Тема 7

Tuesday, April 16, 2024 1:45 AM

Тема 7

Предефиниране на оператори. Приятелски класове и функции.
Пример за реализация на комплексно число и Nvector.

Предефиниране на оператори

Def. Оператор-функция със специален синтаксис

Видове: → унарни (1 аргумент) +, −, *, &, ++, --
→ бинарни (2 аргумента) +, −, *, /
→ тернарен (3 аргумента) ?:

• унарните оператори +, − не правят нищо, но се използват, когато динамично ще се определя коя операция ще се извършва

Характеристики:

1) Приоритет
2) Асоциативност – ляво (сходимо от ляво надясно), дясно → (((a $ b) $ c) $ d)
→ (a $ (b $ (c $ d)))
3) Позиция спрямо аргументите си → инфиксен
→ суфиксен
→ префиксен

Предефиниране (за работа с определен тип)

2 начина

Външна ф-я

Вътрешна ф-я
↳ левият обект е този, от който се извиква операторът

Извикване

A obj1, obj2

obj1.operator$(obj2) | алтернативен синтаксис

obj1 + obj2

this other

При предефиниране на оператори се спазват общоприети стандарти (без значение, че можем да ги накараме да правят каквото и да е)

Примери: 1) +, −, * ... не променят аргументите си, а връщат нов обект
2) +=, -=, -- променят левия аргумент и връщат неконст. реф.
3) ==, >, < ... не променят, връщат bool, сравняват

Особени случаи:

- оператори за поток - външни, за да бъде отляво потокът)
- оп. за инкрементация и декрементация
↳ суфиксни - връщат старата стойност на обекта
↳ [illegible] (вече променен)

- оп. ' за инкрементация и декрементация
  ↳ дефиксни - връща стойността на обекта
  ↳ префиксен - връща обекта (вече променен)

```
A &operator ++( ) { ... return *this; } //prefix
A operator ++ (int) { this->x++;
                      return A(x-1); } //suffix - тук слагаме dummy
                                       //пар-р само за да го различаваме
                                       //от префиксния
```

- предефиниране 2 вида оп. за индексация: за конст. неконст. достъп

Ограничения:

1) =, [ ], ( ), -> - могат да бъдат само член-ф-ии
2) :? , ::, . - не могат да предефинират се
3) не можем да създ. нови оператори
4) не можем да предефо. - асоц.
5) -> трябва да бъде - приоритет
   обект - ук/предефо/конст - позиция
6) ако предефинираме ||, && губим предварителното изчисление

Приятелски класове и функции - външни класове / ф-ии, които имат достъп до private член-данните

Синтаксис:

```
A {
  private:
    int x;
  public:
    friend void f();
    friend class B;
};
```

```
void f() {
  A obj;
  obj.x++;
}
```

```
class B {
  A obj;
  B() { obj.x=7; }
};
```

```
X
friend y;

Z
friendy;
```

- Z не е приятелска за X!