16. Ввод/вывод информации

Любая достаточно сложная программная система должна уметь обрабатывать данные, полученные из внешних источников, а также выводить собственные результаты в форме, понятной для других программ. Maple V release 4 обладает такими возможностями. Так, Maple может осуществлять не только чтение и запись файлов, но и выводить уже готовый код для языков программирования высокого уровня, а также выражения, записанные в терминах текстового процессора LaTeX, который является стандартным средством создания документов среди математиков и физиков всего мира.

16.1 Работа с файлами

В новой версии расширен набор функций для работы с файлами и сняты некоторые ограничения. Если в Maple V release 3 можно было открыть всего навсего один файл, то теперь можно одновременно открыть до семи файлов и работать с ними в различных режимах. Поддерживаются несколько типов файлов, причем их количество зависит от операционной системы, под управлением которой работает Maple.

Типы файлов

ТИП ФАЙЛА	ОПИСАНИЕ
STREAM	буферизованный файл (подобно файлам в C); может быть текстовым (TEXT) или бинарным (BINARY)
RAW	небуферизованный бинарный файл (поддерживается UNIX и некоторыми другими платформами)
PIPE	файл–мост (поддерживается только в UNIX)
PROCESS	файл–мост, через который идет обмен информацией с другим процессом (поддерживается только в UNIX)
DIRECT	прямой доступ к текущему (default) или верхнему (terminal) уровню ввода/вывода (т.е. когда Maple работает в интерактивном режиме); DIRECT имеет всегда текстовый тип.

Функции ввода/вывода

ФУНКИЯ	ОПИСАНИЕ
appendto (file)	исполняемые команды и результаты выполнения дозаписываются в существующий файл
close (files)	закрывает файлы, открытые open
fclose (files)	закрывает файлы, открытые fopen
feof (file)	проверка достижения конца файла
fflush (files)	запись буфера и закрытие буферизированных файлов
filepos (file, pos)	установка или проверка указателя файла
fopen (name, mode, type)	открытие файла для буферизованного чтения, записи или дозаписи
fprintf (file, fmt, x1, ..., xn)	вывод в файл выражений x1,..,xn с использованием строки формата fmt
fremove (files)	удаление файлов
fscanf (file, fmt)	чтение из файла и представление в формате fmt
iostatus	выводит информацию о работе с файлами
lprint (expr1, expr2, ...)	печать выражений Maple как обычный текст
open (file, mode)	открывает файл для небуферизованного чтения или записи
printf (fmt, x1, ...,xn)	печать выражений x1,..,xn по формату fmt
read file	чтение из файла выражений Maple
readbytes (file, len, TEXT)	чтение байт из файла и представление в форме списка или текста
readdata (fileID, format, n)	чтение числовой информации из файла. Числа записаны в n колонках (колонки разделены пробелом)
readlib (name, files)	чтение выражения из библиотеки
readline (file)	чтение строки из файла
readstat(..)	чтение оператора из потока данных
save names, file	запись в файл выражений names
savelib (name, files)	запись выражений в библиотеку
sscanf (str, fmt)	представление строки str в формате fmt
unload (procname)	выгружает ранее загруженную с помощью readlib процедуру
writebytes (file, bytes)	запись байт в файл
writedata (fileID, data, format)	запись числовых значений из перечисляемых типов данных в текстовый файл
writeline (file, str ...)	запись строк в файл
writestat (file, expr ...)	запись выражений в файл
writeto (file)	исполняемые команды и результаты выполнения записываются в новый файл

Режимы доступа

Файл может быть открыт в режиме чтения (READ) или в режиме записи (WRITE). Если файл типа STREAM, RAW или DIRECT открыт в режиме READ и предпринята попытка записи, то файл будет автоматически закрыт и открыт вновь в режиме WRITE, причем указатель файла останется в той же позиции. В обратном случае Maple также проделает аналогичную операцию.

Другие типы файлов не позволяют изменять режим доступа после своего открытия.

Бинарный и текстовый типы файлов

Файлы UNIX не имеют типа, они представляются как поток байт. Другие операционные системы различают текстовые (поток символов) и бинарные (поток байт) файлы. Различие обычно заключается в символе начала новой строки. В DOS и Windows символ новой строки представляется в форме двух символов - "\r\n". В Macintosh символ новой строки представляется символом возврата каретки - "\r". В UNIX символ новой строки представляется символом перевода строки "\n".

В Maple V release 4 содержится несколько избыточное количество функций ввода/вывода, которое позволяет достичь необходимой гибкости в работе с данными. Большинство из функций ввода/вывода приведены в таблице. Исключение составили лишь функции, которые не работают в ОС Windows.

Приведем примеры использования функций ввода/вывода в наиболее часто встречающихся случаях.

Протоколирование сеанса работы в файл

```
> restart;
```

Зададим имя файла в который будет записываться протокол работы.

```
> fn:=`outfile.txt`:
```

Включим режим протоколирования и выполним несколько команд.

```
> writeto(fn);
> Int(x^2,x);
> x:=a+b;
```

Выключим режим протоколирования.

```
> writeto(terminal);
```

Установим указатель на начало файла.

```
> filepos(fn,0);
```

$$0$$

Прочитаем содержимое файла.

```
> while not feof(fn) do readline (fn); od;
```

```
> Int(x^2,x);
                          /
                         |   2
                         |  x  dx
                         |
                         /
> x:=a+b;
                      x := a + b
> writeto(terminal);
```

Закроем файл.

```
> close(fn);
```

Запись и чтение переменных

```
> restart;
```

Определим значения нескольких переменных.

```
> a:=Int(x,x);
```

$$a := \int x \; dx$$

```
> s:=a*r+i;
```

$$s := \int x \; dx \; r + i$$

Запишем значения всех ранее определенных переменных в файл out.

```
> save `out`;
```

Очистим память Maple – системы.

```
> restart;
```

Прочитаем файл out, содержащий значения переменных.

```
> read `out`;
```

$$a := \int x \; dx$$

$$s := \int x \; dx \; r + i$$

Запись и чтение произвольных файлов

```
> restart;
```

Запишем в файл File4test произвольную строку (имя File4test поддерживается только в Windows 95).

```
> fprintf(File4test,`Проверка записи\n`);
```

16

В качестве результата команда вернула количество записанных байт.

Поставим указатель файла на первую позицию.

```
> filepos(File4test,0);
```

0

Проверим, что записалось в файл.

```
> readline(File4test);
```

Проверка записи

Удалим файл File4Test.

```
> fremove(File4test);
```

Проверим, удалился ли файл.

```
> readline(File4test);
```

Error, (in readline) file or directory does not exist

Ввод большого массива данных из файла

Чтобы ввести в память Maple большое количество числовой информации удобно воспользоваться командой readdata. Эта команда считывает из файла, содержащего числа в колонках, определенное коли-

чество колонок чисел. Колонки в файле отделяются пробелами.

В следущем примере считаем с помощью readdata информацию из файла, содержащего следующие цифры:

```
45  45   26  458 65 9897 65
454 5656 454 656 5  656  2
```

```
> restart;
> fn:=`data.num`;
```

$$fn := data.num$$

```
> readdata(fn,7);
```

[[45., 45., 26., 458., 65., 9897., 65.],
[454., 5656., 454., 656., 5., 656., 2.]]

```
> readdata(fn,3);
```

[[45., 45., 26.], [454., 5656., 454.]]

Получение информации об открытых файлах

```
> fopen(MyTestFile,WRITE);
> open(MyTestFile2,WRITE);

> iostatus();
```

[2, 0, 7, [0, *MyTestFile* , *STREAM* , *FP* = 6118408, *WRITE* , *TEXT*],
[1, *MyTestFile2* , *RAW* , *FD* = 6, *WRITE* , *BINARY*]]

```
> fclose(MyTestFile);
> close(MyTestFile2);
```

iostatus возвращает список, состоящий по меньшей мере из трех элементов, которые несут инфрмацию, указанную в следующей таблице:

№	ОПИСАНИЕ ЭЛЕМЕНТОВ СПИСКА
1	число открытых файлов
2	число активных в настоящее время вложенных “read” команд
3	наибольшее значение суммы iostatus( )[1] + iostatus( )[2]

Для каждого открытого файла появляется дополнительный элемент в списке возвращаемых значений, который также является списком.

Расшифровка значений списка приведена ниже.

№	ОПИСАНИЕ ЭЛЕМЕНТОВ СПИСКА
1	дескриптор файла (выданный fopen, open, pipe, popen)
2	имя файла
3	тип файла (STREAM, RAW, PIPE, PROCESS или DIRECT)
4	FP=адрес для STREAM или PROCESS, FD=число в противном случае (внутреннее представление файла)
5	режим файла (READ или WRITE)
6	тип файла (TEXT или BINARY)

Печать данных в заданном формате

Формат задается с помощью символов, записанных после знака %. Описание символов формата приведено в таблице.

```
> sscanf(`X=123.4 Y=-27.9 Z=2.3E-5`,`X=%f Y=%f
%n Z=%s`);
```

[123.4, -27.9, 16, *2.3E-5*]

```
> sscanf(`123.456E7 123.456E7`,`%g%d.%d %[Ee]
%d`);
```

$[.123456\ 10^{10},\ 123,\ 456,\ E,\ 7]$

Символы строки форматирования

СИМВОЛ ФОРМАТА	ОПИСАНИЕ
d или D	целое число с/без знака
o или O	восьмеричное беззнаковое число; значение преобразуется в десятичное
x или X	шестнадчатиричное беззнаковое число; значение преобразуется в десятичное
e, f, или g	число с плавающей запятой
s	строка
a	невычисляемое выражение
m	выражение во внутреннем m-формате
M	математическое выражение
c	символ
[...]	список определяет символы, которые при встрече во входных данных будут выданы как строка символов (^ – исключить из списка)
n	количество просмотренных символов

16.2 Вывод данных в формате C, FORTRAN, LaTeX

Для удобства использования полученных в Maple результатов в других средах предусмотрена возможность представления данных в формате языков C, FORTRAN и текстового редактора LaTeX. Чтобы получить выражения в этих форматах используются одноименные команды.

Например, LaTeX(exp, `filename`), где exp - список выражений, а `filename`-имя файла, в который записывается результат.

```
> latex(Int(x/(x^2+7),x)=int(x/(x^2+7),x),
rez.ltx);
```

В файле с именем rez.ltx, после выполнения этой команды, будет находиться следующая информация:

```
\int \!{\frac {x}{{x}^{2}+7}}{dx}=1/2\,\ln ({x}^{2}+7)
```

Команда latex может переводить следующие функции Maple:

@	@@	D	Diff	Int	Limit	Log
Sum	abs	binomial	diff	exp	factorial	
int	limit	ln	log10	log2	sum	

В командах C и FORTRAN также можно записать результаты в файл. Кроме того, в них существуют возможности оптимизации выражений (вводятся вспомогательные переменные), а также задания точности вычислений. Например, определим матрицу A:

```
> A:=array(1..2,1..2,symmetric):
  A[1,1]:=log(2*x):A[1,2]:=1-3*log(x):print(A);
```

Переведем ее в FORTRAN-код.

```
> fortran(A,optimized,mode=generic);
```

```
      t3 = 1-3*log(x)
      A(1,1) = log(2*x)
      A(1,2) = t3
      A(2,1) = t3
      A(2,2) = undefined
```

Из этого примера видно, что при оптимизации была введена переменная t3. Глобальные имена t0, t1, t2, ... зарезервированы специально для использования в этих целях.

А теперь другой пример:

```
> readlib(C):
> C([s=x^3,t=ln(s),r=2*Pi*t-sqrt(6)*s^3],
```

```
precision=double);
```

$$s = x*x*x;$$

$$t = \log(s);$$

$$r = 2.0*0.3141592653589793E1*t-\mathrm{sqrt}(6.0)*s*s*s;$$

Для определения точности используется параметр precision=single или precision=double, что означает вычисление констант с плавающей запятой с одинарной или двойной точностью.

По умолчанию для C-кода определена двойная точность, для FORTRAN-кода - одинарная точность.

Следует отметить, что команда C должна быть подгружена с помощью функции readlib(C).

Язык FORTRAN имеет ограниченное количество строк (19), которые кончатся по ошибке в течение трансляции, если превысят его. Для больших выражений, которые превышают это ограничение, подпрограмма FORTRAN автоматически разобъет выражение. Глобальные имена s0, s1, s2, ... зарезервированы специально для этой цели.

Функции C и fortran позволяют теперь перевести в код более сложные конструкции, такие как процедуры, циклы и операторы ветвления.

```
> restart:
> f := proc(x::numeric) local i, M; global test;
>     M := array(-2..3, sparse, [(1)=-cos(x),
>       (2)=x^2]);
>     for i from -2 to 3 do
>        if test then
>          print(M[i])
>        else
>          ERROR(`Ошибка`);
>        fi;
>     od;
>     M;
> end:
> readlib(C): C(f);
```

```
void f(x,crea_par)
double x;
double crea_par[6];
{
  int i;
```

```
  double M[6];

  extern int test;

  M[0] = 0;
  M[1] = 0;
  M[2] = 0;
  M[3] = -cos(x);
  M[4] = x*x;
  M[5] = 0;
  for(i = -2;(i <= 3);i++)
    if (test)
      printf( "%e\n" ,M[i+2]);
   else
     {
       fprintf(stderr, "Ошибка" );
       exit(1);
     }
}
```