

Предфиниране на оператори

Def. Оператор-функция със специален синтаксис

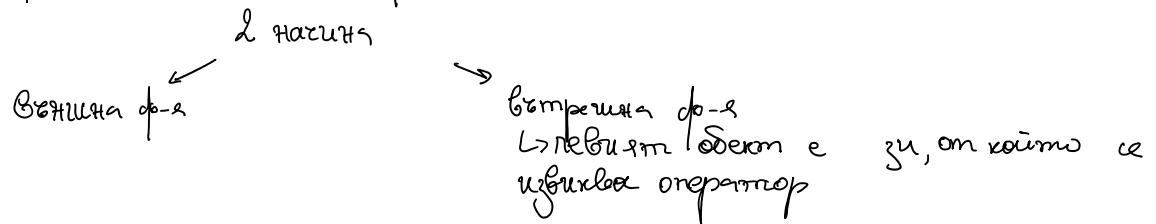
Видове: → унарни (1 аргумент) $+, -, *, \&$, $++, --$
→ бинарни (2 аргумента) $+, -, *, /$
→ тернарни (3 аргумента) $?:$

• унарните оператори $+, -$ не правят нищо, но се използват, когато дясно-ляво ще се определи коя операция ще се извършва

Характеристики:

- 1) Приоритет
- 2) Асоциативност - ляво (скобуваме от ляво надясно), десно $\rightarrow (((a \& b) \& c) \& d)$
- 3) Позиция спрямо аргументите си \rightarrow инфиксен, \rightarrow суфиксен, \rightarrow префиксен $\rightarrow (a \& b \& (c \& d))$

Предфиниране (за работа с определен тип)



Използване

$obj1, obj2$
 $obj1.operator \& (obj2)$ - алтернативен синтаксис
 $obj1 + obj2$
 this other

При предфиниране на оператори се спазват общоприети стандарти (без значение, се може да си направиш да правят каквото и да е)
 Примери: 1) $+, -, *, \dots$ не променят аргументите си, а връщат нов обект
 2) $+=, -=, --$ променят левия аргумент и връщат неговия реф.
 3) $==, >, <, \dots$ не променят, връщат bool, сравняват

Особени случаи:

- оператори за поток - външни, за да бъде отлево потокът
- оп. за инкрементация и декрементация
 - суфиксен - връща старата стойност на обекта
 - префиксен - връща новата стойност (с още променен)

- оп. ¹ за ⁰ инкрементация и декрементация
 - ↳ адреси - връща стойност на обекта
 - ↳ предвикат - връща обекта (всичко променя)

A & operator ++ () { ... return *this; } // prefix
 B operator ++ (int) { ... } // postfix

A operator ++ (int) { this → x ++ ;

```
return A(x-1); // suffix - тук ставяме dummy
// по-просто за да работим с
// от предмети
```

- предопределение 2 пути оп. за индексация: 1) конст. некоего достига

Организация:

- 1) =, [], (), → - могат да бъдат само елемент-и
- 2) :, ::, . - не могат да предизвикват
- 3) не могат да създават оператори
- 4) не могат да предизвикват асоциативност
- 5) → трябва да бъдат позиционни
- 6) ако предизвикват II, & иудни предизвиквателното изчисление

Приятелски класове и функции

- външни хласове /ф-чи, които имат достъп до private член-данните

Синтаксис:

```

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

```

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

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

$$\overset{x}{\text{friend}}\ y; \qquad \overset{z}{\text{friendly}}\ y;$$

- z не е приятелска за x !