Аритметични и логически оператори

доц. д-р Нора Ангелова

Аритметични оператори

- + (събиране);
- - (изваждане);
- * (умножение);
- / (целочислено деление);
- % (остатък от целочислено деление);

Пример

```
std::cout << 11 % 3 << std:: endl;

std::cout << 11 / 3 << std::endl;
```

Пример

```
std::cout << 11 % 3 << std:: endl;
```

Резултат: 2

```
std::cout << 11 / 3 << std::endl;
```

Резултат: 3

Аритметични оператори

Вече познавате някои характеристики на операторите

- **Бинарни** – два операнда

a + b

a – b

- **Унарни** –един операнд

-a

Оператори

Характеристики:

- позиция на оператора спрямо операнда
- приоритет
- асоциативност

Оператори

Позиция на оператора спрямо операнда:

- 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 (+b).

2) 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 -).

https://en.cppreference.com/w/cpp/language/operator_precedence

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.

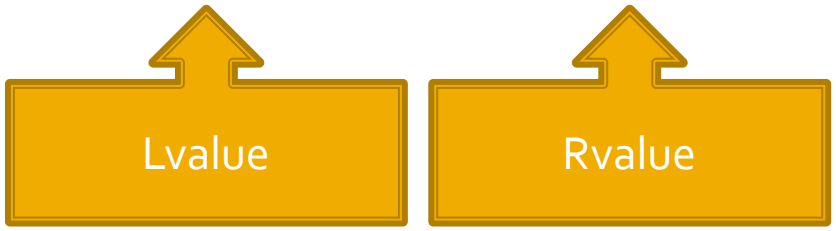
Примери.
1) Аритметичните оператори +, -, * и / са лявоасоциативни. Затова изразът *a - b - c* е еквивалентен на *(a - b) - c*,
а изразът *a / b / c* е еквивалентен на *(a / b) / c*.

2) 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)*.

Оператор за присвояване

Възможен е следният запис:

```
int myFirstIntVar;
myFirstIntVar = 5;
```

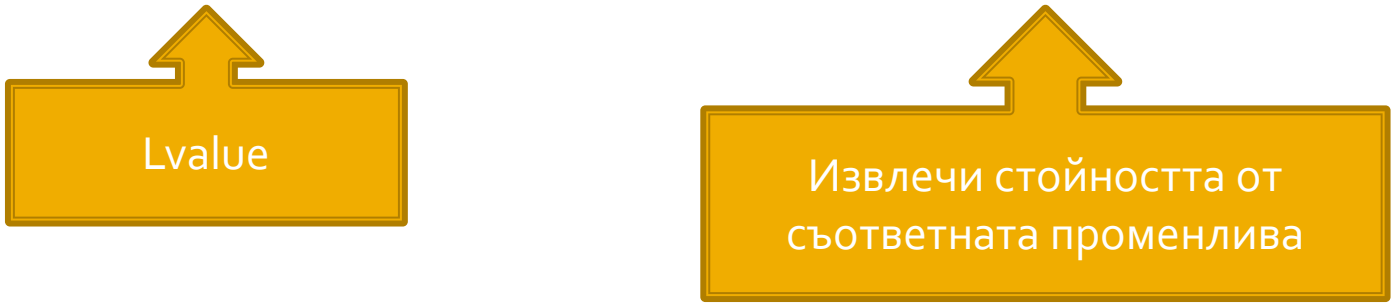


- Стойността 5 ще се присвои на променливата и ще изчезне

Оператор за присвояване

Възможен е следният запис:

```
int myFirstIntVar = 10;
int mySecondIntVar = myFirstIntVar;
```



- Стойността 10 ще се присвои на втората променлива

Оператор за присвояване

Възможен е следният запис:

```
int myFirstIntVar = 5;
myFirstIntVar = myFirstIntVar + 5;
```

Оператор за присвояване

Възможен е следният запис:

```
int myFirstIntVar = 5;
myFirstIntVar = myFirstIntVar + 5;
```

- `+=, -=, *=, /=`

Пример:

1.
```
int myFirstIntVar = 5;
myFirstIntVar += 5; // еквивалентен запис
```
2.
```
int intVar = 5;
std::cout << (intVar += 2) + (intVar += 10) + (intVar -= 3);
// Променя се стойността на intVar и стойността се използва в
// операцията събиране intVar == 5 + 2 + 10 – 3 == 14 => 3*14
```

Оператор за присвояване

Пример:

```
1.
    int intVar = 5;
    std::cout << (intVar += 2) + (intVar += 10) + (intVar -= 3);
    // Променя се стойността на intVar и стойността се използва в
    // операцията събиране intVar == 5 + 2 + 10 – 3 == 14 => 3*14
```

!!! Същият израз може да се пусне в онлайн компилатор
https://www.onlinegdb.com/online_c++_compiler

Резултат

```
int intVar = 5;
std::cout << (intVar += 2) + (intVar += 10) + (intVar -= 3);
// Резултат 48 – intVar == 5 + 2 + 10 = 17 => 17 + 17 + 14 = 48

int intVar = 5;
std::cout << (intVar -= 3) + (intVar += 10) + (intVar += 2);
// Резултат 38 – intVar == 5 - 3 + 10 = 12 => 12 + 12 + 14 = 38
```

Оператор за целочислено делене

```
int a = 123 / 10  // Целочислено делене
                  // a == 12


float b = 123 / 10; // Целочислено делене
                    // b == 12

float c = 123.0 / 10; // c == 12.3
```

Оператор ++/--

```
int myFirstIntVar = 5;

myFirstIntVar++; // myFirstIntVar: 6
++myFirstIntVar; // myFirstIntVar: 7

myFirstIntVar--; // myFirstIntVar: 6
--myFirstIntVar; // myFirstIntVar: 5
```

Оператор ++/--

- Префиксни и постфиксни ++/-- оператори в израз
  - Постфиксните оператори връщат rValue стойност и променя променливата
  - Префиксните оператори променят стойност в клетката и връщат като резултат

Примери:

1.
```
int intVar = 5;
std::cout << intVar++ + 5 << std::endl; // Резултат - 10
```
2.
```
int a = 5;
std::cout << ++intVar + 5 << std::endl; // Резултат – 11
```
3.
```
int a = 5;
std::cout << ++intVar++ << std::endl; // Резултат – грешка
// За оператора ++ е нужно lvalue
```

Логически оператори

- Оператор за логическо умножение (конюнкция)

```
bool var1 = true;
bool var2 = true;

var1 && var2; // true;
```

var1	var2	var1 && var2
false	false	false
false	true	false
true	false	false
true	true	true

Логически оператори

- Оператор за логическо събиране (дизюнкция)

```
bool var1 = true;
bool var2 = false;

var1 || var2; // true;
```

var1	var2	var1 || var2
false	false	false
false	true	true
true	false	true
true	true	true

- Оператор за логическо отрицание

```
bool var1 = false;
!var; // true;
```

var1	! var1
false	true
true	false

Оператори за сравнение

- == - сравнение за равно
- != - сравнение за различно
- > - сравнение за по-голямо
- >= - сравнение за по-голямо или равно
- < - сравнение за по-малко
- <= - сравнение за по-малко или равно

Логически оператори

- Операндите се оценяват отляво-надясно.
- Оценяването продължава докато се получи стойността на израза.

Пример:

```
double x, y;
// …
// проверка дали точка е в 1-ви квадрант
(x > 0) && (y > 0)
```

Следва продължение...