СТРУКТУРЫ УПРАВЛЕНИЯ НА УРОВНЕ ОПЕРАТОРОВ

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

П. Буаст

Языки программирования

Содержание

Операторы

- Операторы – конструкции языка программирования для управления вычислениями.
- Операторы должны обеспечивать:
  - присваивание
  - ветвление
  - повторение
  - безусловный переход
- Какое количество операторов должно быть в языке программирования?
  - Не много – иначе от программиста требуется более глубокое знание языка.
  - Не мало – иначе требуется использование goto, что снижает читабельность программ.

Составные операторы

- *Составной оператор (compound statement)* позволяет создавать набор операторов, рассматриваемый как отдельный оператор.
- Отсутствовал в ранних версиях FORTRAN.
- Впервые введен в ALGOL 60.
- В некоторых языках в начало составного оператора можно добавлять объявления данных (*блок*).
- Составной оператор может использовать "операторные скобки" (begin и end, { и } и т.п.) или интегрирует управляющие структуры без них.

Операторы ветвления

- Двухвариантное ветвление
  - Какой вид и тип имеет выражение ветвления?
  - Можно ли выбрать отдельный оператор, последовательность операторов или составной оператор?
  - Как определить смысл вложенных операторов ветвления?

Двухвариантное ветвление

- FORTRAN
  - ```
          IF (FLAG .NE. 1) GO TO 20
          I=1
          J=2
    20    CONTINUE
    ```
- ALGOL 60
  - ```
    if  (Flag<>1) then
    begin
       i:=1;
       j:=2;
    end;
    ```

Вложенные операторы ветвления

- Язык может не включать синтаксических индикаторов, указывающих на соответствие фразы else фразе then.
  - Pascal

```
if Sum=0 then { 1 }
   if  Count=0 then { 2 }
                Result:=0
else { 2 }
   Result:=1;
```

  - ALGOL

```
if Sum=0 then begin
   if  Count=0 then
                Result:=0
end
else
   Result:=1;
```

Вложенные операторы ветвления

- Язык может включать специальные слова для замыкания оператора ветвления.
  - Ada

```
if A>B then
    Sum:=Sum+A;
    Acct:=Acct+1;
else
    Sum:=Sum+B;
    Acct:=Acct-1;
end if;
```

  - Ada

```
if Sum=0 then
    if  Count=0 then
       Result:=0
    else
        Result:=1;
    end if;
end if;
```

Многовариантное ветвление

- *Оператор многовариантного ветвления (multiple selection)* обобщает оператор ветвления позволяет выбрать для выполнения один оператор или группу операторов из произвольного количества операторов или групп операторов.
  - Какова формат и тип выражения, управляющего ветвлением?
  - Можно ли выбрать отдельный оператор, последовательность операторов или составной оператор?
  - Представляет ли собой конструкция отдельную синтаксическую структуру?
  - Как обрабатывать непредставленные значения условных выражений?

Примеры

- FORTRAN
  - ```
          IF (A*B-C) 10, 20, 30
    10    …                          CC <0
          GO TO 40
    20    …                          CC =0
          GO TO 40
    30    …                          CC >0
    40
    ```
  - ```
          GO TO (10, 20, 30) A*B-C
    10    …                          CC =1
          GO TO 40
    20    …                          CC =2
          GO TO 40
    30    …                          CC =3
    40
    ```

Примеры

- ALGOL-W
  - case A*B-C of { целое }
    begin
      Z :=24; { =1 }
      Z:=A+B; { =2 }
      Z:=0; { =3 }
    end;
- Pascal
  - case I of
    1..3: begin … end;
    6, 8: begin … end;
    else
      WriteLn('Непредвиденное значение: ', I);
    end;

Примеры

- C
  - switch (I){
    case 1:
    case 2:
    case 3: …; break;
    case 6:
    case 8: …; break;
    default: printf("Непредвиденное значение: %d\n", I);
    }
- Ada
  - if Count<10 then B1:=TRUE;
    elsif Count<100 then B2:=TRUE;
    elsif Count<1000 then B3:=TRUE;
    end if;
  - if Count<10 then B1:=TRUE;
      else if Count<100 then B2:=TRUE; end if;
          else if Count<1000 then B3:=TRUE; end if;
    end if;

Операторы цикла

- Операторы цикла (iterative statements) вынуждают оператор или набор операторов выполняться один или сколько угодно раз, или ни разу.
  - Как осуществляется управление циклом?
  - В каком месте цикла находится механизм управления?

Примеры

- FORTRAN
  - DO счетчик=нач_знач, кон_знач, шаг
    END DO
- ALGOL
  - for cnt:=1,2,3,4,5,6,7,8,9,10 do list[cnt]:=0;
  - for cnt:=1, step 1 until 10 do list[cnt]:=0;
  - for cnt:=1, cnt+1 while (cnt<=10) do list[cnt]:=0;
  - for i:=1,4,13,41
    step 2 until 47,
    3*i while i<1000,
    34, 2, -24 do
    sum:=sum+i; { 1,4,13,41,43,45,47,147,441,34,2,-24 }

Примеры

- Ada
  - for счетчик in [reverse] нач..кон loop
    …
    end loop;
  - Счетчик цикла объявляется неявно как целочисленная переменная и не может быть изменен в теле цикла.
- C
  - for (i=0; i<N; i++) sum+=list[i];
  - for (cnt1=0, cnt2=1.0;
    cnt1<10 && cnt2<=100.0;
    sum=++cnt1+cnt2, cnt2*=2.5) {};
- C++
  - for (int i=0, j=0; …)

Логически управляемые циклы

- Pascal
  - while условие do begin
    оператор;
    end;
  - repeat оператор until условие;
- C
  - while (условие) { /* выполняется, пока !=0 */
    оператор;
    }
  - do {
    оператор;
    } while (условие); /* завершается, когда ==0 */
  - while (*dest++=*src++);

Управление внутри цикла

- Ada
  - loop
    …
    if условие then exit end if;
    end loop;
  - loop
    …
    exit when условие;
    end loop;
- C
  - for (i=0; i<N; i++)
    if (a[i]==X) break;
  - for (i=0; i<N; i++) {
    if (a[i]==X) continue;
    process(a[i]);
    }

Циклы на базе структур данных

- Perl
  - @names=("Ada", "BASIC", "C", "FORTRAN");
    foreach $name (@names) {
    print $name;
    }
- C
  - for (ptr=root; ptr!=NULL; process(ptr=ptr->next));

Защищенные операторы

- GSP (Xoap)
  - if i=0 -> sum:=sum+i
    [] i>j -> sum:=sum=j
    [] i<j -> sum:=sum+i
    fi
  - Все булевские выражения вычисляются каждый раз при достижении конструкции. Выполняется оператор(ы) истинного выражения. Если истинны несколько выражений, оператор выбирается случайным образом. Если все выражения ложны, возникает ошибка времени выполнения.

Безусловный переход

- *Оператор безусловного перехода (Goto statement)* передает управление в указанное место программы.
- Оператор Goto – самый мощный оператор управления, все другие управляющие структуры можно построить на основе goto и оператора ветвления.
- Бесконтрольное использование goto может существенно снизить читаемость программы (программа-"спагетти").
- Дискуссия о Goto (1968-74)
  - Э. Дийкстра и др.: "Goto примитивен и запутывает программу", нужно запретить или ограничить использование оператора Goto.
  - Д. Кнут: "Иногда эффективность оператора Goto может перевесить его вред для читабельности программы", оператор Goto нельзя исключать из языка.
- Оператор Goto в различных языках:
  - Modula-2, Java, Oberon, Component Pascal – нет.
  - C, C++, Pascal, BASIC, FORTRAN – есть.

Д. Э. Кнут
(р. 1938)

Э.В. Дийкстра
(1930-2002)

Виды меток

- ALGOL 60, C
  - Метки – идентификаторы
- FORTRAN, Pascal
  - Метки – целые без знака
- Ada
  - Метки – <<идентификаторы>>
- PL/1
  - Метки – переменные (можно присваивать значения и передавать как параметры подпрограмм)

Ограничения переходов


```
Program Sample;
Label 10;

procedure Outer;
Label 20;

procedure Inner;
Label 30;
begin { Inner }
        …
        30: …;
        goto 10;
        goto 20;
        goto 30;
end; { Inner }
…
begin { Outer }
        …
        20: …;
        goto 10;
        goto 20;
        goto 30;

end; { Outer }

begin { Sample }
        …
        10: …;
        goto 10;
        goto 20;
        goto 30;
end. { Sample }
```

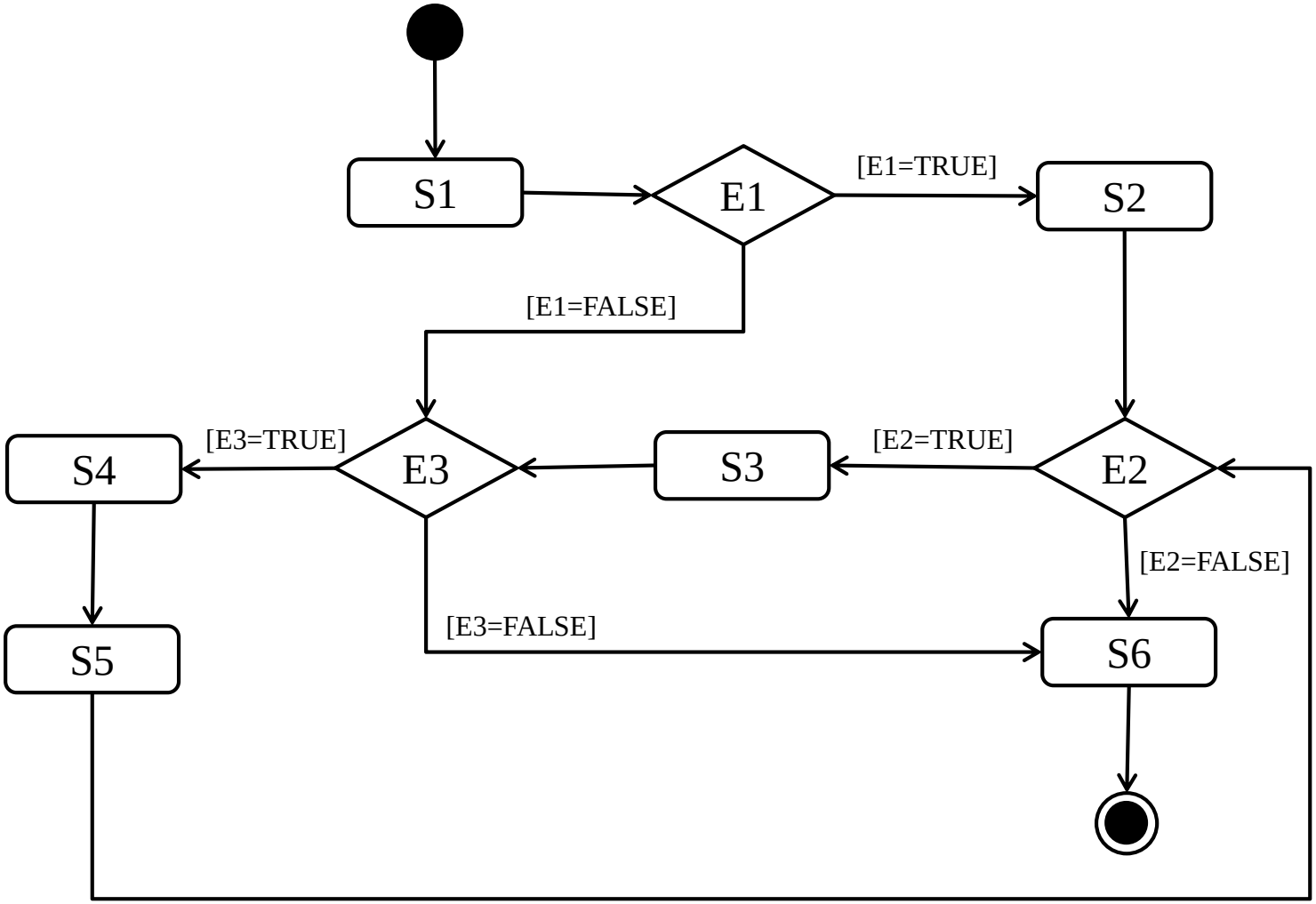
- *Активная группа операторов* – начавшая, но не закончившая свое выполнение.
- Типичное ограничение: целью оператора goto не может быть оператор из неактивной группы операторов.

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).
  - 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.

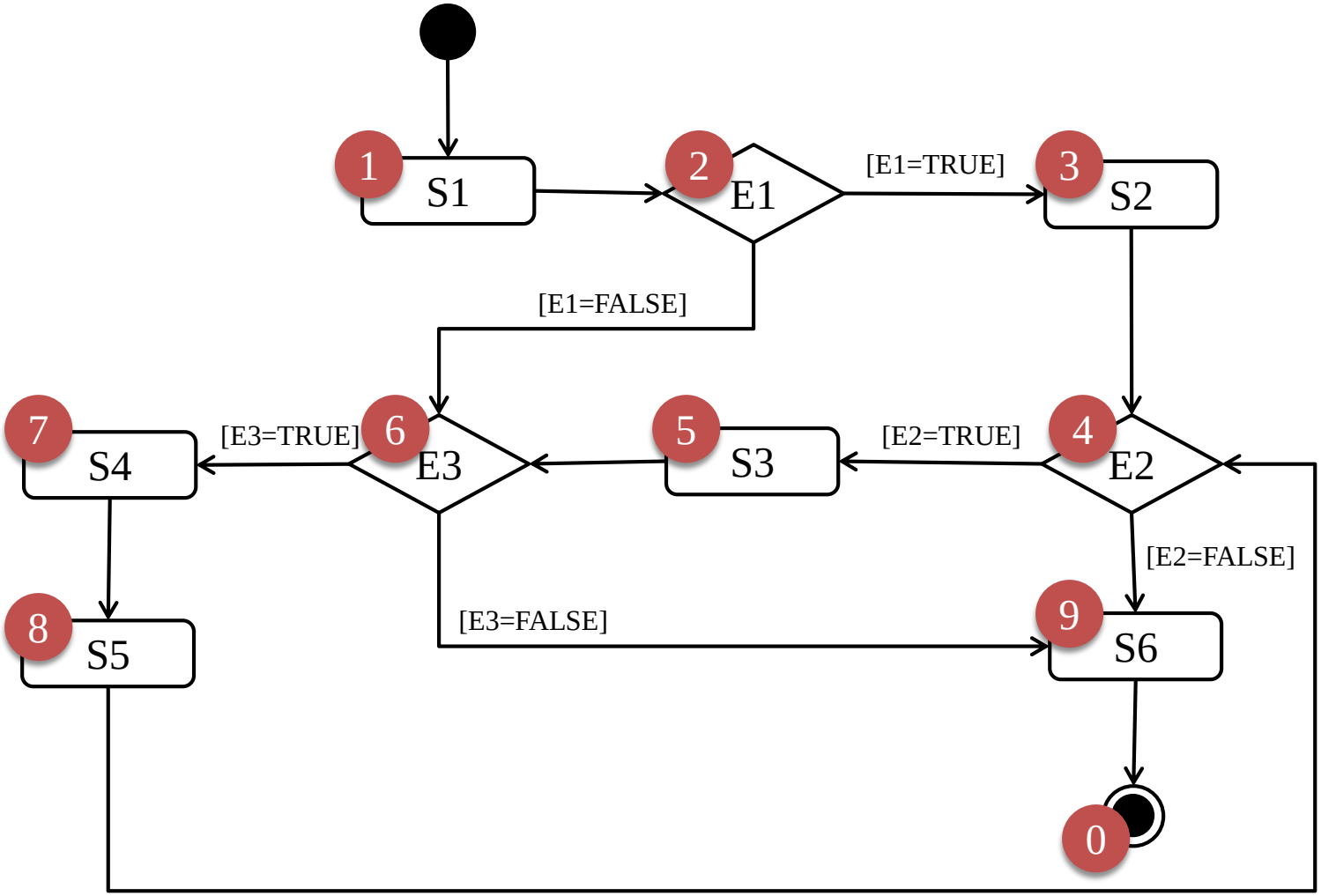
Теорема о структурном программировании


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



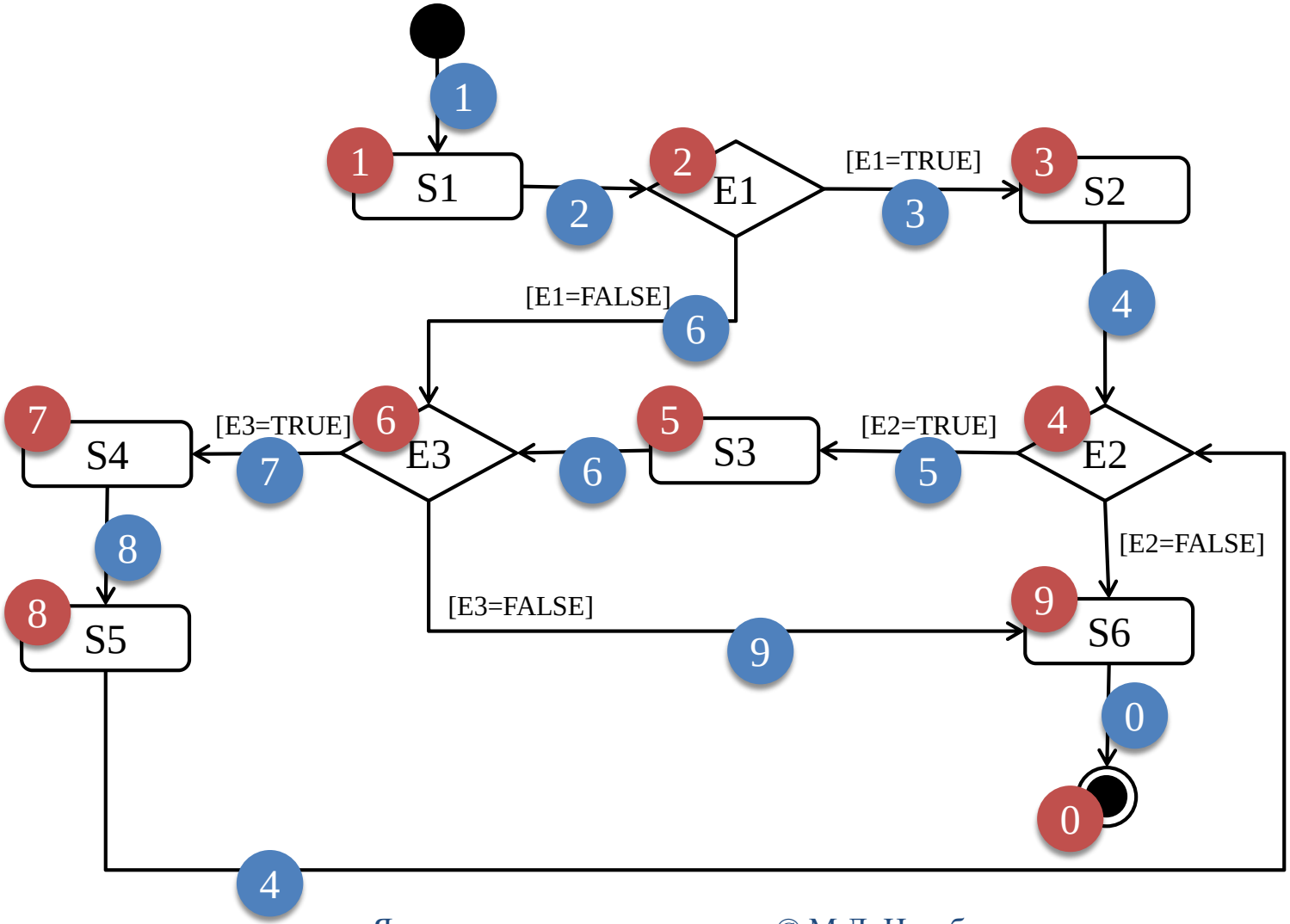
Теорема о структурном программировании


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



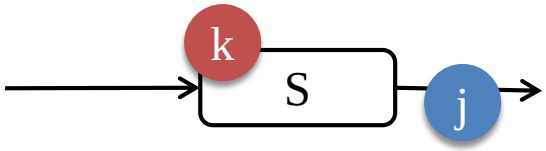
Теорема о структурном программировании

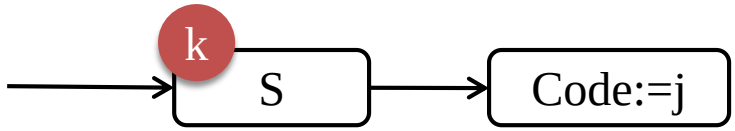

```
S1;
if (E1)
  S2;
else
  goto L1;
while (E2) {
  S3;
L1: ;
  if (E3)
    S4;
  else
    goto L2;
  S5;
}
L2: ;
S6;
```

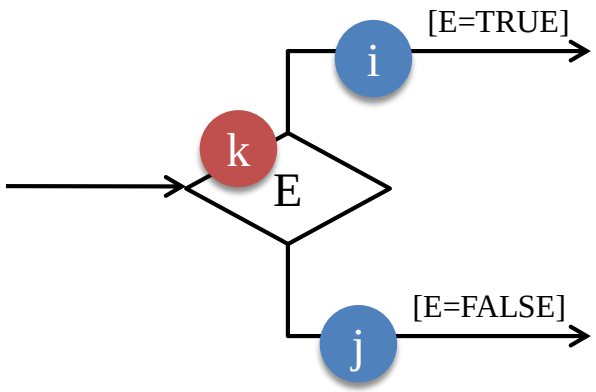


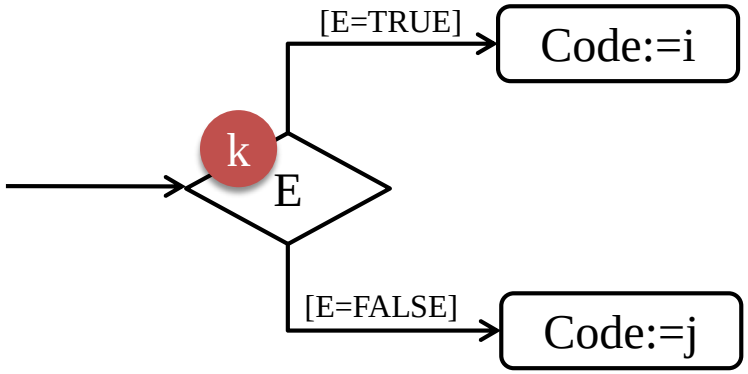
Теорема о структурном программировании


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



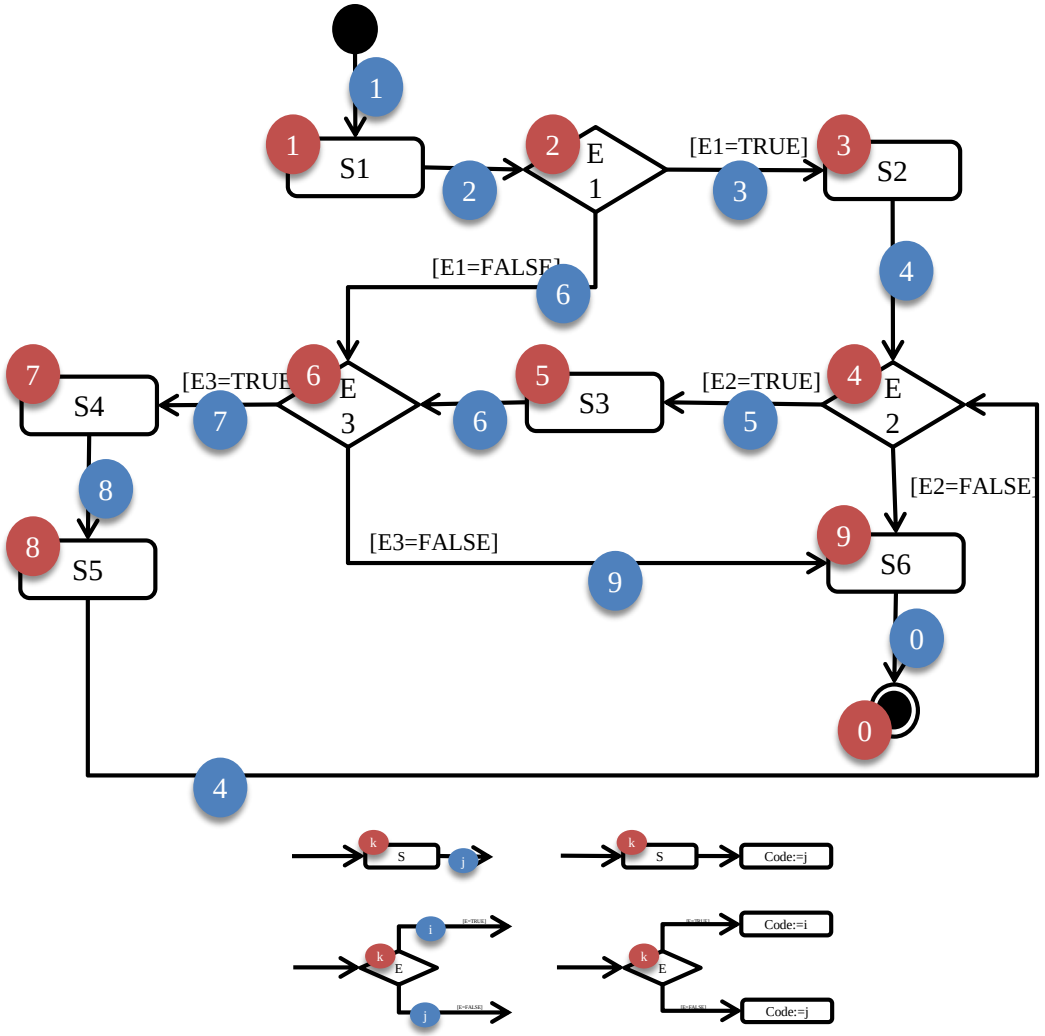






Теорема о структурном программировании


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




```
Code=1;
while (Code!=0) {
   if (Code==1) {
      S1;
   } else {
      if (Code==2) {
         if (E1)
            Code=3;
         else
            Code=6;
      } else {
         if (Code==3) {
            S2;
            Code=4;
         } else {
            if (Code==4) {
               …
            } else {

               if  (Code==9) {
                  S6;
                  Code=0;
               }
            }
         }
      }
   }
}
```

Структурное программирование

- 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 ясности и читаемости программы.
  - Полный отказ, равно как и безусловное использование операторов Goto может понизить ясность, читаемость и ухудшить логику программы

Польза оператора GoTo


```
int matrix[n][m];
int value;
...
for (int i=0; i<n; i++)
  for (int j=0; j<m; j++)
     if (matrix[i][j]==value) {
       printf("Найдено %d==matrix[%d][%d])\n", value, i, j);
     goto end_loop;
}
printf("Не найдено %d!\n",value);
end_loop: ;
```