5. INSTRUKCIJOS PIRMINIUI PROCESORIUI

Šiame skyriuje skaitytojas susipažįsta su įterpimo *#include* ir aprašymo *#define* instrukcijų pirminiui procesoriui naudojimo tvarka.

➢ **Mokymosi tikslai**

Skaitytojas mokydamasis ir baigęs šį skyrių turi:

- žinoti įterpimo *#include* ir aprašymo *#define* instrukcijas pirminiui procesoriui.
- mokėti naudotis įterpimo instrukcija *# include*;
- mokėti naudotis aprašymo instrukcija *# define*.

5.1 Įterpimo instrukcija #include

Prieš kompiliavimą pirminis procesorius ieško programoje komandų, prasidedančių simboliu #, ir jas įvykdo. Instrukcijos pirminiui procesoriui pabaigoje kabliataškis nerašomas.

Įterpimo instrukcija # *include* nurodoma, kokių bibliotekų failų tekstai turi būti įterpti į pradinį programos tekstą. Į programos tekstą įterpiami bibliotekų failuose saugomi įvairių funkcijų aprašai. Bibliotekų failai - tai failai su plėtiniu *.h* (žr. 5.1 lentelė). Standartinių funkcijų sąrašas pateikiamas 2 Priede.

Įterpimo instrukcijos sintaksė:

#include <failo vardas>

5.1 lentelė

Bibliotekų failai	Funkcijos	Aprašymas
iostream.h	cout, cin	įvedimo, išvedimo funkcijos
string.h	strcpy()	simbolių masyvui priskiriama eilutinė konstanta
math.h	sqrt(), fabs(), sin(), pow() ir kt.	matematinės funkcijos
iomanip.h		specialiosios spausdinimo funkcijos
conio.h	clrscr()	tekstinio lango valymas
dos.h	delay	

Kompiliuojant programą, pirmiausiai įvykdomos instrukcijos pirminiui procesoriui.

1 Pratimas

```
// masyvui priskiriama reikšmė ir parodoma ekrane
#include <iostream.h>
#include <string.h>
main()
{
        char mas1[20];                  // aprašomas simbolių masyvas
        strcpy  (mas1, “Tai puiku!”);  // masyvui priskiriama reikšmė
        cout << mas1;                   // spausdinama mas1 masyvo reikšmė
return 0;
}
```

5.2 Aprašymo instrukcija #define

Instrukcija *#define* nurodoma, kokie vardai koki omis simbolių sekomis turi būti keičiami programos tekste.

Aprašymo instrukcijos sintaksė:

#define argumentas1 argumentas2

Programos tekste ieškomas pirmojo argumento ir pakeičiama antruoju.

argumentas1 - konstantos vardas.

argumentas2 - bet kokio tipo konstanta, išraiška.

Šie argumentai dažnai vadinami aprašomosiomis konstantomis.

1 Pavyzdys

#define **ilgis** 22

#define **err1** “Klaida”

Instrukcija #define dažnai naudojama konstantai priskiriant reikšmę. Prieš kompiliavimą programos tekstas peržiūrimas ir vietoje vardo įrašoma reikšmė.

2 Pratimas

```
// aprašoma ir panaudojama eilutinė konstanta
#include <iostream.h>
#define vardas “Jonas”
main()
{
        char m1[ ]=vardas;                          // simbolinio tipo masyvo aprašas
        cout << “Mano vardas “ << m1 << ‘ \n’;      // rezultatas “Mano vardas Jonas”

return 0;
}
```

3 Pratimas

```
/* instrukcija #define gali būti naudojama dažnai pasikartojančioms formulėms programoje
saugoti */
#include <iostream.h>
# define x1 ((b+c)+(b+c))
# define x2 5
main()
{
        int  b=2;
        int  c=3;
        int  e=x1;
cout    << e << “  “ << (b+c) << “  “  << x2 <<endl;
return 0;
}
```

Klausimai

1. Kokias žinote instrukcijas pirminiui procesoriui, naudojamas į programos tekstą įterpiant bibliotekų sąsajų failų vardus?

2. Kokia instrukcija pirminiui procesoriui galima aprašyti konstantas?

3. Kas vykdoma pirmiausiai: kompiliavimas ar instrukcijos pirminiui procesoriui?

4. Ar tiesa, kad ekrane bus rodoma pastaba " *Spauskite Enter*"?

```
# define        pastaba         "Spauskite Enter";
cout    << pastaba;
```

Užduotys

1. Parašykite programą, kuri rodytų ekrane Jūsų vardą. Vardo reikšmei saugoti naudokite aprašomąją konstantą. Šioje programoje nenaudokite simbolių masyvo ir nerašykite savo vardo sakinyje *cout*.
2. Parašykite programą, kurioje trys skaičiai, nuo 1 iki 3, būtų aprašomi, kaip konstantos vienas, du, trys. Sumuokite šiuos skaičius ir rezultatus spausdinkite kompiuterio ekrane.

Santrauka

Šiame skyriuje susipažinote su įterpimo *#include* ir aprašymo *#define* instrukcijų pirminiui procesoriui naudojimo tvarka.

Informacijos šaltiniai

6. ĮVESTIES IR IŠVESTIES SRAUTAI

Šiame skyriuje skaitytojas susipažįsta su standartinių įvesties ir išvesties srautų sąvokomis ir standartiniais įrenginiais. Taip pat susipažįstama su šių srautų naudojimosi taisyklėmis, bei analizuojama įvesties ir išvesties srautų formatavimo tvarka.

➢ Mokymosi tikslai

Skaitytojas mokydamasis ir baigęs šį skyrių turi:

- žinoti standartinių įvesties ir išvesties srautų sąvokas ir standartinius įrenginius;
- mokėti naudotis standartiniais duomenų įvesties ir išvesties srautais;
- žinoti duomenų srautų įvesties ir išvesties formatavimo manipuliatorius, raktus ir metodus;
- mokėti naudotis neparametrizuotais ir parametrizuotais duomenų srautų formatavimo manipuliatoriais;
- naudotis įvairiomis įvesties ir išvesties funkcijomis.

6.1 Standartiniai įvesties ir išvesties įrenginiai

Programavimo kalba *C++* duomenų įvestis ir išvestis suprantama kaip simbolių srautas. Duomenys gali būti įvedami klaviatūra, nuskaitomi iš failų, o išvedami spausdinant ekrane arba spausdintuvu, įrašant į failą, siunčiant modemais.

Aprašant įvesties ir išvesties operacijas naudojama "srauto" sąvoka. Srautas – tai abstrakcija, leidžianti atsiriboti nuo konkretaus įrenginio (disko, juostos, terminalo). Srautai būna dviejų tipų:

tekstinis srautas – įeilutes suskaidyta simbolių seka.

dvejetainis srautas – baitų seka, tiksliai atitinkanti informaciją konkrečiame įrenginyje.

C++ kalboje duomenų įvesties ir išvesties operacijomis yra srautų klasės, tačiau pradedantiesiems patogu naudotis klasikinėmis įvesties ir išvesties bibliotekomis (žr. 6.1 lentelė).

6.1 lentelė

iostream.h	**-** įvesties ir išvesties I/O (input / output) operacijomis yra srautų klasės;
stdio.h	– buferizuota įvestis ir išvestis I/O;
conio.h	- specialiosios konsolės įvesties ir išvesties I/O funkcijos.
io.h	- žemojo lygio nebuferizuota UNIX standarto įvestis ir išvestis I/O;

Kiekviena programa automatiškai atidaro 5 standartinius tekstinius srautus, kurie pateikiami 6.2 lentelėje.

6.2 lentelė

Aprašymas	Srauto vardas
išvestis (ekranas)	stdout
įvestis (klaviatūra)	stdin
išvestis (spausdintuvas)	stdprn
papildomas nuoseklus prievadas	stdaux
pranešimas apie klaidas	stderr

Dažniausiai srautai *stdin*, *stdout*, *stderr* susiejami su konsole (klaviatūra ir ekranas). Tačiau standartinį įrenginį galima paskirti DOS priemonėmis. Srautas reiškia, kad tos pačios duomenų įvedimo ir išvedimo funkcijos yra naudojamos visiems įrenginiams.

cout - siunčia duomenis į standartinį įrenginį *stdout.*

cin - siunčia duomenis iš standartinio įrenginio *stdin.*

6.2 Įvesties ir išvesties srautų klasės *cin* ir *cout*

Standartinės dvipusio ryšio įvesties ir išvesties srautų klasės, palaikančios ryšį su vartotoju (žr.6.3), aprašytos bibliotekoje *iostream.h* .

6.3 lentelė

cin	standartinis įvedimo srautas (klaviatūra)
cout	standartinis išvedimo srautas (ekranas)
cerr	nebuferizuotas klaidų išvedimas
clog	buferizuotas išvedimo srautas

Duomenų siuntimui į srautus naudojami operatoriai >> ir <<. Operatorius **>>i** reiškia, kad kintamasis i gauna duomenis i š srauto. Operatorius **<<i** reiškia, kad kintamasis i siunčia duomenis į srautą. Objektai *cout* ir *cin* aprašyti bibliotekoje ***iostream.h***.. Todėl ją būtina įterpti į programos tekstą.

Įvedimo srauto sintaksė:

cin [>> kintamojo vardas1][>>kintamojo vardas2];

cin priskiria duomenis, įvedamus klaviatūra, operatoriaus >> dešinėje pusėje parašytam kintamajam.

1 Pavyzdys

```
# include <iostream.h>
int x, y, z;
char mas1[5];
cin>>x;
cin>>y>>z;
cin>> mas1;
```

Išvedimo srauto sintaksė:

cout << duomenys [<< duomenys];

cout parodo informaciją kompiuterio ekrane.

2 Pavyzdys

```
1. cout << "lietus lyja";                  // parodo ekrane tekstą "lietus lyja"
2. cout << "Pirma \n Antra";               // parodo ekrane dvi eilutes
3. char pav[ ] = "Jonas Jonaitis";
       cout << pav;                        // parodo ekrane masyvo reikšmę
4. cout << x << y;                         // parodo kintamųjų x ir y reikšmes
```

Išvedimo sraute naudojami 3.3 lentelėje pateikiami valdymo simboliai. Spausdinant lentelę, naudojamas tabuliacijos simbolis (\t). Tabuliacijos veiksmas perkelia žymeklį į dešinę per nustatytą pozicijų skaičių.

3 Pavyzdys

```
// lentelės spausdinimas
#include <iostream.h>
main()
{
…
        cout << “X1 \t  X2 \t  X3 \n”;
        cout << x1 << ‘\t’ << x2 << ’\t’ << x3 << ’\n’;
…
return 0;
}
```

6.3 Srautų formatavimas

Standartiniuose įvesties ir išvesties srautuose galima keisti duomenų savybes. Formatuojant srautus galima nurodyti lauko dydį, tuščių laukų užpildymo būdą, duomenų vaizdavimo tikslumą. Formatavimui skiriami manipuliatoriai, metodai ir raktai. Manipuliatorių aprašai saugomi bibliotekoje **iomanip.h**. Šį failą būtina įterpti į programos tekstą įterpimo instrukcija # *include* < *iomanip.h* >.

Srautais perduodamų duomenų formatai dažniausiai nurodomi tiesiogiai sraute. Tam skiriamos specialiosios funkcijos, vadinamos parametrizuotais manipuliatoriais (žr. 6.5 lentelė), formatavimo metodais(žr. 6.6 lentelė) ir raktais (žr. 6.7 lentelė).

Duomenų formatavimo sintaksė:

išvesties srautas << manipuliatorius[…<<manipuliatorius] << srauto tąsa;

manipuliatoriai – specialiosios formatavimo funkcijos.

srauto tąsa – nurodoma formatavimo raktais ir metodais.

Manipuliatoriai būna su parametrais ir be jų. Neparametrizuoti manipuliatoriai pateikiami 6.4 lentelėje.

Neparametrizuoti manipuliatoriai

6.4 lentelė

endl	**flush**	**dec**	**hex**	**oct**	**ends**
įterpia išvedimo sraute simbolį ‘\n’ ir iškviečia manipuliatorių flush	išvedimo buferio duomenis siunčia į srautui skirtą įrenginį	rodo, kad skaičiai dešimtainiai	rodo, kad skaičiai šešioliktainiai	rodo, kad skaičiai aštuntainiai	įterpia išvedimo sraute simbolį ‘\0’

Parametrizuoti manipuliatoriai

6.5 lentelė

setw(int n)	**setfill(int n)**	**setprecision(int n)**	**setbase(int n)**
duomenims skiriamo lauko dydis	tuščių laukų užpildymo aprašymas	slankaus kablelio skaičių vaizdavimo tikslumas	skaičiavimo sistemos pagrindas (8, 10, 16)

Formatavimo raktai įjungia ir išjungia formatavimo metodus.

Formatavimo raktai

6.6 lentelė

setiosflags(metodas)	**resetiosflags(metodas)**
įjungiamas formatavimo raktas	išjungiamas formatavimo raktas

Formatavimo raktų setiosflags() ir resetiosflags() metodai

6.7 lentelė

Metodai	Paskirtis
ios::skipws	praleidžiama tuščioji sritis
ios::left	nustatoma kairioji lygiuotė
ios::right	nustatoma dešinioji lygiuotė
ios::internal	užpildoma sritis už ženklo
ios::dec	išveda dešimtainėje skaičiavimo sistemoje
ios::oct	išveda aštuntainėje skaičiavimo sistemoje
ios::hex	išveda šešioliktainėje skaičiavimo sistemoje
ios::showbase	parodo skaičiavimo sistemos pagrindą
ios::showpoint	parodo dešimtainį tašką
ios::showpos	parodo ‘+’ prieš teigiamus sveikuosius skaičius
ios::scientific	parodo skaičių eksponentine forma
ios::fixed	spausdina skaičius su fiksuotu tašku
ios:: unitbuf	nuslopina *stdout, stderr* srautų buferizaciją

4 Pavyzdys

1. cout << setw(6) << setprecision(2) << 134.568767; // rezultatas 134.57
2. cout << setw(20) << “Sveiki\n”;
   cout << setw(20) << setiosflags(ios::left) << ”Sveiki \n”;
   Rezultatas:
   Sveiki
Sveiki

6.4 Įvesties ir išvesties funkcijos *get()* ir *put()*

C++ kalboje ryšys su įvedimo ir išvedimo srautais palaikomas ir funkcijomis get () ir put (). Šios funkcijos įrašo į srautą ir skaito iš srauto visus simbolius, įskaitant tarpus ir valdymo simbolius.

get() - įveda po vieną simbolį iš bet kurio standartinio įrenginio (automatiškai iš klaviatūros).

put() - išveda po vieną simbolį į bet kurį standartinį įrenginį (automatiškai į ekraną).

Kreipiantis į funkcijas get() ir put() naudojami sudėtiniai vardai.

Kreipinio sintaksė:

srauto vardas. funkcijos vardas (parametrai);

Funkcija *get()* atlieka buferizuotą duomenų įvedimą. Įvedami duomenys patenka į atminties buferį, o tik po to, nuspaudus klavišą "*Enter*", į operatyviąją atmintį. Naudojantis funkcija *get(),* įvedamus duomenis galima koreguoti iki bus nuspaustas klavišas " *Enter*".

Naudojantis *get(),* galima kurti savo funkcijas, kuriose nėra nurodyto apribojimo, t.y. tarpas nebus įvedimo pabaigos požymis.

5 Pavyzdys

```
// Programa užpildo masyvą simbolių eilute, sudaryta iš žodžių ir tarpų
#include <iostream.h>
#define MAX 25 // simbolinio masyvo elementų skaičius
void iv(char str[],int l)
{
        int i=0;                        // indeksas
        char in;                        // įvestas simbolis
        cin.get(in);                    // įvedamas naujas simbolis
        while(i<(l-1)&&(in!='\n'))
        {
                str[i]=in;              //eilutė įvedama po vieną simbolį
                i++;
                cin.get(in);            //naujo simbolio įvedimas
        }
str[i]='\0';                            //simbolių eilutės pabaigos požymis
return;
}
int main(void)
{
        char ived[MAX];                 //saugoma įvesta informacija
        cout << "Jūsų pilnas vardas ir pavardė\n";
        iv(ived, MAX);                  //klaviatūra įvesta eilutė
        cout << "Jūsų pilnas vardas ir pavardė yra   " << ived <<'\n';
return 0;
}
```

6.5 Funkcijos *getch()* ir *putch()*

Funkcijos *getch()* ir *putch()* atlieka nebuferizuotą įvedimą ir išvedimą. Duomenys perduodami iš karto, kai jie įvesti. Simboliai, įvedami naudojantis funkcija *getch(),* ekrane nerodomi. Norint matyti įvedamus simbolius ekrane, reikia naudotis funkcija *getche()*. Įvedant simbolius su *getch()* funkcija, klavišo "*Enter*" spausti nereikia.

Funkcijos *getch()* ir *putch()* aprašytos bibliotekoje **conio.h**

6 Pavyzdys

```
#include <fstream.h>
#include <conio.h>
main()
{
int s,s1;                       //ciklo valdymo kintamasis
char raid[5];                   // penkiems įvedamiems simboliams
cout << "Įveskite penkias raides";
for (s=0; s<5; s++)
{
        raid[s]=getch();                // simbolis priskiriamas elementui
        ofstream prn("PRN");
        for (s1=0; s1<5; s1++)
        {
        prn.put(raid[s1]); // masyvas spausdinamas
        }
}
return 0;
}
```

🕮 Klausimai

1. Kuo skiriasi sarutai *cout* ir *cin*?
2. Kaip vienu cin sakiniu įvedamos reikšmės keliems kintamiesiems?
3. Priskiriant kintamiesiems reikšmes, galima naudoti priskyrimo sakinį ir srautą *cin*. Koks skirtumas tarp šių metodų?
4. Koks bus šio sakinio rezultatas?

   cout << setw(8) << setprecision (3) << 123.456789;
5. Kuo skiriasi funkcijos *get()* ir *getch()*?

✎ Užduotys

1. Parašyti programą, kuri "mokytų" vaikus daugybos lentelės. Įvesti du skaičius. Paprašyti įvesti atsakymą. Ekrane parodyti teisingą atsakymą.
2. Parašyti programą, kuri įvestų dviejų studentų egzamino balus bei skaičiuotų balų vidurkį. Programos pabaigoje spausdinti gautą balų vidurkį. 3. Parašyti programą, kuri įvestų Jūsų amžių, ūgį ir svorį. Įvestas reikšmes su atitinkamais komentarais rodyti kompiuterio ekrane.

⌛ Santrauka

Šiame skyriuje susipažinote su įvesties ir išvesties srautų sąvokomis ir standartiniais įrenginiais. Taip pat susipažįstama su šių srautų naudojimosi taisyklėmis, bei analizuojama įvesties ir išvesties srautų formatavimo tvarka.

ⓘ Informacijos šaltiniai