4. MASYVAI

Masyvas – vidinis C++ konteineris to paties tipo duomenų grupavimui: duomenų seka, turinti vieną vardą ir daug reikšmių, iš kurių norima atrenkama kreipiantis sekos vardu ir nurodant sekos nario indeksų reikšmes. Iš pradžių nagrinėsime vienmačius masyvus, kurių elementas atrenkamas vienu indeksu. Tokio masyvo atitikmuo matematikoje – vektorius. Masyvai gali būti ir daugiamačiai. C++ kalboje jie kuriami tokiu būdu: dvimatis masyvas yra faktiškai vienmatis masyvas, kurio kiekvienas elementas savo ruožtu yra kitas vienmatis masyvas; trimatis masyvas – vienmatis masyvas, kurio kiekvienas elementas yra kitas vienmatis masyvas, o šio kiekvienas elementas yra irgi vienmatis masyvas, ir t.t. Tuo būdu, C++ dvimatis masyvas gali būti bendresnės formos nei matematikos matrica: kiekviena jo eilutė gali būti skirtingo ilgio (tačiau mes tokių masyvų nenagrinėsime).

Masyvuose galima talpinti tiek vidinių kalbos tipų, tiek ir vartotojo apibrėžtų tipų (struktūrų ar klasių) duomenis.

Didelis C++ masyvų skirtumas nuo mums įprastų matematinių atitikmenų – masyvų indeksai pradedami skaičiuoti nuo nulio.

Masyvų gyvavimo ciklas yra toks pat, kaip ir skaliarinio kintamojo: skelbimas ir apibrėžimas (nurodomas formatas, skiriama atmintis), inicializavimas (suformuojamos reikšmės), darbas su masyvu, masyvo sunaikinimas (išeinant iš matomumo srities, kurioje masyvas paskelbtas ar baigiantis programai, išlaisvinama masyvo užimta atmintis).

4.1 VIENMAČIAI MASYVAI

Skelbimas ir apibrėžimas:

formatas masyvoVardas [matas];

Pavyzdžiui, vienmačiui masyvui x skiriamos 5 4 baitų ilgio ląstelės, o y – 10 tokių pat ląstelių:

int x[5], y[10];

Elementų reikšmės kol kas lieka neapibrėžtos. Masyvų inicializavimą galima sutapdinti su skelbimu ir apibrėžimu taip:

int x[5] = { 10, 5, 3, 7, 0 }; // elementų skaičių „5“ nurodyti nebūtina

int y[ ] = { 10, 5, 3, 7, 0, 10, -1, -1, 0, 0 }; // y bus skirtos 10 ląstelių: tiek yra reikšmių

int z[100] = { 0 }; // visi 100 elementų įgis nulines reikšmes

Jei inicializuojant matai buvo nurodyti, o reikšmių pateikta mažiau nei nurodyta – likę reikšmės bus nulinės; jei reikšmių per daug – bus deklaruota kompiliavimo klaida. Vis tik panašaus inicializavimo savo programose vengsime (išskyrus trečiąjį pavyzdį), kadangi šiuo atveju duomenų

reikšmės programoje perteikiamos literalų pavidalu, ir norint šia programa skaičiuoti kitą panašų uždavinį su kitokiomis masyvų elementų reikšmėmis, tektų keisti programos tekstą. Bendresnis būdas inicializuoti yra įvesti masyvų reikšmes iš klaviatūros ar failų išorinėje kompiuterio atmintyje. Toks inicializavimo būdas taikomas ir žemiau pateiktuose programų pavyzdžiuose.

1 pavyzdys: reikia suskaičiuoti realių skaičių vienmačio masyvo elementų sumą. Elementų masyve gali būti iki 100, o duomenis turime įvesti iš klaviatūros.

```
#include <iostream>
using namespace std;
//
int main( ){
                    // Duomenys
          double x[ 100 ]; // masyvo skelbimas ir apibrėžimas 100-ui ląstelių
          int n;          // masyvo elementų tikrasis skaičius
                  // Rezultatas: elementų suma
          double s = 0.;
                 /*   Duomenų įvedimas, kontrolinis spausdinimas ir skaičiavimas –
                      tame pat cikle*/
          //
          cout<<"Iveskite masyvo elementu skaiciu\n";
          cin>>n;
          cout<<"Masyvo elementu skaicius yra  "<<n<<endl<<endl;
          cout<<"Iveskite masyvo elementus:\n";
          for( int i = 0; i < n; i++ ){
                          cout<<i+1<<" -aji masyvo elementa  ";
                          cin>>x[ i ];
                          cout<<"Ivestas "<<i+1<<"  elementas "<<x[ i ]<<endl;
                          s += x[ i ];
          }
              // Rezultatų spausdinimas
          cout<<endl<<"Rezultatas: elementu suma yra "<<s<<endl;
          //
          system( "pause" );
          return 0;
}
```

Programos spausdiniai, jei įvesime nurodytus duomenis (jie išspausdinti tiesiai ir paryškintai), atrodys taip:

Iveskite masyvo elementu skaiciu
5
Masyvo elementu skaicius yra 5

Iveskite masyvo elementus:
1-aji masyvo elementa **10**

```
Ivestas 1 elementas 10
2-aji masyvo elementa 20
Ivestas 2 elementas 20
3-aji masyvo elementa 0
Ivestas 3 elementas 0
4-aji masyvo elementa -10
Ivestas 4 elementas -10
5-aji masyvo elementa -20
Ivestas 5 elementas -20

Rezultatas: elementu suma yra 0
Press any key to continue
```

2 pavyzdys. Truputį pasunkinkime 1-ojo pavyzdžio užduotį: reikia suskaičiuoti realių skaičių vienmačio masyvo teigiamų elementų vidurkį. Elementų masyve gali būti iki 100, o duomenis turime įvesti iš klaviatūros. Tokių išsamių komentarų kaip 1-ame pavyzdyje neberašysime.

```
#include <iostream>
using namespace std;
//
int main( ){
                    //    Duomenys
          double x[ 100 ];
          int n;
          int k  = 0;  //  teigiamų elementų kiekis
          double s = 0.;
          //
          cout<<"Iveskite masyvo elementu skaiciu\n";
          cin>>n;
          cout<<"Masyvo elementu skaicius yra  "<<n<<endl<<endl;
          cout<<"Iveskite masyvo elementus:\n";
          for( int i = 0; i < n; i++ ){
                              cout<<i+1<<" -aji masyvo elementa  ";
                              cin>>x[ i ];
                              cout<<"Ivestas "<<i+1<<"  elementas "<<x[ i ]<<endl;
                              if( x[i] > 0. ) {
                                   k++;
        s += x[ i ];
                              }
           }
               // Vidurkio skaičiavimas ir spausdinimas
          cout<<"\nRezultatai\n";
          if( k>0 )  cout<<"Teigiamu elementu vidurkis yra  "<<s/k<<endl;
          else cout<<"Teigiamu elementu masyve nera\n";
          //
          system( "pause" );
          return 0;
```

```
}
```

3 pavyzdys. Šiuo uždaviniu parodysim, kaip reikia išrinkti didžiausias ar mažiausias duomenų sekos vertes. Tai – vadinamieji minimakso uždaviniai. Sakykim, turime sveikųjų skaičių masyvą, kuriame gali būti iki 100 elementų. Reikia surasti didžiausią ir mažiausią masyvo elementus bei jų vietas masyve. Jei masyve būtų keli vienodi didžiausi ar mažiausi elementai, rezultatais laikyti pirmąsias iš pasikartojančių reikšmių.

Tokie uždaviniai sprendžiami taip: daroma prielaida, kad didžiausias (mažiausias) elementas yra kažkuris masyvo elementas (pavyzdžiui, pirmasis), vėliau po vieną perrenkami visi likę masyvo elementai ir žiūrima, ar prielaida yra teisinga, ar ne. Jei neteisinga – keičiame prielaidos reikšmę, jei teisinga – jokių papildomų veiksmų neatliekame. Kad logika būtų aiškesnė, duomenų įvedimą ir kontrolinį spausdinimą atliksime viename cikle, o visą paiešką – kitame.

```
#include <iostream>
using namespace std;
//
int main( ){
                    //    Duomenys, įvedimas
        int x[ 100 ];
        int n;
        cout<<"Iveskite masyvo elementu skaiciu\n";
        cin>>n;
        cout<<"Masyvo elementu skaicius yra  "<<n<<endl<<endl;
        cout<<"Iveskite masyvo elementus:\n";
        for( int i = 0; i < n; i++ ){
                        cout<<i+1<<" -aji masyvo elementa  ";
                        cin>>x[ i ];
                        cout<<"Ivestas "<<i+1<<"  elementas "<<x[ i ]<<endl;
        }
            // Paieška
        int max = x[0];  // prielaida didžiausiam elementui
        int imax = 0;    // jei pirmasis elementas toks ir bus – jo vieta yra 0
        int min = x[0];  // prielaidos mažiausiam elementui ...
        int imin = 0;    // ir jo vietai masyve
        for( int i = 1; i < n; i++ ){   // ciklą dabar galima pradėti nuo antrojo elemento
                        if( x[i] > max ) {
                                        max = x[i];
                                        imax = i;
                        }
                        if( x[i] < min ) {
                                        min = x[i];
                                        imin = i;
                        }
        }
              // Rezultatų spausdinimas
        cout<<"\nRezultatai\n";
        cout<<"Didziausias elementas yra "<<max<<", jo vieta yra "<<imax+1<<endl;
```

```
        cout<<"Maziausias elementas yra "<<min<<", jo vieta yra "<<imin+1<<endl;
        //
        system( "pause" );
        return 0;
}
```

4.2 DAUGIAMAČIAI MASYVAI

Skelbimas ir apibrėžimas:

formatas masyvoVardas [1-asis matas][2-asis matas][...];

Pavyzdžiui, dvimačiui masyvui *x* skiriama 20 4 baitų ilgio ląstelių, o trimačiam *y* – 60 tokių pat ląstelių:

int x[5][4], y[5][4][3];

Kaip minėta, C++ faktiškai yra tik vienmačiai masyvai, kurių atskiri elementai gali būti kiti masyvai. Todėl kompiuterio atmintyje daugiamačių masyvų elementai išdėstomi kaip vienmatė duomenų seka tokia tvarka: dešinysis masyvo indeksas kinta pirmiausia, po to – antrasis indeksas nuo galo, ir t.t. Taigi, dvimačio masyvo *x[3][3]* elementai kompiuterio atmintyje būtų saugomi tokia tvarka:

x[0][0] x[0][1] x[0][2] x[1][0] x[1][1] x[1][2] x[2][0] x[2][1] x[2][2]

Daugiamačių masyvų, panašiai kaip ir vienmačių, inicializavimą galima sutapdinti su skelbimu ir apibrėžimu:

int x[2][3] = { { 10, 5, 3 },   { -1, 1, 2 }   }; // 10, 5, 3 – pirmoji eilutė, -1, 1, 2 - antroji

int y[2][3] = { { 10, 5   },   { -1   }   }; // 10, 5, 0 – pirmoji eilutė, -1, 0, 0 - antroji

int z[10][10] = { 0 };   // visi 100 elementų įgis nulines reikšmes

Kaip ir vienmačių masyvų atveju, savo programose dažniausiai rinksimės kitokį masyvų inicializavimo būdą – masyvo elementų įvedimą iš klaviatūros ar failų išorinėje kompiuterio atmintyje. Masyvo elementų perrinkimui dabar reikės jau kelių vienas į kitą įdėtų ciklų. Be to, kad masyvo elementai būtų žmogui patogesniu pavidalu atspausdinti (pavyzdžiui, kontrolinis įvestų duomenų spausdinimas), galima išvesties srautą formatuoti.

Dabar paminėsime tik kelias formatavimo galimybes: vadinamaisiais manipuliatoriais arba formatavimo vėliavėlėmis su numatytomis formatavimo konstantų vertėmis. Naudojant programoje manipuliatorius ir vėliavėles, būtina prijungti antraštinį failą *iomanip*. Vėliavėlės priklauso įvesties-išvesties konstantų klasei *ios*, todėl prieš vėliavėles ją būtina priklausomybės operacija *::* nurodyti. Keli manipuliatoriai:

setw(lauko ilgis) – įvesties ar išvesties lauko duomeniui išvesti ilgis pozicijomis

setprecision(skaitmenų skaičius) – realių duomenų trupmeninės dalies išvedimo tikslumas skaitmenų skaičiumi

setiosflags(vėliavėlė) – vėliavėlės nustatymo manipuliatorius; vėliavėlė veiks visą tolesnį duomenų srautą

resetiosflags – visų vėliavėlių atšaukimo manipuliatorius

Kelios vėliavėlės:

fixed – aritmetinius duomenis išvesti įprastu, ne eksponentiniu pavidalu

showpoint – realiuose aritmetiniuose duomenyse, net esant nulinei trupmeninei daliai, rodyti dešimtainį tašką.

4 pavyzdys. Suskaičiuosime dvimačio realių skaičių masyvo, turinčio iki 10 eilučių ir iki 10 stulpelių, kiekvienos eilutės elementų vidurkius. Vidurkius patalpinsime atskirame vienmačiame masyve. Kad masyvo kontrolinis spausdinimas būtų tvarkingesnis, panaudosime kai kurias formatavimo galimybes ir masyvą spausdinsime atskirame cikle.

```
#include <iostream>
#include <iomanip>     // formatavimo manipuliatoriams
using namespace std;
//
int main( ){
        double x[ 10 ][ 10 ];    // pradinis masyvas
        int n, m;            // tikrieji eilučių ir stulpelių skaičiai
        double v[10];        // rezultatams – eilučių vidurkiams
        //
  // Duomenų įvedimas ir kontrolė
        cout<<"Iveskite masyvo matus\n";
        cin>>n>>m;
        cout<<"Masyvo matai yra "<<setw( 5 )<<n<<setw( 5 )<<m<<endl;
        cout<<"Iveskite masyvo elementus eilutemis:\n";
        for( int i = 0; i < n; i++ ){
                for( int j = 0; j < m; j++ ){
                                        cout<<"Iveskite elementa "<<i+1<<"  "<<j+1<<" : ";
                                        cin>>x[ i ][ j ];
                }
        }
        cout<<"\n          Masyvas\n";
        for( int i = 0; i < n; i++ ){
          cout<<"    Eilute  "<<i+1<<endl;
          for( int j = 0; j < m; j++ )
                        cout<<setiosflags( ios::fixed )<<setiosflags( ios::showpoint )
                            <<setprecision( 2 )<<setw( 12 )<< x[i][j];
```

```
            cout<<endl;   // t.y. kitą masyvo eilutę spausdinsime į kitą išvesties eilutę
        }
                        // Skaičiavimas
        for(  int i = 0; i < n; i++ ){
            v[i] =0.;  // i-osios eilutės elementų suma
            for( int j = 0; j < m; j++ )   v[i] += x[ i ][ j ];
            v[i] /= m;
        }
                        // Rezultatų išvedimas
        cout<<"\n      Rezultatai: eiluciu vidurkiai\n";
        for(  int i = 0; i < n; i++ )
                        cout<<"Eilutes   "<<i+1<<"  vidurkis yra"<<
                                setiosflags( ios::fixed )<<setiosflags( ios::showpoint )<<
                                setprecision( 2 )<<setw( 12 )<<v[i]<<endl;
        //
        system( "pause" );
        return 0;
}
```