Константи, оператори, условни оператори

гл.ас. д-р. Нора Ангелова

Константи

- const <тип> <име_на_променлива> = <стойност>;

```
// Декларация на константа
const float PI = 3.14;

PI = 3.1415 // Грешка
```

** Имената на константите често се изписват с главни латински букви.*

Аритметични оператори

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

Пример

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

2

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

3

Оператори

- **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 инфиксен (a + b) и префиксен (+b).

2) Операторът ++ е постфиксен (a++) 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&
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 a = 5;
a = a + 5;
```

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

Пример:

```
a += 5;
```

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 a = 5;

a++; // a: 6
++a; // a: 7

a--; // a: 6
--a; // a: 5
```

Оператор ++/--

```
int a = 5;

cout << a++; // извежда 5
cout << ++a; // извежда 7

cout << a--; // извежда 7
cout << --a; // извежда 5
```

Логически оператори

- Оператор за логическо умножение (конюнкция)

```
bool A = true;
bool B = true;

A && B; // true;
```

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

Логически оператори

- Оператор за логическо събиране (дизюнкция)

```
bool A = true;
bool B = false;

A || B; // true;
```

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

- Оператор за логическо отрицание

```
bool A = false;
!A; // true;
```

A	!A
false	true
true	false

Логически оператори

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

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

Условен оператор

```
if (<условие>) <оператор>
```

- if – запазена дума
- <условие> – булев израз
- <оператор> – произволен оператор

Пример:

```
* Операторът може да бъде ограден в {} скоби.

if (a < 3) cout << "a е < 3" << endl;
```

Условен оператор

```
if (<условие>) {
    <оператор>;
    {<оператор>;}опц
}
```

Пример:

* Съвкупността от оператори трябва да бъде оградена в {} скоби.

```
bool isSmaller = false;
if (a < 3) {
    cout << "a e < 3" << endl;
    isSmaller = true;
}
```

Оператор if/else

```
if (<условие>) {
    <оператор1>;
    {<оператор1>;}опц
}
else {
    <оператор2>;
    {<оператор2>;}опц
}
```

- if – запазена дума,
- <условие> - булев израз,
- <оператор1> и <оператор2> - произволни оператори,
- ако има само един оператор, скобите могат да се пропуснат.

Пример:

```
if (a < 3) {
  cout << "a е < 3" << endl;
} else {
  cout << "a е >= 3" << endl;
}
```

Оператор if/else

- <оператор1> и <оператор2> са произволни оператори за управление на изчислителния процес - могат да бъдат условни оператори.

Пример:

```
if (a > 4) {
  cout << "a e > 4" << endl;
} else if (a < 4) {
  cout << "a e < 4" << endl;
} else {
  cout << "a e == 4" << endl;
}
```

Оператор if/else

- <оператор1> и <оператор2> са произволни оператори за управление на изчислителния процес - могат да бъдат условни оператори.

Пример:

```
if (a > 4) {
  cout << "a e > 4" << endl;
} else {
  if (a < 4) {
    cout << "a e < 4" << endl;
  } else {
    cout << "a e == 4" << endl;
  }
}
```

Тернарен оператор

- (<условие>) ? <оператор1> : <оператор2>

Пример:

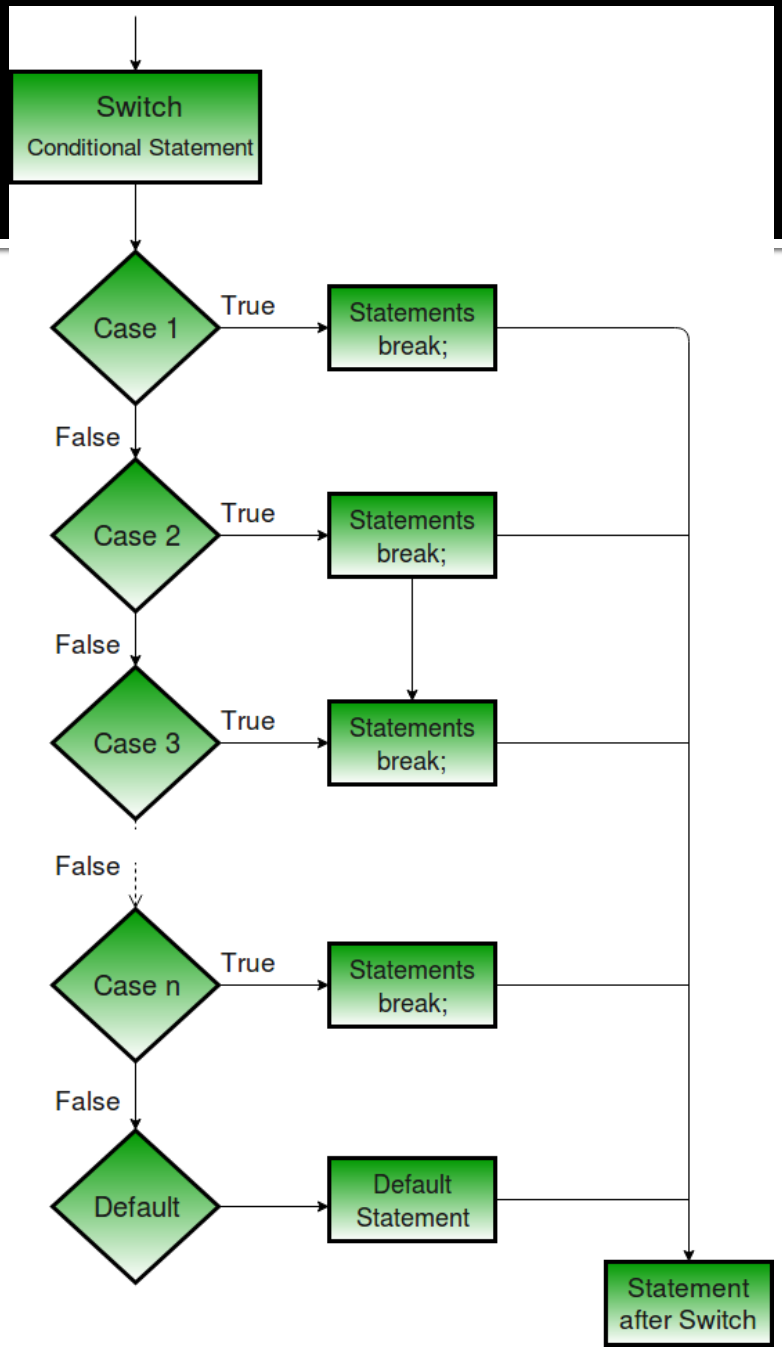
```
int a = 5;
int b = 3;
int larger = (a > b) ? a : b;
```

Оператор switch

- ```
  switch(<израз>) {
    case <израз1> : <редица от оператори 1>
    case <израз2> : <редица от оператори 2>
    ...
    [default: <редица от оператори n>]опц
  }
  ```

- `break` – прекратява изпълнението на най-вътрешния, съдържащ го оператор `switch` или оператор за цикъл.
  ```

Оператор switch



Пример

```
int x = 1;
int i = 1;
switch(x) {
  case 2: i+=2;
  case 1: i++;
  case 3: i+=5;
  default: i++;
}
cout << i << endl;
```

8

Следва продължение...