2-3 PASKAITOS

Turinys: Paprasčiausios programos pavyzdys. Darbas su programavimo terpėmis. Duomenys. Duomenų tipai ir charakteristikos. Paprasčiausia įvestis/išvestis.

Paprasčiausia programa:

```
/*
Pirmoji programa
*/
#include <iostream>
using namespace std;
//
int main( ){   // įėjimo į programą taškas
        cout<< "Pirmoji C++ programa\n";
        return 0;
}
```

Pastabos:

1. Pradedama vykdyti nuo funkcijos *main. int* – funkcijos grąžinamos reikšmės tipas. Skliausteliuose *( )* – argumentų sąrašas (čia – nėra). Funkcijos kūnas – skliaustuose *{ }*. *return 0* – operatorius grąžina tokį funkcijos darbo rezultatą; *int* duomuo. Dauguma operatorių baigiami *;* .
2. *cout << "Eilutė"* – eilutės tipo duomens išvestis į ekraną; << – išvesties operacija. *cout* – išvesties srauto klasės objektas.
3. Eilutės tipo duomuo baigiamas \n: perveda spausdinimą į kitą eilutę.
4. Direktyva *include* – nurodymas priešprocesoriui: įterpti į programos tekstą antraštinį failą *iostream*; jis yra VC98 katalogo INCLUDE subkataloge. Antraštinio failo vardas gali būti arba < >, arba " " viduje. Pirmu atveju kompiliatorius pradės antraštinio failo paiešką nuo INCLUDE katalogo, o antru – nuo darbinio katalogo.
5. Direktyva *using*: nurodo taikyti vardų sritį *std*. Programoje gali būti kelios vardų sritys.
6. Pageidautini komentarai: kelių eilučių /* … */ arba eilutiniai // … .
7. Užrašymo stilius: kompiliatorius ignoruoja daugelį formatavimo simbolių (tarpą, Tab ir pan.) – rašyti taip, kad būtų patogu apžvelgti tekstą.

Programos apdorojimas integruotoje *Developer Studio 97* programavimo terpėje:

1. Sukurti failą **.cpp: File –> New –> Files –> C++ Source File*; surinkti failo vardą be priesagos ir išrinkti reikiamą katalogą.
2. Sukurti projektą ir darbinę sritį (vienoje srityje galima talpinti kelis projektus): *File –> New –> Projects –> Win32 Console Application*. *Location* lauke surinkti kelią iki reikiamo katalogo, *Project Name* lauke – nurodyti failą talpinantį katalogą; įjungti *Create New Workspace* mygtuką.
3. Į projektą įjungti reikiamus failus: *Project –> Add To Project –> Files*.
4. Kompiliavimas, saistymas, vykdymas: *Build –> Compile* (*Build/Rebuild All, Execute*).

Programos apdorojimas integruotoje *Visual Studio 2010* programavimo terpėje (sutrumpintai ją vadinsim VS2010).

Centrinis terpės elementas - *.NET Framework* paketas, faktiškai esantis *Windows* OS dalimi. Savo ruožtu jis susideda iš dviejų dalių:

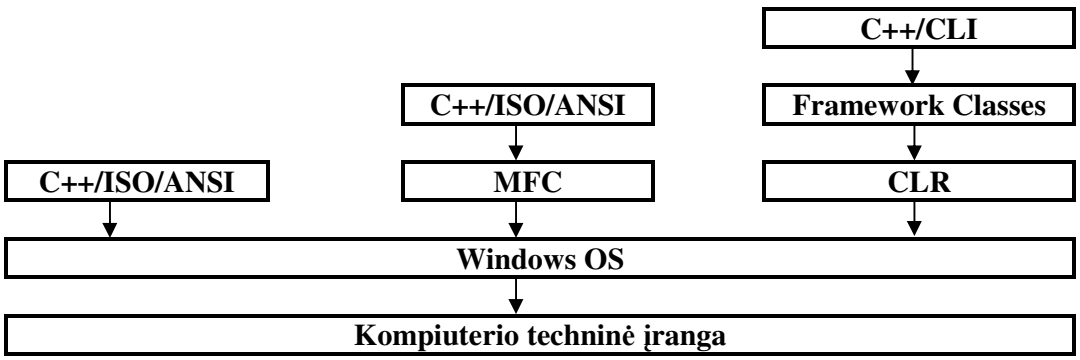
- *Common Language Runtime* (CLR) – bendrųjų programavimo kalbų terpė programai vykdyti;
- *.NET Framework* klasių bibliotekų – visos reikiamus veiksmus teikiančios klasės ir funkcijos, nepriklausančios nuo programavimo kalbos.

VS2010 leidžia kurti ir vykdyti dviejų tipų C++ programas:

- *C++/ISO/ANSI* programas – jos dar vadinamos prigimtinėmis C++ programomis; programa kuriama naudojant tik C++ *ISO (International Standards Organization)/ANSI (American National Standards Institute)* standarte esančias galimybes. Šiuo atveju programa vykdoma tiesiogiai kompiuteryje;
- *C++/CLI* (CLI: *Common Language Infrastructure*) programas – programos skirtos vykdyti CLR aplinkoje. Kalbos standartas turi plėtinių ir skirtumų lyginant su *ISO/ANSI* standartu. Vykdymas truputį lėtesnis nei analogiškos *C++/ISO/ANSI* programos.

CLI faktiškai yra virtuali mašina, į kurios kalbą kompiliuojamas daugelio programavimo kalbų tekstas; ši tarpinė kalba vadinama MSIL (*MicroSoft Intermediate Language*). Iš šios kalbos kompiliuojama jau tiesiai į mašinos kodus.

Abiejų tipų kalbomis galima kurti ne tik įprastas programas duomenims apdoroti, bet ir grafines vartotojo sąsajas (GUI, *Graphical User Interface*). Sąsajos kuriamos arba su senesniu įrankiu – klasių paketu MFC (*Microsoft Foundations Classes*), kuris savo ruožtu naudoja paketą API (*Application Programming Interface*), arba su vėliau sukurtu paketu *Framework Classes*; jame yra patogus grafinis sąsajų kūrimo įrankis, automatiškai sugeneruojantis sąsajos kodą. MFC galima naudoti abiejų tipų C++ programose, o *Framework Classes* – tik su CLR.
Tuo būdu, supaprastintai VS2010 programos kūrimo galimybės yra:



Programavimo terpės esminiai elementai programoms kurti ir vykdyti yra:

- teksto redaktorius;
- kompiliatorius;
- saistytojas;
- bibliotekos:
  - standartinė biblioteka Standard C++ Library – pagrindinės funkcijos ISO/ANSI kompiliatoriui;
  - MFC;
  - *Windows Forms* – analogiškiems dalykams, kaip MFC, tik skirta darbui su *.NET Framework*.

Su VS2010 kuriama programa talpinama į vadinamąjį projektą (*Project*) – tai yra konteineris programos failui ir daugeliui pagalbinių programos failų saugoti. Tam sukuriamas failų aplankas, o jame pagrindinis yra **.vcproj* failas.
Projektas talpinamas į vadinamąją sprendinio aplinką (*Solution*) – tai yra mechanizmas, leidžiantis surinkti į visumą visas programas ir pagalbines priemones konkrečiam uždaviniui (t.y. programai) spręsti. Techniškai tai – failų aplankas vienam ar keliems skirtingiems projektams talpinti. Informacija apie *Solution* projektus saugoma failuose **.sln* ir **.suo*. Jei kuriant projektą kitaip nenurodysim, automatiškai bus sukurtas ir naujas *Solution*.

Projektų kūrimas

Galima sukurti skirtingų tipų projektus. Pats paprasčiausias – **komandinės eilutės projektas (*Win32 Console Application*).**
Kūrimo eiga:

1. *File -> New -> Project..*: kairiajame lange *Project Types* rinktis *Win32*, o dešiniajame lange *Templates* rinktis *Win32 Console Application*.

2. Apatiniuose laukeliuose įvesti norimą projekto pavadinimą, norimą jo saugojimo vietą, o *Solution* pavadinimas automatiškai pasiūlomas toks pat kaip ir pasirinktasis projektui – jį galima palikti.

3. Atveriamas projekto kūrimo vedlio langas. Yra dvi galimybės: rinktis standartines projekto nuostatas (tiesiog spustelėti *Finish*), arba įvesti savas (spustelti *Next*).
3.1. Pirmuoju atveju terpė automatiškai sukuria pradinį programos failą (tarkim, pasirinkom jo pavadinimą *programa.cpp*), antraštinius failus *stdafx.h* ir *targetver.h*, dar *stdafx.cpp* ir *ReadMe.txt* su visais paaiškinimais, kam reikalingi visi šie failai. Faile *programa.cpp* dar bus sugeneruota dalis programos teksto:

```
// programa.cpp : Defines the entry point for the console application.
#include „stdafx.h“

int _tmain( int argc, _TCHAR* argv[] )
{
        return 0;
}
```

- jį lieka tik papildyti mums reikiamu tekstu. Pastaba *.

3.2. Spustelėjus *Next*, pažymėti žymimąjį laukelį *Empty project*, ir tik tada spustelti mygtuką *Finish*: projektas sukuriamas, bet be pradinio failo. Failas sukuriamas dviem būdais taip:
3.2.1. *File-> New -> File..*: atverto dialogo lango *New File* kairiajame laukelyje rinktis *Visual C++*, o tada dešiniajame laukelyje – *C++ File (.cpp)* ir spustelti *Open*: atveriamas tuščias langas programos tekstui spausdinti. Atspausdinus reikiamą tekstą ir jį išsaugojus, failas įjungiamas į projektą *Project -> Add Existing Item*: pasirenkamas failas, spustelti mygtuką *Add*.
3.2.2. *Solution Explorer* lange ant *Source Files* spustelti dešinį pelės klavišą, kontekstiniame meniu rinktis *Add -> New Item*: atsivėrusiame dialogo lange *Add New Item – programa* kairiajame laukelyje rinktis *Code*, tada dešiniajame - *C++ File (.cpp)*, apatiniuose laukeliuose įvesti failo pavadinimą ir norimą vietą bei spustelėti *Add*. Atvertame lauke spausdinti tekstą, išsaugoti. Failas priklausys projektui.

4. Kompiliuoti ir saistyti programą: *Build -> Build Solution*. Pastaba**.

5. Paleisti programą: *Debug -> Start Debugging* (jei programą dar reikia derinti) arba *Debug -> Start Without Debugging* (jei reikia tik vykdyti).

Pastaba *. C++/ISO/ANSI standarte startinės programos funkcijos pavadinimas yra *main*. C++/CLI standarte pavadinimas yra arba *main*, arba *wmain* (jei programa naudoja *Unicode* simbolius; tam programoje turi būti direktyva *#define_UNICODE*). *_tmain* yra tik tarpinis CLI standarto pavadinimas; jis apdorojant programą automatiškai pakeičiamas į *main* arba *wmain*.
Pastaba.** 4-uoju žingsniu galima sukurti dvi programos vykdomojo failo versijas:
Project -> programa Properties: atveriamas langas *programa Property Pages*, jame spustelti mygtuką *Configuration Manager*: bus atvertas dar vienas langas, kuriame yra išsiskleidžiantis sąrašas *Active Solution Configuration* su numatytąja reikšme *Debug*. Išskleidus sąrašą galima rinktis ir reikšmę *Release*. Programos derinimo versija (*Debug*) leidžia vykdyti programą žingsniais (jei programą, iš pradių sudėjus testavimo žymenis, paleisti *Debug -> Start Debugging*) ir stebėti einamąsias programos kintamųjų vertes – tai patogu programai testuoti. Programos leidimo versija (*Release*) neturi tokios derinimo galimybės, tačiau yra žymiai greitesnė. Taigi, ištestavus programą reikėtų ją iš naujo kompiliuoti ir saistyti į efektyvesnę *Release* versiją ir toliau tik ją ir naudoti.

Komandinės eilutės projektas ***CLR Console Project***:

1. *File -> New -> Project..*: dialogo lango *New Project* kairiajame laukelyje rinktis *CLR* , o tada dešiniajame – *CLR Console Application*. Apatiniuose laukeliuose pasirinkti pavadinimą, vietą, *Solution* pavadinimą ir spustelti *OK*. Sukuriami 8 failai ir atveriamas iš dalies parengtas programos pagrindinis failas:

```
// programa.cpp : main project file.

#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args)
{
        Console::WriteLine(L"Hello World");
        return 0;
}
```

Taigi, čia jau taikoma C++/CLI kalbos sintaksė. Kiti programos kompiliavimo, saistymo ir vykdymo žingsniai – tokie pat, kaip anksčiau.

MFC projektas (***MFC Application***):

1. *File -> New -> Project..*: dialogo lango *New Project* kairiajame laukelyje rinktis *MFC* , o tada dešiniajame – *MFC Application*. Apatiniuose laukeliuose pasirinkti pavadinimą, vietą, *Solution* pavadinimą ir spustelti *OK*. Atveriamas *MFC Application* projekto vedlio langas su keliomis siūlomomis numatytosiomis nuostatomis; spustelėjus *Finish* jos visos priimamos. Sukuriami 13 antraštinių, 27 sąsajos elementų (*Resource*), 11 pradinių ir *ReadMe.txt* failai. Du kartus spustelėjus *programa.cpp* failą, atveriamas iš dalies parengtas programos pagrindinis failas.

Šį projektą tokiu būdu kaip anksčiau galima kompiliuoti, saistyti ir paleisti: bus atvertas tuščias grafinės vartotojo sąsajos langas; sąsajos visi grafiniai elementai - veiksnūs.

Projektas *Windows Forms Application*:

1. *File -> New -> Project..*: dialogo lango *New Project* kairiajame laukelyje rinktis *CLR* , o tada dešiniajame – *Windows Forms Application*. Apatiniuose laukeliuose pasirinkti pavadinimą, vietą, *Solution* pavadinimą ir spustelti *OK*. Atveriamas sąsajos su pavadinimu *Form1*langas. *Solution Explorer* lange matomi sukurti 3 antraštiniai, 2 sąsajos elementų (*Resource*), 3 pradiniai ir *ReadMe.txt* failai. Du kartus spustelėjus *programa.cpp* failą, atveriamas iš dalies parengtas programos pagrindinis failas.

Sąsajai kurti dabar gali būti taikomas grafinis sąsajos įrankis, suaktyvinamas *View -> Toolbox*. Reikiami elementai pele tiesiog pertempiami į sąsajos langą.

Tolesni programos apdorojimo žingsniai – tokie pat kaip anksčiau.

DUOMENYS

– visa tai, ką programa įveda, apdoroja ir išveda po apdorojimo.

Kintamieji ir konstantos. Kintamieji duomenys, programoje išreiškiami adresu – kintamojo vardu. Konstantos – išreiškiamos jas sudarančiais simboliais.
Duomenų tipai: aritmetiniai – sveikieji ir realieji; simboliniai, loginiai.

Sveikieji duomenys. Jų rūšys ir charakteristikos (atminties ląstelės ilgis, reikšmių diapazonas:

char (sąlyginai)	1B	-128 – +127
short	2B	-32768 – +32767
short int	2B	-32768 – +32767
int	4B (priklauso nuo OS)	~-2.1e9 – ~+2.1e9
long	4B	~-2.1e9 – ~+2.1e9
long int	4B (priklauso nuo OS)	~-2.1e9 – ~+2.1e9

unsigned porūšiai: beženklė forma, todėl, pavyzdžiui, *unsigned char* diapazonas – 0 – 255.
Alternatyvūs rūšių žymėjimai: formą su ženklu dar galima žymėt papildomu raktažodžiu *signed*: *signed int* ≡ *int* ≡ *signed*; panašiai ir *unsigned int* ≡ *unsigned.*

Sveikosios konstantos: -123, 123, +123, 123*U*, 123*u*, -123*L*, -123*l*, 123*UL*, 123*ul*.

Sveikieji kintamieji: prieš naudojimą turi būti deklaruoti ir apibrėžti bei inicializuoti. Deklaravimas ir apibrėžimas:

```
int var1;
int var2, var3;
```

ir pan.; galima deklaruoti ir kai kurių operatorių viduje. Deklaruoti galima bet kurioje programos vietoje; priimta – viršuje programos. Po šios operacijos kintamieji gauna atmintį. Įmanoma kintamuosius tik deklaruoti nepaskiriant atminties.
var1, var2, var3 – identifikatoriai, arba kintamųjų vardai. Vardai sudaromi iš raidžių, skaitmenų ir kai kurių specialiųjų simbolių; pirmasis simbolis – ne skaitmuo. Vardo ilgis teoriškai yra neribotas, praktiškai – keli šimtai simbolių. Kreipiamas dėmesys į didžiąsias ir mažąsias raides. Negalima vardams taikyti rezervuotų C++ žodžių (yra 64 tokie žodžiai).
Vardų sudarymo stilius: *intvar, intVar* (Sun stilius), *IntVar, int_var, Int_Var, iintvar* (“Hungarian Notation” stilius).
Inicializavimas – reikšmės suteikimas. Paprasčiausiu atveju – prieskyros operatoriumi:

```
int var1;
var1 = 123;
```

Deklaravimą, apibrėžimą ir inicializavimą galima sutapdinti į vieną operatorių:

long var2 = 456L; arba *long var2(456L);*

Jei kintamiesiems priskiriama reišmė išeina už jų kitimo diapazono ribų – į kintamojo ląstelę įrašoma visai kita, pagal gana sudėtingas taisykles gaunama reikšmė.

Simboliniai duomenys
ASCII simboliai:

char 1B -128 - +127

Koduojami pagal ASCII lentelę: simboliai interpretuojami kaip skaičiai – kodai nuo 0 iki 127, todėl sąlyginai juos galima laikyti sveikaisiais duomenimis. Windows OS taiko "išplėstinę ASCII lentelę": 0 – 255; yra ir ne lotyniško alfabeto simboliai, grafiniai simboliai. Kodams virš 127 standarto nėra.
Pavyzdžiai:

8	bakspace
9	Tab
13	Enter
32	<tarpas>
65	A
97	a

Simbolinės konstantos: *'A', 'a'*;
bei konstantos - valdančiosios sekos su \ simboliu, nurodančiu kitaip interpretuoti toliau einantį simbolį:

\b	<tarpas>
\f	<perėjimas į kitą puslapį>
\r	<perėjimas į kitą eilutę; Enter>
\t	Tab
\n	<perėjimas į kitos eilutės pradžią>
\\	\
\'	'
\"	"
\xdd	šešioliktainis kodo simbolis – kai reikia įvesti kokį, pavyzdžiui, grafinį simbolį. Tarkim, užtušuotas kvadratas pagal ASCII lentelę yra 178; jo šešioliktainis kodas – B2, todėl simbolį reikia įvesti kaip '\xB2' arba '\xb2'

UNICODE simboliai (kad programoje būtų galima juos naudoti, būtina įrašyti direktyvą *#define _UNICODE*):

wchar_t 2B 0 – 65535

Simbolinės konstantos: *L'A'*, *L'a'*.

Simboliniai kintamieji: taisyklės analogiškos sveikųjų kintamųjų taisyklėms:

char c = 'A'; arba *char c('A');*
wchar_t w = L'Z'; arba *wchar_t w(L'Z');*

Realieji duomenys – duomenys su trupmenine dalimi (ji gali būti ir nulinė). Realizuoti pagal IEEE 7054 standartą: visas kitimo diapazonas sudalintas į baigtinį realiųjų skaičių kiekį; tarp gretimų skaičių yra tarpai. Kuo skaičiai absoliutiniu dydžiu didesni, tuo tie tarpai didesni. Todėl faktiškai dviejų realiųjų skaičių lyginti tarpusavyje *r1 = r2* negalima – reikia atsižvelgti į skaičių tikslumą: | *r1 – r2* | < *tol* . Rūšys ir charakteristikos (r.s. – tikslumas reikšminiais skaitmenimis):

float 4B ~-1e+38 - ~-1e-38 ir ~+1e-38 - ~+1e+38 7 r.s.
double 8B ~-1e+308 - ~-1e-308 ir ~+1e-308 - ~+1e+308 15-16 r.s.
long double – priklauso nuo kompiliatoriaus; dažnai sutampa su *double*

Konstantos: -1.23*F*, -1.23*f*; -1.23. Eksponentinė (*scientific*) forma: -0.123*E*+1, -0.123*e*+1, .123*E*1, 123*E*-2, 1.23*L*, 1.23*l*.

Kintamieji: analogiškos taisyklės.

Loginiai duomenys:

bool 1B reikšmės *true* arba *false*

Kaip *true* interpretuojama ir bet kokia nenulinė skaitinė reikšmė; kaip *false* – nulinė.

Vardinės konstantos – su modifikatoriumi const. Pavyzdžiui,

```
const float PI = 3.14159f;
```

Bandymas pakeisti PI reikšmę iššauktų kompiliavimo klaidą. Paprastai tokių vardinių konstantų vardai rašomi didžiaisiais simboliais. Tą pat galima atlikti C (ne C++) direktyva, tik konstanta bus betipė:

```
#define PI 3.14159f
```

Paprasčiausia įvestis/išvestis

```
#include <iostream>
using namespace std;
int main( ){
        int a, b;
        cout<<"Iveskite du skaicius:";
        cin>>a>>b;  //1,2
        int c = a + b;  //3
        cout<<"\xB2 a = "<<a<<"\xB2 b = "<<b<<endl;          //1,2
        cout<<"Rezultatas: "<<c<<endl;
        return 0;
}
```

Pastabos:

1. *cout, cin* – išvesties/įvesties klasių objektai: ekranas ir klaviatūra. Galima priskirti ir kitus įrenginius.
2. << – įvesties operacija, >> – įterpties. Jas galima kartoti vienoje eilutėje kaip *//1*.
3. Įvesties į ekraną operacijos išveda duomenis į vieną eilutę. Pereinama į kitą tik valdančiąja seka \n arba manipuliatoriumi *endl* (jis bei *cout* ir *cin* aprašyti antraštiniame faile *iostream*).
4. Iš *//3* matyti, kad kintamąjį skelbti, apibrėžti ir inicializuoti galima ten, kur jis naudojamas.