Statické členy třídy

Jsou označeny klíčovým slovem *static* a platí pro ně, že jsou pro všechny objekty třídy společné a existují nezávisle na objektech třídy.

Přístup ke statickému členu třídy je možný:

- Použitím operátoru rozlišení :: (k tomu není zapotřebí, aby existoval nějaký objekt třídy).
- Přes libovolný existující objekt třídy běžným způsobem (operátorem . nebo ->).

Statické proměnné

Jsou deklarovány ve třídě s klíčovým slovem *static* a musí mít navíc samostatnou definici vně třídy (v definici je zpravidla inicializace statické proměnné).

Statické funkce

Jsou deklarovány ve třídě s klíčovým slovem *static* a nemohou používat jiné proměnné třídy než její statické proměnné.

```
class DVD { static unsigned pocetDVD;
            string nazev;

    public: DVD(const char *nazev): nazev(nazev) { ++pocetDVD; }
            static unsigned pocet() { return pocetDVD; }
            ~DVD() { --pocetDVD; }
};

unsigned DVD::pocetDVD=0;

DVD dN("Dr. No"),qOS("Quantum of Solace");

cout << DVD::pocet();    // 2
cout << dN.pocet();      // 2
```

Dotazovací funkce třídy

Jsou nestatické členské funkce třídy, které nemění stav objektu - nemění hodnoty proměnných objektu (vyjma proměnných označených klíčovým slovem *mutable*). Mohou být označeny klíčovým slovem *const* za hlavičkou funkce.

```
class Jmeno { string jm;

      public: Jmeno(const char *j): jm(j) { }

              const char *jmeno() const { return jm.data(); }
};
```

Ukazatel na aktuální objekt

Je označen klíčovým slovem *this*.

```
class Bod { float x,y;
```

```
    public: Bod(float x,float y) { this->x=x; this->y=y; }
            Bod *ukazatel() { return this; }
            Bod &reference() { return *this; }
};

Bod b(2,3);

cout << "Adresa: " << b.ukazatel() << " " << &b << endl;
```

Konstantní proměnná třídy

Je označena klíčovým slovem *const* a musí jí být přiřazena hodnota inicializátorem v konstruktoru třídy. Její hodnota nemůže být po inicializaci změněna.

```
class Ucet { const unsigned cislo;
             int stav;
     public: Ucet(unsigned c): cislo(c) { stav=0; }
};
```

```
struct Datum { char den,mesic;
               short rok;
    Datum(char d,char m,short r): den(d),mesic(m),rok(r) { }
};
class Osoba { string jmeno,prijmeni;
              const Datum datumNarozeni;
  public:
    Osoba(const char *j,const char *p,char d,char m,short r):
                  jmeno(j),prijmeni(p),datumNarozeni(d,m,r) { }
};
```

Konstantní objekt

Je deklarován se specifikací *const*. Lze u něho volat jen členské funkce, které nemění stav objektu - jsou označeny jako dotazovací klíčovým slovem *const* v hlavičce (ty mohou měnit jen proměnné se specifikací *mutable*).

```
class Jmeno { string jm;
      public: Jmeno(const char *j):jm(j) { }
              void zmenit(const char *j) { jm=j; }
              const char *jmeno() const { return jm.data(); }

};

Jmeno p("Pavel");

p.zmenit("Petr");
```

```
cout << p.jmeno();

const Jmeno e("Eva");

e.zmenit("Jana");      // chyba !! – objekt e je konstantní

cout << e.jmeno();
```

Spřátelené funkce a třídy

V třídě lze uvést deklaraci funkce, která existuje vně třídy, nebo lze uvést jiné třídy, které označíme klíčovým slovem *friend*. Taková funkce nebo třída je považována za spřátelenou funkci nebo třídu a má neomezený přístup k členům dané třídy.

```
class Ucet { const unsigned cislo;
             int stav;
     public: Ucet(unsigned c): cislo(c) { stav=0; }
             void ulozit(int castka) { stav += castka; }
             void vybrat(int castka) { stav -= castka; }
             friend void vypsat(const Ucet &);
             friend void pripsatUrok(Ucet &);
};

void vypsat(const Ucet &u)
{ cout << "Ucet: " << u.cislo << "  stav: " << u.stav << endl; }

void pripsatUrok(Ucet &u)
{ u.stav *= 1.01; }

Ucet u(8319);

u.ulozit(1000);
u.vybrat(200);
pripsatUrok(u);
vypsat(u);     // Ucet: 8139  stav: 808
```

```
class Akcie { unsigned hodnota;
      public: Akcie(unsigned Kc): hodnota(Kc) { }
              unsigned cena() const { return hodnota; }
              friend class Burza;
};

class Burza { Akcie &a;
      public: Burza(Akcie &akcie): a(akcie) { }
```

```
            void rust(unsigned Kc)    { a.hodnota += Kc; }
            void pokles(unsigned Kc)  { a.hodnota -= Kc; }
};

Akcie OLMA(1000);
Burza b(OLMA);
b.rust(5);
```

Třídy mohou být deklarovány i ve funkcích (lokální třídy)

```
float korenLinearniRovnice(float a,float b)
{
  class Rovnice { float a,b;
          public: Rovnice(float a,float b): a(a),b(b) { }
                  float koren() const { return -b/a; }
  } r(a,b);
  return r.koren();
}
```

Třídy mohou být deklarovány i v jiných třídách (lokální třídy)

```
class Usecka
{
  class Bod { float x,y;
      public: Bod(float x,float y): x(x),y(y) { };
              float vzdalenost(const Bod &b) const
              { return sqrt(pow(x-b.x,2)+pow(y-b.y,2));
              }
  };
  Bod b1,b2;
  public: Usecka(float x1,float y1,float x2,float y2):
                                        b1(x1,y1),b2(x2,y2) { }
          float delka() const { return b1.vzdalenost(b2); }
};
```