Programování v C++ 1, 9. cvičení

přetěžování operátorů

Vladimír Jarý[1]

[1]Fakulta jaderná a fyzikálně inženýrská
České vysoké učení technické v Praze

Zimní semestr 2020/2021

Přehled

Shrnutí minule procvičené látky

- spojový seznam se zarážkou
  - nalezení prvku
  - odstranění prvku
- návrhový vzor iterátor
  - abstraktní iterátor
    - potomek může zastoupit předka
    - využití polymorfismu
  - iterátor seznamu
  - iterátor dynamického pole
  - použití pro nalezení nejmenšího prvku v posloupnosti

Přehled

Význam a omezení

- volitelná část jazyka, pro zpřehlednění kódu
- C++ umožňuje definovat operátory pro vlastní objektové a výčtové typy
- omezení:
  - operátory pro vestavěné typy nelze předefinovat
  - nelze měnit prioritu operátoru
  - nelze měnit asociativitu operátoru
  - nelze měnit počet operandů
- operátorová funkce:
  - operátor deklarován jako funkce se jménem `operator` doplněným o symbol operátoru
  - unární operátor: operátorová funkce je volná funkce s jedním parametrem nebo metoda bez parametrů
  - binární operátor: operátorová funkce je volná funkce se dvěma parametry nebo metoda s jedním parametry

Dělení operátorů

1. operátory, které nelze přetížit: `?:`, `.`, `.*`, `::`, **`typeid`**, **`static_cast`**, **`dynamic_cast`**, **`reinterpret_cast`**, **`const_cast`**
2. operátory **`new`** a **`delete`** lze přetížit jen jako volné funkce nebo statické metody objektových typů
3. operátory `()` (tj. volání funkce), `[]`, `->`, `=`, `(typ)` lze přetížit jen jako nestatické metody
4. zbývající operátory lze přetížit jako nestatické metody nebo jako volné funkce mající alespoň jeden parametr objektového nebo výčtového typu
   - aritmetické, porovnávací, logické, bitové operátory

Zadání

Zadání příkladu

Napište třídu `C`, která bude reprezentovat komplexní čísla. Do třídy doplňte přístupové metody a metodu pro výpočet absolutní hodnoty. Dále implementujte následující operátory:

1. aritmetické operátory (+, -, *, /)
2. operátory pro porovnání (==, !=)
3. operátor pro tisk do proudu («)
4. operátor pro načtení z proudu (»)

Třída C (bez přetížených operátorů)

```
1  class C{
2   public:
3     C();
4     C(double = 0, double = 0);
5     double getRe() { return re; }
6     void setRe(double val) { re = val; }
7     double getIm() { return im; }
8     void setIm(double val) { im = val; }
9   private:
10    double re, im;
11 };
12 C::C(){ C(0, 0);}
13 C::C(double r, double i)
14 : re(r), im(i){}
```

Sčítání komplexních čísel

Metoda `add`

```
1 C C::add(C &arg) const{
2     return C(re+arg.re, im+arg.im);
3 }
```

Použití:

```
1 C x(1, 0), y(0, 1);
2 C z = x.add(y);
```

- není příliš intuitivní
- zápis `z = x + y;` způsobí chybu překladu – překladač neví, jak sečíst dvě komplexní čísla
- řešením je přetížit operátor `+`

Sčítání komplexních čísel pomocí operátoru +

1. operátor jako metoda

```
1 C C::operator+(const C &c){
2     return C(re + c.getRe(), im+c.getIm());
3 }
```

   - zápis `C z = x + y;` ekvivaletní zápisu `C z = x.operator+(y);`

2. operátor jako volná funkce

```
1 C operator+(const C &x, const C &y){
2     return C(x.re+y.re, x.im+y.im);
3 }
```

   - operátorová funkce musí být deklarována jako **`friend`** třídy `C` (přistupuje k soukromým atributům)
   - umožní i zápis `C z = 2 + 1;`

Operátory pro porovnávání komplexních čísel

1. rovnost (operátor ==)

```
1 bool operator==(const C &x, const C &y){
2  return ((x.re == y.re) && (x.im == y.im));
3 }
```

2. nerovnost (operátor !=)
   - lze využít přetížený operátor ==

```
1 bool operator!=(const C &x, const C &y){
2   return !(x == y);
3 }
```

Komplexní čísla a datové proudy

1. vložení komplexního čísla do proudu

```
1  ostream& operator<<(ostream& os,const C &x){
2    os << "[" << x.re << ";" << x.im << "i]" << endl;
3    return os;
4  }
```

2. získání komplexního čísla z proudu

```
1  istream& operator>>(istream& is, C &c){
2    double x, y;
3    is >> x >> y;
4    c.setRe(x);
5    c.setIm(y);
6    return is;
7  }
```

Přehled

Operátor indexování []

- operátor chápán jako binární: prvním parametrem instance objektového typu, druhým parametrem je index mezi `[]`
- je-li `o` objekt, pro který máme přetížený operátor `[]`, pak je zápis `o[i]` ekvivaletní zápisu `o.operator[](i)`
- příklad: komplexní čísla

```
1 double& C::operator[](string index){
2   if(index=="re") return re;
3   else if(index=="im") return i;
4   //else chyba;
5 }
```

- co dělat, pokud uživatel zadá index mimo meze?
  - $\Rightarrow$ výjimky
- referenční funkce: vytváří l-hodnotu

Závěr

Shrnutí

- přetěžování operátorů
  - omezení
  - operátorová funkce, dělení operátorů
  - příklad: komplexní čísla
    - aritmetické operátory
    - operátory pro porovnání
    - operátory pro vložení/vyjmutí do proudu
- operátory, které lze přetížit jako nestatické metody
  - operátor indexace []