



# ОПЕРАТОРЫ ЯЗЫКА ВЫСОКОГО УРОВНЯ PASCAL

*Ясность – это не атрибут письма,  
ясность – это само письмо.*

*П. Буаст*

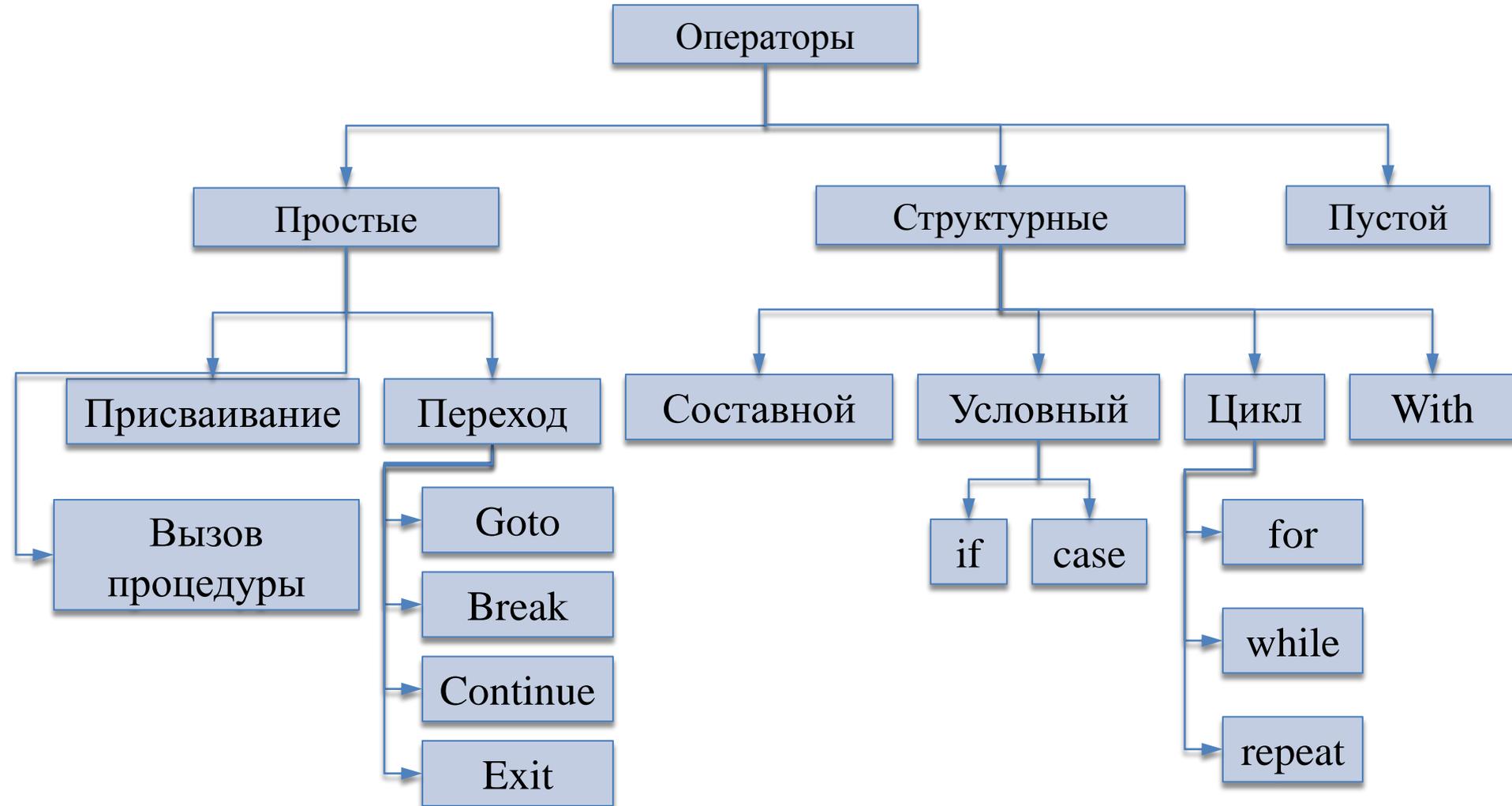
# Содержание

2

- Классификация операторов
- Примеры использования операторов
- Понятие структурного программирования

# Классификация операторов

3



# Пустой оператор

4

- Не обозначается и не вызывает никаких действий (дополнительная точка с запятой в тексте программы). Например: `if A>5 then ; ; ;`
- В целях унификации рекомендуется добавлять пустой оператор как последний оператор составного оператора. Например:  
`if A>5 then begin  
    Z:=3;  
    C:=i+1; { здесь разделитель не обязателен }  
end;`

# Оператор присваивания

5

- `<Переменная> := <Выражение>`
  - ▣ замена текущего значения переменной новым;
  - ▣ определение значения, возвращаемого функцией
- Типы переменной и выражения должны быть **совместимы по присваиванию**.
- Примеры:  
A:=A+3; Ch:='S'; S:='Stop'; K:=8.5;  
Stop := (Z<10) and (i>3); Z:=Sqr(i)\*A+K;  
M[i, j+10, k mod 3] := Z+A;

# Оператор вызова процедуры

6

- `<Имя процедуры>( <Список фактических параметров>)`

Активизирует процедуру с указанным именем, передавая формальным параметрам значения соответствующих фактических параметров.

- Примеры:

```
ClrScr;
```

```
InputData(A, B);
```

```
Calculate(A, B, 23, Result);
```

```
OutputData(Result);
```

# Составной оператор

7

- begin

  - <Список операторов>

  - end

Допустим везде, где допустим один оператор.

- Пример:

  - if  $A > 5$  then

  - begin

    - $C := Z - 10;$

    - $i := j + 2;$

  - end;

# Условный оператор **if**

8

- **if** <Условие> **then**  
    <Оператор1>  
[**else**  
    <Оператор2>];
- Ключевое слово **else**  
всегда сопоставляется ближайшему  
предшествующему и еще не  
сопоставленному ключевому слову  
**then**.

```
if C1 then  
    if C2 then S1 else S2;
```



```
if C1 then  
begin  
    if C2 then  
        S1  
    else  
        S2  
end;
```

# Условный оператор **case**

9

- **case** <Селектор> **of**  
    <Конст11> [.. <Конст12>] : <Оператор1>;  
    <Конст21> [.. <Конст22>] : <Оператор2>;  
    ...  
    [**else**  
        <ОператорN>]  
    **end**
- Селектор должен иметь формат байта или слова. Все константы выбора должны быть уникальны и принадлежать ординальному типу, совместимому с типом селектора.

# Пример: использование оператора **case**

10

**case** ASuit **of**

Spades: WriteLn('Пики');

Clubs : WriteLn('Крести');

Diamonds: WriteLn('Бубны');

Hearts : WriteLn('Черви');

**end;**

**case** HexCh **of**

'0'..'9': N:=Ord(HexCh)-Ord('0');

'a'..'f', 'A'..'F': N:=Ord(Uppcase(HexCh))-Ord('A')+10;

**else**

WriteLn(HexCh, 'не является 16-ричной цифрой!');

**end;**

# Оператор цикла `while`

11

- **while** <Условие> **do**  
    <Оператор>;
- Цикл **while** выполняется *пока* <Условие> истинно.  
Если <Условие> сразу ложно, то цикл не выполняется ни разу.
- Цикл **while** используется, когда заранее не известно число повторений.

# Пример: использование оператора **while**

12

{Вычисление  $r=y \bmod x$ ,  $d=y \operatorname{div} x$  }

$r:=y$ ;

$d:=0$ ;

**while**  $r \geq x$  **do**

**begin**

$r:=r-x$ ;

$d:=d+1$ ;

**end**

# Оператор цикла **repeat**

13

## □ **repeat**

<Оператор>

**until** <Условие>;

- Цикл **repeat** заканчивается, когда <Условие> истинно. Цикл **repeat** выполняется как минимум один раз.
- Цикл **repeat** используется, когда заранее не известно число повторений.

# Пример: использование оператора `repeat`

14

{Ввод 16-ричной цифры с клавиатуры }

**repeat**

    Write('Введите 16-ричную цифру: ');

    ReadLn(HexCh);

**until** HexCh in ['0'..'9', 'A'..'F', 'a'..'f'];

# Оператор цикла for

15

- **for** <Счетчик>:=<Начало> **to** <Конец> **do**  
    <Оператор>
- Объекты цикла
  - ▣ *счетчик, начало, конец* должны иметь ординальный тип
  - ▣ *начало, конец* могут быть выражениями; их вычисление происходит один раз перед первым выполнением цикла.
- Выполнение цикла
  - ▣ для каждого значения *счетчика* из интервала *начало..конец*
  - ▣ значение счетчика цикла при каждой итерации возрастает на один номер
  - ▣ если начальное значение больше конечного, цикл не выполняется ни разу.
- Цикл **for** с **downto** вместо **to** выполняется в обратном порядке; конечное значение не должно превосходить начальное.
- **Не рекомендуется изменять значение переменной цикла внутри цикла.**

# Пример: использование оператора **for**

16

```
const
```

```
    N=3;
```

```
    M=3;
```

```
type
```

```
    Matrix=array [1..N, 1..M] of Real;
```

```
var A, B, C: Matrix;
```

```
{ Сумма матриц  $C_{n \times m} = A_{n \times m} + B_{n \times m}$  }
```

```
for i:=1 to N do
```

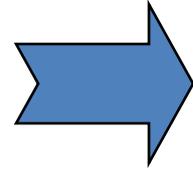
```
    for j:=1 to M do
```

```
        C[i, j]:=A[i, j]+B[i, j];
```

# Эквивалентность операторов цикла

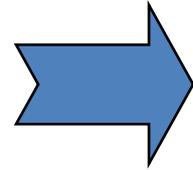
17

□ **while** C **do**  
    S;



● **if** C **then**  
    repeat  
        S  
    **until not** C;

□ **for** i:=Start **to** Stop **do**  
    S;



● i:=Start;  
  **while** i<=Stop **do**  
  begin  
    S;  
    i:=Succ(i);  
  **end**;

# Оператор присоединения **with**

18

- **with** <Ссылка на запись или объект> **do**  
    <Оператор>
- Позволяет ссылаться на поля записи (поля и методы объекта) без указания имени записи (объекта).
- Пример:  
    var Person: TPerson;  
    with Person do begin  
        FirstName:='Владимир';  
        LastName:='Путин';  
    end;

# Операторы перехода

19

- **Goto** <Метка>;
  - ▣ <Метка> представляет собой целое без знака или идентификатор (только Турбо Паскаль), определяется в секции **Label** того же блока и находится в том же блоке, что и оператор **GoTo**.
- Операторы **break** и **continue** (только Турбо Паскаль) – "синтаксический сахар" оператора **Goto**:
  - ▣ **break** досрочно прекращает выполнение цикла;
  - ▣ **continue** досрочно начинает очередную итерацию цикла.

# Пример: использование операторов перехода

20

```
Label 1, 2, 3;  
begin  
  1: ;  
  S1;  
  if C1 then  
    Goto 1;  
  S2;  
  2: ;  
  if C2 then  
    Goto 3  
  else begin  
    S3;  
    Goto 2;  
  end;  
  3: ;  
end.
```

```
begin  
  repeat  
    S1;  
  until not C1;  
  
  S2;  
  
  while not C2 do  
    S3;  
  
end.
```

```
while C1 do begin  
  S1;  
  if C2 then  
    break;  
  S2;  
end;
```

```
while C1 do  
begin  
  S1;  
  if C2 then  
    continue;  
  S2;  
end;
```

```
Label 1;  
while C1 do begin  
  S1;  
  if C2 then  
    Goto 1;  
  S2;  
end;  
1: ;
```

```
Label 1;  
1: ;  
while C1 do  
begin  
  S1;  
  if C2 then  
    Goto 1;  
  S2;  
end;
```

# Пример: использование операторов перехода

21

```
while C1 do
begin
  S1;
  if C2 then
    break;
  S2;
end;
```

```
Label 1;
while C1 do
begin
  S1;
  if C2 then
    Goto 1;
  S2;
end;
1: ;
```

```
if C1 then
  S1;
while C1 and C2 do
begin
  S1;
  if not C2 then
    S2;
end;
```

```
while C1 do
begin
  S1;
  if C2 then
    continue;
  S2;
end;
```

```
Label 1;
while C1 do
begin
  1: ;
  S1;
  if C2 then
    Goto 1;
  S2;
end;
```

```
while C1 do
begin
  if C2 then
    S1
  else begin
    S1;
    S2;
  end;
end;
```

# Структурное программирование

22

- *Структурное программирование* – построение программы с использованием *только* следующих конструкций:
  - следование (операторы присваивания и вызова процедуры);
  - ветвление (условный оператор);
  - повторение (оператор цикла).
- Структурная программа *не должна* использовать *оператор безусловного перехода*.

# Теорема о структурном программировании

- Для любой неструктурной программы существует эквивалентная ей структурная программа.
  - Две программы эквивалентны, если для любых входных данных результаты их работы совпадают (выдают одинаковые выходные данные, завершаются по одной и той же ошибке времени выполнения или зависают).
  - Для любой (не обязательно осмысленной!) программы с операторами Goto существует эквивалентная ей программа без операторов Goto.
- Существует *конструктивное* доказательство данной теоремы (указывающее алгоритм построения эквивалентной структурной программы).  
*Попробуйте доказать ее самостоятельно.*

# Операторы Goto: недостатки

24

- Не являются необходимыми
- Затрудняют чтение программы
- Затрудняют понимание программы
- Затрудняют формальное доказательство правильности программы

# Пример: программы с Goto

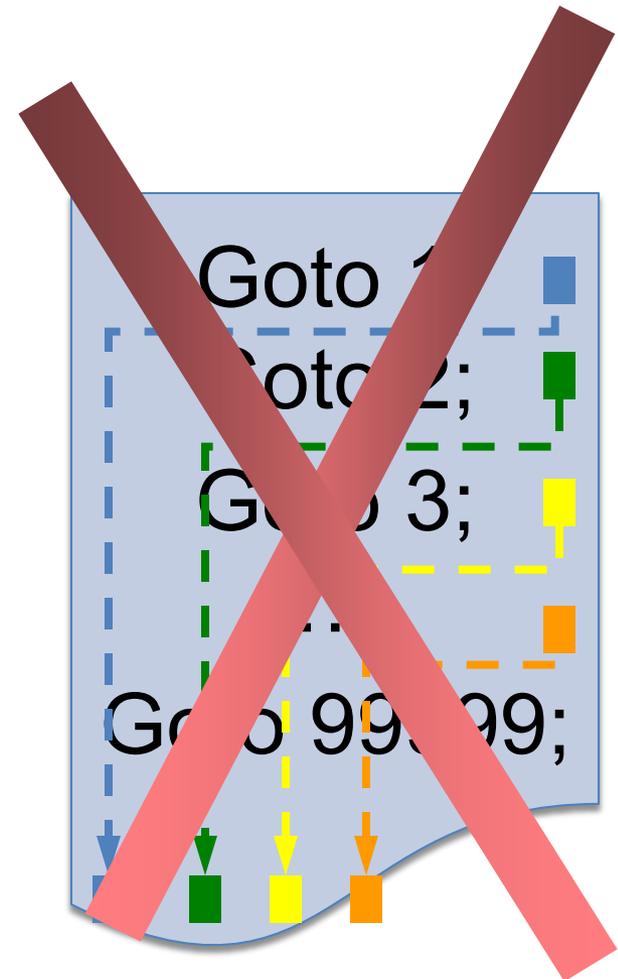
25

```
if not C then  
  Goto 2;  
1: S;  
if C then  
  Goto 1;  
2: ;
```

```
while C do S;
```

```
if C1 then Goto 1;  
if C2 then Goto 2;  
if C3 then Goto 3;  
1: S1; Goto 4;  
2: S2; Goto 4;  
3: S3;  
4: ;
```

```
if C1 then  
  S1  
else  
  if C2 then  
    S2  
  else  
    S3;
```



# Операторы Goto: достоинства

26

- Пример: линейный поиск элемента в массиве.

Label

    NotFound, Found;

for i:=1 to N do

    if A[i]=X then

        Goto Found;

NotFound: ... { не нашли };

Found: ... { нашли };

i:=1;

while (i<=N) and (A[i]<>X) do

    i:=i+1;

if i>N then;

    ... { не нашли }

else

    ... { нашли };

# Операторы Goto: достоинства

27

- Пример: линейный поиск элемента в матрице.

```
procedure FindElement(A: Matrix; E: Integer; var i, j: Integer);
begin
  for i:=1 to i<N do
    for j:=1 to M do
      if A[i,j]=E then
        goto end_loop;
  i:=0;
  j:=0;
  end_loop: ;
end;
```

# Операторы Goto: достоинства

28

## □ Пример: исключительная ситуация.

```
if not C1 then
  ErrorCode:=-1
else
  if not C2 then
    ErrorCode:=-2
  else
    if not C3 then
      ErrorCode:=-3
    else
      if not C4 then
        ErrorCode:=-4
      else begin
        { Основной алгоритм }
      end;
```

```
if (not C1) or (not C2) or (not C3) or (not C4) then
  Goto Error;
{ Основной алгоритм }
Goto Done;
Error:
if not C1 then
  ErrorCode:=-1
else
  if not C2 then
    ErrorCode:=-2
  else
    if not C3 then
      ErrorCode:=-3
    else
      if not C4 then
        ErrorCode:=-4;
Done: ;
```

# Операторы Goto: достоинства

29

## □ Пример: логика алгоритма.

```
function F(...): ...;  
begin  
  S1;  
  S2;  
  if C1 then begin  
    S3;  
    if C2 then begin  
      F:=V1;  
      Goto Done;  
    end  
  else  
    S4;  
end;  
S5;
```

```
  if C3 then begin  
    F:=V2;  
    Goto Done;  
  end;  
  S6;  
  if C3 then begin  
    F:=V3;  
    Goto Done;  
  end;  
  S7;  
  F:=V4;  
Done: ;  
end;
```

# Структурное программирование vs структурированные программы

- Структурное программирование не может улучшить плохо структурированную программу
  - механическая замена операторов Goto на структурные не может повысить ясность и читаемость, улучшить логику плохо спроектированной программы.
- Структурное программирование следует безусловно применять, *но* не ценой ухудшения структуры программы
  - полный отказ, равно как и безусловное использование операторов Goto может понизить ясность, читаемость и ухудшить логику программы

# Заключение

- Операторы делятся на простые и структурные.
- Простые операторы: оператор присваивания, оператор вызова процедуры, оператор перехода Goto.
- Структурные операторы: составной оператор, условные операторы (if, case), операторы цикла (while, repeat, for), оператор with.
- Структурное программирование – программирование с использованием структурных конструкций (следование, ветвление, повторение).
  - Теорема о структурном программировании: для любой программы с операторами Goto существует эквивалентная структурная программа.
  - Следует избегать использования оператора Goto, но не ценой потери ясности и читаемости программы.

# Теорема о структурном программировании

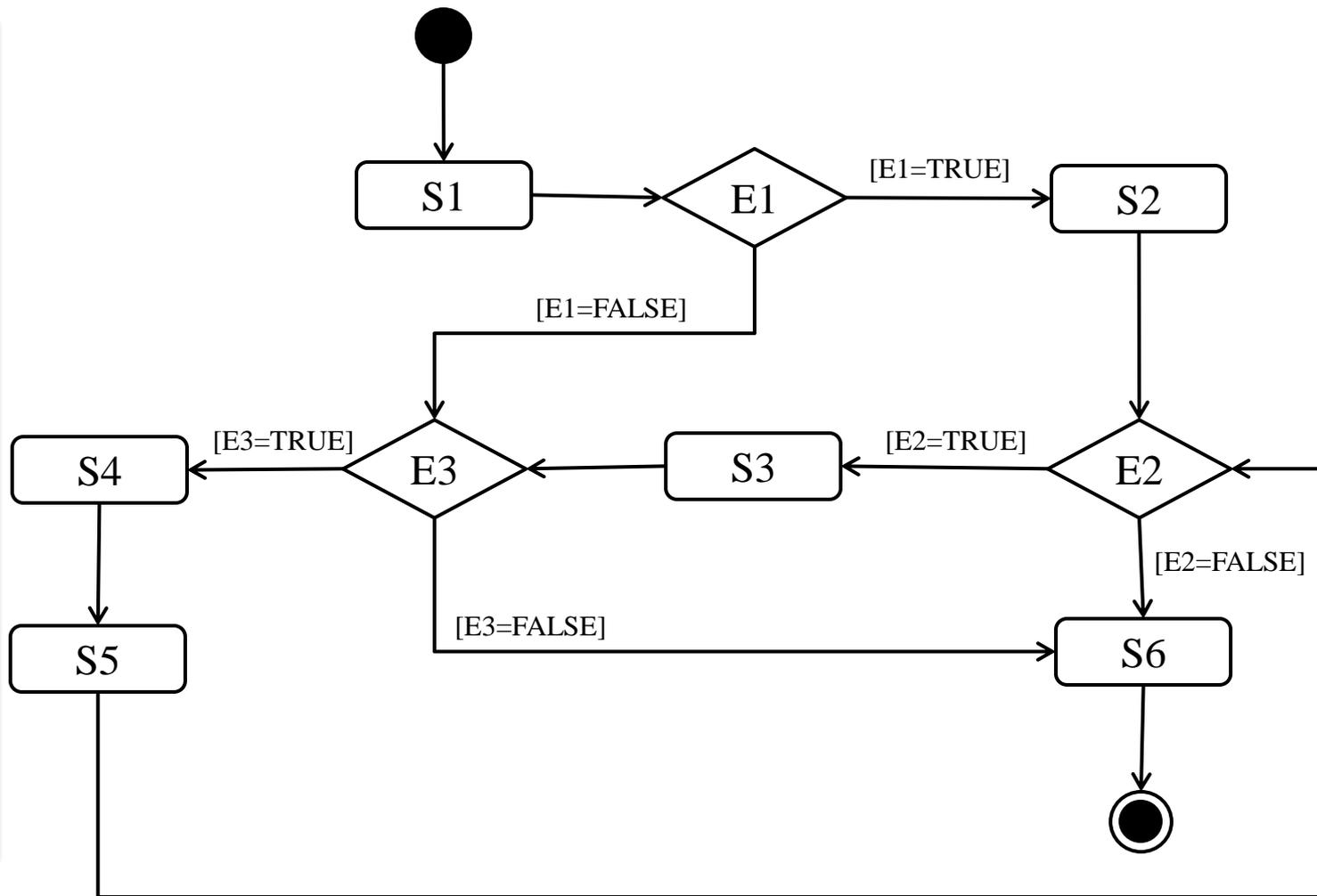
32

- Для любой неструктурной программы существует эквивалентная ей структурная программа.
  - Две программы эквивалентны, если для любых входных данных результаты их работы совпадают (выдают одинаковые выходные данные, завершаются по одной и той же ошибке времени выполнения или зависают).
  - Структурная программа состоит только из следующих конструкций:
    - следование (операторы присваивания и вызова процедуры);
    - ветвление (условный оператор);
    - повторение (оператор цикла).
- Для любой (не обязательно осмысленной!) программы с операторами Goto существует эквивалентная ей программа без операторов Goto.

# Теорема о структурном программировании

33

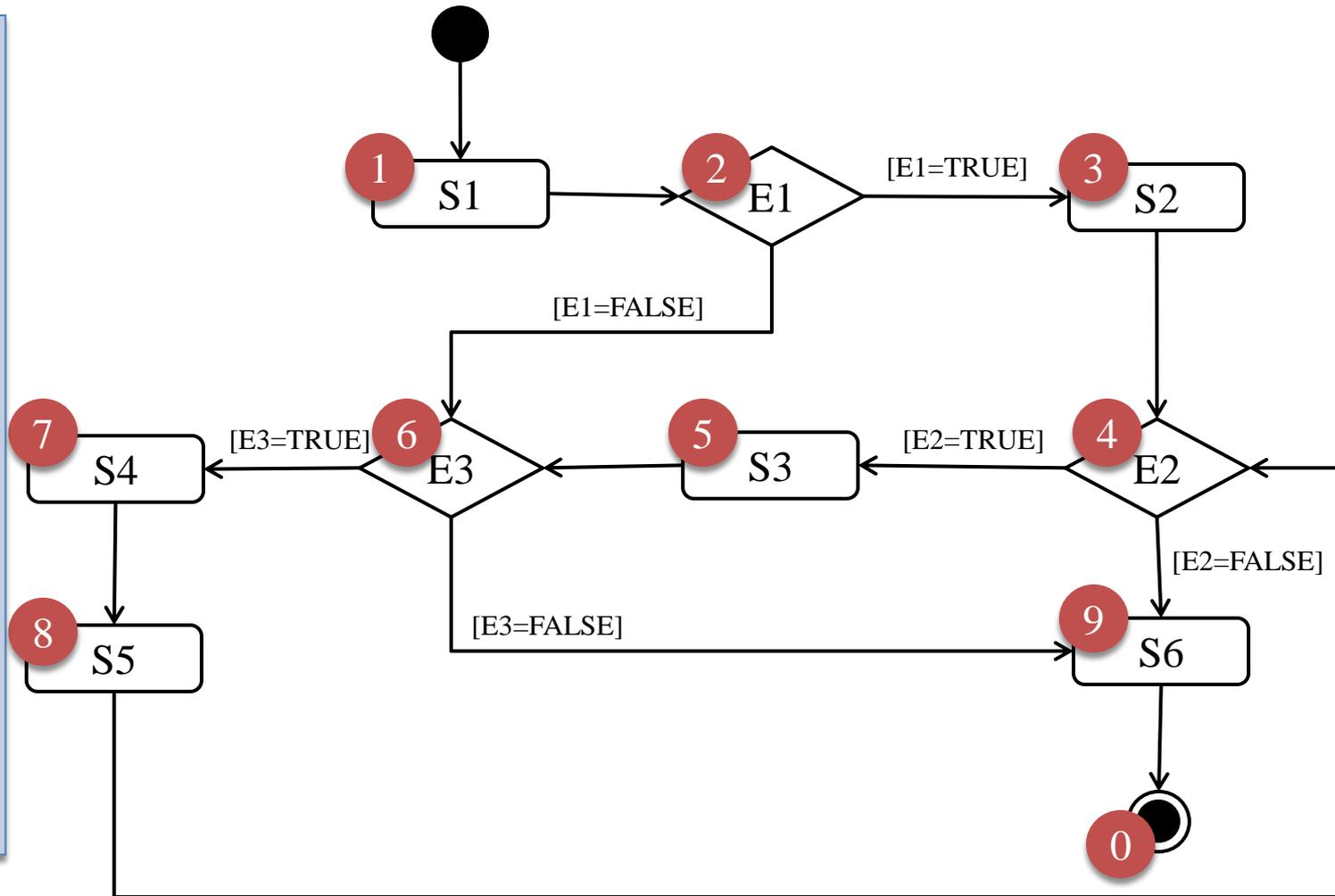
```
S1;  
if E1 then  
  S2  
else  
  goto L1;  
while E2 do  
begin  
  S3;  
L1: ;  
  if E3 then  
    S4  
  else  
    goto L2;  
  S5;  
end;  
L2: ;  
S6;
```



# Теорема о структурном программировании

34

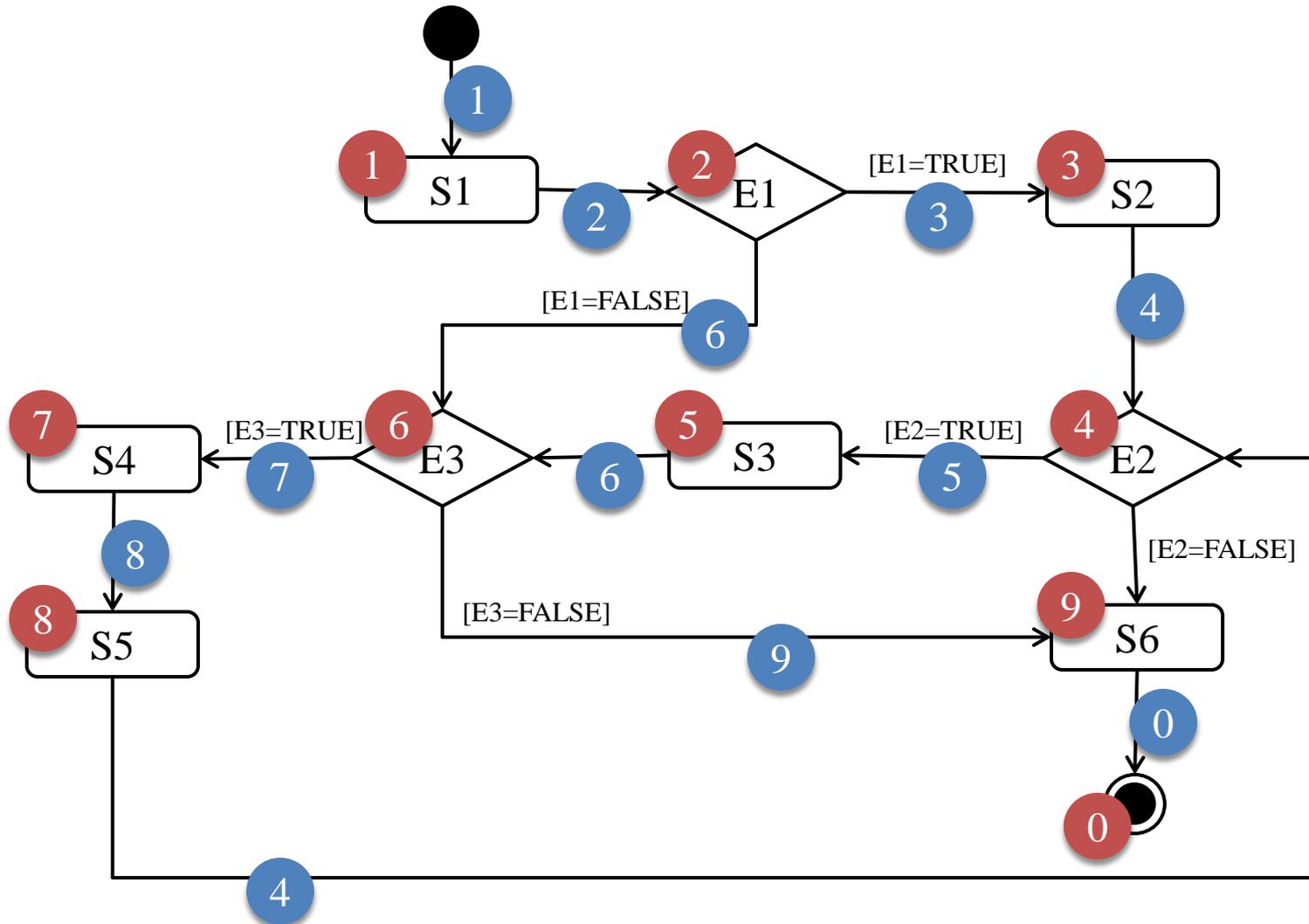
```
S1;  
if E1 then  
  S2  
else  
  goto L1;  
while E2 do  
begin  
  S3;  
L1: ;  
  if E3 then  
    S4  
  else  
    goto L2;  
  S5;  
end;  
L2: ;  
S6;
```



# Теорема о структурном программировании

35

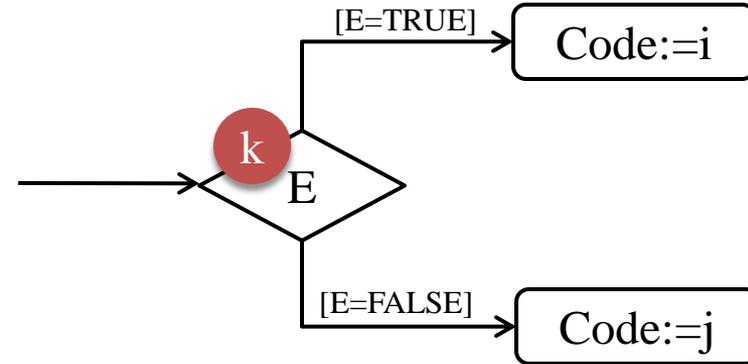
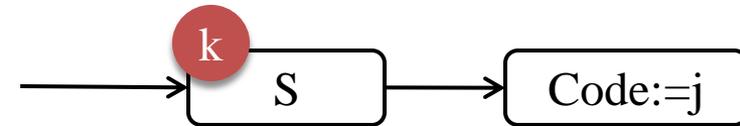
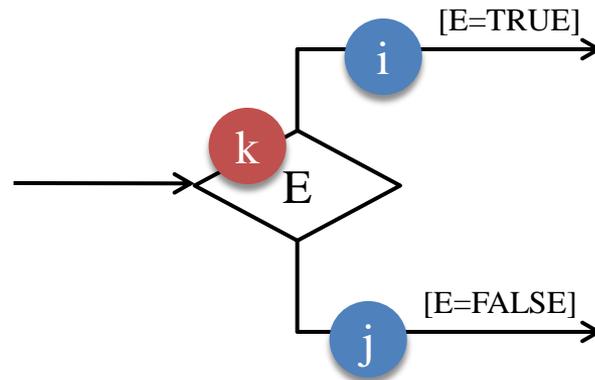
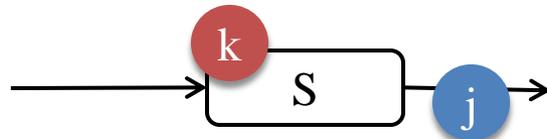
```
S1;  
if (E1)  
  S2;  
else  
  goto L1;  
while (E2) {  
  S3;  
L1: ;  
  if (E3)  
    S4;  
  else  
    goto L2;  
  S5;  
}  
L2: ;  
S6;
```



# Теорема о структурном программировании

36

```
S1;  
if E1 then  
  S2  
else  
  goto L1;  
while E2 do  
begin  
  S3;  
L1: ;  
  if E3 then  
    S4  
  else  
    goto L2;  
  S5;  
end;  
L2: ;  
S6;
```

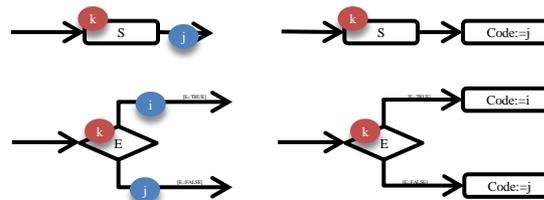
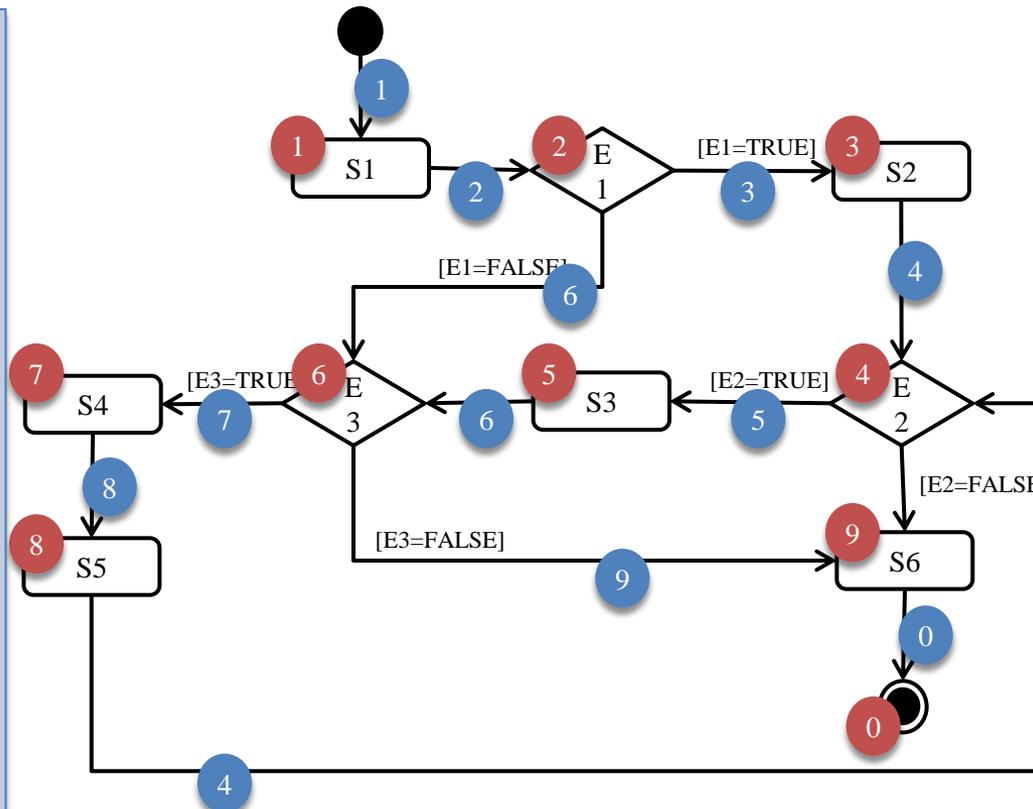


# Теорема о структурном программировании

37

```

S1;
if E1 then
  S2
else
  goto L1;
while E2 do
begin
  S3;
L1: ;
  if E3 then
    S4
  else
    goto L2;
  S5;
end;
L2: ;
S6;
    
```



```

Code:=1;
while Code<>0 do begin
  if Code=1 then begin
    S1
  end
else begin
  if Code=2 then
  begin
    if E1 then
      Code:=3;
    else
      Code:=6;
  end
else begin
  if Code=3 then begin
    S2;
    Code:=4;
  end
else begin
  if Code=4 then begin
    ...
  end
else begin
  if Code=9 begin
    S6;
    Code:=0;
  end
end
end;
end;
end;
end;
    
```