Přetížení operátorů (operator overloading)

Přetížení operátorů je definování jejich chování pro další datové typy. Je specifickým typem polymorfismu. Přetížení operátorů je v C++ u některých operátorů definováno v programovacím jazyce (operátory **<<** a **>>** ve streamech) a rovněž uživatel může definovat své vlastní přetížení operátorů.

U přetížených operátorů zůstávají zachovány jejich charakteristické vlastnosti

- arita
- precedence
- asociativita

V definici přetížení musí aspoň jeden z typů operandů být třída.

Lze přetížit operátory:

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

Operátory **+**, **-**, *****, **&** lze přetížit binární i unární.

Nelze přetížit operátory:

::	.*	.	?:	sizeof

Přetížení lze definovat:

- Členskou funkcí třídy pro unární operátor nebo pro binární operátor, je-li jeho první (levý) operand třída.
- Globální funkcí.

Tvar členské funkce pro unární operátor

```
typ operator operátor () { }
```

Tvar členské funkce pro binární operátor

```
typ operator operátor (pravý_operand) { }
```

Tvar globální funkce pro unární operátor

```
typ operator operátor (operand) { }
```

Tvar globální funkce pro binární operátor

```
typ operator operátor (levý_operand,pravý_operand) { }
```

Úloha: Máme třídu pro uložení datumu

```
class Datum { char den,mesic; short rok; };
```

Potřebujeme setřídit pole datumů. Použijeme k tomu algoritmus třídění přímým vkládáním, který srovnává tříděné prvky operací srovnání **<**. Pro třídění definujeme tuto operaci ve třídě *Datum*.

```
class Datum { char den,mesic; short rok;
  public: bool operator < (Datum d)
          {
            if (rok!=d.rok)      return rok<d.rok;
            if (mesic!=d.mesic) return mesic<d.mesic;
                                 return den<d.den;
          }
};

void tridit(Datum a[], int n)
{
  for (int i=1; i<n; ++i)
  { Datum x=a[i];
    int j;
    for (j=i-1; j>=0 && x<a[j]; --j) a[j+1] = a[j];
    a[j+1] = x;
  }
};
```

Tvar funkce pro konverzi datového typu dané třídy na stanovený datový typ:

```
operator typ () { }
```

Retez << ''

Retez += ""

Retez += Retez

Retez = ""

Retez = Retez

(const char *) Retez

```
class Retez { char *p; int ix;
```

```
  public: Retez(int n) { p=new char[n+1]; ix=0; }
          Retez & operator << (char);
          Retez & operator += (const char *);
          Retez & operator += (const Retez &);
          Retez & operator = (const char *);
          Retez & operator = (const Retez &);
          operator const char * () { return p; }
          ~Retez() { delete [] p; }
};

Retez & Retez::operator << (char z)
{
  p[ix++]=z;
  p[ix]=0;
  return *this;
}

Retez & Retez::operator += (const char *s)
{
  strcpy(p+ix,s);
  ix+=strlen(s);
  return *this;
}

Retez & Retez::operator += (const Retez &r)
{
  strcpy(p+ix,r.p);
  ix+=strlen(r.p);
  return *this;
}

Retez & Retez::operator = (const char *s)
{
  strcpy(p,s);
  ix=strlen(s);
  return *this;
}

Retez & Retez::operator = (const Retez &r)
{
  strcpy(p,r.p);
  ix=strlen(r.p);
  return *this;
}

Retez r(100);
```

```
r << 'C' << '+' << '+';
r += " je ";
Retez s(100);
s = "programovaci jazyk";
r += s;
cout << r << endl;    // C++ je programovaci jazyk
```