МИНИСТЕРСТВО ОБОРОНЫ СОЮЗА ССР

СИСТЕМА „БЭСМ-АЛГОЛ“

Методическое руководство по программированию

1970

I

МИНИСТЕРСТВО ОБОРОНЫ СОЮЗА ССР

СИСТЕМА „БЭСМ-АЛГОЛ“

Методическое руководство по программированию

1970

Данная работа является методическим руководством по использованию системы "БЭСМ-АЛГОЛ" при разработке математических заданий на языке Алгол-60 и состоит из пяти глав.

В первой главе дается описание входного языка. Она разработана на основе опыта эксплуатации системы и трудов по алгоритмическому языку Алгол-60, указанных в разделе "Литература".

Вторая глава посвящена стандартным процедурам. В ней излагаются правила использования, возможности процедур и указываются характерные ошибки, возникающие при неверном их применении.

В третьей главе дана организация архива индивидуальных библиотек процедур.

В четвертой главе приводится инструкция по набивке программ на устройствах подготовки и составлению комплекта перфокарт для их решения на машине БЭСМ-6.

В пятой главе рассматривается вопрос об отладке программ. Здесь указываются средства отладки, причины возникновения некоторых ошибок и способы их устранения.

Всего пронумеровано 100 стр.

I. ВХОДНОЙ ЯЗЫК

§ I. Ограничения

Входным языком системы является алгоритмический язык Алгол-60 с некоторыми уточнениями и несущественными ограничениями; последние носят в основном чисто количественный характер и обусловлены параметрами и особенностями машины БЭСМ-6 и выбранным методом транслирования.

Ниже перечислены наиболее важные уточнения и отличия входного языка от языка Алгол-60.

I. Используется только один регистр русских и латинских букв.

2. В строках можно использовать любые символы УПП за исключением символа ◊ во всех программах и % - в процедурах.

3. Диапазон и точность представления вещественных и целых величин одни и те же , они определяются разрядностью машины БЭСМ-6 (диапазон-$10^{-19} \div 10^{19}$, точность - 10_{10} разрядов).

4. Длина идентификатора не должна превосходить 90 символов.

5. Рабочая программа, административная система (служебные программы, обеспечивающие нормальное функционирование рабочей программы, стандартных процедур и функций; административная система занимает 2 листа МОЗУ) и динамически активные массивы должны умещаться в оперативной памяти машины, т.е. в 24 листа.

6. Статическая глубина рекурсий, т.е. число последовательно вложенных друг в друга описаний процедур, не должна превышать IO.

7. Оператор перехода при неопределенном указателе переключателя не эквивалентен пустому оператору, если при вычислении именующего выражения в результате побочного эффекта производится присваивание значений каким-либо переменным.

8. Все собственные величины (*own*) трактуются статически: величины *own* , описанные в теле какой-либо процедуры, считаются одними и теми же при всех обращениях к этой процедуре.

9. В процессе выполнения некоторых стандартных процедур нельзя обращаться к другим стандартным же процедурам или же к процедурам, содержащим описание собственных массивов (*own*).

При работе процедуры	Нельзя обращаться к процедурам	Обращение к процедурам с массивами *own*
input	READ, WRITE	недопустимо
READ	INPUT	недопустимо
WRITE	INPUT	недопустимо
OUTPUT		недопустимо
MARG		недопустимо

Так,например, нельзя писать программы вида

```
begin integer procedure F;
    begin F:=1; input (A) end;
    array A[1:100];
    WRITE (A, 1,1,1, F);
end
```

или еще

```
begin real procedure F;
begin own real array B[1:20]; F:=3.14 end;
    output ('E', F);
end
```

IO. Идентификаторы стандартных функций и процедур нельзя использовать в качестве фактических параметров.

§ 2. Арифметические выражения

Для обозначения операций употребляются обычные арифметические символы:

\+ сложение;

\- вычитание;

× умножение;

/ деление;

÷ деление нацело;

↑ возведение в степень.

При отсутствии скобок установлен следующий порядок: первыми выполняются все возведения в степень, затем все умножения и деления и, наконец, все сложения и вычитания. Так, следующие два выражения эквивалентны:

Зак.Б-747/742ф.

$A \times B + C/D - E + F \uparrow R$

$(A \times B) + (C/D) - (E \times (F \uparrow R))$

Два знака арифметических операций никогда не пишутся рядом. Так, выражение A×-B является неверным, а выражения A×(-B) и -A×B - верными.

В тех случаях, когда естественная последовательность операций должна быть изменена, применяются круглые скобки. Так, выражение $(X+Y)^3$ должно быть записано в виде (X+Y)↑3, чтобы сохранить смысл выражения;
X+Y↑3 означает $X+Y^3$. Так как возведение в степень является операцией наивысшего ранга, выражение для показателя должно заключаться в скобки, за исключением того случая, когда оно является просто числом без знака или переменной.
Например, математическое выражение $\left(\frac{x}{10}\right)^{2i-1}$ должно быть написано в виде $(x/10)\uparrow(2\times i-1)$. Если оно записано в виде $(x/10)\uparrow 2\times i-1$, то согласно правилу очередности выполнения операций, оно будет интерпретироваться как $\left(\frac{x}{10}\right)^2 \cdot i - 1$, т.е. совсем по-другому.

Если последовательность операций не определяется ни правилами очередности выполнения операций, ни скобками, действия выполняются слева направо. Так A/B×C означает (A:B)C, а не A:(BC) и A↑B↑C означает $(A^B)^C$, а не $A^{(BC)}$.

Операция ÷ определена только для случая, когда оба операнда целого типа. Результат всегда имеет целый тип и получается отбрасыванием дробной части частного (такое отбрасывание не является округлением). Так, 7 ÷ 4 есть I, а не 2; 99÷ I00 равно нулю. При применении этой операции надо внимательно относиться к последовательности действий. Например, 6×4 ÷ 3 = 8; а выражение 6×(4 ÷ 3) дает 6.

Операция ↑ приводит к результатам различного рода в зависимости от величины и типа основания и показателя. Обозначив число целого типа через i, число действительного типа - через z, число произвольного типа - через a, получим:

$a \uparrow i$:
- $a \times a \times \ldots \times a$ (i раз), если $i > 0$; тип тот же, что и у a;
- I, если $i = 0$ и $a \neq 0$; тип тот же, что и у a;
  не определено, если $i = 0$ и $a = 0$;
- $1/(a \times a \times \ldots \times a)$ (в знаменателе i сомножителей), если $i < 0$ и $a \neq 0$ - действительный тип;
  не определено, если $i < 0$ и $a = 0$;

$a \uparrow z$:
- $\exp(z \times \ln(a))$, если $a > 0$; действительный тип, т.е. $e^{z \ln a}$;
- 0 если $a = 0$ и $z > 0$; действительный тип;
  не определено, если $a = 0$ и $z \leq 0$;
  не определено, если $a < 0$.

§3. Стандартные функции

За стандартными функциями закрепляются некоторые определенные идентификаторы, которые используются без предварительного описания. Ниже приводится список стандартных функций входного языка.

- ABS(E) - для модуля значения выражения E;
- SIGN(E) - для знака значения E (+I для $E > 0$, 0 для $E = 0$, -I для $E < 0$);
- SQRT(E) - для квадратного корня из значения E;
- SIN(E) - для синуса значения E;
- COS(E) - для косинуса значения E;
- ARCTAN(E) / ARCTG(E) - для главного значения $[+\pi/2; -\pi/2]$

арктангенса E ;

LN - для натурального логарифма значения E ($\ln E$);

$EXP(E)$ - для показательной функции значения E (e^E);

$ENTIER(E)$ - выделение целой части в виде нормализованного числа (наибольшее число, не превышающее значения E); например, $ENTIER(2,6)=2$, $ENTIER(-2,6)=-3$;

$TAN(E)$, $TG(E)$ - для тангенса значения E ;

$ARCSIN(E)$ - для главного значения арксинуса E $[+\pi/2 , -\pi/2]$;

$MAX(E_1, E_2, \dots E_n)$ - максимальное из значений E_i $(i>1)$; n - любое;

$MIN(E_1, E_2, \dots E_n)$ - минимальное из значений E_i $(i>1)$, n - любое; E - любое арифметическое выражение.

Например:

$SIN(a \times b)$, $\cos(0,5)$, $\tan(a\uparrow 3/(c+d))$.

Использование идентификаторов стандартных функций для других целей не рекомендуется.

§4. Булевские (логические) выражения

Булевским выражением называется выражение, которое в зависимости от значений входящих в него переменных может принимать одно из двух возможных логических значений:

или true , или false.

К булевским выражениям относятся:

I. Операции отношения:

$<$ - меньше;

$\leq$ - меньше или равно;

$\geqslant$ - больше или равно;

$=$ - равно;

$>$ - больше;

$\neq$ - не равно.

2. Логические операции:

$\neg$ - операция отрицания ("НЕ");

$\wedge$ - операция логического умножения ("И");

$\vee$ - операция логического сложения ("ИЛИ");

$\supset$ - импликация;

$\equiv$ - тождество.

Все логические операции приведены в порядке выполнения.

Ниже приводится таблица значений результатов выполнения логических операций:

$b1$ $b2$	true false	true true	false true	false false
$\neg b1$	false	false	true	true
$b1 \wedge b2$	false	true	false	false
$b1 \vee b2$	true	true	true	false
$b1 \supset b2$	false	true	true	true
$b1 \equiv b2$	false	true	false	true

Булевской переменной можно присваивать значение при помощи оператора присваивания; при этом требуется только, чтобы выражение в правой части было булевским выражением. Например, можно писать следующие операторы:

```
A := true ;
B := false ;
C := x<10 ;
```

В результате выполнения последнего оператора величина C станет равной true , если текущее значение X меньше IO и false - в противном случае;

```
D := y=1 ∧ q>8
L := ⌝F ∨ R+S=T
```

В этих примерах величины D и L будут равны true, если в первом примере y=I и q>8, во втором - или F=false, или R+S=T , в противном случае D и L примут значения false

Нельзя использовать в качестве булевских выражений:

```
A := B×C ∧D+E ;
A := B/C ∨D↑2 ;
```

т.е. нельзя использовать в одном выражении арифметические операции и логические операции.

§5. Оператор присваивания

Оператор этот очень простой, но имеет ряд особенностей.

I. Если переменная левой части имеет тип integer,

то, независимо от типа результата правой части, последний будет округлен до ближайшего целого.

Например:

```
A := 2×B+C ;
```

где A типа integer ;

B и C типа real ;

и B=2.3 , а C=3.5 , то A=8 .

2. Все переменные "левой части" должны быть одного и того же типа. Например:

```
x := y := z := 0 ;      x := y := z[i] := 20 ;
x := y := z := z+1 ;
```

где величины x, y, z описаны либо как integer , либо как real .

3. Слева должна стоять переменная, а не число. Поэтому следующая запись во входном языке недопустима:

```
5.25 := a+b↑2 ;
```

4. Переменная слева должна быть записана без знака, поэтому нельзя писать следующий оператор:

```
-a := b↑4
```

5. Переменная слева **не должна** быть идентификатором стандартной функции или процедуры. Так, нельзя писать следующие операторы:

```
sin(x) := y + cos(x) ;
tan(y) := x↑2 ;
слг(a) := 0,357 ;
```

в последнем примере сл2(a) является процедурой.

6. Допустимы следующие операторы:

```
x := -a×y ;
x := -(a+B)↑2+C ;
```

§6. Метки и операторы перехода

Метками во входном языке могут быть любое целое без знака или любой идентификатор. Например:

```
alfa : M20:20:400 :   X1 M2 :
```

Оператор перехода всегда имеет вид:

```
go to M20;   go to X1 M2;   go to alfa;
```

§7. Условные операторы

I. Оператор „if".

Этот оператор имеет следующий вид:

```
if a>b then c:=B+c;
if a=0 ∧ c=1 then x:=y↑2;
if c>a ∨ b=d ∧ e=1 then go to L;
```

Если во всех трех условных операторах выполняются условия, то всегда выполняются операторы, стоящие после символа then, и ничего не выполняется, если не выполняются условия.

Оператор, следующий за then, может быть любого типа, но не может быть снова оператором „if".

Следовательно, следующая конструкция недопустима:

```
if a>b then if c=0 then a:=0;
```

Но можно писать следующий оператор:

```
if a>b then begin if c=0 then a:=0; end;
if a>b then begin if c=0 then a:=0 else a:=1;
                  end;
```

2. Полный условный оператор

Входной язык допускает следующие конструкции данного оператора:

```
if a=b then c:=a+b else c:=d;
if a>b then go to L else if a=0 then a:=20
else if a<b then c:=0 else c:=1;
if c=d then begin if c=0 then a:=c+d end
 else b:=c↑2;
if a=0 then for i:=1 step 1 until 20 do b[i]:=0;
```

Недопустимы конструкции:

а) `if a=b then if a=0 then a:=10 else c:=0;`

В этом случае не известно, к какому символу if относится символ else ;

б)
```
if a>b then for i:=1 step 2 until 23 do
    d[i]:=a×i  else  b:=c;
```

Но можно писать:

```
if a>b then begin for i:=1 step 2 until 23
do d[i]:=a×i end else b:=c;
```

Условные операторы могут применяться в операторах присваивания и в операторах перехода. Например, можно использовать конструкции вида

```
a := if b=0 then 2 else b×c;
a := if b=0 then 3×c+d↑2 else if b=1 then -32
     else 0;
go to if a=0 then M else M1;
```

В конструкциях подобного рода символ **else** обязателен, поэтому не допустимы конструкции вида:

```
a := if b=0 then 2;
go to if b=0 then M;
```

ибо неизвестно, что нужно присвоить a или куда передать управление, если $b \neq 0$

Можно использовать оператор более сложного вида:

```
z := if x<1 then (if x≥1 then 1 else 0) else 2;
```

Во всех условных операторах сложные условия пишутся в виде

```
if a≥0∧a≤1 then go to M;
```

а не так: `if 0≤a≤1 then go to M;`

В условных операторах можно ставить метки:

```
if a=b then M: b:=c+d else M1:a:=b↑2;
go to M;..... go to M1;
```


§ 8. Оператор цикла

Входной язык допускает следующие конструкции оператора цикла:

а) `for i:=1 step 1 until 10 do`

Здесь величина i принимает значения: 1, 2, 3,...,9, 10, а после выхода из цикла она равняется 11, т.е. на шаг больше, чем предел цикла.

б) `for i:=n step -1 until m+2 do`

В этом случае величина i уменьшается на каждом шаге цикла на 1 до $m+2$. По окончании цикла величина $i=m+1$

```
c) for i := if a>0 then m else p+2 step
if b=0 then 1 else 2 until if c=0 then a+b
     else a↑2 do
```

Здесь шаг цикла и предел цикла в зависимости от условий (если переменные b и c изменяются в теле цикла) принимают разные значения на каждом шаге цикла.

Если в конструкции вида: `for i:=n step 1 until m do` $n>m$, то оператор цикла никаких действий не производит, т.е. тело цикла не выполняется.

После выхода из оператора цикла $i=n$

```
d) for i := 1, 3, 4, 10 do
```

В этом случае i принимает значения: 1, 3, 4, 10, и после выхода из цикла $i=10$.

е) for i:=5, 10, 12 step 1 until 16, 18 step -2 until 8 do

Здесь i принимает значения: 5, 10, 12, 13, 14, 15, 16, 18, 16, 14, 12, 10, 8. После выхода из цикла $i=6$

ƒ) for i:=1,5,7 step 1 until 10, i+2 while i≤20 do

В этом операторе цикла i будет принимать значения: 1, 5, 7, 8, 9, 10, 13, 15, 17, 19. При выходе из цикла $i=21$.

Следует обратить внимание на то, что после числа 10 стоит число 13. Дело в том, что элемент типа арифметической прогрессии будет исчерпан, когда выполнится условие $i>10$. Это произойдет при $i=11$, затем i будет изменяться по зависимости $i:=i+2$, пока выполняется условие $i\leq 20$.

Из тела цикла можно выходить по оператору перехода с последующим возвращением в тело цикла. Здесь могут быть несколько случаев:

```
1)  3: for i:=1 step 1 until 10 do begin
       . . . . . .
       go to M;
    M1: . . . . . .
    M2: end;
       . . . . . .
    M: . . . . . go to M1;
```

В этом случае при возврате на метку M1 тело цикла продолжает выполняться с тем же значением параметра i, при котором было передано управление на метку M.

2) Если после метки M стоит оператор go to M2, то выполнение тела начнется с самого начала, но со значением параметра i на шаг цикла больше, чем при выходе из цикла по оператору go to M.

3) Если после метки M стоят операторы i:=6; go to M1; , то продолжение выполнения цикла начнется с $i=6$.

4) Если после метки M стоят операторы i:=8; go to 3; то выполнение тела цикла начнется с $i=1$.

5) Если после метки M стоят операторы i:=i+4; go to M2; то тело цикла начнет выполняться с $i=i+5$

д) for i:=1, i+4 while i<20 do . . .

В этом операторе i будет изменяться:

1, 5, 9, 13, 17.

При $i=21$ произойдет выход из оператора цикла.

§ 9. Переключатели

Входной язык допускает возможность использования переключателя без ограничений для описания алгоритма со сложной системой меток и операторов перехода.

Для использования переключателя в программе необходимо в заголовке блока, в котором данный переключатель будет использован, среди описаний поместить описание переключателя вида

SWITCH L := M1, S, 20, 40, M2;

а в теле блока написать оператор перехода вида

go to L[1];

или

go to L[i];

где i должно принимать значения от 1 до 5.

Метки в описании переключателя считаются пронумерованными от 1 в возрастающем порядке и должны находиться только в том блоке, в котором описан переключатель, даже если они в этом блоке не используются.

Зак. Б-747/742ф.

Например:

```
begin real a; integer i;
      switch L := M1, M2, 3;
      . . . . . .
      M1: . . . . . .
      . . . . . .
      M2: . . . . . .
      . . . . . .
      3: . . . . . .
      . . . . . .
      go to L[2]; . . . i := 2;
      . . . . . .
      go to L[i]; go to L[i-1];
      . . . . . end
```

Нельзя использовать переключатель в следующих программах:

```
begin real a; integer i;
        switch L := M1, M2,3;
        . . . . . .
        a := 1;    i := 0; . . . .
begin real b;  integer c;
      b := a+2;  c := b↑2 + i×2;
        . . . . . .
        M1:
        . . . . . .
        M2:
        . . . . . .
        3:
        . . . . . .
        i := 2; go to L[i]; . . . . go to L[3];
        end;
        . . . . . .
        end
```

В этой программе описание переключателя и его метки M1, M2, 3 находятся в блоках разного уровня.

В этой программе переключатель и его метки используются во внутреннем блоке, поэтому описание переключателя должно

находиться тоже во внутреннем блоке.

```
      begin real a; integer i;
            . . .
            a := 1 ;   i := 0;
      begin real b; integer c;
            switch L := M1, M2, 3;
            b := a+2;  c := b↑2 + i×2;
            . . .
            M1: . . .
            . . .
            3: . . .
i := 2; go to L[i]; . . . go to L[3]    end;   end
```

Если в программе переключатель и его метки используются во внешнем блоке, то описание переключателя и его метки должны находиться тоже во внешнем блоке.

```
begin real a; integer i;
        switch L := M1, M2, 3;
        . . .
        a := 1;  i := 0;  go to L[i+1];
        M1:
  begin real b;  integer c;
        b := a+2; c := b↑2 + i×2;
        . . .
  end;
        . . .
        M2: . . .
        3: . . .
        go to L[1];
  end
```

Если метки описаны через переключатель, то передача управления на них должна осуществляться только оператором перехода вида:

go to L[1]; go to L[2]; go to L[3];

а также оператором

go to M1; go to M2; go to 3;

§ IO. Блоки

Блоком могут быть один или несколько операторов, заключенных в операторные скобки *begin* и *end*, он содержит описания сразу после *begin*.

```
B1: begin real a, b, c;
    . . .
B2: begin real d, e;
    . . .
    end B2;
B3: Begin real f, g;
    . . .
    end B3;
    . . .
    end B1;
```

В этом примере переменные a, b, c являются глобальными для блоков $B2$ и $B3$. Переменные d и e являются локализованными только для блока $B2$ и не являются глобальными для блока $B3$. Переменные f, g локализованы только для блока $B3$ и в других блоках не могут быть использованы.

В этом случае блоки $B2$ и $B3$ являются независимыми.

```
A1: Begin real a, b;
    . . .
A2: begin real c;
    . . .
A3: Begin real d, e;
    . . .
    end A3;
    end A2;
  end A1
```


В этом примере переменные a и b являются глобальными для всех блоков и могут использоваться во всех блоках. Переменная c является глобальной по отношению блока $A3$ и может использоваться в блоках $A2$ и $A3$. Блоку $A1$ она недоступна. Переменные d и e могут использоваться только в блоке $A3$, т.к. они в нем локализованы.

```
C1: begin real a, b, c;
    . . .
C2: begin real a, b, d;
    . . .
    end C2;
    end C1
```

В этом примере глобальные величины a и b при входе в блок $C2$ теряют свою глобальность и значение, т.к. в этом блоке действуют локализованные переменные a и b. И только после выхода из блока $C2$ начинают действовать глобальные величины a и b, описанные в блоке $C1$.

```
D1: begin real a, b;
    . . .
M1: a:=b:=0; go to D1; go to M1;
    go to D2;  go to M2;
D2: begin real c, d;
    . . .
M2: c:=d+2; go to D1; go to D2;
    go to M2;
    end D2;
    end D1
```

Зак.Б-747/742ф.

Из этого примера видно, что из внутреннего блока D2 (подблока) можно передавать управление на любую метку, т.е. на метки D1, D2, M1, M2 , а из внешнего блока D1, можно передавать управление только на метки D1, D2 и M1; на метку M2 передача управления запрещена (оператор обведен пунктиром). Отсюда следует, что передача управления из внешнего блока во внутренний невозможна, в противном случае - разрешается.

```
begin  real a, b;
       . . .
M1:    a:=b+2;
begin  real c;
       . . .
M1:    c:=0,5;
       go to M1;
end;
go to M1;
end
```

В этом примере оператор go to M1, стоящий перед первым end (он помечен точками), будет передавать управление только на метку M1 внутреннего блока, т.е. на оператор c:=0,5; В этом случае переход из внутреннего блока во внешний на метку M1 невозможен.

Второй оператор go to M1, стоящий после первого end (он помечен пунктиром), передает управление на метку M1, стоящую перед оператором a:=b+2;

```
begin real a;
      . . .
   M1:  a:=5;
end;
begin real b;
      . . .
   M2:  b:=4;
end
```


При таком написании блоков выход из них должен происходить через end , а вход в блок - через begin, т.е. выход из середины одного блока на начало или в середину другого блока невозможен. Для того чтобы можно было выходить из одного блока (середины) на начало другого блока, необходимо эти блоки окаймить внешним блоком, т.е. ввести вначале begin и в конце end. Например:

```
   Begin real c;
E1: begin real a;
    . . .
M1: a:=5; go to E2;
    . . .
    end;
E2: begin real b;
    . . .
M2: b:=4; go to E1;
    . . .
    end;    end
```

В описаниях массивов граничные пары могут быть любым числом как положительным, так и отрицательным, например:

```
integer array a[0:20];
        array b[-20:-10];
```

Количество граничных пар в одном массиве может быть любое: real array a[1:10, -2:10, 1:40, 1:50, 0:100]

Если условно пронумеровать граничные пары слева направо, начиная с 1 , то можно сказать, что в МОЗУ массив расположится в следующем порядке.

Сначала в МОЗУ располагаются все элементы массива 5-го индекса, затем 4-й индекс увеличивается на 1 , и снова 5-го индекс пробегает все свои значения и т.д. После того как 4-й индекс пробежит все свои значения, 3-й индекс

увеличивается на 1; 4-й индекс принимает значение 1, а 5-й снова пробегает все свои значения и т.д. до тех пор, пока 1-й индекс пробежит все свои значения.

Граничные пары могут быть идентификаторами, арифметическими выражениями. В этом случае все идентификаторы, входящие в граничные пары, должны быть описаны во внешнем блоке и там же присвоены им численные значения. Например:

```
begin integer i,j,k; integer array c[1:10];
  input (i,j); k:=5; input (c);
begin array a[i:j+5], a1[1:2×k+i↑y], b[0:c[2]];
  . . .
end;
end
```

Если в процессе счета необходимо периодически менять размер некоторых массивов, то нужно сделать следующее:

а) пометить блок, который имеет массивы переменной длины (динамические массивы) меткой;

в) в теле этого блока присвоить новые значения идентификаторам, входящим в граничные пары;

с) передать управление на метку блока.

Например:

```
Begin integer n, m;
    n:=2; m:=10;
M1: Begin array a[1:n], b[1:2×m];
        real c,d; integer i,j;
  . . .
n:=10;  m:=5;
go to M1;
end;
end
```


При первом входе в блоке M1 массив a имеет длину 2 ячейки, а массив b – 20 ячеек. После присвоения переменным n и m новых значений (10 и 5) и обязательной передачи управления на метку M1 массив a займет 10 ячеек, а массив b – 10 ячеек. При этом надо учесть, что старые значения (числовые) этих массивов не будут эквивалентны новым после перераспределения памяти, т.е. нельзя будет использовать старые значения массивов. Если необходимо после перераспределения длин массивов оставить их старые значения, то вначале делается пересылка содержимого каждого массива (a и b) в другие массивы, которые обязательно должны быть описаны во внешнем блоке, содержимое записывается на барабан, или ленту, затем, после перераспределения памяти, производится их восстановление. Переменные c, d, i, j тоже теряют свои значения (старые) после перераспределения памяти. Чтобы этого избежать, необходимо переменные c, d, i, j описать во внешнем блоке.

После выхода из блока значения всех переменных, описанных в блоке, утрачиваются, т.к. на их местах в МОЗУ расположатся переменные следующего блока, в который мы войдем после выхода из предыдущего блока. Даже если мы войдем в тот же самый блок, из которого только что вышли, и в нем не меняем размеры массивов, то, в общем случае, значения переменных могут быть утрачены.

Для того чтобы значения переменных (всех или некоторых) при повторном входе в блоке сохранились такими же, как при выходе, эти переменные должны быть описаны как собственные, например:

```
Begin real a, b; own real c, d;
    own integer i, j, k;
own real array L[1:20];
  . . .
end
```

В этом примере переменные a, b теряют свои значения при выходе из блока, а переменные c, d, i, j, k, l сохраняют их.

Здесь надо обратить внимание на то, что при описании массивов можно опускать символ real , т.е. можно писать array a[1:10], что эквивалентно real array a[1:10].

При описании массивов как собственных отбрасывание символа real не допускается, т.е. всегда надо писать

own real array a[1:10], а не own array a[1:10].

§ II. Комментарий

Для пояснения алгоритма программы или других замечаний в программу можно вставлять текст из основных символов, который не влияет на выполнение программы, а служит лишь комментарием к ней.

Для этих целей существует два правила:

I) любой текст, не содержащий символа „;" и заключенный между символами

comment и ;

можно поместить после ; или после begin.

Например:

```
a:=b+c; comment переменная b является
корнем уравнения; x:=a↑2; go to M;
begin comment в этом блоке рассчитывается величина отклонения расчётной
скорости от номинальной;
array a[1:10];
end
```

2) любой текст, не содержащий „;" или end, или else , можно поместить между символами:


```
end и ;
end и end
end и else
```

Например:

```
Begin array a[1:10]; ...
for i:=1 step 1 until 10 do begin
a[i]:=i+3; b[i]:=c×2; end цикла по i;
if a>b then begin for i:=1 step 1 until 40 do
Begin a[i]:=3×b; b:=c↑2 end формирования
end расчёта при выполнении условия else
b:=0;
...
end программы
```

Символ „;" перед end можно ставить и можно не ставить, но после символа end он (символ „;") обязательно ставится, если за end не следует:

end, комментарий или else

Если за end следует комментарий, а за комментарием следует оператор, то после комментария ставится символ „;" . В конце программы после символа end символ „;" не ставится.

§ I2. Процедуры

Оператор процедуры является самым сложным и многообразным оператором входного языка.

Во входном языке можно использовать как операторы процедур, так и процедуры - функции.

I. Операторы процедур

Чтобы использовать оператор процедуры, необходимо в заголовке блока (программы) среди описаний поместить описание процедуры вида:

```
procedure max (x, y, z);
integer x; array y; real z;                    (1)
begin integer i, j, k; array a, b[1:20];
...
end;
```

После символа *procedure* пишется имя (идентификатор) процедуры, затем в круглых скобках перечисляются формальные параметры через запятые, в конце ставится символ „;". В качестве формальных параметров, как правило, используются входные и выходные значения процедуры.

Формальными параметраии могут быть любые буквы и идентификаторы независимо от того, описаны эти буквы и идентификаторы в начале программы или блока, т.к. они формально описывают, что нужно задать при обращении к процедуре и что можно получить после обращения к ней.

За списком формальных параметров следует спецификация их. Спецификация предназначается для того, чтобы определить, к какому классу относится каждый формальный параметр: то ли к переменной типа *integer*, то ли к переменной типа *real*, то ли к переменной типа *boolean* и т.д.

Следом за спецификацией следует тело самой процедуры. Так как тело самой процедуры является блоком, то в нем могут быть локализованы некоторые идентификаторы путем описания их в теле процедуры.

Например:

```
procedure sh(x,y);
real x,y;                                      (2)
begin real z;
z := exp(x); y := (z-1/z)/2;
end;
```

Категорически запрещается описывать формальный параметр в теле процедуры, т.е. описывать в теле процедуры идентификатор, похожий на идентификатор одного из формальных параметров. Это приведет к неправильной работе процедуры.

Например:

```
procedure sh (x,y);
real x,y;                                      (3)
begin real x;
x := exp(x); y := (x-1/x)/2;
end;
```

В этом примере переменная x описана в теле процедуры $sh(x,y)$, которая совпадает с формальным параметром этой же процедуры. Поэтому значение формального параметра недоступно телу процедуры, т.е. телу процедуры будет доступна только локализованная переменная x. Если мы обратимся к такой процедуре следующим оператором $sh(10,z)$, то в теле процедуры вместо x не будет подставлена величина 10, а будет выбираться величина, которая в данный момент находится в МОЗУ.

Входной язык системы допускает возможность исключения из процедуры спецификаций.

Например, описание процедуры $sh(x,y)$ можно записать так:

```
procedure sh(x,y);
begin real z;
z := exp(x);
y := (z-1/z)/2;
end;
```

(4)

Запись (4) эквивалентна записи (2).

При обращении к процедурам, имеющим приведенные выше описания, в теле происходит замена всех формальных параметров на фактические.

Например: имеем описанную процедуру $aRt(x,y,z)$

```
begin integer i,j,k; array a[1:10];
    real l, v;
procedure aRt(x,y,z);
  real x,y; array z;
begin real d,b,c; integer i;
    d := 3*x +4*y;
  for i:=1 step 1 until 10 do
      z[i] := d*i;
  end;
  . . .
aRt(10, l, a);
end
```


При обращении к этой процедуре оператором $Art(10,l,a)$ тело процедуры примет вид

```
d := 3*10 + 4*l;
for i:=1 step 1 until 10 do
a[i] := d*i;
```

т.е. вместо формальных параметров x,y,z будут подставлены соответственно $10, l, a$.

В качестве фактических параметров всегда можно применять:

- числа;
- идентификаторы;
- арифметические выражения;
- элементы массива;

Спецификаторами могут быть:
label (метка), switch (переключатель), string (строка), real, integer, boolean, procedure, real procedure, integer procedure, boolean procedure, array, real array, integer array, boolean array.

Если необходимо выйти из процедуры по какому-либо условию в основную программу , т.е. в программу, в которой было обращение к процедуре, то в качестве формального параметра используется метка. В теле процедуры при выполнении какого-либо условия передается управление на эту метку. В теле самой процедуры такой метки не должно быть.

```
procedure  BAC (a, b, M); real a; b; label M;
begin real x,y; x:=a+b↑2; if x=0 then go to M;
    end;
```

При обращении к данной процедуре оператором BAC(i,j,k); в случае равенства переменной x нулю будет передано управление на метку K основной программы.

Если при выполнении каких-либо условий выход из тела процедуры должен осуществлятся в разные места основной программы, то в качестве формального параметра можно применить переключатель (switch)

```
procedure  AB(x, y, L);
  real x,y; switch L;
begin real  a, b;
    a:= x+y↑3/2; b:=a+a/3;
 if a=b then go to L[1] else
 if a=0 then go to L[2] else go to L[3];
    end;
```

Если мы обратимся к этой процедуре оператором AB(c,d,L1), где переключатель L1 описан как switch L1:=M1, 2, MS; то в зависимости от условий, проверяемых в процедуре, управление будет передаваться то на метку M1, то на метку 2 или на метку MS описанной программы.

Процедура может выдавать любой текст, если в качестве формального параметра использовать string (строка).

```
procedure mp (x,y,z);
  real x,y; string z;
begin real  a, b;
```



```
    a:= 3.5;  b:= a+40.4;
output ('T', z, '/');
end;
```

При обращении к процедуре mp(x,y,z) оператором mp(6,10,'нормальная␣работа'); на печать будет выдан текст:

нормальная␣работа

В качестве формального параметра может использоваться процедура.

```
procedure  CM(x,y,f);
real x,y; procedure f;
begin real a,b; integer i;
a:=3.52;  i:=a×x↑2; f(a,b,4,6);
end;
```

При обращении к процедуре оператором CM(c,3,Lup); где процедура Lup(x,y,z,p) описана с четырьмя формальными параметрами, процедура CM обратится к процедуре Lup, подставив вместо формальных параметров x,y,z,p фактические a,b,4,6.

В теле одной процедуры могут быть описаны несколько процедур.

```
procedure  CY(x,y);  real x,y;
    begin real a,b;
```

Зак.Б-747/742ф.

```
procedure AB(x,y);
begin real a; x:=y↑3+5; end;
    procedure AC(z);
begin real b; z:=2; end;
  ...
end;
```

Входной язык допускает статическую рекурсию, т.е. процедура f использует процедуру $f1$, , а процедура $f1$ использует процедуру $f2$, процедура $f2$ использует процедуру $f3$ и т.д.

```
procedure f(x);
  begin real a;
procedure f1(y);
  begin real b;
procedure f2(z);
begin real c; c:=2;
  z:=c↑2; end;
  ...
end;
  ...
end;
```

2. Процедуры-функции

Процедуры-функции отличаются от операторов процедур тем, что носителем результата работы самой процедуры является ее имя (идентификатор).

Поэтому процедурами-функциями должны быть только те процедуры, результатом которых является только одна переменная любого типа.

При описании процедуры перед символом procedure ставится описатель типа (real, integer, Boolean)

Если результат работы процедуры определен как real, то перед символом procedure ставится символ real и т.д.

В теле процедуры идентификатору процедуры необходимо присвоить значение результата ее работы.

```
real procedure КОП(x,y); real x,y;
begin real a;   a:= x↑2 +y/2;
                КОП := a↑2 + a↑3;
end;
```

В теле процедуры идентификатор процедуры должен встречаться только в левой части оператора и категорически запрещается использовать его в правой части, т.е. нельзя писать

```
integer procedure курс(x,y);
    begin real a;
a:= x/2 ;  курс :=0;  курс:=курс+a↑4+y/3;
end;
```

В теле программы обращение к описанной процедуре-функции осуществляется указателем функции. Так, обращение к процедуре КОП(X,У),приведенной выше, можно писать так:

```
a:= коп (5,3);
b:= c+коп(d,e) ×2;
f:= b↑2 × коп (5,3);
```

3. Использование символа value

Список значений существенно меняет специфику замены формальных параметров фактическими при обращениях к процедурам. В него могут быть включены некоторые (или все) формальные параметры, относящиеся к классам простых переменных, массивов или меток.

Список значений может отсутствовать совсем.

Если какие-либо величины описаны как value ,то описание типа (real, integer, boolean, array и т.д.) должно быть обязательно. Причем описание типа обязательно ставится после списка значений, а не перед ним.

```
procedure A(x, y, z);
value x,y; real x; array y;
  begin real a;
    a:= x↑2+y[2]×z;
    . . .
  end;
```

И нельзя писать так:

```
procedure A(x, y, z);
real x; array y; value x,y;
  begin real a;
    a:= x↑2+y[2]×z;
    . . .
  end;
```

Если формальный параметр, не являющийся результатом работы процедуры, встречается в левой части оператора присваивания, то он должен быть описан как value.

Список значений применяется для того, чтобы значения фактических параметров не изменялись после выхода из процедуры, т.е. с каким значением фактического параметра вошли в процедуру, с таким же и вышли.

Например:

```
real procedure Det (B,n); value B,n; array B;
                                  integer n;
  begin integer i,j,k; real s;
    for i:=1 step 1 until n-1 do
    for j:=i+1 step 1 until n do
    for k:=i+1 step 1 until n do
B[j,k]:=B[j,k]-B[i,k]/B[i,i]×B[j,i];     s:=1;
    for k:=1 step 1 until n do s:=s×B[k,k];
    Det:=S;
  end;
```

При обращении к данной процедуре указателем функции Det (a, 20) компоненты матрицы a не изменят своих значений .

Если же из описания процедуры Det (B,n) исключить список значений value B,n; то при обращении указателем функции Det (a,20) компоненты матрицы a изменят свои значения.

Замена формальных параметров, содержащихся в списке значений, на фактические производится подстановкой вычисленного значения фактического параметра, а не самого фактического параметра.

Так, если обратиться к процедуре Det(B,n), имеющей список значений value B,n указателем функции Det (a,c) , то

транслятор оттранслирует эту процедуру в виде

```
real procedure Det(b,n); array b[1:n,1:n] ; integer n ;
begin integer i,j,k; real s; n:=c; for j:=1 step 1 until n do
  for i:=1 step 1 until n do b[j,i]:=a[j,i];
  for i:=1 step 1 until n-1 do
  for j:=i+1 step1 until n do
  for k:=i+1 step1 until n do
 b[j,k]:=b[j,k]-b[i,k]/b[i,i]xb[j,i]; s:=1;
    for k:=1 step 1 until n do
        s:=sxb[k,k]; Det:=s;
     end;
```

В случае отсутствия списка значений `value b, n;`

```
 real procedure Det (b,n)
begin integer i,j,k; real s;
  for i:=1 step 1 until c-1 do
  for j:=1 step 1 until c do
  for k:=1 step1 until c do
a[j,k]:=a[j,k]-a[i,k]/a[i,i]xa[j,i]; s:=1;
  for k:=1 step 1 until c do  s:=sxa[k,k];
     Det:=s;
  end;
```

В список значений должны входить только входные формальные параметры, а не выходные, т.к. им в теле процедуры присваиваются значения (результат).

Применение списка значений в процедуре несколько сокращает время ее работы.

4. Использование комментария

Если совокупность формальных параметров в заголовке процедуры (или совокупность фактических параметров в операторе процедуры) содержит два или более параметров, вместо запятой, отделяющей один параметр от другого, можно употребить комментарий особой конструкции.

Например:

```
procedure МУП (a,b,d, i,j, k);
```

Здесь
- a - идентификатор массива;
- b - размерность массива;
- d - максимальный элемент;
- i - номер строки;
- j - номер столбца;
- k - логическая величина.

Для того чтобы сделать заголовок более удобным для чтения, его можно записать в виде

```
procedure МУП (a) размерность: (b) максимальный
 элемент: (d) номер строки: (i) номер столбца:
(j) логическая величина: (k);
```

Использование такой возможности может сделать программу более понятной не знакомому с ней исполнителю. Нельзя использовать эту возможность для первого параметра и, следовательно, нельзя использовать в процедурах с одним формальным параметром.

II. СТАНДАРТНЫЕ ПРОЦЕДУРЫ

§ I. Стандартная процедура INPUT

Оператор *input* осуществляет ввод в машину чисел, логических значений и текста.

Общий вид оператора *input*:

input (список фактических параметров).

Список фактических параметров (объект ввода) представляет собой последовательность разделенных запятыми идентификаторов массивов, простых переменных и переменных с индексами.

Каждому фактическому параметру должна соответствовать одна группа данных. Простым переменным и идентификаторам массивов соответствуют группы числовых или логических данных. Переменным с индексами соответствуют группы текстовых данных. Количество данных в группе не должны превосходить размеры объектов ввода, которые определяются описаниями, но могут быть меньше их.

Группы данных

1. Группа числовых данных

Группа числовых данных состоит из чисел, разделенных запятыми. Заканчивается группа символом " ; ".

Число может иметь комментарий к себе, который пишется перед ним. Комментарий начинается с буквы, может содержать любой символ языка, кроме символа „:" , символа „=" и символа „:=", которыми он заканчивается. При вводе комментарий отбрасывается.

Пример . Пусть группа числовых данных из двух чисел имеет вид: *длина: 30, ширина = 42;*

Эта группа числовых данных эквивалентна группе:

30, 42;

Числа пишутся в форме, принятой в Алгол-60:

35.6, 0.36$_{10}$-5, 1$_{10}$8, $_{10}$8, 25, -3.14;

Допускается ввод чисел, пробитых для машины БЭСМ-6 (с признаком ↑). В этом случае запятые, отделяющие числа одно от другого, отсутствуют, но символ „;" в конце группы должен быть:

↑35.6 ↑0.36$_{10}$-5 ↑$_{10}$8 ↑25 ↑-3.14;

Возможен ввод чисел, пробитых для машины М-20 (М-220). При этом числа следуют одно за другим, а в конце группы ставится особый разделитель:

777 7777 7777 7777

Этот разделитель играет роль символа „;" Допускается любая комбинация чисел, пробитых как для машины БЭСМ-6, так и для машины М-20 (М-220).

2. Группа логических данных

Группа логических данных представляет собой последовательность логических значений *true* и *false* , разделенных запятыми. Как и в случае числовых данных, логические значения могут быть снабжены комментарием, который при

вводе отбрасывается.

Группа данных заканчивается символом „;".

Логические значения true и false пишутся подчеркнутыми и пробиваются как основные символы Алгол-60.

Например:

true, false, false, true;

3. Группа текстовых данных.

Группа текстовых данных пишется в виде строки.

Например: ‘величина ␣ ␣ x =’

При вводе текста символы строки (включая внешние кавычки) заменяются целыми значениями в соответствии с кодами ГОСТа и присваиваются последовательно элементам массива, начиная с элемента, указанного в объекте ввода.

Пример:

```
begin array a[1:25]; input(a[1]);
end
```

Группа текстовых данных имеет вид:

‘значение ␣ x =’

Тогда в массиве a будут находиться следующие величины:

a[1]-26(‘) a[5]-55(т) a[9]-37(е) a[13]-27(>)
a[2]-39(з) a[6]-37(е) a[10]-15(␣)
a[3]-45(н) a[7]-45(н) a[11]-53(х)
a[4]-32(а) a[8]-40(и) a[12]-21(=)

Ввод текста в массив начинается с символа „‘" и продолжается до символа „’", т.е. вводится текст, находящийся между открывающейся кавычкой „‘" и закрывающейся „’".

```
begin boolean a,b,c;
      input (a,b,c);
end
```


И пусть пробиты следующие логические группы:

true; false; true;

Тогда после ввода величина a примет значение true ("I" в 40-м разряде), величина b - значение false ("0" в 40-м разряде) и величина c - true ("I" в 40-м разряде).

§ 2. Стандартная процедура output

Оператор output предназначается для вывода чисел, логических значений и текста на печатающее устройство (АЦПУ) и для соответствующего редактирования всей печатаемой информации. Это редактирование включает в себя, в частности, оформление страниц. Программист может с помощью стандартного оператора MARG задать желаемый вид страницы. Если оператор MARG не используется, то вся печать осуществляется на страницах стандартного вида: см. рис. 1. стр. 44

Страницы отделяются друг от друга строкой из 128 прочерков (— — — — — —). В нижнем правом углу страницы печатается номер страницы.

Переход на новую строку или новую страницу происходит после заполнения предыдущей либо при появлении в выводимом тексте и в группе размещения (см. ниже) явного приказа о таком переходе.

Выполнение нового оператора output не вызывает автоматического перехода на новую строку, если на предыдущей строке осталось еще достаточно свободного места, на нем и будут печататься новые результаты.

Слово "достаточно" имеет здесь следующий смысл:

при выводе текста, если строка не заполнена, то вывод продолжается на нее;

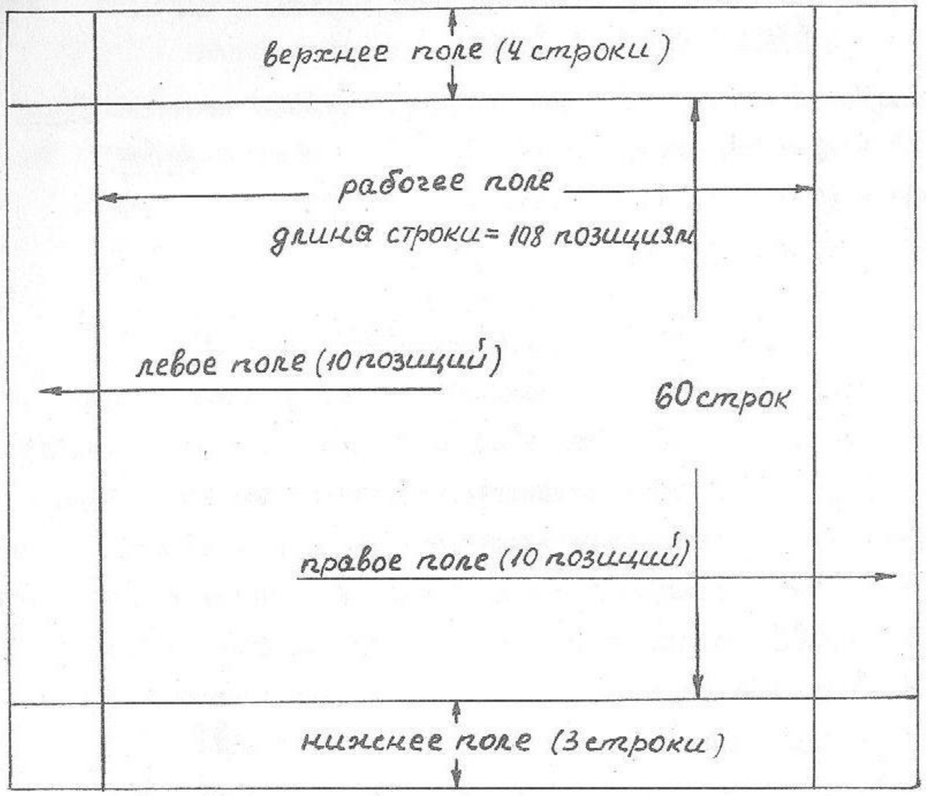

Рис. 1

при выводе значений, если размер поля, определяемый числовым или логическим форматом и уменьшенный на число позиций правого поля, больше оставшегося свободного места в строке, то происходит переход на новую строку.

Список фактических параметров оператора *output* представляет собой одну или несколько разделенных запятыми групп, каждая из которых задает либо вывод числовых значений, либо вывод логических значений, либо текста, либо определяет размещение.

I. Группа вывода числовых значений

Эта группа состоит из числового формата и объектов вывода. Объектом вывода могут быть либо арифметическое выражение, либо идентификатор переменной или массива, либо переменная с индексом.

Числовой формат определяет вид вывода чисел на печать:

а) в экспоненциальной форме;

б) в виде десятичной дроби.

Вывод чисел в экспоненциальной форме

В этом случае оператор *output* имеет вид

output ('E-.DDD$_{10}$+DDBB', a+b, c, d[2]);

Здесь числовой формат 'E-.DDD$_{10}$+DDBB' имеет следующий смысл.

E – признак экспоненциальной формы выдачи;

– – знак минус. Если выводимое число отрицательное, то перед ним печатается знак минус, если выводимое число положительное, то никакого знака не печатается (это правило относится и к знаку порядка);

. – десятичная точка (печатается .);

D – означает десятичную цифру (на печати в соответствующей позиции печатается цифра).

$_{10}$ – означает порядок (печатается $_{10}$);

+ – знак плюс. Если порядок числа положительный, то печатается знак плюс, если отрицательный – то минус (это правило относится и к знаку числа);

B – означает пробел (на печати в соответствующей позиции ничего не печатается).

Пример:

output ('E-.ДДД$_{10}$-ДДВВ'; a+b, c, 6, d[2]);

Пусть $a = 3.56$ $b = 4.2$

$c = -0.003$ $d[2] = -5.68$

тогда на печать будут выданы следующие величины:

␣.7760$_{10}$␣01␣␣-.3000$_{10}$-0.2␣␣.6000$_{10}$␣01␣␣

-.5680$_{10}$␣01␣␣

Эти же числа при выводе в другом формате будут иметь другой вид:

output ('E+ДД.ДВДВДВДВ$_{10}$+ДВДВВВ'; a+b, c, 6, d[2]);

+77.6␣0␣0␣$_{10}$-0␣1␣␣␣

-30.0␣0␣0␣0␣$_{10}$-0␣4␣␣␣

+60.0␣0␣0␣0␣$_{10}$-0␣1␣␣␣

-56.8␣0␣0␣0␣$_{10}$-0␣1␣␣␣

Если объект вывода представляет собой идентификатор массива, то на печать выводится весь массив согласно его описанию.

Например, необходимо вывести на печать массив $d[1:5]$. Для этого пишется следующий оператор:

output ('E-.ДДД$_{10}$+ДДВВ'; d);

На печать будут выданы все пять чисел массива, в то время как в предыдущем примере на печать выводился только один элемент массива $d[2]$

Вывод чисел в виде десятичной дроби

Оператор output имеет вид

output ('Z±ДДД.ДДДВВ'; a+b, c, 6, d[2]);

output ('y±ДДД.ДДДВВ'; a+b, c, 6, d[2];

Здесь Z - признак десятичной дроби с "подавлением" старших незначащих нулей в целой части числа;

y - то же самое, что и Z, только без "подавления" нулей;

+, −, ., Д, В имеют тот же смысл, что и при формате E.

При таких форматах ранее выдаваемые числа будут иметь следующий вид:

при формате Z

7.760␣␣ 6.000␣␣

-0.003␣␣ -5.680␣␣

при формате y

007.760␣␣ 0 06.000␣␣

-000.003␣␣ -005.680␣␣

Если при выводе числа в формате вида Z и y оно не умещается в задаваемый формат, то старшие разряды числа отбрасываются, а младшие округляются.

Например, при выводе числа 2753.827 в формате 'Z+ДД.ДД' будет напечатано +53.83. Если в формате не указан знак, то печатается абсолютное значение (это относится и к формату E).

Можно печатать только целую часть числа. Для этого необходимо задать формат вида

'Z±ДДДДД'

или

'y±ДДДДД'

Пример: output ('Z-ДДД'; a+b, c, 6, d[2]);

В этом случае напечатаются числа:

8

0

6

-5

Размер поля (количество позиций), занимаемого на печати выводимым числом, равняется общему числу символов, использованных при задании формата. Так, длина поля для формата 'E-D.DDDDDBDB$_{10}$+DBB' равна 15 позициям, формата 'Z+DDDD.DDDDDBBB' - 14 позициям. Для удобства записи несколько подряд стоящих букв D или B можно заменить на конструкцию вида

nd или nB,

где n – целое без знака (число).

Так, последние форматы можно записать в виде

'E-D.4DB1DB$_{10}$D2B'

'Z+4D.5D3B'

Для упрощения (в ряде случаев) работы программиста введены 5 стандартных числовых форматов:

'E', 'Y', 'Z', 'Yi', 'Zi'.

Они имеют следующий смысл:

'E' – эквивалентно 'E-.10D$_{10}$+2D2B'

'Y' – — " — · — ·· — 'Y-6D.6D4B'

'Z' – — · — " — ·· — 'Z-6D.6D4B'

'Yi' – — " — " — · — 'Y-6DB6D4B'

'Zi' – — " — · — · — 'Z-6DB6D4B'

Размер поля каждого из этих форматов равен 18.

Признак конца группы вывода числовых значений

Конец группы вывода числовых значений, точнее конец списка объектов вывода, определяется очередным фактическим параметром, являющимся строкой, а если таковой до конца списка параметров не встретится, то закрывающейся скобкой оператора output.

Пример:

output ('Z+DDD.5D2B', a, b, c, d, 'Y+6D.7D', e, f);

Здесь по формату 'Z+DDD.5D2B' выводятся величины a, b, c, d, а по формату 'Y+6D.7D' – e, f.

2. Группа вывода логических значений

Эта группа имеет четыре вида:

1) output ('L', a, b, c); 3) output ('L5F', a, b, c);

2) output ('LF', a, b, c); 4) output ('LB', a, b, c);

Объекты вывода a, b, c представляют собой либо логическое выражение, либо идентификатор массива, простой переменной или переменной с индексами. Очередность вывода элементов массива определяется их лексикографической (по индексам) упорядоченностью.

В операторах 2,3 и 4-го видов после L и перед символом „'" может находиться любое число букв B, означающих, что при печати логического значения надо поставить перед ним и после него соответствующее количество пробелов (как и в числовом формате, несколько подряд стоящих букв B можно заменить на nB, где n – целое без знака).

Зак.Б-747/742ф.

Пример: output ('Ł5BF4B', a, b, c);
output ('Ł4B5F2B', a, b, c);

Смысл конструкций 'ŁF', 'Ł5F' и 'ŁG' следующий:

'ŁF' означает , что логические значения выводятся в виде T или F (true - в виде T , а false - в виде F);

'Ł5F' означает, что логические значения true выводятся в виде true , а false - в виде false, причем при выводе true после этого слова делается один пропуск;

'ŁG' означает, что логические значения true выводятся в виде 1 , а false - в виде 0;

логический формат 'L' эквивалентен формату 'LG'.

Пример. Пусть a = true, b = false, c = true,

тогда, применяя соответствующий формат (вид) печати, получим: output ('Ł', a, b, c);

```
1
0
1
output ('Ł2BF2B', a, b, c);
␣␣T␣␣
␣␣F␣␣
␣␣T␣␣
output ('Ł3B5F3B', a, b, c);
␣␣␣true␣␣␣␣
␣␣␣false␣␣␣␣
␣␣␣true␣␣␣␣
output ('ŁG4B', a, b, c,);
1␣␣␣␣
0␣␣␣␣
1␣␣␣␣
```


Признак конца группы вывода логических значений

Конец группы, точнее конец списка объектов вывода, определяется очередным фактическим параметром, являющимся строкой, а если таковой до конца списка параметров не встретится, то закрывающей скобкой оператора output.

Пример:

output ('ŁF', a, a∧b, 'Ł5F', bvc, d);

Здесь выражения a и a∧b выводятся по формату 'ŁF', а выражения bvc и d - по формату 'Ł5F'.

3. Группа вывода текста

Эта группа имеет два вида:

output ('T', 'величина␣x='); output ('T', a[i]);

Здесь 'T' - признак текстовой информации.

В первом виде выдачи выводимый текст пишется в виде строки (т.е. берется в кавычки) вслед за форматом 'T'.

Пример: output ('T', 'величина x=');

На печать будет выдан текст: величина x=

При выводе на печать текста внешние кавычки, окаймляющие текст, опускаются.

При выдаче подряд нескольких текстов перед каждым из них обязательно ставится формат 'T'.

Пример: output ('T', 'величина␣x=', 'T', 'величина␣y=');

но нельзя писать так:

output ('T', 'величина␣x', 'величина␣y=');

Длина текста не ограничивается.

Во втором виде выдачи выводимый текст помещается в массив, начиная с указанного индекса. В этом случае все элементы массива, начиная с указанного (в качестве формального параметра), представляют собой целые числа, являющиеся кодами (по ГОСТу) последовательных символов выводимой строки ("строка" в смысле Алгола, т.е. первым числом должен быть код символа „ ' " (26), последним - код символа „ ' "(27). Независимо от размера массива на печати выдается текст, находящийся между числами 26 и 27, т.е. между открывающей и закрывающей кавычками. Оператор переводит указанные числа в символы ГОСТа и выдает их на печать.

Пример. Имеем массив $a[1:100]$, содержащий числа:

$a[1]=0 \quad a[2]=-2 \quad a[3]=-100 \quad a[4]=26$ (')

$a[5]=39$(ж) $a[6]=45$(м) $a[7]=32$(а)

$a[8]=35$(з) $a[9]=37$(е) $a[10]=45$(м)

$a[11]=40$(и) $a[12]=37$(е) $a[14]=53$(х)

$a[15]=21$(=) $a[16]=27$(') $a[17]=-2.3$

...

$a[100]=45.356$

Оператор $output('T', a[4])$ выдает на печать следующий текст:

значение ⌴ x =

В одном массиве может находиться любое количество текстов, но все они должны начинаться с символа „ ' " и

кончаться символом „ ' " . В операторе output в этом случае переменная с индексами должна указать начало текста в массиве, т.е. символ „ ' ".

Пример. Если в массиве $a[1:200]$ находятся четыре текста, которые начинаются с индексов $1, 40, 70, 150$ соответственно, то выдача их на печать осуществляется оператором вида

$output('T', a[1], 'T', a[40], 'T', a[70], 'T', a[150]);$

Указание формата 'T' перед каждым объектом вывода (переменной с индексами) обязательно. Следовательно, оператор вида

$output('T', a[1], a[40], a[70]\ a[150]);$

писать нельзя.

Если в массив поместить формат 'T' вместе с текстом, то можно написать оператор вида

$output(a[1]);$

где $a[1]$-(')

$a[2]$-(T)

$a[3]$-(')

$a[4]$-(')

... текст ...

$a[n]$-(')

На строки, используемые в группах вывода текста (и, соответственно, на последовательности целых чисел - элементы массива), накладываются следующие ограничения:

- внутри строк не могут встречаться символы „ ' " и „ ' ";

- символ „ : " используется лишь для изменения смысла следующего за ним символа, и после него могут стоять только символы B, /, x, 1, 2, 3 (x-знак умножения).

Соответствующие комбинации означают:

:B задает пробел;

:/ задает переход к началу следующей строки;

:X задает переход к началу следующей страницы;

:1 задает вывод символа „'" ;

:2 задает вывод символа „'" ;

:3 задает вывод символа „:".

Пример. Оператор вида output('T','величина:B:1X:2');

выдает на печать следующий текст:

величина _ 'X'

4. Группа размещения

Группа размещения состоит из одного или двух фактических параметров. Признаком, определяющим, сколько параметров входят в размещение, является тип фактического параметра, следующего за форматом размещения: если параметр является строкой, то - один, если параметр является выражением, то - два.

Формат размещения представляет собой строку, в кавычках которой содержится любая последовательность из символов B, /, X и целых без знака.

Пример: '15B'
'2/10B'
'3X'

Смысл используемых символов следующий:

B означает пробел;

/ означает переход на новую строку;

X означает переход к началу следующей страницы.

Целое без знака, стоящее перед соответствующим символом, - количество повторений данного символа.

Выполнение формата размещения состоит в последовательном (слева направо) выполнении действий, указанных в формате размещения. Так, вышеприведенные примеры означают:

- сделать 15 пробелов;
- дважды перейти на новую строку и отступить на 10 позиций (сделать 10 пробелов);
- трижды перейти на новую страницу (т.е.прекратить заполнение текущей страницы и пропустить еще две чистые).

В группах размещения с арифметическим выражением последнее определяет число повторений формата размещения.

Пример:

output ('5B2/'; '2/4B'; 'B', 5, '/', a);

Этот оператор выполнит следующие действия:

- сделает пять пробелов и дважды перейдет на новую строку;
- дважды перейдет на новую строку и сделает 4 пробела;
- сделает пять пробелов;
- перейдет на новую строку a раз.

output ('2B3/', a+b);

Здесь оператор сделает 2 пробела и трижды перейдет на новую строку, затем все это повторит (a+b) раз.

В приведенном выше описании первый фактический параметр каждой группы является строкой. В тех случаях, когда это не изменит разбиение параметров на группы, вместо строки можно указать переменную с индексом, определяющую место в массиве,

где содержатся соответствующим образом закодированные символы, аналогично тому, как это описано в разделе "Группа вывода текста".

Пример: output (a[1]);

где a[1]-('c)
a[2]-(3)
a[3]-(B)
a[4]-(2)
a[5]-(1)
a[6]-('')

Этот оператор произведет следующие действия:
сделает три пробела и дважды перейдет на новую строку.

В одном операторе output могут быть применены все группы выдачи (числовые, логические, текстовые и размещения) в любой комбинации. Например:

✓ output('E';a,b,'Z';c,d,'У';e,'T,''конец';'X';'E',...);

Здесь по формату 'E' отпечатаются величины a и b, по формату 'Z' – c и d, по формату 'У' – e, затем текст конец, потом осуществится переход на новую страницу и т.д.

Если выдать на печать символ c n раз подряд, то для этого необходимо сделать следующее:

описать массив a[1:n+2];

составить операторы:

for i:=2 step 1 until n+1 do a[i]:=c;
a[1]:=26; a[n+2]:=27;

null output ('T'; a[1]);
for i:=1 step 1 until n do output('T'; 'c');

где под c подразумевается любой символ АЦПУ (т.е. если c=11, то (-), если c=25, то (*) и т.д.).

§ 3. Стандартная процедура MARG

Оператор MARG используется для изменения размеров страницы. Этот оператор имеет вид

MARG(a,b,c,d,e,f,g);

где

a - размер левого поля;
b - длина печатаемой строки;
c - размер правого поля;
d - размер верхнего поля;
e - число строк в странице;
f - размер нижнего поля;
g - начальный номер страницы.

После выполнения оператора MARG размеры страницы примут те значения, которые указаны в операторе. Эти размеры страницы будут сохраняться до следующего оператора MARG.

Действие оператора MARG начинается только тогда, когда страница полностью заполнится. Следующая страница примет размеры, указанные в операторе.

Если необходимо перейти на новую страницу с новыми размерами, когда текущая еще полностью не заполнена, то необходимо написать следующие операторы:

```
MARG(a,b,c,d,e,f,g);
output ('X');
```

Если эти операторы написать наоборот, т.е.

```
output ('X');
MARG(a,b,c,d,e,f,g);
```

то заполнение текущей страницы прекратится, начнет заполняться новая страница, но ее размеры останутся прежними, и только после нее следующая страница примет новые размеры.

Фактический параметр g может отсутствовать. В этом случае нумерация страниц будет начинаться с первой.

Если $e=0$, то при выводе не будет происходить автоматического разбиения на страницы. Переход на новую страницу программист должен будет в этом случае задавать явно (с помощью символа X в группе размещения оператора *output* или с помощью пары символов :x при выводе текста).

Если $f=0$, то не будут печататься номер страницы и строка из I28 прочерков (разделяющая страницы друг от друга).

На некоторые параметры накладываются ограничения, а именно:

$$d \leqslant 40$$
$$e \leqslant 150$$
$$f \leqslant 40$$
$$a+b+c=128$$

Если в программе оператор *MARG* отсутствует, то выдача происходит на стандартную страницу (см. § 2).

§ 4. Стандартные процедуры READ и WRITE

Стандартные процедуры осуществляют обмен информацией между внутренней и внешней памятью машины. Оператор WRITE

записывает коды из ОЗУ на магнитный барабан или магнитную ленту , а оператор *READ* считывает коды с барабана или ленты в ОЗУ машины.

Общий вид операторов:

```
WRITE(a,b,c,d,e);
Read(a,b,c,d,e);
```

Первый параметр (a) всегда задает начальный адрес обмена во внутренней памяти машины. Он должен быть или идентификатором массива, или простой переменной. Его описание определяет количество слов, участвующих в обмене. Остальные параметры определяют начальный адрес обмена во внешней памяти машины. Они представляют собой арифметические выражения и означают следующее:

b - номер направления. При обмене с барабаном возможны лишь I-е и 2-е направления; при обмене с лентой 3,4,5 и 6-е направление;

c - математический номер ленты или барабана. Он может принимать значение от 0 до 7;

d - номер зоны (если обмен с лентой) или номер тракта (если обмен с барабаном).

Номера зон могут быть от 0 до 5II.

Номера трактов от 0 до 3I;

e -номер слова в зоне или тракте. Он может принимать значения от 0 до I023.

Если записываемый (считываемый) массив слов больше размера зоны (I024) слова, то происходит автоматический переход на следующую зону и т.д. Для записи (считывания) следующего массива слов вслед за предыдущим программист сам рассчиты-

вает, сколько зон занял предыдущий массив и определяет начальную зону и место, откуда начнется следующая запись (считывание) массива.

Пример. Необходимо записать массив a[1:2100] и массив b[1:10] друг за другом на ленту, начиная с зоны № 5.

```
write (a,3,3,5,0);
write (b,33,7,52);
```

Массив a[1:2100] займет полностью 5-ю зону (1024 слова), 6-ю (1024 слова) и 52 слова запишет в 7-ю зону (с 0 слова по 51 слово). Поэтому второй массив b[1:10] пишется в 7-ю зону, начиная с 52 слова.

Аналогичные операции проделываются и при работе с барабаном. Автоматического перехода с барабана на барабан или с ленты на ленту не производится.

При работе с лентой или барабаном необходимо предварительно "расписать" их.

При трансляции транслятор использует два барабана (I-0 и I-I, т.е. на первом направлении 0-й и I-й барабаны). Поэтому для экономии барабанов необходимо начинать запись с 0-го барабана 1-го направления, т.е. b=1, c=0.

Лента с математическим номером направления 3, математическим номером магнитофона 0 (короче - лента 3-0) используется транслятором как рабочая. Её можно использовать как рабочую и в операторах WRITE и READ, если обращение к оператору input происходит в начале программы и только раз (вернее, перед оператором WRITE). В противном случае использование этой ленты не рекомендуется, т.к.можно испортить исходные данные, которые будут вводиться

оператором input после обращения к оператору WRITE.

Категорически запрещается обращаться к лентам 3-1 и 6-7, т.к. первая используется для хранения значений собственных массивов, а вторая содержит архив индивидуальных библиотек.

Если в программе есть обращение и к ленте и к барабану, то нельзя использовать 31-й тракт 7-го барабана 2-го направления. Этот тракт используется как буфер при работе с лентой.

При работе с лентами необходимо в паспорте задачи указать общее количество лент, математические номера направлений и математические номера лент (включая и ленты, используемые транслятором).

Если записываемая на ленту информация не превышает одной зоны (вся информация пишется в одну зону), то запись сначала идет на барабан (буфер), и только после окончания счета происходит запись буфера на ленту в указанную зону.

При обращении к операторам READ, WRITE, последние используют лист ОЗУ как буферную память. Это надо учитывать при разработке алгоритма.

Ш. ОРГАНИЗАЦИЯ АРХИВА ИНДИВИДУАЛЬНЫХ БИБЛИОТЕК

В системе БЭСМ-АЛГОЛ имеется возможность организовать архив индивидуальных библиотек на магнитной ленте. В эти библиотеки помещаются процедуры, записанные на языке Алгол-60. В библиотеку можно записать любую программу, оформив ее как процедуру.

§ I. Операции с архивом

При работе с архивом можно выполнять следующие операции:

- подготовить магнитную ленту для архива;
- завести в архиве индивидуальную библиотеку;
- поместить алгоритм в индивидуальную библиотеку;
- выбросить алгоритм из индивидуальной библиотеки;
- запретить пользование и(или) изменение индивидуальной библиотеки;
- разрешить работу с индивидуальной библиотекой;
- изменить срок существования индивидуальной библиотеки;
- ликвидировать индивидуальную библиотеку.

Для удобства работы с архивом возможно получение непосредственно на машине следующей информации о библиотеке и отдельных алгоритмах:

- отпечатать каталог индивидуальной библиотеки;
- отпечатать текст алгоритма из индивидуальной библиотеки;
- отпечатать список алгоритмов, ссылающихся на данный алгоритм из индивидуальной библиотеки.

Все перечисленные выше операции выполняются с помощью отдельных служебных программ, находящихся у группы, эксплуатирующей транслятор.

I. Подготовить магнитную ленту для архива

Подготовка магнитной ленты для архива производится с помощью соответствующей программы. В паспорте этой программы заказывается лента с заранее заготовленной информацией (лента 6 - 7).

Для архива индивидуальных библиотек берется размеченная лента; программа записывает на нее (зоны I и 2) заготовку каталога алгоритмов индивидуальных библиотек.

2. Завести в архиве индивидуальную библиотеку

Информация для этой операции имеет вид

имя библиотеки ; срок дата

Здесь имя библ. - это идентификатор. Дата указывает срок, до которого данная библиотека должна сохраняться в архиве.

Пример:

волга ; срок 12.10.80
миг ; срок 12.05/79
лоск ; срок 14/11-80

Здесь в архиве создадутся три библиотеки: *волга*

сроком до I2 числа I0 месяца I980 г., *лиг* сроком до I2 числа 5 месяца I979 года и *лоск* сроком до I4 числа II месяца I980 года.

Одновременно можно завести в архиве только одну библиотеку.

3. Поместить алгоритм в индивидуальную библиотеку

Информация имеет вид

имя библиотеки; БИБ имя библиотеки: имя алгоритма,... имя алгоритма; БИБ: имя алгоритма, имя алгоритма; описание процедуры.

Здесь *имя библиотеки, алг.* - идентификаторы.

Перед описанием процедуры находятся имя библиотеки, в которую помещается алгоритм, и имена всех алгоритмов общей и данной индивидуальной библиотеки, к которым обращается данный алгоритм. Если это алгоритмы из общей библиотеки, то после слова *БИБ* имя библиотеки не указывается (следует сразу символ „:").

Внимание:

Алгоритм, помещаемый в индивидуальную библиотеку, не может обращаться к алгоритмам других индивидуальных библиотек.

Пример:

моя; БИБ: simpson, RUNGE;
БИБ моя: пусто;
БИБ: перт;

real procedure A(x, y, z);

begin ... end

В библиотеку с именем *моя* помещается алгоритм *A*.

Этот алгоритм обращается к алгоритмам *simpson, runge, перт* из общей библиотеки и к алгоритму *пусто* из индивидуальной библиотеки *моя*.

Внимание !

В конце процедуры не должно быть символа „;" (после символа end).

4. Выбросить алгоритмы из индивидуальной библиотеки

Информация имеет вид *Имя библиотеки; имя алгоритма, имя алгоритма,... имя алгоритма.*

Из библиотеки с указанным именем выбросятся все алгоритмы, имена которых приведены в списке.

Примечание. Если необходимо заменить текст какого-либо алгоритма, то надо сначала выбросить этот алгоритм, а затем поместить новый текст в библиотеку.

5. Запретить изменения в индивидуальной библиотеке

Информация имеет вид

Имя библиотеки; шифр шифр

За подчеркнутым словом *шифр* следует любой идентификатор, служащий шифром, по которому можно будет "открыть" библиотеку.

Зак.Б-747/742ф.

После выполнения описываемой операции в данной индивидуальной библиотеке нельзя будет производить действия:

а) помещать новые и выбрасывать старые алгоритмы;

в) ликвидировать эту библиотеку;

с) изменить срок существования библиотеки.

Алгоритмами библиотеки, которые "закрыты" описываемой здесь операцией, можно пользоваться. Можно также получать информацию об этой библиотеке с помощью операций, описанных в пунктах 10,11,12.

6. Запретить пользование индивидуальной библиотекой

Информация имеет вид

Имя библиотеки ; шифр шифр

Шифром является любой идентификатор. После выполнения этой операции нельзя будет пользоваться алгоритмами этой библиотеки, производить в ней изменения и получать какую-либо информацию о содержимом библиотеки.

7. Открыть индивидуальную библиотеку

Информация имеет вид

Имя библиотеки ; шифр шифр

Если библиотека была закрыта с помощью операций, описанных в разделах 5 и 6, то данная операция открывает доступ к библиотеке и возможность работы с ней при условии, конечно, что шифр в описываемой операции совпадает с тем, который был использован для ее закрытия.

8. Изменить срок существования индивидуальной библиотеки

Информация имеет вид *Имя библиотеки ;*

срок дата

Время существования указанной библиотеки устанавливается в соответствии с приведенной датой. Способ задания даты такой же, как и в операции "Завести в архиве индивидуальную библиотеку" (см.пункт 2).

9. Ликвидировать индивидуальную библиотеку

Информация имеет вид

Имя библиотеки

В результате выполнения описываемой операции в архиве индивидуальных библиотек ликвидируется библиотека с указанным именем.

Примечание. Индивидуальная библиотека может быть ликвидирована в архиве также в том случае, если истечет срок ее хранения.

10. Отпечатать каталог индивидуальной библиотеки

Информация имеет вид

Имя библиотеки

Если не запрещено пользование данной библиотекой (пункт 6), то печатается ее каталог.

11. Отпечатать алгоритмы из индивидуальной библиотеки

Информация имеет вид *Имя библиотеки ; имя алгоритма, ... имя алгоритма*

Печатаются тексты всех указанных в операции алгоритмов индивидуальной библиотеки (если она не закрыта для пользова-

ния (см. пункт 6).

I2. Указать, кто ссылается на данный алгоритм из индивидуальной библиотеки

Информация имеет вид

Имя библиотеки ; имя алгоритма

Печатаются имена всех алгоритмов, которые непосредственно или через другие алгоритмы ссылаются на данный. Информация, получаемая от этой операции, может облегчить решение вопроса о том, можно ли выбросить из библиотеки какой-либо алгоритм.

I3. Упорядочить архив индивидуальных библиотек

При замене какого-либо алгоритма другим последний помещается в конце всех алгоритмов, а не на месте выброшенного. Поэтому время от времени появляются "пустоты". Для ликвидации этих "пустот" и применяется данная операция. Для этого на магнитофонах устанавливаются две ленты: на магнитофон с большим номером ставится лента с архивом, на магнитофон с меньшим номером - "чистая" размеченная лента, на которой будет располагаться упорядоченный архив индивидуальных библиотек.

IУ. ИНСТРУКЦИЯ ПО СОСТАВЛЕНИЮ КОМПЛЕКТА ПЕРФОКАРТ ДЛЯ ЗАДАЧ, РЕШАЕМЫХ В СИСТЕМЕ "БЭСМ-АЛГОЛ"

Вводимая в машину информация состоит из программы, написанной на входном языке системы, и числового материала, если таковой имеется.

§ I. Общий вид программы

Перед алгол-программой (перед первым символом *begin* может находиться (если надо) *один или* более списков имен алгоритмов из библиотек. Всем именам алгоритмов одного списка предшествует подчеркнутое слово *БИБ* и двоеточие (*БИБ:*), если алгоритмы берутся из общей библиотеки, либо слово *БИБ*, имя библиотеки и двоеточие (*БИБ моя:*), если алгоритмы берутся из какой-либо индивидуальной библиотеки. Списки друг от друга и от алгол-программы отделяются точкой с запятой. Внутри списка имена алгоритмов отделяются друг от друга запятой. Каждое имя алгоритма - это идентификатор соответствующей процедуры, текст которой находится в библиотеке. Появление перед алгол-программой списков имен алгоритмов эквивалентно внесению описаний всех этих алгоритмов в дополнительный блок, заключающий в себе алгол-программу. В списках имен алгоритмов должны содержаться идентификаторы всех библиотечных алгоритмов, явно встречающихся, но не описанных в алгол-программе. Стандартные функции и процедуры не относятся к числу библиотечных, и их идентификаторы помещать в эти списки не следует.

Пример.

БИБ: график , minmax ;

Зак.Б-747/742ф.

БИБ моя: АУ20, ПУ40;

БИБ: симпсон, лагранж;

begin... end

Здесь в алгол-программу будут включены алгоритмы:

график, minmax, симпсон, лагранж

из общей библиотеки и алгоритмы АУ20, ПУ40 из индивидуальной библиотеки с именем МОЯ.

§2. Составление комплекта перфокарт задачи для решения

Программа и числовой материал пробиваются на перфокарты на УПП. Массив перфокарт для программы и числовых данных будем в дальнейшем называть массивом "Алгол-программы и данных."

К массиву "Алгол-программы и данных" добавляются 3 стандартных массива, которые будут описаны в дальнейшем и которые для определенности обозначим через A, B, C .

Если объем массива "Алгол-программы и данных" не превышает размера отводимой для него части ОЗУ (22 листа, т.е. 800 пф), то весь вводимый массив перфокарт составляется из массива A , "Алгол-программы и данных" и C по схеме:

1	массив A
2	массив „Алгол-программа"
3	массив C

Если же объем массива "Алгол-программы и данных" превышает 22 листа ОЗУ, то он делится на подмассивы, между которыми вкладывается массив B . В этом случае вводимый массив перфокарт составляется из массивов A , "Алгол-программы и данных", B и C по схеме:

массив A
массив „Алгол-программа"
массив B
массив „Алгол-программа"
массив B
массив „алгол-программа"
массив C

МАССИВ "АЛГОЛ-ПРОГРАММЫ И ДАННЫХ"

состоит из:

1) массива перфокарт, содержащих программу;
2) перфокарты с символами ◊2 , если к программе прилагается числовой материал, пробитый на УПП, или перфокарты с символами ◊5 , если числовой материал пробит для машины М-20 (М-220);
3) массива перфокарт, содержащих числовой материал.

МАССИВ А состоит из:

I) перфокарт, содержащих паспорт транслятора и стандартный конец паспорта символ *E* ;

2) 4-5 пустых карт;

3) карты стандартного запуска транслятора;

4) 4-5 пустых карт;

5) перфокарт, содержащих паспорт вводимого массива "Алгол-программы и данных" (для 073 экстракода) и стандартный конец с символом *E* ;

6) *4-5* пустых карт;

7) карты с признаками для вывода промежуточной информации при трансляции;

8) карты с вводным словом для массива "Алгол-программы и данных";

9) карты с указующим символом *A3* .

МАССИВ В состоит из:

I) карты с символами ◊ *8* , которые обозначают конец подмассива "Алгол-программы и данных";

2) карты с признаком конца ввода (I в 40- й и 80- й колонках);

3) 4-5 пустых карт;

4) перфокарт, содержащих паспорт массива "Алгол-программы и данных" и стандартного конца с символом *E* ;

5) 4-5 пустых карт;

6) карты с вводным словом для массива "Алгол-программы и данных";

7) карты с указующим символом *A3* .

МАССИВ С состоит из:

I) карты с символами ◊ *9* , которые обозначают конец массива "Алгол-программы и данных";

2) карты с признаком конца ввода (I в 40- й и 80- й колонках).

Паспорт транслятора одновременно является паспортом задачи. Поэтому в нем должны быть указаны необходимые для трансляции и решения задачи ресурсы: а) число листов ОЗУ, б) число трактов МБ, в) количество МЛ с указанием номеров направлений и магнитофонов.

При трансляции используются 24 листа ОЗУ ($0 \div 27_8$), 64 тракта барабана, две ленты (3-е направление, *0-й* магнитофон (информационная) и 6-е направление, *7-й* магнитофон (библиотека). Если в программе есть собственные массивы, то требуется еще одна лента (3-е направление, I-й магнитофон).

Пример паспорта (I и 2 пф):

B	*00*	*000*	*7541*	*вводное слово*
	00	*000*	*0000*	
C	*00*	*00*	*0000*	*число листов ОЗУ(30_8)*
	00	*00*	*0030*	*количество МЛ (2)*
C	*00*	*00*	*0002*	*количество трактов МБ(100_8)*
	00	*00*	*0100*	*номер задачи*
C	*00*	*00*	*0000*	*вход в транслятор (37 ст)*
	02	*10*	*0037*	*номера листов ОЗУ($0^{ой}$,$1^{ый}$,...)*
C	*00*	*01*	*0203*	
	04	*05*	*0607*	
C	*10*	*11*	*1213*	
1пер.	*14*	*15*	*1617*	

```
C   20   21   2223 }
     24   25   2627 }  ——— " ——— " ———
C   00   00   0000 }
     00   00   0000 }
C   67   30   0000 }  номера лент (6-7, 3-0)
     00   00   0000 }
2пф E                  конец паспорта
```

После трансляции указанные в паспорте тракты барабана могут быть использованы для решения задачи.

Стандартное начало трансляции (3-я карта):

```
B   00    000   0036 }  вводное слово
     00    000   0000 }

C   00    14    0000 }  информационное
     00    21    0015 }  слово для 7070

K   00    070   0036 }  считывание блока
     00    000   0000 }  „ввод информации"
```


Паспорт массива для ввода по 073 экстракоду (4 пф):

```
B   00   000   7541
     00   00    0000

     40   00    0000
     00   00    0000

     00   00    0000
     00   00    0005

     00   00    0000 }  шифр задачи (21)
     00   00    0021 }

     00   00    0000
     00   00    0000

4пф  00   00    0000
     00   00    0000
```

5пф E

Вводное слово для массива (6 карта):

```
     B    00   000   4000
6пф       00   000   0000
```

Признак ввода (7 карта)

```
7пф  A3
```

Карта с признаком вывода на печать исходной информации (карта 5,Б"):

В	00	000	0034
	00	000	0000
К	00	000	0000
	00	000	0001

Карта с признаком вывода на печать таблицы идентификаторов и таблицы строк (карта 5,В"):

В	00	000	0033
	00	000	0000
К	00	000	0000
	00	000	0001

Карта с признаком вывода на печать рабочей программы (карта 5,Д"):

В	00	000	0037
	00	000	0000
К	00	000	0000
	00	000	0001

Карта с признаком разделения трансляции и счета (карта 5,Г"):

В	00	000	0024
	00	000	0000
К	00	000	0000
	00	000	0001

§ 3. Перфорация информации

Подчеркнутые слова (основные символы входного языка) пробиваются следующим образом:

а)бьется знак (подчеркивание);

б)бьется слово (основной символ);

в)бьется символ ␣ (пробел).

Знак присваивания (: =)бьется как "двоеточие" и "равно".

Слово (символ) go to можно бить как _go␣_to␣, так и _go to␣.

Каждая строка текста, написанная на бланке, пробивается на отдельной карте и заканчивается символом "конец строки" (I74) .

В середине подчеркнутого слова нельзя бить символ "конец строки".

У. ОТЛАДКА И РЕШЕНИЕ ЗАДАЧИ

§ I. Информация, выдаваемая при трансляции

Транслятор выявляет все синтаксические и многие семантические ошибки. Эти ошибки делятся на три группы:

I) ошибки пробивки (написания) основных символов;

2) синтаксические ошибки;

3) ошибки в описании идентификаторов.

За просмотр транслятор выдает только ошибки одной группы, после чего прекращает трансляцию.

Ошибки пробивки (написания) основных символов

Ошибки этой группы выдаются в первую очередь. В процессе трансляции происходит сравнение пробитых основных символов входного языка с эталоном, а также проверка на четность введенной в память машины информации. При нарушении четности на печать выдается следующая информация:

строка *n*

ошибки в четности

где *n* - номер строки.

Если введен неверный основной символ, представляющий собой подчеркнутое слово, то на печать выдается текст:

строка *n*

в языке нет слова...

многоточие здесь обозначает неверное слово (основной символ).

Если неверный символ не является словом, то тогда печатается следующий текст:

строка *n*

в языке нет символа...

многоточие обозначает неверный символ (основной).

За просмотр выявляются все ошибки данной группы.

Синтаксические ошибки

Если не обнаружено ошибок первой группы, то начинается проверка ошибок второй группы. При обнаружении синтаксической ошибки на печать выдается следующая информация: номер строки, в которой обнаружена ошибка (номер строки после метки или процедуры), печатается часть текста от начала строки до места ошибки и печатается информация о характере ошибки.

За просмотр выявляются все ошибки, если они не влияют на поиск других ошибок этой группы, т.е. если в одном операторе допущено более одной ошибки, то все ошибки могут быть не обнаружены.

Ошибки в описании идентификаторов

Эта группа ошибок проверяется самой последней, и, если ошибки не выдавались, считается, что ошибок, нарушающих конструкции входного языка, нет. При обнаружении ошибок данной группы на печать выдается следующая информация: номер строки, где обнаружена ошибка, и идентификатор (неописанный). Если же нет соответствия между описанием идентификатора и его использованием в программе (описан как массив, а используется как

простая переменная, или наоборот), то на печать будет выдана информация, в которой указывается номер строки и неправильно используемый идентификатор.

Если какой-нибудь идентификатор используется (описан) дважды в одном блоке, то на печать выдается номер строки и идентификатор, который описан вторично. Этот случай может возникнуть в ситуациях, когда одна и та же величина описана как

real	и	real
real	и	integer
real	и	array
real	и	метка
integer	и	integer
integer	и	array
integer	и	метка
array	и	array
array	и	метка
метка	и	метка
boolean	и	boolean
boolean	и	real
boolean	и	integer
boolean	и	array
boolean	и	метка

За просмотр выявляются все ошибки данной группы, если каждая ошибка не влияла на выявление другой.

Дополнительная информация для отладки

При трансляции по желанию программиста на печать могут быть выданы:

- первоначальный текст программы на входном языке. Печать осуществляется по строкам с соответствующей их нумерацией. Используемые процедуры из библиотеки (общей или индивидуальной) выдаются плотно, одной строкой. Для такой выдачи необходимо перед шестой картой поставить карту 5"Б";
- таблица идентификаторов и таблица строк. Таблица идентификаторов является таблицей распределения памяти. Для каждого идентификатора печатается: идентификатор, номер блока, в котором идентификатор описан, и адрес ячейки, которая отводится для величины, обозначенной данным идентификатором. Адрес ячейки состоит из номера индекс-регистра (первые два разряда слева) и номера.
  В таблице строк для участков исходной программы, которые соответствуют "концам строк", приводятся соответствующие им номера блоков и номера команд в рабочей программе. Эта информация выводится по карточке 5"В", которая ставится перед шестой картой. Эта информация выдается для определения места ошибки в записи на входном языке при счете;
- рабочая программа в коде ЭЦВМ БЭСМ-6. Эта информация является вспомогательной и выдается программистом очень редко. Эту информацию можно выдавать только тем исполнителям, которые знают систему команд машины БЭСМ-6, в противном случае эта информация будет малоэффективной. Данная информация выдается по карте 5 "А", которая подкладывается также перед шестой картой.

Зак.Б-747/742ф.

Карты 5"А",5"Б",5"В" могут применяться в любой комбинации друг с другом, при этом каждая будет выдавать только свою информацию. Эти карты подкладываются перед шестой в любой последовательности, т.е. не играет роль здесь, какая карта будет первой, какая последней.

Информация по карте 5"Б" выдается первой. Только после этого проверяется информация на ошибки. Таблица строк и рабочая программа выдаются после трансляции, т.е.перед счетом. Таблица идентификаторов выдается в середине трансляции.

Информация, указывающая на нарушение условий ограничения на входном языке

Если число содержит больше I9 цифр (во входной информации, т.е. алгольной записи), то на печать выдается текст:

число содержит больше I9 цифр

Если идентификатор (любой) содержит более 90 символов, то печатается текст:

идентификатор содержит больше 90 символов

При записи числа в алгольной записи больше допустимого (например, $2.3_{I0}20$) на печать выдается текст:

записано число больше допустимого

Число блоков в "Алгол-программе" не должно превышать 254. При нарушении этого условия на печать будет выдан текст:

число блоков в программе больше 254

Число незакрытых блоков не должно превышать 62. При нарушении условия печатается текст:

число незакрытых блоков в программе больше 62

Нумерация блоков всегда начинается с I. Если программа начинается с метки, то включается дополнительный внешний блок с номером 1 в программу, в который входит метка и вся программа.

Если встречается описание процедуры, то появляются два новых блока, т.к. формальные параметры выделяются в отдельный блок, а тело процедуры оформляется всегда в самостоятельный блок.

Под таблицу идентификаторов в памяти отводится 4000 ячеек. Идентификаторы в таблице сгруппированы по блокам в порядке их закрытия в программе. В одной ячейке расположено 6 символов. В конце каждого идентификатора в отдельной ячейке находится информационное слово. Для каждого *for* в программе заводится два дополнительных идентификатора *for 1* и *for 2*. При выходе за границы (4000 яч.) таблицы идентификаторов печатается текст:

нет места в текущей таблице
идентификаторов
или
нет места в окончательной таблице
идентификаторов

Общее количество целых меток в программе не должно превышать I70.

При невыполнении этого условия печатается текст:

нет места в текущей таблице целых меток
или
нет места в окончательной таблице целых меток

Общая длина таблицы констант и строк 3000 яч.
Если встретилось число в "Алгол-программе, то оно переводится в двоичную систему и заносится в таблицу со своим информационным словом. Таким образом, для каждого числа в таблице отводится 2 ячейки. Константы по блокам не группируются, поэтому одинаковые константы имеют один и тот же номер во всей программе.

Если в качестве фактического параметра процедуры (сюда относится и процедура *output*) используется строка, то все символы строки, включая " ' " и " , ", заносятся в таблицу. Для строки из n символов в таблице отводится *entier (n/6 +3)* ячеек

При выходе за пределы таблицы (3000 яч.) печатается текст:

нет места в таблице строк и констант

Во всех случаях после выдачи текста на печать трансляция задачи прекращается. Для ликвидации ошибки необходимо уменьшить объем той группы объектов, которые вызывают нарушение граничных условий.

Ошибки, выдаваемые транслятором при решении

ошибка 0I (выдается при записи числа в массив или не выдается при выборе его из массива) - адрес элемента одномерного массива выходит за границы массива, т.е. за границы его описания. Например, если имеем описание массива $a[1:10]$, то в операторе $a[i]:= \ldots$; возникает ошибка 0I , если $i<1$ и $i>10$ или напишем операторы вида $a[0]:= \ldots$; $a[12]:= \ldots$;

ошибка 02 (выдается при записи числа в массив) - адрес элемента двумерного массива выходит за границы массива. Например, имеем массив $a[1:5, 1:10]$. Ошибка 02 появится в операторе $a[i,j]:= \ldots$; если $1>i>5$, а также тогда, когда $i=5$, $j>10$, но не выдается

ошибка 02, если $i=5$, а $j<1$, но не меньше (-50), т.е. если j выходит за свои граничные условия, а адрес не выходит за пределы массива, то ошибка 02 не выдается;

ошибка 05 - то же самое, только для массива размерности больше второй, т.е. третьей, четвертой и т.д.;

ошибка 06 - в процедуре элемент массива (любой размерности) вышел за пределы. Эта ошибка может появиться и тогда, когда число фактических параметров больше числа формальных;

ошибка 03
ошибка 04 } -массив не помещается в ОЗУ;

ошибка 30 - нарушение условий п.9 § I в главе I;

ошибка 07 - неверно заданы фактические параметры в обращении к процедуре *MARG* ;

ошибка I0 - неверно задан формат для печати (число не умещается в одной строке). Это возникает тогда, когда мы оператором *MARG* задаем размер строки n позиций, а в операторе *output* задаем размер числа на m позиций, где $m>n$
Например: *MARG* (40,I0,78,5,60,5);
output ('Z6D.5D', b); Здесь число b будет выдаваться на печать, занимая I2 позиций, а оператор *MARG* задал длину строки в странице всего I0 позиций;

ошибка I7 - недопустимы операнды для возведения в степень. Это случается в следующих случаях:

Зак.Б-747/742ф.

a↑i (a=0 и i=0) a - любого типа;
a↑i (a=0 i<0) i - типа integer
a↑r (a=0 r≤0) a - любого типа;
a↑r (a<0) r - типа real

ошибка 22 - неверно обращение к оператору output;
ошибка I2 - нижняя граница массива больше верхней.

Вслед за номером ошибки на печать выдается номер ячейки, где обнаружена ошибка. По номеру этой ячейки и информации, выдаваемой по карте 5 "В", определяется строка (перфокарта), где появилась ошибка.
На печати выдача ошибки выглядит так:

ошибка 0I см.яч.05343
ошибка 03 см.яч.I0345 и т.д.

Обнаружение и устранение "ошибок математика"

При обнаружении ошибки в программе диспетчер выдает на печать информацию вида:
ошибка в программе математика: текст – n m NK M17 MI6 MI5 03200 MI4 MI3 MI2 MII MI0 M7 M6 M5 M4 M3 M2 MI 0000 0000
где текст - характер ошибки (деление: 0, переполнение АУ и т.д.);
n - десятичное число (содержимое сумматора);
m - восьмеричное число (- " -);
NK - номер команды (ячейка), где обнаружена ошибка;
M17, M16, ... M1 - значения индекс-регистров (модификаторов).

По содержимому модификатора MI4 определяется номер строки (по информации, выдаваемой с помощью карты 5 "В") в "Алгол-программе", где возникла ошибка.

Ниже приводится список характерных ошибок и указывается причина их возникновения:

1) ошибка в программе математика: чужой лист – n m NK M17 M16 M15 03200 M14 M13 M12 M11 M10 M9 M8 M7 M6 M5 M4 M3 M1 0000

NK = 00044 - нет в конце программы карты с символами ↓9 Если же она есть, то не вводится в машину. В первом случае необходимо эту карту вставить, во втором - заменить на другую;
NK = 00120 - под задачу не хватает памяти (24 листа);
N17 ≥ 60000 - необходимо уменьшить длину массивов или программы или использовать внешнюю память машины (МБ, МЛ) при помощи операторов read , WRITE ;
NK = 00577 - выбираемый на МОЗУ адрес элемента массива
M16 ≥ 60000 вышел за пределы 24 листов.

Например:

```
b[5] := b[27000]
a[i] := b[j];   j > 24000
c[2] := a[i]×3 + b[i]
i > 24000
```

2) переполнение АУ (ошибка в пр.математика):
4000≤M14≤M1 - перемножаются два больших числа, дающих в результате число 10^{I9}, складываются, вычитаются или делятся два числа, которые тоже в результате дают число 10^{I9} или любые другие действия, в результате которых получаются числа 10^{I9}. Такая ситуация

может возникнуть и в операторе *output*, если выдается на печать не полностью заполненный массив, который в начале не был прописан нулями;

- ошибка в операторе *output*. Это случается тогда, когда после текстовой выдачи не указан характер следующей выдачи. Например:

```
output ('T', 'x=', a, b);
```

Здесь после выдачи текста *x=* не ясно, что делать с идентификаторами *a*, *b*. Здесь пропущена строка, определяющая вид печати. Надо писать так:

```
output ('T', 'x=', 'Z', a, b);
```

или вместо 'Z' может стоять 'Y', 'E' и т.д. ;

3) ошибка в программе математика: контр.ком / запр.ком } *n m NK M17 M16*

M15 03200 M14 M13 M12 M11 M10 M9 M8 M7 M6 M5 M4 M3 M2 M1 0000

NK > M1 - осуществляется передача управления на величину, а не на метку. Это может быть в следующих случаях:

а)
```
begin  real L ;
...
go to L ;
end
```


б)
```
begin  real L ; ... integer i ;
...
begin  real a ;
for  i := 1  step 1 until 10 do begin
...
go to L ; ... end ;
end L : end
```

Здесь после *end* (обведенный пунктиром) не стоит символ " ; ", поэтому введенная метка *L :* воспринимается как комментарий. Следовательно, передача управления *go to L* будет осуществляться на переменную *L* ;

в) это может произойти и при использовании переключателя, когда его индекс выходит за пределы описания. Например:

```
begin  switch L := M1, M2, M ; ... go to L[5];
... end
```

Здесь индекс 5 вышел за пределы описания переключателя (т.е. индекс не может быть больше 3).

В этом случае передача управления неопределена, поэтому может возникнуть описываемая ошибка.

4000 ≤ NK ≤ M1 - неисправность в машине ;

NK < 4000 - см.п.3 описываемой ошибки;

NK = 00037 - между второй и третьей картами паспорта (запуска) транслятора меньше трех пустых карт, либо третья карта неверна или отсутствует. Подложить чистые карты или заменить третью карту.

Зак.Б-747/742ф.

ошибка в программе математика: деление на ноль п т NK М17 М16 М15 03200 М14 М13 М12 М11 М10 М9 М8 М7 М6 М5 М4 М3 М2 М1 0000 0000

Это ошибка возникает как и при обычном делении, так и при вычислении тангенса угла ($tg\,\alpha$). Поэтому, если в указанной строке деления нет, надо искать формулу, где вычисляется величина $tg(x)$. Это происходит при вычислении величин $tg(90°)$ и $tg(270°)$.

Ошибки в числовых данных

При ошибках в числовых данных на печать выдаются два вида информации об этих ошибках.

Группа данных >длины массива оператор input n объект ввода m

Эта информация говорит о том, что мы пытаемся ввести в массив (простую переменную) больше чисел, чем это требует его описание. Это может произойти тогда, когда в конце группы нет символа " ; " или 777 7777 7777 7777 . Если они есть, то надо проверить, правильно ли они набиты. Если все верно, то необходимо проверить числовые данные; может быть где-нибудь пробито лишнее число или лишний символ " , ".

Место ошибочного ввода определяется по величинам

m , n.

n - десятичный номер оператора *input* . При последовательных обращениях к операторам *input* в разных частях программы величина n при

каждом обращении к оператору увеличивается на единицу. Если к какому-либо оператору *input* в цикле обращаемся несколько раз, то все равно величина n при каждом обращении к одному и тому же оператору будет увеличиваться на единицу;

m - номер переменной, стоящей в операторе, т.е. номер фактического параметра. Например:

input (a, b, c, d, e);

При $m=2$ это будет переменная b , при $m=5$ - переменная e .

Неверная запись элемента ввода l оператор input n объект ввода m.

Это случается тогда, когда неверно пробито или неверно вводится l-е число, в n-м операторе *input* и в m-й переменной.

Под неверной записью подразумевается:

1) нарушение четности при пробивке l-го числа;
2) число (l-е) записано не в алгольной записи;
3) нет $\Diamond 2$ или $\Diamond 5$ перед числовым материалом;
4) после $\Diamond 2$ стоят числа, пробитые для М-220;
5) после $\Diamond 5$ стоят числа, пробитые для БЭСМ-6;
6) на перфокартах пробит не числовой, а текстовой материал;
7) число записано больше допустимого (10^{19}).

Величина l указывает номер числа в массиве, которое не может быть введено в машину. Величины n и m имеют те же значения, что описаны в первом виде ошибки.

Величины l, n, m выдаются в десятичном виде.

§ 2. Отделение счета от трансляции

В трансляторе предусмотрен режим отделения счета от трансляции. После трансляции (если перед шестой картой паспорта (запуска) стояла карта 5 „Г") рабочая программа пишется на информационную магнитную ленту. После этого составляется новый паспорт, в котором указывается то количество листов ОЗУ, которые необходимы для решения задачи. Пример паспорта:

B	00	0000	7541
	00	0000	0000
C	00	000	000
	00	000	$0n_1n_2$
C	00	000	$0ml$
	00	000	100
C	00	000	$0N_1N_2$
	N_3N_4	$N_5$22	000
C	$\alpha\alpha$	$\alpha_1\alpha_1\alpha_2$	$\alpha_2\alpha_3\alpha_3$
	$\alpha_1\alpha_4$	$\alpha_5\alpha_5\alpha_6$	$\alpha_6\alpha_7\alpha_7$
.....			
.....			
C	$\alpha_{30}\alpha_{30}$	$\alpha_{31}\alpha_{31}\alpha_{32}\alpha_{32}$	$\alpha_{33}\alpha_{33}$
	$\alpha_{34}\alpha_{34}$	$\alpha_{35}\alpha_{35}\alpha_{36}$	$\alpha_{36}\alpha_{37}\alpha_{37}$
C	MM	$M_1M_1M_2$	$M_2M_3M_3$
	M_4M	$M_4M_6M_6$	$M_6M_7M_7$

$n_1\ n_2$ – количество листов ОЗУ, необходимое для решения задачи (задается в восьмеричном виде);

l – количество магнитных лент, используемых в задаче (задается в восьмеричном виде);

m – количество магнитных лент с заранее заготовленной информацией из общего количества l (можно и не указывать);

$N_1\ N_2\ N_3\ N_4\ N_5$ – шифр задачи;

$\alpha\alpha\ \alpha_1\alpha_1\ \ldots\ \alpha_{n-1}\alpha_{n-1}$ – математические номера листов ОЗУ (задаются в порядке возрастания).

Например: $n_1n_2 = 10$, тогда $\alpha\alpha = 00$, $\alpha_1\alpha_1 = 01$, $\alpha_{22} = 02$; $\alpha_3\alpha_3 = 03$, $\alpha_4\alpha_4 = 04$ $\alpha_5\alpha_5 = 05$, $\alpha_6\alpha_6 = 06$, $\alpha_7\alpha_7 = 07$

$MM\ M_1M_1\ \ldots\ M_{e-1}M_{e-1}$ – математические номера магнитофонов. Первая буква обозначает номер направления, вторая – номер магнитофона. Например: $MM = 67$, $M_1M_1 = 33$ и т.д.

ПРИМЕЧАНИЕ. Магнитная лента с математическим номером $3-0$ должна быть обязательно указана в паспорте.

Затем в кодах машины БЭСМ-6 пробивается следующая программа

B	00	000	2000
	00	000	0000
K	00	070	2004
	00	000	0000
K	00	010	2005
	00	075	0055

```
К     00     010     0002
      00     015     2004
К     00     000     0002
      00     300     0051
С     00     100     000
      00     300     000
К     02     350     0052
      00     000     0000
Е
```

Эта программа осуществляет вызов с магнитной ленты в ОЗУ рабочей программы и передачу управления на начало счета.

Количество листов ОЗУ, необходимое для задачи, определяется содержимым I7-го индекс-регистра (MI7).

Если имеются массивы с переменными границами, то весь объем ОЗУ, занимаемый задачей, определяется следующим образом:

I) вводятся максимальные границы в описании массивов;

2) вводится искусственная ошибка (например, деление на ноль) в том месте программы, где максимально используется память машины;

3) по выданному на печать содержимому I7-го индекс-регистра (MI7) определяется весь объем ОЗУ, который необходим для решения задачи.

Содержимое первого индекс-регистра (MI) определяет длину программы.

Для быстроты перевода содержимых I7-го и I-го индекс-регистров (они указывают объем памяти в ячейках) в листы ОЗУ ниже приводится таблица перевода ячеек в листы ОЗУ.

Номера ячеек	Номера листов (восьмер.)	Номера ячеек	Номера листов (восьмер.)
00000 ÷ 0I777	0	40000 ÷ 4I777	20
02000 ÷ 03777	I	42000 ÷ 43777	2I
04000 ÷ 05777	2	44000 ÷ 45777	22
06000 ÷ 07777	3	46000 ÷ 47777	23
I0000=÷ II777	4	50000 ÷ 5I777	24
I2000 ÷ I3777	5	52000 ÷ 53777	25
I4000÷ I5777	6	54000 ÷ 55777	26
I6000 ÷ I7777	7	56000 ÷ 57777	27
20000 ÷ 2I777	I0	60000 ÷ 6I777	30
22000 ÷ 23777	II	62000 ÷ 63777	3I
24000 ÷ 25777	I2	64000 ÷ 65777	32
26000 ÷ 27777	I3	66000 ÷ 67777	33
30000 ÷ 3I777	I4	70000 ÷ 7I777	34
32000 ÷ 33777	I5	72000 ÷ 73777	35
34000 ÷ 35777	I6	74000 ÷ 75777	36
36000 ÷ 37777	I7	76000 ÷ 77777	37

Литература

А.Н.БАЛУЕВ, Р.И.БЕЛИХ, В.А.ДАУГАВЕТ, Н.А.ШИДЛОВСКАЯ. Сборник упражнений по АЛГОЛ-60.

МАК-КРАКЕН. Программирование на АЛГОЛ-60.

В.М.КУРОЧКИН, Д.Б.ПОДШИВАЛОВ, А.И.СРАГОВИЧ, Г.И.СЕДАНКИНА, Н.Н.СТРЕЛКОВА, А.Я.ФАЛЕТОВА. Система "БЭСМ-АЛГОЛ" (методическое руководство по программированию).

ЛАВРОВ С.С. Универсальный язык программирования.

СОДЕРЖАНИЕ

Стр.

I. ВХОДНОЙ ЯЗЫК 3

§ I. Ограничения 3

§ 2. Арифметические выражения..... 5

§ 3. Стандартные функции 7

§ 4. Булевские (логические) выражения..... 8

§ 5. Оператор присваивания 10

§ 6. Метки и операторы перехода..... 12

§ 7. Условные операторы..... 12

§ 8. Оператор цикла 15

§ 9. Переключатели 17

§ 10. Блоки 20

§ 11. Комментарий..... 26

§ 12. Процедуры 27

1. Операторы процедур 28

2. Процедуры-функции 34

3. Использование символа VALUE..... 36

4. Использование комментария 39

II. СТАНДАРТНЫЕ ПРОЦЕДУРЫ 40

§ I. Стандартная процедура INPUT..... 40

1. Группа числовых данных 40

2. Группа логических данных 41

3. Группа текстовых данных 42

§ 2. Стандартная процедура OUTPUT..... 43

1. Группа вывода числовых значений..... 45

2. Группа вывода логических значений 49

3. Группа вывода текста..... 51

4. Группа размещения..... 54

§ 3. Стандартная процедура MARG 57

§ 4. Стандартные процедуры READ и WRITE 58

III. ОРГАНИЗАЦИЯ АРХИВА ИНДИВИДУАЛЬНЫХ БИБЛИОТЕК..... 62

§ I. Операции с архивом..... 62

1. Подготовить магнитную ленту для архива..... 63

2. Завести в архиве индивидуальную библиотеку..... 63

3. Поместить алгоритм в индивидуальную библиотеку..... 64

4. Выбросить алгоритм из индивидуальной библиотеки.. 65

5. Запретить изменения в индивидуальной библиотеке... 65

6. Запретить пользование индивидуальной библиотекой.. 66

7. Открыть индивидуальную библиотеку..... 66

8. Изменить срок существования индивидуальной библиотеки..... 66

9. Ликвидировать индивидуальную библиотеку..... 67

10. Отпечатать каталог индивидуальной библиотеки..... 67

11. Отпечатать алгоритмы из индивидуальной библиотеки..... 67

12. Указать, кто ссылается на данный алгоритм из индивидуальной библиотеки 68

13. Упорядочить архив индивидуальных библиотек..... 68

IV. ИНСТРУКЦИЯ ПО СОСТАВЛЕНИЮ КОМПЛЕКТА ПЕРФОКАРТ ДЛЯ ЗАДАЧ, РЕШАЕМЫХ В СИСТЕМЕ "БЭСМ-АЛГОЛ"..... 69

§ I. Общий вид программы..... 69

§ 2. Составление комплекта перфокарт задачи для решения... 70

§ 3. Перфорация информации..... 77

V. ОТЛАДКА И РЕШЕНИЕ ЗАДАЧИ..... 78

§ I. Информация, выдаваемая при трансляции..... 78

Ошибки пробивки (написания) основных символов 78

Синтаксические ошибки..... 79
Ошибки в написании идентификаторов..... 79
Дополнительная информация для отладки..... 80
Информация, указывающая на нарушение условий
ограничения во входном языке.....82
Ошибки, выдаваемые транслятором при решении..... 84
Обнаружение и устранение "ошибок математика"..... 86
Ошибки в числовых данных..... 90
§ 2. Отделение счета от трансляции..... 92

Технический редактор Фролова Т.А.

Подписано в печать 8.07.70г.
Формат 60x90/16. Печ.л. 6,25. Уч.-изд.л. 5,9.
Г-863517. Зак.Б-747/742ф.