reikšmės šiems kintamiesiems gali būti rašomos tiesiog kabutėse arba su funkcija Chr, kurios bendra išraiška yra:

function Chr(<simbolio kodas>) : char;

Leistina kodų reikšmių atkarpa [0 .. 255]. Simbolio kodą galima sužinoti funkcijos Ord pagalba. Jos sintaksė tokia:

function Ord(<simbolis>) : integer;

čia simbolis yra pats abėcėlės elementas, parašytas kabutėse, kurio kodą norime sužinoti, arba simbolinio tipo kintamasis. Simbolinio tipo kintamuosius galima palyginti santykio ženklais. Didesniu laikomas tas simbolis, kurio kodas yra didesnis.

Tarp 255 simbolių kodų yra ne tik mums įprasti, bet ir psiaudografikos simboliai.

9.5. Kontroliniai klausimai.

1. Ką vadinam sudėtiniu sakiniu?
2. Kokia yra žinomo kartojimų skaičiaus sakinio struktūra?
3. Kokios gali būti ciklo parametro reikšmės?
4. Kada sakinyje vartoti bazinį žodį *to*, kada *downto*?
5. Kokie kintamieji yra simbolino tipo?
6. Kaip sužinoti simbolio kodą?
7. Kaip galima parašyti simbolį?
8. Ką duoda funkcija Chr?
9. Pagal ką lygimami simboliai?
10. Pateikite bent vieną psiaudografikos pavyzdį.

9.6 Kontrolinės užduotys.

1. Duotas toks kartojimo sakinys:

```
for       i := 1     to   5   do
  begin
    WriteLn('spausdinu ', i, ' -jį kartą');
```

```
    i := i + 2;
  end;
```

Kiek kartų kartojimo sakinys spaudins pranešimą?
Ar negalima šio sakinio supaprastinti? Jeigu ne, tai kodėl?

2. Kaip reikia pakeisti pirmosios užduoties kartojimo sakinį, kad programa spausdintų tik lygines i reikšmes?

3. Sudaryta tokia programa:

```
program      Atbulai;
 var      a, b, x, s : integer;
  begin
   WriteLn('Įveskite intervalo [a, b] kraštines reikšmes');
    ReadLn(a, b);
    s := 0;
    for    x := a    to    b    do
           s := s + x;
    WriteLn('Intervale [a,b] sveikų skaičių suma = ', s);
  end.
```

Kai paprašė įvesti intervalo reikšmes, surinkote:
40 30 Enter;
Kokį spaudins atsakymą? Kodėl?
Ką reikia pakeisti kartojimo sakinyje, kad atsakymas būtų: "Intervale [a,b] sveikųjų skaičių suma = 385", įvedus tuos pačius skaičius ir tokia pačia tvarka?

4. Duota programa:

```
program      Kodai;
 uses Crt;
  var    x : char;
       i, n : integer;
   begin
    ClrScr;
    Randomize;
    for    i := 1    to    10    do
      begin
              n := Random(150);
              x := Chr(n);
```

```
            WriteLn('.....', n, '.....', x);
        end;
    end.
```

Daugtaškių vietoje įrašykite komentarus atsakymams.
Kaip pakeisti pirmąjį priskyrimo sakinį, kad nespausdintų valdančių simbolių?

10 tema. Garsai. Ciklas repeat...until.

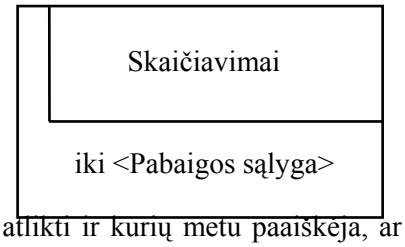
10.1. Garsai.

Modulyje **Crt** yra šios garsų valdymo procedūros:

procedure **Sound(<Dažnis>: *word*);**
Garsiakalbio įjungimo, nurodant garso dažnį, procedūra.
procedure **Delay(<Trukmė milisekundėmisd>: *word*);**
Pauzė, kurios metu kompiuteris veiksmų neatlieka.
procedure **NoSound;** Garsiakalbio išjungimas.

Visos trys šios procedūros yra vartojamos kartu - įjungiamas pageidaujamo dažnio (tono) garsas, leidžiama jam kurį laiką skambėti - Delay ir jis išjungiamas arba keičiamas tono aukštis.

10.2. Ciklas repeat...until.

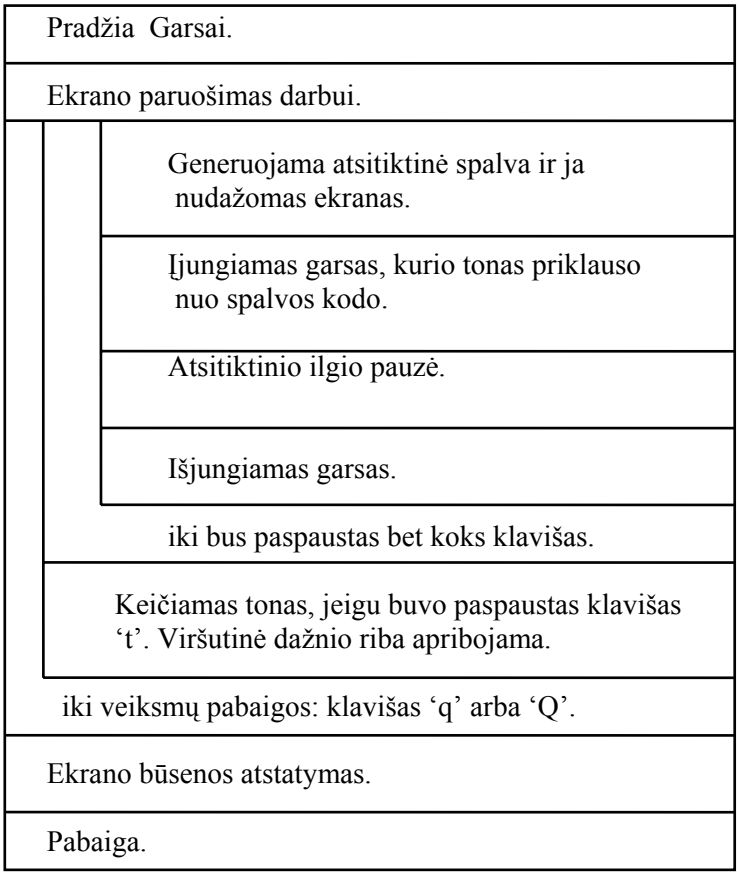


Cikliniams procesams aprašyti buvo naudojamos dvi ciklų formos:
while ir *for*.
Esti tokių ciklinių skaičiavimų, kuriuos visuomet bent kartą būtina atlikti ir kurių metu paaiškėja, ar skaičiavimai bus tęsiami. Tokius veiksmus patogu aprašyti ciklo sakiniu *repeat...until*.
Skaičiavimai vykdomi tol, kol ciklo pabaigos sąlygos rezultatas bus reikšmė False. Skaičiavimuose turi būti veiksmai, kurių metu gaunamos reikšmės keičia arba nekeičia ciklo pabaigos sąlygos rezultatą, tačiau būtina, kad kada nors būtų gautos reikšmės, kurioms esant ciklas baigtusi. Ciklo sakinio struktūra:

repeat **<Skaičiavimai>** ***until*** **<Pabaigos sąlyga>**

10.3. Pratimai.

- Atsitiktinis spalvų ir garsų generavimas. Garsiakalbio valdymą geriausiai aprašyti natomis, bet įdomius efektus galima gauti ir naudojant atsitiktinius skaičius. Kaip tai yra daroma, iliustruojama pavyzdyje.




```
program Garsai;
uses    Crt;
var     sb, se: byte;           (*  Spalvos     *)
        bg,  g: longint;        (*  Garsai      *)
          t,  p: integer;       (*  Tempas  *)
              c: char;
 begin
     TextBackground (Black);
     se:= TextAttr;             (*  Spalvų įsiminimas    *)
     ClrScr;                    (*  Ekrano valymas       *)
     GoToXY(20,  25);           (*  Kursoriaus perkėlimas    *)
```

```
    Write('Tonas - "t". Pabaiga - "q". ');
    Window(10, 3, 70,  20);
    bg:= 1000;                  (*   Bazinis tonas              *)
    t:= 200;                    (*   Bazinis tempas             *)
    repeat
    repeat
      Randomize;  sb:=Random(6)+1;(* Atsitiktiniai skaičiai *)
      TextBackground(sb); ClrScr;  (* Lango dažymas          *)
      p:= bg div 10+20;
      Sound(bg+ p * sb);           (* Tono keitimas          *)
      Delay(t + Random(t));   NoSound;    (* Trukmė          *)
    until KeyPressed;              (* Klavišų kontrolė  *)
      c:= ReadKey;                      (* Klavišo kodas          *)
      if  c='t'  then  bg:= bg+ 500   (* Ar keisti toną?        *)
      if  bg>5000  then  bg:= 10;   (* Ribojimas dažniui      *)
    until  (c= 'q')  or  (c= 'Q');   (* Nutraukimo sąlyga     *)
    TextAttr:= se;     ClrScr;           (* Spalvų atstatymas      *)
end.
```

- Pakeiskite programą taip, kad su klavišu "g" būtų galima keisti kompiuterio grojamos melodijos greitį, o ekrano centre būtų matomas pranešimas apie bazinį dažnį bg. Greitį keičia programos parametras **t**.

- Programoje **GarsoLangai** keičiami spalvotų langų dydžiai. Grojanti programa bus įdomesnė, jei sinchroniškai keisis ne tik ekrano spalvos, bet ir dažomų langų dydžiai. Kaip tai yra daroma iliustruojama programoje.

```
program GarsoLangai;
uses      Crt;
var       sb, se: byte                (*   Spalvos       *)
          bg,  g: longint;            (*   Garsai        *)
          t,    p: integer;           (*   Tempas  *)
                c: char;
     x1, y1, x2, y2: byte;            (* Langų koordinatės *)
begin
  TextBackground(Black);      se:= TextAttr;          ClrScr;
  GoToXY(20, 25);                  (* Kursoriaus perkėlimas *)
  Write('Tonas - "t". Pabaiga - "q". ');          Window(1, 1, 80, 24);
  bg:= 1000;   t:= 200;                      (* Bazinis tonas ir greitis*)
repeat
```

```
  repeat
     Randomize;          sb:=Random(6)+1;
     Randomize;                    (* Atsitiktinės koordinatės *)
      x1:=  1 + Random(50);        y1:=    2+Random(12);
     x2:= x1 + Random(80-x1);     y2:= y1 + Random(25-y1);
     Window(x1, y1, x2, y2);
     TextBackground(sb);   ClrScr;        (* Lango dažymas  *)
                                    (* Garsiakalbio įjungimas   *)
     p:= bg div 10+20;   Sound(bg+ p * sb);
     Delay(t + Random(t));   NoSound;        (* Trukmė          *)
  until KeyPressed;                          (* Klavišų kontrolė *)
  c:= ReadKey;                                  (* Klavišų apklausa *)
     if c='t' then bg:= bg+ 500;  (* Bazinio dažnio keitimas       *)
     if bg>5000 then bg:= 10;
until c= 'q';
TextAttr:= se;              ClrScr;
end.
```

- Garsiniai ir vaizdiniai efektai dažnai vartojami programų darbo pradžioje ir gale. Pavyzdžiui, programos prisistatymui galima vartoti "augantį" ekrano vaizdą. Kaip tai padaryti, iliustruojama programoje Auga.

```
program Auga;
uses Crt;
var     s, x1, x2, y1, y2 : byte;   (* Spalvos ir langų koordinatės *)
                  i, j, d : integer;
begin
  TextBackground (Black);      ClrScr;      (* Ekrano valymas *)
  s:= TextAttr;                GoToXY(30,  1);
  Write('Langų augimo demonstravimas');  Delay (1000);
  for i:=1 to 12 do                          (* Langų dažymo ciklai *)
    begin
      TextBackground(Blue);                 (* Lango spalva              *)
                                  (* Dydžio keitimas ir dažymas  *)
      Window(37-3*i,   13-i,   44+3*i,   13+i);    ClrScr;
      Window(37, 13, 44, 14);  Write ('Startas!');(* Pranešimas *)
      d:= -50;
      for   j:= 1   to   3   do       (* Garsinio efekto ciklas *)
          begin
           d:= d*(-1);                    Sound(500 + d*j);
           Delay(100 - (d*j) div 2);  NoSound;
```

```
            end;
        end;
       Delay(1000);
       for   j:= 1   to   3   do                (* Atsisveikinimo ciklas *)
        begin
          Sound(50); Delay(300);          NoSound;
          Window(37,  13, 44,  14);              Write ('Bye-bye!');
          Delay(500);                    ClrScr;
        end;
       TextAttr:= s;        ClrScr;
end.
```

- Pakeiskite programą taip, kad langas ne didėtų, o nuosekliai mažėtų.

10.4. Savarankiško darbo užduotis.

Parinkite kiekvienai lotyniško alfabeto raidei skirtingo tono ir dažnio garsus. Iš klaviatūros įvedinėjamas raides užrašykite atsitiktine spalva atsitiktinėje ekrano vietoje, palydint atitinkamais garsais. Numatykite darbo pabaigos formą.

10.5. Ką turėjau išmokti?

Modulyje Crt be ekrano valdymo struktūrų yra ir garso valdymo trys procedūros: *Sound, Delay, NoSound*. Jungiant šias tris procedūras galima gauti įvairius garsus.

Cikliniams procesams aprašyti be operatorių *while* ir *for*, yra dar vienas *repeat*. Jis vartojamas tuo atveju, jeigu kartojimo veiksmus reikia atlikti bent vieną kartą, o tik po to paaiškėja ar jie bus kartojami toliau. Šio ciklo užrašymo sintaksė tokia:

repeat <skaičiavimai> until <pabaigos sąlyga>;

Skaičiavimai, priešingai negu cikle *while*, yra kartojami tol, kol pabaigos sąlygos rezultatas yra *false*. Ciklo viduje skaičiavimai turi būti vykdomi taip, kad kada nors ciklo pabaigos sąlygos rezultatas įgytų reikšmę *false*, kitaip ciklas bus begalinis.

10.6. Kontroliniai klausimai.

1. Kokios yra garso valdymo procedūros modulyje Crt?
2. Kada kartojimams aprašyti yra vartojamas ciklo sakinys *repeat*?

3. Kuo *repeat* skiriasi nuo *while*?
4. Kaip reikia organizuoti skaičiavimus ciklo viduje?

10.7 Kontrolinės užduotys.

1. Duota programa:

```
program      Melodija;
  {garsas}
  begin
    Sound(500);
    Delay(2000);
    NoSound;
  end.
```

Ar ji teisinga? Jeigu ne pataisykite.

2. Ką girdėsime įvykdę šią programą? Padarykite, kad garsas ir tyla periodiškai pasikartotų 5 kartus.
3. Duotas kartojimo sakinys:

```
repeat
   sum  :=  sum + x;
   x      :=  x     + 3;
until   (x < 10);
```

Kiek kartų bus pakartotas sumavimas, jeigu programos pradžioje buvo parašytas toks sakinys: x := 3;? Kodėl?

4.Kaip pasikeis programa, jeigu priskyrimo sakinys bus:
x := 15? Kodėl?

5. Pataisykite kartojimo sakinį taip, kad jis sumuotų sveikus skaičius, besidalinančius iš trijų intervale [0, 20].
6. Reikia įvedinėti skaičių seką, kol bus įvestas neigiamas skaičius ir suskaičiuoti kiek kartų buvo įvestas nulis. Kurį kartojimo sakinį repeat ar while geriau naudoti ciklo organizavimui. Įrodykite tai sudarydami dvi programas.

Užduotys savarankiškam darbui.

Pirmas variantas.

U-1. Klaviatūra įvedami trys skaičiai, kurie reiškia trikampio kraštinių ilgius. Suskaičiuoti ir atspausdinti trikampio plotą.

U-2. Klaviatūra įvedami du skaičiai, kurie reiškia stačiakampio kraštinių ilgius. Suskaičiuoti ir atspausdinti stačiakampio plotą.

U-3. Klaviatūra įvedami keturi skaičiai, kurie reiškia atkarpos, nubrėžtos koordinačių plokštumoje, galų taškų koordinates (x, y). Suskaičiuoti ir atspausdinti atkarpos ilgį.

U-4. Klaviatūra įvedami trys skaičiai, kurių pirmas reiškia apskritimo spindulį, o kiti du apskritimo, nubrėžto koordinačių plokštumoje, centro koordinates (x,y). Suskaičiuoti ir atspausdinti apkritimo ilgį ir centro atstumą nuo koordinačių centro.

U-5. Klaviatūra įvedami du skaičiai, kurie reiškia taško koordinates plokštumoje (x, y). Suskaičiuoti ir atspausdinti taško atstumą nuo koordinačių pradžios taško.

U-6. Klaviatūra įvedami keturi skaičiai, kurie reiškia stačiakampio, nubrėžto taip, kad kraštinės būtų lygiagrečios koordinačių ašims, priešingų dviejų kampų koordinates (x,y). Suskaičiuoti ir atspausdinti stačiakampio plotą.

U-7. Klaviatūra įvedami trys skaičiai, kurie reiškia kvadratinės lygties $ax^2+bx+c=0$ koeficientus. Suskaičiuoti ir atspausdinti šaknų reikšmes. *Pastaba*: koeficientų reikšmės įvedamos tokios, kurioms esant sprendiniai bus.

U-8. Klaviatūra įvedami šeši skaičiai, kurie reiškia trikampio, nubrėžto koordinačių plokštumoje, viršūnių koordinates (x,y). Suskaičiuoti ir atspausdinti trikampio kraštinių ilgius.

U-9. Klaviatūra įvedami keturi skaičiai, kurie reiškia stačiakampio, nubrėžto taip, kad kraštinės būtų lygiagrečios koordinačių ašims, priešingų dviejų kampų koordinates (x,y). Skaičius parinkite taip, kad jie apibrėžtų kairio viršutinio kampo ir dešinio apatinio kampo

koordinates. Suskaičiuoti ir atspausdinti stačiakampio centro koordinates bei jo atstumą nuo koordinačių pradžios taško.

U-10. Klaviatūra įvedami keturi skaičiai, kurie reiškia atkarpos, nubrėžtos koordinačių plokštumoje, galų taškų koordinates (x, y). Suskaičiuoti ir atspausdinti apskritimo ilgį ir centro koordinates, jeigu atkarpa apskritimui yra diametras.

Antras variantas.

Užduotys tos pačios. Duomenų yra daug. Reikia su visais atlikti skaičiavimus. *Sugalvokite* duomenų srauto, įvedamo iš klaviatūros, pabaigos požymį ir realizuokite programoje.

Trečias variantas.

Papildykite ir modifikuokite savo ankstesnę programą. Duomenys yra surašomi iš anksto į tekstinį failą. Programa juos skaito, atlieka nurodytus skaičiavimus ir išveda rezultatus į kitą tekstinį failą su paaiškinimais ir pradiniais duomenimis. Tarp duomenų gali būti ir tokių, kurie Jūsų uždaviniui bus netinkami (pvz., kraštinės ilgis neigiamas arba nulinis). Juos praleiskite.

Ketvirtas variantas.

Papildykite ir modifikuokite savo ankstesnę programą. Sugalvokite ir realizuokite programoje duomenų ir rezultatų pavaizdavimą kompiuterio ekrane tokį, kuris būtų gražus, aiškus ir suprantamas. Tam naudokite spalvas ir ekraninius langus.

Penktas variantas.

Papildykite ir modifikuokite savo ankstesnę programą naujais nurodytais skaičiavimais. Sugalvokite ir realizuokite papildomas priemones, kurios bus naudingos dialogui su vartotoju.

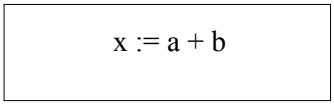
U-1. Suskaičiuoti ir atspausdinti visų trikampių plotą ir stačių trikampių kiekį.
U-2. Suskaičiuoti ir atspausdinti visų stačiakampių plotą ir kiek buvo kvadratų.
U-3. Suskaičiuoti ir atspausdinti visų atkarpų ilgius ir kiek buvo atkarpų, kurios pilnai nubrėžtos pirmame ketvirtyje.

U-4. Suskaičiuoti ir atspausdinti visų apkritimų ilgius ir jų atstumus iki koordinačių centro ir kiek yra apskritimų pilnai nubrėžtų tik pirmame ketvirtyje.
U-5. Suskaičiuoti ir atspausdinti visų taškų atstumus iki koordinačių pradžios taško. Suskaičiuoti vidutinį taško atstumą iki koordinačių pradžios taško.
U-6. Suskaičiuoti ir atspausdinti visų stačiakampių plotą ir rasti stačiakampių, kurie yra pilnai pirmame ketvirtyje, plotų sumą.
U-7. Suskaičiuoti ir atspausdinti visų lygčių šaknų reikšmes. Jeigu realios šaknys neegzistuoja, tuomet tai nurodyti. Suskaičiuokite visų šaknų aritmetinį vidurkį.
U-8. Suskaičiuoti ir atspausdinti visų trikampių kraštinių ilgius. Suskaičiuoti vidutinį kraštinės ilgį.
U-9. Suskaičiuoti ir atspausdinti visų stačiakampių centrų koordinates bei jų atstumus nuo koordinačių pradžios taško. Kiek tų centrų yra pirmame ketvirtyje.
U-10. Suskaičiuoti ir atspausdinti visų apskritimų ilgius ir centrų koordinates, jeigu kiekviena atkarpa yra apskritimo diametras. Kurie apskritimai yra nubrėžiami tik pirmame ketvirtyje? Kiek yra tokių apskritimų?

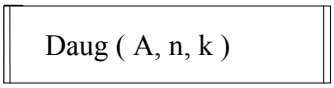
Struktūrogramų elementai

(Nassi-Schneiderman diagrams)

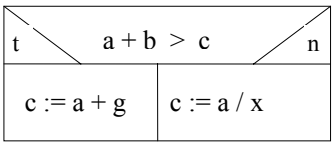
Uždavinių sprendimo algoritmus vaizduoti galima įvairiais būdais: pseudokodu (labai detaliai), simbolinėmis (“blokinėmis”) schemomis, struktūrogramomis. Programuotojai renkasi labiausiai atitinkantį jų konkrečius poreikius būdą. 1973 metais amerikiečių autoriai I.Nassi ir B.Shneiderman pasiūlė algoritmų užrašymo būdą, kuris žinomas struktūrogramų vardu (Vakaruose žinomas vardu Nassi-Schneiderman diagrams). Tai būdas, kuris gerai tenkina struktūrinio programavimo technologijos reikalavimus: vaizdus, lakoniškas, paprastas, neleidžiantis formuoti nestruktūrinių elementų. Tobulėjant programavimo priemonėms, struktūrogramų elementai taip pat dalinai pakito. Taip, pavyzdžiui, atsitiko su *Išrinkimo* pagal požymį struktūrogramos elementas: autorių pasiūlyta forma retai naudojama. Toliau yra parodyti pagrindiniai struktūrogramų elementai.



Priskyrimas. Duomenų įvedimas bei išvedimas. Duomenų tipų aprašymas. Kintamųjų aprašymas. Programos bloko vardas.

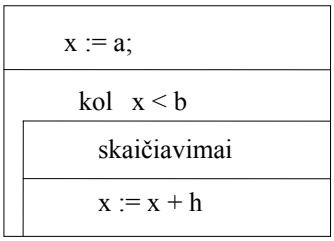


Kreipinys į procedūrą.



Sąlygos sakinys:

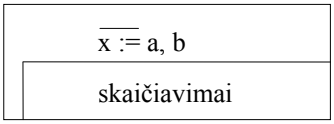
```
if a + b > c then    c := a+g
                else     c := a/x;
```



Ciklo sakinys:

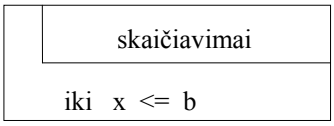
```
x := a;

while x <= b do begin
          (* skaičiavimai *)
      x := x + h;
end;
```



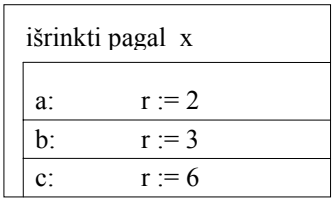
Ciklo sakinys

```
for x := a to b do begin
     (* skaičiavimai *)
end;
```



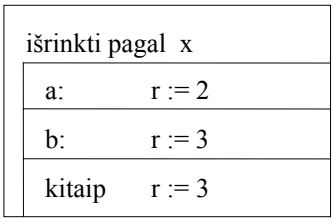
Ciklo sakinys

```
repeat
    (* skaičiavimai *)
until   x <= b;
```



Išrinkimas pagal požymį

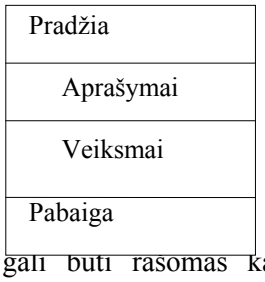
```
case X of
  a : r := 2;
  b : r := 3;
  c : r := 6;
end;
```



Išrinkimas pagal požymį

```
case X of
  a : r := 2;
  b : r := 3;
 else r := t;
end;
```

Programos, paprogramių, programinio bloko struktūrograma.



Pradžia. Čia užrašomas programos vardas, paprogramės antraštė arba programos bloko vardas. *Programos blokas* yra programos teksto dalis, kuri sudaro vieningą veiksmų grupę ir kuri, kaip taisyklė, užrašoma tarp skliaustų *begin ... end***.** Programoje bloko vardas gali būti rašomas kaip komentarai bloko pradžioje.

Aprašymai. Ši dalis gali būti sudaryta iš kelių specializuotų skyrių, kuriuose aprašomi konstantos, duomenų tipai, kintamieji ir kitos struktūros. Surašant kintamuosius siūloma juos grupuoti pagal paskirtį ir tipus. Rekomenduotina tokia seka: duomenis aprašantys kintamieji, rezultatus aprašantys kintamieji ir įvairūs pagalbiniai kintamieji. Šios dalies gali ir nebūti.

Jeigu aprašų dalis yra didelė, ją galima aprašyti atskirai už struktūrogramos ribų. Kintamieji, kurių paskirtis ir tipai akivaizdūs, kaip pvz., ciklo for parametras, gali būti visai nenurodomi aprašymuose. Rašant programos tekstą, būtina įsitikinti, kad visi kintamieji aprašymuose buvo nurodyti. Jeigu ne, tuomet reikia, peržvelgus algoritmą, papildyti programos kintamųjų sąrašą.

Veiksmai. Čia parodomi uždavinio algoritmą sudarantys veiksmai, jų loginė seka. Kiekvienas struktūrinis elementas gali būti sudėtingas: jo viduje gali būti braižomi kiti elementai. Labai sudėtingus elementus tikslinga vaizduoti atskiromis struktūrogramomis. Pagrindinėje struktūrogramoje tokie elementai nurodomi bloko vardu. Taip gaunamos hierarchiškai susietos strutūrogramos.

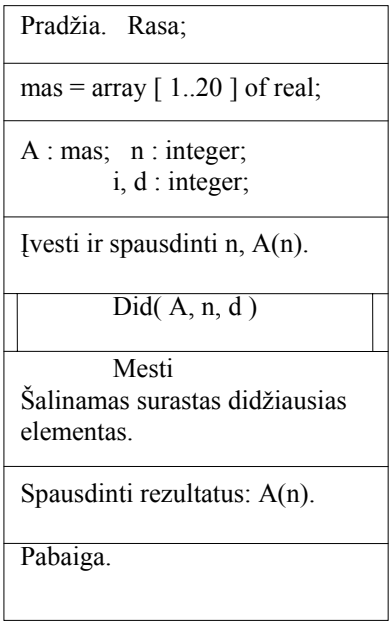
Hierarchinėse sistemose pirmiausia aprašoma uždavinio sprendimo eiga stambiomis funkcinėmis veiksmų grupėmis. Šios grupės yra komentuojamos ir o po to atskirai yra detalizuojamos. Tai leidžia

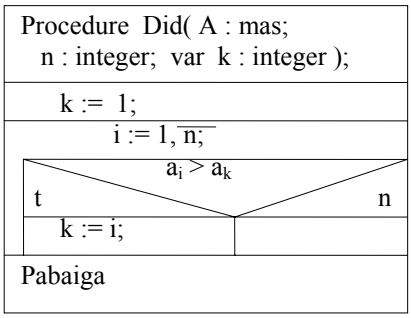
lengviau suvokti uždavinio sprendimo esmę, be to detalizacijos lygį galima riboti pagal tai, kokią patirtį turi programuotojas, kuris rašo tekstą. Detalizacijos lygis taip pat priklauso nuo algoritmo pobūdžio ir nuo to, kiek jis susietas su realizacija konkrečia programavimo kalba. Struktūrograma turi būti piešiama visa: nuo pradžios iki pabaigos, t.y. negalima nutraukti jos paveikslą bet kurioje vietoje ir tęsti kitoje vietoje ar kitame lape. Jeigu struktūrograma netelpa, tuomet dalį veiksmų įvardiname bloku ir atskirai detalizuojame. Paprogramių algoritmai braižomi atskirai. Algoritme kreipiniui į procedūrą yra skiriamas vienas atskiras elementas. Kreipinys į funkciją yra išraiškose, todėl jam nėra skiriamas atskiras elementas.
Pabaiga. Čia galima užrašyti struktūrogramos antraštėje esantį vardą.
Komentarai struktūrogramoje rašomi tuose elementuose, kurių veiksmus jie aiškina. Juos galima surašyti elemento viršuje, apačioje arba šalia veiksmų, tačiau siūloma rašyti taip, kad juos galima būtų perkelti į programos tekstą be papildymų arba pataisymų.

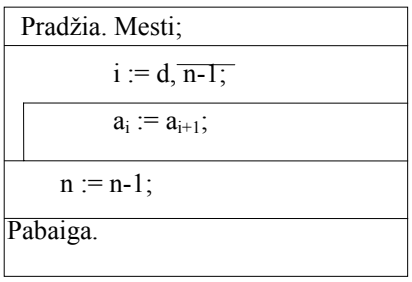
Pavyzdys.

Užduotis. Turime realių skaičių masyvą. Reikia surasti ir pašalinti didžiausią skaičių.

Uždavinio algoritmas **Rasa** iliustruoja kreipinį į procedūrą **Did** ir sudėtinį sakinį **Mesti**. Atskirai yra parodytas procedūros **Did** algoritmas. Sudėtinio sakinio **Mesti** detalizacija rodoma atskirai tikslu logiškai atskirti uždavinio veiksmų loginę seką nuo jų detalizacijos. Toks algoritmo detalizacijos būdas naudojamas tuomet, kai algoritmas netelpa viename lape, nes algoritmo pradžia ir pabaiga negali būti atskiruose lapuose: visas aprašymas turi būti vieno stačiakampio viduje.







Programavimo kultūra.

Algoritmas (programa) tampa žmogaus intelektualinės veiklos produktu, skirtu tiek kompiuteriui, tiek žmogui. Jis turi atitikti ir kompiuterio, ir žmogaus reikalavimus. Kompiuterio keliami reikalavimai: algoritmas turi būti užrašytas taip, kad jį galima būtų koduoti programavimo kalba ir, gautą programą vykdant, gauti teisingus rezultatus. Žmogaus keliami reikalavimai: algoritmas turi būt toks, kad jį galima būtu kuo lengviau suprasti. Programa kompiuteriui gali būti charakterizuojama dviem dydžiais: atminties kiekiu ir vykdymo trukme. Algoritmas žmogui charakterizuojamas laiku, per kurį žmogus suvokia algoritmą. Darbas su kompiuteriu bus sėkmingas, jei dirbantysis skaitydamas programą sugaiš kuo mažiau laiko, greitai ir lengvai ją supras.

Pagrindiniai programavimo kultūros elementai:

- algoritmas parenkamas toks, kad jis kuo geriau tiktu duotam uždaviniui spręsti: *būtų aiškus, trumpas, logiškai pagrįstas;*
- algoritme turi būti viskas, ko reikalauja uždavinio sąlyga, bet neturi būti nereikalingų elementų ar konstrukcijų;
- veiksmai surašomi logiškai pagrįsta eilės tvarka;
- algoritmo struktūra turi būti *aiški* ir atspindėti uždavinio struktūrą;
- savarankiškos algoritmo dalys aprašomos procedūromis bei funkcijomis su minimaliais ryšiais tarp jų;
- duomenų tipai ir struktūros parenkami taip, kad jie atspindėtų jų atstovaujamų objektų prasmę;
- kintamųjų, konstantų, tipų, procedūrų, funkcijų vardai turi būti prasmingi ir galimai trumpesni;
- programos tekstas išdėstomas laikantis tam tikrų logiškai pagrįstų taisyklių;
- komentarai turi susieti uždavinio formuluotę su algoritmu.

Programos rašomos ne tiktai tam, kad jos teiktų informaciją kompiuteriui, bet ir tam, kad jas skaitytų žmonės dėl tokių priežasčių:

- programos labai retai užbaigiamos per pirmą kartą. Parašyti užbaigtą programą reikia keletos dienų ir netgi metų. Dažnai reikia peržiūrėti ką esi padaręs;
- programa labai retai iš karto būna teisinga; tikrinimo procese ji perskaitoma kelis kartus iš naujo;
- uždavinys labai dažnai padalomas tarp keletos vykdytojų, kurie turi skaityti ir suprasti kitų programas, kad galėtų tinkamai bendradarbiauti;
- programų dalys dažnai naudojami kituose uždaviniuose. Tam tikslui jas irgi reikia skaityti ir suprasti, kas daroma toje dalyje;
- programos dažnai yra keičiamos, nes keičiasi užduotis.

Programos stilius vertinamas tokiais pat kriterijais kaip ir natūralios kalbos stilius. Programos tekstas turi būti aiškus, vaizdus, lengvai skaitomas ir suprantamas, save dokumentuojantis. Rašydami programą, turime galvoti, kaip padėti skaitytojui.

Programų komentarai

Komentarų buvimas ar nebuvimas ir jų turinys visai neturi įtakos programos darbui. Bet vartotojas, kuris skaitys programą, galės pasirinkti ką skaityti: *tik komentarą, ar visą programą.*

Jis gali skaityti komentarą, jei nori sužinoti, ką daro programa arba skaitys programą, jei norės sužinoti, kaip ji atlieka užduotį. Komentarai programą turi papildyti, bet netrukdyti ją skaityti, todėl juos siūloma rašyti programos dešiniame krašte. Komentarų skliaustai pagal galimybę atitolinami nuo teksto, lygiuojami vertikaliai iš abiejų pusių (sudaromas komentarinio bloko vaizdas).

Komentarai neturi trukdyti programos skaitymui, t.y. jie turi būti trumpi, lakoniški. Jie turi tik taikliai papildyti programą, o ne užgožti ją. Komentarus reikėtų atitolinti nuo programos teksto taip, kad akys jų "nepastebėtų" ir lengvai atskirtų. Jų neturi būti per daug, bet jie turi pakankamai dokumentuoti programą, kad nebereikėtų kitų dokumentų, išskyrus programos tekstą ir instrukciją vartotojui. Komentarai gali užimti apie 20% programos apimties. Rekomenduojama rašyti komentarus :

- programos pradžioje po antrašte (arba prieš), nurodant programos paskirtį, autorių, programos sudarymo datą ir kitus duomenis;
- tipų bei kintamųjų aprašuose, nurodant jų paskirtį;

- prieš procedūras, funkcijas, blokus, nurodant kokias funkcijas atlieka;
- prieš išorinius modulius, blokus, nurodant jų atliekamą funkciją, įėjimo, išėjimo duomenis;
- sakiniuose, paaiškinant neaiškios paskirties veiksmus, pavyzdžiui, sąlygos sakinyje komentuoti atskirai *then* ir *else* dalis.

Komentarai nuo programos teksto gali būt atskirti tuščiomis eilutėmis, jeigu jie užima visą eilutę.

***Taip nereikia rašyti komentarų aprašant tipus ir kintamuosius**:*

```
type  tekstas    = string ( 15 );
      pazymiai = array [ 1..5 ] of integer;
      e          = record
                   pav,  (*  studento pavardė *)
                   vard : tekstas; (* jo vardas *)
                   p : pazymiai; (* sesijos pažymiai *)
                   vid : real; (* sesijos vidurkis *)
                 end;
      mas    =  array [ 1..50 ] of  e;
var
      if21,  if22:  mas; (* studentų sąrašai *)
      n, m: integer: (* sąrašų ilgiai *)
      i, j: integer;       s, v: real;
```

Komentarai turėtų neužgožti programos teksto:

```
type
   teks = string ( 15 );
   paz  = array  [ 1..6 ] of integer;
   egz  = record
          pav,             (*   studento pavardė  *)
          vard : teks;     (*     jo vardas                *)
          p     : paz;     (*   sesijos pažymiai           *)
          vid   : real;    (*     vidurkis                 *)
        end
   mas =   array [  1..50  ]  of   egz;
var
   if21, if22 : mas;     (*     studentų sąrašai  *)
   n,    m,              (*    jų ilgiai                 *)
   i, j     : integer;   (*      pagalbiniai             *)
   s, v          : real; (*     kintamieji               *)
```

Programų teksto rašymo kultūra

Tekstas rašomas laikantis bendrų rašybos taisyklių. Tarp žodžių paliekamas bent vienas tarpas. Paliekami tarpai tarp skyriklių ir žodžių pradžių, tarp atskirų sakinių ir kitur, kur Paskalio kalba leidžia nepalikti tarpo, tačiau bendrosios rašybos taisyklės tam prieštarauja. *Pavyzdžiui:*

Taip nereikia rašyti:	***Turėtų būti:***

var pirmas,antras,trecias:real;
begin
  Read(pirmas,antras,trecias);
  WriteLn(pirmas,antras,trecias);
end.
	var pirmas, antras, trecias: real;
	begin
	  Read (pirmas, antras, trecias);
	  WriteLn (pirmas, antras, trecias);
	end.

Programos sakiniuose vardus, žodžius ir kt. siūloma rašyti grupuojant, išlyginant pagal pozicijas. Tai leidžia greičiau ir lengviau skaityti programą, rasti joje klaidas, taisyti bei modifikuoti ją. Žmogaus akys ir pasąmonė neapkraunama papildomu darbu ieškant tekste ir išskiriant dedamąsias dalis. *Pavyzdžiui:*

```
Type
zodis=string(15);
inf=record        v1:zodis;
                  kitas:real;
                  sveikas_sk:integer;
       end;
sarasas=array[ 1..50 ]of inf;
dok=record
                  vardas:zodis;
                  kiekis:integer;
                  sar:sarasas
        end;
mas = array[ 1..50 ]  of  real;
  var pirmas:inf;antras:sarasas;
            dokumentas:dok;
            a,b,c:mas;
            n,n1,n2,n3,i,j:integer;
```

```
Type
  zodis  =  string ( 15 );
  inf     =  record
                              v1    :      zodis;
                              kitas :      real;
                        sveikas_sk:        integer;
              end;
  sarasas =  array [ 1..15 ]  of   inf;
  dok       = record   vardas  :  zodis;
                              kiekis   :  integer;
                              sar      :   sarasas;
              end;
var
   pirmas          :  inf;
   antras          :  sarasas;
  dokumentas   :  dok;
   a, b, c          :  mas;
  n1, n2, n3      :  integer;
  i, j, n           :  integer;
```

Taisyklingai rašydami prarasime šiek tiek laiko, tačiau laimėsime žymiai daugiau: taip rašydami mažiau padarysime klaidų, greičiau jas pastebėsime.

Programos teksto rašymas.

Programos rašymo stilius vertinamas tokiais pat kriterijais kaip ir natūralios kalbos stilius. Programos tekstas turi būti aiškus, vaizdus, lengvai skaitomas ir suprantamas. Tekstas lengviau suprantamas, jei jis patogiai išdėstytas lape, eilutės neperkrautos informacija, naudojami atitraukimai ir tarpai. Teksto išdėstymas turi:

- atitikti veiksmų hierarchiją ir prioritetus,
- atitikti loginę algoritmo struktūrą,
- naudoti atitraukimus sakinių pavaldumui išryškinti,
- naudoti tuščias eilutes programos dalims išskirti ar skaitomumui pagerinti:
  - –tarp pagrindinių dalių iki trijų tuščių eilučių,
  - – tarp ne pagrindinių - iki dviejų,
  - – prieš bloką - vieną.
- lygiuoti vertikaliai: *then...else* sąlygos sakinyje, *begin...end* blokuose, *repeat...until* kartojimo sakinyje, *record...end* aprašymuose.

```
(* ------------------- komentarų pašalinimas --------------*)
i  :=  1;
while   i <= n -1   do
  begin
   if (  a[ i ] = '('  ) and (  a[ i + 1 ] = '*'  )
     then   (*     radome atidarančius ženklus *)
       begin
          pr :=  i;     yra :=  false;
                 (*   ieškome uždarančių ženklų          *)
          while (  i <= n - 1  ) and  not yra    do
            begin
              if (  a[ i ] = '*'  ) and (  a[ i + 1 ] = ')'  )
                 then yra := true;
              i := i + 1
            end;
          if yra then
            begin
              g := i;   ilg := g – pr + 1;
                   (*     šalinami komentarai      *)
              for   k :=  pr   to    n – ilg    do
                    a[ k ] := a[ k + ilg ];
              n := n – ilg;
              i := i – ilg + 1;
            end
       end
    else    i := i + 1
  end;
```

Programos modulių pradžią sūloma pažymėti komentarų eilute, kurioje be skiriamosios linijos dar būtų ir modulio vardas, pvz. :

(*---------- D a u g y b a ---*)

Pabaigą taip pat reikia pažymėti, tačiau vardas nebūtinas. Jeigu toliau seka kito modulio tekstas, tai jo pradžios žymė gali būti ankstesnio modulio pabaigos požymiu. Programos pagrindinio modulio pradžią reikėtų pažymėti kitokia eilute, pvz.:

(*==========*)

Programinius modulius, esančius kitų modulių viduje reikia pažymėti taip, kad būtų akivaizdi jų priklausomybė, pvz.:

```
(*..... Raso ...*)
        (*..... Raso ...*)
(*      ==>  Raso  <==*)
```

Tuščios eilutės kaip modulių skyrikliai nėra labai akivaizdūs, tad jas tikslinga naudoti programos blokų ar kitų programos elementų atskyrimui.

Programoje naudojami vardai rašomi laikantis sutartų reikalavimų:

- programos, paprogramių vardai pradedami didžiaja raide. Jeigu vardą sudaro keli žodžiai, tuomet kiekvienas iš jų pradedamas didžiaja raide;
- tarnybiniai (baziniai) žodžiai rašomi iš mažosios raidės;
- konstantų vardai pradedami didžiaja raide;
- programuotojo aprašomł duomenų tipų vardai pradedami mažaja raide. Jeigu tipas yra sudėtingos struktūros ir vartojamas globaliai, tuomet vardas pradedamas iš didžiosios raidės;
- kintamųjų vardai pradedami rašyti:
  - *lokalinių* iš mažosios raidės;
  - *tarnybinių* iš mažosios raidės;
  - *globalinių* iš didžiosios raidės kiekvienas žodis, jeigu vardas sudurtinis;
  - *paprogramių parametrų* iš didžiosios raidės, jeigu jie yra struktūrinio tipo (visų, išskyrus standartinius paprastus duomenų tipus);
  - *pagrindinės programos* duomenų ir rezultatų iš didžiosios raidės kiekvienas žodis, jeigu vardas sudurtinis; jeigu vardas standartinio tipo, tuomet iš mažosios raidės;
  - *failo tipo* vardai iš didžiosios raidės.

Atminkime. Rašydami programos tekstą tvarkingai sugaišime daugiau laiko, tačiau:

- paliksime mažiau klaidų;
- klaidas pastebėsime greičiau;
- surasti klaidas bus lengviau;
- suderinsime programą daug kartų greičiau;
- legviau bus padaryti programoje pakeitimus, papildyti naujais veiksmais;
- didesnės galimybės keistis tarpusavyje programomis ar jų fragmentais;

- sudėtingos programos rašomos kolektyviai, – bus lengviau įsijungti į profesionalių programuotojų grupę.

Programos apiforminimas

Prieš programos tekstą reikia surašyti būtiniausią informaciją apie programos paskirtį, jos galimybes, apribojimus vartotojui ir pan. Tikslinga tai grupuoti tokia tvarka :

– programos autorius, jo adresas ir kita būtina bendravimui informacija;

– programos paskutinės redakcijos data, versijos vardas;

– programos paskirtis, taikymo sritys;

– užduoties sprendimo būdas, metodas arba trumpa anotacija;

– programiniai apribojimai;

– kita informacija (būtina autoriaus požiūriu).

```
(*     1993.06.12.      ************************************************)
(*  Petraitis Petras.   KTU  Informatikos  fak.  1-as  kursas, IF-2/5 grupė     *)
(*-------------------------------------------------------------------------------------------------*)
(*  Sutvarko duotą darbuotojų sąrašą profesijos ir darbo stažo mažėjimo tvarka          *)
(*  Duomenys tekstiniame faile, kurio vardo paprašys programa. Rezultatai bus Jūsų     *)
(*  nurodyto  vardo tekstiniame faile. Naudojamas burbuliuko būdas.                     *)
(*-------------------------------------------------------------------------------------------------*)
(*  Duomenys skaitomi į masyvą, kurio apimtis iki 1000 elementų. Duomenų failo          *)
(*  eilutės struktūra:  Pavardė, vardas, profesija, profesinės veiklos pradžios data    *)
(*****************************************************************)
```

Atsakymai į kontrolinius klausimus ir užduotis

1. tema. Atsakymai į kontrolinius klausimus

1. Programos dalys: programos antraštė, aprašai, pagrindinė dalis.
2. Visos Turbo Paskalio programos pradedamos antraštės sakiniu:*program* <Programos vardas>;
3. Kintamųjų vardai sudaromi tik iš lotyniškų raidžių ir skaitmenų, negali būti tarpų, sintaksės ženklų, simbolių. Vardas būtinai turi prasidėti lotyniška raide.
4. Komentaras rašomas, bet kurioje programos vietoje, kur galima rašyti tarpo simbolį, specialiuose skliaustuose:
(* *) arba { }.
5. Į rezultatų langą galima pereiti naudojant ekrano langų perjungimo komandą **Alt+F5**.
6. Patikrinti ir vykdyti programą galima komanda **Ctrl+F9**.
7. Tarnybiniai žodžiai: *begin* ir *end* vadinami operatoriniais skliaustais.
8. Skaičiavimai porgramoje užrašomi priskyrimo sakiniais, kurių bendra struktūra yra:
<Kintamasis> := <Aritmetinė išraiška>;
9. Visi programos pagrindinėje dalyje naudojami kintamieji, turi būti aprašyti aprašų dalyje *var* tokio tipo sakiniais:
<Vardų sąrašas> : <Tipas>;
10. Aritmetinės išraiškos sudaromos iš kintamųjų, konstantų, funkcijų sujungiant jas aritmetinių operacijų ženklais bei skliaustais.
11. Aritmetinėje išraiškoje pirma atliekami veiksmai skliaustuose, po to dalyba, daugyba, o vėliau sudėtis ir atimtis. Vienodo pirmumo veiksmai atliekami iš kairės į dešnę.
12. Veiksmų tvarką galima pakeisti skliaustais.
13. Rašymo sakinio bendra struktūra yra: *WriteLn*(Siunčiamų duomenų sąrašas);
14. Duomenų išvedimo šablonas: <Duomenų elementas>:<Bendras lauko ilgis>;
15. Skaitymo sakinio struktūra: *ReadLn*(<kintamojo vardas>);
16. Galimi du atvejai: kiekvieną kintamąjį atskirai ir visus iš karto.
17. Užklausa reikalingam, kad būtų aišku kokius kintamuosius įvesime į kompiuterio atmintį.

1. tema. Kontrolinių užduočių atsakymai

1. b, c ir f.
2. Teisinga programa:

```
program   Antroji;          {progamos pradžia}
                            (*testo užduotis*)
  var     sk  :  real;
   begin
     WriteLn('Jus sveikina kompiuteris!');
     WriteLn('Įveskite savo skaičių '); ReadLn(sk);
     WriteLn(' Jūs įvedėte: ', sk:5:0);
     WriteLn('Ačiū, kad bendravote.')
   end.
```

3. a, c ir e.

2. tema. Atsakymai į kontrolinius klausimus:

1. Valdančių struktūrų žinome du sakinius: sąlygos ir ciklo.
2. Sąlygos sakinio sintaksė: *if* <Sąlyga> *then* <V1> *else* < V2>;
3. Sąlygos sintaksė tokia: <I1> <Santykio opertacija> <I2>.
4. V1 ir V2 yra alternatyvos, aprašytos operatoriais.
5. Santykio operacijos ženklai yra: =, <>, <, >, <=, >=
6. Santykis gali įgyti logines reikšmes **True** ir **False**.
7. Jeigu santykis įgyja reikšmę **True**, tai vykdoma alternatyva V1, o jeigu - **False**, tai vykdoma alternatyva V2.
8. Sutrumpintas sąlygos operatorius atrodys taip: *if* <Sąlyga> *then* <V1>;
9. Sutrumpintas sąlygos sakinys vartojamas tada, kai tam tikrą operatorių arba jų grupę reikia vykdyti arba ne, priklausomai nuo sąlygos, t. y. alternatyva yra tarp vykdymo ir nevykdymo.
10. Ciklo sakinio sintaksė yra tokia: *while* <Sąlyga> *do* <Kartojimo sakinys arba jų grupė>;
11. Kartojimas vykdomas tol, kol sąlyga yra tenkinama, t.y. kol santykis įgyja reikšmę **True**.
12. Norint, kad ciklas nebūtų begalinis reikia žiūrėti, kad sąlyga priklausytų nuo kartojamo sakinio rezultatų.
13. Jeigu norim kartoti operatorių grupę, juos būtina rašyti operatoriniuose skliaustuose. Tokio sakinio sintaksė būtų:

```
while <sąlyga> do
      begin
      <operatorius 1>;
      <operatorius2 >;
      .....
      end;
```

14. Valdančių struktūrų sakiniai dažnai vartojami duomenų kontrolei. Jeigu tolimesniam uždavinio sprendimui tinka ne bet kokios argumentų reikšmės, tai jų įvedimą reikia organizuoti kartojimo sakinio pagalba taip, kad būtų įvesti tik tie duomenys, kurie tenkina sąlygas.

2. tema. Kontrolinių užduočių atsakymai

1. Klaidų paaiškinimai:
   a) sąlygos sakinio viduryje kabliataškis nededamas. Sakinys turėtų atrodyti taip:
   if a<b *then* s := s + a
   else s := s +b;
   b) blogas santykio operacijos ženklas. Sakinys turėtų atrodyti taip:
   if s >= 0 *then* d := Sqrt(s) *else* d := Sqr(s);
   c) vienodos alternatyvos negali būti. Sakinys pavyzdžiui galėtų atrodyti taip:
   if k<> 0 then a := k+1 else a := k - 1;
   d) neteisinga rašymo sakinio sintaksė. Tekstas turi būti rašomas apostrofuose. Ištaisytas sakinys:
   d) *if* r=0 *then* WriteLn ('Tai nėra apskritimas');
2. Bus atspausdinta: a=5.0 b= 5.0 c= 6
3. a) ciklas nebus begalinis.
   b) ciklas bus begalinis, nes kartojamas tuščias sakinys (taip vadinamas ;, parašytas po tarnybinio žodžio do). Nereikalingas ;.
   c) ciklas bus begalinis, nes kartojamas tik spausdinimo sakinys, o a reikšmė nesikeičia. Trūksta operatorinių skliaustų.
   d) ciklas nebus vykdomas nė karto.
4. Kartojimo sakinys atrodys taip:

```
while  x<10  do
    begin
        WriteLn(' Jūsų prašoma įvesti dviženklį skaičių, o jūs įvedėte:', x);
          WriteLn(' Įveskite dviženklį skaičių iš naujo');
         ReadLn(x)
    end;
WriteLn(' Ačiū, skaičius geras!');
```

3 tema. Atsakymai į kontrolinius klausimus

1. Skaičių srauto pabaigą nustato kartojimo sakinio sąlyga. Ji parenkama pagal tai, kokių skaičių sraute negali būti. Dažniausiai tai būna nulinė argumento reikšmė.
2. Kaupimo uždaviniuose dažniausiai reikia dviejų kintamųjų. Vienas įvedamai argumento reikšmei saugoti, o kitas - pačiai kaupimo reikšmei skaičiuoti.
3. Sumos kaupimo kintamajam reikia suteikti skaičiavimo pradžioje nulinę reikšmę.
4. Sudauginimo programa skiriasi nuo sumos skaičiavimo dviem operatoriais: sandaugos reikšmei suteikiama vienetinė reikšmė, o pats sandaugos skaičiavimo veiksmas bus užrašomas operatoriumi s := s * x.
5. Kiekio skaičiavimo programa skiriasi nuo sumavimo tuo, kad prie senos riekšmės yra pridedamas vienetas, o ne argumento reikšmė, t.y. s := s+1;
6. Sumavimo veiksmas aprašomas tokiu operatoriumi s := s+ <kintamasis>;
7. Šio operatoriaus prasmė yra ta, kad abiejose priskyrimo ženklo pusėse yra pavartotas tas pats kintamojo vardas, bet jo reikšmės yra nevienodos laiko atžvilgiu, t.y. nauja kintamojo reikšmė gaunama prie senosios pridėjus tam tikrą dydį.

8. Kaupimo kintamiesiems reikia suteikti tokias pradines reikšmes, nuo kurių neprikalusytų rezultatas, nes jau pirmo kartojimo metu nuo senos kintamojo reikšmės skaičiuojama nauja. Tad svarbu, kokia buvo pradinė reikšmė.

3 tema. Kontrolinių užduočių atsakymai

1. Ši programa skaičiuoja įvedamo skaičių srauto sandaugą. Įvedimas vykdomas tol, kol bus įvestas nulis.
2. Ši programa, bet kokiu atveju, duoda nulinį atsakymą, o tai yra todėl, kad pirma įvedama nauja reikšmė, po to dauginama ir tik tada tikrinama pabaigos sąlyga. Vadinasi, įvedus 0, jis bus sudaugintas su turima sandauga ir pavers ją nuliu. Pataisyta programa atrodys taip:

```
program  Nul_ats;
        var   ats, k : real;
         begin
            WriteLn(' Darbas nutraukiamas įvedus nulį');
             ats := 1;
             WriteLn( 'įveskite kintamojo reikšmę'); ReadLn(k);
               while  k<>0  do
                 begin
                    ats := ats * k
                     WriteLn( 'įveskite kintamojo reikšmę');  ReadLn(k);
                          end;
```

```
        WriteLn(‘ Atsakymas bus:’, ats : 3 : 0);
        ReadLn;
    end.
```

3. Kartojimo sakinys atrodys taip:

```
while  k<>x   do
    begin
      if  x >0   then   kt := kt + 1
            else   if  x<0   then   kn := kn + 1
                        else       n := n + 1;
      k  := x;
      WriteLn(‘įveskite sekos narį’);              ReadLn(x)
    end;
```

4 tema. Atsakymai į kontrolinius klausimus

1. Didžiausios ir mažiausios reikšmės ieškojimo algoritmai dar vadinami ekstremumų uždaviniais.
2. Didžiausios ir mažiausios reikšmės ieškojimo algoritmai skirstomi į dvigrupes pagal tai ar reikia nustatyti didžiausios (mažiausios) reikšmės eilės numerį sraute , ar ne. O taip pat pagal tai iš kokios aibės elementų ieškosime ekstremumo.
3. Kai reikia surasti tik patį ekstremumą ***var*** srityje reikia aprašyti du kintamuosius, dažniausiai baziniu žodžiu ***real***.
4. Kai reikia nustatyti kelintas buvo tas didžiausias (mažiausias) elementas sraute, ***var*** srityje reikia aprašyti tris kintamuosius: du - dažniausiai baziniu žodžiu **real**, o trečiąjį - ***integer***. nes jis reiškia eilės numerį, kuris gali būti tik sveikas skaičius.
5. Didžiausios reikšmės ieškojimo sąlygos sakinys bus: *if* d < x *then* d := x;
6. Mažiausios reikšmės ieškojimo sąlygos sakinys bus: *if* m > x *then* m := x;
7. Tam kintamajam, kuris saugos didžiausą ar mažiausią reikšmę skaičiavimų pradžioje reikia suteikti tam tikrą reikšmę. Dažniausiai tai būna pirmoji reikšmė iš srauto arba bet koks skaičius, už kurį nėra didesnių, ar mažesnių sraute.
8. Tie kintamieji, kurie programos vykdymo metu gali įgyti tik sveikų skaičių reikšmes, ***var*** srityje aprašomi baziniu žodžiu ***integer***.
9. Nevisuomet ekstremumo saugojimo kintamajam galima suteikti pirmą iš srauto reikšmę, nes jeigu tą srautą reikia peržiūrėti ne visą, o tam tikrą poaibį, tai jame gali nebebūti didesnių ar mažesnių reikšmių. Pvz.: jeigu ekstremumą reikia išrinkti tik iš tam tikrą sąlygą patenkinančių srauto reikšmių, o pirmasis srauto elementas gali tos sąlygos netenkinti.

4 tema. Kontrolinių užduočių atsakymai

1. Sakinys nevisuomet teisingas. Jis blogas tuo atveju, jeigu pirmų dviejų argumentų reikšmės tarpusavyje lygios. Pavyzdžiui: a := 2, b := 2, c := 3.

2. Ištaisytas jis atrodys taip:

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

3. Atsakymas ne visuomet bus teisingas. Jis bus teisingas tik tuomet, kai pirmoji srauto reikšmė bus neigiama. Kitais atvejais joks didžiausias neigamas skaičius nebus didesnis už teigiamą.

4. Pataisyta programa:

```
program   Klaida;
    var     did, x : real;
begin
      did := -1E + 06;
      WriteLn (‘Įveskite x’);      ReadLn(x);
        while (x<>0) do          begin
            if x < 0 then if x > did then did := x;
            WriteLn (‘Įveskite x’);  ReadLn(x);
      end;
      Writeln(‘Didžiausia neigiama reikšmė:’, did :5:2);
end.
```

5 tema. Atsakymai į kontrolinius klausimus

1. Apdorojant duomenų srautus, kiekvienai skaitomai srauto reikšmei yra taikomi du veiksmiai: filtravimas ir apdorojimas.
2. Filtruojant yra nustatoma, ar reikšmė priklauso tiriamai duomenų grupei, o skaičiuojant (apdorojant) yra nustatomos įvairios tiriamos grupės charakteristikos.
3. Paprastos filtravimo sąlygos yra aprašomos operatoriumi *if* ir santykiais.

4. Loginės operacijos gali būti:

not (neigimas),
or (veiksmas arba),
and (veiksmas ir),
xor (griežtas arba).

5. Logoniai kintamieji aprašomi baziniu žodžiu boolean.

6. Pirmiausia atliekamas neigimas, po to operacija **and** (ir) ir pabaigoje **or** (arba). Natūralią veiksmų tvarką galima pakeisti skliaustais.

5 tema. Kontrolinių užduočių atsakymai

1. Sakinio dalis: ***if (x<0)*** then - atlieka srauto filtravimą, o sakinio dalis: ***if (x > did) then did := x;*** atlieka apdorojimą.

2. Sutrumpintas sąlygos sakinys atrodys taip:

```
if (a<=b) and (a<= c) then          min := a
          else      if  (a>=b) and (b<c)    then  min := b
                                          else  min := c;
```

3. Šis sakinys reikalingas norint nustatyti ar trikampis yra lygiakraštis. Sakinys teisingas, bet viena sąlyga yra perteklinė. Suprastintass sakinys atrodytų taip:

lk := (a=b) *and* (b=c);

6 tema. Atsakymai į kontrolinius klausimus

1. Norint failinį kintamąjį susieti su OS failu naudojama procedūra *Assign.*

2. Procedūra *Reset* atidaro failą skaitymui, o *Rewrite* - rašymui.

3. Failo kintamąjį būtina.aprašyti taip: <Vardas>:<Failo tipas>

4. Norint nuskaityti duomenis iš failo, reikia juos pirmiausiai tinkamai paruošti. Po to, naudojant procedūrą *Reset,* paruošti failą skaitymui ir procedūrų *Read* arba *ReadLn* pagalba jau galima skaityti.

5. Failų uždarymui naudojama procedūra *Close*.

6. Procedūra *Read* skaito iš failo tik tiek reikšmių, kiek yra kintamųjų sąraše, o procedūra *ReadLn* skaito visus duomenis iki eilutės pabaigos simbolio, o tik po to skirsto konkretiems kintamiesiems ir, jeigu reikšmių yra per daug, jos atmetamos.

7. Procedūra *WriteLn* skiriasi nuo *Write* tuo, kad ji, rašydama duomenis į failą, papildo jį eilučių pabaigos simboliais.

8. Loginėmis funkcijomis EoLn ir Eof , skaitant duomenis iš failo, galima tikrinti atitinkamai eilučių ir failo pabaigos žymes

9. Duomenis, skaitomus iš failo, reikia paruošti iš anksto su tekstiniu redaktoriumi. Rašant skaičius, juos reikia atskirti tarpais. Įrašyti duomenų failą reikia į tą patį katalogą, kur yra ir jį apdorojanti programa, arba procedūroje *Assign* reikia nurodyti kelią iki to failo.

6 tema. Kontrolinių užduočių atsakymai.

1. Programa nėra teisinga, nes į tą patį failą duomenys rašomi ir iš jo skaitomi. Pakanka panaikinti eilutę *ReadLn(f, sum);* ir programa bus teisinga.

2. Bus atspausdinta: a=-3 b=2 c=5 d=-4, nes procedūraReadLn perskaičiusi pirmą eilutę priskiria reikšmes pirmiems trims kintamiesiems: a, b, c, o perskaičiusi antrą eilutę, priskiria reikšmę ketvirtam kintamąjam, o kitos dvi yra perteklinės.

3.Tada bus atspausdinta: a= -2 b= 1 c=0 d=0.

4.

```
program      Neig_skai;
  var  x     : integer;
             f, f1 : text;
  begin
    Assign(f, 'd.dat');        Assign(f1, 'd1.dat);
    Reset(f);                        Rewrite(f1);
    while not Eof do
       begin
            Read(f, x);
            if  x<0   then       Write(f1,x);
        end;
     Close(f, f1);
  end.
```

7 tema. Atsakymai į kontrolinius klausimus

1. Atsitiktiniai dydžiai dažniausiai naudojami kompiuteriu modeliuojant žaidimus ir kai skaičiavimams reikalingi didesni skaičių masyvai.

2. Funkcijos *Random* pagalba galima gauti atsitiktinius skaičius.

3. Programuojant žaidimus, svarbu iš anksto gerai apgalvoti ir aprašyti programos scenarijų (algoritmą).

4. Funkcija *Random* gali neturi parametrų, o jos generuojami skaičiai tolygiai pasiskirsto intervale.

5. Didesnes skaičių sekas patogų pakeisti atsitiktiniais dydžiais, nes nesugaištama tiek daug laiko jų įvedimui iš klaviatūros, todėl galima daugiau kartų ir su įvairesnėmis reikšmėmis atlikti skaičiavimus. Lengviau tikrinti programos teisingumą.

6. Norint, kad nenukentėtų skaičiavimų tikslumas, reikia daugiau kartų skaičiuoti ir lyginti rezultatus.

7 tema. Kontrolinių užduočių atsakymai.

1.Priskyrimo sakinys įrašytas į kartojimo sakinį atrodytų taip: a := 5 - Random(10);

2. Rezultatų sutapimas priklauso nuo įvedamų skaičių.

- Jeigu pirmas įvedamas skaičius nebus nulis, o tarp įvedamų skaičių bus bent vienas 1, tai minimalus skaičius beveik visada sutaps ir bus lygus 1.
- Jeigu didžiausias įvedamas skaičius bus 10, tai atsakymai niekada nesutaps, nes apvalinant sugeneruotus skaičius niekada negausim 10.
- Jei didžiausias įvedamas skaičius bus 9, tada atsakymai dažniausiai turėtų sutapti.

8 tema. Atsakymai į kontrolinius klausimus

1. Ekrano spalvų valdymo biblioteka naudojimui prijungiama užrašant programos pradžioje sakinį: **uses Crt;**

2. Programos valdymo ir jos pateikiamų duomenų vaizdavimo priemonės bei būdai yra vadinami programos išoriniu interfeisu.

3. Norint gauti išsamesnių žinių, apie kurią nors Crt bibliotekos struktūrą, pakanka TP redaktoriaus lange surinkti jos pavadinimą, prie jo perkelti kursorių ir paspausti klavišų porą Ctrl +F1.

4. Ekrane koordinačių pradžia yra kairiajame viršutiniame kampe ir to taško koordinatės yra (1, 1).

5. Įprastinėje koordinačių sistemoje Y ašis yra nukreipta į viršų, o ekrano ordinačių ašis yra nukreipta žemyn.

6. Užrašom procedūra TextBackground(<spalva>), o po to ClrScr, kad pasikeistų spalva.

7. Standartinis ekrano dydis pagal koordinates yra 80x25

8. Ekrano spalvoms gali būti tik pirmos aštuonios spalvos 0..7 ir tik tos, kurias palaiko ekrano valdymo įtaisas, esanti kompiuterio viduje.

9. Ekrano spalvos nurodomos sveiku skaičiumi arba standartiniu žodžiu.

10. Norint gauti pulsuojančią spalvą prie spalvos kodo reikia pridėti žodelį *blink*.

11. Norint grąžinti ekranui standartinę spalvą, programos pradžioje reikia kintamojo TextAttr pagalba įsiminti spalvos kodą, o po to jį vėl suteikti kintamajam TextAttr.

8 tema. Kontrolinių užduočių atsakymai.

1. Ši programa neteisinga, nes po komandos *TextBackground* nėra ekrano valymo procedūros ir fono spalva nesikeis. Taip pat ne visuomet bus matomas užrašas *fonas: spalvos kodas*, nes kartojimo metu teksto ir fono spalvos pirmam W*riteLn* bus vienodos.

2. Ji turėtų skirtingo fono ekrane, kitoka spalva spausdinti fono ir raidžių spalvų kodus ir po kiekvieno prašyti paspausti ENTER.

3.

```
program   Spalvos;
     uses Crt;
      var   j, ek : integer;
       begin
           ek := TextAttr;
           ClrScr;
           j := 1;
           while   j <= 7   do            begin
                TextBackground(j);                ClrScr;
                TextColor(j+1);
                WriteLn('fonas:', j, ',', ' spalva: ', j+1);
                j := j+1;
                  WriteLn(' Kai pasižiūrėsite, paspauskite ENTER');
                   ReadLn;
             end;
           TextAttr :=ek;
            ClrScr;
       end.
```

9 tema. Atsakymai į kontrolinius klausimus

1. Sudėtiniu sakiniu vadiname tokį sakinį, kuriame pavartoti operatoriniai skliaustai.
2. Žinomo kartojimų skaičiaus sakinys atrodys taip: *for* <p> := <i1> *to* <i2> *do* <kartojamas sakinys>;
3. Ciklo parametro reikšmės gali būti tik sveiko tipo.
4. Jeigu i1<i2, tai kartojimo sakinyje vartojame.bazinį žodį *to,* o jeigu i1 >i2 tai vartojame *downto*.
5. Simbolinio tipo kintamieji yra tokie, kurie programos darbo metu įgyja simbolių reikšmes.
6. Simbolio kodą galime sužinoti panaudoję funkciją Ord.
7. Simbolį galima parašyti tiesiogiai kabutėse arba naudojant funkciją Chr.

8. Funkcija Chr pagal kodo reikšę duoda simbolį.
9. Simboliai lygimami pagal kodų reikšmes.
10. Pavyzdžiui :

9 tema. Kontrolinių užduočių atsakymai.

1. Kartojimo sakinys turėtų spausdinti pranešimą 5 kartus, nes jame ciklo parametro reikšmė gali keistis tik kas vienetą, todėl priskyrimo sakinys: i := i + 2; čia yra nereikalingas ir netgi klaidingas, todėl kartais kompiuteris gali ir visai atsakymo neduoti. Tad šis sakinys turėtų būti užrašytas taip:

```
for  i := 1  to  5  do
    WriteLn('spausdinu ', i, ' -tąjį kartą);
```

2. Norint, kad programa spausdintų tik lygines i reikšmes, reikia vartoti kartojimo sakinį while. Jis užsirašytų taip:

```
i := 2;
while  I <= 5  do begin
      WriteLn('spausdinu ', i, ' -tąjį kartą);
      i := i + 2;
      end;
```

3. Spausdins taip: Intervale [a, b] sveikų skaičių suma = 40, nes kartojimo sakinys for atliekamas bent vieną kartą nepriklausomai nuo a ir b reikšmių. Todėl x bus priskirta a reikšmė, t.y. 40 ir prisumuota.
Kartojimo sakinį reikia pakeisti taip: *for* x := a *to* b *do*

4. Programa, įvedus komentarus bus:

```
program  Kodai;
 uses Crt;
 var   x : char;
       i, n : integer;
begin
      ClrScr;
      Randomize;
      for  i := 1  to  10  do
        begin
n := Random(150);
x := Chr(n);
WriteLn(' Sugalvojau simbolio kodą: ', n, ', jį atitinka simbolis: ', x);
       end
end.
```

Priskyrimo sakinį reiktų pakeisti taip: n := Random(150) + 30;

10 tema. Atsakymai į kontrolinius klausimus

1. Modulyje Crt yra trys garso valdymo procedūros: Sound, Delay, NoSound.
2. Ciklo aprašymui vartojamas operatorius *while* tuo atveju, jeigu kartojimo veiksmus reikia atlikti bent vieną kartą, o tik po to paaiškėja ar jie bus kartojami toliau.
3. Ciklo sakinyje *while* pabaigos sąlyga tikrinama pradžioje ir todėl kartojimai gali būti neatlikti nė karto, o *repeat* atlieka kartojimus bent vieną kartą. Taip pat sakinyje *while* kartojimai vykdomi tol, kol pabaigos sąlyga įgyja reikšmę *true*, o sakinyje *repeat - false.*
4. Ciko viduje skaičiavmus reikia organizuoti taip, kad jų rezultai, kada nors kartojimo pabaigos sąlygai suteiktų reikšmę false, o kitaip ciklas nesibaigs.

10 tema. Kontrolinių užduočių atsakymai.

1. Programa neteisinga, nes garso valdymui, kaip ir spalvų reikia naudoti modulį Crt.

```
program  Melodija;
{garsas}
  uses Crt;
  begin
    Sound(500);
    Delay(2000);
    NoSound;
  end.
```

2. Įvykdę programa, 50 Hz dažnio garsinį signalą girdėsime 2 sekundes.

```
program  Melodija;
{garsas}
  uses Crt;
```

```
begin
  for   j := 1   to   5   do
    begin
     Sound(500);
     Delay(2000);
     NoSound;
    Delay(2000);
     end;
end.
```

3. Kartojimo nebus. Cikle esantys operatoriai bus atlikti vieną kartą, nes 6<10, o *repeat* kartojamas, kol sąlygą įgyja reikšmę *false*.

4. Tokiu atveju ciklas bus begalinis, nes skaičiai 15, 18, 21 ir t.t. bus visada >10. Pabaigos sąlyga visą laiką įgys reikšmę *false* ir kartojimas vyks be pabaigos.

5. Priskyrimo ir kartojimo sakinys bus toks:

```
x := 3;              sum := 0;
repeat
   sum :=  sum+x;
   x := x+3;
until (x > 20);
```

6. Programa bus trumpesnė naudojant kartojimo sakinį *repeat*, nes programą sudaro 14 eilučių, o su *while* - 16, esant tam pačiam sakinių išdėstymui.

```
program   Nuliai_sekoje;
                                          {ciklas repeat}
var             k : integer;
          x  : real;
begin
     k := 0;
     WriteLn('Kai norėsite baigti, įveskite neigiamą skaičių'); WriteLn;
     WriteLn('Įvedinėkite sekos narius:');
     repeat
          ReadLn(x);
          if   x = 0   then   k := k + 1;
     until x < 0;
     if   k = 0          then        WriteLn(' Jūs neįvedėte nė vieno nulio')
                         else  WriteLn('Jūs įvedėte ', k, 'nulių');
end.

     program    Nuliai_sekoje;
                                          {ciklas while}
     var k : integer;
  x : real;
begin
     k := 0;
     WriteLn('Kai norėsite baigti, įveskite neigiamą skaičių'); WriteLn;
     WriteLn('Įvedinėkite sekos narius:');
     ReadLn(x)
     while   x > 0   do
       begin
       if   x = 0   then   k := k + 1;
       ReadLn(x);
     end;
     if   k = 0          then   WriteLn(' Jūs neįvedėte nė vieno nulio')
                         else   WriteLn('Jūs įvedėte ', k, ' nulių');
end.
```