Programování v C++

Dědičnost, šablony a kontejnery

Pavel Fibich

pavel.fibich@prf.jcu.cz

24-11-2010

přístupová práva

```
class Kniha
{ private: // privatni (neverejne) promenne
    string autor;
    string nazev;
  public:                          // verejne pristupne
    Kniha();                       // prazdny konstruktor
    Kniha(string a,string n) {autor=a; nazev=n;};
    string getAutor() {return autor;}
  private: // nasledujici metodu mohou pouzivat
           // jen jine metody teto tridy
    void pomocna() {};
};
int main() {
 Kniha k("Milan Kundera","Nesmrtelnost");
 cout << k.getAutor() << endl;
 // cout << k.autor << endl;
 // k.pomocna();
 return 0;
}
```

dědičnost

Třída B je podtřídou třídy A, napíšeme jako

```
class B: public A { ...
```

podtřída B zdění funkce a třídní proměnné od třídy A jako public. Třída může být podtřídou více tříd

```
class B: public A, public B, public C { ...
```

```
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int delka, vyska;
  public:
    Polygon() {cout << "Polygon" << endl;};
    void nastav (int a, int b)
      { delka=a; vyska=b;}
    virtual int plocha (void) =0;
  };
...
```

dědičnost

```
class Obdelnik: public Polygon {
  public:
    Obdelnik() {cout << "Obdelnik" << endl;};
    int plocha () { return (delka * vyska); }
  };
class Trojuhelnik: public Polygon {
  public:
    Trojuhelnik() {cout << "Trojuhelnik" << endl;};
    int plocha () { return (delka * vyska / 2); }
  };
int main () {
  Obdelnik obd;   Trojuhelnik tr;
  Polygon * poly = &obd;
  obd.nastav (4,5);  tr.nastav (4,5);
  poly->nastav(3,2);
  cout << obd.plocha() << endl;
  cout << tr.plocha() << endl;
  cout << poly->plocha() << endl;
  return 0; }
```

template (šablona) funkce

Obecný (neurčený) typ v definici místo konkrétního datové typu.

```
#include <iostream>
using namespace std;

template <class T> // misto datoveho typu mame T
T GetMax (T a, T b) {
  return (a>b?a:b);
}

int main () {
  int i=5, j=6, k;
  long l=10, m=5, n;
  k=GetMax(i,j);
  n=GetMax(l,m);
  cout << k << endl;
  cout << n << endl;
  return 0;
}
```

template (šablona) třídy

```
#include <iostream>
using namespace std;
template <class T>
class mypair {
    T a, b;
  public:
    mypair (T first, T second) {a=first; b=second;}
    T getmax ();
};
template <class T>
T mypair<T>::getmax () {
  T retval;
  retval = a>b? a : b;
  return retval;
}
int main () {
  mypair <int> myobject (100, 75); // specifikujeme typ
  cout << myobject.getmax();
  return 0;
}
```

kontejnery

STL obsahuje datové kontejnery, iterátory a šablony algoritmů.

Vybrané kontejnery (šablony tříd) ▸ http://www.cplusplus.com/reference/

- list - oboustranně zřetězený seznam
- map - asiciativni pole (indexováno ne jen čísly)
- set - množina
- queue - fronta
- stack - zásobník
- vector - jednorozměrné pole

```
#include <stack>
#include <iostream>
int main() {
 std::stack<int> zas;
 for(int i = 0; i < 10; i++) zas.push(i); // ulozi prvek
 std::cout << "Pocet prvku " << zas.size() << std::endl;
 while (!zas.empty()) {
  std::cout << zas.top() << std::endl; // prvek na vrcholu
  zas.pop();  // vyhodi prvek na vrcholu
 }
 return 0; }
```

vector

Kontejner blízký poli. ▸ http://www.cplusplus.com/reference/stl/vector/

```
#include <vector>
#include <iostream>
using namespace std;
int main() {
 vector<int> a;
 for(int i = 0; i < 5; i++) {
  a.push_back(i+1);
  cout << "Posledni prvek : " << a.back() << endl;
  }
 for(int i = 0; i < a.size(); i++) cout <<a[i]<< '\t';
 cout << endl;
 while (!a.empty()) a.pop_back(); // odeber posledni
 cout << "Velikost " << a.size() << endl;
 return 0;
}
```

map

Asiciativní kontejner s klíčem a hodnotou.

- http://www.cplusplus.com/reference/stl/map/

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main() {
 map<string,int> telSeznam;
 /* Klic je string, hodnota je int */
 telSeznam.insert(make_pair(string("Dostal"),4578963));
 telSeznam.insert(make_pair(string("Hasici"),150));
 telSeznam.insert(make_pair(string("Policie"),158));
 cout << "Hasici " << telSeznam[string("Hasici")] << endl;
 cout << "Dostal " << telSeznam[string("Dostal")] << endl;

 telSeznam.erase(string("Dostal"));
 cout << "Velikost " << telSeznam.size() << endl;
}
```

algoritmy

Součást `algorithm` knihovny, pracují s kontejnery nebo bez nich. Pár základních: `find` (hledá prvek v rozsahu hodnot), `count` (počet výskytů prvku), `sort` (třídí), `reverse` (obrací pořadí), `random_shuffle` (míchá prvky), `fill` a `generate` (plní kontejner)

```
int myints[] = { 10, 20, 30 ,40, 10 };
int * p;
p = find(myints,myints+5,30);
mycount = (int) count (myints, myints+5, 10);

int myints2[] = {32,71,12,45,26,80,53,33};
vector<int> myvector (myints, myints+8);
sort (myvector.begin(), myvector.end());

reverse(myvector.begin(),myvector.end());

random_shuffle ( myvector.begin(), myvector.end() );
```

- http://www.cplusplus.com/reference/algorithm/

iterátory

Iterátor ukazuje na nějaký prvek z rozsahu hodnot a takové hodnoty umí procházet. Každý kontejner má vlastní iterátor pro efektivní procházení.

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
 vector<int> the_vector;
 vector<int>::iterator the_iterator;

 for( int i=0; i < 10; i++ ) the_vector.push_back(i);
 int total = 0;
 the_iterator = the_vector.begin();
 while( the_iterator != the_vector.end() ) {
  total += *the_iterator;
  ++the_iterator;
 }
 cout << "Total=" << total << endl;
 return 0; }
```

jak generovat náhodná čísla

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <algorithm>
#include <ctime>
using namespace std;
// generator jako funkce
int rnumber () { return (rand()%2); }
int main() {
 vector<int> v (10);
 // seminku generatoru jako aktualni cas
 srand ( unsigned ( time(NULL) ) );
 generate( v.begin(), v.end(),rnumber); // naplni vektor
 //vypis
 for(vector<int>::iterator it=v.begin(); it!=v.end(); ++it)
  cout << " " << *it;
 return 0;
}
```

Ukol 4. - podmínky

- odvezdání přes ▸ http://moodle.prf.jcu.cz/course/view.php?id=16
- 0-10 bodů, za opisování jen jedno varování a mínus body pro oba účastníky
- odevzdáte formátovaný a komentovaný zdrojový soubor (prijmeni_jmeno.cpp)
- na začátku .cpp souboru bude zakomentovaná a vyplněná následující hlavička

```
AUTOR: (jmeno prijmeni, osobni cislo zacinajici B)
EMAIL:
SKUPINA: (kam chodite a jste zapsani na cviceni)
VERZE G++: (ziskate pomoci prikazu g++ -v)
PRIKAZ PREKLADU: (jaky prikazem byl program prelozen)
OS: (operacni system, napr. Win 7 64b)
CISLO UKOLU:
POZNAMKY AUTORA:
```

- 2 bonusové body získáte, když místo Explodera se bude dělat Glider Gun (viz. odkazy dále), bude třeba větší plocha a víc kroků než pro Exploder (to klidně změnte)

Ukol 4. - hra Život

- cílem je napsat hru Život ▸ http://en.wikipedia.org/wiki/Conway's_Game_of_Life, pro spustění v prohlížeči přes ▸ http://www.bitstorm.org/gameoflife/
- hrací plocha bude mít 20 x 20 políček, realizujte ji pomocí `vector`u (nebo pole ale dostanete za to o 1b míň)
- každé políčko je buď obsazené (živé) nebo prázdné (mrtvé), s tím že krajní buňky (hranice) budou pořád prázdné
- každá buňka má 8 sousedních buněk (nejbližší sousedi)
- na všech buňkách (krom hraničních) se provádí následující pravidla v určených počtech kroků
  - má-li buňka víc jak 3 nebo méně než 2 sousedy umírá na stísnění nebo samotou
  - živá buňká která má 2 nebo 3 sousedy zůstává dál živou
  - mrtvá buňka která má 3 sousedy se stává se živou
- v cyklech se vždy mění plocha tak, že nová plocha se vyplňuje na základě hodnot buněk ze předchozí (starší) plochy (tj. změny v rámci jednoho kroku se neovlivňují)

Ukol 4. - podrobnosti

- dodržte formát výstupu jako je v příkladech, tj. na začátku a v každém kroku se výpíše co je na ploše, číslo kroku a počet živých
- program akceptuje jeden (od 2 do 100) nebo žádný argument
- pokud je zadán jeden argument, vygenerují se náhodné hodnoty buněk (krom hraničních, ty jsou pořád prázdné); argument určuje kolik cyklu hry proběhne
- pokud není zadán žádný argument provede se 30 kroků a úvodní stav plochy bude typu Exploder, viz. výstup programu: ‣ http://botanika.prf.jcu.cz/fibich/cpp/exploder.txt Váš program puštěný bez argumentu by měl mít stejný výstup.
- `ukol4.exe 4 5`
  `Program akceptuje zadny nebo jeden parametr 2-100`
- `ukol4.exe 40`
  Bude vypisovat něco jako ‣ http://botanika.prf.jcu.cz/fibich/cpp/nahodny.txt. Jen upozorním, že Vaše náhodná čísla budou nejspíš jiná, takže i výstup bude jiný.