5. Proseminář

Přetěžování operátorů

Přetěžování operátorů

- Přetížení operátorů:
  - klíčové slovo **operator,**
  - přetížení pomocí funkce,
  - přetížení pomocí metody.
- Metodou lze přetížit všechny operátory kromě:

  :: . .* ?:
- Funkcí dále nelze přetížit operátory:

  -> ->* = () []
- Přetížit funkcí či třídní metodou lze i operátory:

  new new[] delete delete[]
- Metodou lze přetížit i operátory přetypování.

Přetěžování operátorů funkcí

```
struct TCplx
 {
   double re, im;
 };
TCplx operator + ( TCplx a, TCplx b )
 {
   TCplx c;
   c . re = a . re + b . re;
   c . im = a . im + b . im;
   return c;
 }
TCplx x = { 1, 0 }, y = { 2, 1 }, z;
z = x + y;
z = operator + ( x, y ); // jiny zapis volani op.
```

Přetěžování operátorů funkcí

- Operátor + přetížený v příkladu nebude použit pro:

  TCplx a, b;
  a = b + 3;
  a = 4 + b;

- Řešení 1:
  - přetížit operátor + i pro parametry typu **double,**
  - pro každý operátor musíme přetěžovat 3 varianty.

TCplx operator+ (TCplx a, double b)
{ … }
TCplx operator+ (double a, TCplx b)
{ … }

Přetěžování operátorů funkcí

- Řešení 2:
  - využít konstruktor uživatelské konverze z typu **double na typ TCplx,**
  - ponechat pouze jeden přetížený operátor +.

```
TCplx::TCplx ( double a )
 {
   re = a;
   im = 0;
 }

TCplx a, b;
a = 4 + b; // a = TCplx ( 4 ) + b;
```

Přetěžování operátorů funkcí

- Parametry funkce přetíženého operátoru:

TCplx operator+ (TCplx a, TCplx b)
{ … } // ok, ale pomale pro velke instance

TCplx operator+ (TCplx & a, TCplx & b)
{ … } // !!! mj. nebude asociativni.Proc?

TCplx operator+ (const TCplx & a,
const TCplx & b)
{ … } // ok

Přetěžování operátorů metodou

- Přetížení operátoru funkcí:
  - nelze pro všechny operátory,
  - problémy s přístupem ke členským proměnným objektů.
- Řešení – přetížení metodou:
  - přístup ke členským proměnným,
  - použitelné pro všechny operátory, které lze přetížit.
- Realizace:
  - jméno metody – **operator …,**
  - parametry – o jeden méně, než je arita operátoru,
  - levý operand – instance nad kterou je metoda spuštěna.

Přetěžování operátorů metodou

```
class CCplx
{
   double re, im;
  public:
   CCplx ( double r, double i=0 ) : re(r), im(i) {}
   CCplx operator - ( void ) const;
   CCplx operator + ( const CCplx & x ) const;
};

CCplx CCplx::operator - ( void ) const
{ // unarni minus – 0 parametru
  return ( CCplx ( -re, -im ) );
}
CCplx CCplx::operator + ( const CCplx & x ) const
{ // binarni plus – 1 parametr
  return ( CCplx ( re + x. re, im + x . im ) );
}
```

Přetěžování operátorů metodou

```
CCplx a (3, 4), b (2);

CCplx c = a + b;
  // c = a . operator + ( b );

CCplx d = a + 4;
  // ok, d = a . operator + ( CCplx ( 4 ) );

CCplx e = 5 + a;
  // chyba, neexistuje operator + ( int, CCplx & )
```

Přetěžování operátorů – **friend**

- Přetížit operátor+ pro **(double,const CCplx&):**
  - nelze metodou (**int není třída),**
  - musí se přetížit funkcí.
- Problém s funkcí – přístup ke členským proměnným třídy **CCplx:**
  - zviditelnit je **public (nevhodné),**
  - přístup pomocí čtecích metod (getter) – zdržení,
  - delegovat na tuto funkci právo přístupu ke členským proměnným (friend funkce).

Přetěžování operátorů – **friend**

```
class CCplx
 {
   double re, im;
  public:
   CCplx ( double r, double i=0 ) : re(r), im(i) {}
   CCplx operator - ( void ) const;
   CCplx operator + ( const CCplx & x ) const;
   friend CCplx operator + ( double a,
                             const CCplx & b );
 };
CCplx operator + ( double a, const CCplx & x )
 { // binarni plus funkci – 2 parametry
  return ( CCplx ( a + x. re, x . im ) );
 }
```

Přetěžování operátoru ()

- Pouze pro datový typ třída, přetížit metodou.
- Arita operátoru – libovolná (jediný takový operátor).
- Typ operandů – libovolný (velká variabilita).
- Využití – hlavně jako funktor v STL.
- Nazývá se také někdy funkční objekt.

```
class CStr
 {
   char * str;
   int    len;
   CStr   ( const char * str, int len );
  public:
   CStr   operator () ( int from, int to ) const;
    // priklad - podretezec.
   …
 };
```

Zadání proseminář – Příklad 1

- Napište program, který ze vstupního textu spočítá počet slov delších než 10.
- Program upravte tak, aby spočítal počet slov velikosti 1 až 9 a větších než 10.
- Pro implementaci použijte algoritmy z std. Pro ukládání slov použijte kolekci vector.

Zadání proseminář – Příklad 2

- Implementujte třídu smart_ptr, která se bude chovat jako „chytrý“ ukazatel. To znamená, že při jeho použití se nebudeme muset starat o jeho uvolnění.

  smart_ptr x(new MojeTrida());

  f(x);

  x->metodaA();
- Použijte metodu počítání ukazatelů.