Programowanie obiektowe (TZ1E2010)

Politechnika Białostocka - Wydział Elektryczny

Elektronika i telekomunikacja, semestr II
studia niestacjonarne I stopnia

Rok akademicki 2020/2021

Pracownia nr 5 (16.04.2021)

dr inż. Jarosław Forenc

Przeładowanie (przeciążanie) operatorów

- przeładowanie operatora polega na nadaniu mu specjalnego znaczenia w momencie, gdy stoi on obok obiektu jakiejś klasy

obiektA operator obiektB

```
klasa obiektA, obiektB, obiektC;
obiektC = obiektA + obiektB;
```

operator (wymaga napisania własnej funkcji)

operator (może wymagać napisania własnej funkcji)

```
obiektC = obiektA + 5;          obiektC = 5 + obiektB;
```

operator (wymaga napisania własnych funkcji)

Przeładowanie operatorów w klasie

- przeładowanie operatora dokonuje się definiując funkcję o postaci:

słowo kluczowe — nazwa operatora (np. +, -, itp.)

```
typ_zwracany operator @ (argumenty)
{
    // ciało funkcji
}
```

jako co najmniej jeden z argumentów musi wystąpić obiekt klasy zdefiniowanej przez użytkownika

Przeładowanie operatorów w klasie

- można przeładować praktycznie wszystkie operatory
- nie można wymyślać swoich operatorów
- nie można zmieniać priorytetu operatorów
- automatycznie tworzone są operatory:
  - przypisania (=)
  - pobrania adresu (&)
  - new, new [], delete i delete [] (tworzenie i usuwanie obiektów)
- ten sam operator można przeładować wielokrotnie, ale za każdym razem funkcja operatorowa musi mieć inny typ lub kolejność argumentów

Funkcja przeładowująca operator

- definiowana jako funkcja składowa klasy

obiektA @ obiektB

```
klasa klasa::operator @ (klasa obiektB)
{
    ...
}
```

- do funkcji przekazywany jest tylko jeden argument (obiektB), argument obiektA przekazywany jest domyślnie przez wskaźnik this
- funkcja operatorowa, która jest składową klasy wymaga, aby obiekt stojący po lewej stronie operatora był obiektem tej klasy, np.
  - obiektA + obiektB - można przeładować
  - obiektA + 5 - można przeładować
  - 5 + obiektB - nie można przeładować!!!

Funkcja przeładowująca operator

- definiowana jako funkcja globalna (zaprzyjaźniona z klasą)

obiektA @ obiektB

```
klasa operator @ (klasa obiektA, klasa obiektB)
{
    ...
}
```

- aby funkcja globalna mogła korzystać z pól prywatnych klasy musi być funkcją zaprzyjaźnioną z klasą:

```
friend klasa operator @ (klasa obiektA, klasa obiektB);
```

umieszczone w definicji klasy

- operatory >> i << można przeładowywać tylko jako funkcje globalne

Przykład: klasa kwadrat

- Dane składowe klasy:
  - długość boku (a)
- Funkcje składowe klasy / globalne (zaprzyjaźnione z klasą - friend):
  - konstruktor
  - przeładowanie operatora +
  - przeładowanie operatora – (friend)
  - przeładowanie operatora << (friend)
  - przeładowanie operatora >> (friend)
  - przeładowanie operatora ++ (pre- i postinkrementacji)
- Funkcje składowe klasy (nie ma konieczności ich definiowania):
  - konstruktor kopiujący
  - destruktor
  - przeładowanie operatora =

Przykład: klasa kwadrat (1/6)

```
#include <iostream>
using namespace std;
#include <cmath>

class kwadrat
{
   float a;
public:
   kwadrat(float bok);
   kwadrat operator + (kwadrat kw2);
   friend kwadrat operator - (kwadrat kw1, kwadrat kw2);
   friend ostream & operator << (ostream & ekran, kwadrat kw);
   friend istream & operator >> (istream & klawiatura, kwadrat & kw);
   kwadrat operator ++ ();     // preinkrementacja
   kwadrat operator ++ (int);  // postinkrementacja

   kwadrat(const kwadrat & kw);
   ~kwadrat();
   kwadrat & operator = (const kwadrat & kw);
};
```

Przykład: klasa kwadrat (2/6)

```
kwadrat::kwadrat(float bok=0)
{
   a = bok;
}

kwadrat kwadrat::operator + (kwadrat kw2)
{
   float bok = a + kw2.a;
   kwadrat kw(bok);
   return kw;
}

kwadrat operator - (kwadrat kw1, kwadrat kw2)
{
   float bok = kw1.a - kw2.a;
   if (bok < 0) bok = 0;
   kwadrat kw(bok);
   return kw;
}
```

konstruktor

przeładowanie operatora +
(funkcja składowa klasy)

przeładowanie operatora -
(funkcja globalna - friend)

Przykład: klasa kwadrat (3/6)

```
ostream & operator << (ostream & ekran, kwadrat kw)
{
   ekran << "[a = " << kw.a << "]";
   return ekran;
}

istream & operator >> (istream & klawiatura, kwadrat & kw)
{
   cout << "a: ";
   klawiatura >> kw.a;
   return klawiatura;
}
```

przeładowanie operatora <<
(funkcja globalna)

przeładowanie operatora >>
(funkcja globalna)

Przykład: klasa kwadrat (4/6)

```
kwadrat kwadrat::operator ++ ()
{
   a = a + 1;
   return *this;
}

kwadrat kwadrat::operator ++ (int)
{
   a = a + 1;
   return *this;
}
```

przeładowanie operatora ++
(preinkrementacji)

przeładowanie operatora ++
(postinkrementacji)

Przykład: klasa kwadrat (5/6)

```
kwadrat::kwadrat(const kwadrat & kw)
{
   a = kw.a;
}

kwadrat::~kwadrat()
{

}

kwadrat & kwadrat::operator = (const kwadrat & kw)
{
   if (&kw == this)
      return *this;
   a = kw.a;
   return *this;
}
```

konstruktor kopiujący

destruktor

przeładowanie operatora =
(funkcja składowa klasy)

Przykład: klasa kwadrat (6/6)

```
int main(void)
{
   kwadrat K1(8), K2(6), K3;

   cout << "K1: " << K1 << endl;
   cout << "K2: " << K2 << endl;
   cout << "K3: " << K3 << endl;
   K3 = K1 + K2;
   cout << "K3 = K1 + K2: " << K3 << endl;
   K3 = K1 - K2;
   cout << "K3 = K1 - K2: " << K3 << endl;
   K3 = K2 - K1;
   cout << "K3 = K2 - K1: " << K3 << endl;
   ++K1;
   cout << "++K1: " << K1 << endl;
   K1++;
   cout << "K1++: " << K1 << endl;
   cin >> K3;
   cout << "K3: " << K3 << endl;
}
```

```
K1: [a = 8]
K2: [a = 6]
K3: [a = 0]
K3 = K1 + K2: [a = 14]
K3 = K1 - K2: [a = 2]
K3 = K2 - K1: [a = 0]
++K1: [a = 9]
K1++: [a = 10]
a: 12
K3: [a = 12]
```