



## СТРУКТУРЫ УПРАВЛЕНИЯ НА УРОВНЕ ОПЕРАТОРОВ

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

*П. Буаст*

Языки программирования

---

---

---

---

---

---

---

---

---

---

## Содержание

2

- Понятие оператора
- Составные операторы
- Операторы ветвления
- Операторы цикла
- Оператор безусловного перехода

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Операторы

3

- 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.

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Составные операторы

4

- *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.

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Операторы ветвления

5

- 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?

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Двухвариантное ветвление

6

- 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;

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Вложенные операторы ветвления

7

- 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;

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Вложенные операторы ветвления

8

- 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;

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Многовариантное ветвление

9

- *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?

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Примеры

10

### □ 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

```

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Примеры

11

### □ 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;

```

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Примеры

12

### □ 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;

```

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Операторы цикла

13

- 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?

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Примеры

14

- 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 }

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Примеры

15

- 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; ...)

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Логически управляемые циклы

16

- 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++);

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Управление внутри цикла

17

- 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]); }

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Циклы на базе структур данных

18

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

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Защищенные операторы

19

### □ GSP (Хоар)

- 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.

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Безусловный переход

20

- 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. Дijkstra 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 читаемости 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. Дijkstra (1930-2002)

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Виды меток

21

- 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)

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

## Ограничения переходов

22

```

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.

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

## Теорема о структурном программировании

23

- 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.

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

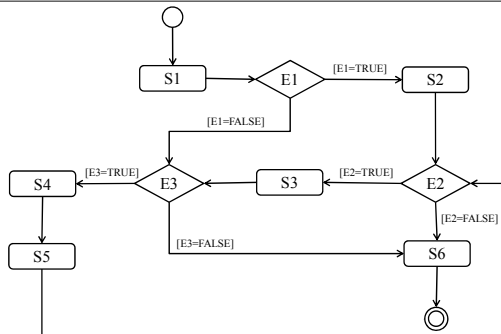
---

## Теорема о структурном программировании

24

```

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



Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---



### Теорема о структурном программировании

25

```

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

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

### Теорема о структурном программировании

26

```

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

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

### Теорема о структурном программировании

27

```

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

Языки программирования © М.Л. Цымблер

---

---

---

---

---

---

---

---

---

---

