5. Итерации

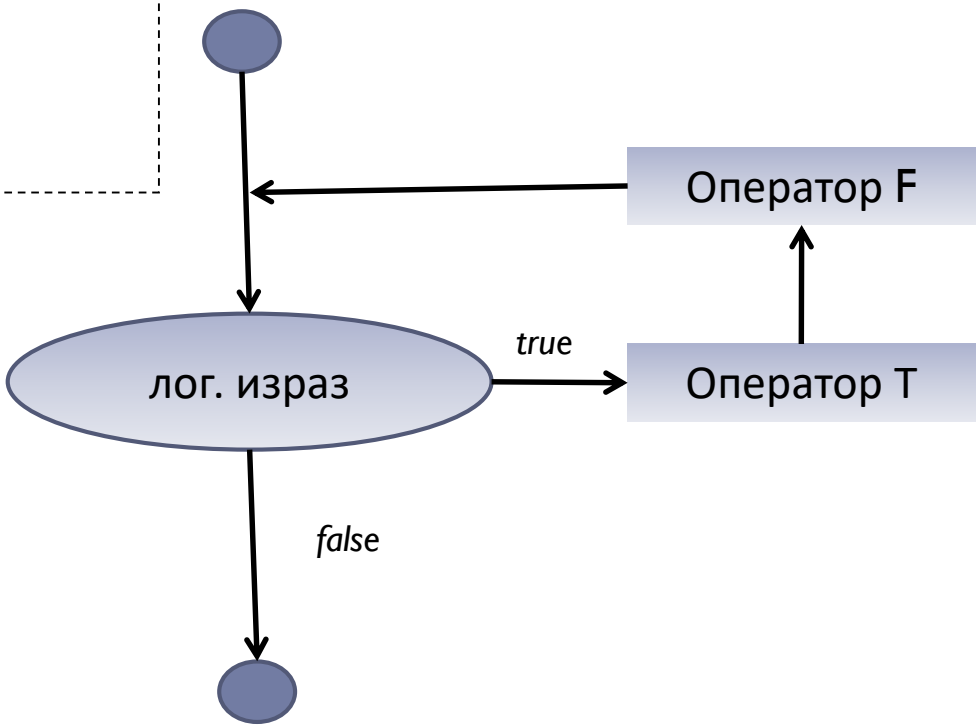
Лекционен курс “Програмиране на Java”
проф. д-р Станимир Стоянов

Структура на лекцията

- Циклични контролни потоци
- Оператори
- Примери
- Класификация на операторите

while оператор: циклична структура

```
while (логически израз) {
    оператор 1;
    оператор 2;
}
```



Синтаксис

EBNF:

```
while ( условие )
        оператор
```

Докато ‘условие’ е изпълнено, повтори ‘оператор’

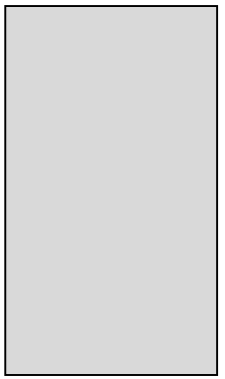
Съществен специален случай:

операторът не се обработва

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
   System.out.println(i + " " + v);
   i = i + 1;
   v = 2 * v;
}
```

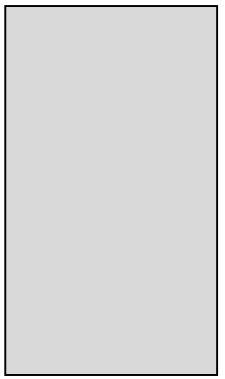


N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

i
0



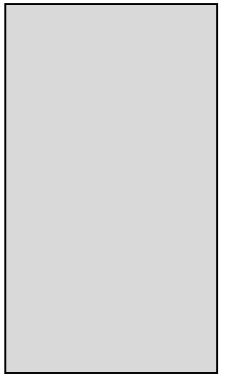
```
int i = 0;
int v = 1;
while (i <= N) {
   System.out.println(i + " " + v);
   i = i + 1;
   v = 2 * v;
}
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

i	v
0	1



```
int i = 0;
int v = 1;
while (i <= N) {
   System.out.println(i + " " + v);
   i = i + 1;
   v = 2 * v;
}
```

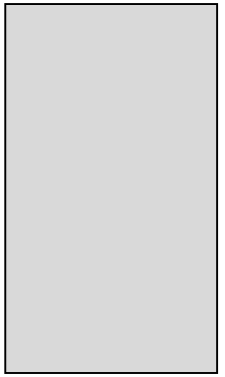
N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true



N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true

```
0 1
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1		

```
0 1
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	

```
0 1
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true

```
0 1
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true

```
0 1
1 2
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2		

```
0 1
1 2
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	

```
0 1
1 2
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true

```
0 1
1 2
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true

```
0 1
1 2
2 4
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3		

```
0 1
1 2
2 4
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	

```
0 1
1 2
2 4
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true

```
0 1
1 2
2 4
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true

```
0 1
1 2
2 4
3 8
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4		

```
0 1
1 2
2 4
3 8
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	

```
0 1
1 2
2 4
3 8
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true

```
0 1
1 2
2 4
3 8
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true

```
0 1
1 2
2 4
3 8
4 16
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5		

```
0 1
1 2
2 4
3 8
4 16
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	

```
0 1
1 2
2 4
3 8
4 16
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true

```
0 1
1 2
2 4
3 8
4 16
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true

```
0 1
1 2
2 4
3 8
4 16
5 32
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6		

```
0 1
1 2
2 4
3 8
4 16
5 32
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	

```
0 1
1 2
2 4
3 8
4 16
5 32
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true

```
0 1
1 2
2 4
3 8
4 16
5 32
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7		

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	

N = 6

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

Степен на две: трасиране

- Отпечатай степените на 2, които $\leq 2^N$.
  - Увеличаване на `i` от `0` до `N`.
  - Всеки път удвояване на `v`.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

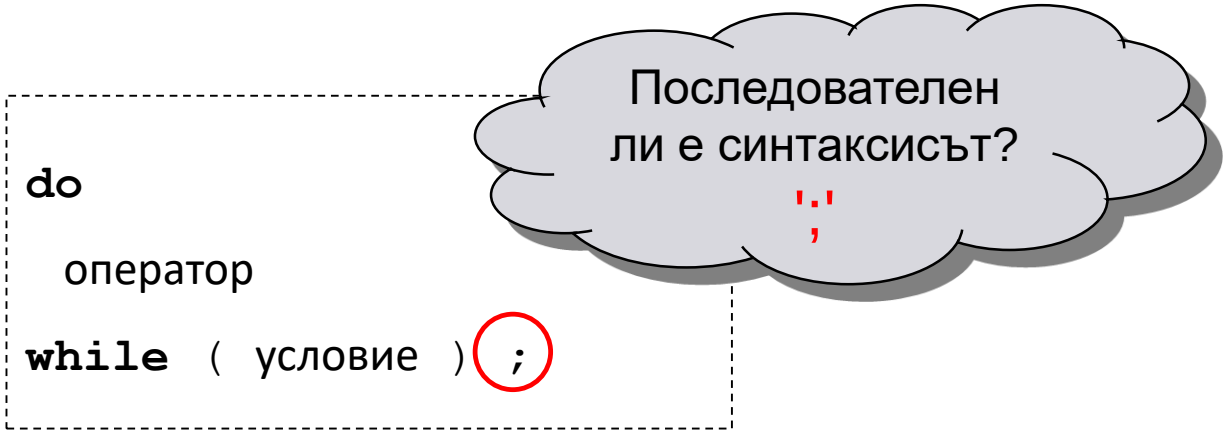
i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

Do-while-оператор

EBNF:



Докато ‘условие' е изпълнено, повтори ‘оператор', при което условието се тества след изпълнение на оператора

Приложение:

Операторът се обработва поне веднъж

Do-while-оператор: представен чрез while-оператор

```
do
  оператор
while ( условие ) ;
```

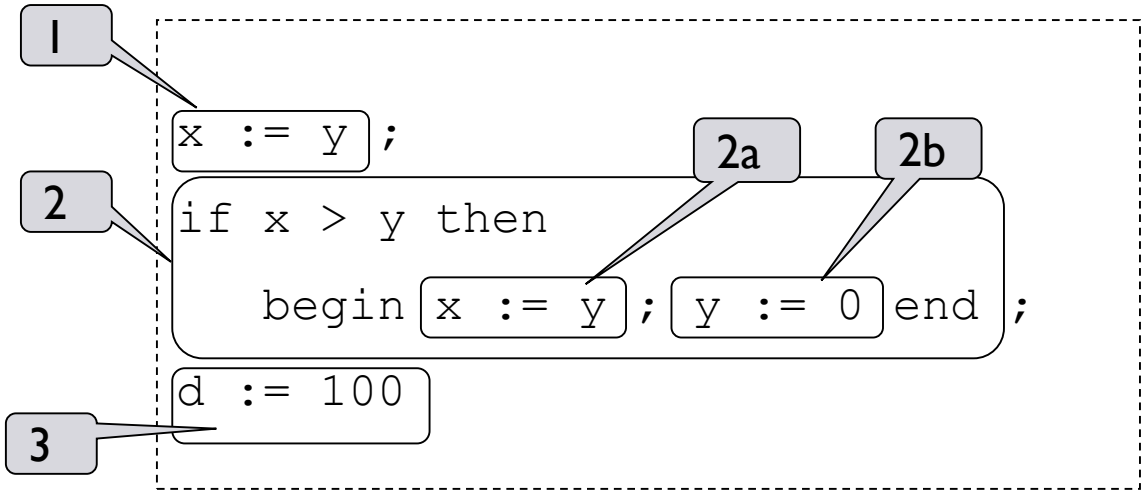
Еквивалентен на:

```
оператор
while ( условие )
    оператор
```

Роля на ';': Разделяне или край на оператори ?

Pascal, Modula-2, Ada, ... :

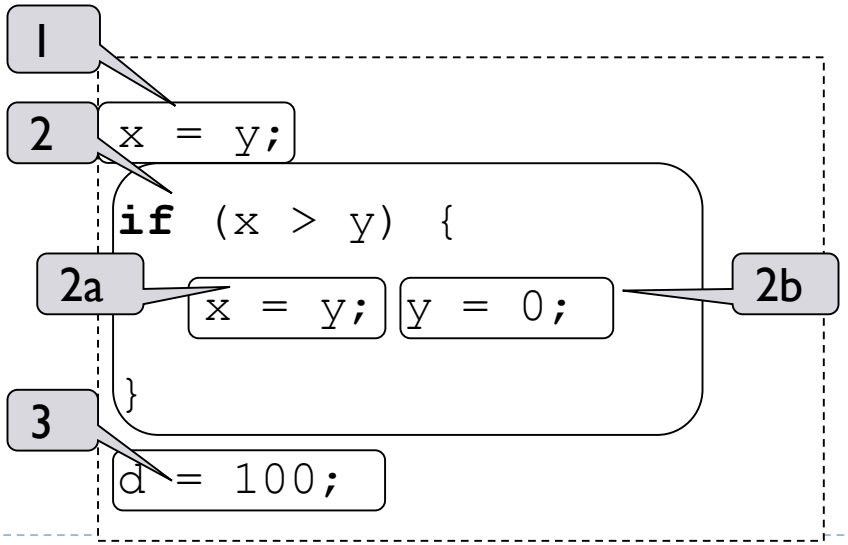
Разделяне на оператори



Ролята на ';'

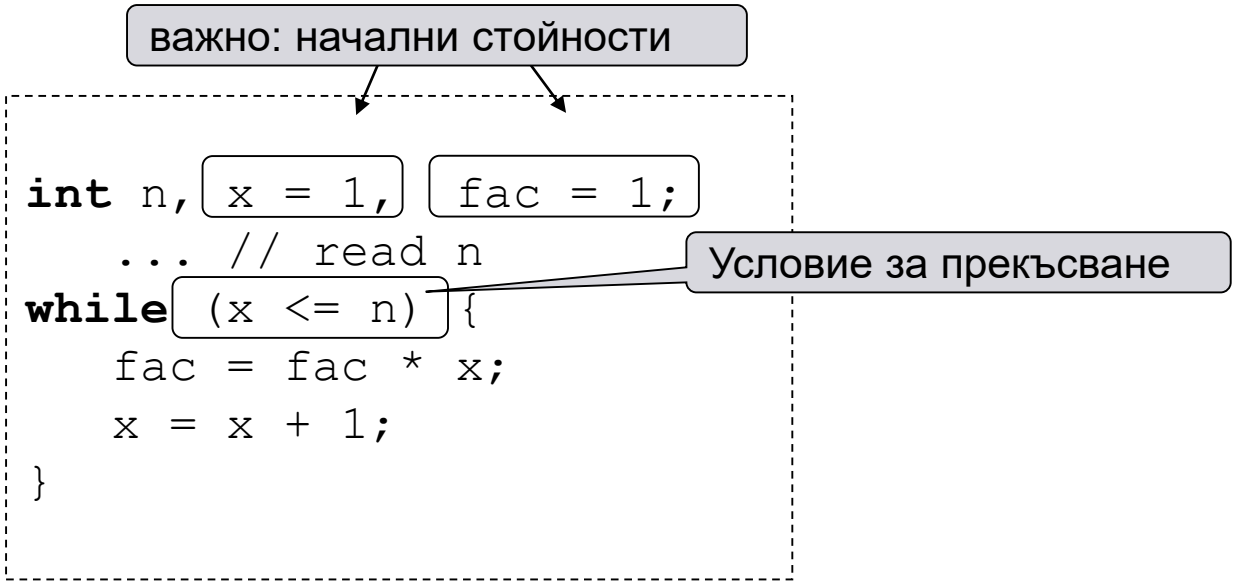
C, C++, **Java**:
- Край на оператори
  (синтактически – част от операторите)
- но: с много изключения
  (без ';' : while, if, ...)

→ "нечиста" езикова дефиниция
(неединен принцип: причина за грешки)



Пример: Функцията “!”

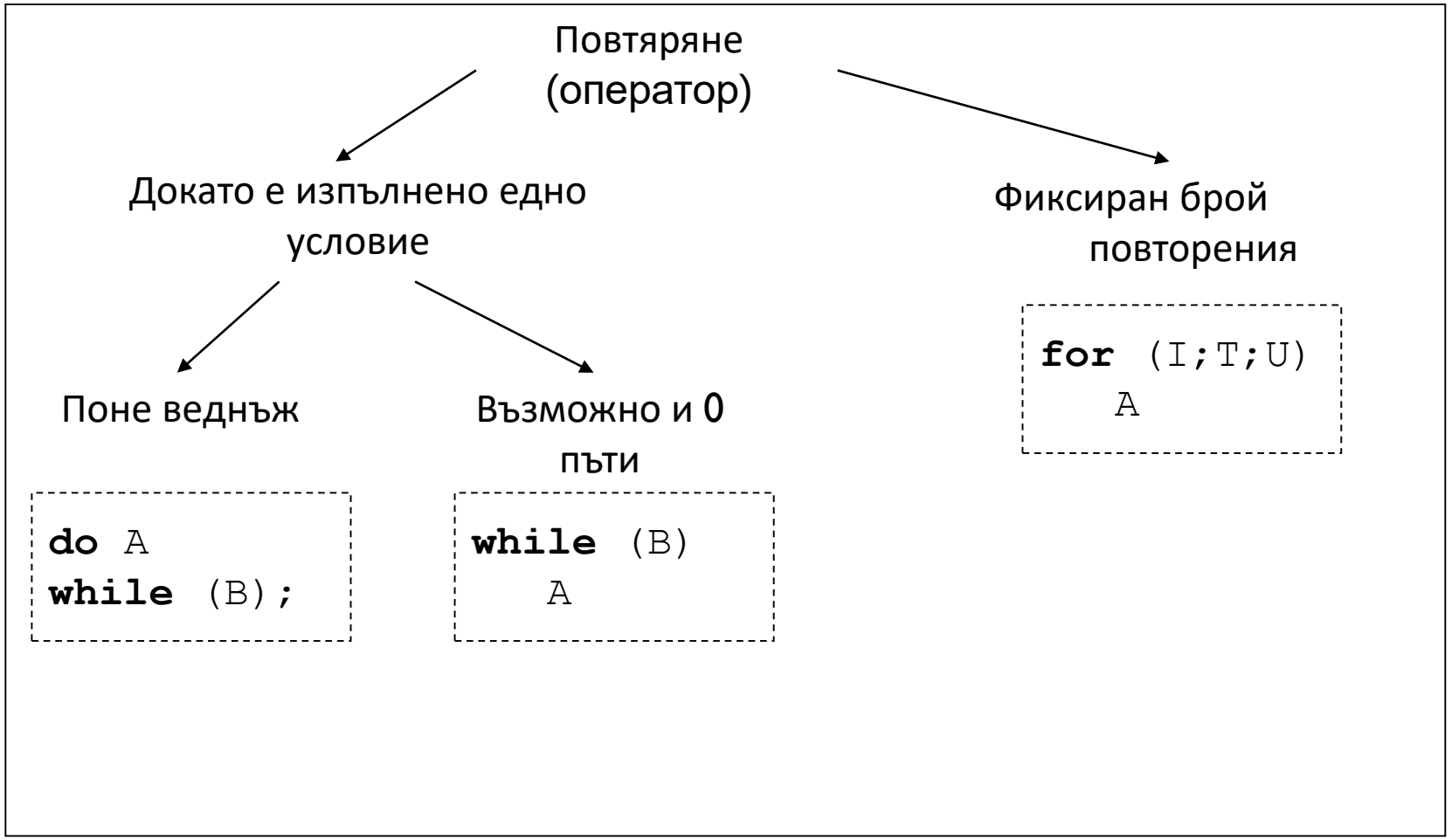
n! = 1 * 2 * . . . * n



напр. за n = 4:

- x получава стойностите: 1, 2, 3, 4, 5
  (при x = 5 не важи повече: x <= n)
- При това fac получава стойностите: 1, 1, 2, 6, 24

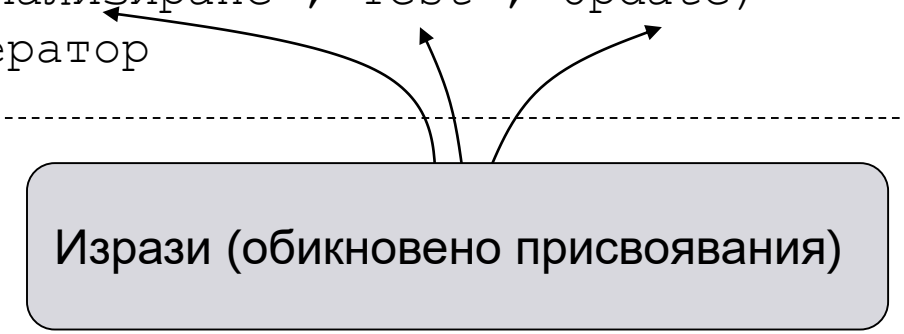
Избор на повтаряеми оператори



For-цикъл

EBNF:

```
for (инициализиране ; Test ; Update)
        оператор
```



Еквивалентно на:

```
инициализиране ;
while ( Test )    {
    оператор ;
    Update ;
}
```

Пример: ! с 'for'

```
int n, x, fac = 1;
   ... // read n
for (x = 1; x <= n ; x = x + 1)
   fac = fac * x;
```

Кратка форма: x++
Аналог на: x-- като x = x - 1

Приложение: броят на повторенията е познат (тук: n) !

For-оператор в Java (C, C++): от езиково-теоретичен аспект е неуспех

- Знаем броя на повторенията: n
- Променлива-брояч **i** преминава от една начална стойност (напр. 1) до позната крайна стойност (n)

For-оператор в Java (C, C++): възможност за непрегледни програми

```
for (a = 1 ; b * c == 0 ; k = k + 1)
        x = y;
```

→ променливата-брояч **a** няма отношение към условието за прекъсване и промяната

```
for ( ; ; ) {readNumber(); ... }
```

Съответства на:

```
while (true){
    readNumber();...
}
```

Цикли: следващ пример

Таблица:Трансформиране Celsius → Fahrenheit

Grad C	Grad F
-10.0	14.0
-9.0	15.8
-8.0	17.6
...	
10.0	50.0

→ Примерни програми с while и for оператор

с while-оператор

```
class TemperatureTable {
 // Table with C/F temperatures
   public static void main (String[] args) {
       final double
          LOW_TEMP = -10.0,
          HIGH_TEMP = 10.0;

       double
          cent, // degree Celsius
          fahr; // degree Fahrenheit

       System.out.println("\tGrad C\t\tGrad F");
       cent = LOW_TEMP;
       while (cent <= HIGH_TEMP) {
          fahr = (9.0/5.0) * cent + 32.0; // C -> F
          System.out.println("\t" + cent + "\t\t" + fahr);
          cent = cent + 1.0;
       }
   }
}
```

табулатор

cent++

с for-оператор

```
class TemperatureTable {
 // Table with C/F temperatures
   public static void main (String[] args) {
       final double
          LOW_TEMP = -10.0,
          HIGH_TEMP = 10.0;

       double
          cent, // Grad Celsius
          fahr; // Grad Fahrenheit

       System.out.println("\tGrad C\t\tGrad F");
       for(cent = LOW_TEMP; cent <=HIGH_TEMP; cent++) {
          fahr = (9.0/5.0) * cent + 32.0; // C -> F
          System.out.println("\t" + cent + "\t\t" + fahr);
       }
   }
}
```

Управление на цикъла: break оператор

```
while (true) {
    ... четене ...
    if (Keyboard.eof())
        break;
    ... обработка на входа ...
}
```

Напускане на цикъла без изпълнение на останалите конструкции.

Break-оператор

➔ Също без break:

```
... четене ...
while (! Keyboard.eof())
    ... обработка на входа ...
    ... четене ...
}
```

Управление на цикъла: continue оператор

- Спира изпълнението на текущата итерация
- Връща се в началото на цикъла
- Започва нова итерация

Демонстрация: break и continue

```
public class BreakAndContinue {
      public static void main(String[ ] args) {
              for (int i=0; i<100; i++) {
                    if (i==74) break;
                    if (i % 9 !=0) continue;
                    System.out.println(i);
               }
               int i=0;
               while (true) {
                    i++;
                    int j = i*27;
                    if (j == 1269) break;
                    if (i % 10 != 0) continue;
                    System.out.println(i);
                }
        }
}
```

Резултат от изпълнението ?

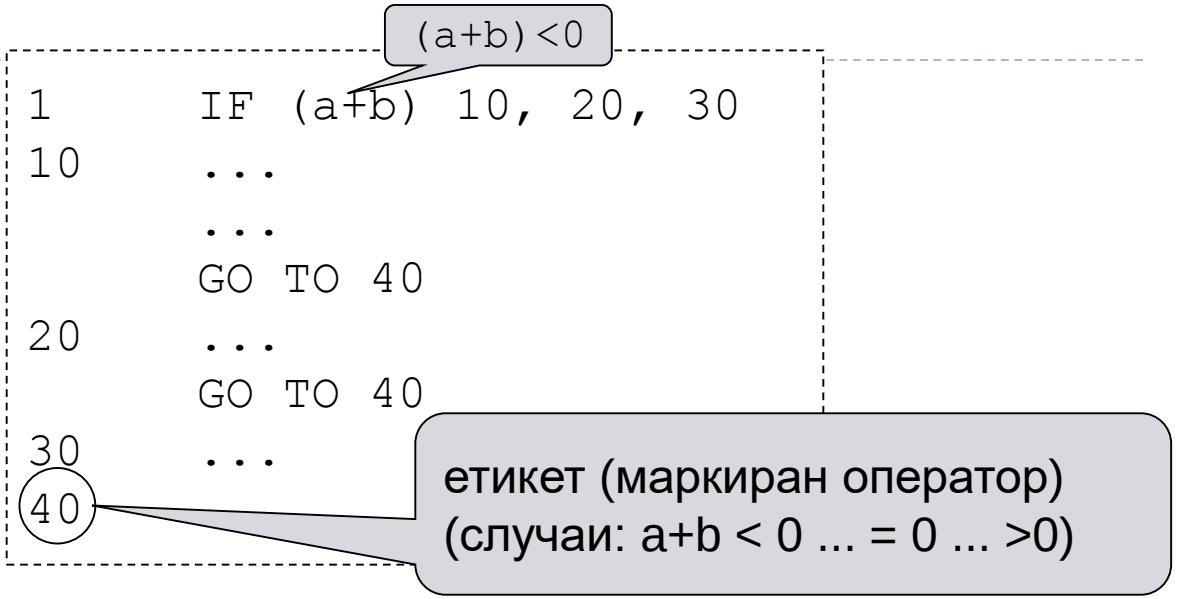
0
9
18
27
36
45
54
63
72
10
20
30
40

Безкрайни цикли

- while (true) { …
- for (;;) { …
  - Обработват се от компилатора по един и същи начин
  - Кой от двата варианта?
    - Предпочитание на програмиста

Оператор goto

Преди структурното програмиране: (Fortran)



Структурно програмиране:

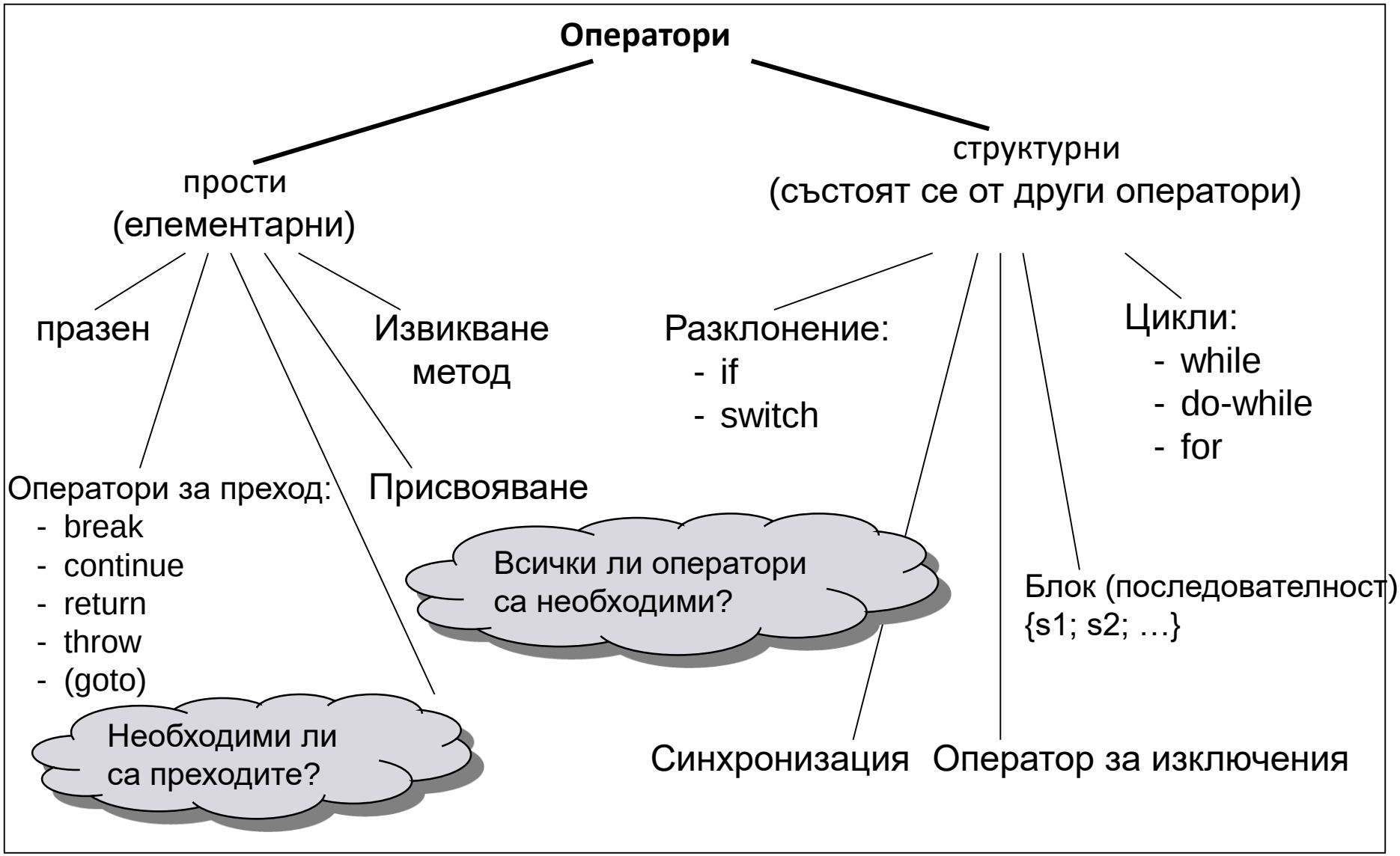
- Може без преходи
- Достатъчни за изграждане на алгоритми:
  - Присвояване
  - Последователно изпълнение (a1; a2; ...)
  - Разклонение (пълна форма на 'if')
  - Повторение (while)

В Java няма оператор goto!

Java и оператора goto

- James Gosling създава първоначалната JVM с поддръжка на оператора goto.
- В последствие премахва този оператор като ненужен, но goto остава в списъка със запазените думи.
- Причината е, че goto може да бъде заменен с използването на по-четими изрази като:
  - break/continue
  - отделяне на парче код като отделен метод
- James Gosling за Apple, Apache, Google, Oracle и бъдещето на Java (2010)
  - http://www.youtube.com/watch?v=9ei-rbULWoA&t=1m30s

Структурно програмиране: Класификация на операторите



Благодаря за вниманието!

Край лекция 5. “Итерации”