C++ STANDARTAI

Naujausias C++ kalbos standartas ISO/IEC 14882:2017 (žinomas kaip C++17, ankstesnės juodraštinės (angl. Draft) versijos vadinamos C++0x, C++11, C++14) pateikė naujausią C++ kalbos raidos iteraciją.

Jau ISO/IEC 14882:2011 C++11 standartas pateikia reikalavimus kalbos realizacijai ir įvedė tam tikrus esminius papildymus, tiek kalbos branduoliui, tiek C++ standartinei bibliotekai. Šis standartas buvo patvirtintas ISO komiteto 2011 m. rugpjūčio 12 d. ir pakeitė ankstesnį C++03.

Naujausias laisvai prieinamas juodraštinis C++17 standarto dokumentas (galutinis standartas yra mokamas) prieinamas adresu:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf

C++ išlieka bendros paskirties parogramavimo kalba, grindžiama C programavimo kalba (specifikuota standartu ISO/IEC 9899:1999). Lyginant su C, C++ teikia papildomus duomenų tipus, klases, šablonus (*templates*), išimtis (*exceptions*), vardų apibrėžimo sritis (*namespaces*), operacijų ir funkcijų perkrovimą (*operator overloading, function name overloading*), nuorodas (*references*) ir atminties valdymo operacijas, papildomas bibliotekines funkcijas ir galimybes.

STANDARTINĖ C++ KALBOS BIBLIOTEKA

Standartinė C++ kalbos biblioteka apibrėžta tarptautiniu standartu ISO/IEC 14882 (ISO WG21, 1997-1998, naujausia – ISO/IEC 14882:2017). Šis standartas tarsi užbaigė pagrindinį C++ kalbos vystymosi laikotarpį nuo pirmo kalbos aprašo ir pirmo kompiliatoriaus atsiradimo (B. Stroustrup, 1983) ir vėlesnio intensyvaus naudojimo laikotarpio, patikslino ir susistemino egzistavusias ir įvedė naujas kalbos priemones, nurodė tikslius rėmus kompiliatorių, bibliotekų ir kitų programavimo priemonių kūrėjams.

Standartinė C++ kalbos biblioteka susideda iš tokių dalių:

- Adaptuota standartinė šablonų biblioteka **STL**;
- Srautų įvedimo – išvedimo biblioteka **iostream**;
- Nacionalinių ypatumų apibrėžimo priemonės (valiutos žymėjimas, datos ir laiko formatai, realiųjų ir didelių skaičių vaizdavimas ir kt.);
- Šabloninė klasė simbolių eilučių atvaizdavimui;
- Šabloninė klasė kompleksinių skaičių atvaizdavimui;
- Konkrečios operacinės terpės aprašų (pvz., skaitmeninių reikšmių intervalų) sudarymo priemonės;
- Atminties valdymo priemonės;
- Kalbų palaikymo priemonės (pvz., įvairių kalbų pranešimų palaikymas);
- Apibrėžtos kritinių situacijų klasės;

- Standartinė C kalbos biblioteka.

Visos standartinės bibliotekos komponentės yra naujai perdarytos, atnaujintos, įvesta standartinė vardų sritis (kontekstas **std**, apibrėžiamas programoje konstrukcija **using namespace std;**), pasikeitė ir failų antraščių sistema, neliko priesagos ***.h***, pvz., vietoj **#include <iostream.h>** naudojama **#include <iostream>**

ĮVEDIMAS – IŠVEDIMAS C++ KALBOJE: SRAUTŲ BIBLIOTEKA

Jau žinome, kad C++ kalboje įvedimui ir išvedimui (angl. *Input/Output, I/O*) naudojama srautų biblioteka ***iostream.h,*** jos klasės ir atitinkamos operacijos:

- išvedimui: **cout<< išraiška ;**
- įvedimui: **cin>> išraiška ;**

C++ įvedimo – išvedimo programos pavyzdys:

```
#include <iostream.h>
main()
{
        int i;
        cout << "Įveskite sveikąjį skaičių: ";
        cin >> i;
        cout << " Įvesto skaičiaus kvadratas: " << i*i << "\n";
        return 0;
}
```

Iš tikrųjų C++ naudojamos 2 įvedimo – išvedimo bibliotekos versijos: tradicinė, apibrėžta preliminariame C++ standarte (ANSI C++), ir nauja, apibrėžta tarptautiniame ISO standarte. Naudojimo požiūriu jos analogiškos, tačiau jų realizacija yra skirtinga, pastaroji naudoja klasių šablonus.

Įvedimo ir išvedimo (I/O) srautai – tai loginiai įrenginiai, skirti duomenų gavimui (iš vartotojo) ir duomenų (rezultatų) išvedimui (vartotojui). C++ kalbos įvedimo – išvedimo sistema (iš esmės – srautinė biblioteka) sieja šiuos loginius įrenginius su tam tikrais fiziniais įrenginiais (pvz., monitoriaus ekranu, klaviatūra, pele, diskais ir t.t.).

Toks fizinių įrenginių ir loginių duomenų srautų atskirimas užtikrina vieningą, nuo aparatūros nepriklausomą tvarką informacijos įvedimui ir išvedimui.

Kai pradedama vykdyti C++ kalbos programa automatiškai sukuriami standartiniai įvedimo – išvedimo srautai, kurie pagal nutylėjimą siejami su terminalu:

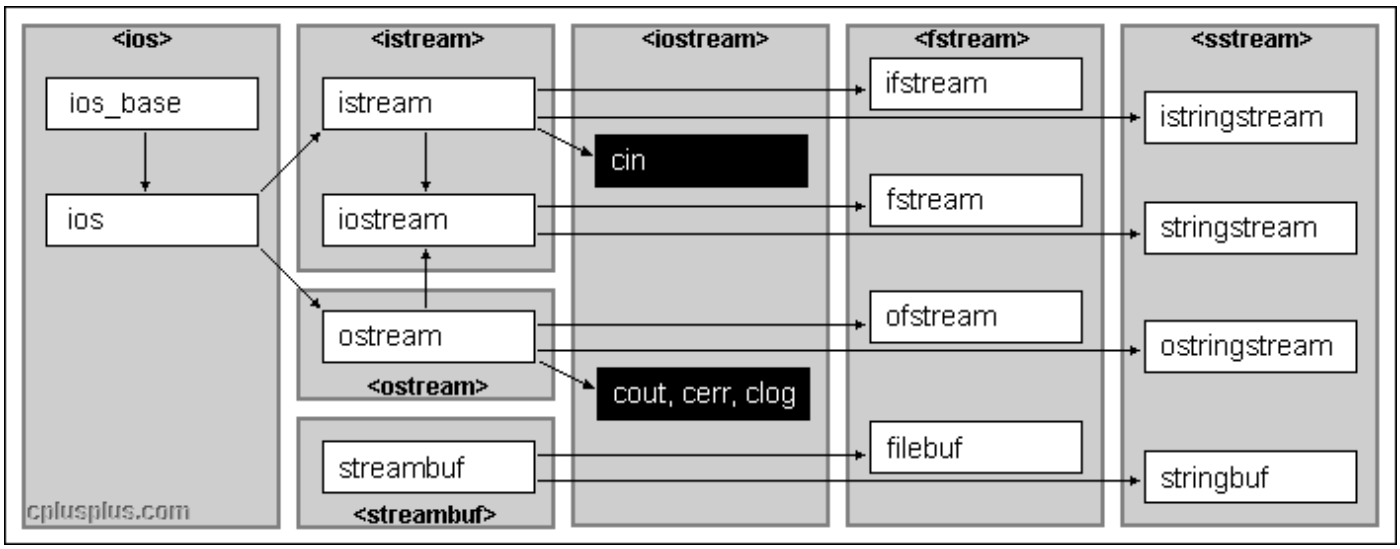
Srautas	**Standartiniai C++ kalbos srautai**	**Standartiniai C kalbos failai**
Įvedimo	cin	stdin
Išvedimo	cout	stdout
Klaidų	cerr	stderr
Registravimo	clog	stderr

Įvedimo – išvedimo (I/O) biblioteka suskirstyta į 4 failus:

- *iostream.h* – apibrėžias žemo lygio (betipių simbolių) ir aukšto lygio (tipizuotas) I/O. Čia apibrėžiamos *ios, istream, ostream,* and *iostream* klasės;
- *fstream.h* – apibrėžiamas išvestinių (iš *iostream.h* apibrėžtų) klasių rinkinys darbui su failais: klasės *ifstream, ofstream*, *fstream*;
- *strstream.h* – apibrėžiamas išvestinių (iš *iostream.h* apibrėžtų) klasių rinkinys darbui su eilutėmis (simbolių masyvais): klasės *istrstream, ostrstream, strstream;*
- *iomanip.h* – apibrėžiamas manipuliatorius (duomenų formatavimo priemonės).

Kad programuotojui būtų užtikrintas tradicinės ir naujos įvedimo – išvedimo (I/O) bibliotekų suderinamumas, naujoms srautų klasėms apibrėžti sinonimai, kurie sutampa su klasių vardais, naudojamais tradicinėje bibliotekoje, pvz., klasės šablonui **basic_ios** įvestas sinonimas **ios** ir t.t.

Klasių hierarchija C++ įvedimo – išvedimo bibliotekoje:



Standartinių srautų savybės:

Srautas	Tipas	Buferizavimas	Susietumas
cin	istream	Taip	Standartinis įvedimas (klaviatūra)
cout	ostream	Taip	Standartinis išvedimas (monitorius)
clog	ostream	Taip	Standartinių registracijos duomenų (logų) išvedimas (monitorius)
cerr	ostream	Ne	Standartinių klaidų išvedimas (monitorius)

FORMATINIS ĮVEDIMAS – IŠVEDIMAS.

C++ srautai gali būti formatuojami 3 būdais: ***formatavimo funkcijomis, žymenimis (flags) ir manipuliatoriais*.** Šios priemonės apibrėžtos aukščiausioje klasėje **basic_ios** ir todėl yra prieinamos visiems srautams.

Formatavimo funkcijos.

Funkcija **width(int)** nurodo minimalų lauko plotį reikšmei išvesti. Kai naudojama įvedime, nusako maksimalų nuskaitomų simbolių skaičių. Tai klasės **ostream** metodas.

Pavyzdys IO_1.

```
#include <iostream.h>
#include <iomanip.h>
    void main() {
        for(int i=1; i<10; i++) {
          cout.width(4);
          cout << "i=";
          cout.width(i);
          cout << i << endl; }
    }
```

Funkcija **fill(int)** leidžia nuskaityti arba nustatyti užpildymo simbolį (pvz., pakeisti norimu).

Pavyzdys IO_3.

```
#include <iomanip.h>
#include <iostream.h>
  void main() {
     cout.fill('_');
     for(int i=1; i<10; i++) {
       cout.width(4);
       cout << "i=";
```

```
            cout.width(i);
            cout << i << endl; }
        }
```

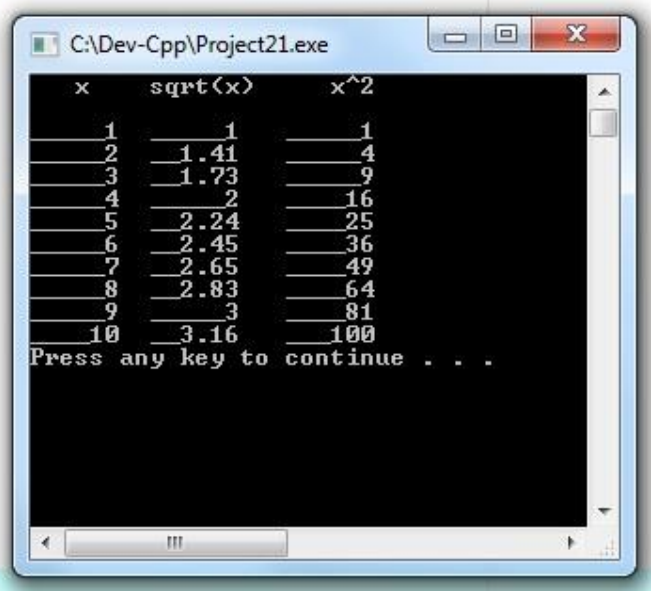
Funkcija **precision(int)** leidžia nustatyti realiųjų skaičių išvedimo tikslumą, t.y. kiek skaitmenų po kablelio bus išvesta. Pagal nutylėjimą išvedami **6** trupmenos skaitmenys. Jei nenustatyti žymenys *scientific* arba *fixed*, funkcija užduoda bendrą skaitmenų skaičių.

Pavyzdys IO_3. Funkcijų tabuliacija.

```
#include <iostream>
#include <cmath>
   using namespace std;
   void main() {
      double x;
      cout.precision(3);
      cout.fill('.');
      cout << “   x    sqrt(x)    x^2\n\n”;
      for(x = 1.0; x <= 10.0; x++) {
        cout.width(6);
        cout << x<< “  “;
        cout.width(6);
        cout << sqrt(x) << “   “;
        cout.width(6);
        cout << x*x << endl; }
 }
```

Programos vydymo pavyzdys:

```
#include <iostream>
#include <cmath>
    using namespace std;
    int main() {
        double x;
        cout.precision(3);
        cout.fill('_');
        cout << "   x    sqrt(x)     x^2\n\n";
        for(x = 1.0; x <= 10.0; x++) {
            cout.width(6);
            cout << x<< "  ";
            cout.width(6);
            cout << sqrt(x) << "   ";
            cout.width(6);
            cout << x*x << endl; }
     system("PAUSE");
    return EXIT_SUCCESS;
}
```



Žymenys (Flags). Su kiekvienu srautu yra susieti žymenys, kurie taip pat leidžia tvarkyti srauto formatą. Žymenys – tai bitų kaukės, kurios apibrėžtos bazinėje klasėje *ios_base* kaip vardinio (*enum*) tipo *fmt_flags* duomenys. Pagrindiniai žymenys:

boolalpha	Išvesti ir įvesti logines reikšmes kaip simbolių sekas „true“ ir „false“
ends	Išvesti nulinį baitą (simbolių eilutės pabaigos simbolį)
flush	Išlaisvinti buferį
unitbuf	Išlaisvinti buferį po kiekvieno įvedimo
dec	Išvesti skaičių dešimtainėje sistemoje (pagal nutylėjimą)
oct	Išvesti skaičių aštuntainėje sistemoje
hex	Išvesti skaičių šešioliktainėje sistemoje
fixed	Išvesti realaus tipo skaičių fiksuoto kablelio formoje
scientific	Išvesti realaus tipo skaičių slenkančio kablelio (eksponentinėje) formoje
showbase	Rodyti skaičiavimo sistemos pagrindą
showpoint	Rodyti tašką ir nulį po jo
showpos	Rodyti “+” prieš teigiamą skaičių
uppercase	Išvesti didžiosios „E“ (eksponentinėje) ir „X“ šešioliktainėje formoje
internal	Jei skaičius užpildo lauką, tarpai įterpiami tarp ženklo ir bazinio simbolio
left	Išvedimas išlyginamas pagal kairįjį kraštą
right	Išvedimas išlyginamas pagal dešinįjį kraštą
skipws	Nustato užpildymo simbolių praleidimą įvedime (=1)

Žymenis valdo klasės ***ios*** funkcijos **flags()**, **setf()** ir **unsetf().** Visi žymenis, išskyrus **skipws**, pagal nutylėjimą turi reikšmę 0 (nenustatyti).

Praktikoje paprastai naudojama funkcija **setf()**, leidžianti nustatyti vieno ar daugiau žymenų reikšmes, pvz.:

```
cout.setf (ios::hex | ios::uppercase)
```

Manipuliatoriai – tai specialių tipų klasės **ios** objektai, kurie apibrėžia kaip srautai **istream** ir **ostream** apdoroja pateiktus argumentus, t.y. leidžia formatuoti skaitmeninę ir simbolinę informaciją bei naudoti specialius simbolius. Manipuliatorių be parametrų pavadinimai sutampa su žymenų pavadinimais, jie ir nustato žymenis. Pagrindiniai manipuliatoriai:

endl	Išvedant pereiti į naują eilutę (įrašo '**\n**' – eilutės pabaiga)
ends	Išvesti nulinį baitą (simbolių eilutės pabaigos simbolį)
flush	Išlaisvinti buferį
dec	Išvesti skaičių dešimtainėje sistemoje (pagal nutylėjimą)
oct	Išvesti skaičių aštuntainėje sistemoje
hex	Išvesti skaičių šešioliktainėje sistemoje
left	Išvedimas išlyginamas pagal kairįjį kraštą
right	Išvedimas išlyginamas pagal dešinįjį kraštą
internal	Jei skaičius užpildo lauką, tarpai įterpiami tarp ženklo ir bazinio simbolio
fixed	Išvesti realaus tipo skaičių fiksuoto kablelio formoje
scientific	Išvesti realaus tipo skaičių slenkančio kablelio (eksponentinėje) formoje
boolalpha **noboolalpha**	Išvesti ir įvesti logines reikšmes kaip simbolių sekas „true“ ir „false“
showbase **noshowbase**	Rodyti/nerodyti skaičiavimo sistemos pagrindą
showpoint **noshowpoint**	Rodyti/nerodyti tašką ir nulį po jo
showpos **noshowpos**	Rodyti/nerodyti “+” prieš teigiamą skaičių
skipws **noskipws**	Nustato užpildymo simbolių praleidimą įvedime (=1)
unitbuf **nounitbuf**	Išlaisvinti buferį po kiekvieno įvedimo
uppercase **nouppercase**	Išvesti didžiosios „E“ (eksponentinėje) ir „X“ šešioliktainėje formoje
ws	Nustato užpildymo simbolių praleidimą įvedime
Manipuliatoriai su parametrais, apibrėžti <iomanip>	
setw(int n**)**	Nustatyti išvedimo lauko plotį – *n* simbolių
setfill(char c)	Nustatyti išvedimo lauko užpildymo simbolį, *c* – likusių iki nustatyto lauko pločio tarpų užpildymo simbolis
setprecision(int n**)**	Nustatyti skaitmenų po kablelio skaičių, išvedant trupmeninį skaičių, n – skaitmenų skaičius
setbase (int n**)**	Nustatyti išvedimų skaičių skaičiavimo sistemos pagrindą (n = 0, 2, 8, 10, 16). Pvz, **setbase(16)** = hex

setiosflags(int n**)** **resetiosflags(int** n**)**	Nustato žymenis

Visi manipuliatoriais nusakyti pakeitimai sraute galioja iki kito nurodymo, išskyrus **setw()**, kuris galioja artimiausiai išvedamai reikšmei. Jei **setw()** nurodyto lauko pločio skaičiui nepakanka, manipuliatorius ignoruojamas.

Kai manipuliatoriai naudojami su argumentais, turi būti prijungtas antraštinis failas **<iomanip>** (arba **iomanip.h).**

Manipuliatoriais paprasta naudotis, jie tiesiog nurodomi išvedimo sraute.

Pavyzdys IO_4 (analogiškas IO_3, tik vietoj formatavimo funkcijų naudojami manipuliatoriai).

```
#include <iostream>
#include <iomanip>
#include <cmath>
   using namespace std;
   void main() {
      double x;
      cout << setprecision(3);
      cout << setfill('.');
      cout << "   x16     x    sqrt(x)     x^2\n\n";
      for(x = 1.0; x <= 10.0; x++) {
        cout << setw(6) << x<< "  ";
        cout << setw(6) << sqrt(x) << "   ";
        cout << setw(6) << x*x << endl; }
}
```

FAILŲ ĮVEDIMAS – IŠVEDIMAS TAIKANT SRAUTUS.

C++ įvedimo – išvedimo (I/O) bibliotekoje failas traktuojamas kaip baitų seka. Šios sekos skaitymas ir rašymas vyksta nuosekliai. Galimas ir tiesioginis priėjimas nustatant einamosios pozicijos faile vietą.

Įvedimui į failą ir išvedimui iš failo skirtos klasės atitinkamai klasės **ifstream** ir **ofstream.** Tai išvestinės klasės iš istream ir ostream, jos aprašytos faile **fstream.h**.

Pačios įvedimo – išvedimo operacijos vykdomos taip pat kaip ir kituose srautuose naudojant operacijas >> ir <<.

Pradedant darbą su failais naudojami klasių **ifstream, ofstream** ir **fstream** konstruktoriai. Jų forma:

*ifstream (const char *name, ios::openmode = ios::in, filebuf::openprot);*

*ofstream (const char *name, ios::openmode = ios::out | ios::trunc, filebuf::openprot);*

*fstream (const char *name, ios::openmode = ios::in | ios::out, filebuf::openprot);*

Pirmas parametras, failo vardas – vienintelis būtinas, antras parametras nusako failo atidarymo režimą.

Failų atidarymo režimai.

ios::in	sukuriamas įvedimo srautas, jei failas jau egzistuoja, jis išlaikomas (nepraranda informacijos), t.y. failas atidaromas skaitymui
ios::out	sukuriamas išvedimo srautas, t.y. failas atidaromas rašymui (pagal nutylėjimą)
ios::ate	kai sukuriamas išvedimo srautas, išvedimas pradedamas nuo failo galutinės pozicijos (*at end*), rašymas vyksta į einamąją poziciją
ios::app	failas atidaromas papildymui, rašymas bet kuriuo atveju vyksta į galutinę poziciją
ios::trunc	naikinamas failo turinys, t.y., jei failas egzistuoja, jo turinys naikinamas ir failo ilgis tampa lygus 0. Vyksta pagal nutylėjimą, jei ne užduoti ios::in, ios::ate arba ios::app
ios::binary	failas atidaromas dvejetainiame režime, pagal nutylėjimą naudojami simboliniai duomenys

Prijungti failą prie srauto galima taip pat metodu **open().**

Duomenys į failą rašomi naudojant operaciją << arba metodus **write()** ir **put()**.

Skaitymas iš failo vykdomas naudojant operacija >> arba metodus **read()** ir **get()**:

write(const char* *pointer***, int** *nCount* **);**

read (char* *pointer***, int** *nCount* **);**

Metodas **eof()** skirtas failo pabaigos simboliui nustatyti: grąžina reikšmę 1, kai failo pabaiga pasiekta ir 0, kai dar nepasiekta.

Metodas **fail()**, skirtas patikrinti, ar įvedimo išvedimo operacija pasibaigė sėkmingai (rezultatas yra 0 sėkmingos operacijos atveju ir 1, jei įvyko klaida.

Failo uždarymui naudojamas metodas **close()**, kuris užbaigia rašymą/skaitymą iš buferio ir atjungia srautą nuo failo.