Операторите

Операторите

- **описват действията, изпълнявани от програмата;**
- **управляват последователността на изпълнение на програмата.**

Съставен оператор (блок)

Блок – последователност от оператори и декларации на променливи, заградени в скоби.

```
{
  <декларация на локални променливи>
  <оператор>
  ...
}
```

Оператори за деклариране на локални променливи

```
<тип> <идентификатор> = <израз>, ...
<тип> <идентификатор>[ ] = <инициализатор_на_масив>, ...
```

Празен оператор

```
;
```

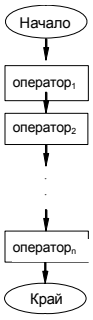
Базови управляващи структури

Управляваща структура е част от програмата, която може да се изпълнява.

1. **Верига**
2. **Разклонение**
3. **Цикъл**

Верига

Верига е линейна последователност от действия. Описва се чрез съставен оператор (блок).

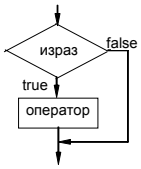



```
{
  <декларация на локални променливи>
  <оператор1>
  <оператор2>
  …
  <операторn>
}
```

Разклонение

Разклонение (алтернатива) е управляваща структура, при която изчислителният процес се разклонява, в зависимост от стойността на определено условие (израз).

1. Оператор if (с една алтернатива)

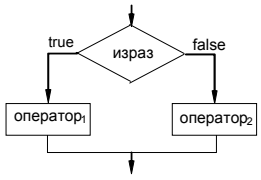



```
if (<израз>)
    <оператор>
```

<израз> – **тип** boolean

Първо се изчислява <израз>: ако има стойност true, се изпълнява <оператор> и if завършва; ако има стойност false – <оператор> не се изпълнява и if завършва.

2. Оператор if-else (с две алтернативи)

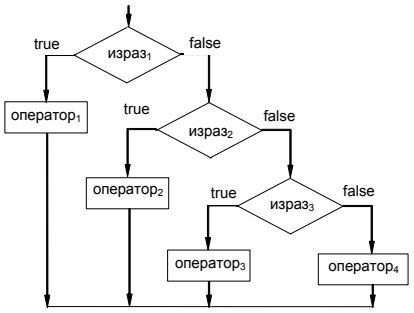



```
if (<израз>)
  <оператор1>
else
  <оператор2>
```

<израз> – тип boolean

Първо се изчислява <израз>: ако има стойност true, то се изпълнява <оператор1> и if-else завършва; ако има стойност false, се изпълнява <оператор2> и if-else завършва.

3. Оператор if-else-if




```
if (<израз1>)
   <оператор1>
else if(<израз2>)
   <оператор2>
else if(<израз3>)
   <оператор3>
else
   <оператор4>
```

Пример: Играч отгатва число, което се генерира с генератор на случайни числа.

```
// Оператор if
import java.io.*;
import java.util.Random;
class Bingo {
  public static void main(String[] args) throws IOException {
    Random generator = new Random();
    BufferedReader stdin;
    stdin = new BufferedReader(new InputStreamReader (System.in));
    int number, guess;
    number = generator.nextInt();
    System.out.print ("Отгатни числото: ");
    guess = Integer.parseInt(stdin.readLine());
    if (guess == number)
    System.out.println ("БИНГО!!! " + "Отгатнахте числото!");
  }
}
```

```
// оператор if-else
if (guess == number)
   System.out.println ("БИНГО!!! " + "Отгатнахте числото!");
else
   System.out.println("ГРЕШКА!!!");
```

```
// оператор if-else-if
if (guess == number)
   System.out.println ("БИНГО!!! " + "Отгатнахте числото!");
else if (guess > number)
   System.out.println ("Грешка, числото е много голямо!");
else
   System.out.println ("Грешка, числото е много малко!");
```

Клас Random **(**java.util**)**

Генератор на псевдослучайни числа.

public int nextInt()

Връща число в областта на int **стойностите.**

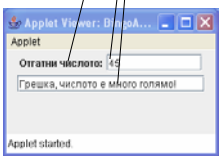
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
public class BingoApplet extends JApplet implements ActionListener {
  JLabel label = new JLabel("Отгатни числото:");
  JTextField input = new JTextField(10);
  JTextField output = new JTextField(20);
  int number, guess;
  public void init() {
    setSize (250,100);
    Container container = getContentPane();
    container.setLayout(new FlowLayout());
    container.add(label);
    container.add(input);
    container.add(output);
    input.addActionListener(this);
    Random generator = new Random();
    number = generator.nextInt();
  }
```

Самият аплет е слушател

JLabel label

JTextField input

JTextField output




```
  public void actionPerformed(ActionEvent e) {
    guess = Integer.parseInt(input.getText());
    if (guess == number)
      output.setText("БИНГО!!! " + "Отгатнахте числото!");
    else if (guess > number)
      output.setText ("Грешка, числото е много голямо!");
    else
      output.setText ("Грешка, числото е много малко!");
  }
}
```

Извиква се автоматично при натискане на <Enter> в input

Клас JApplet **(**javax.swing**)**

public void init()

Инициализира аплета. Извиква се от браузъра или applet viewer, за да информира, че аплетът е зареден в системата.

Клас JLabel **(**javax.swing**)**

Изобразява област за къс текст, изображение или и двете. Не реагира на входни събития.

public JLabel(String text)

Създава етикет с определен текст text**.**

Клас JTextField (javax.swing)

Представя текстово поле. При натискане на <Enter> се вдига събитието ActionEvent, което се регистрира чрез метода addActionListener и се обработва автоматично от метода actionPerformed.

public JTextField(String text)

Създава текстово поле, инициализирано с определен текст text.

public void addActionListener(ActionListener l)

Добавя слушател, за да приеме събитията от текстовото поле.

void actionPerformed(ActionEvent e)

Извиква се автоматично, когато се случи събитието.

public Container getContentPane()

Връща съдържанието на аплета, т.е. мястото, където се разполагат графичните компоненти.

public void setLayout(LayoutManager mgr)

Установява менажера за разполагане на компонентите в контейнера.

Клас FlowLayout (java.awt)

Реализира менажер като разполага компонентите последователно (както текстови линии в параграф).

Пример: Корени на квадратно уравнение

$$ax^2 + bx + c = 0$$

$a = 0, b = 0$ *изродено уравнение*

$a = 0, b \neq 0$ *линейно уравнение* $x = -\frac{c}{b}$

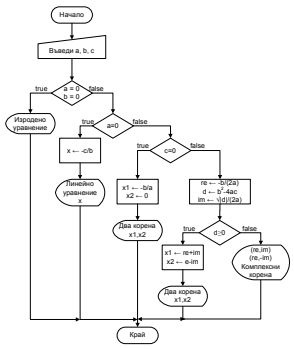
$a \neq 0, b \neq 0, c = 0$ 2 *корена* $x_1 = -\frac{b}{a}, x_2 = 0$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$d = b^2 - 4ac$

$d \geq 0$ 2 *реални корена*

$d < 0$ 2 *комплексни корена*



4. Оператор switch-case (многозначно разклонение)


```
switch (<израз>)
{
  case <константа1>:
       <оператор1>
       break;
  case <константа2>:
       <оператор2>
       break;
  .
  .
  case <константаn>:
       <операторn>
       break;
  default:
       <операторn+1>
       break;
}
```

израз – тип char**,** byte**,** short **или** int

Първо се изчислява <израз>.

- **ако има <константанта$_i$>, равна на стойността на <израз>, изпълняват се последователно всички оператори след <константанта$_i$>;**
- **ако нито една <константанта$_i$> не е равна на стойността на <израз>, то се изпълняват операторите след клаузата default;**
- **ако някой <оператор$_i$> съдържа оператор break без етикет, switch завършва веднага;**
- **ако някой <оператор$_i$> съдържа оператор break с етикет, управлението се предава на оператора с този етикет.**

Пример: Брой дни в месец в зависимост дали годината е високосна

```
import java.io.*;
class Days {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int month, year, number = 0;
    System.out.print ("Въведи месец: ");
    month = Integer.parseInt(stdin.readLine());
    System.out.print ("Въведи година: ");
    year = Integer.parseInt(stdin.readLine());
```

```
    switch (month) {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12: number = 31;
               break;
      case 4:
      case 6:
      case 9:
      case 11: number=30;
               break;
      case 2: if ((year%4 == 0) && !(year % 100 == 0) || (year % 400 == 0))
                number = 29;
              else
                number = 28;
              break;
    }
```

```
    System.out.print("Месец ");
    switch (month) {
      case 1: System.out.print("януари"); break;
      case 2: System.out.print("февруари"); break;
      case 3: System.out.print("март"); break;
      case 4: System.out.print("април"); break;
      case 5: System.out.print("май"); break;
      case 6: System.out.print("юни"); break;
      case 7: System.out.print("юли"); break;
      case 8: System.out.print("август"); break;
      case 9: System.out.print("септември"); break;
      case 10: System.out.print("октомври"); break;
      case 11: System.out.print("ноември");  break;
      case 12: System.out.print("декември"); break;
    }
    System.out.println(" от " + year + " г. има " + number + " дни.");
  }
}
```

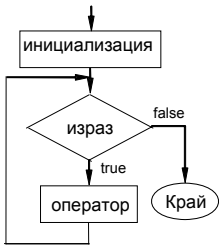
Резултати:

```
Въведи месец: 2
Въведи година: 2000
Месец февруари от 2000 г. има 29 дни.
```

Цикъл

Цикъл е последователност от инструкции, която се повтаря многократно, докато се достигне дадено условие.

1. Оператор за цикъл с предусловие while



while (<израз>)
  <оператор>

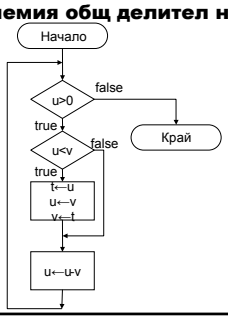
<израз> – boolean

Първо се изчислява <израз>. Докато неговата стойност е true, изпълнява се <оператор>, като <израз> се преизчислява. Когато неговата стойност стане false, while завършва.

Алгоритъм на Евклид за намиране на най-големия общ делител

Приложение – при съкращаване на дроби.

Ако u>v, то най-големият общ делител е същият като най-големия общ делител на v и u-v.



Пример: Съкращаване на дроби

```
import java.io.*;
public class Gcd {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    System.out.print ("Въведи числител: ");
    int nom = Integer.parseInt(stdin.readLine());
    System.out.print ("Въведи знаменател: ");
    int denom = Integer.parseInt(stdin.readLine());
    int u = nom;
    int v = denom;
    while (u > 0) {
      if (u < v) {
        int t = u;          // u и v си разменят стойностите
        u = v;
        v = t;
      }
      u = u - v;
    }
    System.out.println(nom + "/" + denom + " = " + nom/v + "/" + denom/v);
  }
}
```

Резултати:

```
Въведи числител: 120
Въведи знаменател: 100
120/100 = 6/5
```

Пример: Въвежда последователност от цели числа, която се прекъсва чрез въвеждане на числото 999 и изчислява тяхната сума.

Алгоритъм:

```
сума ← 0
въведи число
докато число ≠ 999 изпълнявай
  добави число към текущата сума
  въведи число
отпечатай сума
```

```
import java.io.*;
public class Suma {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int sum = 0;
    System.out.println("Въведи последователност от числа (за край 999): ");
    int number = Integer.parseInt(stdin.readLine());
    while (number != 999) {
      sum += number;
      number = Integer.parseInt(stdin.readLine());
    }
    System.out.println("Сума = " + sum);
  }
}
```

Резултати:

```
Въведи последователност от числа (за край 999):
5
7
8
999
Сума = 20
```

Пример: Въвежда последователност от точки (за край -1). Изчислява средния, максималния и минималния брой точки.

Алгоритъм:

```
сума ← 0
брой ← 0
въведи точки
максимум ← точки
минимум ← точки
докато точки ≠ -1 изпълнявай
  добави точки към текущата сума
  увеличи брой
  ако точки > максимум
    максимум ← точки
  ако точки < минимум
    минимум ← точки
  въведи точки
ако брой ≠ 0
  среден брой точки ← сума / брой
  отпечатай среден брой точки, максимум, минимум
в противен случай
  отпечатай „Не са въведени точки“
```

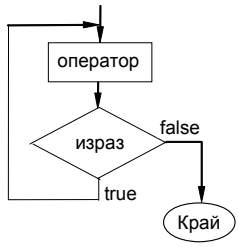
```
import java.io.*;
public class Average {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int sum = 0;
    int count = 0;
    System.out.println("Въведи последователност от точки (за край -1): ");
    int points = Integer.parseInt(stdin.readLine());
    int max = points;
    int min = points;
    while (points != -1) {
      sum += points;
      count++;
      if (points > max)
        max = points;
      if (points < min)
        min = points;
      points = Integer.parseInt(stdin.readLine());
    }
```

```
    if (count != 0) {
      float average = (float)sum / count;
      System.out.println("Общ брой студенти = " + count);
      System.out.println("Среден брой точки = " + average);
      System.out.println("Максимален брой точки = " + max);
      System.out.println("Минимален брой точки = " + min);
    }
    else
      System.out.println("Не са въведени оценки!!!");
  }
}
```

Резултати:

```
Въведи последователност от точки (за край -1):
56
76
82
43
65
-1
Общ брой студенти = 5
Среден брой точки = 64.4
Максимален брой точки = 82
Минимален брой точки = 43
```

2. Оператор за цикъл с постусловие do-while




```
do
   <оператор>
while (<израз>);
```

<израз> – **тип** boolean

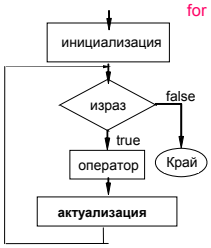
Първо се изпълнява <оператор>. След това се изчислява <израз>: ако е true, do-while се изпълнява отново; ако е false, do-while завършва.

Пример: Проверка за коректно въведени данни

```
BufferedReader stdin;
stdin = new BufferedReader (new InputStreamReader (System.in));
int number;
do {
   System.out.println ("Въведи число между 1 и 100: ");
   number = Integer.parseInt(stdin.readLine());
}
while (number<1 || number>100);

String word;
do {
   System.out.println("Въведи дума (за край Край)");
   word = stdin.readLine();
   // Обработка …
}
while (! word.equals("Край"));
```

3. Оператор за цикъл for




```
for (<инициализация>; <израз>; <актуализация>
     <оператор>
```

Изпълнява се код за <инициализация>, след това се изчислява <израз>: ако е true, се изпълнява <оператор>, <актуализация> и отново се изчислява <израз>; ако е false, for завършва.

Пример: Сумата от n реални числа, n≤10.

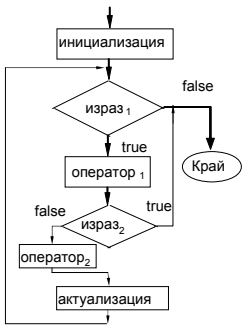
```
import java.io.*;
public class Suma {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int n;
    float number, sum;
    sum = 0.0f;
    do {
      System.out.print ("Въведете n в интервала [1, 10]: ");
      n = Integer.parseInt(stdin.readLine());
    }
    while (n < 1 || n > 10);
    System.out.println ("Въведете " + n + " реални числа:");
    for (int i=1; i<=n; i++) {
      number = Float.parseFloat(stdin.readLine());
      sum += number;
    }
    System.out.println ("Сума = " + sum);
  }
}
```

4. Оператор break

```
break;
break <етикет>;
```

<етикет> – **идентификатор**

break **предава управлението навън от оператора, в който е поместен.** break **без етикет се използва само в операторите** switch, while, do-while **и** for**.** break <етикет> **предава управлението на оператор с този** <етикет> **(не е необходимо да бъде** switch, while, do-while **или** for**).**




```
for (<инициализация>;
     <израз1>;
     <актуализация>)
{
  <оператор1>
  if (<израз2>)
    break;
  <оператор2>
}
```

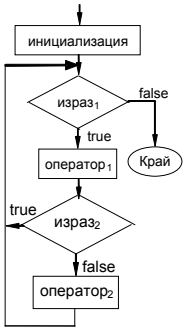
Пример: Въвежда последователност от цели числа, която се прекъсва чрез въвеждане на числото 999 и изчислява тяхната сума.

```
import java.io.*;
public class Suma {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int sum = 0;
    System.out.println("Въведи последователност от числа (за край 999): ");
    for (; ;) {
      int number = Integer.parseInt(stdin.readLine());
      if (number == 999)
        break;
      sum += number;
    }
    System.out.println("Сума = " + sum);
  }
}
```

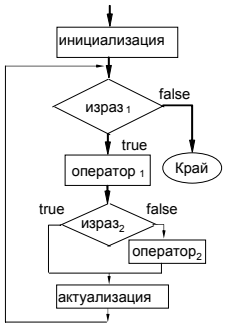
5. Оператор continue **за завършване итерация на цикъл**

continue;
continue <етикет>;
<етикет> – **идентификатор**

continue **се използва само в оператори** while, do-while **и** for**. Прекъсва изпълнението на текущата итерация и предава управлението: за** while **и** do-while **към проверка на условието за край (**<израз>**), а за** for **– към изчисление на** <актуализация>**.** continue **без етикет предава управлението на най-вътрешния оператор за цикъл, който завършва незабавно текущата итерация и започва нова.** continue <етикет> **предава управлението на оператор с този** <етикет>**.**




```
<инициализация>
while (<израз1>)
{
  <оператор1>
  if (<израз2>)
    continue;
  <оператор2>
}
```




```
for (<инициализация>;
      <израз1>;
      <актуализация>)
{
  <оператор1>
  if (<израз2>)
    continue;
  <оператор2>
}
```

Пример: Сумата на положителните от n реални числа, n≤10.

```
import java.io.*;
public class Suma {
  public static void main(String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int n;
    float sum = 0.0f;
    do {
      System.out.print ("Въведете n в интервала [1, 10]: ");
      n = Integer.parseInt(stdin.readLine());
    }
    while (n < 1 || n > 10);
    System.out.println ("Въведете " + n + " реални числа:");
    for (int i=1; i<=n; i++) {
      float number = Float.parseFloat(stdin.readLine());
      if (number <= 0.0f)
        continue;
      sum += number;
    }
    System.out.println ("Сума от положителните числа = " + sum);
  }
}
```

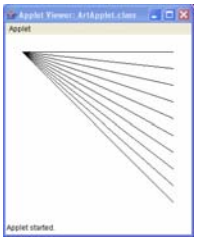
```
for (int i=1; i<=n; i++) {
  float number = Float.parseFloat(stdin.readLine());
  if (number >= 0.0f)
    sum += number;
}
```

Пример: Използване на цикли при чертане

```
import javax.swing.JApplet;
import java.awt.*;
public class ArtApplet extends JApplet {
  private final int MAX_WIDTH = 300;
  private final int NUM_RINGS = 5;
  private final int RING_WIDTH = 25;
  public void init() {
    setSize(300, 300);
  }
```

```
  public void paint(Graphics g) {
    int x = 0, y = 0, diameter;
    setBackground(Color.CYAN);
    diameter = MAX_WIDTH;
    g.setColor(Color.WHITE);
    for (int count = 0; count < NUM_RINGS; count++) {
      if (g.getColor() == Color.BLACK)
        g.setColor(Color.WHITE);
      else
        g.setColor(Color.BLACK);
      g.fillOval(x, y, diameter, diameter);
      diameter -= (2 * RING_WIDTH);
      x += RING_WIDTH;
      y += RING_WIDTH;
    }
    g.setColor(Color.RED);
    g.fillOval(x, y, diameter, diameter);
  }
}
```

Пример: Използване на цикли при чертане



```
import javax.swing.JApplet;
import java.awt.*;
public class ArtApplet extends JApplet {
  public void init() {
    setSize(300, 300);
  }
  public void paint(Graphics g) {
    // Мащабира изображението
    Graphics2D g2 = (Graphics2D)g;
    g2.scale(getWidth()/11, getHeight()/11);
    g2.setStroke(new BasicStroke(1.0f/getWidth()));
    for (int line = 1; line <= 10; line++)
      g2.drawLine(1, 1, 10, line);
  }
}
```

Клас Graphics2D **(**java.awt**) – наследява класа** Graphics**, като координатната система не зависи от устройството.**

public abstract void scale(double sx, double sy)

Извършва мащабиране – умножава x координатите със sx **и y координатите със** sy**.**

public abstract void setStroke(Stroke s)

Установява щриха Stroke **за Graphics2D съдържанието.**

Клас BasicStroke **(**java.awt**) дефинира атрибутите за графичните примитиви с** Graphics2D **обект.**

public BasicStroke(float width)

Конструира плътен щрих с определена широчина width **на линията.**