РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ
ПРОБЛЕМ БЕЗОПАСНОГО
РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ

RUSSIAN ACADEMY OF
SCIENCES

NUCLEAR
SAFETY INSTITUTE

Препринт № IBRAE–97–07

Preprint IBRAE–97–07

В.В. Варенков, В.А. Первичко, А.Г. Пошков, В.В. Чуданов

DBV — подсистема для работы с базой данных по переменным

Москва
1997

Moscow
1997

УДК 519.682

Варенков В.В., Первичко В.А., Попков А.Г., Чуданов В.В. DBV — ПОДСИСТЕМА ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ ПО ПЕРЕМЕННЫМ. Препринт № IBRAE–97–07. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1997. 30 с.

Аннотация

В препринте описывается подсистема работы с данными в вычислительном коде RASPLAV. Входные и выходные данные хранятся в файлах специального формата. Текстовый файл параметров задает входные и выходные потоки данных для вычислительного модуля. В работе приводится описание формата файла параметров и библиотеки по работе с данными на языке ФОРТРАН. На примере демонстрируются основные приемы работы с подсистемой.

Varenkov V.V., Pervichko V.A., Popkov A.G., Chudanov V.V. DBV — SUBSYSTEM FOR WORKING WITH DATA BASE FOR VARIABLES. (in Russian). Preprint IBRAE–97–07. Moscow: Nuclear Safety Institute, April 1997. xx p.

Abstract

This paper describes subsystem for working with data in computational code RASPLAV. Input and output data are stored in files with special format. Text file of parameters defines input and output data flows for calculation module. The paper describes format of file of parameters and library for work with data on FORTRAN language. Example demonstrates the main ways of work with the subsystem.

DBV —
подсистема для работы с базой данных по переменным

В.В.Варенков, В.А.Первичко, А.Г.Попков, В.В.Чуданов

ИНСТИТУТ ПРОБЛЕМ БЕЗОПАСНОГО РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ
113191, Москва, ул. Б. Тульская, 52
тел.. (095) 955-28-88, факс: (095) 230-20-29, эл. почта: pkv@ibrae.ac.ru

Содержание

1 Введение ... 3
2 Файл параметров PAR ... 5
2.1 Синтаксис ... 5
2.2 Раздел [FILES] ... 6
2.3 Раздел [MODEL VARIABLES] ... 6
2.4 Раздел FILE VARIABLES ... 7
2.5 Атрибуты переменных ... 7
2.6 Ввод/вывод модельных переменных ... 8
3 Описание библиотеки DBV ... 9
3.1 Состав библиотеки ... 10
3.2 Файл MEMORY.INC ... 11
3.3 Файл DBV.INC ... 11
3.4 Основные подпрограммы ... 14
3.5 Внутренние подпрограммы ... 19
3.6 Сообщения об ошибках ... 22
4 Создание файла параметров из системы RCS ... 23
5 Пример использования DBV ... 24
Литература ... 30

1 Введение

При проведении сложных вычислительных расчётов большое практическое значение имеет организация работы с данными. Как правило, большие вычислительные коды имеют специальные подсистемы для ввода/вывода данных. В них обеспечивается проверка правильности входных данных и производятся необходимые преобразования. Кроме поддержки входных и выходных потоков вычислительные коды могут содержать также средства для работы с данными внутри вычислительного кода (например, поддержка доступа к данным по именам или универсальные процедуры преобразования в различные системы единиц). Подавляющее большинство распространённых вычислительных кодов в области математического моделирования тяжёлых аварий реализовано на языке ФОРТРАН.

В данной работе описывается подсистема DBV (Data Base for Variables), предназначенная для поддержки работы с данными в вычислительном коде RASPLAV[1]. Код RASPLAV предназначен для моделирования тяжёлых аварий на АЭС. Для этого кода реализована специальная система RCS (RASPLAV Command System) [2], содержащая программные средства организации вычислений и работы с данными. Подсистема DBV является составной частью программного комплеса RCS.

Подсистема DBV выполняет следующие функции:

- ведение внутренней базы переменных в оперативной памяти;
- поддержка обмена данными между внутренней базой переменных и SDF-файлами.

Место подсистемы DBV в архитектуре RCS показано на рисунке 1. Этот рисунок не отражает структуру всего комплекса RCS. Здесь делается акцент на потоках данных между вычислительными модулями и подсистемами для подготовки входных и обработки выходных данных.

Прежде чем пояснить рисунок 1 сделаем важное замечание о том, что в системе RCS данные рассматриваются как *переменные* - объекты, имеющие символьные имена и атрибуты, отображающие характеристики данных. Обращение к переменным производится по именам. Атрибуты дают возможность контролировать соответствие типов и размерностей данных, а также строить универсальные процедуры ввода/вывода, автоматически производящие необходимые преобразования. Для хранения переменных в файлах используется специальный формат данных - SDF-формат [3].

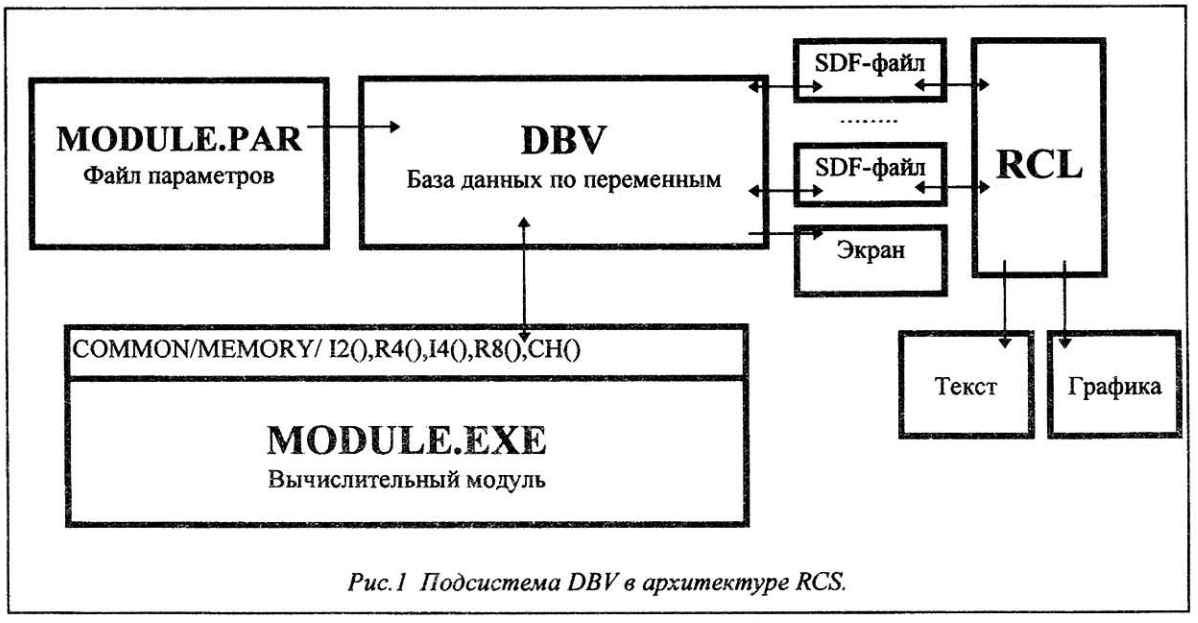

Рис.1 Подсистема DBV в архитектуре RCS.

В центральная части рисунка 1 находится блок DBV. Вычислительный модуль содержит COMMON-блок MEMORY, являющийся местом хранения переменных, с которыми работает DBV Размеры COMMON-блока MEMORY задаются для каждого типа данных в соответствии с потребностями вычислительного модуля.

Вычислительный модуль получает информацию о переменных и входных/выходных файлах из текстового файла параметров (PAR-файла), используя в качестве посредника подсистему DBV. При этом DBV анализирует синтаксическую правильность входного файла и выполняет необходимые преобразования данных.

В процессе работы вычислительный модуль при помощи подсистемы DBV может добавлять новые переменные во внутреннюю базу данных, а также использовать переменные для вычислений и преобразований данных.

Файл параметров создаётся и редактируется любым текстовым редактором (при этом должен соблюдаться описанный в главе 1 формат). Файл параметров может быть создан также при помощи системы RCS. При этом появляются дополнительные возможности инициализации и преобразования данных средствами командного языка RCS.

Подсистема DBV осуществляет чтение и запись переменных в SDF-файлах, а также вывод на экран. При этом автоматически производятся необходимые преобразования. Вычислительный модуль осуществляет ввод-вывод, обращаясь к соответствующим подпрограммам подсистемы DBV.

Входные данные и полученные результаты, хранимые в SDF-файлах, обрабатываются и визуализируются при помощи командного языка системы RCS - RCL (RASPLAV command language). При этом возможно представлять данные как в текстовом, так и в графическом видах.

Данная работа имеет следующую структуру.

Глава 1 описывает синтаксис входного файла параметров и обьясняет как производятся автоматические преобразования данных при вводе и выводе переменных.

Глава 2 описывает библиотеку DBV. Перечисляются компоненты библиотеки. Приводится полный список основных и вспомогательных подпрограмм библиотеки.
Глава 3 посвящена созданию файла параметров средствами системы RCS.

В главе 4 приводится пример использования подсистемы DBV. Глава содержит текст демонстрационного вычислительного модуля на языке ФОРТРАН.

2 Файл параметров PAR

В данной глава описывается синтаксис входного файла параметров.

Содержимое файла параметров определяет не только сами переменные и их принадлежность к конкретным SDF-файлам: дополнительно задаются автоматические преобразования данных при вводе/выводе. Для файловых операций назначаются конкретные записи SDF-файлов. Поддерживается использование рестарт-файлов и вывод на экран.

2.1 Синтаксис

PAR-файл описывает переменные модели, файлы данных и переменные из этих файлов данных, а также управляет вводом и выводом данных.

На рисунке 2 в качестве примера приведён текст простого PAR-файла, демонстрирующий структуру файла, способы описания переменных и рабочих файлов.

[FILES]

% IDENT	**R/W**	**NREC**	**MODREC**	**FILENAME**
INIT	R	1	1	INIT.SDF
STEP	W	1	100	STEP.SDF
1D	W	1	1	1D.SDF

[MODEL VARIABLES]

%VARNAME	**N1**	**N2**	**N3**	**DIM**	**TYPE**	**KEY**	**SCALE**	**SHIFT**	**TUNITS**	**INIT**	**STEP**	**1D**
N	*	*	*	*	*	0	1	0	N_UNITS	*	NN	-
M	1	1	1	1	I2	0	1	0	M_UNITS	*	-	-
K	*	*	*	*	*	*	*	*	*	*	-	-
Time	1	1	1	1	R4	0	1	0	Time_UNITS	Time	*	-
Te	*	*	*	*	R8	*	1	0	Te_UNITS	Te	*	-
% --- WORK												
Ent	N	N2(Te)	1	2	R8	2	2.5	0	-			
Work	15	15	1	2	R8	2	2.5	0	-			
WKK	K	K	1	2	(Ent)	K	(Ent)	(Ent)	-			
WNK	N	K	1	2	(Time)	N	(Time)	(Time)	-			

% * FILE VARIABLES *****

[FILE=STEP]

%VARNAME	**N1**	**N2**	**N3**	**DIM**	**TYPE**	**KEY**	**SCALE**	**SHIFT**	**TUNITS**
Te	~0.1	~0.1	*	*	R4	5	*	*	*

Рис. 2. Пример PAR-файла.

PAR-файл состоит из трех частей:

[FILES]
Описание SDF-файлов, с которыми работает вычислительный модуль.

[MODEL VARIABLES]
Описание модельных переменных.

FILE VARIABLES
Описание файловых переменных.

Содержит последовательность разделов **[FILE=IDENT]** , где IDENT - идентификатор файла из раздела [FILES].

Разделы PAR-файла отделяются друг от друга произвольным количеством пустых строк. Символ **%** в первой позиции строки означает, что данная строка является комментарием и она игнорируется.

Далее рассмотрим подробно каждый из разделов этого PAR-файла.

2.2 Раздел [FILES]

Раздел описывает SDF-файлы, с которыми работает вычислительный модуль. Каждая строка содержит описание отдельного файла и включает в себя следующие поля:

IDENT	Идентификатор файла (CHARACTER*8); позволяет внутри программы ссылаться не на конкретный файл, а на его короткий псевдоним. Возможность именно в PAR-файле указывать конкретное имя файла даёт гибкость в настройке файлов расчётных данных на конкретные директории.
R/W	Тип файла: R(чтение), W(запись), RW(чтение/запись), RS(рестарт), SC(экран).
NREC	Начальный номер записи.
MODREC	Модуль для номера записи; если номер записи для файловой переменной достигает этого значения, то следующая запись - первая.
FILENAME	Полное имя файла с указанием директорий.

Для каждой файловой переменной ведётся свой номер записи, который автоматически увеличивается при каждой операции чтения/записи.

Ниже приведён фрагмент PAR-файла, описывающего файлы.

[FILES]

% IDENT	**R/W**	**NREC**	**MODREC**	**FILENAME**
INIT	R	1	1	INIT.SDF
STEP	W	1	100	STEP.SDF
1D	W	1	1	1D.SDF

Рис.3. Пример раздела [FILES] PAR-файла.

2.3 Раздел [MODEL VARIABLES]

Раздел описывает модельные переменные. Каждая строка содержит описание отдельной переменной и включает в себя следующие поля:

VARNAME	Имя переменной.
N1	Первый размер переменной.
N2	Второй размер переменной.
N3	Третий размер переменной.
DIM	Размерность переменной.
TYPE	Тип переменной.
KEY	Ключ переменной.
SCALE	Коэффициент масштабирования значений переменной.
SHIFT	Коэффициент сдвига значений переменной.

TUNITS	Единицы измерения переменной.
ФАЙЛЫ	Колонки для каждого файла из раздела [FILES] для задания связи модельных и файловых переменных.

Ниже приведён фрагмент PAR-файла, описывающего модельные переменные.

[MODEL VARIABLES]

%VARNAME	N1	N2	N3	DIM	TYPE	KEY	SCALE	SHIFT	TUNITS	INIT	STEP	1D
N	*	*	*	*	*	0	1	0	N_UNITS	*	NN	-
M	1	1	1	1	I2	0	1	0	M_UNITS	*	-	-
K	*	*	*	*	*	*	*	*	*	*	-	-
Time	1	1	1	1	R4	0	1	0	Time_UNITS	Time	*	-
Te	*	*	*	*	R8	*	1	0	Te_UNITS	Te	*	-
% --- WORK												
Ent	N	N2(Te)	1	2	R8	2	2.5	0	-			
Work	15	15	1	2	R8	2	2.5	0	-			
WKK	K	K	1	2	(Ent)	K	(Ent)	(Ent)	-			
WNK	N	K	1	2	(Time)	N	(Time)	(Time)	-			

Рис.4. Пример раздела [MODEL VARIABLES] PAR-файла.

2.4 Раздел FILE VARIABLES

Содержит последовательность разделов **[FILE=IDENT]** , где IDENT - идентификатор файла из раздела [FILES].

Каждый такой раздел описывает переменные для указанного файла.

Поля файловой переменной такие же, как и в разделе модельных переменных, за исключением последних полей для связи с файлами данных.

Если в разделе файловых переменных указан READ-файл, то описания его переменных должны совпадать с реальными атрибутами в SDF-файле.

Ниже приведён фрагмент PAR-файла, описывающего файловые переменные.

% * FILE VARIABLES *****

[FILE=STEP]

%VARNAME	N1	N2	N3	DIM	TYPE	KEY	SCALE	SHIFT	TUNITS
Te	~0.1	~0.1	*	*	R4	5	*	*	*

Рис.5. Пример раздела FILE VARIABLES PAR-файла.

2.5 Атрибуты переменных

Модельные и файловые переменные описываются в файле параметров следующим набором атрибутов:

Атрибут	Содержимое
VARNAME	имя переменной (модельная - до 20, файловая - до 8 символов)
N1	размер по x-координате
N2	размер по y-координате
N3	размер по z-координате
DIM	размерность (1,2,3 , 0 - отдельное число)

TYPE	тип значений: I2,I4,R4,R8,CH
KEY	целочисленное значение ключа
SCALE	Коэффициент масштабирования значений переменной.
SHIFT	Коэффициент сдвига значений переменной.
TUNITS	текстовое представление единиц измерения (до 15 символов)

Имя переменной

Имя модельной переменной должно быть уникальным среди всех модельных переменных, имя файловой переменной должно быть уникальным среди всех переменных данного файла.

Для каждой модельной переменной в файле параметров задаются связи с SDF-файлами.

Формы задания атрибутов переменных:

Форма	**Значение**
*	для модельной переменной - взять атрибут соответствующей считываемой файловой переменной; для записываемой файловой переменной - атрибут соответствующей модельной переменной
<целое число>	значение для N1,N2,N3,DIM,KEY
<вещественное число>	значение для SHIFT, SCALE
I2|I4|R4|R8|CH	значение для TYPE
<текст без пробелов>	значение для TUNITS
<идентификатор>	имя ранее определенной числовой переменной с одним элементом, значение которой присваивается данному атрибуту
(<идентификатор>)	имя ранее определенной переменной, соответствующий атрибут которой присваивается атрибуту данной переменной

Для размеров N1,N2,N3 имеются дополнительные формы задания:

Форма	**Значение**
N1<идентификатор>	значение атрибута N1 ранее определенной переменной
N2<идентификатор>	значение атрибута N2 ранее определенной переменной
N3<идентификатор>	значение атрибута N3 ранее определенной переменной
~<вещественное число>	коэффициент разрежения (>1) или сжатия (<1)

Таким образом, атрибуты новых переменных могут определяться на основе ранее определенных, в том числе и заранее неизвестных атрибутов входных файловых переменных. Это свойство позволяет создавать файлы параметров, не требующие частого редактирования (например, при изменении размеров расчетной сетки).

Требование ссылаться только на ранее определенные переменные позволяет исключить перекрестные ссылки, которые потенциально могут быть противоречивыми.

2.6 Ввод/вывод модельных переменных

На пересечении строки с описанием модельной переменной и столбца с идентификатором SDF-файла могут помещаться следующие элементы:

— модельная переменная с этим файлом никак не связана

* модельной переменной соответствует переменная с таким же именем в данном файле

<идентификатор> модельной переменной соответствует переменная с указанным именем в данном файле

Для ввода могут быть использованы только файлы, помеченные R («чтение») или RW («чтение/запись»).

Для вывода могут быть использованы только файлы, помеченные W («запись») или RW («чтение/запись»).

DBV содержит два типа функций для ввода значений модельных переменных из входных файлов: всех переменных сразу и для отдельного файла. Для вывода задается конкретный файл.

Переменные, которые нужно ввести либо вывести из конкретного файла, определяются в соответствующей колонке при описании модельной переменной. Отсутствие связи переменной с файлом обозначена знаком '—'

При вводе всех входных данных сразу для каждой модельной переменной просматривается список входных файлов, из которых можно читать и выбирается последний файл, для которого определена связь.

При операциях ввода/вывода автоматически учитывается различие в атрибутах переменных и осуществляются необходимые преобразования, а именно:

- преобразование типов данных
- разрежение/сжатие
- перевод в другие единицы измерения на основе коэффициентов

3 Описание библиотеки DBV

Данная глава содержит полное описание библиотеки DBV

Ниже перечислены действия, которые могут осуществлять вычислительные модули при помощи соответствующих подпрограмм библиотеки DBV

Инициализировать подсистему DBV·
SUBROUTINE DBV_INI()

Получить атрибуты переменных:
SUBROUTINE L_FULLNANE(ARGNAME,FULLNAME)
FUNCTION L_N1(ARGNAME)
FUNCTION L_N2(ARGNAME)
FUNCTION L_N3(ARGNAME)
FUNCTION L_TYPE(ARGNAME)
FUNCTION L_DIM(ARGNAME)
FUNCTION L_KEY(ARGNAME)
SUBROUTINE L_UNITS(ARGNAME,TUNITS,SCALE,SHIFT)

Получить адрес переменной:
FUNCTION L_P(ARGNAME)

Узнать, существует ли переменная с таким именем:
FUNCTION L_MEM(ARGNAME)

Получить внутренний индекс переменной:
FUNCTION L_INDEX(ARGNAME)

Добавить переменную в базу данных:
SUBROUTINE DBV_ADD(ARGNAME,FULLNAME,INX,INY,INZ,TYPE,IKEY,
SCALE,SHIFT,TUNITS,NMAX)

Прочитать данные из входных SDF-файлов во внутреннюю базу данных на основе информации, содержащейся в PAR-файле:
SUBROUTINE DBV_READ(FILEPAR,IFILE, I2,R4,I4,R8,CH,NI2,NR4,NI4,NR8,NCH)

Записать данные из внутренней базы в выходные SDF-файлы на основе информации, содержащейся в PAR-файле:
SUBROUTINE DBV_WRITE(IFILE,I2,R4,I4,R8,CH)

Прочитать данные из SDF-файла:
SUBROUTINE DBV_FREAD(FIDENT,IFILE,I2,R4,I4,R8,CH)

Записать данные в SDF-файл:
SUBROUTINE DBV_FWRITE(FIDENT,IFILE,I2,R4,I4,R8,CH)

Создать файл параметров:
SUBROUTINE DBV_WPAR(FILEPAR,IFILE)

Вывести содержимое внутренней базы данных:
SUBROUTINE DBV_OUT(FNAME,IFILE)

Вывести таблицу использования памяти во внутренней базе данных:
SUBROUTINE DBV_OUTMEM(NI2,NR4,NI4,NR8,NCH)

Установить коэффициенты пересчёта для переменных с данным качеством:
SUBROUTINE DBV_QLT_SETSCALE(QLT,SCALE,SHIFT)

Установить диапазоны изменения для переменных с заданным качеством:
SUBROUTINE DBV_QLT_SETMINMAX(QLT,QMIN,QMAX)

Установить коэффициенты пересчёта значений для переменной:
SUBROUTINE DBV_SETSCALE(VARNAME1,SCALE,SHIFT)

Установить диапазоны изменения значений для переменной:
SUBROUTINE DBV_SETMINMAX(VARNAME1,QMIN,QMAX)

Проверить значения переменных, с данным качеством на диапазон:
SUBROUTINE DBV_QLT_CHECKMINMAX(QLT)

3.1 Состав библиотеки

Подсистема DBV включает в себя несколько программных модулей:

MEMORY.INC	содержит COMMON-блок MEMORY, являющийся местом хранения значений переменных для разных типов данных; пользователь задаёт необходимые ему размеры памяти в соответствии с потребностью конкретной задачи;
DBV.INC	содержит внутренние структуры и параметры подсистемы DBV;
DBV.FOR	основная библиотека подпрограмм подсистемы;
DBV1.FOR	вспомогательная внутренняя библиотека;
DBVSUB.FOR	вспомогательная внутренняя библиотека для различных преобразований данных;
DBVSCR.FOR	вспомогательная внутренняя библиотека для вывода на экран дисплея;
SDF.FOR	внутренняя библиотека для работы с SDF-файлами.

Самой первой подпрограммой при обращении к DBV должна быть вызов подпрограммы инициализации DBV_INI. Как правило, вычислительной программе требуется осуществлять ввод-вывод из файлов данных. Поэтому во вторую очередь требуется вызвать подпрограмму DBV_READ, которая на основе информации, содержащейся в PAR-файле установит интерфейс между переменными внутри программы и файлами данных, а также прочитает данные из входных SDF-файлов во внутреннюю базу

Если при работе библиотеки происходит ошибка, то на экран дисплея выдаётся соответствующее диагностическое сообщение и программа останавливается. Как правило, в случае ошибки дальнейшая работа не может быть корректно продолжена без изменений данных либо программы. Механизмы RESTART-файлов и периодического сброса данных в выходные SDF-файлы позволяют изучить накопленную информацию средствами RCS и понять причины аварийной ситуации.

3.2 Файл MEMORY.INC

Файл MEMORY.INC содержит COMMON-блок MEMORY, буфером для хранения значений переменных разных типов данных.

Пользователь задаёт необходимые ему размеры памяти в соответствии с потребностью в конкретной задаче:

NI2 — размер буфера для типа INT_ (INTEGER);
NR4 — размер буфера для типа FLOAT_ (INTEGER);
NI4 — размер буфера для типа LONGINT_ (INTEGER);
NR8 — размер буфера для типа DOUBLE_ (INTEGER);
NCH — размер буфера для типа CHAR_ (INTEGER).

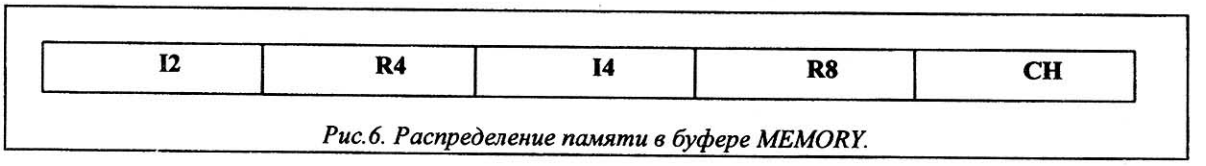

Рис.6. Распределение памяти в буфере MEMORY.

```
C ****************************************************************
C MEMORY.INC - COMMON-BLOCK FOR DATA BASE MEMORY
C
C ****************************************************************
      PARAMETER(NI2=1000,NR4=1000,NI4=1000,NR8=1000,NCH=1000)
      COMMON/MEMORY/ I2(NI2),R4(NR4),I4(NI4),R8(NR8),CH(NCH)
      INTEGER*2 I2
      REAL*4   R4
      INTEGER*4 I4
      REAL*8   R8
      CHARACTER CH
```

Рис.7 Пример установок размеров памяти в файле MEMORY.INC

При нехватке памяти подсистема DBV выдаёт соответствующее диагностическое сообщение и работа программы завершается. Внутри вычислительной программы переменные могут динамически добавляться, поэтому потребности в памяти могут зависеть как от размеров используемых массивов, так и от включения в работу тех или иных веток кода.

3.3 Файл DBV.INC

Содержимое файла DBV.INC приведено на рисунке 8.

Опишем переменные и параметры, входящие в этот файл.

MAX_DBVVARS
Максимальное количество переменных в базе данных.

MAX_DBVFILES
Максимальное количество файлов в базе данных.

LEN_DBVBUF
Длина внутреннего буфера для проведения необходимых преобразований данных во время чтения/записи.

```
C == DBV.INC
C COMMON-BLOCK FOR DATA BASE WITH VARIABLES
C
    PARAMETER(MAX_DBVVARS=300,MAX_DBVFILES=8,LEN_DBVBUF=50000)

    INTEGER  VOID_,CHAR_,STRUCT_,INT_,FLOAT_,
   *      TABLE_,LONGINT_,DOUBLE_,END_
    PARAMETER(VOID_=0,CHAR_=1,STRUCT_=2,INT_=3,FLOAT_=4,
   *      TABLE_=5,LONGINT_=6,DOUBLE_=7,END_=99)

    PARAMETER(IFILE_READ=1,IFILE_WRITE=2,IFILE_RW=3,
   *      IFILE_SCREEN=4,IFILE_RESTART=5,IFILE_NORESTART=6)

    INTEGER*4    VARNUM,FILENUM

    CHARACTER*15  VARNAME   (MAX_DBVVARS)
    CHARACTER*40  VARFULLNAME(MAX_DBVVARS)
    INTEGER*2    VARN1    (MAX_DBVVARS)
    INTEGER*2    VARN2    (MAX_DBVVARS)
    INTEGER*2    VARN3    (MAX_DBVVARS)
    INTEGER*2    VARDIM    (MAX_DBVVARS)
    INTEGER*2    VARTYPE   (MAX_DBVVARS)
    INTEGER*2    VARKEY    (MAX_DBVVARS)
    CHARACTER*15  VARTUNITS (MAX_DBVVARS)
    REAL*4       VARSCALE  (MAX_DBVVARS)
    REAL*4       VARSHIFT  (MAX_DBVVARS)
    REAL*4       VARMIN    (MAX_DBVVARS)
    REAL*4       VARMAX    (MAX_DBVVARS)
    INTEGER*4    VARPTR    (MAX_DBVVARS)

    CHARACTER*15  FVARNAME   (MAX_DBVFILES,MAX_DBVVARS)
    CHARACTER*40  FVARFULLNAME(MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARN1    (MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARN2    (MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARN3    (MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARDIM    (MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARTYPE   (MAX_DBVFILES,MAX_DBVVARS)
    INTEGER*2    FVARKEY    (MAX_DBVFILES,MAX_DBVVARS)
    CHARACTER*15  FVARTUNITS (MAX_DBVFILES,MAX_DBVVARS)
    REAL*4       FVARSCALE  (MAX_DBVFILES,MAX_DBVVARS)
    REAL*4       FVARSHIFT  (MAX_DBVFILES,MAX_DBVVARS)

    CHARACTER*80  FILENAME (MAX_DBVFILES)
    CHARACTER*8   FILEIDENT (MAX_DBVFILES)
    INTEGER*2    FILEREC   (MAX_DBVFILES)
    INTEGER*2    FILEMODREC(MAX_DBVFILES)
    INTEGER*2    FILERW    (MAX_DBVFILES)
```

Рис.8. Содержимое файла DBV.INC (начало).

```
      INTEGER*4     IOFFSI2,IOFFSR4,IOFFSI4,IOFFSR8,IOFFSCH

      CHARACTER*1   CHBUF(LEN_DBVBUF)
      INTEGER*2     I2BUF(LEN_DBVBUF)
      REAL*4        R4BUF(LEN_DBVBUF)
      INTEGER*4     I4BUF(LEN_DBVBUF)
      REAL*8        R8BUF(LEN_DBVBUF)
      EQUIVALENCE(CHBUF(1),R8BUF(1))
      EQUIVALENCE(I2BUF(1),R8BUF(1))
      EQUIVALENCE(I4BUF(1),R8BUF(1))
      EQUIVALENCE(R4BUF(1),R8BUF(1))

      COMMON /DBV/ VARNUM,FILENUM,
     *  IOFFSI2,IOFFSR4,IOFFSI4,IOFFSR8,IOFFSCH,
     *  VARNAME,VARFULLNAME,VARN1,VARN2,VARN3,VARDIM,
     *  VARTYPE,VARKEY,VARTUNITS,VARSCALE,VARSHIFT,
     *  VARMIN,VARMAX,VARPTR,
     *  FVARNAME,FVARFULLNAME,FVARN1,FVARN2,FVARN3,FVARDIM,
     *  FVARTYPE,FVARKEY,FVARTUNITS,FVARSCALE,FVARSHIFT,
     *  FILENAME,FILEIDENT,FILEREC,FILEMODREC,FILERW
      COMMON /DBVBUF/ R8BUF
```

Рис.8. Содержимое файла DBV.INC (окончание).

PARAMETER(VOID_=0,CHAR_=1,STRUCT_=2,INT_=3,FLOAT_=4,
TABLE_=5,LONGINT_=6,DOUBLE_=7,END_=99)
Целочисленные обозначения типов данных.

PARAMETER(IFILE_READ=1,IFILE_WRITE=2,IFILE_RW=3,
IFILE_SCREEN=4,IFILE_RESTART=5,IFILE_NORESTART=6)
Целочисленные обозначения типов файлов.

VARNUM
Количество переменных в базе данных.

FILENUM
Количество файлов в базе данных.

VARNAME	Имя переменной.
VARFULLNAME	Полное имя переменной.
VARN1	Первый размер переменной.
VARN2	Второй размер переменной.
VARN3	Третий размер переменной.
VARDIM	Размерность переменной.
VARTYPE	Тип переменной.
VARKEY	Ключ переменной.
VARTUNITS	Единицы измерения переменной.
VARSCALE	Коэффициент масштабирования значений переменной.
VARSHIFT	Коэффициент сдвига значений переменной.
VARMIN	Минимально допустимое значений переменной.
VARMAX	Максимально допустимое значений переменной.
VARPTR	Адрес переменной.

FVARNAME	Имя файловой переменной.

FVARFULLNAME	Полное имя файловой переменной.
FVARN1	Первый размер файловой переменной.
FVARN2	Второй размер файловой переменной.
FVARN3	Третий размер файловой переменной.
FVARDIM	Размерность файловой переменной.
FVARTYPE	Тип файловой переменной.
FVARKEY	Ключ файловой переменной.
FVARTUNITS	Единицы измерения файловой переменной.
FVARSCALE	Коэффициент масштабирования значений файловой переменной.
FVARSHIFT	Коэффициент сдвига значений файловой переменной.
FVARMIN	Минимально допустимое значений файловой переменной.
FVARMAX	Максимально допустимое значений файловой переменной.

FILENAME	Имя файла.
FILEIDENT	Идентификатор файла.
FILEREC	Номер записи файла.
FILEMODREC	Модуль номера записи файла.
FILERW	Тип файла (чтение,запись,рестарт).

IOFFSI2	Заполненность буфера для переменных типа INTEGER*2.
IOFFSR4	Заполненность буфера для переменных типа REAL*4.
IOFFSI4	Заполненность буфера для переменных типа INTEGER*4.
IOFFSR8	Заполненность буфера для переменных типа REAL*8.
IOFFSCH	Заполненность буфера для переменных типа CHARACTER.

CHBUF,I2BUF,R4BUF,I4BUF,R8BUF
Буфер для внутренних преобразований при чтении/записи переменных.

3.4 Основные подпрограммы

Данный раздел описывает подпрограммы, содержащиеся в файле DBV.FOR

В описываемых ниже подпрограммах параметры I2, R4, I4, R8, CH, NI2, NR4, NI4, NR8, NCH являются переменными include-файла MEMORY.INC.

Тип переменной задаётся целым числом. В include-файле DBV.INC содержатся символические имена типов:

PARAMETER (VOID_=0,CHAR_=1,STRUCT_=2,INT_=3,FLOAT_=4,
TABLE_=5,LONGINT_=6,DOUBLE_=7,END_=99)

SUBROUTINE DBV_INI()
Инициализация подсистемы DBV
Эту подпрограмму необходимо вызвать один раз в начале вычислительного кода.

SUBROUTINE L_FULLNANE(ARGNAME,FULLNAME)
Получить полное имя переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
FULLNAME - полное имя переменной (CHARACTER*40).

FUNCTION L_N1(ARGNAME)
Получить первый размер переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
первый размер (INTEGER).

FUNCTION L_N2(ARGNAME)
Получить второй размер переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
второй размер (INTEGER).

FUNCTION L_N3(ARGNAME)
Получить третий размер переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
третий размер (INTEGER).

FUNCTION L_TYPE(ARGNAME)
Получить тип переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
тип (INTEGER).

FUNCTION L_DIM(ARGNAME)
Получить размерность переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
размерность (INTEGER).

FUNCTION L_KEY(ARGNAME)
Получить ключ переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
ключ (INTEGER).

SUBROUTINE L_UNITS(ARGNAME,TUNITS,SCALE,SHIFT)
Получить единицы измерения переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
TUNITS - текстовое представление единиц измерения (CHARACTER*15);
SCALE - коэффициент масштабирования (REAL*4);
SHIFT - коэффициент сдвига (REAL*4).

FUNCTION L_P(ARGNAME)
Получить адрес переменной.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:
адрес (INTEGER).

FUNCTION L_MEM(ARGNAME)
Узнать. существует ли переменная с таким именем.
Вход:
ARGNAME - имя переменной (CHARACTER*15).
Выход:

1 - переменная существует, 0 - не существует (INTEGER).

FUNCTION L_INDEX(ARGNAME)

Получить внутренний индекс переменной.

Вход:

ARGNAME - имя переменной (CHARACTER*15).

Выход:

индекс переменной (INTEGER).

SUBROUTINE DBV_ADD(ARGNAME,FULLNAME,INX,INY,INZ,TYPE,IKEY, SCALE,SHIFT,TUNITS,NMAX)

Добавить переменную в базу данных.

DBV_ADD можно вообще не использовать, если все переменные описывать во входном PAR-файле.

Вход:

ARGNAME - имя переменной (CHARACTER*15).
FULLNAME - полное имя переменной (CHARACTER*40);
INX - размер по x (INTEGER);
INY - размер по y (INTEGER);
INZ - размер по z (INTEGER);
TYPE - тип данных (INTEGER);
IKEY - ключ (INTEGER);
SCALE - коэффициент масштабирования (REAL*4);
SHIFT - коэффициент сдвига (REAL*4);
TUNITS - текстовое представление единиц измерения (CHARACTER*15);
NMAX - размер буфера для типа данных TYPE
(соответствующая переменная из файла MEMORY.INC) (INTEGER).
Этот параметр позволяет контролировать переполнение буфера.

Выход:

нет.

SUBROUTINE DBV_READ(FILEPAR,IFILE, I2,R4,I4,R8,CH,NI2,NR4,NI4,NR8,NCH)

Прочитать данные из входных SDF-файлов во внутреннюю базу данных на основе информации, содержащейся в PAR-файле. Как правило, эта подпрограмма вызывается во вторую очередь после вызова подпрограммы DBV_INI. Она устанавливает интерфейс между переменными внутри программы и файлами данных.

Вход:

FILEPAR - имя PAR-файла (CHARACTER*80);
IFILE - номер канала ввода/вывода (INTEGER);
I2 - буфер для типа INT_ (INTEGER*2);
R4 - буфер для типа FLOAT_ (REAL*4);
I4 - буфер для типа LONGINT_ (INTEGER*4);
R8 - буфер для типа DOUBLE_ (REAL*8);
CH - буфер для типа CHAR_ (CHARACTER);
NI2 - размер буфера для типа INT_ (INTEGER);
NR4 - размер буфера для типа FLOAT_ (INTEGER);
NI4 - размер буфера для типа LONGINT_ (INTEGER);
NR8 - размер буфера для типа DOUBLE_ (INTEGER);
NCH - размер буфера для типа CHAR_ (INTEGER).

Выход:

нет.

SUBROUTINE DBV_WRITE(IFILE,I2,R4,I4,R8,CH)

Записать данные из внутренней базы в выходные SDF-файлы на основе информации, содержащейся в PAR-файле.

Вход:

IFILE - номер канала ввода/вывода (INTEGER);
I2 - буфер для типа INT_ (INTEGER*2);

R4 - буфер для типа FLOAT_ (REAL*4);
I4 - буфер для типа LONGINT_ (INTEGER*4);
R8 - буфер для типа DOUBLE_ (REAL*8);
CH - буфер для типа CHAR_ (CHARACTER).

Выход:

нет.

SUBROUTINE DBV_FREAD(FIDENT,IFILE,I2,R4,I4,R8,CH)

Прочитать данные из SDF-файла.

Вход:

FIDENT - идентификатор файла (CHARACTER*8);
IFILE - номер канала ввода/вывода (INTEGER);
I2 - буфер для типа INT_ (INTEGER*2);
R4 - буфер для типа FLOAT_ (REAL*4);
I4 - буфер для типа LONGINT_ (INTEGER*4);
R8 - буфер для типа DOUBLE_ (REAL*8);
CH - буфер для типа CHAR_ (CHARACTER).

Выход:

нет.

SUBROUTINE DBV_FWRITE(FIDENT,IFILE,I2,R4,I4,R8,CH)

Записать данные в SDF-файл.

Вход:

FIDENT - идентификатор файла (CHARACTER*8);
IFILE - номер канала ввода/вывода (INTEGER);
I2 - буфер для типа INT_ (INTEGER*2);
R4 - буфер для типа FLOAT_ (REAL*4);
I4 - буфер для типа LONGINT_ (INTEGER*4);
R8 - буфер для типа DOUBLE_ (REAL*8);
CH - буфер для типа CHAR_ (CHARACTER).

Выход:

нет.

SUBROUTINE DBV_WPAR(FILEPAR,IFILE)

Создать файл параметров. В качестве исходного материала служит файл параметров, считанный подпрограммой DBV_READ. В отличие от этого входного файла данная подпрограмма создаёт подробный файл без каких-либо сокращений.

Вход:

FILEPAR - имя PAR-файла (CHARACTER*80);
IFILE - номер канала ввода/вывода (INTEGER).

Выход:

нет.

SUBROUTINE DBV_OUT(FNAME,IFILE)

Вывести содержимое внутренней базы данных переменных со значениями атрибутов.

Вход:

FNAME - имя файла (CHARACTER*80);
IFILE - номер канала ввода/вывода (INTEGER).

Выход:

нет.

SUBROUTINE DBV_OUTMEM(NI2,NR4,NI4,NR8,NCH)

Вывести на стандартный вывод таблицу использования памяти во внутренней базе данных: сколько зарезервировано под каждый тип и сколько реально используется.

Вход:

NI2 - размер буфера для типа INT_ (INTEGER);
NR4 - размер буфера для типа FLOAT_ (INTEGER);

NI4 - размер буфера для типа LONGINT_ (INTEGER);
NR8 - размер буфера для типа DOUBLE_ (INTEGER);
NCH - размер буфера для типа CHAR_ (INTEGER).

Выход:

нет.

SUBROUTINE DBV_QLT_SETSCALE(QLT,SCALE,SHIFT)

Установить коэффициенты пересчёта значений для всех переменных с заданным качеством (единицей измерения).

Вход:

QLT - качество (CHARACTER*15);
SCALE - коэффициент масштабирования (REAL*4);
SHIFT - коэффициент сдвига (REAL*4).

Выход:

нет.

SUBROUTINE DBV_QLT_SETMINMAX(QLT,QMIN,QMAX)

Установить диапазоны изменения значений для всех переменных с заданным качеством (единицей измерения).

Вход:

QLT - качество (CHARACTER*15);
QMIN - минимальное значение (REAL*4);
QMAX - максимальное значение (REAL*4).

Выход:

нет.

SUBROUTINE DBV_SETSCALE(VARNAME1,SCALE,SHIFT)

Установить коэффициенты пересчёта значений для переменной.

Вход:

VARNAME - имя переменной (CHARACTER*15);
SCALE - коэффициент масштабирования (REAL*4);
SHIFT - коэффициент сдвига (REAL*4).

Выход:

нет.

SUBROUTINE DBV_SETMINMAX(VARNAME1,QMIN,QMAX)

Установить диапазоны изменения значений для переменной.

Вход:

VARNAME - имя переменной (CHARACTER*15);
QMIN - минимальное значение (REAL*4);
QMAX - максимальное значение (REAL*4).

Выход:

нет.

SUBROUTINE DBV_QLT_CHECKMINMAX(QLT)

Проверить значения переменных, имеющие данное качество на нахождение в ранее заданном диапазоне.

Вход:

QLT - качество (CHARACTER*15);

Выход:

нет.

Следующие три подпрограммы, включённые в файл DBV.FOR являются внутренними и не должны вызываться извне:

SUBROUTINE DBV_RPAR(FILEPAR,IFILE,NI2,NR4,NI4,NR8,NCH)

Чтение файла параметров.

SUBROUTINE DBV_WVINF(IFILE)

Создание файлов для записи и помещение в них описаний переменных.
SUBROUTINE DBV_RVPAR(IFILE,I2,R4,I4,R8,CH)
Чтение входных переменных во внутреннюю базу данных.

3.5 Внутренние подпрограммы

Перечисленные в данном разделе подпрограммы являются внутренними и не должны вызываться извне.

Файл **DBV1.FOR** содержит следующие подпрограммы:

FUNCTION L_INDEX1(ARGNAME)
Получить индекс переменной.

FUNCTION INDEX_DBV(ARGNAME,IFLGSTOP)
Получить индекс переменной и, в зависимости от флага реагировать на отсутсвие переменной как на ошибку или нет.

SUBROUTINE DBV_VALUE(JVAR,JFILE,JRW,IFILE,I2,R4,I4,R8,CH)
Считать или записать значение переменной с учётом единиц измерения, сжатия/растяжения и смены типа.

FUNCTION IDBV_TYPE(TYPE)
Получить номер типа данных по названию.

SUBROUTINE DBV_TYPE(ITYPE,TYPE)
Получить название типа данных по номеру

FUNCTION IDBV_DIM(INX,INY,INZ)
Получить размерность переменной по размерам массива.
FUNCTION IDBV_RW(RW)
Получить номер типа файла по названию.

SUBROUTINE DBV_RW(IRW,RW)
Получить название типа файла по номеру

SUBROUTINE DBV_STRATTR(NFILE,STRING,ASTERISK,DIMSCALE,VARREF)
Синтаксический разбор строки с описанием атрибутов.

SUBROUTINE DBV_I2ATTR(I2ATTR,IVARNUM,STRING,VARREF,ASTERISK,IPOS)
Синтаксический разбор атрибута типа INTEGER*2.

SUBROUTINE DBV_R4ATTR(R4ATTR,IVARNUM,STRING,VARREF,ASTERISK,IPOS)
Синтаксический разбор атрибута типа REAL*4.

SUBROUTINE DBV_N123ATTR (I2ATTR,IVARNUM,STRING,VARREF,ASTERISK,
DIMSCALE,IPOS)
Синтаксический разбор атрибута «размер».

SUBROUTINE DBV_TYPEATTR(I2ATTR,IVARNUM,STRING,VARREF,ASTERISK,IPOS)
Синтаксический разбор атрибута «тип данных».

SUBROUTINE DBV_CHATTR(CHATTR,IVARNUM,STRING,VARREF,ASTERISK,IPOS)
Синтаксический разбор атрибута типа CHARACTER.

FUNCTION IDBV_NFILE(IVAR,IRW)
Получить номер файла для чтения/записи конкретной переменной.

FUNCTION IDBV_VAL1(IVAR,IVARATTR,IFILE)
Получить значение переменной типа INTEGER*2 для занесение в значение атрибута другой переменной.

FUNCTION RDBV_VAL1(IVAR,IVARATTR,IFILE)
Получить значение переменной типа REAL*4 для занесение в значение атрибута другой переменной.

SUBROUTINE DBV_INS_(I)
Вставить переменную.

SUBROUTINE DBV_DEL_(I)
Удалить переменную.

FUNCTION IDBV_BEGINT(C)
Проверить может ли являться данный символ началом целочисленного значения.

FUNCTION IDBV_BEGREAL(C)
Проверить может ли являться данный символ началом вещественного значения.

SUBROUTINE DBV_ERRMEM(NI2,NR4,NI4,NR8,NCH,LI2,LR4,LI4,LR8,LCH)
Сообщение о нехватке памяти.

SUBROUTINE DBV_ERRMEM1(TY,NMAX,N)
Сообщение о нехватке памяти для заданного тиа данных.

SUBROUTINE DBV_ERROR(I,C1TEXT,C2TEXT,ITEXT)
Выдача диагностических сообщений об ошибках.

Файл **DBVSUB.FOR** содержит следующие подпрограммы:
Подпрограммы для работы с текстовыми строками:
FUNCTION ISTRLEN(SS)
SUBROUTINE STRCLR(SS)
SUBROUTINE STRCAT(SRC,APPSTR)
SUBROUTINE STRSTR(DST,SRC,IBEG,ILEN)
FUNCTION ISTRTOK(SRC,TOK)
FUNCTION ISTRCMP(STR1,STR2)
SUBROUTINE STRIDENT(SS)
FUNCTION ISTRPOS(STR,SUBSTR)

Подпрограммы для изменения типа значений переменной:
SUBROUTINE RETYPE_I2I4(I2,I4,LEN)
SUBROUTINE RETYPE_I2R4(I2,R4,LEN)
SUBROUTINE RETYPE_I2R8(I2,R8,LEN)
SUBROUTINE RETYPE_I2CH(I2,CH,LEN)
SUBROUTINE RETYPE_I4I2(I4,I2,LEN)
SUBROUTINE RETYPE_I4R4(I4,R4,LEN)
SUBROUTINE RETYPE_I4R8(I4,R8,LEN)
SUBROUTINE RETYPE_I4CH(I4,CH,LEN)
SUBROUTINE RETYPE_R4I2(R4,I2,LEN)
SUBROUTINE RETYPE_R4I4(R4,I4,LEN)
SUBROUTINE RETYPE_R4R8(R4,R8,LEN)
SUBROUTINE RETYPE_R4CH(R4,CH,LEN)
SUBROUTINE RETYPE_R8I2(R8,I2,LEN)
SUBROUTINE RETYPE_R8I4(R8,I4,LEN)
SUBROUTINE RETYPE_R8R4(R8,R4,LEN)
SUBROUTINE RETYPE_R8CH(R8,CH,LEN)

SUBROUTINE RETYPE_CHI2(CH,I2,LEN)
SUBROUTINE RETYPE_CHI4(CH,I4,LEN)
SUBROUTINE RETYPE_CHR4(CH,R4,LEN)
SUBROUTINE RETYPE_CHR8(CH,R8,LEN)

Подпрограммы для изменения масштаба значений переменной:
SUBROUTINE SCALE_I2(I2,LEN,COEFSRC,COEFDST)
SUBROUTINE SCALE_I4(I4,LEN,COEFSRC,COEFDST)
SUBROUTINE SCALE_R4(R4,LEN,COEFSRC,COEFDST)
SUBROUTINE SCALE_R8(R8,LEN,COEFSRC,COEFDST)
SUBROUTINE SCALE_CH(CH,LEN,COEFSRC,COEFDST)

Подпрограммы для сдвига значений переменной:
SUBROUTINE SHIFT_I2(I2,LEN,SHIFT)
SUBROUTINE SHIFT_I4(I4,LEN,SHIFT)
SUBROUTINE SHIFT_R4(R4,LEN,SHIFT)
SUBROUTINE SHIFT_R8(R8,LEN,SHIFT)
SUBROUTINE SHIFT_CH(CH,LEN,SHIFT)

Подпрограммы для сжатия/растяжения значений переменной:
SUBROUTINE REDIM_POINT(V222,A,V)
FUNCTION IREDIM_I2(FROM,TO,N1_FROM,N2_FROM,N3_FROM,N1_TO,N2_TO,N3_TO)
FUNCTION IREDIM_I4(FROM,TO,N1_FROM,N2_FROM,N3_FROM,N1_TO,N2_TO,N3_TO)
FUNCTION IREDIM_R4(FROM,TO,N1_FROM,N2_FROM,N3_FROM,N1_TO,N2_TO,N3_TO)
FUNCTION IREDIM_R8(FROM,TO,N1_FROM,N2_FROM,N3_FROM,N1_TO,N2_TO,N3_TO)
FUNCTION IFILE_EXIST(FILENAME,IFILE)

Файл **DBVSCR.FOR** содержит подпрограммы выдачи значений переменных на экран дисплея:
SUBROUTINE WVSCRI2(VARNAME1,I2BUF,N1,N2,N3)
SUBROUTINE WVSCRI4(VARNAME1,I4BUF,N1,N2,N3)
SUBROUTINE WVSCRR4(VARNAME1,R4BUF,N1,N2,N3)
SUBROUTINE WVSCRR8(VARNAME1,R8BUF,N1,N2,N3)
SUBROUTINE WVSCRCH(VARNAME1,CHBUF,N1,N2,N3)

Файл **SDF.FOR** содержит подпрограммы низкого уровня для работы с SDF-файлами.
С целью повышения скорости чтения/записи предусмотрен внутренний буфер размером 1024 байта. Создаваемые средствами DBV SDF-файлы имеют длину, кратную этой величине.

SUBROUTINE OPENSDF(FILENAME,IFILENAME,I)
Открыть SDF-файл.

SUBROUTINE CLOSESDF(IFILENAME)
Закрыть SDF-файл.

SUBROUTINE RFSDFINF(IFILENAME,IKEY,COMMENT,ICOMMENT,
ISHORT,IFULL,INRECS,INFIELDS)
Прочитать заголовок SDF-файла.

SUBROUTINE RVSDFINF(IFILENAME,IDENTV,INFIELDV,IKEYV,
IDTYPEV,IDIMV,INXV,INYV,INZV,TEXTUNITS,SCALE,SHIFT)
Прочитать атрибуты переменной из SDF-файла.

Подпрограммы для чтения значений переменных различных типов:
SUBROUTINE RVSDFI2(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE RVSDFR4(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE RVSDFI4(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE RVSDFR8(IFILENAME,IDENTV,IREC,VALUE)

SUBROUTINE WFSDFINF(IFILENAME,KEY,COMMENT,ICOMMENT,ISHORT,IFULL)
Записать заголовок SDF-файла.

SUBROUTINE WVSDFINF(IFILENAME,IDENTV,FULLIDENTV,IKEYV,
IDTYPEV,IDIMV,INXV,INYV,INZV,TEXTUNITS,SCALE,SHIFT)
Записать атрибуты переменной из SDF-файла.

Подпрограммы для записи значений переменных различных типов:
SUBROUTINE WVSDFI2(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE WVSDFR4(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE WVSDFI4(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE WVSDFR8(IFILENAME,IDENTV,IREC,VALUE)

Следующие подпрограммы являются вспомогательными:
SUBROUTINE BUF_CLEAR
SUBROUTINE BUF_FILE(IFILE,IPOS,S,N,LENS,IRWFLAG)
SUBROUTINE RVSDFVAL(IFILENAME,IDENTV,IREC,VALUE)
SUBROUTINE GETSDFINF(IFILENAME,SLABEL,IKEY,COMMENT,ICOMMENT,
ISHORT,IFULL,INRECS,ILREC,INFIELDS)
SUBROUTINE GETVARINF(IFILENAME,IDENTV,FULLIDENTV,
TEXTUNITS,SCALE,SHIFT,INFIELDV,IKEYV,
IDTYPEV,IDIMV,INXV,INYV,INZV,ILENV,IOFFSV,
ISIZE1,ISIZE2,IRECS,ILREC,IFIELDS)
SUBROUTINE GETFLDINF(IFILENAME,INFIELDV,IDENTV,FULLIDENTV,
TEXTUNITS,SCALE,SHIFT,IKEYV,
IDTYPEV,IDIMV,INXV,INYV,INZV,ILENV,IOFFSV,
ISIZE1,ISIZE2,IRECS,ILREC,IFIELDS)
SUBROUTINE WRRECFIELD(IFILENAME,IRECS,ILREC,IFIELDS)
SUBROUTINE PUTVARINF(IFILENAME,IDENTV,FULLIDENTV,
TEXTUNITS,SCALE,SHIFT,IKEYV,
IDTYPEV,IDIMV,INXV,INYV,INZV,
INFIELDV,ILENV,IOFFSV,
ISIZE1,ISIZE2,IRECS,ILREC,IFIELDS)
SUBROUTINE CLENDOFS(S)
SUBROUTINE RTOSR(IFILE,IPOS,S,N)
SUBROUTINE RTOCR(IFILE,IPOS,C,N)
SUBROUTINE GETOFFSET(IFILENAME,IFIELD,IOFFS)
SUBROUTINE WRENDOFS(S,I)
SUBROUTINE GETENDOFS(C)
SUBROUTINE WTOSR(IFILE,IPOS,S,N)
SUBROUTINE WTOCR(IFILE,IPOS,C,N)
SUBROUTINE ERROR_SDF(I,CTEXT,ITEXT,RTEXT)

3.6 Сообщения об ошибках

VARIABLE «VARNAME» IS ABSENT IN DATA BASE
Внутренняя база данных не содержит переменную VARNAME.

INVALID DIMENSION OF VARIABLE «VARNAME»
Для переменной VARNAME указана неверная размерность.

INVALID TYPE OF VARIABLE: «VARNAME»
Для переменной VARNAME указан неверный тип данных.

MAX NUMBER OF VARIABLES IN DATA BASE = NUMBER
Превышено максимально допустимое количество переменных NUMBER.

MAX NUMBER OF FILES IN DATA BASE = NUMBER
Превышено максимально допустимое количество файлов NUMBER.

VARIABLE «VARNAME» ALREADY EXISTS IN DATA BASE
Внутренняя база данных уже содержит переменную с именем VARNAME.

INVALID READ/WRITE FLAG FOR FILE: «FILEIDENT»
Попытка читать файл, открытый только на запись или записать в файл, открытый только на чтение.

NO SUCH FILEIDENT «FILEIDENT»
Файл с идентификатором FILEIDENT отсутствует.

INVALID ATTEMPT WRITE TO FILE «FILEIDENT» WITH FLAG: «FLAG»
Попытка записать в файл, открытый только на чтение.

WRONG INDEX OF VARIABLE FOR DATA BASE: NUMBER
Внутренняя ошибка: неверный индекс переменной в базе данных.

WRONG INDEX OF FILE FOR DATA BASE: NUMBER
Внутренняя ошибка: неверный индекс файла в базе данных.

DIFFERENT ATTRIBUTES FOR VARIABLE «VARNAME» IN DBV AND IN FILE «FILEIDENT»
Несовместимые атрибуты переменной VARNAME во внутренней базе данных и в файле FILEIDENT

THERE IS NO READING FILE FOR VARIABLE «VARNAME» TO DEFINE ATTRIBUTES
Невозможно определить атрибуты переменной VARNAME, так как они заданы по умолчанию как во входном файле, а входного файла с типом «на чтение» не задано. Данная ошибка возникает при чтении и анализе PAR-файла в подпрограмме DBV_READ.

BUFFER FOR DATA TRANSFORMATION TOO SMALL: DEFINE LEN_DBVBUF >= NUMBER
Буфер для проведения необходимых преобразований во время операций чтения/записи слишком мал. Требуется увеличить значение параметра LEN_DBVBUF, находящегося во входном файле DBV.INC до значения не меньшего чем NUMBER.

INVALID ATTRIBUTES OF VARIABLE «VARNAME1» TO DEFINE ATTRIBUTES VARIABLE «VARNAME2»
Неверные атрибуты переменной VARNAME1, значение которой используется как некоторый атрибут переменной VARNAME2. Данная ошибка возникает при чтении и анализе PAR-файла в подпрограмме DBV_READ.

NO SUCH FILE: «FILENAME» OR FILE OPEN ERROR
Нет файла с именем FILENAME или ошибки во время открытия файла.

4 Создание файла параметров из системы RCS

Так как файл параметров имеет дело с переменными SDF-файлов, а командный язык RCL имеет разнообразные средства для работы с переменными, то с целью расширения возможностей пользователя в RCL включены возможности по генерации файла параметров.

Прежде всего необходимо как то особо выделить модельные переменные. В системе RCS предполагается, что все модельные переменные находятся в отдельном SDF-файле.

Эта особая роль конкретного файла указывается при помощи команды

SET PARAMETERS <fident>

Файл <fident> должен быть ранее открыт или создан.

В дальнейшем система RCS считает этот файл особым для команд:

PARAMETER, SAVE PARAMETERS, VIEW PARAMETERS.

Во всех других отношениях - это обычный SDF-файл, к которому применимы все команды языка RCL.

Связь между модельной и файловой переменной задается командой:

PARAMETER <new_var> = <old_var>

Структура этой команды аналогична оператору присваивания в языках программирования. Левая часть задает новую переменную, а правая часть - существующую. Атрибуты новой переменной задаются в явном виде либо заимствуются у существующей.

```
par  p:X = fi:X
par  fstep:Te = p:Te
```

Если в левой части стоит модельная переменная, то она является входной для вычислительного модуля и правая часть указывает источник ввода. Если же в левой части указана файловая переменная, то она - выходная и правая часть указывает модельную переменную-источник. Одна и таже переменная может встречаться несколько раз и являться как источником ввода, так и вывода. Не имеет смысла и распознается как ошибка задание в обоих частях команды PARAMETER модельных либо файловых переменных.

В команде PARAMETER параллельно с формированием атрибутов модельной переменной запоминается и сама связь с файловой переменной, что является необходимым для формирования текстового файла параметров. Повторный вызов команды SET PARAMETERS очищает информацию о связях и отменяет действие ранее введенных команд PARAMETER.

Команда

SAVE PARAMETERS <"filename">

формирует и записывает на диск текстовый файла параметров.

Команда

VIEW PARAMETERS

высвечивает на экране дисплея последовательность ранее введенных команд PARAMETER.

Приведем пример CMD-файла для задания файла параметров.

```
open finit "init.sdf"
open fstep "step.sdf" /new
open p "par.sdf" /new

set parameters p
par double Te = finit:Te
par      fTe = finit:Te

par float fstep:X(51)    = float X(51)
par float fstep:Y(51)    = float Y(51)
par float fstep:Te(51,51) = double Te(51,51)
view par
save par "test.par"

close p
close finit
close fstep
```

5 Пример использования DBV

Приведем пример использования подсистемы DBV в вычислительном модуле на языке FORTRAN.

Пример был составлен исключительно для демонстрационных целей. В нём зачитывается входной файл параметров IN.PAR, затем во внутреннюю базу данных добавляются четыре новые переменные,

после чего на экран дисплея выводится таблица распределения памяти во внутренней базе данных, содержимое внутренней базы выводится в файл TEST.DBV и создаётся файл параметров OUT.PAR. Далее простейший прототип вычислительного процесса вычисляет значение переменной Te, в переменную Time заносится некоторое значение и производится запись переменных в первую запись SDF-файла STEP Далее в переменную Time заносится другое значение, значения переменной Te изменяются и производится запись переменных в следующую (вторую) запись SDF-файла STEP.

Содержимое входного файла параметров IN.PAR приведено в первой главе на рисунке 2.

Атрибуты модельных переменных, заданные в этом файле, будут восприниматься следующим образом:

Переменная N получает значения N1,N2,N3,DIM,TYPE из входного файла INIT Атрибуты KEY,SCALE,SHIFT,TUNITS заданы непосредственно в PAR-файле.

Для переменной M все атрибуты заданы непосредственно.

Для переменной K все атрибуты заносятся из входного файла INIT

Для переменной Time все атрибуты заданы непосредственно.

Для переменной Te атрибуты TYPE,SCALE,SHIFT заданы непосредственно, остальные заносятся из входного файла INIT

У переменной Ent атрибут N2 имеет такое же значение, как и атрибут N2 переменной Te.

Переменная WKK имеет атрибуты TYPE,SCALE,SHIFT такие же, как и у переменной Ent.

Переменная WNK имеет атрибуты TYPE,SCALE,SHIFT такие же, как и у переменной Time.

В файле IN.PAR заданы также атрибуты переменной Te для файла STEP

Переменная Te в файле имеет тип REAL*4 и ключ (KEY) равный 5, N1 и N2 будут уменьшены в 10 раз от аналогичных атрибутов модельной переменной Te, а все остальные атрибуты будут совпадать с исходными.

Из входного файла INIT во внутреннюю базу данных будут читаться переменные: N,M,K,Time,Te.

В выходной файл STEP будут занесены три переменные: N (под именем NN), Time, Te. Причем переменные N и Time будут копией модельных переменных с теми же атрибутами и значениями, в то время как переменная Te будет преобразованием с типа REAL*8 в тип REAL*4 и значения проредятся в 10 раз.

Если в файле IN.PAR у выводной переменной задать значения коэффициентов SCALE и/или SHIFT, отличающиеся от соответствующей модельной переменной, то при записи в файл будут пересчитываться значения с учетом этих коэффициентов, отражая таким образом перевод в другие единицы измерения.

Содержимое входных и выходных SDF-файлов можно исследовать при помощи системы обработки и визуализации данных RCS.

Ниже приводится текст программы на языке FORTRAN, демонстрирующий основные принципы использования подсистемы DBV

```
C ---------------------------------------
C  Демонстрационный пример
C  вычислительного модуля
C  для работы с подсистемой DBV
C ---------------------------------------
      PROGRAM TEST
C Включить INC-файл с выделенной буферной памятью для хранения переменных.
C Размеры резервируемой памяти регулируются параметрами
C NI2,NI4,NR4,NR8,NCH внутри файла memory.inc.
C
      include 'memory.inc'
C Задать канал для ввода/вывода 10
C
      IFILE=10

C Инициализация подсистемы DBV.
```

```
C Осуществляется один раз в начале работы модуля.
C
      CALL DBV_INI()

C Чтение входного файла параметров in.par по каналу ввода IFILE.
C На основе информации, содержащейся в этом файле, создаются
C переменные и производится чтение входных данных из SDF-файлов
C во внутреннюю базу данных.
C Параметры I2,R4,I4,R8,CH являются буферными переменными из файла
C MEMORY.INC.
C Параметры NI2,NR4,NI4,NR8,NCH являются параметрами из файла MEMORY.INC,
C контролирующими нехватку памяти.
C В данном примере в базе данных появляются следующие переменные:
C   N,M,K,Time,Te,Ent,Work,WKK,WNK
C Причем первые пять переменных (N,M,K,Time,Te) связаны с переменными
C из входного файла INIT, открытого на чтение. Поэтому значения этих
C переменных считываются из этого файла во внутреннюю базу данных.
C Остальные четыре переменные (Ent,Work,WKK,WNK) не имеют связи с файлами
C и выполняют роль рабочих переменных.
C
      CALL DBV_READ('in.par',IFILE, I2,R4,I4,R8,CH,
     +          NI2,NR4,NI4,NR8,NCH)

C Подпрограмма DBV_ADD задает следующие параметры:
C  имя, полное имя, размеры по x,y и z,тип данных,ключ,
C  коэффициенты масштабирования и сдвига, текст единиц измерения,
C  размер буфера для этого типа данных (из файла MEMORY.INC)
C Ниже создаются и добавляются четыре рабочих переменные:
C  REAL*8  Q(51,51)
C  REAL*8  FU(51,51)
C  REAL*8  FL(51,51)
C  REAL*4  Te+(100,100)
C
      CALL DBV_ADD('Q',  '*',51, 51, 1,'R8',0,2.,0.,'UNITS',NR8)
      CALL DBV_ADD('FU',  '*',51, 51, 1,'R8',0,2.,0.,'UNITS',NR8)
      CALL DBV_ADD('FL',  '*',51, 51, 1,'R8',0,2.,0.,'UNITS',NR8)
      CALL DBV_ADD('Te+', '*',100,100,1,'R4',0,2.,273.,'UNITS',NR4)

C Вывести на стандартный вывод таблицу распределения памяти
C во внутренней базе данных: сколько зарезервировано под каждый тип
C и сколько реально используется.
C
      CALL DBV_OUTMEM(NI2,NR4,NI4,NR8,NCH)

C Вывод содержимого внутренней базы данных (дампинг) в файл TEST.DBV
C Данное средство является дополнительным при отладке программ.
C
      CALL DBV_OUT('test.dbv',IFILE)

C Создание файла параметров out.par. В отличии от входного файла
C in.par выводится подробная информация об атрибутах всех переменных.
C Данное средство является дополнительным при отладке программ.
C
      CALL DBV_WPAR('out.par',IFILE)
C Получить индекс переменной 'Te' во внутренней базе данных и занести
C его в рабочую переменную L_TE.
C Использование локальных переменных для хранения индексов часто
```

```
C используемых данных может заметно повлиять на скорость работы,
C так как частый поиск во внутренней базе данных с большим количеством
C переменных требует затрат некоторого времени.
C
      L_TE = L_P('Te')

C Вызов простейшего прототипа вычислительного процесса
C (текст следует ниже).
C В подпрограмме инициализируется двумерный вещественный массив
C двойной точности.
C  Первый параметр R8(L_TE) задает начало массива.
C  Второй параметр L_N1('Te') задает первую размерность массива 'Te'
C  Третий параметр L_N2('Te') задает вторую размерность массива 'Te'
C Функции L_N1() и L_N2(), при помощи которых извлекаются размерности
C переменных, находятся в библиотеке DBV.
C
      CALL  FILL_TE( R8(L_TE), L_N1('Te'), L_N2('Te'))

C Занести в переменную 'Time' значение 1234. Индекс переменной 'Time'
C во внутренней базе данных поступает при обращении к функции L_P()
C подсистемы DBV
      R4(L_P('Time')) = 1234.

C Записать данные в SDF-файл 'STEP' в первую запись.
C Переменные, подлежащие записи определяются на основе ранее считанного
C файла параметров in.par.
C В SDF-файл 'STEP' попадут следующие переменные из внутренней
C базы данных:
C  N (под именем NN), Time, Te.
C Причем переменная Te будет преобразована в выходном файле: она поменяет
C тип с REAL*8 на REAL*4 и будет разрежена в 10 раз по обоим размерностям.
C
      CALL DBV_FWRITE('STEP',IFILE,I2,R4,I4,R8,CH)

C Занести в переменную 'Time' значение 5678. Индекс переменной 'Time'
C во внутренней базе данных поступает при обращении к функции L_P()
C подсистемы DBV
C
      R4(L_P('Time')) = 5678.

C Вызов простейшего прототипа вычислительного процесса
C (текст следует ниже).
C В подпрограмме модифицируется двумерный вещественный массив
C двойной точности: все его элементы делятся на 2.
C  Первый параметр R8(L_TE) задает начало массива.
C  Второй параметр L_N1('Te') задает первую размерность массива 'Te'.
C  Третий параметр L_N2('Te') задает вторую размерность массива 'Te'
C Функции L_N1() и L_N2(), при помощи которых извлекаются размерности
C переменных, находятся в библиотеке DBV
C
      CALL  MODIFY_TE( R8(L_TE), L_N1('Te'), L_N2('Te'))

C Записать данные в SDF-файл 'STEP' во вторую запись.
C Остаются в силе замечания, сделанные при формировании первой
C записи файла.
C
      CALL DBV_FWRITE('STEP',IFILE,I2,R4,I4,R8,CH)
```

```
      STOP
      END

C ============================================================
C Вспомогательные подпрограммы демонстрационного примера
C ============================================================

C Инициализация двумерного вещественного массива двойной точности.
C Параметры:
C  TE - массив
C  N1 - первая размерность массива
C  N2 - вторая размерность массива
C
      SUBROUTINE FILL_TE(TE,N1,N2)
      REAL*8 TE(N1,N2)
      DO I=1,N1
      DO J=1,N2
      TE(I,J) = I + (J-1)*N1
      ENDDO
      ENDDO
      RETURN
      END

C Модификация двумерного вещественного массива двойной точности:
C все его элементы делятся на 2.
C Параметры:
C  TE - массив
C  N1 - первая размерность массива
C  N2 - вторая размерность массива
C
      SUBROUTINE MODIFY_TE(TE,N1,N2)
      REAL*8 TE(N1,N2)
      DO I=1,N1
      DO J=1,N2
      TE(I,J) = TE(I,J) * 0.5
      ENDDO
      ENDDO
      RETURN
      END
```

На рисунке 9 приведено содержимое файла OUT.PAR, в котором можно увидеть как интерпретировались атрибуты переменных, заданные во входном файле IN.PAR.

```
*** FILES ***
IDENT     R/W     NREC MODREC FILENAME
-------------------------------------------------------------------------
INIT       R         1      1 init.sdf
STEP       W         1    100 step.sdf
1D         W         1      1 1d.sdf

*** MODEL VARIABLES ***
VARNAME N1    N2    N3   DIM TYPE KEY SCALE SHIFT TUNITS       INIT   STEP   1D
-------------------------------------------------------------------------------
N       1     1     1    0   I2   0   1.0   0     N_UNITS      N      NN     -
M       1     1     1    1   I2   0   1.0   0     M_UNITS      M      -      -
K       1     1     1    0   I2   0   1.0   0     -            K      -      -
Time    1     1     1    1   R4   0   1.0   0     Time_UNITS   Time   Time   -
Te      100   100   1    2   R8   0   1.0   0     Te_UNITS      Te    Te     -
Ent     100   100   1    2   R8   2   2.5   0     -             -     -      -
Work    15    15    1    2   R8   2   2.5   0     -             -     -      -
WKK     33    33    1    2   R8   33  2.5   0     -             -     -      -
WNK     100   33    1    2   R4   100 1.0   0     -             -     -      -
Q       51    51    1    2   R8   0   2.0   0     UNITS         -     -      -
FU      51    51    1    2   R8   0   2.0   0     UNITS         -     -      -
FL      51    51    1    2   R8   0   2.0   0     UNITS         -     -      -
Te+     100   100   1    2   R4   0   2.0   273   UNITS         -     -      -

*** FILE VARIABLES ***

*** FILE: INIT
VARNAME  N1    N2    N3  DIM TYPE   KEY  SCALE SHIFT TUNITS
-------------------------------------------------------------------------
N        1     1     1   0   I2     0    1.0   0
M        1     1     1   0   I2     0    1.0   0
K        1     1     1   0   I2     0    1.0   0
Time     1     1     1   0   R4     0    1.0   0
Te       100   100   1   2   R4     0    1.0   0

*** FILE: STEP
VARNAME  N1    N2    N3  DIM TYPE   KEY  SCALE SHIFT TUNITS
-----------------------------------------------------------------
NN       1     1     1   0   I2     0    1.0   0.0   N_UNITS
Time     1     1     1   1   R4     0    1.0   0.0   Time_UNITS
Te       10    10    1   2   R4     5    2.0   15    Te_UNITS

*** FILE: 1D
VARNAME  N1    N2    N3  DIM TYPE   KEY   SCALE SHIFT TUNITS
-----------------------------------------------------------------
```

Рис 9. Содержимое файла OUT.PAR.

Литература.

1. *Арутюнян Р.В., Большов Л.А., Головизнин В.М., Варенков В.В., Попков А.Г., Стрижов В.Ф., Чуданов В.В.* Комплекс программ «РАСПЛАВ'' для анализа взаимодействия расплава с бетоном. Сб. Проблемы безопасного развития атомной энергетики. М.. Наука, 1993.
2. *Варенков В.В., Первичко В.А., Попков А.Г., Чуданов В.В.* Программная система RCS: организация вычислений для моделирования физических процессов. Препринт IBRAE-95-05. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1995.
3. *Варенков В.В., Первичко В.А., Попков А.Г* Самодокументированный файл данных. Препринт NSI-16-93. М.: Институт проблем безопасного развития атомной энергетики РАН, 1993.
4. *Aksenova A.E., Chudanov V V., Goloviznin V.M., Pervichko V.A., Popkov A.G., Varenkov V V* Interactive Postprocessor VV-2D for scientific and engineering applications. Preprint NSI-3-94. Moscow· Nuclear Safety Institute, 1994.
5. *Аксенова А.Е., Варенков В.В., Первичко В.А., Попков А.Г., Чуданов В.В.* Интерактивный пакет подготовки данных для решения задач математической физики. Препринт NSI-32-94. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1994.