Программирование в Maple

Основные конструкции языка. Процедуры

Условный оператор в Maple

Условный оператор

```
> restart :
  if 2 = 3 then print( 2 = 3 ); else print( 2 ≠ 3 ); fi;
  if 2 = 3 then print( 2 = 3 ); else print( 2 ≠ 3 ); end if;
```

$$2 \neq 3$$

$$2 \neq 3$$

Условный оператор в Maple

Полный вид условного оператора

```
> x := 1;
  if x ≠ 2 then print(x = 2);
  elif x = 3 then print(x = 3);
  elif x = 4 then print(x = 4);
  else print(x);
  fi;
```

$$x := 1$$

$$1 = 2$$

Операторы цикла. For

```
> for i from 1 by 2 to 9 do
      print(i) ;
  od;
  #end do;
```

```
1
3
5
7
9
```

Операторы цикла. For

Если шаг равен 1, то можно использовать сокращенный оператор

```
> for i from 1 to 3 do
    print(i);
  od;
```

1

2

3

Операторы цикла. For

Пример. Сумма всех двузначных нечетных чисел

```
> s := 0 :
  for i from 11 by 2 to 99 do s := s + i end do:
  s;
```

2475

Операторы цикла. For x in

```
> for x in [10, 20, 30] do
      x;
  od;
```

```
10
20
30
```

Операторы цикла. While

Оператор while

```
> x := 2;
  while x = 2 do
      x := x - 1;
  od;
```

$x := 2$

$x := 1$

Операторы цикла. While

Симбиоз for и while

```
> x := -1 :
  step := 3 :
  for i from 1 by step while x < 3 do
      x := x + step :
  od;
```

$x := 2$

$x := 5$

Примеры

Пример. Сумма элементов списка

```
> s := 0 :
  for z in [ 1, x, y, q^2, 3, t, t^2 ] do s := s + z end do:
  s;
```

$$q^2 + t^2 + t + y + 34$$

Пример. Произведение элементов последовательности

```
> p := 1 :
  for z in [ 1, 2, 3, 4, 5, 6, 7 ] do p := p· z end do:
  p;
```

$$5040$$

Примеры

Пример. Сумма элементов списка

```
> s := 0 :
  for z in [ 1, x, y, q^2, 3, t, t^2 ] do s := s + z end do:
  s;
```

$$q^2 + t^2 + t + y + 34$$

Пример. Произведение элементов последовательности

```
> p := 1 :
  for z in [ 1, 2, 3, 4, 5, 6, 7 ] do p := p· z end do:
  p;
```

$$5040$$

Примеры

```
> restart :
```

$$f := x^2 + x + \frac{1}{x} :$$

```
for s in f do
    s;
od;
```

$$x^2$$

$$x$$

$$\frac{1}{x}$$

Итеративные команды Maple

seq

add и mul

select, remove, selectremove

map

zip

Итеративные команды Maple

```
> $2..5;
```

$$2, 3, 4, 5$$

```
> a[i]$i=1..5;
```

$$a_1, a_2, a_3, a_4, a_5$$

```
> seq(x^2, x=1..4);
```

$$1, 4, 9, 16$$

```
> seq(i, i="a".."e");
```

"a", "b", "c", "d", "e"

```
> seq(x[i], i=1..5);
```

$$x_1, x_2, x_3, x_4, x_5$$

```
> seq(x, x in [Pi, sin(2·Pi), cos(3·Pi)]);
```

$$\pi, 0, -1$$

Итеративные команды Maple

add, mul

> $add(i, i = 1..5);$

$$15$$

> $add(\sin(i), i = 1.0..3);$

$$1.891888420$$

> $mul(i, i = 1..5);$

$$120$$

> $mul(\sin(i), i = 1.0..3.0);$

$$0.1079776075$$

> $mul\left(x, x \text{ in } \left[\text{Pi}, \sin\left(\frac{2}{3}\cdot\text{Pi}\right), \cos(3\cdot\text{Pi})\right]\right);$

$$-\frac{\pi\sqrt{3}}{2}$$

Итеративные команды Maple

add, mul

> $add(i, i = 1..5);$

$$15$$

> $add(\sin(i), i = 1.0..3);$

$$1.891888420$$

> $mul(i, i = 1..5);$

$$120$$

> $mul(\sin(i), i = 1.0..3.0);$

$$0.1079776075$$

> $mul\left(x, x \text{ in } \left[\text{Pi}, \sin\left(\frac{2}{3}\cdot\text{Pi}\right), \cos(3\cdot\text{Pi})\right]\right);$

$$-\frac{\pi\sqrt{3}}{2}$$

Итеративные команды Maple

Команды извлечения и удаления (аналог цикла for/in)
select(proc,expression) - извлекает те операнды из выражения expression, которые удовлетворяют булевой функции (процедуре) proc.
remove(proc,expression) - удаляет те операнды из выражения expression, которые не удовлетворяют булевой функции (процедуре) proc.
selectremove(proc,expression) - сначала извлекает те операнды из выражения expression, которые удовлетворяют булевой функции (процедуре) proc, а затем те, которые ей не удовлетворяют.
Каждая команда возвращает объект того же типа, что и исходное выражение

Итеративные команды Maple

```
> ints := [$ 10 ..20];
                ints := [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

> select( isprime, ints );
                                [11, 13, 17, 19]

> remove( isprime, ints );
                             [10, 12, 14, 15, 16, 18, 20]

> q1, q2 := selectremove( isprime, ints );
          q1, q2 := [11, 13, 17, 19], [10, 12, 14, 15, 16, 18, 20]

> q1;
                                [11, 13, 17, 19]

> q2;
                             [10, 12, 14, 15, 16, 18, 20]
```

Итеративные команды Maple

```
> f := x→x > 15;
```

$$f := x \mapsto 15 < x$$

```
> select( f, ints );
```

$$[16, 17, 18, 19, 20]$$

Итеративные команды Maple

map

```
> map(p, [a, b, c]);
> map(f, {a, b, c});
> map(sqrt, ints);
```

$$\left[\sqrt{10}, \sqrt{11}, 2\sqrt{3}, \sqrt{13}, \sqrt{14}, \sqrt{15}, 4, \sqrt{17}, 3\sqrt{2}, \sqrt{19}, 2\sqrt{5}\right]$$

```
> g := (T)→sin(x + T);
```

$$g := T \mapsto \sin(x + T)$$

```
> map(x→x^2, ints);
```

$$[100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400]$$

```
> map(g, ints);
```

$$[\sin(x+10), \sin(x+11), \sin(x+12), \sin(x+13), \sin(x+14), \sin(x+15), \sin(x+16), \sin(x+17), \sin(x+18), \sin(x+19), \sin(x+20)]$$

Итеративные команды Maple

zip

```
> zip(`+`, [1, 2, 3], [a, b, c]);
```

$$[1 + a, 2 + b, 3 + c]$$

```
> zip((x, y)→10·x + y, [1, 2], [3, 4, 5]);
```

$$[13, 24]$$

help("statements")

List of Maple Statements

Description

- See ?topic for any of the following topics:

!	#	assignment	break
by	catch	description	do
done	elif	else	empty
end do	end if	error	export
finally	for	from	function
if	in	local	module
next	option	proc	quit
read	restart	return	save
separator	stop	then	to
try	use	uses	while

ПРОЦЕДУРЫ

ПРОЦЕДУРЫ

```
> hello := proc( ) return "Hello World"; end proc;
> hello( );
# Синтаксис:
# Имя:=proc(параметр1::type1,параметр2::type2,...)
        local l1,l2...;
        global g1,g2...;
        options op1, op2, ...;
        description  string;
          тело процедуры;
end;
```

ПРОЦЕДУРЫ

 # Пример 1. Очень простая процедура

```
> sum2 := proc(a, b);
    a^2 - b^2;
    a^2 + b^2;
  end:
```

```
> sum2(2, 3);
```

13

```
> sum2(0.1, 0.2);
```

0.05

ПРОЦЕДУРЫ

 # Пример 1. Продолжение. Описан тип формальных параметров

```
> sum_2 := proc(a :: integer, b :: float);
    a^2 + b^2;
  end:
```

```
> sum_2(1, 2.15);
```

5.6225

```
> sum_2(2.1, 1); # несоответствие типов
Error, invalid input: sum_2 expects its 1st argument
to be of type integer, but received 2.1
```

ПРОЦЕДУРЫ

```
# Описание. Description

> restart :
> power := proc(a :: integer, b :: integer)
  description "возведение в степень";
  a^b;
  end:

> power(2, 4);
```

16

ПРОЦЕДУРЫ

```
# Локальные переменные

> sumList := proc(a :: list, b :: integer) :: integer;
    local i, s;
    s := 0;
    for i in a do
      s := s + a[i];
    end do;
    s := s * b;
  end proc:

> rez := sumList([1, 2, 3, 4, 5], 5);
                                    rez := 75
```

ПРОЦЕДУРЫ

```
# Глобальные переменные

> constPi := 3.15 :
  Length := proc(r :: float)
    global constPi;
    constPi := 3.1415 :
    2·constPi·r;
   end proc:

> Length(1.);
   constPi;
```

6.2830

3.1415

ПРОЦЕДУРЫ

```
# Локальные переменные. Еще раз

> constPi := 3.15 :
  Length := proc(r :: float)
    local constPi;
    constPi := 3.1415 :
    2·constPi·r;
  end proc:

> Length(1.);
  constPi;
```

6.2830

3.15

ПРОЦЕДУРЫ

```
# Сообщение об ошибках. Использование функции error

#  Вычисление квадратного корня из числа
> sqroot := proc(x)
    if not (type(x, numeric) or type(x, realcons))
     or signum(x) = -1
     then error "CHECK THE INPUT DATA  x : %1", x
     else evalf(x^(1/2));
    end if
   end proc:


> sqroot(2);
                                1.414213562

> sqroot(-2); sqroot('asdf');
Error, (in sqroot)  CHECK THE INPUT DATA  x: -2
Error, (in sqroot)  CHECK THE INPUT DATA  x: asdf
```

ПРОЦЕДУРЫ

Пример процедуры с использованием uses

```
> LastWord := proc(s :: string)
    uses StringTools :
    Split(s) :
    %[-1];
    end proc:
> LastWord("Hello world");
                                    "world"
```

ПРОЦЕДУРЫ

Пример процедуры. Возврат нескольких значений из процедуры

```
> LastWords2 := proc(s :: string)
  uses StringTools :
  local L :
  L := Split(s) :
  L[-1], L[-2];
  end proc:
> LastWords2("We say: Hello world");
                                "world", "Hello"
```

ПРОЦЕДУРЫ

Вывод кода процедуры на экран print, eval

```
> print(LastWords2);
proc(s::string)
    local L;
    L := StringTools:-Split(s); L[-1], L[-2]
end proc
```

```
> eval(LastWords2);
proc(s::string)
    local L;
    L := StringTools:-Split(s); L[-1], L[-2]
end proc
```

ПРОЦЕДУРЫ

Вывод кода процедуры из библиотеки Maple на экран
(кроме встроенных процедур, с опцией builtin)

```
> interface(verboseproc = 2) :
> print(issqr);
proc(n)
    option
    Copyright (c) 1990 by the University of Waterloo. All rights reserved.;
    if type(n, integer) then
        evalb(isqrt(n)^2 = n)
    elif type(n, numeric) then
        false
    else
        'issqr(n)'
    end if
end proc
```

ПРОЦЕДУРЫ

_passed – последовательность всех аргументов, переданных процедуре при ее вызове (устаревший вариант: args), имеет тип exprseq
_npassed – число всех аргументов, переданных процедуре при ее вызове (устаревший вариант: nargs)

```
> maximum := proc ( )
  local max, i;
    max := _passed[ 1 ];
    for i from 2 to _npassed do
    if _passed[ i ] > max then
          max := _passed[ i ]
      end if
    end do;
    max;
  end proc:

> maximum( 1, 100, -4 ) ;
                                    100

> maximum( 1, 100, -4, 1000, -3 ) ;
                                    1000
```

ПРОЦЕДУРЫ

_nrest – число недекларированных аргументов, переданных процедуре
_rest – число «лишних» аргументов, переданных процедуре при ее вызове

```
> sum2 := proc(a, b)
  local x, s :
  s := 0 :
  for x in _rest do s := s + x : end do:
  return a + b, s, a + b + s;
  end proc:

> sum2(1, 2, 3, 4, 5);
```

3, 12, 15

ПРОЦЕДУРЫ

Опция **remember**
позволяет кодировать функцию с рекурсивным определением наиболее естественным образом, бе
потери эффективности.

```
>
```

#требуется экспоненциальное время для вычисления

```
> F1 := proc(n) if n < 2 then n else F1(n-1) + F1(n-2) end if end proc;
```

$$F1 := \mathbf{proc}(n)\ \mathbf{if}\ n < 2\ \mathbf{then}\ n\ \mathbf{else}\ F1(n-1) + F1(n-2)\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

```
>
```

#требуется линейное время для вычисления

```
> F2 := proc(n) option remember;
    if n < 2 then n else F2(n-1) + F2(n-2) end if end proc;
```

$$F2 := \mathbf{proc}(n)\ \mathbf{option}\ remember;\ \mathbf{if}\ n < 2\ \mathbf{then}\ n\ \mathbf{else}\ F2(n-1) + F2(n-2)\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

```
> time(F1(25));
```

$$0.171$$

```
> time(F2(25));
```

$$0.$$

ПРОЦЕДУРЫ

Оператор return

При выполнении команды return все оставшиеся команды в теле процедуры игнорируются.

```
> f := proc(a, b)
    return a;
    return a + b;
    end proc:

> f(2, 100);
```

2

Спасибо за внимание!