2.3 Типы данных

Большое разнообразие типов данных переводит Maple V из разряда прикладной программы для решения математических задач в класс систем математического программирования. Около ста зарезервированных имен типов данных может встретить пользователь на необъятных просторах Maple V. Существует большое разнообразие функций для работы с данными, в структуре последних можно просто запутаться. Чтобы хоть как-то пролить свет на типы данных, рассмотрим основные, с которыми мы встречаемся при выполнении различных вычислений.

Целые

В Maple V выражение принадлежит к целому типу (тип integer), если оно состоит из последовательности цифр, не разделенных между собой никакими знаками. Длина последовательности ограничена лишь ресурсами системы, но, обычно, больше чем пользователь может себе представить – более 500000 цифр. Число типа integer может быть как положительным, так и отрицательным.

С целыми числами возможны следующие операции:

abs – модуль числа;

factorial или n! – нахождение факториала.

А также многие другие.

```
> abs(-10420);
```

$$10420$$

```
> factorial(5); 5!;
```

$$120$$

$$120$$

Для проверки принадлежности выражения к определенному типу служит команда *type*.

Формат команды:

type(x, t),

где x - любое выражение, t - название типа.

Например:

```
> type(-102,integer);
```

true

Дробные

Тип fraction – дробный тип. Дроби представляются в виде: a/b, где a – целое число со знаком, b – целое число без знака. В выражении типа fraction обязательно присутствие двух полей: числитель и знаменатель.

Функция op от дроби возвращает два числа – числитель и знаменатель.

```
> op(2/7);
```

2, 7

Числа с плавающей точкой

Тип float – числа с плавающей точкой. Тип float в среде Maple определен как:

1. последовательность чисел, разделенных точкой:
   а) <integer>.<integer>
   б) <integer>.
   в) .<integer>
2. число может быть представлено в виде:
   Float(mantissa, exponent)
   т.е. <mantissa>*10^<exponent>

```
> type(.1234,float);
```

true

```
> Float(2,4);
```

20000.

Обратное представление числа реализуется функцией op, которая возвращает два числа – мантиссу и экспоненту.

```
> op(0.02234);
```

2234, -5

Для приближения чисел с плавающей точкой служит команда evalf:

```
> evalf(Pi,5);
```

3.1416

Строковые типы

В среде Maple определены тип string и тип name. Выражение типа *string*

может содержать цифры и буквы, строчные и прописные. Строка с двух сторон должна быть окружена символом ' . Если же в строку требуется вставить символ ', то его надо удвоить.

```
> var:=`It``s a string`;
```

$$var := It`s\ a\ string$$

Из строки можно выделить подстроку:

```
> substring(abcdefgh,3..5);
```

$$cde$$

Определим длину строки:

```
> length(abcdef);
```

$$6$$

Булевы выражения

Для логических операций в среде Maple предусмотрен специальный тип данных – *`boolean`*, а зарезервированные слова *true* и *false* используются для работы с булевыми выражениями.

В булевских выражениях можно использовать следующие операторы:
`=`, `<>`, `<`, `<=`, `and`, `or`, `not`.

Для работы с булевыми выражениями предусмотрена команда *evalb* – вычисление сложного логического выражения:

```
> evalb(f=f);
```

$$true$$

```
> ToBe or not ToBe;
```

$$true$$

Последовательности

Последовательность – это набор элементов, разделенных запятыми, без скобок.

```
> S:=1,2,3,4,5,6;
```

$$S := 1, 2, 3, 4, 5, 6$$

Для генерации последовательности в среде Maple служит команда *seq*. Синтаксис вызова данной команды:

seq(y, i = m..n)

seq(y, i = x),

где **y** – любое выражение; **i** – имя; **m, n** – численные параметры; **x** – выражение.

Наиболее распространенный вызов: seq(f(i), i = 1..n) , который генерирует

последовательность f(1), f(2), ..., f(n).

Менее употребимый – seq(f(i), i = m..n) – создает последовательность f(m), f(m+1), f(m+2), ..., f(n). Здесь m и n могут быть не только типа integer.

```
> seq(sin(Pi*i/6), i = 0..3);
```

$$0, \frac{1}{2}, \frac{1}{2}\sqrt{3}, 1$$

Множества

Множества принято обозначать фигурными скобками. Для них присущи все правила преобразования, принятые в классической математике.

```
> set1:={sin,cos,tan,cos};
```

$$set1 := \{\cos, \sin, \tan\}$$

Извлечение элементов с первого по второй из множества set1:

```
> op(1..2,set1);
```

$$\cos, \sin$$

Определение количества элементов в множестве set1:

```
> nops(set1);
```

$$3$$

Объединение двух множеств {a,b} и {b,c}:

```
> {a,b} union {b,c};
```

$$\{a, c, b\}$$

Пересечение:

```
> {a,b} intersect {b,c};
```

$$\{b\}$$

Вычитание:

```
> {a,b} minus {b,c};
```

$$\{a\}$$

Принадлежность элементов 'a' и 'cos' множеству set1:

```
> member(a,set1);
```

false

```
> member(cos,set1);
```

true

Зададим множество L в виде последовательности:

```
> L := {seq(y[i],i=1..4)};
```

$$L := \{y_1, y_2, y_3, y_4\}$$

Добавление элемента к множеству L:

```
> L := {op(L),z[5]};
```

$$L := \{y_1, y_2, y_3, y_4, z_5\}$$

Удаление второго элемента из множества L:

```
> L := subsop(2=NULL,L);
```

$$L := \{y_1, y_3, y_4, z_5\}$$

Списки

Список принято обозначать квадратными скобками.

```
> list1:=[sin,cos,tan,cos];
```

$$list1 := [\sin, \cos, \tan, \cos]$$

Со списками можно производить математические операции, например, дифференцирование:

```
> D(list1);
```

$$[\cos, -\sin, 1 + \tan^2, -\sin]$$

По отношению к спискам и множествам допустимы операции присваивания :

```
> list2:=list1;
```

$$list2 := [\sin, \cos, \tan, \cos]$$

Массивы

Массив – конечномерный список с целочисленными индексами. Операции, применяющиеся к массивам:

array – создание массива;

print – распечатка содержимого массива;

map – задание операции над всеми элементами массива;
op – извлечение элементов (уточнение задания массива).

Создадим массив v:

```
> v := array(1..4);
```

$$v := \mathrm{array}(1 .. 4, [\ ])$$

Заполним этот массив элементами и распечатаем его содержимое:

```
> for i to 4 do v[i]:=i od:
> print(v);
```

$$[1, 2, 3, 4]$$

Создадим одномерный массив ‘s’ с нулевыми значениями:

```
> s:=array(1..2,[0,0]);
```

$$s := [0, 0]$$

Создадим двумерный массив ‘m’:

```
> m:=
array(symmetric,1..2,1..2,[[cos(y),0],[0,sin(y)]]);
```

$$m := \begin{bmatrix} \cos(y) & 0 \\ 0 & \sin(y) \end{bmatrix}$$

Зададим операцию дифференцирования над всеми элементами массива ‘m’:

```
> map(diff,m,y);
```

$$\begin{bmatrix} -\sin(y) & 0 \\ 0 & \cos(y) \end{bmatrix}$$

Таблицы

В отличие от массивов, где индексы - целочисленные значения, расположенные по порядку номеров, индексы у таблиц - любые значения.

Таблица задается указанием слова "table":

```
> table();
```

table([

])

Команда "table( )" создала таблицу с неопределенными значениями.

Определим значения таблицы, выполнив команду:

```
> table([22,42]);
```

table([

$1 = 22$

$2 = 42$

])

Так как индексы таблицы не были определены, то программа сама присвоила им целочисленные значения, расположенные по порядку.

Индексы таблицы можно задать произвольным образом:

```
> S := table([(red)=45,(green)=61]);
```

S := table([

$green = 61$

$red = 45$

])

Обратимся к элементам таблицы:

```
> S[1],S[red];
```

$$S_1, 45$$

С таблицами возможны также следующие операции.

Зададим таблицу F, элементами которой являются операторы:

```
> F:=table([y=(x->x^2),cos=-sin]):
```

Распечатаем таблицу:

```
> print(F);
```

table([

$y = (x \to x^2)$

$\cos = -\sin$

])

Вычислим значение элемента таблицы 'y' от аргумента, равного 3:

```
> F[y](3);
```