KAUNO KOLEGIJA

DALĖ LUKŠAITĖ

PROGRAMAVIMAS C++ KALBA

2001

PRATARMĖ

Dauguma programavimo knygų, kurias randame mūsų knygynuose parašytos angliškai arba rusiškai. Programuotojams skirtose knygose taip pat gana dažnai apsiribojama formalia programavimo kalbos konstrukcijų analize ir rečiau nagrinėjami praktiniai programų taikymo klausimai. Autorės nuomone, mokantis programavimo kalbos, labai svarbu ugdyti praktinius programų sudarymo įgūdžius. Šioje mokomojoje knygoje C++ kalba pasirinkta todėl, kad tai viena populiariausių šiuolaikinių programavimo kalbų, nors kol kas dar labai mažai leidžiama lietuviškų C++ kalbos vadovėlių. Šioje mokymosi knygoje nagrinėjami algoritmų sudarymo principai, pagrindinės C++ kalbos struktūros, programų projektavimo priemonės, bei paprasčiausios ryšio su naudotojais struktūros. Pagrindinis dėmesys koncentruojamas nuosekliam pagrindinių C++ kalbos struktūrų įsisavinimui.

Mokomoji knyga parengta Kauno kolegijoje skaitomų programavimo paskaitų pagrindu. Kiekvieno skyriaus pradžioje pateikiami mokymosi tikslai, o pabaigoje mokymosi rezultatų apžvalga. Be to, knygoje pateikiami programų pavyzdžiai, kartojimo klausimai ir užduotys, kurie skatina skaitytoją aktyviai studijuoti pateiktą medžiagą. Kartojimo klausimai ir užduotys diferencijuojami pagal lygius. Knygos pabaigoje pateikiami atsakymai į kartojimo klausimus ir dalykinė rodyklė. Todėl ši mokomoji knyga sukuria prielaidas savarankiškoms studijoms. Mokomoji knyga skiriama kompiuterių tinklų administravimo, automatizuoto valdymo specialybių dieninio ir neakivaizdinio skyrių studentams. Tačiau ja gali naudotis ir kitų specialybių studentai, besidomintys programavimu.

Mokomąja medžiaga siekiama:

- suteikti programavimo žinių ir įgūdžių sudarant programas C++ kalba:
- supažindinti su programavimo integruotos sistemos Borland C++ aplinka;
- supažindinti su programavimo ir algoritmo sudarymo principais;
- supažindinti su pagrindinėmis programavimo kalbos C++ komandomis;

Darbui su paketu bus reikalinga integruota aplinka Borland C++.

Mokomojoje knygoje imtos šios simbolių reikšmės:

➢ Tikslai
Tai svarbu
Klausimai
Užduotys
Pratimas
Informacijos šaltiniai
Santrauka

TURINYS

1. ĮVADAS

Šiame įvadiniame skyriuje skaitytojas susipažįsta su programavimo pagrindinėmis sąvokomis ir terminais, duomenų tipų savybėmis ir jų klasifikavimu C++ kalboje, programavimo kalbų raida ir integruotų programavimo aplinkų sudėtinėmis dalimis. Ypatingas dėmesys skiriamas programų projektavimo principams ir eigai bei uždavinių sprendimo algoritmų vaizdavimo būdams. Be to šiame skyriuje aiškinami programų testų sudarymo principai ir pateikiami reikalavimai programų bei jų modulių aprašymams.

➢ Mokymosi tikslai

Baigęs šį skyrių skaitytojas turi:

- Suprasti informacijos, duomenų, algoritmo, programos sąvokas;
- Žinoti duomenų tipų savybes ir klases;
- Žinoti programavimo kalbų raidą;
- Suprasti programavimo aplinkų sudėtinių dalių paskirtį;
- Žinoti programos projektavimo tvarką;
- Mokėti sudaryti programos struktūrinę ir modulinę schemas;
- Mokėti sudaryti uždavinio sprendimo algoritmo struktūrogramą;
- Mokėti sudaryti programos testus;
- Mokėti aprašyti programas ir jų modulius.

1.1 Pagrindinės sąvokos ir terminai

Jūs tikriausiai jau žinote, kad informacija vadinamos žinios, reiškiamos bendra, dažniausiai žodine forma. Daugelis mokslininkų pateikia įvairius **informacijos** apibrėžimus:

- *Žinios, perduodamos vienų asmenų kitiems.*
- *Žodžiu, raštu ar ryšio priemonėmis perduodamos žinios.*
- *Neapibrėžtumo skaitmeninis matas.*

Mes, žmonės, turime išskirtinę savybę suvokti informaciją ir mąstymu sukurti naujas žinias. Be to informacija naudojamės, kaupiame, perduodame vieni kitiems. Žinių perdavimas reiškia, kad jas priimantysis sužino kažką naujo. Informacija yra nežinojimo, neapibrėžtumo priešybė. Kasdieniniame gyvenime informacijos kiekį nustatome intuityviai. Todėl tikslus informacijos sąvokos apibrėžimas nereikalingas. Tačiau informatika yra mokslas, susijęs su kompiuterių naudojimu. Taigi informacijos apibrėžimas turi atitikti uždavinius, sprendžiamus kompiuteriu.

Labai dažnai terminai *informacija* ir *duomenys* naudojami kaip sinonimai, tačiau jie skiriasi savo semantine prasme tiek ir esme.

Informacija *- tai gaunamos žinios ir jų tarpusavio ryšių visuma, duomenų turinys.*

Duomenimis *vadinamos konkretesnės žinios, kurios pateikiamos skaičiais arba turi matematizuotą formą.*

Šiuolaikinių informacijos apdorojimo metodų ir būdų sistema, leidžianti paversti informaciją duomenimis, ieškoti, rinkti, kaupti, saugoti, keisti, apdoroti tuos duomenis ir vėl pateikti kaip naują informaciją, vadinama *informacijos technologijomis*.

Suprantama, kad apdorojant informaciją, būtina turėti papildomus duomenis, kurie rodo kaip tai atlikti. Dažniausiai tokia informacija tai yra duomenys apie duomenis vadinami algoritmais ir programomis, arba metainformacija.

Veiksmų ir taisyklių visuma, parodanti kaip iš pradinių duomenų gauti teisingą rezultatą, vadinama ***algoritmu****.*

Algoritmai, užrašyti programavimo kalbomis, vadinami ***kompiuterių programomis****.*

Kompiuteris arba kompiuterinė technika yra svarbiausia informacjos technologijų techninės įrangos dalis. Kompiuteris atlieka veiksmus su duomenimis, t.y. programas.

Duomenys, pateikiami kompiuteriui, vadinami **pradiniais duomenimis**, o iš jų gaunami rezultatai, vadinami **galutiniais duomenimis**. Apdorojimo metu taip pat gaunami įvairūs duomenys.

Programoje naudojami duomenys - tai įvardinti dydžiai, kurių reikšmės saugomos kompiuterio atmintyje. Duomenis sudaro: vardas ir reikšmė. Veiksmai su duomenimis rašomi programoje naudojantis jų vardais. Programoje naudojami dviejopi duomenys – konstantos ir kintamieji.

Konstantos – duomenys, kurių reikšmės išlieka pastovios.

Kintamieji – įvardinti dydžiai, kuriais operuojama programose. Dažnai kintamųjų reikšmės iš anksto nežinomos ir, atliekant programą, kinta.

1.2 Duomenų tipai

Programavime vartojami labai įvairūs duomenys. Pagal savybes jie skirstomi į klases, vadinamais **duomenų tipais**. Pavyzdžiui:

- Sveikieji skaičiai priklauso *sveikųjų skaičių* tipui;
- Duomenys, turintys dvi logines reikšmes – tiesa ir netiesa (true ir false), priklauso *loginiam* tipui;
- ir kiti.

Duomenų tipas rodo kiek reikia skirti kompiuterio atminties tokio tipo kintamojo reikšmei saugoti. Be to tipas rodo kokius veiksmus galima atlikti su šio tipo kintamųjų reikšmėmis.

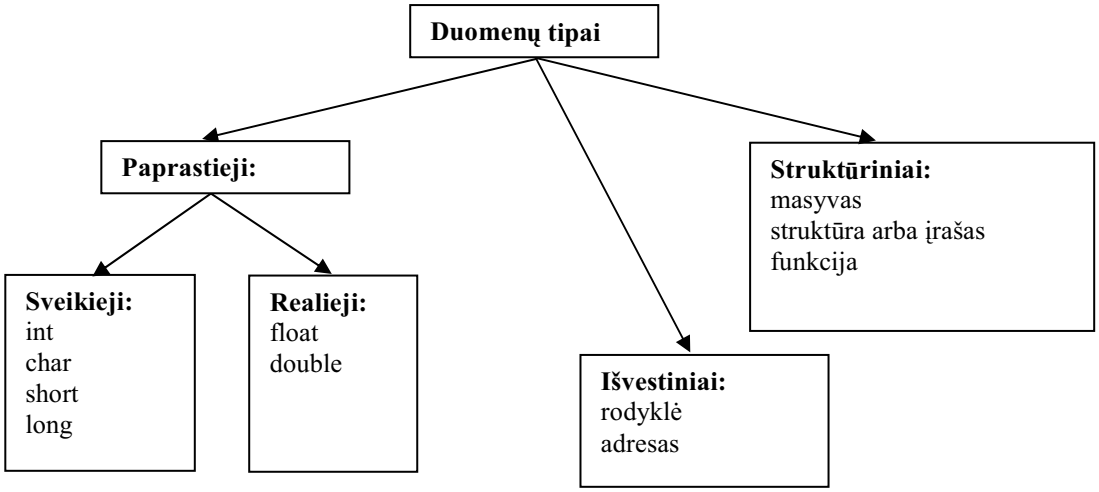
Griežtos duomenų tipo apibrėžties nėra. Tačiau bendrąsias duomenų tipų savybes, būdingas daugeliui aukštojo lygio programavimo kalbų, suformulavo Horas (C.A.R.Hoare, 1975) :

1. Tipas apibrėžia reikšmių, kurias gali įgyti kintamasis ar reiškinys, klasę.
2. Kiekviena reikšmė priklauso vienam tipui.
3. Konstantos, kintamojo arba reiškinio tipą galima nustatyti iš konteksto, arba iš paties operacijos operando pavidalo, nepriklausomai nuo reikšmių, gautų atliekant programą.
4. Kiekvienos operacijos operandų ir rezultato tipai yra apibrėžti.
5. Informacija apie tipus aukštojo lygio kalboje leidžia išvengti klaidų programoje ar jas rasti ir nustatyti, kaip duomenys vaizduojami kompiuteryje ir kokie su jais atliekami veiksmai.

Duomenų tipai bei duomenų struktūrinimo būdai pradėti naudoti kartu su aukštojo lygio programavimo kalbomis, kurios aptariamos sekančiame skirsnyje.

Pagal reikšmių struktūrinimo laipsnį duomenų tipai skirtomi į tris klases (žr. 1.1 pav.):

1. *Paprastuosius* – loginiai, simboliniai, sveikieji, realieji, vardiniai, atkarpos. *Paprastųjų* tipų reikšmės yra nedalomos – tai skaičius arba simbolis.
2. *Struktūrinius* – masyvas, įrašas arba struktūra, failo rekursyvios struktūros. *Struktūrinių* tipų reikšmės sudaromos iš kitų paprastųjų ar struktūrinių reikšmių.
3. *Išvestinius* – rodyklės, adresas. *Išvestiniai* duomenų tipai apibrėžiami specialiomis sintaksinėmis struktūromis, kurios leidžia paslėpti fizinį duomenų vaizdavimo ir apdorojimo lygį.



1.1 pav. Duomenų tipai C++ kalboje

Apibendrinant galima pasakyti:

Duomenų tipas apibrėžiamas to tipo duomenų reikšmių aibe ir operacijomis, atliekamoms su tomis reikšmėmis.

🕮 Kontroliniai klausimai

1. Paaiškinkite skirtumą tarp informacijos ir duomenų.
2. Ką vadiname informacinėmis technologijomis?
3. Paaiškinkite algoritmo sąvoką.
4. Ką vadiname kompiuterine programa?
5. Paaiškinkite skirtumą tarp kintamojo ir konstantos.
6. Paaiškinkite duomenų tipo sąvoką.
7. Paaiškinkite C++ kalbos duomenų tipologiją.

1.3 Programavimo kalbos ir aplinkos

Jums puikiai žinoma, kad visas kompiuterių programas galima suskirstyti į tris pagrindines grupes:

- *taikomąsias* programas, skirtas įvairių profesijų atstovų praktiniams uždaviniams spręsti;
- *sistemines* programas, garantuojančias kompiuterio ir taikomosios programinės įrangos darbą;
- *programavimo aplinkas*, leidžiančias kurti įvairias programas kompiuteriams.

Kompiuterių programos kuriamos naudojant programavimo kalbas. Pirmųjų programavimo kalbų komandos vadinamos "mašininiu kodu", nes kompiuteris suprasdavo jas tiesiogiai, o pačios kalbos *žemojo lygio* kalbomis.

Programuotojai, kurdami įvairias sistemines programas, ilgą laiką naudojosi žemojo lygio programavimo kalbomis. Populiariausia programavimo kalba buvo *Assemblerio* kalba. Tačiau programavimas žemojo lygio kalbomis, tokia kaip *Assembleris,* kurios tiesiogiai aprašo kompiuterio valdomus fizinius procesus, reikalauja aukštos programuotojo kvalifikacijos, nes žemojo lygio programavimo kalbų komandos panašios į kompiuterio komandų sistemą. Todėl tokias programas yra sunku suprasti, kadangi net kelios eilutės gali būti skirtos tik vienam simboliui spausdinti. Be to žemojo lygio kalbos yra suderintos su tam tikro mikroprocesoriaus tipu.

Todėl vėliau buvo sukurtos aukštesniojo lygio programavimo kalbos, kurios sėkmingai naudojamos tiek sisteminėms, tiek ir taikomosioms programoms kurti.

Pirmoji tokios paskirties programavimo kalba buvo kalba C, kurią 1978 m sukūrė B.W.Karnighan ir D.M.Ritchie, aprašę ją knygoje "C programavimo kalba (The C Programming Language). Nuo 1983 m. kalbos C programavimo priemonės reglamentuojamos ANSI (American National Standarts Institute) standartais. Naują šios programavimo kalbos variantą, vadinamą C kalba su klasėmis, 1980 m. sukūrė B.Straustrupas, kuriam suteiktas C++ pavadinimas. C++ nauja programavimo kalba, skirta sudėtingų programų sistemų kūrimui, panaudojant objektinio programavimo technologiją.

Jums jau žinoma, kad kompiuterių programa iš pradžių rašoma pasirinkta programavimo kalba, kuria aprašomi informaciniai procesai. Po to programos tekstas įvedamas į kompiuterio atmintį ir pervedamas į vidinę kompiuterio kalbą. Tai atlieka specialios pagalbinių programų sistemos, kurios vadinamos integruotomis **programavimo aplinkomis**.

Viena iš populiariausių šiuolaikinių programavimo aplinkų, tai Borland C++ operacinėms sistemoms MS DOS, Windows, Windows NT.

Integruotą Borland C++ programavimo aplinką sudaro šių programų rinkinys:

1. *Pirminis procesorius. C++* nuo kitų programavimo kalbų skiriasi tuo, kad prieš kompiliavimą programos tekstą apdoroja pirminis procesorius. Jei pirminiame programos tekste yra instrukcijos pirminiam procesoriui, tai jos vykdomos prieš kompiliavimą.
2. *Tekstų redaktorius*. Programos tekstas į kompiuterio atmintį įvedamas tekstų redaktoriumi. Kompiuterio atmintyje užrašytas programos tekstas vadinamas pirminiu programos moduliu (.cpp).

3. *Kompiliatorius*. Kompiliatorius perveda programos pirminį tekstą, parašytą aukštojo lygio programavimo kalba, į kompiuterio komandų sistemą. Kompiliavimo metu sukuriamas vykdomasis programos failas (*.exe*). Tai sukompiliuota ir paruošta vykdymui programa. Kompiliatoriaus rastos klaidos parodomos pastabų lange.
4. *Sisteminių ryšių redaktorius*. Sukompiliuota programa gali būti komponuojama, tai yra į vieną vykdomąjį failą yra komponuojami keli, iš anksto sukompiliuoti failai. Ryšių redaktorius sujungia atskiras programos dalis į vientisą programą.
5. *Derinimo ir analizės priemonės* leidžia atlikti eksperimentinį programos patikrinimą: kontrolinių duomenų patikrinimą ir formavimą; programos stabdymą kontroliniuose taškuose; programos vykdymą atskiromis komandomis (trasavimą); kontrolinių duomenų reikšmių keitimą programos darbo metu.
6. *Funkcijų ir paprogramių bibliotekos.* Funkcijos ir paprogramės pateikiamos kartu su programavimo aplinka skirtinguose rinkiniuose, vadinamose bibliotekose. Naudojantis tokiomis funkcijomis ir paprogramėmis pakanka žinoti jų vardą, paskirtį ir kreipinio struktūrą.

Šioje mokomojoje knygoje analizuojamos C++ kalbos struktūros ir programų projektavimo priemonės. Integruota Borland C++ aplinka, leidžia rengti įvairių tipų projektus.

Projektas - tai visų failų, kurie sudaro programos vykdomąjį failą, komplektas.
Integruotoje Borland C++ aplinkoje projektą sudaro šie failai:

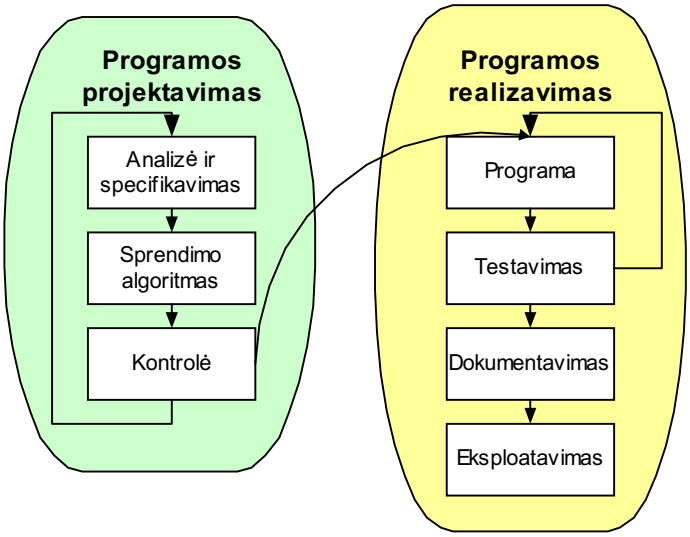
projekto failas	.ide
pirminiai programų moduliai	.cpp
išteklių failai	.rc
apibrėžčių failai	.def
bibliotekų sąsajų failai	.h
vykdomieji failai	.exe

Sekančiame skirsnyje aptariama programos kūrimo eiga.

1.4 Programos kūrimas

Mokantis programuoti, iš pradžių rašomos nedidelės programos, kuriomis aiškinamasi programavimo kalbos komandos ir duomenų struktūros bei nesudėtingi algoritmai. Tik išsiaiškinus pagrindines programavimo kalbos konstrukcijas ir įgyjus programavimo patirtį, galima pereiti prie sudėtingesnių programų projektavimo.

Dažniausiai programavimo literatūroje skiriami du labai svarbūs programų kūrimo etapai: projektavimas ir realizavimas. Visi būtini programos kūrimo ciklo žingsniai rodomi 1.2 paveiksle.



1.2 pav. Programos kūrimo ciklas

1) Programos projektavimas

a) Uždavinio analizė ir specifikavimas. Analizuojamas uždavinys ir programos pagrindinės charakteristikos. Nustatomi pradiniai programos duomenys ir darbo rezultatai bei sudaroma programos specifikacija – t.y. surašomi išsamūs programai keliami reikalavimai.

b) Algoritmo sudarymas. Uždavinys skaidomas į dalinius uždavinius, išskiriant bendruosius programos modulius. Po to paruošiama programos struktūrinė schema ir programos modulinė schema, bei sudaromi kiekvieno modulio sprendimo algoritmai.

c) Uždavinio algoritmo kontrolė. Atliekama uždavinio sprendimo teisingumo kontrolė.

2) Programos realizavimas

a) Programos kodavimas. Kiekvienas programos modulis užrašomas (koduojamas) pasirinkta aukštojo lygio programavimo kalba.

b) Programos testavimas. Naudojantis įvairiais pradinių duomenų rinkiniais aptinkamos padarytos klaidos, o derinimo metu jos ištaisomos.

c) Programos dokumentavimas. Pagrindinė programos dokumentų dalis sudaroma uždavinio specifikavimo ir programų projektavimo bei kodavimo metu.

d) Programos eksploatavimas. Naudotojams nuolat pateikiamos instrukcijos apie išryškėjusias programų klaidas, jų ištaisymą ar padarytus pakeitimus.

Sekančiuose šios mokomosios knygos skirsniuose plačiau supažindinama tik su keletu labai svarbių programos kūrimo žingsnių – programos projektavimu, programos testavimu, ir dokumentavimu.

Programos projektavimas

Siūloma susipažinti ir praktiškai nudotis šia programos projektavimo tvarka:

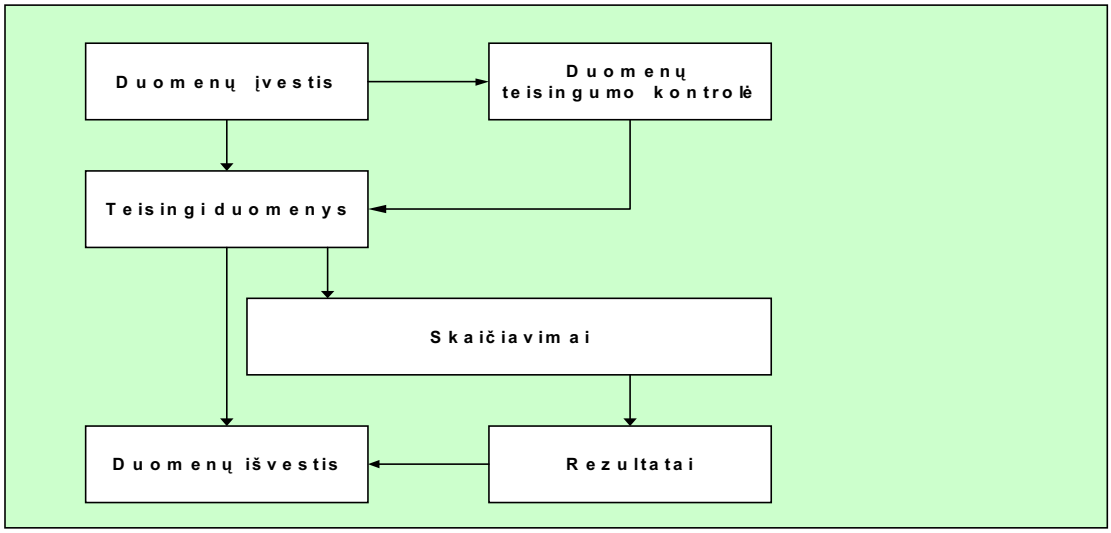
Uždavinio analizė ir specifikavimas.

Kruopščiai parenkamos duomenų struktūros. Tai svarbus algoritmo sudarymo etapas, nes kiekvienam duomenų tipui taikomas tik jam skirtas operacijų rinkinys. Nuo duomenų saugojimo, įvesties ir išvesties būdo priklauso programos struktūra.

Algoritmo sudarymas.

Sprendžiamą uždavinį suskaidžius į dalinius uždavinius ir išskyrus bendruosius programos modulius, paruošiama *programos struktūrinė schema*, kurios pavyzdys pateikiamas 1.3 paveiksle. Parenkamos programavimo priemonės. Dažniausiai sudėtingos programos turi struktūrą, kurią sudaro atskiri moduliai. Duomenų įvesties ir išvesties priemonės atskiriamos nuo skaičiavimų ir pertvarkomos į savarankiškus modulius. Numatomos duomenų teisingumo kontrolės priemonės. Patariama duomenų kontrolę atskirti nuo skaičiavimo proceso ir sudaryti atskirą modulį arba programą. Taip pat patariama apriboti duomenų galimų taisymų skaičių. Pats paprasčiausias duomenų kontrolės būdas – patikrinti ar duomenų elementas priklauso galimų reikšmių aibei. Skiriama tokie duomenų kontrolės tipai:

- *Abėcėlinė,* kai tikrinami ar visi simboliai leistini.
- *Leksinė*, tikrinama ar simboliai taisyklingai vartojami.
- *Sintaksinė*, kai ar duomenų surašymo forma atitinka sutartąją.
- *Semantinė,* kai tikrinama pagal prasmę.
- *Lokalioji*, kai tikrinama kokia nors duomenų rinkmena (failas).
- *Globalioji*, kai analizuojami duomenys skirtingose duomenų rinkmenose (failuose).
- *Nepriklausoma arba neatskiriama* nuo skaičiavimų (tikrinama ar duomenys tenkina konkrečias skaičiavimams būtinas sąlygas).



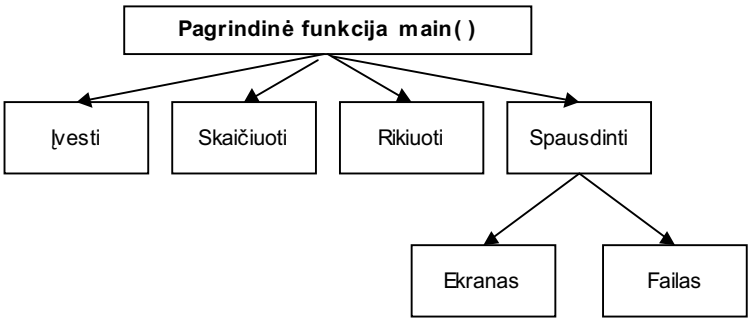
1.3 pav. Programos struktūrinė schema

Sudarius programos struktūrinę schemą, paruošiama *programos modulinė schema*, tai programos struktūrinių elementų aprašymas. (žr. 1.4 pav.).

Programos modulinė schema parodo įvairių programos modulių tarpusavio priklausomybę.

Modulis - tai programos dalis, atliekanti tam tikrą funkciją.

Pavyzdžiui moduliai, įvedantys duomenis, atliekantys skaičiavimus, surandantys mažiausią reikšmę tarp duotų ir kiti. Kiekvienas programos modulis turi unikalų vardą, kuris turi būti žymimas schemoje. Programos modulinė schema turi kelių lygių hierarchinę struktūrą. Viršutiniame lygyje braižomas pagrindinis programos modulis. Pirmame lygyje braižomi moduliai, naudojami tik pagrindiniame modulyje. Antrame lygyje braižomi moduliai, naudojami pagrindiniame modulyje ir pirmo lygio moduliuose ir t.t. (žr. 1.4).



1.4 pav. C++ programos modulinė schema

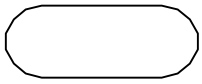
Modulinė schema padeda sudaryti programų darbo testus, programos derinimo metu, testuojant ryšius tarp modulių. Be to, išskirtos dalys gali būti pavedamos programuoti atskiriems programuotojams.

Algoritmų vaizdavimas

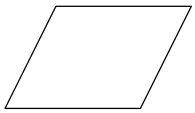
Uždavinių sprendimo algoritmai vaizduojami įvairiais būdais: pseudokodu (labai detaliai), blokinėmis schemomis, struktūrogramomis. Programavimo specialistų pripažįstama, kad pradedantieji lengviau įsisavina blokinę schemą, kaip algoritmų vaizdavimo būdą. Tačiau blokinė schema šiuo metu keičiama struktūrograma. Šioje mokomojoje knygoje rekomenduojame naudotis struktūrogramomis. Todėl tik labai trumpai apžvelgsime blokinių schemų sudarymo principus.

Blokinės schemos

Blokinėse schemose naudojami grafiniai simboliai (stačiakampis, rombas, lygiagretainis ir kt.) atitinka tam tikrą veiksmo tipą. Linijos, jungiančios šiuos blokus, rodo veiksmų atlikimo tvarką. Linijų normali kryptis – iš viršaus į apačią ir iš kairės į apačią. Kartais linijos gale braižoma rodyklė, rodanti perėjimo kryptį. Linijos gale gali būti tik vertikalios ir horizontalios, todėl jų kryptis keičiama tik stačiu kampu. Sudarant blokines schemas naudojami tokie grafiniai simboliai:

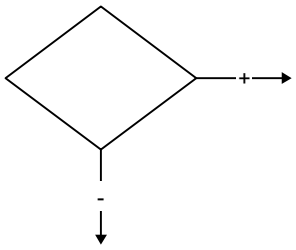


Pradžia ir Pabaiga. Rodoma algoritmo pradžia ir pabaiga. Atitinkamai įrašomas reikalingas žodis.

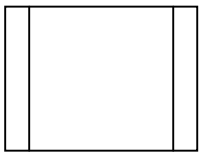


Įvestis ir Išvestis. Rodomi kintamųjų vardai, kurių reikšmės tai įvedami pradiniai duomenys arba išvedami programos rezultatai

Skaičiavimai. Aprašomi skaičiavimai į bloko vidų įrašant formulę.



Sąlygos tikrinimas. Įrašoma sąlyga, kuri programos vykdymo metu tikrinama. Nuo sąlygos tikrinimo rezultatų (sąlyga teisinga arba neteisinga), priklauso tolesnė skaičiavimo kryptis.



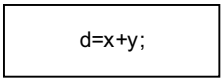
Kreipinys. Kreipinys nusakomas modulio vardu ir parametrų sąrašu, per kurį moduliui siunčiami pradiniai duomenys.

Jungtis rodo linijos perkėlimą į kitą vietą.

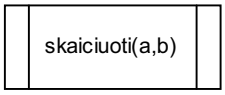
Grafinai vaizdai numeruojami, dažniausiai naudojant skaičius. Ši koordinatė rašoma bloko viršutinės linijos trūkyje.

Struktūrograma

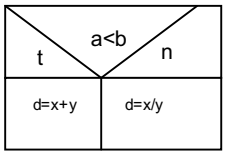
Naują algoritmo vaizdavimo būdą, struktūrogramą, sukūrė I.Nassi ir B.Shneiderman (1973). Struktūrogramoje naudojami šie pagrindiniai elementai:



Stačiakampis. Įrašomas programos arba jo modulio vardas, įvairūs skaičiavimai, aprašomi duomenų tipai ir kintamieji.

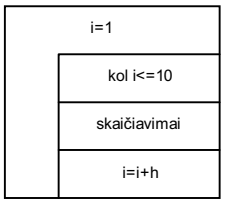


Kreipinys. Kreipinys nusakomas modulio(funkcijos) vardu ir parametrų sąrašu , per kurį siunčiami pradiniai duomenys.



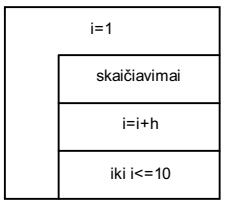
Sąlyginė valdymo struktūra.

```
if ( a < b )
  d = x + y;
else
  d = x / y;
```



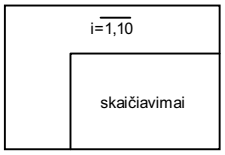
Ciklinė valdymo struktūra while.

```
i = 1;
while ( i <= 10 )
{
  skaičiavimai
  i = i + h;
}
```



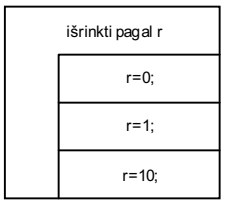
Ciklinė valdymo struktūra do while.

```
i = 0;
do
{
  skaičiavimai
  i = i + h; (arba i++)
}
while ( i <= 10 );
```



Ciklinė valdymo struktūra for.

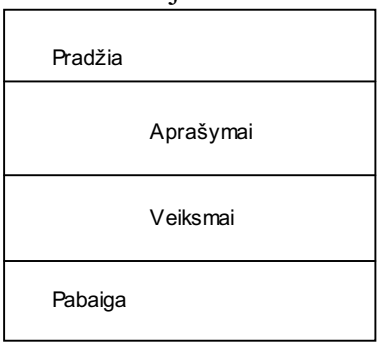
```
for ( i=0; i <= 10; i++ )
{
  skaičiavimai
}
```



Išrinkimas pagal požymį.

```
switch (kintamasis)
{
case (reikšmė 1):
sakiniai;
case (reikšmė 2):
sakiniai;
…
case (reikšmė n):
sakiniai
default:
sakiniai;
}
```

Programos modulio struktūrogramos bendroji forma rodoma 1.5 paveiksle.



1.5 pav. Programos modulio struktūrogramos bendroji forma.

Pradžios langelyje rašomas programos ar jos modulio vardas.

Aprašymų dalyje gali būti keli skyriai, kuriuose aprašomi sugrupuoti duomenų tipai, kintamieji, konstantos, kitos struktūros. Kintamieji aprašomi tokia tvarka: pradinių duomenų kintamieji, rezultatų kintamieji ir kiti kintamieji. Ciklų kintamųjų struktūrogramoje aprašyti nebūtina.

Veiksmų dalyje rašomi veiksmai jų logine seka.

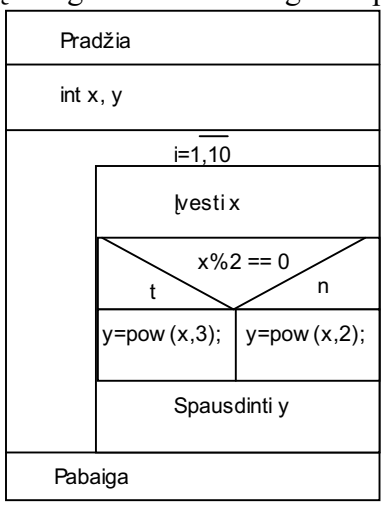
Pabaigoje rašomas programos ar jos modulio vardas.

Kiekvienos dalies viduje gali būti braižomi kiti struktūriniai elementai. Sudėtingesni elementai vaizduojami atskirose struktūrogramose. Šiuo atveju pagrindiniame modulyje įrašomas elemento vardas.

Vieną struktūrogramą patogiau sudaryti viename lape. Jeigu struktūrograma didelė, tai dalis veiksmų įvardinami blokais ir braižomos atskiros šių blokų struktūrogramos. Paprogramių algoritmų struktūrogramos braižomos atskirai.

1 Pavyzdys

Programa analizuoja įvedamus sveikuosius skaičius. Jeigu skaičius lyginis, skaičiuoja kubą, jeigu nelyginis skaičiuoja kvadratą. Rezultatus spausdina kompiuterio ekrane. Įvedus nulį, programa nutraukia darbą. Programos struktūrograma pateikiama 1.6 paveiksle.



1.6 pav. Programos struktūrograma pateikiama

Programos realizavimas

Programos realizavimo procesą sudaro šie žingsniai: programos kodavimas, programos testavimas, dokumentavimas ir programos eksploatavimas. Šiame skirsnyje plačiau kalbama apie programos testavimo ir dokumentavimo žingsnius.

Programos testavimas

Programos testavimo ir derinimo metu randamos ir ištaisomos aptiktos programos vykdymo klaidos.

Programų testai - tai įvairūs pradinių duomenų rinkiniai.

Skiriami du programos testavimo būdai - kompleksinis visos programos ir atskirų jos modulių testavimas.

Modulio testavimas - tai modulio vykdymas, pateikiant pradinius duomenų rinkinius.

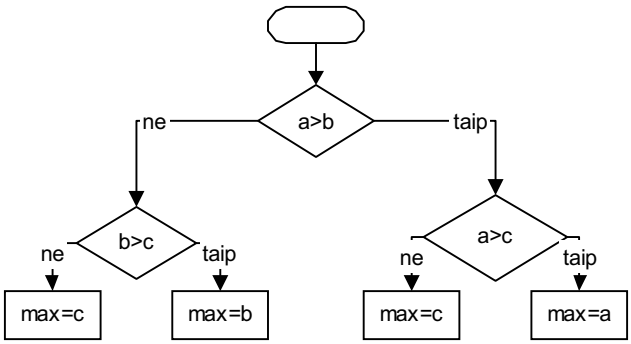
Kompleksinis programos testavimas - tai visos sistemos vykdymas, pateikiant testus, ir gautos informacijos lyginimas su numatomais rezultatais.

Testavimo procesą sudaro tokie žingsniai:

- testo projektavimas;
- testo pilnumo tikrinimas;
- testo vykdymas.

Testo projektavimas

1. Pagal modulio reikalavimus parenkamas testas kiekvienai situacijai, kiekvienai ribai, kiekvienai neleistinai sąlygai patikrinti.
2. Paruošiami testai kiekvienos sąlygos visoms kryptims patikrinti;
3. Patikrinama, ar sudaryti testai tikrina visus galimus kelius (pvz. kiekvienai ciklinei valdymo struktūrai būtini trys testai kai: ciklas visai nevykdomas; ciklas vykdomas tik vieną kartą; ciklas vykdomas daug kartų).
4. Nustatoma, kuriems duomenims programa yra jautri (pvz. dalyba iš nulio, pošaknio leistina reikšmė, arba kai kurių trigonometrinių funkcijų argumentų leistinų reikšmių aibė (ne mažesni už nulį), įrašų sąrašo ribos (tuščias, turi tik vieną įrašą, jau yra surūšiuotas, turi vienodas reikšmes).



1.7 pav. Didžiausios reikšmės radimas tarp trijų duotų skaičių a, b ir c.

Testų lentelė

1.1 lentelė

Testo nr.	**Testai**			**Numatomi rezultatai**
	a	**b**	**c**	
1.	10	6	5	max =a
2.	10	6	11	max=c
3.	6	10	8	max=b
4.	6	10	11	max=c

Testo pilnumo tikrinimas

Paruošus testą, rengiama testų pilnumo lentelė (žr. 1.2 lentelė) ir tikrinama ar dar reikalingi papildomi testai, ar jau visi keliai tikrinami.

Testų pilnumo lentelė

1.2 lentelė

Sąlyga	Kryptis	Testai			
		1	2	3	4
Sąlyga 1	Taip	+	+		
Sąlyga 1	Ne			+	
Sąlyga 2	Taip				
…	…		+		
…	…				

Rengiant testų pilnumo lentelę, peržiūrima kiekviena sąlyga ir kiekvienas kelias bei pažymimas testo, tikrinančio kelią, langelis. Šioje lentelėje negali likti eilučių, kuriose nėra nei vienos žymės. Jau minėjome anksčiau, kad patikrinus testų pilnumą, papildomai reikia parinkti testus, kurie patikrintų programos jautrumą įvairiems duomenims.

Testo vykdymas

Kiekvienas programos modulis testuojamas atskirai nuo visos programos. Testuojamas modulis vykdomas keletą kartų paeiliui pateikiant jam iš anksto paruoštus testus. Sudėtingi programų komplektai gali būti testuojami dviem būdais:

1. Tikrinant modulius, esančius žemiausiame lygyje, o po to testuojant aukštesniojo lygio modulius. Testavimas kartojamas, kol patikrinamas aukščiausio lygio modulis.
2. Pirmiausiai testuojami moduliai, esantys aukščiausiame lygyje, po to moduliai, esantys žemesniuose lygiuose. Jeigu kurių nors žemesniojo lygio modulių trūksta, jie keičiami intarpais. Dažniausiai intarpuose programuojamas kokio nors teksto spausdinimas, arba imituojami išvesties duomenys.

Programos dokumentavimas

Tai labai svarbus programinės įrangos realizavimo žingsnis, kurio metu paruošiami visos programos ir atskirų jos modulių aprašymai. Pagrindinė programos dokumentų dalis sudaroma uždavinio specifikavimo ir programų projektavimo ir kodavimo metu. Programų ir programų komplektų aprašymai perduodami programų projektuotojams, naudotojams ir asmenims, prižiūrintiems kompiuterių programas. Rekomenduojamos tokios programos ir kiekvieno modulio aprašymo dalys:

- Programos(modulio) antraštė, programos autorius, paskutiniojo modifikavimo data.
- Parametrų sąrašas ir parametrų paskirties aprašymas.
- Programos (modulio) paskirtį aiškinantis tekstas bei naudojami metodai ir būdai.
- Programos (modulio) modulinė schema.
- Pagrindinės programos ir kiekvieno jos modulio sprendimo algoritmas.
- Programavimo kalba užrašytas pagrindinės programos ir kiekvieno jos modulio tekstas.
- Duomenų apribojimai ir avarinės situacijos.
- Programos (modulio) panaudojimo iliustracija (duomenų ir rezultatų pavyzdys).

Kontroliniai klausimai

1. Paaiškinkite skirtumą tarp informacijos ir duomenų sąvokų.
2. Ką vadiname informacinėmis technologijomis?
3. Paaiškinkite algoritmo sąvoką.
4. Ką vadiname kompiuterine programa?
5. Paaiškinkite skirtiumą tarp kintamojo ir konstantos.
6. Apibūdinkite duomenų tipo sąvoką.
7. Paaiškinkite C++ kalbos duomenų tipų tipologiją.
8. Apibūdinkite skirtumus tarp žemojo ir aukštojo lygio programavimo kalbų.
9. Kokio lygio programavimo kalba yra C++ kalba?
10. Kokių operacinių sistemų aplinkoje gali dirbti *Borland C++* programos?
11. Išvardinkite programavimo aplinkos sudėtines dalis ir paaiškinkite jų paskirtį.
12. Paaiškinkite projekto sąvoką.
13. Išvardinkite programos projektavimo ir programos realizavimo etapų žingsnius.
14. Pakomentuokite jums žinomus duomenų kontrolės tipus.

Užduotys

1. Sudarykite strukūrogramą programai, kuri tarp trijų, įvestų klaviatūra skaičių, rastų mažiausią reikšmę ir ją spausdintų.
2. Sudarykite pirmosios užduoties programos testų ir testų pilnumo lenteles.

Santrauka

Šiame skyriuje susipažinote su programavimo pagrindinėmis sąvokomis ir terminais, duomenų tipų savybėmis ir jų klasifikavimu C++ kalboje, programavimo kalbų raida ir integruotų programavimo aplinkų sudėtinėmis dalimis. Jūs jau žinote programų projektavimo principus ir eigą bei uždavinių sprendimo algoritmų vaizdavimo būdus. Be to šiame skyriuje išsiaiškinome programų testų sudarymo principus ir reikalavimus programų bei jų modulių aprašymams.

Informacijos šaltiniai

2. C++ PROGRAMOS SUDARYMAS

Šiame skyriuje skaitytojas susipažįsta su populiariomis programavimo metodikomis ir C++ kalbos programos struktūra. Be to pateikiami reikalavimai programos stiliui ir jo vertinimo kriterijai. Taip pat skaitytojas supažindinamas su Borland C++ programavimo aplinkos projekto sudarymo tvarka, pirminio programos modulio kompiliavimu ir projekto redagavimu.

➢ Mokymosi tikslai

Mokydamasis ir baigęs šį skyrių skaitytojas turi:

- susipažinti su C++ kalbos programos struktūra;
- išmokti sudaryti C++ kalbos programas ir jų modulius;
- išmokti rašyti gero stiliaus programos tekstą;
- mokėti sukurti naują projektą;
- mokėti papildyti projektą naujais failais.

2.1 Programavimo metodikos

Pastaruoju metu naudojamos dvi programų sudarymo metodikos - struktūrinis programavimas ir objektinis programavimas.

Programos lengvai skaitomos, taisomos, pertvarkomos, kai jos struktūruotos, tai yra sudarytos iš nedidelių dalių. Tipiniai programų struktūros elementai išskiriami vadovaujantis programų struktūravimo principais.

Struktūrinio programavimo metodika nurodo, kad programose iš pradžių sukuriamos duomenų struktūros ir jų tvarkymo priemonės ir tik po to, panaudojant šias priemones, aprašomas uždavinio sprendimo algoritmas. Struktūrinio programavimo principai:

- Formuluojama sprendžiama problema.
- Nustatomi programos duomenys ir rezultatai.
- Nustatomos duomenų apdorojimo procedūros.
- Aprašomi programos struktūriniai elementai (funkcijos, paprogramės).
- Parengiamas programos tekstas.
- Programa kompiliuojama, derinama ir testuojama.

Taikant objektinio programavimo metodiką, programa sudaroma iš abstraktaus tipo objektų. Objektinio programavimo principai:

- Formuluojama sprendžiama problema
- Aprašomi objektai (meniu, langai, mygtukai ir t.t.)
- Nustatomi ryšiai tarp objektų.
- Sukuriamos objektų klasės, aprašomi kintamieji, nurodantys galimas objektų būsenas.
- Aprašomi pranešimai objektams.
- Pranešimus valdančios funkcijos (metodai) prijungiamos prie objektų klasių.
- Klasės testuojamos.
- Aprašomi klasių objektai.
- Apibrėžiama pradinė sistemos būsena.
- Kompiliuojama, komponuojama ir vykdoma programa.

2.2 C++ Programos struktūra

Prieš sudarant programas C++ kalba, siūloma atkreipti dėmesį į šias taisykles:

- naudojama tik viena programų struktūravimo priemonė – funkcijos;
- programos sąvoką atitinka pagrindinis modulis (funkcija), žymima vardu *main*;
- programoje turi būti tik viena *main* funkcija;
- programos objektų (struktūrų, kintamųjų, funkcijų) apibrėžtys ir aprašai gali būti bet kurioje programos vietoje. Svarbu tik tai, kad objektas būtų aprašytas prieš jį naudojant;
- aprašuose naudojami funkcijų prototipai;
- C++ kalboje realizuojant standartinius veiksmus naudojamos įvairios bibliotekų sistemos.

Funkcijos aprašo sintaksė

reikšmės tipas f-jos vardas ([parametrų sąrašas])
{
[lokaliųjų kintamųjų apibrėžtys]
tekstas
return (reikšmė);
}

C++programos pagrindinės funkcijos *main* tipinė struktūra pateikiama 2.1 paveiksle. Naudojantis funkcijomis patariama laikytis šių taisyklių:

- Funkcijos *main* apraše jos tipą galima praleisti.
- Parametrai – tai ryšio su kitais programos struktūriniais elementais priemonės.
- Lokaliųjų kintamųjų apibrėžtys rodo tik funkcijos viduje naudojamas struktūras.
- Jeigu skaičiuojama funkcijos reikšmė, jos tekste turi būti sakinys *return (reikšmė);*
- Jeigu funkcijos reikšmė neskaičiuojama, tai funkcijos reikšmės tipas turi būti *void*, o sakinį *return* galima praleisti.

```
[/* instrukcijos pirminiui procesoriui]
        .
        .
[/* globaliosios apibrėžtys]
        .
        .

[/* naudotojo funkcijų prototipai]
        .
        .

main()                                  // pagrindinė funkcija
        {                               // bloko pradžia
        [ lokaliosios apibrėžtys ]
        tekstas
        return 0;
        }                               // bloko pabaiga

[/* naudotojo funkcijų realizacijų aprašai]
```

2.1 pav. Funkcijos main tipinė struktūra

Programos failo pradžioje rašomos *instrukcijos pirminiui procesoriui*, nurodančios kokie pakeitimai turi būti atlikti programos tekste prieš kompiliavimą, kokius aprašus saugomus kituose failuose reikia įtraukti į programą.
Instrukcijų pirminiam procesoriui sintaksė:

 # include <bibliotekos sąsajos failo vardas>

C++ programavimo aplinkos bibliotekos sudaromos iš funkcijų realizacijų aprašų objektinių modulių ir sąsajų, saugomų atskiruose failuose. Bibliotekų sąsajų failai – tai ryšio su bibliotekomis priemonės. Jų vardai sudaryti panaudojant plėtinį (.h).Pirminio procesoriaus instrukcijos *include* rodo, kokių bibliotekų sąsajų tekstus reikia įtraukti į kompiliuojamą programą.

#include <iostream.h>	įvesties ir išvesties
#include <math.h>	matematinių ir trigonometrinų funkcijų
#include <string.h>	simbolių eilučių apdorojimo

ir kitos.

Globaliosios apibrėžtys rodo tuos programos objektus, kurie gali būti vartojami visuose programos moduliuose.
Programos dalis, kuri rašoma už funkcijos *main()* atidarančio figūrinio skliaustelio, vadinama *pagrindine funkcija*.

C++ programoje gali būti ir kitų funkcijų, kurios rašomos prieš arba už pagrindinės funkcijos. Šiuo atveju prieš pagrindinę funkciją rašomas funkcijos prototipas (aprašas).

Programos dalis, ribota figūriniais skliausteliais, vadinama *bloku*.

Visi programos sakiniai baigiami kabliataškiu.

Lokaliosios apibrėžtys rodo funkcijos objektus, kurie naudojami tik toje funkcijoje, kurioje jie aprašyti.

Funkcijos prototipu vadinamas funkcijos antraštės sakinys. Rašant pagalbinių funkcijų aprašus po pagrindinės funkcijos, programos pradžioje būtina surašyti pagalbinių funkcijų protototipus. *Funkcijos realizacijos aprašas* – tai funkcijos veiksmų aprašas. Veikiančios C++ programos pavyzdys pateikiamas 2 pavyzdyje.

2 Pavyzdys

```
/* programa, kuri leidžia įvesti sveikąjį skaičių. Jeigu jis lyginis, skaičiuojamas šio
skaičiaus kvadratas. Jeigu skaičius nelyginis skaičiuojamas jo kubas. Programoje
naudojamos dvi pagalbinės funkcijos – skaičiaus kvadratui ir kubui skaičiuoti.*/

# include <iostream.h>                  //instrukcija pirminiam procesoriui
# include <math.h>

int kvadratas(int n);              // f-jos prototipas
int kubas(int n);                         // f-jos prototipas

main ()                                   // pagrindinė funkcija
{
int n;
        cout << "Įveskite sveikąjį skaičių\n";
        cin >> n;                 // klaviatūra įvedamas sveikasis skaičius
        if ( n % 2 == 0)
                cout << "skaiciaus"<<n<<"kvadratas yra"<< kvadratas (n)<<endl;
        else
                cout << "skaiciaus"<<n<<"kubas yra"<< kubas (n)<<endl;
return 0;       // programa užbaigiama komanda return
}
//----------------------- Funkcija kvadratas----------------------
int kvadratas(int n)
{
        int y;
        y=pow(n,2);
return (y);
}
//-----------------------Funkcija kubas----------------------
int kubas(int n)
{
        int y;
        y=pow(n,3);
return (y);
}
```

2.3 C++ programos stilius

Programa turi būti lengvai skaitoma, tuomet ją lengviau taisyti. Toliau pateikiama keletas patarimų, kuriais siūloma praktiškai naudotis:

- Programos tekstą galima pradėti rašyti bet kuria pozicija.
- Programos dalys viena nuo kitos skiriamos tuščiomis eilutėmis.
- Pirmosios programos eilutės pradedamos rašyti pirmąja pozicija, o visas programos kamienas ir veiksminiai programos teksto blokai pastumiami į dešinę.

- Visi rezervuoti žodžiai ir didesnė programos dalis yra rašoma mažosiomis raidėmis.
- Globaliųjų kintamųjų ir konstantų vardai rašomi didžiosiomis raidėmis.
- Klasių, funkcijų - konstruktorių ir funkcijų - destruktorių vardų pirmoji raidė yra didžioji.

Komentarai

Komentarus galima rašyti bet kurioje programos vietoje. Komentarai pradedami dviem į dešinę pasvirusiais brūkšniais (//). Jei komentarai užima kelias eilutes, jų pradžią galima žymėti (/*) ir baigti (*/).
Komentarai papildo programą, tačiau neturėtų trukdyti skaityti programą. Todėl siūloma rašyti komentarus:

- Programos pradžioje, nurodant programos paskirtį ir autorių bei paskutiniojo modifikavimo datą.
- Kintamųjų ir objektų aprašuose nurodant jų papskirtį.
- Prieš funkcijas ir veiksminius programos blokus nurodant jų paskirtį.
- Lakoniški komentarai rašomi programos tekste dešiniajame krašte.

Teksto rašymo kultūra

Gero stiliaus programa lengvai suprantama ir skaitoma. Tokios programos tekstas patogiai išdėstomas lape panaudojant atitraukimus ir tarpus tarp eilučių (žr. 2 Pavyzdys).
Programos tekste:

- Išdėstymas atitinka atliekamų veiksmų prioritetus;
- Atitraukimai išryškina sakinių pavaldumą;
- Naudojamos tuščios eilutės:
  - Tarp pagrindinių dalių (iki trijų);
  - Tarp nepagrindinių dalių (iki dviejų);
  - Prieš bloką (viena).
- Lygiuojama pagal poziciją struktūrose, blokuose, funkcijose;
- Funkcijos pradžia pažymima komentarų eilute (//----------funkcijos vardas).

2.4 Borland C++ 5.0 aplinkos projektų tvarkymas

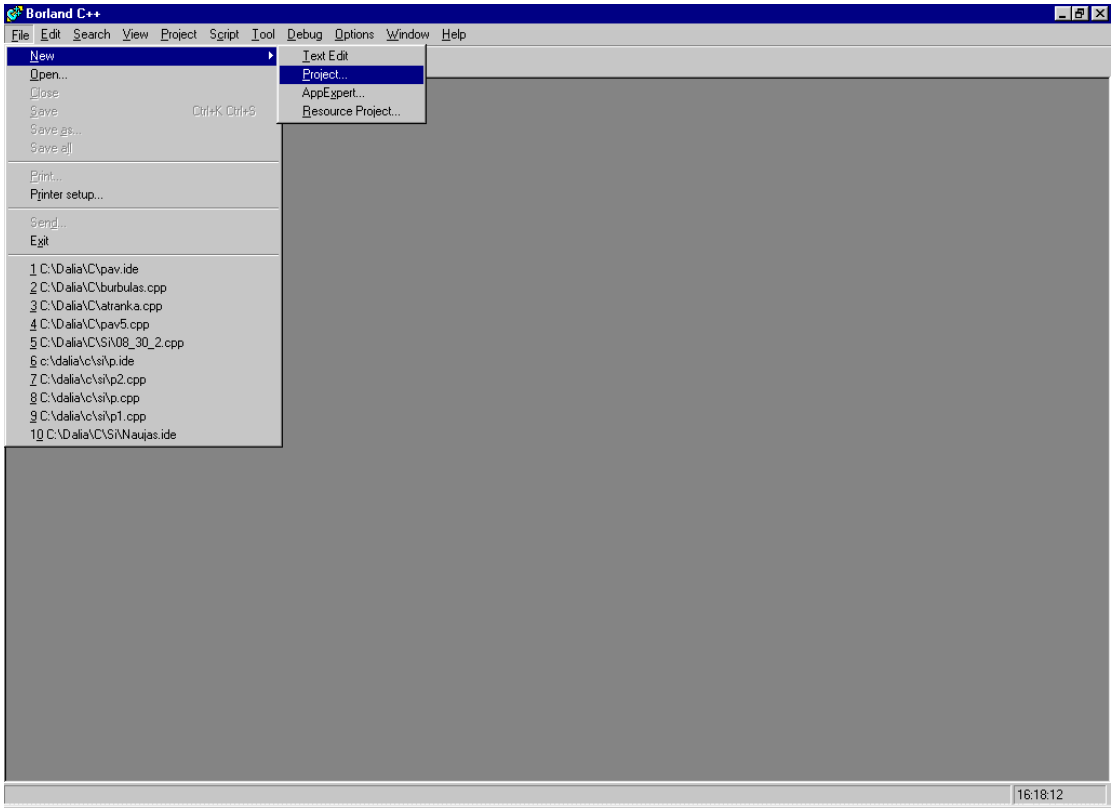
Borland C++ aplinkoje kuriami įvairių tipų ir paskirties projektai. Projektą sudaro keletas pirminių programos modulių ir duomenų failų, kurie rengiami programavimo aplinkoje.

Projektą sudaro šie failai:

- pirminiai programų moduliai .cpp
- vartojamų grafinės aplinkos išteklių aprašas .rc
- projekto savybių apibrėžčių failai .def
- projektas .ide
- bibliotekų sąsajų failai .h
- vykdomieji failai .exe

Projektas -failų, sudarančių programos vykdomąją bylą, komplektas.

Integruota programavimo aplinka startuojama panaudojant pradinį meniu arba piktogramą Windows aplinkos darbalaukyje. Bendras integruotos aplinkos vaizdas pateikiamas 2.2 pav. Po to projektai rengiami ir tvarkomi programos meniu arba komandų įrankiais.

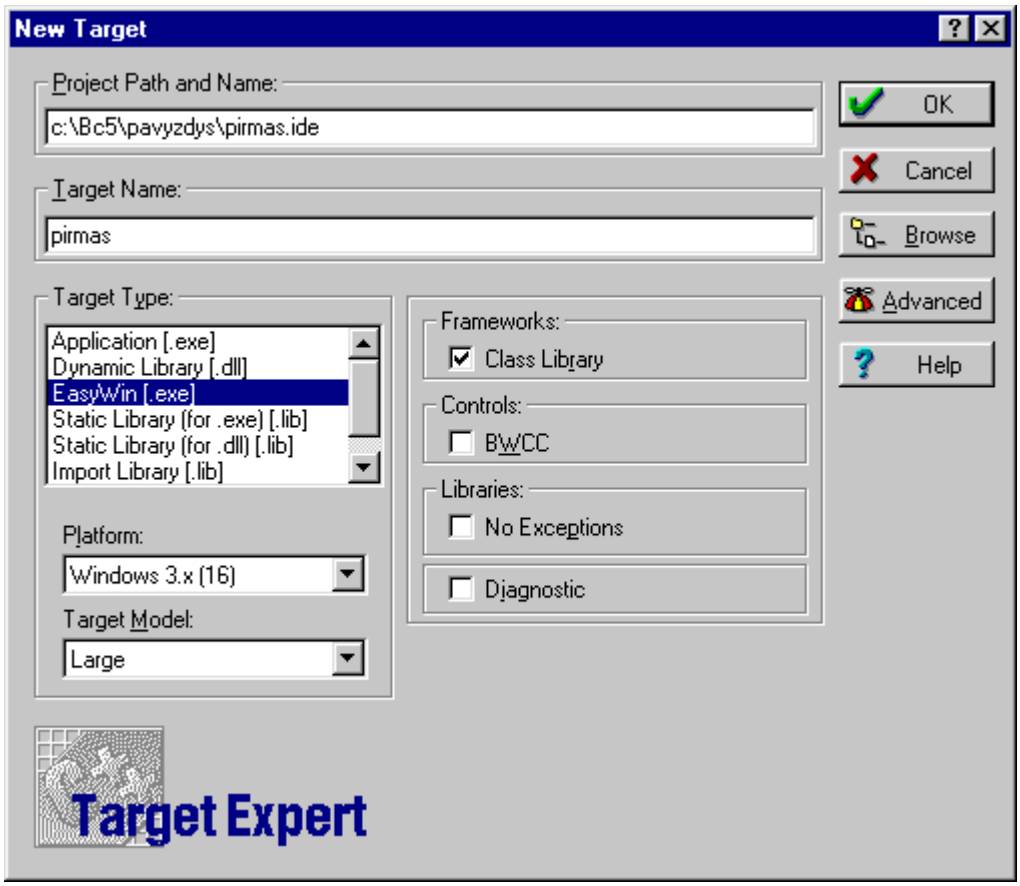


2.2 pav. Bendras Borland C++ integruotos aplinkos ekrano vaizdas

Kuriant naują projektą, naudojamos meniu elemento **File** komandos, o redaguojant jau egzistuojantį projektą, - meniu elemento **Project** komandos.

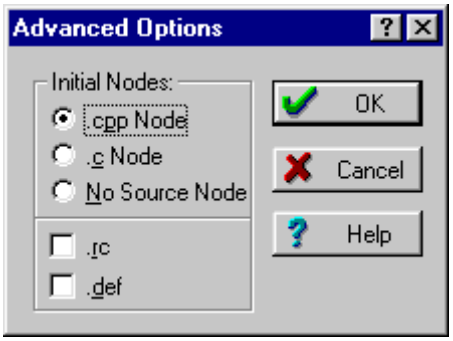
Naujo projekto sudarymo tvarka:

- Parenkama naujų projektų konstravimo komanda **File.New.Project** (žr. 2.2 pav.).
- Lango **New Target** teksto lauke **Project Path and Name** (Projekto vieta ir vardas) nurodomi darbo katalogo ir naujo projekto vardai (Pvz., pirmas.ide)
- Srityje **Target Type** (užduoties tipas) parenkamas **EasyWin** (lengvas langas) tipas (žr.2.3 pav.).



2.3 pav.Naujo projekto parametrų parinkimas

- Įvykdoma komanda **Advanced** , kurios dialogo langelis parodytas 2.4 pav., ir srityje **Inicial Node** išjungiami **.rc, .def** parametrai.

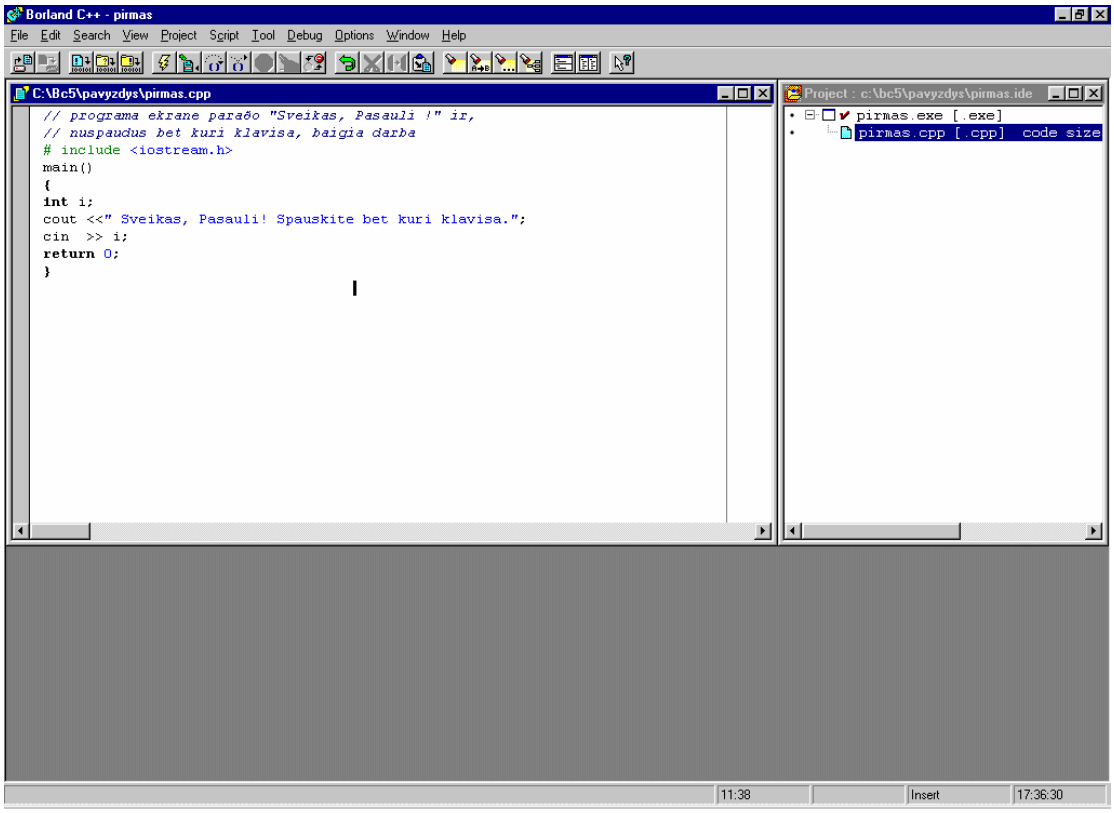


2.4 pav. Papildomi projekto parametrai

- Naujai kuriamo projekto parametrų lange pele parenkamas projekto elemento **.cpp** failo pavadinimas.
- Atsidariusiame tekstiniame lange surenkamas programos tekstas.

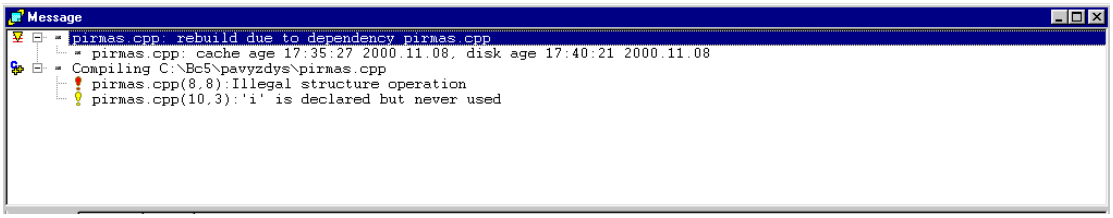
```
/* programa ekrane parašo “Sveikas, Pasauli !” ir, nuspaudus bet kurį klavišą, baigia
darbą*/
# include <iostream.h>
main()
{
        int i;
        cout      <<” Sveikas, Pasauli! Spauskite bet kurį klavišą.”;
        cin       >> i;
        return 0;
}
```

- Komanda **File.Save** užrašius programos tekstą į diską (žr. 2.5 pav.), galima pirminį programos modulį kompiliuoti.



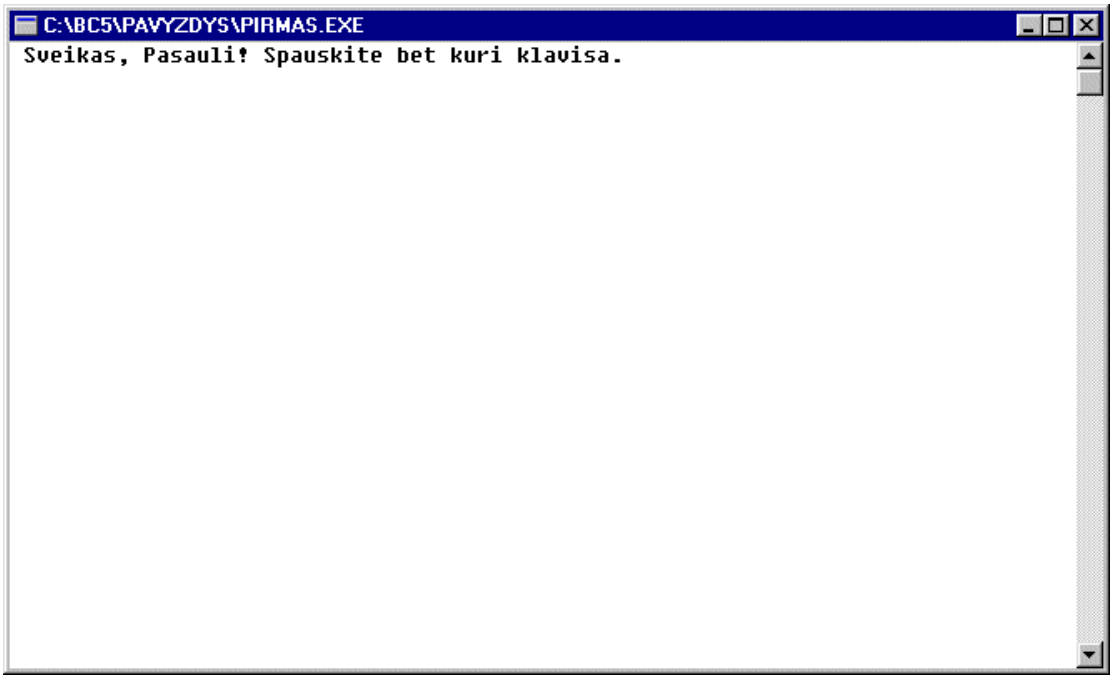
2.5 pav. Integruotos aplinkos Borland C++ išorinės sąsajos ekrano vaizdas

- Pirminis programos tekstas kompiliuojamas komanda **Debug.Run** (**Ctrl+F9**). Pranešimai apie programos kompiliavimo rezultatus ir sintaksės klaidas rodomi lange **Message** (žr. 2.6 pav.)



2.6 pav. Programos kompiliavimo rezultatai ir sintaksės klaidos

Programos darbo rezultatai rodomi tekstiniame lange (žr. 2.7 pav.).

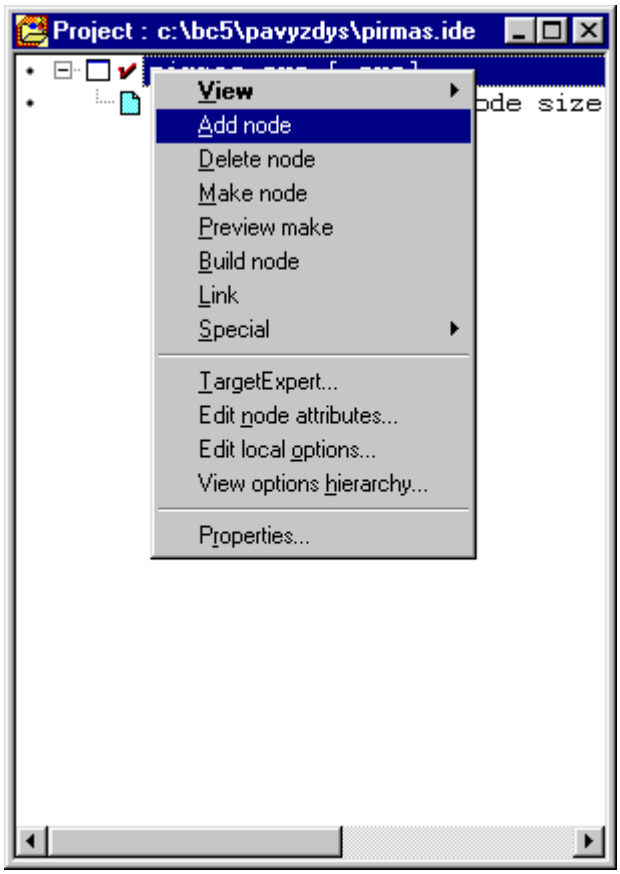


2.7 pav. Programos darbo rezultatai

Projekto redagavimas

- Komanda **View/Project** atveriamas aktyvaus projekto struktūros parinkimo langas.

Projekto struktūros elementai (node) redaguojami parenkant pagalbinio meniu komandas (žr. 2.8 pav):



2.8 pav. Projekto struktūros elementų pagalbinis meniu

Add node - prijungti naują failą.

Delete node - nereikalingo failo naikinimas.

Edit node atributte - koreguoti projekto failo parametrus.

Projekto pildymas naujais failais (žr. 2.8pav.):

1. Nauja vykdomoji byla įterpiama įrankiu **Add target to project** 
2. Nauja programos pradinio teksto bylą įdedama įrankiu **Add file to project**. 

Šie įrankiai randami standartinėje įrankių juostoje.

🕮 Kontroliniai klausimai

1. Paaiškinkite pagrindinės funkcijos tipinę struktūrą.
2. Kokia programa naudojama įvedant programos pradinį tekstą į kompiuterio atmintį?
3. Paaiškinkite kompiliatoriaus paskirtį.
4. Koks yra pirminio programos modulio failo plėtinys?
5. Kokiais simboliais yra pradedami programos komentarai?
6. Paaiškinkite funkcijos aprašo sintaksę.
7. Kokius žinote svarbiausius reikalavimus teksto rašymui?
8. Paaiškinkite instrukcijų pirminiam procesoriui paskirtį.

✎ Užduotys

1. Sukurkite naują projektą *antras.ide*.
2. Naujame programos pirminio programos teksto lange surinkite pavyzdyje pateiktos programos tekstą. Sukompiliuokite, ištaisykite klaidas, jeigu tokių atsirastų ir įvykdykite šią programą.

⧖ Santrauka

Šiame skyriuje susipažinote su populiariomis programavimo metodikomis ir C++ kalbos programos struktūra. Be to, žinote reikalavimus programos stiliui ir jo vertinimo kriterijus. Taip pat išmokote sudaryti ir tvarkyti projektus Borland C++ programavimo aplinkoje, kompiliuoti pirminįo programos modulį ir redauoti projektą.

ⓘ Informacijos šaltiniai