# ОПЕРАТОРИ

доц. д-р Нора Ангелова

# ОПЕРАТОРИ

Операторите в C++ са:

– унарни (с един операнд)

– бинарни (с два операнда).

Всеки оператор се характеризира с:

– позиция на оператора спрямо операнда (операндите) му;

– приоритет;

– асоциативност.

# ОПЕРАТОРИ

**Позицията** на оператора спрямо операнда (операндите) му го определя като:

- префиксен (операторът е пред единствения си операнд),
- инфиксен (операторът е между двата си операнда),
- постфиксен (операторът е след единствения си операнд).

*Примери.*

1) Операторът * е както инфиксен (a * b), така и префиксен (*ptr);

2) операторът + е както инфиксен (a + b), така и префиксен (+b);

3) операторът ++ е както постфиксен (a++), така и префиксен (++a).

# ОПЕРАТОРИ

**Приоритетът** определя реда на изпълнение на операторите в израз (операторен терм).

Оператор с по-висок приоритет се изпълнява преди оператор с по-нисък приоритет.

*Пример.*

Приоритетът на операторите умножение и деление (* и /) е по-висок от този на операторите за събиране и изваждане (+ и -).

# ОПЕРАТОРИ

**Асоциативността** определя реда на изпълнение на оператори с еднакъв приоритет в израз. В езика C++ има лявоасоциативни и дясноасоциативни оператори.

Лявоасоциативните оператори се изпълняват отляво надясно, а дясноасоциативните – отдясно наляво.

*Примери.*

а) Аритметичните оператори +, -, * и / са лявоасоциативни. Затова изразът a-b-c е еквивалентен на (a-b)-c, а изразът a/b/c е еквивалентен на (a/b)/c.

б) Операторът за присвояване = е дясноасоциативен. Затова изразът a = b = c (a, b и c са променливи величини или обекти на клас) е еквивалентен на a = (b = c).

# ОПЕРАТОРИ

В езика C++ не е възможно да бъдат създавани нови оператори, но са дадени средства за предефиниране на съществуващи оператори.

Предефинираният оператор запазва всички характеристики на оригиналния, т.е. не могат да се променят приоритетът, асоциативността, броят и позицията на аргументите му.

# Оператори

Операторите:

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

могат да бъдат предефинирани, стига поне един операнд на оператора да е обект на клас.

# ОПЕРАТОРИ

Операторите, които не могат да се предефинират в C++ са:

- . - оператор за избор на компонента на клас,
- .* - оператор за избор на компонента на клас чрез указател,
- :: - оператор за присъединяване,
- ?: - тернарен оператор
- sizeof

При първите три оператора причината е, че имат за операнд име, а не стойност, а при другите два – спецификата им.

# ОПЕРАТОРИ

Предефинирането на оператори в езика C++ се осъществява чрез дефиниране на специален вид функции, наречени **операторни функции**.

Операторните функции имат синтаксис, подобен на синтаксиса на обикновените функции, но името им се състои от запазената дума **operator**, следвана от означението на предефинирания оператор.

Пример:

```
operator +
operator >
```

# ОПЕРАТОРИ

Когато предефинирането на оператор изисква достъп до компонентите на клас, обявени като private или protected, операторната дефиниция трябва да е член-функция или функция-приятел на класа.

Когато първият (или единственият) операнд на оператора, който искаме да предефинираме е обект на клас или псевдоним на обект на клас, операторът може да се предефинира и като член-функция, и като функция-приятел на класа.

# ОПЕРАТОРИ

При предефиниране като член-функция на клас, първият (или единственият) операнд не се задава като параметър. Ролята му се изпълнява от обекта, сочен от указателя this.

# ОПЕРАТОРИ

- Когато първият (или единственият) операнд на оператор, който искаме да предефинираме е обект на друг клас или псевдоним на обект на друг клас, или е от стандартен в езика тип (`int`, `double`, `char` и т.н.), операторът трябва да се реализира като приятелска функция на класа.

# УНАРНИ ОПЕРАТОРИ

**Унарен оператор на клас** може да се предефинира като:

- външна операторна функция;
- член-функция на класа без явно указан аргумент;
- приятелска функция на класа с един аргумент, който трябва да е или обект на класа, или псевдоним на обект на класа.

# Унарни оператори

Задача
Предефиниране на унарния оператор ! за класа Rat.
Операторът да проверява дали обект на класа Rat е равен на 0.

# Унарни оператори

- Външна операторна функция

```
bool operator!(Rat const & r) {
    return r.getNumer() == 0;
}
```

- Член-функция на класа Rat

```
bool Rat::operator!() const {
  return numer == 0;
}
```

- Функция приятел на класа Rat

```
bool operator!(Rat const & r) {
    return r.numer == 0;
}
```

*и в декларацията на класа Rat се добавя*

```
friend bool operator!(Rat const &);
```

# Унарни оператори

Оператор ++ - променя стойността на числителя да бъде сборът от числителя и знаменателя на рационалното число.

- Rat& operator++(); // префиксен ++

```
Rat& Rat::operator++() {
    numer = numer + denom;
    return *this;
}
```

```
Rat ratNumb(1, 2);
ratNumb.printRat();       // 1/2
(++ ratNumb).printRat(); // 3/2
ratNumb.printRat();       // 3/2
```

- Rat operator++(int); // постфиксен ++

```
Rat Rat::operator++(int) {
    Rat temp = *this;
    numer = numer + denom;
    return temp;
}
```

```
Rat ratNumb(1, 3);
ratNumb.printRat();     // 1/3
ratNumb++.printRat(); // 1/3
ratNumb.printRat();     // 4/3
```

++ и . имат равен приоритет и са лявоасоциативни

# БИНАРНИ ОПЕРАТОРИ

**Бинарен оператор на клас** може да се предефинира като:

- Външна операторна функция с два аргумента

(Единият от тези аргументи трябва да е обект на класа или псевдоним на обект на класа)

- Член-функция на класа с един явно указан аргумент

(Първият аргумент трябва да е обект на класа)

- Функция-приятел с два аргумента

(Единият от тези аргументи трябва да е обект на класа или псевдоним на обект на класа)

# Бинарни оператори

- Външна операторна функция с два аргумента

```
Rat operator+(Rat const & r1, Rat const & r2) {
    Rat r(
      r1.getNumer()*r2.getDenom() +
      r1.getDenom()*r2.getNumer(),
      r1.getDenom()*r2.getDenom()
    );
    return r;
}
```

Този начин не е ефективен заради непрекия достъп до член-данните на класа (достъпът до числителя и знаменателя на рационално число се осъществява чрез член-функциите за достъп getNumer и getDenom) !!!

# Бинарни оператори

- Член-функция на класа с един явно указан аргумент

```
Rat Rat::operator+(Rat const & r2) const {
    Rat r(
      numer*r2.denom + denom*r2.numer,
      denom*r2.denom
    );
    return r;
}
```

# Бинарни оператори

- Функция-приятел с два аргумента.

```
// Предварителна декларация на класа Rat
class Rat;

// Декларации на предефинирания оператор +
Rat operator+(Rat const &, Rat const &);

class Rat {
   friend Rat operator+(Rat const &, Rat const &);
   ...
};

Rat operator+(Rat const & r1, Rat const & r2) {
   Rat r(
     r1.numer*r2.denom + r2.numer*r1.denom,
     r1.denom*r2.denom
   );
   return r;
}
```

# ВРЕМЕ ЗА ВАШИТЕ ВЪПРОСИ