# Duomenų (kintamųjų reikšmių) įvedimo sakiniai

Paprasčiausias duomenų pateikimo vykdomai programai būdas yra jų įvedimas klaviatūra.

Tokių duomenų skaitymas programoje aprašomas sakiniu:

```
cin >> a >> b >> c;
```

Įvedimo sakinys sustabdo programos darbą. Laukiama kol bus įvesti duomenys kompiuterio klaviatūra. Tarkime , surenkame skaičius **4, 5, 6** atskirdami juos tarpo simboliu arba įvedus kiekvieną skaičių spaudžiame klavišą ENTER. Kompiuteris į savo atmintį įsirašo: a = 4, b = 5, c = 6.

Kad galėtume programoje naudoti duomenų įvedimo sakinius, naudojamų bibliotekų aprašų srityje reikia įtraukti failą **#include <iostream>**

# Rezultatų rašymo (spausdinimo) sakiniai

Norint spaudinti programos darbo rezultatus kompiuterio ekrane, naudojami rašymo (spausdinimo) sakiniai. Pvz.:

```
cout << a << b << ... << n;    cout << "Labas";
```

Norint spausdinti kintamojo reikšmę, rašome tik kintamojo vardą, o jei reikia spausdinti tekstą (simbolių eilutę), tuomet šis tekstas sakinyje rašomas kabutėse. Atskiri spausdinami komponentai atskiriami simboliu **<<.** Pvz.:

```
cout << "X1 = " <<  x1 << " X2 = " << 5;
```

Jei kintamojo x1 reikšmė yra 7, ekrane matysime:

```
X1 = 7 X2 = 5
```

Kad galėtume programoje naudoti duomenų rašymo (spausdinimo) sakinius, naudojamų bibliotekų aprašų srityje reikia įtraukti failą

```
#include <iostream>
```

# Specialių simbolių panaudojimas išvedimui programuoti

Tam, kad perkelti žymeklį į kitos eilutės pradžią, galima įterpti naujos eilutės simbolį **(\n)** į išeinantį srautą. C++ kalboje galimi du naujos eilutės formavimo būdai. Pirma, galima patalpinti **\n** simbolį išeinančio srauto viduje. Programa TWOLINES.CPP vaizdą pateikia dviejų eilučių pavidalu naudojant naujos eilutės simbolį:

```
#include <iostream>
int main() {
 setlocale (LC_ALL,"Lithuanian");
 cout << "Ši eilutė pirma\nŠi eilutė antra";
 return 0;
}
```

Kai sukompiliuosite ir paleisite šią programą, naujos eilutės simbolis leis pateikti vaizdą dviejų eilučių pavidalu, kaip parodyta žemiau:

```
Ši eilutė pirma
Ši eilutė antra
```

Programa NEWLINES.CPP išveda skaičius 1,0,0,1 skirtingose eilutėse:

```
#include <iostream>
int main() {
 cout << 1 << '\n' << 0 << '\n' << 0 << '\n' << 1;
 return 0; }
```

```
1
0
0
1
```

Žymeklio perstūmimui į sekančią eilutę galima panaudoti simbolį **endl** (eilutės pabaiga). Programa iliustruoja **endl** panaudojimą žymeklio perstūmimui į naujos eilutės pradžią:

```
#include <iostream>
int main {
 setlocale (LC_ALL,"Lithuanian");
 cout << "O dabar…" << endl;
 << "Mokomės programuoti C++ kalba";
return 0; }
```

Ekrane vaizdas bus pateiktas dviejų eilučių pavidalu.

```
O dabar
Mokomės programuoti C++ kalba
```

# Išvedimo formato valdymas

Kad teisingai išvedant išdėstyti skaičius, galima įterpti tarpo (space) simbolius. Bet tai galima padaryti ir su modifikatoriumi **setw** (pločio nustatymas). Jo pagalba nurodomas minimalus simbolių kiekis, kurį skaičius užims. Programa panaudoja setw pločio parinkimui. Tam papildomai reikia prijungti failą **iomanip**:

```
#include <iostream>
#include <iomanip>
int main(){
setlocale (LC_ALL,"Lithuanian");
cout << "Mano skaičius lygus" << setw(3) << 1001 << end1;
cout << "Mano skaičius lygus" << setw(4) << 1001 << end1;
cout << "Mano skaičius lygus" << setw(5) << 1001 << end1;
cout << "Mano skaičius lygus" << setw(6) << 1001 << end1;
return 0; }
```

Kai sukompiliuosite ir paleisite šitą programą, ekrane pamatysite:

```
Mano skaičius lygus 1001
Mano skaičius lygus 1001
Mano skaičius lygus _1001
Mano skaičius lygus __1001
```

Su **setw** nurodomas minimalus pozicijų kiekis, kurį skaičius užims.

Prieš tai esančioje programoje modifikatorius **setw(3)** nurodė tris simbolius. Tačiau skaičiui 1001 reikia daugiau simbolių, todėl **cout** panaudojo realiai reikalingą simbolių kiekį, kuris duotu atveju yra keturi. Verta paminėti, kad naudojant **setw** pločio parinkimui, nurodytas plotis tinka tik vieno skaičiaus išvedimui. Jei jums reikia nurodyti keleto skaičių plotį, jūs turite panaudoti **setw** keletą kartų.

**setw (int n) – nustato išvedamų duomenų lauko plotį n.**

**setprecision (int n)** – nustato realiojo skaičiaus išvedamų skaitmenų kiekį.

Jeigu skaičiau sveikojoje dalyje yra daugiau skaitmenų, tuomet jis vaizduojamas standartine išraiška.

```
int main(){
setlocale (LC_ALL,"Lithuanian");
double f =35687.14159;
cout << setprecision(4) << f << '\n';
cout << setprecision(2) << f << '\n';
return 0;}
```

```
3.569e+004
3.6e+004
```

Jeigu prieš manipuliatorių **setprecision** rašomas **fixed ,** tuomet n nustato kiek trupmeninės dalies skaitmenų reikia išvesti

```
45
  _45
78.56000
x1 =_78.56
```

```
// spausdinimo pavyzdys
#include <iostream>
#include <iomanip>
int main(){
setlocale (LC_ALL,"Lithuanian");
int a = 45;
double b = 78.56;
cout << a << endl;
cout << setw (4) << a << endl;
cout  <<  fixed  <<  setprecision(5)  <<  b
  << endl;
cout  << "x1 =" << setw (6) << fixed <<
setprecision (2) << b << endl;
return 0;}
```

```
3.142
3.14159
3.14
3.141590000
```

```
// setprecision pavyzdys
#include <iostream>
#include <iomanip>
int main () {
setlocale (LC_ALL,"Lithuanian");
double f =3.14159;
cout << setprecision(4) << f << '\n';
cout << setprecision(9) << f << '\n';
cout << fixed;
cout << setprecision(2) << f << '\n';
cout << setprecision(9) << f << '\n';
return 0; }
```