14. Управляющие конструкции

Любой язык программирования не может обойтись без управляющих структур – ветвления и циклов. В язык Maple V также входят такие операторы, причем их синтаксис очень похож на синтаксис управляющих операторов в традиционных языках программирования высокого уровня. Кроме этого, в Maple V отсутствует оператор безусловного перехода и, следовательно, программы Maple должны поддерживать истинный принцип модульного программирования.

14.1 Оператор ветвления

Оператор ветвления изобразим следующей синтаксической диаграммой:

if - then
elif - then
…
elif - then
else
fi

Или в развернутом виде:

if <условие> then <последовательность операторов>
| elif <условие> then <последовательность операторов> |
| else <последовательность операторов> |
fi

Условие - булево выражение.

Выражение elif расшифровывается как else - if, т.е. “в противном случае проверить следующее условие”.

Строк elif может быть неограниченное количество.

fi - признак конца оператора ветвления.

Пусть имеем следующие равенства:

```
> cond:=true: x:=2: y:=3:
```

Тогда:

```
> if cond then r:=2 else r:=3 fi;
```

$$r := 2$$

```
> if not cond then r:=0 elif y>x then r:=1 elif
  x>y then r:=2 else r:=3 fi;
```

$$r := 1$$

Оператор ветвления допустимо задавать и в форме функции, при этом if должно быть заключено в обратные кавычки.

```
> `if`(false,`истина`,`ложь`);
```

ложь

```
> `if`(true,`истина`,`ложь`);
```

истина

14.2 Оператор цикла

Можно задать циклы двух типов: “for – to” и “while”.

Синтаксис описания оператора цикла следующий:

|for <имя>| |from <выражение>| |by <выражение>|
|to <выражение>| |while <выражение>|
do <последовательность операторов> od;

или

|for <имя>| |in <выражение>| |while <выражение>|
do <последовательность операторов> od;

По умолчанию значения выражений from и by равны единице.

od – признак конца тела цикла.

Конструкция in предназначена для работы с перечисляемыми типами данных (массивы, списки и т.д.), в этом случае цикл будет выполняться для каждого элемента массива или списка.

Выражение в to и в in вычисляется перед началом цикла.

```
> a:=array(1..2);
```

$$a := \text{array}(1 .. 2, [\ ])$$

```
> for i to 2 do a[i]:=i od;
```

$$a_1 := 1$$

$$a_2 := 2$$

```
> i:=0:
```

```
> while i<>2 do i:=i+1: a[i]:=1: od;
```

$$i := 1$$

$$a_1 := 1$$

$$i := 2$$

$$a_2 := 1$$

```
> restart: s:=0:
> for z in [1,2,3] do s:=s+z od:
> s;
```

6

В следующем примере находится сумма всех четных чисел от 12 до 100:

```
> sum1 := 0:
> for i from 12 by 2 while i < 100 do
>    sum1 := sum1 + i
> od:
> sum1;
```

2420

Можно организовать вложенные циклы:

```
> b:=array(1..2,1..2);
```

$$b := \text{array}(1 .. 2, 1 .. 2, [\ ])$$

```
> for i to 2  do for j to 2  do
  b[i,j]:=sqrt(i+j) od od;
> print(b);
```

$$\begin{bmatrix} \sqrt{2} & \sqrt{3} \\ \sqrt{3} & 2 \end{bmatrix}$$

Оператор next

Если в цикле необходимо пропустить одну итерацию, то для этой цели можно использовать оператор next. Действие этого оператора заключается в том, что на текущей итерации не будет выполнено никаких вычислений, а начнется выполнение следующей итерации; next используется только в циклах.

```
> ar:=array(1..4):
> for i to 4 do if i=3 then next else ar[i]:=0
```

```
  fi od;
> print(ar);
```

$$[0, 0, ar_3, 0]$$

Оператор break

Для прерывания выполнения цикла используется оператор break; break используется только в циклах.

```
> ar:=array(1..4):
> for i to 4 do if i=3 then break else ar[i]:=i
  fi od;
> print(ar);
```

$$[1, 2, ar_3, ar_4]$$