C++ programavimo kalba

Standartinė šablonų biblioteka (STL)
Objektiškai orientuotas projektavimas (OOD)

(13 paskaita)

Asociatyviniai konteineriai

Nuosekliųjų konteinerių trūkumas – lėta paieška, einant nuo vieno elemento prie kito.

Asociatyviniuose konteineiriuose elementai nėra rikiuojami eilėje, bet siejami su raktais, kurie atlieka indekso funkcijas. Raktais dažniausiai būna skaičiai arba eilutės. Pagrindinis asociatyvinių konteinerių privalumas – l.greita paieška.

STL apibrėžia dvi pagrindines asociatyvinių konteinerių grupes:

- set (aibė)
- map (žemėlapis)

Aibėse saugomi unikalūs raktai, žemėlapiuose – raktų ir reikšmių poros.

Aibėse ir žemėlapiuose raktai unikalūs, tačiau multiaibėse bei multižemėlapiuose raktai gali kartotis.

Norint naudoti asociatyvinius konteinerius, reikia prisijungti antraštės failus map.h arba set.h

Aibės (1)

Aibės dažnai naudojamos vartotojo sukurtų objektų saugojimui. Pvz. darbuotojų duomenų bazė. Tačiau aibės gali saugoti ir paprastesnius elementus – eilutes.

```
#include <iostream>
#include <set>
#include <string>

void main()
{ string names [ ] = {“Jonas”, Petras”, “Ona”, “Antanas”};   // string objektų masyvas
   set<string, less<string> >  nameSet(names, names+4);  // aibė į masyvą
   set<string, less<string> >::iterator iter;                    // iteratorius

   nameSet.insert(“Jurgis”);                                    // įterpiamas naujas vardas
   nameSet.insert(“Mikas”);
   nameSet.erase(“Mikas”);                                      // šalinamas elementas
   cout<<“Size”<<nameSet.size()<<endl;

   iter = nameSet.begin();
   while (iter != nameSet.end() )
      cout<<*iter++<<“\n”;
```

Aibės (2)

```
String searchName;
cout << Iveskite ieskoma varda: “;
cin >> searchName;

iter = nameSet.find(searchName);
if ( iter == nameSet.end() )
   cout << “ Vardas “<< searchName << “nerastas” << endl;
else
   cout << “Vardas “<<*iter << “rastas…”<<endl;
}
```

Šiame pavyzdyje vykdoma sukuriama aibė, kurią sudaro eilutės. Vykdoma elemento paieška, aibės papildymas ir elemento šalinimas.

Žemėlapiai (map)

Žemėlapiai – tai poros, kurias sudaro *rakto objektas* ir *reikšmės objektas*. Raktas – tai indeksas, pagal kurį randama reikšmė. Raktais dažniausiai būna skaičiai arba eilutės, bet tai gali būti ir objektai arba konteineriai. Žemėlapiai kartais naudojami kaip asociatyviniai masyvai.

```
#include <iostream>
#include <map>
#include <string>

void main()
{ string name;  int pop;
  string states[ ] = {“Montana”, “Arizona”, “Nevada”, “Colorado”, “Florida”};
  int pops[ ] = {470, 280, 787, 985, 562};
  map <string, int, less<string> > mapStates;          //map
  map <string, int, less<string> > :: iterator iter;   //iterator


 // string – eilute(raktas), int - reikšmė,  less – išrūšiavimas mažėjimo tvarka
```

Žemėlapiai (map) tęsinys

```
for  (int j = 0; j<6; j++)
{ name = states[ j ];
  pop = pops[ j ];
  mapstates [name] = pop;
}
  cout << “Įveskite valstiją: “;   cin >> name;
  cout << “Gyventojų skaičius “<<mapStates[name]<<endl;

  for (iter = mapStates.begin();  iter != mapStates.end();  iter++ )
      cout << (*iter).first <<‘  ‘ <<(*iter).second <<endl;

}
```

(*iter).first - išveda raktą,

(*iter).second - reikšmę

Funkcijos objektai

Funkcijos objektai – tai funkcijos, kurios iterpiamos į klases taip, kad atrodo, kaip objektai. Tokia klasė neturi duomenų ir ją sudaro tik vienas metodas, kuris yra perkrautas operatorius. Tokia klasė yra šabloninė, todėl gali dirbti su skirtingais duomenų tipais.

Funkcijų objektai plačiai naudojami STL, kaip algoritmų argumentai. Jie leidžia modifikuoti algoritmų veikimą.

Funkcija objektas	Grąžinama reikšmė
T=plus(T, T)	x+y
T=minus(T, T)	x-y
T=times(T, T)	x*y
T=divide(T, T)	x/y
bool = equal_to(T, T)	x = = y
bool = greater(T, T)	x>y
bool = less(T, T)	x<y
bool = greater_equal(T, T)	x >=y
bool = less_equal(T, T)	x<=y

Pavyzdys

```
#include <iostream>
#include <list>
#include <numeric>                    // del funkcijos accumulate()
class airtime
{private:  int hours;
           int minutes;
 public:   airtime(): hours (0), minutes (0)
           {  }
           airtime (int h, int min): hours (h), minutes (min)
           {  }
           void display ( ) const
           { cout << hours << " : " <<minutes ; }
           void get( )
           { char dummy;
             cout<< Iveskite laika (12:23): ";
             cin >> hours >> dummy>>minutes;
            }
```

Pavyzdys (tęsinys)

```
airtime operator + (const airtime right) const        // perktrautas operatorius
{ int n;
  int tmph = hours + right.hours;
  int tmpmin = minutes + right.minutes
  if ((n=tmpmin/60) > 0)
       {tmph=tmph+n; tmpmin -= 60*n; }
  return airtime(tmph, tmpmin);
}
bool operator = = (const airtime& at2) const
{return (hours = = at2.hours)&&(minutes = = at2.minutes) }
};

void main ()
{char answer;
 airtime temp, sum;
 list <airtime> airlist;            // sąrašas padarytas iš airtime objektų
```

Pavyzdys (tęsinys)

```
do {
      temp.get ();
      airlist.push_back (temp);          // įterpiamas objektas temp sąrašo gale
      cout >> " Dar įvedinėsite (y/n)? " ;
      cin >> answer;
      } while (answer != 'n');

sum = accumulate (airlist.begin(), airlist.end(), airtime(0,0), plus<airtime>() );
sum.display();
cout << endl; }
```

Funkcija ***accumulate()*** sudeda sąrašo elementus pradedant nuo pirmo ir baigiant paskutiniuoju.

Sudėčiai naudojama funkcija objektas ***plus<airtime>,*** sumavimas atliekamas į airtime(0,0) objektą.

Objektiškai orientuotas projektavimas

OOD (object oriented design) – tai programuojamos problemos modelio sukūrimas, naudojant klases bei jų savybes.

OOD sudaro trys fazes:

1. CRC card (CRC kortelė)
2. Use cases (Panaudojimo stadija)
3. Class diagrams (Klasių diagramos)

CRC cards – tai problemos analizė netechniniu požiūriu. Tai paprastai atlieka business domain experts (BDEs), kurie žino kaip programa turi veikti. Kiekviena kortelė atitinka tam tikrą objektą, kuris bus naudojamas programoje.

Use case – tai programos specifinių operacijų aprašymas. Šitoje stadijoje gali išaiškėti, kad CRC cards turi būti modifikuotos ar sukurtos naujos papildomos.

Class diagrams – tai stadija, kai nustatomas ryšys tarp planuojamų sukurti klasių bei įrašoma informacija į CRC cards. Šioje stadijoje naudojama UML (Universal Modeling Language).