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 jiné třídy označené 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 vypocitatUrok(Ucet &);
};

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

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

Ucet u(8319);

u.ulozit(1000);
u.vybrat(150);
vypocitatUrok(u);
vypsat(u);
```

```
class Akcie { int hodnota;
      public: Akcie(int h):hodnota(h) { }
              int cena() { return hodnota; }
              friend class Burza;
};

class Burza { Akcie &a;
      public: Burza(Akcie &akcie): a(akcie) { }
              void rust(int Kc)   { a.hodnota += Kc; }
              void pokles(int 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; }
  };
  Rovnice r(a,b);
  return r.koren();
}
```

Třídy mohou být deklarovány i v jiných třídách (vnořené 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); }
};
```