# АННОТАЦИЯ

Данный документ содержит описание языка ФОРТРАН для CP/M.

В разделе 1 описаны элементы и структура программы, написанной на языке ФОРТРАН.

В разделе 2 дано описание формата строки.

Элементы и структура данных языка ФОРТРАН описаны в разделе 3, где дается подробное описание компонент операторов языка и типов данных.

Раздел 4 содержит описание выражений.

Операторы языка описаны в разделе 5.

Раздел 6 посвящен описанию подпрограмм, их параметров, арифметических операторов-функций и подпрограмм-функций.

В документе имеется 2 приложения, содержащих справочные сведения по языку ФОРТРАН.

В приложении 1 кратко изложены основные сведения об операторах языка.

В приложении 2 дан перечень библиотечных функций.

**Содержание**

АННОТАЦИЯ 1

Введение 4

1. Общие сведения и способ описания языка 4

1.1. Компоненты программы на языке ФОРТРАН 4

1.1.1. Операторы 5

1.1.2. Комментарии 5

2. Элементы и основные конструкции языка ФОРТРАН 5

2.1. Набор символов языка ФОРТРАН 5

2.2. Задание формата строки 6

2.2.1. Использование бланка программирования на языке ФОРТРАН 6

2.2.2. Использование редактора текстов 6

3. Элементы данных языка ФОРТРАН 7

3.1. Идентификаторы 7

3.2. Типы данных 8

3.3. Константы 9

3.3.1. Целые константы 9

3.3.2. Вещественные константы 9

3.3.3. Константы с двойной точностью 10

3.3.4. Шестнадцатеричные константы 10

3.3.5. Логические константы 11

3.3.6. Текстовые константы 11

3.4. Переменные 12

3.4.1. Спецификация типа данных 13

3.4.2. Оператор задания типа данных по первой букве имени (IMPLICIT) 13

3.5. Массивы 14

3.5.1. Описатели массивов 14

3.5.2. Списки индексов 14

3.5.3. Расположение массивов в памяти 15

3.5.4. Тип данных массива 16

3.5.5. Обращения к массивам без использования списка индексов 16

3.5.6. Массивы с переменными границами 16

4. Выражения 17

4.1. Арифметические выражения 17

4.1.1. Применение скобок 18

4.1.2. Тип значения арифметического выражения 19

4.2. Выражения отношений 20

4.3. Логические выражения 20

5. Операторы языка ФОРТРАН 22

5.1. Операторы присваивания 22

5.1.1. Оператор присваивания 22

5.1.2. Оператор "ASSIGN" 23

5.2. Операторы управления 24

5.2.1. Оператор "GO TO" 24

5.2.2. Операторы "IF" 26

5.2.3. Оператор "DO" 27

5.2.4. Оператор "CONTINUE" 30

5.2.5. Оператор "CALL" 30

5.2.6. Оператор "RETURN" 31

5.2.7. Оператор "PAUSE" 32

5.2.8. Оператор "STOP" 32

5.2.9. Оператор "END" 32

5.3. Ввод/вывод данных 32

5.3.1. Номера логических устройств ввода/вывода 33

5.3.2. Форматный последовательный ввод/вывод 34

5.3.3. Неформатный ввод/вывод 36

5.3.4. Ввод/вывод с прямым доступом 37

5.3.5. Единицы ввода/вывода 37

5.3.6. Списки ввода/вывода 37

5.4. Оператор "FORMAT" 39

5.4.1. Спецификации преобразования 40

5.4.2. Масштабный коэффициент 47

5.4.3. Группирование и повторяемые группы 48

5.4.4. Управление кареткой 49

5.4.5. Разделительные знаки списка спецификаций 49

5.4.6. Взаимодействие управления форматом и списка ввода/вывода 50

5.4.7. Сводка правил для оператора "FORMAT" 51

5.5. Операторы описания 52

5.5.1. Операторы описания типа 53

5.5.2. Оператор "DIMENSION" 53

5.4.3. Оператор "COMMON" 54

5.5.4. Оператор "EQUIVALENCE" 55

5.5.5. Оператор "EXTERNAL" 57

5.5.6. Оператор "DATA" 58

5.5.7. Оператор "PROGRAM" 59

6. Подпрограммы 59

6.1. Параметры подпрограмм 59

6.2. Пользовательские подпрограммы 60

6.2.1. Арифметические операторы-функции 60

6.2.2. Подпрограмма-функция 61

6.2.3. Подпрограмма 62

6.2.4. Подпрограмма данных "BLOCK DATA" 64

6.2.5. Библиотечные функции языка ФОРТРАН 64

Приложение 1 Основные сведения о языке ФОРТРАН 65

Приложение 2 Библиотечные функции языка ФОРТРАН 71

# Введение

Обязательным условием применения языка ФОРТРАН является знание пользователем документа [1], содержащего разъяснение основных понятий и общей организации CP/M 2.2, список принятых в документации по CP/M 2.2 сокращений и условных обозначений, а также способов ввода управляющих и графических символов клавиатуры видеотерминала.

Язык ФОРТРАН для CP/M представляет собой реализацию ФОРТРАН, принятого в качестве стандарта со следующими расширениями:

1. При задании операторов "STOP С" и "PAUSE C" может быть не только десятичным числом, но и любой цепочкой, содержащей до 6 символов в коде, принятом в CP/M 2.2;
2. В операторах READ и WRITE реализована возможность передачи управления при ошибке (ERR=) и концу файла (END=);
3. В операторной функции можно использовать переменные с индексами;
4. Во всех случаях возможно использование вместо целых констант шестнадцатеричных констант;
5. Допустимо использование в выражениях литералов вместо целых констант;
6. Не накладываются ограничения на количество строк продолжения;
7. В выражениях и операторах присваивания возможно использование данных разных типов.

Ниже перечислены ограничения языка ФОРТРАН для CP/M по сравнению со стандартным:

1. Не реализованы данные типа COMPLEX;
2. Операторы спецификации должны располагаться в таком порядке:

* операторы PROGRAM, SUBROUTINE, FUNCTION, BLOCK DATA;
* операторы спецификации типа EXTERNAL и DIMENSION;
* операторы COMMON;
* оператор EQUIVALENCE;
* операторы DATA;
* операторные функции;

1. Знак равенства в операторе присваивания и первая запятая в операторе DO не могут быть помещены в строке продолжения;
2. Целые переменные типа INTEGER\*4 не могут быть использованы при вычислении индексов, а также в качестве переменной цикла DO и его параметров.

# 1. Общие сведения и способ описания языка

## 1.1. Компоненты программы на языке ФОРТРАН

Программа на языке ФОРТРАН состоит из операторов и необязательных комментариев. Операторы объединяются в логические блоки, называемые программными единицами. Программная единица является последовательностью операторов, определяющей некоторую процедуру вычисления. Программная единица может быть или программой, или подпрограммой, или подпрограммой-функцией, или подпрограммой BLOCK DATA. Выполняемая программа состоит из одной основной программы и любого количества подпрограмм и подпрограмм-функций.

Компилятор различает следующие типы программных единиц:

1. Программу - последовательность операторов, начинающуюся с PROGRAM и заканчивающуюся END;
2. Подпрограмму, начинающуюся с SUBROUTINE и кончающуюся END;
3. Подпрограмму-функцию, начинающуюся с FUNCTION и кончающуюся END;
4. Подпрограмму BLOCK DATA, начинающуюся с BLOCK DATA и завершающуюся END.

Компилятор языка ФОРТРАН может обрабатывать файл, содержащий любое количество различных программных единиц.

### 1.1.1. Операторы

Все операторы делятся на два класса: выполняемые и невыполняемые операторы. невыполняемые операторы описывают размещение и характеристики данных, а также содержат информацию, необходимую для редактирования и преобразования данных.

Оператор состоит из одной или нескольких строк. Строкой называется последовательность символов до 80 знаков длиной, заканчивающаяся управляющим символом CR. Если размер оператора превышает допустимую длину строки, он может быть продолжен на следующей строке или на нескольких последующих строках, называемых строками продолжения. Строка продолжения идентифицируется наличием символа продолжения в шестой позиции этой строки.

Оператор может иметь метку, по которой другие операторы получат возможность обращаться к нему либо за информацией, либо для передачи ему управления. Метка оператора является целым числом, записанным в первых пяти позициях первой строки оператора.

### 1.1.2. Комментарии

Наличие комментария не влияет на выполнение программы. Он служит вспомогательной информацией для программиста. Комментарий используется для описания работы программы, идентификации частей программы и процессов обработки, для удобства чтения листинга программы на входном языке.

Буква "C" в первой позиции исходной строки указывает, что эта строка является комментарием.

# 2. Элементы и основные конструкции языка ФОРТРАН

## 2.1. Набор символов языка ФОРТРАН

При записи программ на языке ФОРТРАН используются следующие символы:

* прописные буквы латинского алфавита от "A" до "Z";
* цифры от "0" до "9";
* следующие специальные знаки:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | пробел; |  | ( | круглая скобка левая; |
| = | равно; |  | ) | круглая скобка правая; |
| + | плюс; |  | , | запятая; |
| - | минус; |  | . | точка; |
| \* | звездочка; |  | ' | апостроф; |
| / | косая (дробная) черта; |  | $ | знак денежной единицы. |

В набор символов языка ФОРТРАН не входят, но используются компилятором следующие управляющие символы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SI | вход; |  | LF | перевод строки; |
| SO | выход; |  | CR | возврат каретки. |
| HT | горизонтальная табуляция; |  |  |  |

В нечисловых литералах и комментариях могут использоваться все графические символы, допустимые в CP/M 2.2, в частности русские буквы. Особенности использования русских букв подробно описаны в документе [1].

## 2.2. Задание формата строки

Формат строки ФОРТРАН является одинаковым как для строк, вводимых с перфоленты, так и для строк, вводимых с консоли при помощи редактора текста. Различаются лишь способы формирования строк.

### 2.2.1. Использование бланка программирования на языке ФОРТРАН

Строка ФОРТРАН делится на поля, предназначенные для записи меток операторов, признаков продолжения, текста операторов и порядковых номеров операторов. Каждая позиция строки соответствует одному символу.

Метка оператора (или номер оператора) состоит из одной или нескольких (не более пяти) десятичных цифр, расположенных в первых пяти позициях начальной строки оператора. Пробелы и левые нули игнорируются. Запрещается использование метки оператора, состоящей только из нулей.

Любой оператор, на который существует ссылка из другого оператора, должен иметь метку. Никакие два оператора в программной единице не должны иметь одинаковых меток.

В позиции 1 может быть помещена буква "C". Это означает, что данная строка является комментарием. Компилятор выводит содержимое этой строки на листинг исходной программы, после чего эта строка игнорируется. Позиция 6 строки ФОРТРАН зарезервирована для признака продолжения.

Любой символ, за исключением нуля или пробела, помещенный в эту позицию, распознается как признак продолжения. Символы, начинающиеся с позиции 7 строки продолжения, рассматриваются как следующие за последним символом предыдущей строки.

Строки комментариев не могут быть продолжены. Все строки комментариев должны начинаться с буквы "C" в позиции 1. Строки комментариев не должны вставляться между начальной строкой оператора и его строкой продолжения, либо между идущими подряд строками продолжения.

Текст оператора ФОРТРАН помещается в позициях с 7 по 72. Так как компилятор игнорирует пробелы (за исключением находящихся в текстовых константах и буквенно-цифровых литералах), пользователь может вставлять в текст пробелы в любом месте для достижения большего удобства чтения.

В позициях с 73 по 80 любой строки исходного текста может помещаться порядковый номер строки или другая идентифицирующая информация. Текст будет игнорироваться без предупреждающего сообщения, если длина строки оператора окажется больше 72.

### 2.2.2. Использование редактора текстов

Если исходная программа вводится с консоли с помощью редактора текстов [2], пользователь может печатать строки по принципу "символ на позицию", либо использовать управляющий символ ТАВ для облегчения задания формата строк.

Если символ табуляции является первым в строке или стоит после метки оператора и следующим символом является цифра, то компилятор рассматривает цифру как признак продолжения, а следующий символ как начало текста оператора. Если символ, следующий за начальным символом табуляции, не является цифрой, то компилятор рассматривает его как первый символ текста оператора. Если признак продолжения отсутствует, то данная строка является начальной строкой.

Символ пробела может использоваться в операторе ФОРТРАН для удобства чтения строки. Транслятор игнорирует все пробелы в поле оператора за исключением пробелов внутри текстовых констант или внутри буквенно-цифровых литералов. Управляющий символ ТАВ в поле оператора воспринимается компилятором так же, как и пробел. Если в исходной программе используется управляющий символ ТАВ, то компилятор выставляет автоматически следующие позиции табуляции 9, 17, 25, 33 и т.д.

# 3. Элементы данных языка ФОРТРАН

Основными элементами операторов языка ФОРТРАН являются:

|  |  |  |
| --- | --- | --- |
| **Константа** | - | представляет фиксированную величину, которая не меняется в ходе выполнения программы; |
| **Переменная** | - | символическое имя (идентификатор), представляющее некоторую величину, хранящуюся в памяти; |
| **Массив** | - | группа однотипных величин, расположенных в памяти последовательно друг за другом. Отдельные величины, составляющие массив, называются элементами массива. Можно обращаться не только к массиву, но и к любому элементу массива. Для обращения к массиву используется идентификатор; |
| **Выражение** | - | может быть простой константой, переменной, элементом массива или обращением к функции. Выражение также может быть комбинацией вышеперечисленных элементов и элементов, называемых операциями. Операции определяют вид вычислений, которые надо выполнить над элементами для того, чтобы получить значение выражения; |
| **Обращение к функции** | - | идентификатор функции, за которым следует список аргументов (параметров). Появление в программе обращения к функции вызывает выполнение вычислений над параметрами согласно определению функции. |

## 3.1. Идентификаторы

Идентификатор - это последовательность букв и цифр, обязательно начинающаяся с буквы.

Примеры правильных и неправильных идентификаторов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Правильная запись** | |  | **Неправильная запись** | |
|  |  |  |  |  |
|  | NUMBER |  | 5Q | (начинается с цифры) |
|  | K9 |  | B.4 | (содержит специальный символ) |

В таблице 1 перечислены все объекты, которые обозначаются идентификаторами. В пределах программной единицы один и тот же идентификатор не может быть присвоен более, чем одному объекту, за исключением специальных случаев.

Внутри подпрограммы идентификаторы могут использоваться также в качестве формальных параметров. Формальный параметр может представлять переменную, массив, элемент массива, константу, выражение или подпрограмму.

**Таблица 1 Классификация идентификаторов**

| **Объект** | **Имеет тип** | **Единственный в программе** |
| --- | --- | --- |
| Переменная | да | нет |
| Массив | да | нет |
| Арифметический оператор-функция | да | нет |
| Библиотечная функция | да | да |
| Подпрограмма-функция | да | да |
| Подпрограмма | нет | да |
| Общий блок | нет | да |
| Основная программа | нет | да |
| Блок данных | нет | да |

## 3.2. Типы данных

Каждый основной элемент представляет данные одного из нескольких различных типов. Тип данных элемента может определяться самой конструкцией элемента, может задаваться по умолчанию, либо явным образом.

Типы данных, допустимые в ФОРТРАН следующие:

|  |  |  |
| --- | --- | --- |
| **Целый** | - | целое число; |
| **Вещественный** | - | число с десятичной точкой. Это может быть целое число, десятичная дробь или комбинация этих значений; |
| **С двойной точностью** | - | аналогично вещественным, но с повышенной точностью представления числа; |
| **Логический** | - | данное, принимающее логическое значение "истина" или "ложь". |

Важной характеристикой каждого типа данных является объем памяти, требующийся для хранения величины этого типа. Различия внутри основных типов выражаются либо точностью представления величины, либо диапазоном допустимых значений.

Текстовые константы и литералы не имеют типа данных. Они принимают тип данных в зависимости от того контекста, в котором они встречаются.

Для оптимального использования памяти и повышения производительности компилятора допускается использование дополнительных типов данных.

Распределение памяти по типам данных следующее:

|  |  |  |
| --- | --- | --- |
| INTEGER (INTEGER\*2) | - | занимает 2 байта памяти; |
| INTEGER\*1 | - | отводится 1 байт памяти; |
| REAL (REAL\*4) | - | отводится 4 байта памяти; |
| DOUBLE PRECISION (REAL\*8) | - | занимает 8 байтов памяти; |
| LOGICAL | - | отводится 1-байтовая область памяти, которая может содержать логические величины "истина" или "ложь", один буквенно-цифровой символ или целые величины от минус 128 до +127; |
| INTEGER\*4 | - | занимает 4 байта памяти; |
| LOGICAL\*2 | - | отводится 2-байтовая область памяти. Эквивалентно INTEGER\*2; |
| BYTE | - | занимает один байт. |

## 3.3. Константы

Константа представляет неизменяемую величину: логическую, числовое значение или последовательность символов.

### 3.3.1. Целые константы

Целая константа - это число, которое не имеет десятичной точки. Константе может предшествовать знак числа. Целая константа интерпретируется как десятичное число и имеет следующий вид:

snn

где

|  |  |  |
| --- | --- | --- |
| s | - | знак числа; |
| nn | - | последовательность десятичных цифр. |

Левые незначащие нули (если они есть) игнорируются. Отрицательной целой константе должен предшествовать знак "-". Знак "+" является необязательным символом. Если константа не имеет знака, она считается положительной.

Целая константа не должна содержать никаких символов, кроме цифр от 0 до 9 (знак числа ставится перед константой). Абсолютная величина целой константы не должна превышать 2147483647.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Правильные записи целой константы:** | |  | **Неправильные записи целой константы:** | |
| 0  +127  -32123 |  |  | 999999999999 | (слишком велика) |
|  | 3.14 | (не допускается использование  десятичной точки или запятой). |
|  | 25,154 |

### 3.3.2. Вещественные константы

Вещественная константа - это последовательность десятичных цифр, включающая одну десятичную точку.

Допустимы следующие записи вещественной константы:

s.nn или snn.nn или snn.

где

|  |  |
| --- | --- |
| s | - знак числа; |
| nn | - последовательность цифровых символов. |

Десятичная точка может находиться в любом месте последовательности. Число цифр не ограничивается, но в качестве значащих рассматриваются только восемь старших цифр.

Вещественная константа может быть записана также в таком виде:

s1d1...dk.dk+1...dnxs2pp

где

|  |  |
| --- | --- |
| s1 | - знак числа; |
| d1...dk | - целая часть константы; |
| dk+1...dn | - дробная часть константы; |
| x | - признак начала поля степени; |
| s2 | - знак степени; |
| pp | - однозначная или двузначная беззнаковая целая константа. |

Если степень положительна, знак "+" может быть опущен. Перед отрицательной степенью обязательно должен стоять знак "-".

Эта запись изображает степень десяти, на которую должна быть умножена предшествующая ей целая или вещественная константа (например, 1E6 представляет величину 1.0\*10\*\*6).

Вещественная константа занимает четыре байта и интерпретируется как вещественное число, имеющее степень точности немного больше, чем семь десятичных цифр.

За исключением знаков "+" и "-", десятичной точки и буквы "E" запись вещественной константы не должна содержать никаких символов, кроме цифр от 0 до 9.

Если в записи вещественной константы присутствует буква "E", то за ней обязательно должна следовать однозначная или двузначная целая константа. Поле степени не может быть пропущено, но может быть нулевым. Абсолютная величина вещественной константы не может быть меньше 0.29\*10\*\*(-38) и не может быть больше 1.7\*10\*\*38.

Примеры:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Правильная запись вещественных** **констант:** | |  | **Правильная запись вещественных** **констант:** | |
| 3.14159 |  |  | 1,234,567 | (запятые запрещены) |
| 621312. |  |  | 325Е-75 | (слишком мала) |
| -.00127 |  |  | 100 | (отсутствует десятичная точка) |
| +5.0E3 |  |  |  |  |

### 3.3.3. Константы с двойной точностью

Константа с двойной точностью занимает 8 байтов памяти и интерпретируется как вещественное число, имеющее до 16 значащих цифр.

Константа с двойной точностью может быть представлена так же, как и вещественная константа, в экспоненциальной форме с буквой "D" вместо "E" в качестве признака начала поля степени:

s1d1...dk.dk+1...Dnxs2pp

Число цифр , которые могут предшествовать указанию степени, не лимитировано, однако только 16 старших цифр являются значащими.

Отрицательной константе с двойной точностью обязательно должен пред-шествовать знак "-". Наличие знака "+" перед положительной константой не является обязательным.

Степень, следующая за буквой "D", не может быть опущена, но она может быть нулевой.

Абсолютная величина константы с двойной точностью не может быть меньше, чем 0.29\*10\*\*(-38) и не может быть больше 1.7\*10\*\*38.

Примеры:

1234567890D+5

+2.71828182846182D00

-72.5D-13

1D0

### 3.3.4. Шестнадцатеричные константы

Шестнадцатеричная константа - это буква z или x, за которой следует от 1 до 4 шестнадцатеричных цифр (0-9, A-F), заключенных в апострофы.

Примеры:

Z'1C'

X'AB1F'

Z'FFFF'

X'FFFF'

Шестнадцатеричная константа выравниваются в поле хранения вправо.

### 3.3.5. Логические константы

Логическая константа принимает логическое значение "истина" или "ложь". Следовательно, могут быть только две логические константы, они имеют следующий вид:

.true. или .false.

ограничивающие точки являются обязательной составной частью каждой логической константы.

### 3.3.6. Текстовые константы

Текстовая константа - это последовательность буквенно-цифровых символов, которой предшествует указатель числа символов и буква "H".

Текстовая константа имеет следующий вид:

nHc1c2...ci

где

|  |  |  |
| --- | --- | --- |
| n | - | константа, указывающая число символов в последовательности, включая пробелы и знаки табуляции, может иметь значение от 1 до 255. |
| ci | - | обозначает некоторый символ. Максимальное число символов равно 255. |

Текстовая константа запоминается как последовательность байтов по одному символу на байт. Текстовые константы не имеют определенного типа данных. Они принимают тип данных, указанных в операторе FORMAT, или устанавливаются по правилам, описанным в подпункте 3.3.6.2.

Примеры правильной записи текстовых констант:

14H Сегодн. Дата;

1HF

Примеры неправильной записи текстовых констант:

3HABCD (неверное число

2HB:1 символов)

#### 3.3.6.1. Литералы

Литерал - это альтернативная форма текстовой константы. Литерал имеет следующий вид:

'c1c2c3c4...ci'

где

каждое из c1,c2,c3...,ci обозначает некоторый символ.

Присутствие обоих ограничивающих апострофов является обязательным.

Максимальное число символов в литерале равно 255.

Внутри литерала символ "апостроф" обозначается двумя последовательными апострофами.

Примеры:

'Назначьте устройство ввода данных'

'Программа ''Таблица'' завершена'

#### 3.3.6.2. Правила, определяющие тип данных, представленных в виде текстовых констант

Если в выражении используется литерал или текстовая константа, то тип данного, который предполагается для этой константы, регулируется следующими правилами:

1. В операторе присваивания тип текстовой константы определяется типом данных операнда, которому присваивается эта константа.

Например:

|  |  |  |
| --- | --- | --- |
| Оператор | Тип данных | Длина константы |
| REL = 'ABCD' | REAL\*4 | 4 |
| IF(I.EQ.'XY') GO TO 3 | INTEGER\*2 | 2 |
| M = N - 'ABC' | INTEGER\*2 | 2 |
| X = 'Z' | REAL\*4 | 4 |

1. В контекстах, где используется определенный тип данных (обычно целое число), тип этих данных предполагается для константы. Например, при индексации элемента массива:

|  |  |  |
| --- | --- | --- |
| Y(IX)=Y('ABC')+3 | INTEGER\*2 | 2 |

1. Если константа используется в качестве фактического аргумента, то тип константы не определяется. Например:

|  |  |  |
| --- | --- | --- |
| CALL APAC ('ABCDEFGNI') |  | 9 |

1. Во всех других контекстах предполагается тип INTEGER\*2. Например:

|  |  |  |
| --- | --- | --- |
| IF ('AB') 1,2,3 | INTEGER\*2 | 2 |
| I= 'C'-'A' | INTEGER\*2 | 2 |
| J= .NOT. 'B' | INTEGER\*2 | 2 |

Если длина константы меньше длины, подразумеваемой типом данных, то константа справа дополняется пробелами. Если длина константы больше длины, подразумеваемой типом данных, то константа усекается справа.

## 3.4. Переменные

Переменная - это величина, обращение к которой осуществляется с помощью ее идентификатора. Идентификатор переменной может превышать по размеру шесть знаков. Однако используются только первые шесть знаков, которые должны быть уникальны среди имен переменных данной программной единицы. Значение переменной может быть изменено присваиванием ей нового значения. Переменные классифицируются по типу данных точно так же, как и константы. Тип переменной определяет, какой тип данных она представляет. Если переменной присваиваются данные другого типа, то они преобразуются к типу переменной. Тип переменной может быть установлен операторами описания типа, либо неявным заданием типа.

Если две или более переменных используют одну и ту же область памяти, то эти переменные эквивалентны друг другу. Эквивалентность может быть и частичной, когда часть области памяти (но не вся область), соответствующая одной переменной, является в то же время частью или всей областью памяти, соответствующей другой переменной. Эквивалентность устанавливается операторами "COMMON", "EQUIVALENCE" и при использовании фактических и формальных параметров в обращениях к подпрограммам. О переменной говорится, что она определена, если соответствующая ей область памяти содержит заданную величину того же типа, что и идентификатор. Переменная может быть определена еще до выполнения программы посредством оператора "DATA" или во время выполнения программы при помощи операторов присваивания или операторов ввода.

Если одной и той же области памяти соответствуют (или частично соответствуют) переменные различных типов, то определение значения одной переменной (например, присваиванием) приводит к тому, что значения других переменных перестают быть определенными.

### 3.4.1. Спецификация типа данных

Операторы явного описания типа определяют, что переменные будут представлять указанные типы данных.

Например:

REAL VAR1

DOUBLE PRECISION VAR2

Эти операторы указывают, что переменной "VAR1" должна соответствовать область памяти длиной в 4 байта, которая будет содержать действительные данные, и что переменной "VAR2" должна соответствовать область памяти в 8 байтов для размещения данных с двойной точностью.

Все переменные, идентификаторы которых начинаются с букв I, J, K, L, M, N, представляют переменные целого типа. Переменные, идентификаторы которых начинаются с любой другой буквы, представляют вещественные переменные.

Например:

|  |  |
| --- | --- |
| Вещественные переменные | Целые переменные |
| ALPHA | KOUNT |
| BETA | ITEM |
| TOTAN | NIOTAL |

### 3.4.2. Оператор задания типа данных по первой букве имени (IMPLICIT)

Оператор используется для переопределения неявного типа данных (определяемого по первой букве имени). Оператор имеет вид:

IMPLICIT T1(D1), T2(D2),...

где

|  |  |
| --- | --- |
| Т1,Т2,... | * тип переменной - INTEGER, REAL, LOGICAL, DOUBLE PRECISION, BYTE, INTEGER\*1, INTEGER\*2, REAL\*4, INTEGER\*4, REAL\*8; |
| D1,D2,... | * диапазон - последовательность букв, разделенных запятыми или знаком минус. |

Например:

IMPLICIT INTEGER (A,X-Z), REAL (B-E)

LOGICAL (F-H,W)

Все переменные, за исключением явно определенных, имена которых начинаются с букв А, X, Y, Z, будут типа INTEGER. Переменные, имена которых начинаются с букв B, C, D, E, будут типа REAL, а с букв F, G, N и W – типа LOGICAL.

Оператор IMPLICIT задает тип, который будет присваиваться переменным по умолчанию. Любой оператор IMPLICIT должен находиться в программе перед любым оператором спецификации типа.

Если оператор IMPLICIT встретится в программе после операторов явной спецификации типа или оператора DIMENSION, типы переменных, уже определенных, не будут переопределяться.

## 3.5. Массивы

Массив - это множество последовательно расположенных ячеек памяти, имеющих общее название - идентификатор (имя) массива. Обращение к отдельным областям памяти, которые называются элементами массива, осуществляется по идентификатору массива, к которому добавляется список индексов.

Массив имеет размерность в зависимости от количества индексов. Индексов может быть от одного до трех. Примером массива, имеющего размерность единица, является столбец чисел. Несколько столбцов чисел представляет собой двумерный массив. Для обращения к определенной величине из этого массива должны указываться номер строки и номер столбца, в которых находится эта величина. Для обращения к величине из трехмерного массива должны быть заданы номер строки, номер столбца и номер страницы.

Массивы описываются при помощи следующих операторов ФОРТРАН:

* операторы описания типа переменной;
* DIMENSION;
* COMMON.

Каждый из этих операторов содержит описатель массива и указывает идентификатор массива, его размерность и число элементов в каждом измерении.

### 3.5.1. Описатели массивов

Описатель массива определяет идентификатор массива внутри программного модуля и число позиций в каждом измерении.

Описатель массива имеет следующий формат:

A(D1[,D2]...)

где

|  |  |
| --- | --- |
| A | - идентификатор массива; |
| D1,D2,... | - индексы массива. |

Число индексов определяет размерность массива. Минимальная размерность массива равна единице, а максимальная - трем.

Значение индекса определяет число позиций в данном измерении. Например, если значение индекса равно 50, то данное измерение содержит 50 позиций. Индексы могут быть константами или переменными. Переменные индексы используются для определения массивов с переменными границами. Число элементов в массиве равно произведению чисел позиций во всех измерениях.

Внутри программной единицы идентификатор массива может встречаться только в одном описателе массива.

### 3.5.2. Списки индексов

Идентификатор массива может дополняться списком индексов. Список индексов представляет собой перечень индексных выражений, заключенных в круглые скобки, который определяет, к какому именно элементу массива выполняется обращение.

Список индексов имеет следующий вид:

(s1[,s2]...)

где

s1,s2,... - выражения индексов.

В любом обращении к элементу массива должно присутствовать по одному индексному выражению для каждого измерения. Например, чтобы обратиться к элементу первой строки третьего столбца второй страницы массива "DOS" используется следующая запись:

DOS(1,3,2)

Индексное выражение может быть записано в одной из следующих форм:

k, v, c\*v, v+k, c\*v+k

где

|  |  |
| --- | --- |
| k, c | - целые константы; |
| v | - целая двухбайтная переменная. |

### 3.5.3. Расположение массивов в памяти

Об измерениях массивов принято говорить как о "строках", "столбцах", "страницах" и т.д. Однако ФОРТРАН всегда запоминает массив в памяти в виде непрерывной последовательности значений. Многомерный массив хранится в памяти таким образом, что, чем левее индекс, тем быстрее он изменяется от ячейки к ячейке.

Пример расположения элементов одномерного массива ARC(6) в ячейках памяти с условной нумерацией:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер ячейки | Элемент массива |  | Номер ячейки | Элемент массива |  | Номер ячейки | Элемент массива |
| 1 | ARC(1) |  | 3 | ARC(3) |  | 5 | ARC(5) |
| 2 | ARC(2) |  | 4 | ARC(4) |  | 6 | ARC(6) |

Пример расположения элементов двумерного массива TAN(3,4) в ячейках памяти с условной нумерацией:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | TAN(1,1) |  | 5 | TAN(2,2) |  | 9 | TAN(3,3) |
| 2 | TAN(2,1) |  | 6 | TAN(3,2) |  | 10 | TAN(1,4) |
| 3 | TAN(3,1) |  | 7 | TAN(1,3) |  | 11 | TAN(2,4) |
| 4 | TAN(1,2) |  | 8 | TAN(2,3) |  | 12 | TAN(3,4) |

Пример расположения элементов трехмерного массива COS(3,3,3) в ячейках памяти с условной нумерацией:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | COS(1,1,1) |  | 10 | COS(1,1,2) |  | 19 | COS(1,1,3) |
| 2 | COS(2,1,1) |  | 11 | COS(2,1,2) |  | 20 | COS(2,1,3) |
| 3 | COS(3,1,1) |  | 12 | COS(3,1,2) |  | 21 | COS(3,1,3) |
| 4 | COS(1,2,1) |  | 13 | COS(1,2,2) |  | 22 | COS(1,2,3) |
| 5 | COS(2,2,1) |  | 14 | COS(2,2,2) |  | 23 | COS(2,2,3) |
| 6 | COS(3,2,1) |  | 15 | COS(3,2,2) |  | 24 | COS(3,2,3) |
| 7 | COS(1,3,1) |  | 16 | COS(1,3,2) |  | 25 | COS(1,3,3) |
| 8 | COS(2,3,1) |  | 17 | COS(2,3,2) |  | 26 | COS(2,3,3) |
| 9 | COS(3,3,1) |  | 18 | COS(3,3,2) |  | 27 | COS(3,3,3) |

### 3.5.4. Тип данных массива

Тип данных массива определяется таким же образом, как и тип данных переменной, т.е. либо неявно - по первой букве идентификатора, либо явным образом - с помощью оператора описания типа.

Все величины одного массива имеют один и тот же тип данных. Любое значение, присвоенное элементу массива, преобразуется к типу данных массива. Если, например, массив описан оператором DOUBLE PRECISION, компилятор резервирует для каждого элемента массива 8 байтов в памяти. Если любому из элементов этого массива присваивается значение какого-нибудь другого типа, оно преобразуется к типу с двойной точностью.

### 3.5.5. Обращения к массивам без использования списка индексов

Ниже перечислены типы операторов, в которых идентификатор массива может появляться без списка индексов для указания того, что должен быть использован или определен весь массив:

* операторы описания типа;
* COMMON;
* DATA;
* EQUIVALENCE;
* FUNCTION;
* SUBROUTINE;
* операторы ввода/вывода.

Идентификаторы массива без списка индексов могут также использоваться в качестве фактических параметров. Использование неиндексированных идентификаторов массивов во всех остальных типах операторов недопустимо.

### 3.5.6. Массивы с переменными границами

При составлении подпрограмм и подпрограмм-функций для обработки формальных массивов, у которых верхние значения индексов должны меняться от одного обращения к другому, удобно пользоваться так называемыми массивами с переменными границами. Конкретные значения границ индексов этих массивов устанавливаются при каждом обращении к подпрограмме или подпрограмме-функции. Для этого в операторе описания размерностей верхние границы индексов задаются целыми переменными (а не целыми константами). Эти целые переменные должны быть формальными параметрами. Соответствующие им фактические параметры должны иметь определенные значения в момент обращения к подпрограмме или к подпрограмме-функции. Тогда при замене формальных параметров фактическими параметрами (в процессе вызова) определяются фактические границы массивов.

Значения формальных параметров, используемых в описателях массивов с переменными границами, не должны изменяться внутри данной подпрограммы.

Для массива с переменными границами должно существовать описание этого массива в основной программе. Фактический размер массива, используемого в подпрограмме, не должен превышать размера, заданного в основной программе.

Функция SUM в следующем примере вычисляет сумму элементов двумерного массива. Следует обратить внимание на использование параметров "M" и "N" для управления шагом итераций.

FUNCTION SUM(A,M,N)

DIMENSION A(M,N)

SUM = 0.0

DO 10, I = 1,М

DO 10, J = 1,N

10 SUM = SUM + A(I,J)

RETURN

END

ниже приведены примеры обращения к "SUM":

DIMENSION A1(10,35), A2(3,56)

SUM1 = SUM(A1,10,35)

SUM2 = SUM(A2,3,56)

SUM3 = SUM(A1,10,10)

# 4. Выражения

Выражение может быть константой, переменной, либо комбинацией из этих компонент и одного или более знаков операций.

Знаки операций показывают, какие вычисления над основными компонентами должны быть выполнены для того, чтобы получить значение выражения.

В ФОРТРАН различаются три класса выражений:

* арифметические выражения;
* выражения отношения;
* логические выражения.

Результатом вычисления арифметического выражения является число. Выражения отношений и логические выражения могут быть истинными или ложными (т.е. их результатом являются логические величины).

## 4.1. Арифметические выражения

Арифметические выражения формируются из арифметических элементов и знаков арифметических операций. Значением такого выражения является число.

К арифметическим элементам относятся следующие:

* константа;
* переменная;
* элемент массива;
* арифметическое выражение;
* обращение к арифметической функции.

Знаки арифметических операций определяют, какие вычисления надо выполнить над значениями арифметических элементов. В результате выполнения этих операций получается числовое значение. В языке ФОРТРАН используются следующие арифметические операции:

|  |  |
| --- | --- |
| \*\* | (возведение в степень); |
| \* | (умножение); |
| / | (деление); |
| + | (сложение и одноместный плюс); |
| - | (вычитание и одноместный минус). |

Вышеперечисленные операции называются двуместными операциями, так как каждая из них используется в сочетании с двумя элементами. "+" и "-" могут, кроме того, использоваться как знаки одноместных операций, когда они помещаются непосредственно перед арифметическим элементом.

Любая арифметическая операция может быть использована в сочетании с любым допустимым арифметическим элементом, за исключением некоторых ограничений, указанных ниже.

Переменной или элементу массива должно быть присвоено значение перед тем, как их использовать в арифметическом выражении.

Таблица 2 иллюстрирует допустимые сочетания типов данных основания и показателя степени (экспоненты) для операции возведения в степень.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| Основание | Показатель степени | | |
| Целый | Вещественный | C двойной точностью |
| Целое | да | нет | нет |
| Вещественное | да | да | да |
| С двойной точностью | да | да | да |

Отрицательный элемент можно возводить только в целую степень. Элемент, имеющий нулевое значение, нельзя возводить в нулевую степень.

В любом допустимом степенном выражении результат имеет всегда тот же тип данных, что и основание, за исключением случая операции с вещественным основанием и показателем степени, имеющим двойную точность. В этом случае результат имеет двойную точность.

Арифметические выражения вычисляются в порядке убывания приоритетности операций: "\*\*" имеет первый приоритет, "\*" и "/" - второй приоритет, "+" и "-" - третий приоритет.

В тех случаях, когда имеется две или более операций с одинаковым приоритетом (такие, например, как "+" и "-"), они могут выполняться в любом порядке, выбираемом компилятором (для обеспечения оптимальности вычисления). Экспоненциальное выражение "A\*\*B\*\*C" вычисляется, как "A\*\*(B\*\*C)".

### 4.1.1. Применение скобок

Для изменения обычного порядка вычислений могут использоваться скобки. Выражение, заключенное в скобки, рассматривается как отдельный арифметический элемент. Это означает, что вначале вычисляется его значение, а затем это значение используется в вычислении выражения. Ниже приведены примеры использования скобок. Цифры под знаком операций указывают порядок, в котором выполняются вычисления.

4 + 3 \* 2 - 6 / 2 = 7

3 1 4 2

(4 + 3) \* 2 - 6 / 2 = 11

1 2 4 3

(4 + 3 \* 2 - 6) / 2 = 2

2 1 3 4

((4 + 3) \* 2 - 6) / 2 = 4

1 2 3 4

Вычисление выражений, находящихся в скобках, подчиняется обычному порядку приоритетов.

Несущественные скобки, такие как в выражении "4+(3\*2)-(6/2)" не оказывают никакого влияния на вычисление выражения.

Использование скобок для определения порядка вычислений часто оказывается важным в программах обработки чисел с высокой степенью точности.

Последовательности вычислений, алгебраически эквивалентные друг другу, могут быть неэквивалентными по степени точности вычислений (вследствие переносов и потерь младших разрядов).

### 4.1.2. Тип значения арифметического выражения

Если все элементы арифметического выражения имеют один и тот же тип, то результат выражения будет иметь этот же тип. Если в выражении имеются элементы различных типов, то вычисление этого выражения и тип результирующей величины зависят от ранга, соответствующего типу данных. Ранги типов данных следующие:

|  |  |  |
| --- | --- | --- |
| Логический | = | 1 (низкий); |
| Целый | = | 2; |
| Вещественный | = | 3; |
| С двойной точностью | = | 4 (высокий). |

Тип результата операции над двумя арифметическими элементами различного типа такой же, как и у элемента с наиболее высоким рангом. Тип значения выражения такой же, как и тип результата последней операции в вычислении данного выражения. Способ определения типа значения выражения заключается в следующем:

|  |  |  |
| --- | --- | --- |
| **Целые операции** | **-** | целые операции выполняются только над целыми элементами. Когда в арифметическом контексте встречаются восьмеричные константы и логические объекты, они рассматриваются как целые. В арифметике целых чисел любая дробная часть, которая может быть результатом деления, отбрасывается, а не округляется. Например, значение выражения  1/3 + 1/3 + 1/3  равно нулю, а не единице; |
| **Вещественные операции** | **-** | вещественные операции выполняются только над вещественными элементами либо над комбинацией из вещественных и целых элементов все присутствующие целые элементы преобразуются к вещественному типу добавлением к каждому из них дробной части, равной нулю. Однако необходимо заметить, что в операторе "Y = (I/J)\*X" над величинами "I" и "J" выполняется операция целого деления, а над результатом деления и переменной "X" выполняется вещественное умножение; |
| **Операции с двойной точностью** | **-** | любой вещественный или целый элемент в операции с двойной точностью преобразуется к типу с двойной точностью, причем имеющийся элемент становится старшей значащей частью данных с двойной точностью, а младшая значащая часть заполняется нулями. После этого выполняется вычисление с двойной точностью. |

**Примечание**

Преобразование вещественного элемента к типу двойной точности вовсе не увеличивает степень точности вычисления этого элемента. Например, вещественное число 0.3333 после преобразования станет числом 0.33330000, а не 0.33333333. Нужно также отметить, что вещественные элементы с двойной точностью являются только приблизительными представлениями фактических чисел. Результирующие значения вещественных выражений или выражений с двойной точностью в действительности точны лишь настолько, насколько позволяет степень точности этого типа данных.

# 4.2. Выражения отношений

Выражение отношения состоит из двух арифметических выражений, разделенных операцией отношения. Значение данного выражения есть либо "истина", либо "ложь" в зависимости от того, выполняется или нет указанное отношение.

Операции отношения следующие:

|  |  |
| --- | --- |
| **Знак операции** | **Отношение** |
| .LT. | меньше |
| .LE. | меньше или равно |
| .EQ. | равно |
| .NE. | не равно |
| .GT. | больше |
| .GE. | больше или равно |

Ограничивающие точки - это часть знака операции отношения и их наличие в тексте обязательно.

В выражении отношения прежде всего вычисляются значения арифметических выражений. Затем эти величины сравниваются для проверки того, выполняется ли отношение, заданное операцией. Например, выражение

APP + PEA .GT. PAR + ORA

устанавливает отношение следующего вида: "сумма вещественных переменных "APP" и "PEA" больше суммы вещественных переменных "PAR" и "ORA". Если это соотношение выполняется, значение выражения равно "истина", а если нет - то "ложь".

Все операции отношения имеют одинаковый приоритет. Следовательно, если в выражении встречаются два или более выражений отношения, операции отношения выполняются слева направо. Арифметические операции имеют более высокий приоритет, чем операции отношения.

Для изменения порядка вычислений арифметических выражений в выражении отношения скобки используются точно так же, как и в любом другом арифметическом выражении. Однако поскольку арифметические операции выполняются до операции отношения, то заключать в скобки арифметическое выражение целиком не обязательно.

Если в выражении отношения сравниваются два выражения разных типов данных, перед выполнением сравнения значение выражения, имеющего тип более низкого ранга, преобразуется к типу более высокого ранга.

# 4.3. Логические выражения

Логическое выражение может быть отдельным логическим элементом или комбинацией логических элементов и логических операций. Логическое выражение имеет значение "истина" или "ложь".

Логический элемент может быть одним из следующих:

* целая или логическая константа;
* целая или логическая переменная;
* элемент целого или логического массива;
* выражение отношения;
* логическое выражение, заключенное в скобки;
* обращение к целой или логической функции.

Логические операции следующие:

|  |  |  |
| --- | --- | --- |
| .AND. | - | логическая конъюнкция (логическое "и"). Выражение А.АND.В истинно тогда и только тогда, когда и "А", и "В" истинны; |
| .OR. | - | логическая дизъюнкция (логическое "или"). Например, выражение "A.OR.B" истинно тогда и только тогда, когда истинно либо "А", либо "В", либо оба истинны; |
| .XOR. | - | исключающее "или". Например, "A.XOR.B". Выражение истинно, если "А" истинно, а "В" ложно, либо, если "В" истинно, а "А" ложно. Выражение ложно, если оба элемента имеют одинаковое значение; |
| .NOT. | - | логическое отрицание (логическое "не"). Например, ".NOT.A". Выражение истинно тогда и только тогда, когда ложно "А". |

Если выполняется логическая операция над логическими элементами, результирующая величина будет логического типа. Если логическая операция выполняется над целыми элементами, то ее выполнение происходит поразрядно над соответствующими разрядами внутреннего (двоичного) представления целых элементов. Результирующая величина относится к типу "целый". Если в логической операции есть операнды как целого, так и логического типа, то в начале логическая величина преобразуется к целой величине, а затем операция выполняется так же, как и в случае двух целых элементов результат имеет целый тип.

Вычисление логического выражения происходит согласно старшинству его операций. Некоторые логические выражения могут быть вычислены без вычисления всех подвыражений. Например, если "А" равно ".FALSE.", то выражение "A.AND.(F(X,Y).GT.2.0).AND.B" равно ".FALSE.". Значение этого выражения может быть определено только проверкой "А" без вычисления F(X,Y).

При таком методе вычислений не требуется вызов подпрограммы функции "F", и не будут иметь места действия, связанные с вызовом.

Ниже приведен перечень всех операций, которые могут встречаться в логическом выражении, и порядок, в котором они выполняются.

|  |  |
| --- | --- |
| **Операция** | **Выполняется** |
| \*\* | первой |
| \*,/ | второй |
| +,- | третьей |
| отношения | четвертой |
| .NOT. | пятой |
| .AND. | шестой |
| .OR.,.XOR. | седьмой |

Операции с одинаковым приоритетом выполняются слева направо.

Пример логического выражения, содержащего выражения отношений:

A\*B + C\*ABC.EQ.X\*Y + DM\*ZZ.AND. .NOT.K\*B.GT.TT

Это выражение вычисляется в следующем порядке:

(((A\*B)+(C\*ABC)).EQ.((X\*Y)+(DM\*ZZ))).AND.(.NOT.((K\*B).GT.TT))

Как и в арифметических выражениях для изменения обычного порядка выполнения операций могут использоваться скобки.

Два знака логических операций не могут располагаться подряд, за исключением того случая, когда вторая операция есть ".NOT.".

# 5. Операторы языка ФОРТРАН

## 5.1. Операторы присваивания

Операторы присваивания определяют значение переменной или элемента массива путем вычисления некоторого выражения и присваивания результирующего значения переменной или элементу массива.

Существует три типа оператора присваивания:

* арифметический;
* логический;
* оператор "ASSIGN".

### 5.1.1. Оператор присваивания

Оператор присваивает значение выражения, расположенного справа от знака равенства, переменной или элементу массива, находящемуся слева от знака равенства. Если у переменной до этого было какое-то значение, оно теряется.

Оператор присваивания имеет следующий вид:

W = A

где

|  |  |
| --- | --- |
| W | - идентификатор переменной или массива; |
| A | - выражение. |

Знак равенства здесь не означает "равно", как обычно в математике. Он означает "заменяется". Таким образом, оператор I=I+1 означает: "текущее значение целой переменной "I" заменяется суммой этого текущего значения и целой константы "1".

Идентификатор слева от знака равенства может быть предварительно не определен, но всем идентификаторам справа от знака равенства, к которым имеются обращения в выражении, предварительно должны быть присвоены значения.

Результатом выражения должно быть значение, удовлетворяющее требованиям типа переменной или элемента массива, которым оно присваивается.

Например, вещественное выражение, результатом которого является величина, превышающая 32767, недопустимо, если объектом слева от знака равенства является переменная типа "INTEGER\*2".

Если тип переменной или элемента массива, находящегося слева от знака равенства, точно такой же, как у выражения справа, оператор непосредственно выполняет присваивание. Если типы различаются, то перед присваиванием значение выражения преобразуются к типу объекта, находящегося слева от знака равенства.

Правила преобразования для операторов присваивания приведены в таблице 3.

Таблица 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип переменной | Тип выражения (а) | | | |
| INTEGER | REAL | LOGICAL | DOUBLE |
| INTEGER | y | ya | yb | ya |
| REAL | yc | y | yc | ye |
| LOGICAL | yd | ya | y | ya |
| DOUBLE | yc | y | yc | y |

где

Буква y обозначает пересылку результата без преобразования, а буквы a, b, c, d и e указывают на совершаемые перед пересылкой преобразования.

|  |  |  |
| --- | --- | --- |
| а | - | значение действительного выражения преобразуется в целое; |
| b | - | знак распространяется по второму байту; |
| c | - | целое преобразуется в действительное; |
| d | - | значение, присвоенное переменной, представляет собой младшую часть целого выражения (используется младший байт, невзирая на знак); |
| e | - | переменной назначается округленное значение действительного выражения. |

Примеры записи допустимых операторов:

BETA = -1./(2.\*X) + A\*A/(4.\*(X\*X))

P1 = 3.14159

SUM = SUM + 1

PAGEND = .FALSE.

PULL = LINE.LE.132.AND..NOT.ROBIN

BILL = A.GT.B.AND.A.GT.C.AND.A

Примеры записи недопустимых операторов:

|  |  |
| --- | --- |
| 3.14 = A - B | (объект в левой части должен быть переменной или элементом массива); |
| -J = 1\*\*4 | (объект в левой части не должен иметь знака); |
| ALPHA = ((X+6)\*B\*B/(X-Y) | (число открывающихся скобок не равно числу закрывающихся скобок). |

### 5.1.2. Оператор "ASSIGN"

Оператор "ASSIGN" используется для присвоения целой переменной значения метки оператора. Затем эта переменная может быть использована для указания места передачи управления в последующем операторе перехода по предписанию "GO TO".

Оператор "ASSIGN" имеет следующий вид:

ASSIGN S TO W

где

|  |  |  |
| --- | --- | --- |
| S | - | метка выполняемого оператора из того же программного модуля, что и оператор "ASSIGN" (она не может быть меткой оператора "FORMAT"); |
| W | - | целая переменная. |

Оператор "ASSIGN" присваивает переменной номер оператора таким же образом, как это делается в арифметическом операторе присваивания. Переменная, получившая значение метки, не может быть использована как переменная, имеющая числовое значение, до тех пор, пока она его не получит в арифметическом операторе присваивания или в операторе ввода.

Например, оператор

ASSIGN 100 ТО NUMBER

ставит в соответствие переменной "NUMBER" метку оператора 100. После этого арифметические операции над этой переменной, такие, как в операторе

NUMBER = NUMBER + 1

будут недопустимыми, так как метка оператора не может быть изменена.

Оператор

NUMBER = 10

присваивает переменной "NUMBER" значение 10 и возвращает эту переменную в ее исходное состояние в качестве целой переменной. Теперь эта переменная не может быть использована в операторе перехода по предписанию.

Примеры использования оператора "ASSIGN"

ASSIGN 10 TO NSTART

ASSIGN 9999 TO NSTOP

ASSIGN 350 TO ERAS (ERAS должна быть определена как целая переменная).

## 5.2. Операторы управления

Операторы, как правило, выполняются в том порядке, в каком они записаны. Однако иногда необходимо прервать обычный порядок выполнения программы передачей управления в другую часть программы или подпрограмму. Передача управления из заданной точки программы может быть выполнена всякий раз, когда эта точка встретится в процессе выполнения программы. Возможен также вариант, когда передача управления будет каждый раз осуществляться в одну и ту же точку программы.

Передача управления, как внутри одной программной единицы, так и в другой программной единице, выполняется при помощи операторов управления.

Кроме того, эти операторы управляют итерационной обработкой, приостановкой выполнения программы и прекращением выполнения программы.

Имеются следующие операторы управления:

|  |  |  |
| --- | --- | --- |
| GO TO | CONTINUE | PAUSE |
| IF | CALL | STOP |
| DO | RETURN | END |

### 5.2.1. Оператор "GO TO"

Оператор "GO TO" передает управление внутри программной единицы либо каждый раз на один и тот же оператор, либо на один из операторов, принадлежащих некоторому множеству, в зависимости от значения некоторого выражения.

Имеются три типа операторов "GO TO":

* оператор безусловного перехода;
* вычисляемый оператор перехода;
* оператор перехода по предписанию.

#### 5.2.1.1. Оператор безусловного перехода

Оператор безусловного перехода передает управление одному и тому же оператору.

Оператор безусловного перехода имеет следующий вид:

GO TO S

где

|  |  |  |
| --- | --- | --- |
| S | - | метка некоторого выполняемого оператора, находящегося в той же самой программной единице, что и данный оператор "GO TO". |

Примеры:

GO TO 7740

GO TO 99999

GO TO 16.5 (недопустимо, неправильно записана метка оператора)

#### 5.2.1.2. Вычисляемый оператор перехода "GO TO"

Вычисляемый оператор перехода передает управление оператору, определяемому по значению выражения, входящего в оператор "GO TO".

Вычисляемый оператор перехода "GO TO" имеет следующий вид:

GO TO (K1,K2,...,Kn),M

где

|  |  |  |
| --- | --- | --- |
| K1,K2,...,Kn | - | метки выполняемых операторов. Список меток называется списком переходов; |
| M | - | арифметическое выражение, значением которого является целое число, лежащее внутри интервала от 1 до n (n - это число меток операторов в списке переходов). |

Вычисляемый оператор перехода вычисляет выражение "M". После этого оператор "GO TO" передает управление метке оператора, порядковый номер которой в списке переходов равен полученному значению выражения. Например, если список имеет вид (30,20,50,40), а значение выражения "M" равно "2", то оператор "GO TO" передает управление на оператор с меткой "20".

Если значение выражения меньше единицы или больше, чем число меток в списке переходов, управление передается первому выполняемому оператору, следующему за вычисляемым оператором перехода "GO TO".

Примеры:

GO TO (17,25,36),KEND

GO TO (340,360,500,700), SITU(J,K)+1

#### 5.2.1.3. Оператор перехода по предписанию "GO TO"

Оператор перехода по предписанию имеет следующий вид:

GO TO U[[,](К1,К2,...Кn)]

где

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная; |
| К1,К2,...Кn | - | метки выполняемых операторов. |

Оператор перехода по предписанию передает управление оператору, метка которого была последней присвоена переменной "U" оператором "ASSIGN".

Переменная "U" должна быть переменной целого типа и ей должно быть присвоено значение метки некоторого оператора с помощью оператора "ASSIGN" (а не арифметического оператора присваивания) до выполнения оператора "GO TO".

Оператор перехода по предписанию и соответствующий оператор (или операторы) "ASSIGN" должны быть расположены внутри одной и той же программной единицы. Операторы, которым передается управление, также должны быть расположены в той же программной единице.

Примеры:

GO TO IGО

GO TO INDEX,(300,400,1000,700)

### 5.2.2. Операторы "IF"

Оператор "IF" вызывает условную передачу управления или условное выполнение некоторого оператора. Имеется два типа операторов "IF":

* арифметический оператор условного перехода "IF";
* логический условный оператор "IF".

В обоих случаях решение о передаче управления или о выполнении оператора основывается на результате вычисления выражения, входящего в состав оператора "IF".

#### 5.2.2.1. Арифметический оператор условного перехода "IF"

Этот оператор имеет следующий вид:

IF (M) K1,K2,K3

где

|  |  |  |
| --- | --- | --- |
| M | - | арифметическое выражение; |
| K1,K2,K3 | - | метки выполняемых операторов, расположенных в той же самой программной единице. |

В записи оператора должны присутствовать все три метки. Эти метки необязательно должны относиться к трем разным операторам. Если необходимо, две метки могут относиться к одному оператору.

При выполнении арифметического оператора условного перехода прежде всего вычисляется выражение в скобках, а затем управление передается оператору с меткой "K1", "K2" или "K3", если значение выражения "M" соответственно меньше, равно или больше нуля.

Примеры:

IF (GAMMA-CHI) 50,50,100

Этот оператор передает управление оператору 50, если вещественная переменная "GAMMA" меньше или равна вещественной переменной "CHI". Управление передается оператору 100 только в том случае, когда "GAMMA" больше, чем "CHI".

IF (NUMBER/2\*2-NUMBER) 20,40,20

Этот оператор передает управление оператору 40, если значение целой переменной "NUMBER" четно, и оператору 20, если оно нечетно. В этом случае третья метка оператора обязана присутствовать, хотя она никогда не используется. Значение выражения будет либо нулем, либо отрицательным числом.

#### 5.2.2.2. Логический условный оператор "IF"

Логический условный оператор "IF" имеет следующий вид:

IF (E) S

где

|  |  |  |
| --- | --- | --- |
| E | - | логическое выражение; |
| S | - | оператор языка ФОРТРАН, за исключением оператора "DO" или другого логического условного оператора "IF". |

При выполнении логического условного оператора прежде всего вычисляется значение логического выражения. Если это значение выражения равно ".TRUE.", то происходит выполнение оператора "S". Если значение выражения равно ".FALSE.", управление передается первому выполняемому оператору, следующему за оператором "IF" (при этом оператор "S" не выполняется).

Примеры:

IF (J.GT.4.OR.J.LT.1) GO TO 200

IF (REC(J,K).NE.HOLD) REC(J,K)=REC(J,К)\*2

IF (ENDRUN) CALL EXIT

### 5.2.3. Оператор "DO"

Оператор "DO" позволяет повторить определенное число раз выполнение группы операторов, находящихся в области его действия.

Оператор "DO" имеет следующий вид:

DO S I=М1,М2[,М3]

где

|  |  |  |
| --- | --- | --- |
| S | - | метка последнего выполняемого оператора. Этот оператор должен физически располагаться после оператора "DO" в той же самой программной единице; |
| I | - | целая переменная типа INTEGER\*2 или INTEGER\*1; |
| М1,М2,М3 | - | целые беззнаковые константы или положительные целые переменные типа INTEGER\*2 или INTEGER\*1. |

Переменная "I" называется переменной цикла, М1,М2,М3 – параметрами цикла. Причем М1 - начальное значение переменной цикла, М2 – конечное значение, а М3 - шаг цикла. Шаг может быть опущен, при этом он считается равным единице.

Использование оператора "DO" определяется следующими правилами:

1. Оператор "DO" и первая запятая должны располагаться на первой строке;
2. Последний оператор, включенный в цикл, указывается с помощью метки "S". Все операторы, начиная с оператора, непосредственно следующего за "DO", вплоть до последнего с меткой "S" (и включая его), образуют область действия оператора "DO" (область действия цикла). Последний оператор области действия цикла должен быть выполняемым оператором;
3. Последний оператор области действия "DO" не может быть оператором "GO TO", "RETURN", "STOP", "PAUSE", арифметическим "IF" и другим оператором "DO";
4. Переменная цикла "I" должна быть положительной и не может использоваться ни одним оператором области действия цикла;
5. Если М1,М2,М3 - переменные типа INTEGER\*1 или константы, оператор "DO" выполняется быстрее и занимает меньше места, однако число итераций при этом ограничено 127;
6. При первом выполнении группы операторов в области действия "DO" переменная цикла "I" равна М1, при в втором I=М1+М3, при третьем I=М1+2\*М3 и т.д. до тех пор, пока I<М2.

Логика выполнения оператора цикла "DO" такова:

│

┌──┴───┐

│ I=М1 │

└──┬───┘

│

┌───────────────>┤

│ ┌────────┴──────────┐

│ │ Оператор следую- │

│ │ щий за DO │

│ └────────┬──────────┘

│ │

│ . . . . . . . . . . .

│ │

│ ┌─────┴──────┐

│ │ Оператор S │

│ └─────┬──────┘

│ │

│ ┌─────┴──────┐

│ │ I = I + М3 │

│ └─────┬──────┘

│ │

│ │

│ Нет ┌────┴────┐

└───────────┤ I > М2 │

└────┬────┘

│Да

│ Выход из цикла

Таким образом, число выполнений группы операторов из области действия оператора цикла, называемое числом итераций, определяется по формуле:

[(M2 - M1)/M3] + 1

Здесь величина в квадратных скобках представляет наибольшее абсолютное целое значение выражения.

Выполнение цикла может быть прекращено с помощью оператора, находящегося в области действия цикла, который передает управление за пределы цикла. При этом переменная цикла сохраняет значение, равное ее текущему значению.

Если последний оператор цикла входит и в другие циклы, то по окончании внутреннего цикла управление передается на самый внутренний из оставшихся внешних циклов. Если этот последний оператор не входит ни в какие другие циклы, либо если данный цикл был самым внешним, управление передается первому выполняемому оператору, следующему за последним оператором цикла.

Значения переменной цикла, конечного параметра и шага не могут изменяться внутри области действия оператора "DO". Внутри этой области допускается использование переменной цикла как обычной переменной.

В области действия оператора "DO" могут находиться другие операторы "DO" при условии, что они удовлетворяют определенным требованиям.

Возможна передача управления из тела цикла во вне, но нельзя передать управление в тело цикла из какого-либо другого места программы, если этот цикл еще не начат, или он уже закончен.

#### 5.2.3.1. Вложенность операторов "DO"

Область действия оператора "DO" может содержать один или более других операторов "DO". Область действия внутреннего вложенного цикла должна полностью лежать внутри области действия охватывающего его цикла (или самого внутреннего из охватывающих). Такая совокупность операторов "DO" и их областей действия называется полностью вложенной. Циклы в цикле могут иметь один и тот же последний оператор.

|  |  |  |
| --- | --- | --- |
| Пример допустимой вложенности операторов "DO": |  | Пример недопустимой вложенности операторов "DO": |
| ┌─ DO 90 K=1,10  │ ...  │ ...  │  │┌─ DO 35 L=2,50,2  ││ ...  ││ ...  │└─ 35 CONTINUE  │ ...  │ ...  │  │┌─ DO 90 М=1,20  ││ ...  ││ ...  │└─ 90 CONTINUE  └─ |  | ┌─ DO 80 K=1,10  │ ...  │ ...  │  ┌─┤ DO 25 I=1,20  │ │ ...  │ │ ...  │ └─ 80 CONTINUE  │ ...  │ ...  │ ┌─ DO 30 M=1,15  │ │ ...  │ │ ...  └─┤ 25 CONTINUE  │ ...  │ ...  └─ 30 CONTINUE |

#### 5.2.3.2. Передача управления во вложенных "DO"

Внутри вложенной структуры "DO" управление может передаваться из внутреннего цикла во внешний. Передачи управления из внешнего цикла во внутренний запрещены.

Если два или более вложенных циклов имеют один и тот же последний оператор, управление этому оператору может быть передано только из области действия самого внутреннего цикла. Любая другая передача управления этому оператору будет рассматриваться как передача управления из внешнего цикла во внутренний, так как этот общий оператор является частью области действия самого внутреннего цикла.

#### 5.2.3.3. Расширенная область действия

Об операторе цикла "DO" говорят, что он имеет расширенную область действия, если область цикла содержит оператор управления, который передает управление за пределы области цикла, и, если после выполнения одного или нескольких операторов, другой оператор управления возвращает управление назад в цикл. Таким образом, область цикла будет расширена за счет включения всех выполняемых операторов, расположенных между оператором, который получил управления из цикла, и оператором, который вернул управление в цикл.

|  |  |  |
| --- | --- | --- |
| Пример допустимых передач управления с расширенной областью действия: |  | Пример недопустимых передач управления: |
| ┌── DO 35 К=1,10  │ ...  │┌─ DO 15 L=2,20  ││ ...  ││ GO TO 20 ──┐  ││ ... │  │└─ 15 CONTINUE │  │ ... │  │ 20 A=B+C <─────┘  │ ...  │┌─ DO 35 M=1,15  ││ ...  ││ GO TO 50 ─┐  ││ ... │  ││ 30 X=A\*D │<──┐  └┴─ 35 CONTINUE │ │  ... │ │  ... │ │  50 D=E/F <───┘ │  Расши- ... │  ренная ... │  область ... │  GO TO 30 ────────┘ |  | GO TO 70 ──┐  ... │  ┌── DO 50 K=1,10 │  │ ... │  │ 70 A=B+C <─────┘  │ ...  │┌─ DO 35 L=2,20  ││ ...  ││ 30 D=E/F <──┐  ││ ... │  │└─ 35 CONTINUE │  │ ... │  │ GO TO 40 ──┐│  │ ... ││  │┌── DO 45 M=1,15 ││  ││ ... ││  ││ 40 X=A\*D <───┘│  ││ ... │  │└── 45 CONTINUE │  │ ... │  └─── 50 CONTINUE │  ... │  GO TO 30 ───┘ |

Использование расширенной области действия оператора "DO" подчиняется следующим правилам:

* Передача управления в область действия оператора "DO" разрешается только тогда, когда эта передача управления производится из расширенной области действия этого оператора "DO";
* В расширенной области действия оператора "DO" запрещается изменение управляющей переменной, шага, а также начального и конечного параметров.

### 5.2.4. Оператор "CONTINUE"

Оператор "CONTINUE" передает управление следующему выполняемому оператору. Он главным образом используется как последний оператор в области действия оператора "DO" в тех случаях, когда последний оператор не может иметь метки завершения цикла (является либо оператором "GO TO", либо арифметическим оператором условного перехода "IF", либо другим запрещенным оператором управления).

Оператор "CONTINUE" имеет следующий вид:

CONTINUE

### 5.2.5. Оператор "CALL"

Оператор "CALL" вызывает выполнение подпрограммы и определяет список фактических параметров, передаваемых в подпрограмму.

Оператор "CALL" имеет следующий вид:

CALL S[(P1, Р2,...,Рn)]

где

|  |  |  |
| --- | --- | --- |
| S | - | имя подпрограммы, написанной на языке ФОРТРАН или на языке макроассемблер, или одной из системных подпрограмм; |
| P1, Р2,...,Рn | - | список фактических параметров, соответствующий списку формальных параметров в вышеперечисленных подпрограммах. |

Оператор "CALL" устанавливает соответствие между значениями списка фактических параметров (если список присутствует) и формальными параметрами в подпрограмме и затем передает управление первому выполняемому оператору подпрограммы. Описание подпрограмм дается в разделе 6.

Фактические параметры, указанные в операторе "CALL", должны по своему количеству, порядку следования и типу данных соответствовать формальным параметрам в определении подпрограммы. В качестве фактических параметров допускается использование переменных, массивов, элементов массива, констант, выражений и буквенно-цифровых литералов или идентификаторов подпрограмм, если эти идентификаторы были описаны в операторе "EXTERNAL".

Идентификатор массива без списка индексов, расположенный в списке параметров, относится ко всему массиву в целом.

Примеры:

CALL PROCENT(BPL,3.14159+X,Y,LIMIT,R(LT+2))

CALL CALUE(A,B)

CALL EXIT

### 5.2.6. Оператор "RETURN"

Оператор "RETURN" используется для возврата управления в вызывающую подпрограмму. Оператор "RETURN" имеет следующий вид:

RETURN

При выполнении оператора "RETURN" в подпрограмме-функции он передает управление оператору, содержащему обращение к функции. При выполнении в подпрограмме оператора "RETURN" управление передается первому выполняемому оператору, следующему непосредственно за оператором "CALL", который вызвал выполнение подпрограммы.

Оператор "RETURN" не должен присутствовать в программе.

Пример

SUBROUTINE CART(N,A,DATA,P,K)

DIMENSION DATA(N),P(N)

IF(N.LT.10) GO TO 200

DATA(K+2)=N-(N/10)\*N

N=N/10

DATA(K+1)=N

P(K+2)=A(DATA(K+2)+1)

P(K+1)=A(DATA(K+1)+1)

RETURN

200 P(K+2)=A(N+1)

RETURN

END

### 5.2.7. Оператор "PAUSE"

Оператор "PAUSE" временно приостанавливает выполнение программы, позволяя оператору выполнить какие-либо необходимые действия.

Оператор "PAUSE" имеет следующий вид:

PAUSE [S]

где

S - строка от 1 до 6 символов.

Оператор "PAUSE" выводит строку символов на консоль и приостанавливает выполнение программы, ожидая ввода символа с консоли. Если введенный символ буква "T", то программа завершается. Любой другой символ возобновляет выполнение программы с первого выполняемого оператора, следующего за оператором "PAUSE".

Примеры:

PAUSE 1982

PAUSE CP/M

### 5.2.8. Оператор "STOP"

Оператор "STOP" используется для прекращения выполнения программы.

Оператор "STOP" имеет следующий вид:

STOP [S]

где

S - строка от 1 до 6 символов.

Когда оператор "STOP" встречается во время выполнения программы, символы, составляющие строку s, выводятся на консоль, и выполнение программы завершается. Таким образом, оператор "STOP" является логическим концом программы.

### 5.2.9. Оператор "END"

Оператор "END" отмечает конец программной единицы. Оператор "END" должен быть последним в исходном тексте программного модуля.

Оператор "END" имеет следующий вид:

END

## 5.3. Ввод/вывод данных

Ввод данных в программе, написанной на языке ФОРТРАН, выполняется с помощью оператора "READ". Вывод данных выполняется оператором "WRITE". Эти операторы используются в сочетании со спецификациями формата, которые управляют редактированием и преобразованием данных из внутреннего представления в символьную форму и наоборот.

Каждый оператор "READ" или "WRITE" содержит указание логического устройства, с которого или на которое должна быть осуществлена передача данных. Логическое устройство может соответствовать либо внешнему устройству, либо файлу.

Операторы "READ" и "WRITE" могут принадлежать к одной из следующих категорий:

|  |  |  |
| --- | --- | --- |
| **Неформатный последовательный ввод/вывод** | - | при выполнении операторов неформатного последовательного ввода/вывода осуществляется передача двоичных данных без преобразования; |
| **Форматный последовательный ввод/вывод** | - | при выполнении операторов форматного последовательного ввода/вывода осуществляется передача символьных данных, причем для преобразования данных в символы при выводе и во внутреннюю форму при вводе используются спецификации формата; |
| **Неформатный ввод/вывод с прямым доступом** | - | операторы неформатного ввода/вывода с прямым доступом служат для передачи данных без преобразования на устройствах прямого доступа. |

Оператор "READ" или "WRITE" любого вида может передать управление другому оператору, если при его выполнении будет обнаружен конец файла или ошибочное состояние.

Вспомогательные операторы ввода/вывода "REWIND" и "BACKSPASE" не выполняют передачи данных, а служат для управления положением указателей файла. Оператор "ENDFILE" записывает специальную запись, которая будет являться признаком конца файла (и передачи управления "END="). Операторы "ENCODE" и "DECODE" выполняют преобразование данных в оперативной памяти.

### 5.3.1. Номера логических устройств ввода/вывода

ФОРТРАН CP/M 2.2 может пользоваться всеми внешними устройствами, поддерживаемыми операционной системой. Операторы ввода/вывода обращаются к устройствам ввода/вывода с помощью номеров логических устройств. Номер логического устройства представляет собой целую константу или переменную с положительным значением. Соответствие между логическими и физическими устройствами определяется таблицей $LUNTB. Она содержит адрес драйвера (программы ввода/вывода) для каждого логического номера устройства: первый байт этой таблицы равен числу номеров логических устройств плюс 1. Эта таблица стандартно имеет следующий вид:

EXTRN TTYDRV, LPTDRV, DSKDRV

ENTRY $LUNTB

$LUNTB: DB (LUNTBE - $LUNTB)/2 + 1

DW TTYDRW ;LUN=1

DW LPTDRW ;LUN=2

DW ТТYDRW ;LUN=3

DW TTYDRW ;LUN=4

DW TTYDRW ;LUN=5

DW DSKDRW ;LUN=6

DW DSKDRW ;LUN=7

DW DSKDRW ;LUN=8

DW DSKDRW ;LUN=9

DW DSKDRW ;LUN=10

LUNTBE: EQU $

END

то есть стандартно-логические устройства с номерами 1, 3, 4, 5 назначены на логическое устройство CON (обычно консоль), с номером 2 - на логическое устройство LST (обычно печатающее устройство). Остальные логические номера (6-10) назначены на диск и связаны с файлами FORT06.DAT, FORT07.DAT,...,FORT10.DAT. Логические устройства с номерами 1-5, так же как и другие, могут быть назначены любому файлу. Для этого можно воспользоваться подпрограммой OPEN. В обращении к подпрограмме задаются номер логического устройства, имя файла и номер диска. При выполнении подпрограммы OPEN с несуществующим файлом создается файл с нулевой длиной. Если же файл уже существует, то при первом выводе в него он уничтожается и начинает создаваться новый файл со старым именем и новым содержимым. Форма обращения к подпрограмме OPEN такова:

CALL OPEN(LUN,FILENAME,DRIVE)

где

|  |  |  |
| --- | --- | --- |
| LUN | - | логический номер устройства (1-10); |
| FILENAME | - | полное имя файла, задаваемое в виде литерала или текстовой константы, либо имя массива, содержащего имя файла; |
| DRIVE | - | номер дискового устройства: 1 - диск A, 2 - диск B и т.д. Если номер равен 0, то будет использоваться активизированный диск. |

Полное имя файла должно иметь длину ровно 11 символов и не содержать точки, разделяющей имя файла и его тип. Имя файла дополняется справа пробелами до 8 символов. Например, файл ABC.DAT должен быть открыт так:

CALL OPEN(LUN,'ABC DAT', DSK)

операторы "REWIND LUN" и "ENDFILE LUN" позволяют управлять дисковыми файлами. Оператор "ENDFILE LUN" закрывает файл, связанный с LUN. Оператор "REWIND LUN" закрывает соответствующий файл, а затем открывает его снова.

Пользователь может заменять таблицу $LUNTB и драйверы. При этом надо иметь в виду следующее. Каждый драйвер должен начинаться с таблицы адресов семи подпрограмм, соответствующих выполнению следующих операций на устройстве:

1. форматный ввод;
2. форматный вывод;
3. неформатный ввод;
4. неформатный вывод;
5. выполнение оператора "REWIND";
6. выполнение оператора "BACKSPACE";
7. выполнена "ENDFILE".

Глобальная метка $BF содержит адрес буфера для чтения и записи. При выполнении вывода число выводимых байтов находится по адресу $BL. При выполнении ввода число вводимых байтов должно быть помещено по адресу $BL.

Все операции ввода/вывода перед возвратом в вызывающую подпрограмму должны устанавливать коды условий по следующим правилам:

a) carry=1 z=(0,1) ошибка ввода/вывода

b) carry=0 z=0 встретился конец файла

c) carry=0 z=1 нормальный возврат

Система ввода/вывода периода выполнения проверяет коды условия при обращении к драйверу. При обнаружении ненормального завершения управление передается по адресу, указанному в операторах END= или ERR=. Если эти адреса не заданы, программа завершается аварийно.

### 5.3.2. Форматный последовательный ввод/вывод

Операторы форматного ввода/вывода используются в сочетании с операторами "FORMAT" (или со спецификациями формата, хранящимися в массивах) для преобразования и редактирования выводимых данных, а при вводе для преобразования данных из внешнего формата во внутренний.

#### 5.3.2.1. Оператор форматного последовательного ввода "READ"

Оператор форматного последовательного ввода "READ" вводит данные с указанного логического устройства, преобразуя их во внутренний формат согласно спецификации формата. Полученные значения присваиваются элементам списка ввода.

Оператор форматного последовательного ввода "READ" имеет вид:

READ (U,F[,END=S][,ERR=S])[K]

где

U - номер логического устройства;

F - метка соответствующего оператора "FORMAT";

S - метка выполняемого оператора;

K - список ввода.

Если оператор "FORMAT", соответствующий оператору форматного ввода, содержит текстовую константу или буквенно-цифровой литерал, вводимые данные будут считываться и заноситься непосредственно в список спецификаций.

Например, операторы

READ (5,100)

100 FORMAT (6HДАННОЕ)

вызывают ввод шести символов и запоминание их в спецификации формата типа "H".

Операторы форматного ввода/вывода читают или пишут записи определенной длины в соответствии с форматом. Запись состоит из ряда символов, передаваемых под управлением форматной спецификации, за которыми следует знак разделения записи.

Если число элементов в списке меньше, чем число полей во вводимой записи, оставшаяся часть записи отбрасывается. Если число элементов в списке ввода превышает число полей во вводимом списке, то будет зафиксировано ошибочное состояние, если только в спецификации формата не указано, что должны быть введены одна или более дополнительных записей.

Примеры:

|  |  |
| --- | --- |
| READ (1,700) ARRAY  700 FORMAT (20F8.2) | Считывается единица записи с логического устройства 1. Значения полей присваиваются элементам массива "ARRAY". |
| READ (5,50)  50 FORMAT (19H ЗАГОЛОВОК СТРАНИЦЫ) | По этим операторам считывается 19 символов с логического устройства 5 и помещаются в оператор "FORMAT". |

#### 5.3.2.2. Оператор форматного последовательного вывода "WRITE"

Оператор форматного последовательного вывода "WRITE" передает данные на указанное логическое устройство. Список вывода определяет значения, которые преобразуются в символы и выводятся согласно спецификации формата.

Оператор форматного последовательного вывода "WRITE" имеет следующий вид:

WRITE (U,F[,ERR=S][END=S]) [K]

где

U - номер логического устройства;

F - метка соответствующего оператора "FORMAT";

S - метка выполняемого оператора;

K - список вывода.

Если список вывода отсутствует, обмен данными полностью происходит между записью и спецификацией формата.

Данные, передаваемые оператором форматного последовательного вывода "WRITE", обычно составляют одну форматную запись. Однако в спецификации формата может быть указано, что при выполнении того же самого оператора "WRITE" должны быть выведены дополнительные записи.

Записи, выводимые оператором форматного вывода "WRITE", должны иметь длину, не превышающую значения, характерного для данного внешнего устройства. Например, устройство печати не может напечатать запись, длина которой превышает 132 символа.

Примеры:

|  |  |
| --- | --- |
| WRITE (6,60)  60 FORMAT ('PROGRAM') | вывод содержимого оператора "FORMAT" на логическое устройство 6; |
| WRITE (1,95) AU,BE,CE  95 FORMAT (F8.5,F8.5,F8.5) | вывод одной записи, состоящей из трех полей, на логическое устройство 1; |
| WRITE(1,950) AU,BE,CE  950 FORMAT (F8.5) | вывод трех отдельных записей по одному полю каждая на логическое устройство 1. |

### 5.3.3. Неформатный ввод/вывод

Неформатный ввод/вывод (т.е. без преобразования данных) выполняется при помощи операторов:

READ (U,[,ERR=S1][,END=S2]) K

WRITE (U,[,ERR=S1][,END=S2]) K

где

|  |  |  |
| --- | --- | --- |
| U | - | номер логического устройства; |
| S1 | - | метка оператора, которому передается управление при ошибке ввода/вывода; |
| S2 | - | метка оператора, которому передается управление при возникновении условия конца файла; |
| K | - | список ввода/вывода. |

Неформатный ввод/вывод осуществляет передачу данных между внешним устройством и памятью без преобразования данных или их редактирования. Количество передаваемых при чтении данных определяется числом переменных в списке и их типом. Общая длина данных в списке ввода/вывода не должна превышать длины записи. Если длина логической записи и длина данных в списке ввода/вывода совпадают, передается вся запись. Если длина данных в списке меньше длины логической записи, то оставшиеся в записи данные теряются.

Оператор "WRITE (U) K" записывает одну логическую запись. Логическая запись может состоять более чем из одной физической записи.

### 5.3.4. Ввод/вывод с прямым доступом

Ввод/вывод с прямым доступом выполняется при помощи операторов

READ (U,[,F],REC=N[,ERR=S]) K

WRITE (U,[,F],REC=N[,ERR=S]) K

где

|  |  |  |
| --- | --- | --- |
| U | - | номер логического устройства; |
| F | - | метка соответствующего оператора "FORMAT"; |
| REC=N | - | ключевое слово |
| N |  | номер записи; |
| S | - | метка оператора, которому передается управление при возникновении ошибки ввода/вывода; |
| K | - | список ввода/вывода. |

Длина записи любого файла с прямым доступом равна 128 байтам (1 сектору), а операнд REC=N определяет номер записи. Если вводится запись с номером I и ее не существует в файле, то возникает ошибка ввода/вывода. Если производится вывод несуществующей записи, то файл расширяется.

Файлы с прямым доступом могут создаваться как форматными, так и неформатными операторами "WRITE". Если длина данных списка ввода/вывода меньше длины записи (128 байтов), то остаток записи заполняется нулями.

### 5.3.5. Единицы ввода/вывода

Обмен информацией с помощью операторов ввода/вывода ведется определенными порциями, которые называются записями. Количество информации, которое может содержаться в одной записи, а также способы разделения этих записей зависят от используемого носителя информации.

Для неформатного ввода/вывода количество данных, которое должно быть передано, определяется оператором ввода/вывода. В случае форматного ввода/вывода количество передаваемой информации определяется совместно с оператором ввода/вывода и спецификациями в соответствующем описании формата.

Начало выполнения оператора ввода/вывода инициирует передачу новой записи. Если оператор ввода потребовал только часть записи, остальная часть записи теряется. В случае форматного последовательного ввода/вывода отдельным оператором ввода/вывода может быть передано несколько записей.

### 5.3.6. Списки ввода/вывода

Список ввода/вывода определяет элементы данных, передаваемые между памятью и внешним устройством.

Список имеет следующий вид:

М1, М2,..., Мк

где

Мi (i=1,2,...,k) - элемент списка.

Элементом списка может быть простой или сложный элемент.

Простой элемент - это:

1. Имя простой переменной (например, C);
2. Элемент массива (например, A(1,2), B(3,4), N(5));
3. Несколько таких элементов, разделенных запятыми и заключенных в круглые скобки.

Сложным элементом может быть:

1. Имя массива, рассматриваемое как список всех элементов массива.  
   Пусть, например, А - имя массива размерности М\*N. Тогда сложный элемент А в списке ввода/вывода эквивалентен такому списку:  
     
   А(1,1), А(2,1),..., А(1,2), А(2,2),..., А(М,N);
2. Элемент типа неявное "DO"

Такой элемент имеет следующий вид:

(S, I=E1,E2[,E3])

где

|  |  |  |
| --- | --- | --- |
| S | - | список ввода/вывода; |
| I | - | переменная цикла; |
| E1,E2,E3 | - | параметры цикла. |

Величины I, E1, E2, E3 имеют такой же вид, как и в операторе "DO", а список S может состоять как из простых, так и из сложных элементов, включая элементы типа неявного "DO". Глубина вложенности не должна превышать 3.

Например:

WRITE (4,500) (A,B,C, F=1,3)

WRITE (6,70) L,M,(I,(J,P(I),Q(I,J),J=1,L),I=1,M)

READ (1,100)((ARRAY(M,N,I),I=2,9),N=2,9),M=2,9)

Определение первой переменной цикла эквивалентно самому внутреннему оператору "DO" из множества вложенных циклов, и, следовательно, эта переменная будет изменяться чаще. Например, операторы

WRITE (5,150) ((FAST(K,L),L=1,10),K=1,10,2)

150 FORMAT (F10.2)

аналогичны следующим операторам:

DO 50 K=1,10,2

DO 50 L=1,10

WRITE (5,150) FAST(K,L)

150 FORMAT (F10.2)

50 CONTINUE

Так как внутренний цикл выполняется десять раз для каждой итерации внешнего цикла, второй индекс "l" пробегает значения от 1 до 10 для каждого значения первого индекса. Это является обратным порядком по отношению к индексной прогрессии. Кроме того, т.к. "K" увеличивается на "2", будут выведены только четные столбцы массива. Перед очередным увеличением переменной цикла выполняется передача всего списка с неявным циклом.

Например:

READ (5,1000) (P(I),(Q(I,J),J=1,10),I=1,5)

присваивает введенные значения элементам массивов "P" и "Q" в следующей последовательности:

P(1),Q(1,1),Q(1,2),...,Q(1,10)

P(2),Q(2,1),Q(2,2),...,Q(2,10)

. . .

P(5),Q(5,1),Q(5,2),...,Q(5,10)

При обработке многомерных массивов допускается использование комбинации из фиксированных индексов и индексов, изменяющихся по неявному циклу.

Например, оператор

READ (3,2222) (BOX(1,J), J=1,10)

Присваивает вводимые значения элементам массива от BOX(1,1) до BOX(1,10), а затем прекращает ввод, не затрагивая других элементов данного массива.

Можно также вывести непосредственно значения переменной цикла, как, например, в операторе,

WRITE (6,3300)(I,I=1,20)

который выводит на печать целые числа от 1 до 20.

## 5.4. Оператор "FORMAT"

Оператор "FORMAT" является невыполняемым оператором, используемым совместно с операторами форматного ввода/вывода и с операторами "ENCODE" и "DECODE".

Оператор "FORMAT" описывает формат, в котором передаются поля данных, а также преобразование и редактирование, которые должны быть выполнены для приведения данных к этому формату.

Оператор "FORMAT" имеет следующий вид:

FORMAT (Q1F1S1Q2F2S2...FnSnQn)

где

|  |  |  |
| --- | --- | --- |
| Q1,...,Qn | - | группа дробных черт, общим числом от нуля и более, каждая из которых является признаком начала новой единицы записи; |
| F1,...,Fn | - | спецификация преобразования или группа спецификаций, заключенных в скобки; |
| S1,...,Sn | - | разделительный знак. |

Все выражение, включая скобки, называется списком спецификаций. Наличие внешних скобок является обязательным.

Спецификация преобразования имеет следующий вид:

[R]CW[.K]

где

|  |  |  |
| --- | --- | --- |
| R | - | коэффициент повторения, который указывает, что данная спецификация должна быть применена к "R" последовательным полям. Если коэффициент повторения опущен, он полагается равным единице; |
| C | - | тип спецификации; |
| W | - | определяет длину поля; |
| K | - | определяет число символов справа от десятичной точки. |

Величины "R", "W", "K" должны быть целыми константами без знака, не превышающими 255.

К разделительным знакам относятся запятая и дробная черта, дробная черта кроме того, служит признаком начала новой записи.

В списке спецификаций используются следующие спецификации преобразования:

* спецификация целых чисел: IW;
* логическая спецификация: LW;
* спецификация вещественных чисел и с двойной точностью: FW.K или EW.K, DW.K, GW.K;
* литеральные и редактирующие спецификации: AW, NH, '...', NX.

В буквенно-цифровых и редактирующих спецификациях "N" определяет число символов или символьных позиций.

Каждой из спецификаций типа "F", "E", "D" или "G" может предшествовать масштабный коэффициент в виде:

NP

где

|  |  |  |
| --- | --- | --- |
| N | - | это целая константа с необязательным знаком, которая может иметь значения в интервале от минус 127 до +127. Эта константа указывает, на сколько позиций влево или вправо сдвигается положение десятичной точки; |
| P | - | идентификатор масштабного коэффициента. |

Во время передачи данных выполняется сканирование списка спецификаций слева направо. Величины из списка ввода/вывода преобразуются под управлением соответствующих спецификаций преобразования.

В случае спецификаций типа "H" и литеральных спецификаций весь обмен данными происходит между списком спецификаций и внешней записью.

### 5.4.1. Спецификации преобразования

Спецификации преобразования игнорируют пробелы, предшествующие значащей части внешнего поля, а завершающие пробелы и пробелы внутри поля рассматриваются как нули.

#### 5.4.1.1. Спецификация "I"

Спецификация "I" управляет преобразованием данных целого типа. Она имеет следующий вид:

IW

Спецификация "I" вызывает считывание оператором ввода "W" символов из внешней записи (поля) и присваивание целого значения соответствующему целому элементу из списка ввода. Внешние данные должны быть целыми величинами. Они не должны содержать ни десятичной точки, ни поля указателя степени. Если значение внешнего поля превышает допустимое значение соответствующего целого элемента списка, фиксируется ошибка.

Если первый отличный от пробела символ внешнего поля является знаком "-", то по спецификации "I" это поле запоминается как отрицательная величина. Поле, которому предшествует знак "+", или поле без знака рассматривается как положительная величина.

Например:

|  |  |  |
| --- | --- | --- |
| Спецификация | Внешнее поле | Внутреннее представление |
| I4 | 2788 | 2788 |
| I3 | -35 | -35 |
| I9 | 415 | 415 |
| I9 | 3.12 | не допускается, ошибка |

Спецификация "I" вызывает передачу оператором вывода значения соответствующего списка вывода во внешнее поле, имеющее длину "W" символов и выравненное справа, причем незначащие левые нули заменяются пробелами. Если этот элемент списка имеет отрицательную величину в качестве первого левого символа, отличного от пробела, во внешнем поле будет знак "-". Поэтому в "W" должен учитываться знак "-", если значение меньше нуля. С другой стороны, знак "+" не выводится и его не надо учитывать в "W". Если количество символов "W" слишком мало, чтобы поле вместило запись выводимой величины, то выводятся младшие W-1 знаков, перед которыми помещается звездочка (\*).

Спецификации "I" для оператора вывода:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Спецификация | Внутренняя величина | | Внешнее представление | |
| I3 |  | 284 |  | 284 |
| I4 |  | -125 |  | -125 |
| I5 |  | 174 |  | 174 |
| I2 |  | 3244 |  | \*4 |
| I3 |  | -473 |  | \*73 |
| I7 |  | 29.905 | не допускается, ошибка | |

#### 5.4.1.2. Спецификация "F"

Спецификация "F" определяет преобразование и редактирование вещественных величин, величин с двойной точностью. Она имеет следующий вид:

FW.K

При вводе спецификация "F" вызывает считывание "W" символов из внешней записи (поля) и присваивание их в виде вещественного значения соответствующему элементу списка ввода. Если первым отличным от пробела символом во внешнем поле является знак "-", значение, записанное в том поле, рассматривается как отрицательная величина. Если полю предшествует знак "+" либо если поле не имеет знака, то оно считается положительным. Во всех случаях в записи спецификации "F" величина "W" должна быть больше или равна величине "K+1".

Если поле не содержит ни десятичной точки, ни экспоненты, оно рассматривается как содержащее запись вещественного числа, состоящего из "W" цифр, в котором правые "K" цифр находятся справа от десятичной точки. Если в поле содержится десятичная точка в явном виде, то величина "K" не принимается во внимание. Если поле содержит указание степени, то эта степень используется в формировании величины перед ее присваиванием элементу списка.

Примеры использования спецификации "F" для преобразования вещественных величин при вводе:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Спецификация | Внешнее поле | | Внутреннее представление | |
| F8.5 |  | 135679452 |  | 135.67945 |
| F8.5 |  | 24.77e+2 |  | 2477.0 |
| F5.2 |  | 1234567.89 |  | 123.45 |

При выводе спецификация "F" вызывает округление значения соответствующего элемента списка вывода до "K" десятичных знаков в дробной части, передачу его во внешнее поле, имеющее длину в "W" символов и выравненное справа. Если преобразуемые данные состоят менее чем из "W" символов, в поле заносятся левые пробелы. Если данные состоят более чем из "W" символов, выводятся младшие "W-1" символов, которым предшествует звездочка "\*".

Общее число символов "W" должно быть достаточно велико, чтобы учесть следующее:

* наличие знака "-", если он имеется (знак "+" не выводится);
* как минимум одной цифры слева от десятичной точки;
* самой десятичной точки;
* наличие "K" цифр справа от точки.

Поэтому величина "W" должна всегда быть больше или равна величине "K+3". Примеры использования спецификации "F" для редактирования вещественных величин при выводе:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | F8.5 |  | 2.4875189 |  | 2.48752 |
|  | F9.3 |  | 8837.4761 |  | 8837.476 |
|  | F10.4 |  | -23.28732 |  | -23.2873 |
|  | F5.2 |  | 765.014 |  | \*5.01 |
|  | F5.2 |  | -.2 |  | -0.20 |

#### 5.4.1.3. Спецификация "E"

Спецификация "E" определяет передачу вещественных величин или величин с двойной точностью в экспоненциальной форме. Она имеет следующий вид:

EW.K

Спецификация "E" вызывает ввод оператором ввода "W" символов из внешней записи (поля). Эти данные интерпретируются и присваиваются точно таким же образом, как и в случае спецификации "F".

Примеры использования спецификации "E" для операторов ввода:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внешнее поле | | Внутреннее представление | |
|  | E9.3 |  | 689.234E3 |  | 689234.0 |
|  | Е12.4 |  | 1042.34E-6 |  | 1042.34E-6 |
|  | Е15.12 |  | 52.3759663 |  | 52.3759663 |
|  | Е12.5 |  | 210.3271D+10 |  | 210.3271E10 |

Необходимо заметить, что в последнем примере спецификация "E" игнорирует дополнительное назначение признака поля порядка "D", как указателя двойной точности, и рассматривает его, как признак "E".

Спецификация "E" вызывает передачу оператором вывода значения соответствующего элемента списка вывода во внешнее поле длиной в "W" символов, выравненное по правой позиции. Если число символов в преобразованных данных меньше "W", то в левые позиции поля заносятся пробелы. Если число символов превышает "W", то могут быть потеряны значащие символы.

Данные, выводимые под управлением спецификации "E", передаются в стандартной форме, состоящей из знака "-", если величина отрицательная (знак "+" не выводится), нуля, десятичной точки, "K" цифр справа от десятичной точки и четырехсимвольного порядка числа в виде:

E+NN или E-NN

где

|  |  |  |
| --- | --- | --- |
| NN | - | двузначная целая константа. |

"K" цифр справа от десятичной точки представляют величину, сведенную к правильной десятичной дроби.

Так как длина поля должна быть достаточно большой, чтобы вместить знак "-" (если он присутствует), нуль, десятичную точку, порядок и "K" цифр, величина "W" должна быть всегда больше или равна величине (K+7).

Примеры использования спецификации "E" для операторов вывода:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | E9.2 |  | 475867.233 |  | 0.48E+06 |
|  | E12.5 |  | 475867.233 |  | 0.47587E+06 |
|  | E12.3 |  | 0.00069 |  | 0.690E-3 |
|  | E10.3 |  | -0.5555 |  | -0.556E+00 |
|  | E7.3 |  | 56.12 |  | \*0.561E+2 |

#### 5.4.1.4. Спецификация "D"

Спецификация "D" определяет передачу вещественных величин или величин с двойной точностью. Она имеет следующий вид:

DW.K

При вводе спецификация "D" выполняет те же функции, что и эквивалентная спецификация "E", за исключением того, что вводимые данные преобразуются и присваиваются объекту с двойной точностью, как показано в следующих примерах:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внешнее поле | | Внутреннее представление | |
|  | D10.2 |  | 12345 |  | 12345000.0D0 |
|  | D10.2 |  | 123.45 |  | 123.45D0 |
|  | D15.3 |  | 367.4571263D+04 |  | 3.674571263D+06 |

При выводе спецификация "D" работает аналогично спецификации "E", за исключением того, что вместо признака порядка "E" используется признак порядка "D".

Например:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | D14.3 |  | 0.0463 |  | 0.463D-01 |
|  | D23.12 |  | 5573.86789251 |  | 0.557386789251D+04 |

#### 5.4.1.5. Спецификация "G"

Спецификация "G" управляет передачей данных вещественного типа и с двойной точностью в формате, являющимся комбинацией "F" и "E". Эта спецификация имеет следующий вид:

GW.K

При вводе работа спецификации "G" идентична работе спецификации "F".

При выводе спецификация "G" вызывает передачу значения соответствующего элемента списка вывода во внешнее поле длиной в "W" символов, выравненное по правой позиции. Формат, в котором выводится значение, является функцией величины этого значения.

Зависимость фактического формата от величины порядка числа при выводе по спецификации "G" следующая:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Величина данных | Результирующее преобразование | |
|  | M<0.1 |  | EW.K |
|  | 0.1<=M<1.0 |  | F(W-4).K,' ' |
|  | 1.0<=M<10.0 |  | F(W-4).(K-1),' ' |
|  | . |  | . |
|  | . |  | . |
|  | . |  | . |
|  | 10\*\*(K-2)<=M<10\*\*(K-1) |  | F(W-4).1,' ' |
|  | 10\*\*(K-1)<=M<10\*\*K |  | F(W-4).0,' ' |
|  | M>=10\*\*K |  | EW.K |

Спецификация ' ', которая вносится (в результате) спецификацией "G" внутрь поля величины, означает, что за представлением числовых данных должны следовать четыре пробела.

В длине поля "W" должно быть учтено место для знака "-", если он есть (знак "+" не выводится), как минимум для одной цифры слева от десятичной точки, для самой десятичной точки, для "K" цифр справа от десятичной точки и (для величин, выходящих за пределы рабочего диапазона спецификации "G") для четырехсимвольного показателя степени. Таким образом, величина "W" всегда должна быть больше или равна величине (K+7).

Примеры преобразования выводимых данных по спецификации "G":

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | G13.6 |  | 0.01234567 |  | 0.123457E-01 |
|  | G13.6 |  | -0.12345678 |  | -0.123457 |
|  | G13.6 |  | 1.23456789 |  | 1.23457 |
|  | G13.6 |  | 12.34567890 |  | 12.3457 |
|  | G13.6 |  | 123.45678901 |  | 123.457 |
|  | G13.6 |  | -1234.56789012 |  | -1234.57 |
|  | G13.6 |  | 12345.67890123 |  | 12345.7 |
|  | G13.6 |  | 123456.78901234 |  | 123457. |
|  | G13.6 |  | -1234567.89012345 |  | -0.123457E+07 |

Ниже приведены примеры вывода этих же самых значений под управлением эквивалентной спецификации "F".

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | f13.6 |  | 0.01234567 |  | 0.012346 |
|  | f13.6 |  | -0.12345678 |  | -0.123457 |
|  | f13.6 |  | 1.23456789 |  | 1.234568 |
|  | f13.6 |  | 12.34567890 |  | 12.345679 |
|  | f13.6 |  | 123.45678901 |  | 123.456789 |
|  | f13.6 |  | -1234.56789012 |  | -1234.567890 |
|  | f13.6 |  | 12345.67890123 |  | 12345.678901 |
|  | f13.6 |  | 123456.78901234 |  | 123456.789012 |
|  | f13.6 |  | -1234567.89012345 |  | \*34567.890123 |

#### 5.4.1.6. Спецификация "L"

Спецификация "L" определяет передачу логических данных. Она имеет следующий вид:

LW

Спецификация "L" вызывает считывание оператором ввода "W" символов из внешней записи. Если первым отличным от пробела символом в этой записи будет "T", соответствующему элементу списка ввода присваивается значение ".TRUE." (соответствующий элемент списка должен иметь логический тип). Если первым отличным от пробела символом в записи является буква "F", или если вся внешняя запись состоит из пробелов, то выполняется присваивание значения ".FALSE.". Любое другое значение, находящееся во внешней записи, вызывает ошибку.

Спецификация "L" вызывает передачу во внешнее поле оператором вывода либо буквы "T", если значение соответствующего элемента списка равно ".TRUE.", либо буквы "F", если это значение равно ".FALSE.". Внешнее поле состоит из "W" символьных позиций. Буквы "T" или "F" записываются в самую правую позицию этого поля, а остальные позиции заполняются пробелами.

Например:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | L5 |  | .TRUE. |  | T |
|  | L1 |  | .FALSE. |  | F |

#### 5.4.1.7. Спецификация "A"

Спецификация "A" определяет передачу буквенно-цифровых данных. Она имеет следующий вид:

AW

При вводе спецификация "A" вызывает считывание "W" символов из внешней записи и запоминание их в соответствующем элементе списка (соответствующий элемент списка ввода может иметь любой тип). Максимальное число символов, которое может быть запомнено в переменной или элементе массива, зависит от типа этого элемента или переменной.

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент списка ввода/вывода | | Максимальное число символов | |
|  | LOGICAL\*1 |  | 1 |
|  | INTEGER\*2 |  | 2 |
|  | INTEGER\*4 |  | 4 |
|  | REAL |  | 4 |
|  | DOUBLE PRECISION |  | 8 |

Если "W" больше, чем максимальное число символов, которое может быть запомнено в соответствующем элементе списка, то этому объекту будут присвоены только один, два, четыре или восемь самых правых символов (в зависимости от типа данных этой переменной или элемента массива). Избыточные левые символы при этом теряются. Если "W" меньше, чем число символов, которое может быть запомнено в данном объекте, то элементу списка будут присвоены слева "W" символов, а в оставшиеся незаполненными правые символьные позиции занесутся пробелы.

Примеры преобразования буквенно-цифровых данных при вводе по спецификации "A":

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внешнее поле | | Внутреннее представление | |
|  | A6 |  | PAGE \* | \* | (LOGICAL\*1) |
|  | A6 |  | PAGE \* | \* | (INTEGER\*2) |
|  | A6 |  | PAGE \* | GE \* | (REAL) |
|  | A6 |  | PAGE \* | PAGE \* | (DOUBLE PRECISION) |

При выводе спецификация "A" вызывает передачу содержимого соответствующего элемента вывода во внешнее поле, состоящее из "W" символьных позиций. Если элемент списка вывода содержит менее "W" символов, данные записываются выравненными по правой позиции, а в оставшиеся левые позиции заносятся пробелы. Если элемент списка содержит более "W" символов, то будут переданы только старшие "W" символов.

Примеры преобразования буквенно-цифровых данных при выводе по спецификации "A":

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | A5 |  | BETA |  | BETA |
|  | A5 |  | VOLTS |  | VOLTS |
|  | A5 |  | ALPHAES |  | ALPHA |

#### 5.4.1.8. Спецификация "H"

Спецификация "H" имеет вид текстовой константы:

nHc1c2c3...cn

либо литерала:

'c1c2c3...cn'

где

|  |  |  |
| --- | --- | --- |
| n | - | число символов, которые должны быть переданы; |
| H | - | буква h; |
| ci (i=1,2,...,n) | - | символы. |

При использовании в литерале в качестве символа ci апострофа ('), его необходимо удваивать.

Вывод по спецификации "H" вызывает передачу символов ci на внешний носитель.

Примеры:

|  |  |
| --- | --- |
| Спецификация | Вывод |
| 1HA ИЛИ 'A' | A |
| 8H STRING | STRING |
| //HX(2,3)=12.0 | X(2,3)=12.0 |
| 'X(2,3)=12.0' | X(2,3)=12.0 |

При вводе n символов ci заменяются n символами, передаваемыми из входной записи. В результате получается спецификация формата с новым содержимым.

Примеры:

|  |  |  |
| --- | --- | --- |
| Спецификация | Ввод | Результирующая спецификация |
| 4H1234 | ABCD | 4HABCD |
| '1234' | ABCD | 4HABCD |
| 7H FALSE | FALSE | 7H FALSE |
| ' FALSE' | FALSE | 7H FALSE |

#### 5.4.1.9. Спецификация "X"

Спецификация "X" имеет следующий вид:

nX

По этой спецификации оператор "WRITE" засылает n пробелов во внешнюю запись. Например:

WRITE(6,100) NPAGE

100 FORMAT(16H1Страница номер ,I2,16X,6HГрафик)

Оператор "WRITE" напечатает запись следующего вида:

Страница номер nn График

где

nn - текущее значение переменной "NPAGE".

Цифра "1" из первой спецификации "H" не печатается, а используется для подачи бумаги на устройстве печати к началу нового листа. При вводе по этой спецификации будет пропущено "n" знаков вводимой записи, начиная от текущей позиции.

### 5.4.2. Масштабный коэффициент

С помощью масштабного коэффициента при выполнении ввода или вывода может быть изменено положение десятичной точки в вещественных величинах с двойной точностью. Масштабный коэффициент имеет следующий вид:

nP

где

|  |  |  |
| --- | --- | --- |
| n | - | целая константа со знаком или без знака, лежащая в диапазоне от минус 127 до +127, определяющая число позиций, на которое должна быть сдвинута десятичная точка вправо или влево. |

Масштабный коэффициент может встречаться в любом месте списка спецификаций, но он должен предшествовать спецификации, к которой он относится.

Допускаются следующие записи:

nPFW.K nPEW.K nPDW.K nPGW.K

Данные, введенные под управлением одной из вышеприведенных спецификаций, перед их присваиванием соответствующему элементу списка ввода/вывода умножаются на 10\*\*(-n). Например, масштабный коэффициент "2P" вызывает умножение введенной величины на 0.01, перемещая десятичную точку на две позиции влево, а масштабный коэффициент "-2P" вызывает умножение введенной величины на 100, перемещая десятичную точку на две позиции вправо. Если же внешнее поле содержит показатель степени в явном виде, масштабный коэффициент вычитается.

Например:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внешнее поле | | Внутреннее представление | |
|  | 3PE10.5 |  | 47.518 |  | .047518 |
|  | 3PE10.5 |  | 47.518E5 |  | 4751.8 |
|  | -3PE10.5 |  | 47.518 |  | 47518.0 |

Влияние масштабного коэффициента при выводе зависит от типа спецификации, которой он соответствует. Для спецификации "F" значение элемента списка ввода/вывода перед передачей во внешнюю запись умножается на 10\*\*n. Таким образом, положительный масштабный коэффициент перемещает десятичную точку вправо, а отрицательный - влево.

Величины, выводимые под управлением спецификации "E" и "D" с масштабным коэффициентом, преобразуются умножением вещественной константы, входящей в состав каждой величины, на 10\*\*n и вычитанием "n" из ее порядка.

Таким образом, положительный масштабный коэффициент перемещает десятичную точку вправо и уменьшает порядок, а отрицательный масштабный коэффициент перемещает десятичную точку влево и увеличивает порядок.

Действие масштабного коэффициента приостанавливается при вводе под управлением спецификации "G", когда значения выводимых данных лежат внутри рабочего диапазона спецификации "G", так как эта спецификация сама выполняет масштабирование. Если же значение величины лежит за пределами рабочего диапазона спецификации "G", то влияние масштабного коэффициента будет точно таким же, как и при спецификации "E" (так как в этом случае сама спецификация "G" эквивалентна спецификации "E").

Необходимо отметить, что при вводе и выводе под управлением спецификации "F" масштабный коэффициент фактически изменяет величину данных, а при выводе под управлением спецификации "E", "D" и "G" масштабный коэффициент всего лишь изменяет форму, в которой передаются данные. При вводе положительный масштабный коэффициент перемещает десятичную точку влево, а отрицательный - вправо, в то время как при выводе - наоборот.

Если в спецификации не указан масштабный коэффициент, то он считается равным нулю. Если масштабный коэффициент был один раз определен, то он влияет на последующие спецификации вещественных и с двойной точностью преобразований, которые находятся в этом же списке спецификаций до тех пор, пока не встретится другой масштабный коэффициент, который изменяет управление. Повторная форматизация не влияет на масштабный коэффициент. Нулевой коэффициент может быть восстановлен только явной спецификацией "0P".

Примеры влияния масштабного коэффициента при выводе:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Спецификация | | Внутреннее значение | | Внешнее представление | |
|  | 1PE12.3 |  | -480.148 |  | -4.801E+02 |
|  | 1PE12.2 |  | -480.148 |  | -4.80E+02 |
|  | -1PE12.2 |  | -480.148 |  | -0.05E+04 |

### 5.4.3. Группирование и повторяемые группы

Ряд одинаковых, расположенных друг за другом спецификаций в операторе "FORMAT" можно заменить одной спецификацией с предшествующим ей коэффициентом повторения. Такое объединение можно производить для спецификаций "I", "F", "E", "G", "D", "L" и "A". Коэффициентом повторения может быть только целая константа без знака. Например, операторы:

30 FORMAT (E10.3,E10.3,E10.3,I4,I4,I4,I4)

и

30 FORMAT (3E10.3,4I4)

идентичны.

Аналогично допускается повторение группы спецификаций, если ее заключить в скобки и перед открывающей скобкой поставить целую константу, означающую количество повторений этой группы. Например, оператор

77 FORMAT (2I5,3(F4.3,E12.5)

эквивалентен оператору

77 FORMAT (I5,I5,F4.3,E12.5,F4.3,E12.5,F4.3,E12.5)

Спецификацию "H", которая обычным образом не может быть повторена, можно заключить в скобки и рассматривать как повторяемую группу. Таким образом, она может быть повторена нужное число раз.

Если коэффициент повторения группы опущен, он подразумевается равным единице.

### 5.4.4. Управление кареткой

Первый символ каждой единицы записи, передаваемой на устройство печати, никогда не печатается: этот символ интерпретируется как символ управления кареткой. Действие этих символов следующее:

|  |  |
| --- | --- |
| Управляющий символ | Действие, выполняемое перед печатью |
| 0 | Пропуск 2 строк |
| 1 | Продвижение до первой строки следующей страницы |
| + | Отсутствие продвижения бумаги |
| любой другой | Пропуск 1 строки |

### 5.4.5. Разделительные знаки списка спецификаций

Спецификации преобразования в списке спецификаций, как правило, отделяются друг от друга запятыми. Для разделения спецификаций может также использоваться дробная черта "/" - признак начала новой записи. Дробная черта вызывает завершение вводимой или выводимой записи и инициирование новой записи. Например, такие операторы:

WRITE (6,70) I,L,N,M,O,P

70 FORMAT (306/I5,2F4.2)

эквивалентны следующей группе операторов:

WRITE (6,70) I,L,N

70 FORMAT (306)

WRITE (6,40) M,O,P

40 FORMAT (I5,2F4.2)

С помощью нескольких, идущих подряд признаков начала новой записи, можно выполнить пропуск нескольких вводимых записей или вывод нескольких пустых записей. Если между двумя спецификациями преобразования расположены N идущих подряд наклонных черт, то это вызовет пропуск (N-1) записей при вводе или выдачу (N-1) пустых (состоящих из пробелов) записей, т.е. пустых строк при выводе. Первая наклонная черта заканчивает текущую запись, вторая наклонная черта заканчивает первую пропускаемую или пустую запись и так далее. Но если N подряд расположенных наклонных черт встречаются в начале или в конце списка спецификаций, они вызывают N пропущенных или пустых записей, т.к. открывающая или закрывающая скобки списка спецификаций являются признаком начала новой записи и признаком завершения записи.

#### 5.4.5.1. Переменный список спецификаций

Списки спецификаций могут храниться в массивах. Такой список спецификаций (называемый переменным списком спецификаций) может быть построен или изменен непосредственно при выполнении программы. Вид списка спецификаций в массиве идентичен оператору "FORMAT" за исключением того, что в нем отсутствует слово "FORMAT" и метка оператора. Однако наличие открывающей и закрывающей скобок является обязательным. Переменный список спецификаций не может храниться в виртуальных массивах.

Пример переменного списка спецификаций:

REAL FACT(5,5)

DOUBLE PRECISION FORM(6),REC,FEST,FEB,SML

DATA FORM(1),REG,FEST,FEB,SML/'(',')','F8.2','F9.4','F9.6'/

DO 30 J=1,5

DO 31 I=1,5

IF (FACT(I,J).LT.100.AND.FACT(I,J).GT.0.1) FORM(I+1)=FEB

IF (FACT(I,J).GE.100.)FORM(I+1)=FEST

IF (FACT(I,J).LE.0.1) FORM(I+1)=SML

31 CONTINUE

FORM(I+1)=REG

WRITE(5,FORM) (FACT(K,J),K=1,5)

30 CONTINUE

END

В этом примере оператор "DATA" присваивает код левой скобки первому элементу массива "FORM" и присваивает коды правой скобки и три спецификации переменным для последующего использования. Затем выбираются соответствующие спецификации для включения их в список спецификаций, исходя из значений отдельных элементов массива "FACT". Затем списку спецификаций перед его использованием в операторе "WRITE" добавляется правая скобка. Таким образом, список спецификаций изменяется с каждым повторением цикла оператора "DO".

### 5.4.6. Взаимодействие управления форматом и списка ввода/вывода

Управление форматом инициируется началом выполнения оператора форматного ввода/вывода. Каждое действие по управлению форматом зависит от информации, предоставляемой совместно очередным элементом списка ввода/вывода (если он имеется) и очередной спецификацией оператора "FORMAT" или массива, содержащего список спецификаций. Как список спецификаций, так и список ввода/вывода просматривается слева направо (за исключением случаев повторений).

Если оператор ввода/вывода содержит список ввода/вывода, то в списке спецификаций должна иметься хотя бы одна спецификация преобразования, отличная от спецификаций типа "H", "X" и "P". Если это условие не выполняется, то при выполнении программы фиксируется ошибка.

При выполнении оператора форматного ввода считывается одна запись с указанного устройства и инициируется управление форматом, после этого могут быть считаны дополнительные записи, как указано в списке спецификаций. Управление форматом требует ввода новой записи всякий раз, когда в списке спецификаций встречается дробная черта или когда достигается последняя внешняя закрывающая скобка списка спецификаций, а в списке ввода еще остались незаполненные элементы. При считывании новой записи теряются все оставшиеся символы текущей записи.

При выполнении оператора форматного вывода запись передается на указанное устройство только по завершении управления форматом. Возможен также вывод нескольких записей при выполнении управления форматом, если в списке спецификаций встретится дробная черта или если достигается последняя внешняя закрывающая скобка, а в списке вывода еще имеются элементы, подлежащие выводу.

Каждая спецификация типа "I", "F", "E", "D", "G", "L" и "A" соответствует одному элементу списка ввода/вывода. Спецификациям "H", "X", "P" и спецификации в виде буквенно-цифрового литерала не ставится в соответствие ни один элемент списка. В случае спецификации типа "H" или спецификации в виде буквенно-цифрового литерала передача данных осуществляется непосредственно между записью и списком спецификаций.

Если управление форматом обнаруживает спецификацию типа "I", "F", "E", "D", "G", "L" и "A", то определяется наличие соответствующего элемента в списке ввода/вывода. Если такой элемент имеется, то управление форматом осуществляет передачу соответствующим образом преобразованных данных между записью и элементом списка, а затем переходит к обработке следующей спецификации (если не требуется повторения текущей спецификации). Если же соответствующего элемента в списке не обнаружено, то управление форматом завершается.

При достижении последней закрывающей скобки списка спецификаций управление форматом определяет, имеются ли еще в списке ввода/вывода элементы, подлежащие обработке. Если таковых не имеется, управление форматом завершается. Если в списке ввода/вывода остались необработанные элементы, то текущая запись завершается, а управление форматом возвращается к самой правой повторяемой группе спецификаций самого высокого уровня (открывающая скобка которой составляет пару с предпоследней закрывающей скобкой списка спецификаций). Этот принцип известен, как повторная форматизация. Если в списке спецификаций отсутствуют повторяемые группы, управление форматом вернется к начальной открывающей скобке списка спецификаций. Действия по управлению форматом будут продолжены с этой точки.

### 5.4.7. Сводка правил для оператора "FORMAT"

Ниже приводится перечень правил, связанных со структурой и использованием оператора "FORMAT" (или массива, содержащего список спецификаций) и его составных частей, а также правил, связанных со структурой внешних полей и записей, с которыми взаимодействует список спецификаций.

#### 5.4.7.1. Общие правила

Правило 1. Оператор "FORMAT" обязательно должен иметь метку.

Правило 2. В спецификациях, таких как "RIW" или "NX", величины "R", "W" и "N" должны быть целыми положительными константами без знака. Коэффициент повторения и спецификация ширины поля могут быть опущены.

Правило 3. В спецификациях, таких как "FW.K", величина "K" должна быть целой константой без знака. Она обязана присутствовать в спецификациях "F", "E", "D" и "G", даже если она равна нулю. Присутствие точки также является обязательным. Величина ширины поля "W" должна быть больше или равна "K". Величины "W" и "K" должны либо обе присутствовать, либо обе быть опущены.

Правило 4. В спецификации преобразования "NC1C2C3...Cn" после кода спецификации "H" должно присутствовать ровно "N" символов. В этой спецификации может присутствовать любой символ кода, принятого в CP/M 2.2 (спецификация в виде буквенно-цифрового литерала подчиняется тому же правилу).

Правило 5. В масштабном коэффициенте вида "MP" величина "M" должна быть целой константой со знаком или без него, лежащей в диапазоне от минус 127 до +127 включительно. Масштабный коэффициент применяется только к спецификациям "F", "E", "D" и "G". Будучи один раз определен, масштабный коэффициент действует на все последующие спецификации из списка спецификаций. Действие этого масштабного коэффициента прекращается, когда встречается другой масштабный коэффициент. Для того, чтобы восстановить нулевой масштабный коэффициент, надо явно определить его в виде "0P". Повторная форматизация не оказывает влияния на масштабный коэффициент.

Правило 6. Не допускается использование коэффициента повторения перед спецификациями "H" и перед спецификацией в виде буквенно-цифрового литерала, за исключением тех случаев, когда эти спецификации заключены в скобки и рассматриваются как повторяемая группа спецификаций.

Правило 7. Если в операторе ввода/вывода имеется список ввода/вывода, то список спецификаций содержит как минимум одну спецификацию, отличную от спецификации "H", "X", "P" и буквенно-цифровых литералов.

Правило 8. Список спецификаций в массиве должен быть построен так же, как и список спецификаций в операторе "FORMAT", включая открывающую и закрывающую скобки. При использовании идентификатора массива со списком спецификаций вместо оператора "FORMAT" этот идентификатор не должен быть индексирован.

#### 5.4.7.2. Правила для ввода

Правило 1. Внешнему полю вводимых данных, содержащему отрицательную величину, должен предшествовать знак "-". Наличие знака "+" перед записью положительного значения не является обязательным.

Правило 2. Внешнее поле, преобразование которого при вводе выполняется под управлением спецификации "I", должно иметь формат целой константы.

Правило 3. Внешнее поле, преобразование которого при вводе выполняется спецификацией "F", "E" или "G", должно иметь формат целой или вещественной константы, или константы с двойной точностью. Оно может содержать десятичную точку и/или поле показателя степени типа "E" или "D".

Правило 4. Если внешнее поле содержит десятичную точку, то фактический размер дробной части поля, указываемый этой точкой, переопределяет для этого поля спецификацию "K" из соответствующей спецификации действительного преобразования.

Правило 5. Если внешнее поле содержит показатель степени, это вызывает игнорирование масштабного коэффициента (если он имеется) при выполнении преобразования этого поля.

Правило 6. Спецификация длины поля должна быть достаточно велика для того, чтобы охватить кроме последовательности цифровых символов внешнего поля и другие символы, которые могут в нем присутствовать (алгебраический знак, десятичная точка и/или признак показателя степени).

#### 5.4.7.3. Правила для вывода

Правило 1. Список спецификаций не должен задавать вывод большего числа символов, чем их может содержать внешняя запись.

Правило 2. Спецификация длины поля "W" должна быть достаточно велика, чтобы охватить все символы, которые могут быть сформированы при выводе, включая алгебраический знак, десятичную точку и признак показателя степени. Например, спецификация длины поля в спецификации преобразования "E" должна быть достаточно велика, чтобы поле вмещало (K+7) символов.

Правило 3. Первый символ записи, выводимый на устройство печати или на консоль, используется для управления кареткой. Этот символ никогда не печатается. Первый символ такой записи должен быть пробелом, нулем, единицей, знаком денежной единицы или знаком "+". Любой другой символ, находящийся на этом месте, считается пробелом и исключается из записи.

## 5.5. Операторы описания

Эти операторы являются невыполняемыми операторами. Они обеспечивают компилятор информацией, необходимой для правильного размещения переменных и массивов, задания их начальных значений, а также определяют другие характеристики символических имен, используемых в программе.

### 5.5.1. Операторы описания типа

Операторы описания типа явным образом определяют тип указанных идентификаторов.

Оператор описания типа имеет следующий вид:

TYP V1,V2,...,Vn

где

|  |  |  |
| --- | --- | --- |
| TYP | - | один из следующих описателей типа данных: BYTE, LOGICAL, LOGICAL\*1, LOGICAL\*2, INTEGER, INTEGER\*1, INTEGER\*2, INTEGER\*4, REAL, REAL\*4, REAL\*8, DOUBLE PRECISION; |
| V1,V2,...,Vn | - | идентификаторы переменной, массива, оператора-функции или подпрограммы-функции, или описатель массива. |

Оператор описания типа вызывает присваивание идентификатору определенного типа. Он переопределяет тип данных, неявно определяемый первой буквой идентификатора.

Оператор описания типа может определять массивы включением в его список описателей массивов. В каждом программном модуле идентификатор массива может появляться только один раз в описателе массива.

Идентификатору может приписываться справа указатель длины типа данных, имеющий вид "\*S" ("S" - одна из допустимых длин для объявленного типа данных). Такая запись переопределяет характеристику длины элемента, к которому она относится.

Например:

INTEGER\*2 L, I, J, M12\*1, S, IVEN\*1(10)

REAL\*8 ZX1, ZWX, WX2\*4, WX4, WX1\*8

Операторы описания типа должны предшествовать всем выполняемым операторам. Они обязаны указываться до первого употребления любых идентификаторов, которые они определяют.

Тип данных каждого идентификатора может быть описан явным образом только один раз.

Операторы описания типа не должны иметь меток.

Например:

INTEGER CART, MAS(4,4), RES

REAL MES, ITEM

LOGICAL OPTION

### 5.5.2. Оператор "DIMENSION"

Оператор "DIMENSION" определяет размерность массива, указывая число элементов в каждом измерении.

Оператор "DIMENSION" имеет следующий вид:

DIMENSION A1(D1),A2(D2),...,An(Dn)

где

|  |  |  |
| --- | --- | --- |
| A1,A2,...,An | - | идентификаторы массива; |
| D1,D2,...,Dn | - | значения индекса. |

Каждое выражение "Ai(Di)" (i=1,2,...,n) представляет собой описатель массива.

Оператор "DIMENSION" отводит некоторое число областей памяти по одной для каждого элемента каждого массива, указанного в операторе. Каждая область памяти состоит из одного, двух, четырех или восьми байтов в зависимости от типа данных массива. Общее число областей памяти, отводимых для массива, равно произведению всех значений индексов из описания этого массива. Например, запись

DIMENSION COUNT(4,4), MAS(5,5,5)

определяет массив "COUNT", как 16 вещественных элементов по два слова каждый, и массив "MAS", как имеющий 125 целых элементов по одному слову каждый.

Описатели массивов могут также встречаться в операторах описания типа и "COMMON", но внутри одного программного модуля идентификатор массива может встретиться только в одном описании массива.

Оператор "DIMENSION" не должен иметь метки.

Например:

DIMENSION BAL(7,8,12)

DIMENSION A(4,4,4),B(2,19),C(50)

DIMENSION MAS(3,7,6)

### 5.4.3. Оператор "COMMON"

Оператор "COMMON" определяет одну или более непрерывных областей, которые называются общими блоками (блоками COMMON). Блоку COMMON в операторе "COMMON" может быть присвоено имя. Такие блоки называются именованными COMMON. Если имя не задано, блок называется безымянным блоком COMMON. Оператор "COMMON" также определяет порядок следования переменных и массивов, составляющих каждый общий блок.

Допускается обращение к данным общего блока из различных подпрограмм по одному и тому же идентификатору блока.

Оператор "COMMON" имеет следующий вид:

COMMON [/[A1]/K1] [[,]/[A2]/К2]...

где

|  |  |  |
| --- | --- | --- |
| Ai (i=1,2,...) | - | идентификатор, являющийся именем общего блока, или пробел. Если первый идентификатор опущен, то первая пара дробных черт также может быть опущена; |
| Ki (i=1,2,...) | - | список идентификаторов переменных, идентификаторов массивов и описателей массивов, разделенных запятыми. |

Идентификатор общего блока в выполняемой программе может быть таким же, как идентификатор переменной или массива, но не должен совпадать с идентификатором функции или подпрограммы или с идентификатором точки входа в функцию или подпрограмму.

При компоновке программы все общие блоки, имеющие одно и то же имя в разных подпрограммах, будут размещены в одной и той же области памяти.

Ввиду того, что занесение элементов в общий блок выполняется последовательно один за другим, то элементы, объявленные в операторе "COMMON" в одной программе, должны соответствовать по типу данных элементам, помещаемым в общий блок в другой программе. Например, если одна программная единица содержит оператор:

COMMON COUNT

а другая программная единица содержит оператор:

COMMON MORES

то может получиться некорректный результат, так как однословная переменная "MORES" поставлена в соответствие вещественной переменной "COUNT", представленной большим количеством слов.

Примеры:

|  |  |
| --- | --- |
| Вызывающая программа | Подпрограмма |
| COMMON HET,X/BLK/ITEM,Q | SUBROUTINE SUM |
| . | COMMON/BLK/LERO,R//A,B |
| . | . |
| . | . |
| CALL SUM | RETURN |
| . | END |
| . |  |

Оператор "COMMON" в вызывающей программе помещает "HET" и "X" в неименованный общий блок и помещает "ITEM" и "Q" в именованный блок "BLK". Операторы "COMMON" в подпрограмме устанавливают соответствие между "A" и "B", с одной стороны, и "HET" и "X", с другой стороны, в неименованном блоке, а также между "LERO" и "R", с одной стороны, и "ITEM" и "Q", с другой стороны, в общем блоке "BLK".

#### 5.5.3.1. Неименованный и именованный "COMMON"

Во всей выполняемой программе может быть только один неименованный общий блок. Операторы "COMMON" могут быть использованы для построения любого числа именованных общих блоков.

Переменным из именованного и неименованного блоков можно присваивать начальные значения оператором "DATA".

#### 5.5.3.2. Операторы "COMMON" с описателями массивов

В операторе "COMMON" могут использоваться описания массивов. Идентификаторы массивов не должны быть индексированы (не допускается приписывание к общему блоку отдельных элементов массива). В программном модуле идентификатор массива может встретиться только в одном описании массива.

### 5.5.4. Оператор "EQUIVALENCE"

Оператор "EQUIVALENCE" описывает два или более объектов как размещаемые (полностью или частично) в одной и той же области памяти. Операторы "EQUIVALENCE" работают с элементами, находящимися в одном и том же программном модуле.

Оператор "EQUIVALENCE" имеет следующий вид:

EQUIVALENCE (K1)[,(K2)]...

где

|  |  |  |
| --- | --- | --- |
| Кi (i=1,2,...) | - | список переменных и элементов массивов, разделенных запятыми. В каждом списке должны присутствовать как минимум два объекта. |

Оператор "EQUIVALENCE" вызывает такое распределение памяти, что все элементы из списка в скобках будут размещаться начиная с одной и той же ячейки оперативной памяти. Необходимо заметить, что если установить эквивалентность целой переменной с какой-нибудь вещественной переменной, то при этом целая переменная отображается на старшее слово области вещественной переменной. Смешивание подобным образом различных типов данных является допустимым. Несколько элементов одного типа данных могут отображаться на память, отведенную одному элементу, имеющему тип данных более высокого ранга.

Например:

DOUBLE PRECISION DEN

INTEGER\*2 IAS(4)

EQUIVALENCE (DEN,IAS(1))

Оператор "EQUIVALENCE" вызывает такое распределение памяти, при котором четыре элемента массива занимают ту же самую область памяти, что и переменная с двойной точностью "DEN".

Оператор "EQUIVALENCE" может также использоваться для указания эквивалентности идентификаторов переменных. Например, оператор:

EQUIVALENCE (F1,F2,F3)

устанавливает эквивалентность идентификаторов "F1", "F2", "F3" при условии, что они имеют одинаковый тип данных.

Оператор "EQUIVALENCE" в подпрограмме не должен содержать формальных параметров.

Примеры:

EQUIVALENCE (A,B),(B,C) выполняется так же, как оператор

"EQUIVALENCE (A,B,C)";

EQUIVALENCE (B(1),X), (B(2),Y), (B(3),Z)

#### 5.5.4.1. Установление эквивалентности массивов

При установлении эквивалентности между двумя элементами разных массивов оператор "EQUIVALENCE" также устанавливает эквивалентность между остальными соответствующими элементами этих двух массивов. Таким образом, если первые элементы двух массивов одной размерности эквивалентны, то оба массива целиком отображаются на одну область памяти.

Оператор "EQUIVALENCE" не может выделить одну и ту же область памяти двум или нескольким элементам одного и того же массива. Запрещаются также любые присваивания областей памяти, нарушающие нормальное последовательное запоминание элементов массива в памяти (например, установление эквивалентности между двумя первыми элементами разных массивов, а затем попытка установления эквивалентности между вторым элементом одного массива и шестым элементом другого).

В операторе "EQUIVALENCE" (и только в нем) допускается указание элемента массива по одному индексу (т.е. последовательному номеру элемента), даже если этот массив был определен как многомерный.

Например, операторы:

DIMENSION TOM(2,2),TER(2,2,2)

EQUIVALENCE (TOM(4),TER(7))

вызовут отображение массива "TOM" на часть области памяти, предназначенной для массива "TER", как показано в таблице 4.

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| Массив "TER" | | Массив "TOM" | |
| Элемент массива | Номер элемента | Элемент массива | Номер элемента |
| TER(1,1,1) | 1 |  |  |
| TER(2,1,1) | 2 |  |  |
| TER(1,2,1) | 3 |  |  |
| TER(2,2,1) | 4 | ТОМ(1,1) | 1 |
| TER(1,1,2) | 5 | ТОМ(2,1) | 2 |
| ТER(2,1,2) | 6 | ТОМ(1,2) | 3 |
| TER(1,2,2) | 7 | ТОМ(2,2) | 4 |
| TER(2,2,2) | 8 |  |  |

Таблица 4 показывает, что операторы

EQUIVALENCE (TOM(1),TER(4)) и

EQUIVALENCE (TER(1,2,2),TOM(4))

в результате дают одно и то же размещение двух массивов.

#### 5.5.4.2. Взаимодействие операторов "COMMON" и "EQUIVALENCE"

При установлении эквивалентности между некоторым элементом и объектом, хранящимся в общем блоке, общий блок может быть расширен за его первоначальные границы. Это делается только за счет расширения области за последний элемент предварительно построенного общего блока (но не за счет помещения дополнительной области перед первым элементом существующего общего блока).

Пример допустимого расширения общего блока:

DIMENSION C(4),E(6)

COMMON C

EQUIVALENCE (C(2),E(1))

Существующий общий блок Область расширения

C(1) C(2) C(3) C(4)

E(1) E(2) E(3) E(4) E(5) E(6)

Пример недопустимого расширения общего блока:

DIMENSION C(4), E(6)

COMMON C

EQUIVALENCE (C(2),E(3))

Область Существующий Область

расширения общий блок расширения

C(1) C(2) C(3) C(4)

E(1) E(2) E(3) E(4) E(5) E(6)

Если два элемента приписаны одному и тому же общему блоку или разным общим блокам, между ними нельзя устанавливать эквивалентность.

### 5.5.5. Оператор "EXTERNAL"

Оператор "EXTERNAL" позволяет использовать идентификаторы внешних процедур (функций, подпрограмм и библиотечных функций на языке ФОРТРАН) в качестве фактических параметров для других подпрограмм.

Оператор "EXTERNAL" имеет следующий вид:

EXTERNAL V1,V2,...,Vn

где

|  |  |  |
| --- | --- | --- |
| Vi (i=1,2,...,n) | - | идентификатор подпрограммы или идентификатор формального параметра, который поставлен в соответствие идентификатору подпрограммы. |

Оператор "EXTERNAL" объявляет каждый идентификатор из списка как идентификатор внешней процедуры. Такой идентификатор затем может использоваться как фактический параметр для подпрограммы. После этого подпрограмма может использовать идентификатор соответствующего формального параметра для обращения к функции или в операторе "CALL".

### 5.5.6. Оператор "DATA"

Оператор присваивания начальных значений "DATA" позволяет выполнить присваивание начальных значений переменным и элементам массива еще до выполнения программы. Оператор "DATA" имеет следующий вид:

DATA K1/CONST1/[[,]К2/CONST2/]...

где

|  |  |  |
| --- | --- | --- |
| Кi (i=1,2,...) | - | список, состоящий из одного или более идентификаторов переменных, идентификаторов массивов или идентификаторов элементов массивов. Все эти идентификаторы отделяются друг от друга запятыми. Индексные выражения должны быть константами; |
| CONSTi (i=1,2,...) | - | список констант. Константы имеют одну из следующих форм:  V или n\*V |
| n |  | ненулевая целая константа без знака, которая указывает, сколько раз значение "V" должно быть присвоено последовательным элементам из соответствующего списка. |

Оператор "DATA" вызывает присваивание постоянных значений из каждого списка "CONST" элементам из предыдущего списка "К". Присваивание значений выполняется в том порядке, в каком они расположены.

Если в операторе "DATA" встречается неиндексированный идентификатор массива, то выполняется присваивание значений каждому элементу этого массива. Таким образом, соответствующий список констант должен содержать достаточное количество величин, чтобы заполнить весь массив. Заполнение элементов массива выполняется в порядке индексной прогрессии.

Если выполняется присваивание текстовых констант переменной или элементу массива, число байтов, которое может быть присвоено, зависит от типа переменной. Если число символов в текстовой константе или в буквенно-цифровом литерале меньше, чем число байтов, отведенных переменной или элементу массива, то эта константа дополняется справа пробелами. Если число символов в текстовой константе больше, чем максимальное число, которое может быть запомнено, лишние правые символы не будут использованы.

Число констант в списке констант должно в точности соответствовать числу объектов, определенных в предшествующем списке идентификаторов. Типы данных и типы соответствующих им объектов должны быть согласованы (за исключением буквенно-цифровых и шестнадцатеричных данных).

Шестнадцатеричные данные запоминаются так же, как текстовые, но они дополняются нулями слева (а не пробелами справа).

Пример

INTEGER B(10), ARNO

DATA B,ARNO,STR/10\*0,7,'\*\*\*\*'/

Оператор "DATA" присваивает нулевые значения всем десяти элементам массива "B", значение "7" - переменной "ARNO" и код символа "\*" в каждый из четырех байтов, выделенных для вещественной переменной "STR".

### 5.5.7. Оператор "PROGRAM"

Оператор "PROGRAM" описывает идентификатор программы.

Оператор "PROGRAM" имеет следующий вид:

PROGRAM NAM

где

NAM - идентификатор.

Оператор "PROGRAM" (если он есть) обязан быть первым оператором программы, его использование необязательно. Идентификатор программы не должен совпадать ни с каким идентификатором внутри программы, а также ни с каким идентификатором подпрограммы, точки входа или общего блока этой программы.

# 6. Подпрограммы

Все подпрограммы, написанные на языке ФОРТРАН, делятся на два общих класса: подпрограммы, написанные пользователем и подпрограммы, предоставляемые системой ФОРТРАН.

Подпрограммы также делятся на следующие категории: арифметические операторы-функции, подпрограммы-функции и подпрограммы.

## 6.1. Параметры подпрограмм

Параметры подпрограмм описываются - в виде формальных параметров и в виде фактических параметров.

Формальные параметры используются в определении подпрограммы для представления соответствующих фактических параметров. Формальные параметры присутствуют в операторах "FUNCTION" и "SUBROUTINE" или в определении арифметического оператора-функции как неиндексированные идентификаторы переменных. Фактические параметры присутствуют в обращении к функции или в операторе "CALL" и обеспечивают фактические значения, которые должны использоваться при вычислениях. Фактические параметры могут быть константами, переменными, массивами, элементами массивов, выражениями или идентификаторами подпрограмм.

Между фактическими и формальными параметрами устанавливается соответствие во время передачи управления в подпрограмму. Фактический и формальный параметры, между которыми устанавливается соответствие, должны иметь одинаковый тип данных. Формальный параметр, описанный как массив, может быть поставлен в соответствие массиву или элементу массива.

Если фактическим параметром является константа, выражение или идентификатор подпрограммы, подпрограмма не должна изменять значение соответствующего формального параметра.

Формальные параметры не должны встречаться в операторах "COMMON", "DATA" и "EQUIVALENCE".

## 6.2. Пользовательские подпрограммы

Передача управления функции (подпрограмме-функции или оператору-функции) осуществляется по обращению к функции, в то время как передача управления подпрограмме осуществляется по оператору "CALL". Обращение к функции представляет собой идентификатор функции со списком параметров, который встречается в выражении. Функция всегда передает результат в вызывающую программу. Как функции, так и подпрограммы могут передавать в вызывающую программу некоторые дополнительные значения, присваивая их своим параметрам. Подпрограмма может обращаться к другим подпрограммам, но она не может прямо или косвенно обращаться к самой себе.

### 6.2.1. Арифметические операторы-функции

Арифметический оператор-функция представляет собой процедуру, определяемую отдельным оператором. Этот оператор похож на арифметический оператор присваивания. Появление обращения к функции вызывает выполнение вычислений и передачу полученного значения в то выражение, в котором встретилось обращение к оператору-функции.

Арифметический оператор-функция определяется следующей конструкцией:

F(P1,P2,...,Pn) = E

где

|  |  |  |
| --- | --- | --- |
| F | - | идентификатор функции; |
| P1,P2,...,Pn | - | список формальных параметров; |
| E | - | выражение. |

Выражение представляет собой арифметическое выражение, определяющее вычисления, которые выполняются оператором-функцией.

Обращение к оператору-функции имеет следующий вид:

F(Q1,Q2,...,Qn)

где

|  |  |  |
| --- | --- | --- |
| F | - | идентификатор функции; |
| Q1,Q2,...,Qn | - | список фактических параметров. |

При появлении в выражении оператора-функции выполняется подстановка фактических параметров на место соответствующих им формальных параметров в определении оператора-функции. После этого выполняется вычисление выражения и полученное значение используется для вычисления выражения, содержащего обращение к функции.

Тип результата оператора-функции определяется либо неявным образом по первой букве идентификатора оператора-функции, либо явным образом в операторе описания типа этого имени.

Формальные параметры в определении арифметического оператора-функции служат только для указания количества, порядка расположения и типа фактических параметров. Те же самые имена могут использоваться для представления любых других объектов в любом месте данного программного модуля. Необходимо заметить, что за исключением типа данных описывающая информация (такая, как расположение в общем блоке или описание массива), относящаяся к такому объекту, никак не относится к формальным параметрам оператора-функции. Идентификатор арифметического оператора-функции не может использоваться для представления никакого другого объекта в той же программной единице.

Выражение в определении оператора-функции может содержать обращения к функциям. Если в этом выражении встречается обращение к другому оператору-функции, то необходимо, чтобы другой оператор-функция был определен ранее.

Все обращения к арифметическому оператору-функции должны быть расположены в той же программной единице, что и определение этой функции.

Обращение к оператору-функции может быть либо частью выражения, либо самим выражением. Оно не должно использоваться в качестве левой части в операторе присваивания. Фактические параметры по своему количеству, порядку расположения и типу должны соответствовать формальным параметрам. Перед любым обращением к арифметическому оператору-функции фактическим параметрам должны быть присвоены значения.

Примеры определения арифметических операторов-функций:

VALUE(RADIUS)=4.189\*RADIUS\*\*3

SER(X)=(EXP(X)-EXP(-X))\*2.5

WEG(A,B,C,5.0)=(A+B+C)/5.0

последняя запись недопустима. Запрещается использовать константу в качестве формального параметра.

Обращения к арифметическим операторам-функциям:

ARC(A,B,C)=(A+B+C)/5.0 ; Определение

VEROC=ARC(TECO1,TECO2,XLEB)

IF (ARC(P,Q,D).LT.ARC(X,Y,Z)) GO TO 600

FINIS = ARC(TECO3,TECO4,MET2)

последняя запись неверна. Тип данных третьего фактического параметра не совпадает с типом данных формального параметра.

### 6.2.2. Подпрограмма-функция

Подпрограмма-функция представляет собой программную единицу, начинающуюся с оператора "FUNCTION", за которым идет последовательность операторов, определяющих вычислительную процедуру. Передача управления в подпрограмму-функцию осуществляется по обращению к функции, а возврат управления в вызывающий программный модуль осуществляется по оператору "RETURN".

Подпрограмма-функция передает в вызывающую программу единственное значение, присваивая это значение имени функции. Тип данных передаваемого значения определяется типом идентификатора функции.

Оператор "FUNCTION" имеет следующий вид:

[TYP]FUNCTION NAM[\*N] [(P1,P2,...,Pn)]

где

|  |  |  |
| --- | --- | --- |
| TYP | - | указатель типа данных имени функции; |
| NAM | - | идентификатор функции; |
| \*N | - | указатель длины типа данных; |
| P1,P2,...,Pn | - | список формальных параметров. |

Обращение к функции, передающее управление подпрограмме-функции, имеет следующий вид:

NAM (Q1,Q2,...,Qn)

где

|  |  |  |
| --- | --- | --- |
| NAM | - | имя функции; |
| Q1,Q2,...,Qn | - | список фактических параметров. |

Когда управление передается в подпрограмму-функцию, в операторе "FUNCTION" происходит замена формальных параметров значениями соответствующих фактических параметров. После этого выполняются операторы подпрограммы. Перед выполнением оператора "RETURN" в подпрограмме-функции идентификатору функции должно быть присвоено значение. После того, как управление вернется в вызывающую программу, значение, присвоенное идентификатору функции, будет доступным для выражения, содержащего обращение к функции, и будет использовано для вычисления значения этого выражения.

Тип данных идентификатора может быть определен либо неявно, либо явно в операторе "FUNCTION", либо явно в операторе описания типа. Тип идентификатора функции, определяемый в подпрограмме-функции, должен быть таким же, как и тип идентификатора функции в вызывающей программе.

Оператор "FUNCTION" должен быть первым оператором в подпрограмме-функции. Он не должен иметь метку.

Подпрограмма-функция не должна содержать операторы "SUBROUTINE", "BLOCK DATA" или другой оператор "FUNCTION", отличный от ее первого оператора.

Пример

FUNCTION ROOT(B)

X=1.0

3 EX=EXP(X)

EPIC=1./EX

ROOT=((EX+EPIC)\*.5+COS(X)-B)/((ЕX-EPIC)\*.5-SIN(X))

IF(ABS(X-ROOT).LT.1E-6) RETURN

X=ROOT

GO TO 3

END

В этом примере подпрограмма-функция использует итерационный метод Ньютона-Рафсона для вычисления корня уравнения:

F(X)=COSH(X)+COS(X)-B=0

значение "B" пересылается в качестве параметра. Итерационная формула для этого корня имеет следующий вид:

X(I+1)=X(I)-(COSH(XI)+COS(XI)-B)/(SINH(XI)-SIN(XI))

Это вычисление повторяется до тех пор, пока разность между "X(I)" и "X(I+1)" не станет меньше, чем 1\*10\*\*(-6). Подпрограмма-функция использует библиотечные функции языка ФОРТРАН "EXP", "SIN ", "COS " и "ABS".

### 6.2.3. Подпрограмма

Подпрограмма является программной единицей, начинающейся с оператора "SUBROUTINE", за которым идет последовательность операторов, определяющих вычислительную процедуру. Передача управления подпрограмме осуществляется по оператору "CALL", а возврат в вызывающую программу - по оператору "RETURN".

Оператор "SUBROUTINE" имеет следующий вид:

SUBROUTINE NAM[(P1,P2,...,Pn)]

где

|  |  |  |
| --- | --- | --- |
| NAM | - | идентификатор подпрограммы; |
| P1,P2,...,Pn | - | список формальных параметров. |

При передаче управления в подпрограмму формальные параметры оператора "SUBROUTINE" заменяются значениями, используемыми как фактические параметры в операторе "CALL". После этого выполняются операторы подпрограммы.

Оператор "SUBROUTINE" обязан быть первым оператором подпрограммы. У него не должно быть метки.

Программа не может содержать операторы "FUNCTION", "BLOCK DATA" или "SUBROUTINE" (за исключением первого оператора подпрограммы).

Пример:

Вызывающая программа

COMMON NFACES,ERCE,VOLUME

ACCEPT 70,NFACES,ERCE

70 FORMAT (I2,F8.5)

CALL OBMNG

TYPE 72,VOLUME

72 FORMAT (' Объем=',F8.5)

STOP

END

Подпрограмма

SUBROUTINE OBMNG

COMMON NFACES, ERCE, VOLUME

CUBED =ERCE\*\*3

GO TO (6,6,6,1,6,2,6,3,6,6,6,4,6,6,6,6,6,6,6,6,6,5,6), NFACES

1 VOLUME = CUBED\*0.11785

RETURN

2 VOLUME = CUBED

RETURN

3 VOLUME = CUBED\*0.47140

RETURN

4 VOLUME = CUBED\*7.66312

RETURN

5 VOLUME = CUBED\*2.18170

RETURN

6 TYPE 25, NFACES

25 FORMAT(' Неправильный многоугольник имеет',I3,' граней.')

RETURN

END

В этом примере подпрограмма вычисляет объем правильного многогранника, заданного числом граней и длиной ребра. Она использует вычисляемый оператор перехода "GO TO" для определения того, является ли данный многогранник тетраэдром, гексаэдром, додекаэдром или икосаэдром, и для передачи управления соответствующей процедуре для вычисления объёма. Если число граней тела отлично от 4, 6, 8, 12 или 20, подпрограмма выводит сообщение об ошибке на консоль.

### 6.2.4. Подпрограмма данных "BLOCK DATA"

Подпрограмма данных используется для присваивания начальных значений объектам, находящимся в именованных общих блоках, одновременно с определением этих блоков. Она начинается с оператора "BLOCK DATA", за которым следует ряд операторов описания.

Оператор "BLOCK DATA" имеет следующий вид:

BLOCK DATA [NAM]

где

|  |  |  |
| --- | --- | --- |
| NAM | - | идентификатор. |

В подпрограмме данных разрешено использовать следующие операторы: операторы описания типа, операторы "IMPLICIT", "DIMENSTION", "COMMON", "EQUIVALENCE" и "DATA".

Операторы описания в подпрограмме данных определяют общие блоки, размещают переменные и массивы в этих блоках и присваивают начальные значения этим элементам.

Оператор "BLOCK DATA" обязан быть первым оператором подпрограммы данных. Он не должен иметь метку.

Подпрограмма данных не должна содержать выполняемых операторов.

Если в подпрограмме данных задаются начальные значения некоторого элемента именованного общего блока, то в подпрограмме должен присутствовать полный набор операторов описания, задающих весь блок, даже если некоторые из элементов этого блока не появляются в операторе "DATA". Одной и той же подпрограммой могут быть определены несколько общих блоков.

Пример

BLOCK DATA

INTEGER X,Y

LOGICAL Q,W

DOUBLE PRECISION D

DIMENSION R(4)

COMMON/BLK1/R,X,Q,D/BLK2/W,Y,Z

DATA R/2.0,3\*4.0/,Q/.FALSE./,D/0.317638D-7/,W

1/.TRUE./Z/3.5/

END

### 6.2.5. Библиотечные функции языка ФОРТРАН

Перечень библиотечных функций языка ФОРТРАН приведен в приложении 2.

Обращения к библиотечным функциям записываются в таком же виде, как и обращения к пользовательским функциям. Например:

R = ABS(X-1)

выполняет вычисление абсолютного значения "X-1" и присваивает полученное значение переменной "R".

В приложении 2 также указывается тип результата каждой библиотечной функции и тип фактических параметров. Параметры, передаваемые в эти функции, не должны быть идентификаторами массивов или идентификаторами подпрограмм.

# Приложение 1 Основные сведения о языке ФОРТРАН

**1. Операции**

Операции каждого типа приведены в таблице в порядке убывания приоритета.

Операции языка ФОРТРАН

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип операции** | **Знак операции и описание** | | **Операнды** |
| Арифметический | \*\* | возведение в степень | Арифметические или логические константы и выражения |
| \*,/ | умножение, деление |
| +,- | сложение, вычитание, унарные плюс, минус |
| Отношения | .GT. | больше | Арифметические или логические константы, переменные и выражения (все операции отношения имеют одинаковый приоритет) |
| .GE. | больше или равно |
| .LT. | меньше |
| .LE. | меньше или равно |
| .EQ. | равно |
| .NE. | не равно |
| Логический | .NOT. | .NOT.A истинно тогда и только тогда, когда "A" ложно | Логические или целые константы, переменные и выражения |
| .AND. | A.AND.B истинно тогда и только тогда, когда оба операнда "A" и "B" истинны |
| .OR. | A.OR.B истинно тогда и только тогда, когда либо "A", либо "B", либо "A", либо "B", либо и "A" и "B" истинны |
| .XOR. | A.XOR.B истинно тогда и только тогда, когда "A" истинно, а "B" ложно или когда "B" истинно, а "A" ложно |

**2. Операторы**

Ниже приведен перечень операторов языка ФОРТРАН, дан основной формат каждого оператора и краткие пояснения.

ACCEPT[[1]](#footnote-1)

|  |  |
| --- | --- |
| - | см. READ, форматный последовательный ввод; |
| - | см. READ, ввод под управлением списка. |

Арифметическое/логическое присваивание W=A

|  |  |  |
| --- | --- | --- |
| W | - | идентификатор переменной или идентификатор элемента массива; |
| A | - | выражение. |
| Переменной W присваивается значение арифметического или логического выражения A. | | |

Арифметический оператор-функция F(P1,P2,...,Pn)=E

|  |  |  |
| --- | --- | --- |
| F | - | идентификатор; |
| Pi (i=1,2,...,n) | - | идентификатор; |
| E | - | выражение. |
| Строится функция пользователя, в которой "P" является формальным параметром. При обращении к ней выполняется вычисление выражения с использованием значений фактических параметров. | | |

ASSIGN S TO W

|  |  |  |
| --- | --- | --- |
| S | - | метка выполняемого оператора; |
| W | - | идентификатор целой переменной. |
| Устанавливается соответствие между меткой оператора "S" и целой переменной "W" для последующего использования в операторе перехода по предписанию GO TO. | | |

BACKSPACE U

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа. |
| Возврат к предыдущей записи в файле, открытом в текущий момент времени на логическом устройстве "U". | | |

BLOCK DATA [NAM]

|  |  |  |
| --- | --- | --- |
| NAM | - | идентификатор |
| Определяет следующую за ним подпрограмму как подпрограмму данных. | | |

CALL S[(P1,P2,…,Pn]

|  |  |  |
| --- | --- | --- |
| S | - | идентификатор подпрограммы; |
| Pi (i=1,2,...n) | - | выражение, идентификатор процедуры или идентификатор массива. |
| Вызывается подпрограмма с идентификатором "S". При этом передаются фактические параметры "P" для замены формальных параметров в определении подпрограммы. | | |

COMMON [/[A1]/ K1][[,]/[A2]/K2]...

|  |  |  |
| --- | --- | --- |
| Ai (i=1,2,...) | - | идентификатор общего блока; |
| Ki (i=1,2,...) | - | список, состоящий из одного или нескольких идентификаторов переменных, идентификаторов массивов или описателей массивов, разделенных запятыми. |
| Резервирует участок оперативной памяти, помеченный идентификатором блока COMMON, для размещения переменных, связанных с этим блоком. | | |

CONTINUE

|  |
| --- |
| Передает управление на следующий выполняемый оператор. |

DATA К1/C1/[[,] К2/C2/]...

|  |  |  |
| --- | --- | --- |
| Ki (i=1,2,...) | - | список, состоящий из одного или нескольких идентификаторов переменных, идентификаторов массивов или идентификаторов элементов массивов, разделенных запятыми. Индексные выражения должны быть константами; |
| Ci (i=1,2,...) | - | список, состоящий из одной или нескольких констант разделенных запятыми, причем каждой константе может предшествовать необязательная форма j\* (j - ненулевая целая константа без знака). |
| Вызывает присваивание начальных значений из списка констант соответствующим элементам списка идентификаторов переменных. | | |

DIMENSION A1(D1) ,A2(D2),...,An(Dn)

|  |  |  |
| --- | --- | --- |
| Ai(Di) (i=1,2,...,n) | - | описатель массива. |
| Определяет объём памяти, требующейся для массивов. | | |

DO S I=E1,E2[,E3]

|  |  |  |
| --- | --- | --- |
| S | - | метка последнего выполняемого оператора в цикле; |
| I | - | идентификатор переменной цикла; |
| E1,E2[,E3] | - | целые положительные переменные или беззнаковые целые константы. |
| Выполняются следующие действия:   1. установка I=E1; 2. выполнение операторов до оператора с номером "S" включительно; 3. вычисление I=I+E3; 4. повторение пунктов 2 и 3 "M" раз. (M = MAX(1,INT((E2-E1)/E3)+1)) | | |

END

|  |
| --- |
| Завершает программную единицу. |

END FILE U

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или константа. |
| На логическое устройство "U" записывается признак конца файла. | | |

END=S, ERR=S

|  |  |  |
| --- | --- | --- |
| S | - | метка выполняемого оператора. |
| Передача управления по концу файла или ошибочному состоянию, являющаяся необязательным элементом любого оператора ввода/вывода. Этот элемент позволяет программе перейти к оператору с номером "S" при условии конца файла (END=) или при ошибочном состоянии (ERR=). | | |

EQUIVALENCE (K1),(K2),...,(Kn)

|  |  |  |
| --- | --- | --- |
| Ki (i=1,2,...,n) | - | список, состоящий из двух или более идентификаторов переменных, идентификаторов массивов или идентификаторов элементов массивов, разделенных запятыми. Индексные выражения должны быть константами. |
| Каждый из идентификаторов, принадлежащих списку, размещается в одной и той же области памяти. | | |

EXTERNAL V1,V2,...,Vn

|  |  |  |
| --- | --- | --- |
| Vi (i=1,2,...,n) | - | идентификатор подпрограммы. |
| Описывает идентификаторы, как идентификаторы внешней процедуры. | | |

FORMAT (спецификация преобразования)

|  |
| --- |
| Описывает формат, в котором должна быть передана одна или несколько записей. Оператор должен иметь метку. |

[TYP] FUNCTION NAM[\*N] [(P1,P2,...,Pn)]

|  |  |  |
| --- | --- | --- |
| TYP | - | указатель типа данных; |
| NAM | - | идентификатор; |
| \*N | - | указатель длины данных; |
| Pi (i=1,2,...,n) | - | идентификатор. |
| Является первым оператором подпрограммы-функции, указывая ее идентификатор и формальные параметры "Pi". Может включать необязательный указатель типа. | | |

GO TO S

|  |  |  |
| --- | --- | --- |
| S | - | метка выполняемого оператора. |
| Безусловный переход. Передает управление оператору с меткой "S". | | |

GO TO (K1,К2,...,Kn),M

|  |  |  |
| --- | --- | --- |
| К1,К2,...,Кn | - | список, состоящий из одной или нескольких меток выполняемых операторов, разделенных запятыми; |
| M |  | целое выражение. |
| Вычисляемый оператор перехода. Передает управление оператору с меткой, определяемой значением выражения "M". Если "M" равно 1, управление передается оператору с первой меткой из списка. Если "M" равно 2, управление передается оператору со второй меткой и т.д. Если "M" меньше единицы или больше, чем имеющееся число меток, передача управления происходит на следующий выполняемый оператор. | | |

GO TO V [[,](K1,K2,...,Kn)]

|  |  |  |
| --- | --- | --- |
| V | - | идентификатор целой переменной; |
| К1,К2,...,Кn | - | список, состоящий из одной или нескольких меток выполняемых операторов, разделенных запятыми. |
| Переход по предписанию. Передает управление оператору, который был последним установлен в соответствие переменной "V" оператором "ASSIGN". | | |

IF (M) K1,K2,K3

|  |  |  |
| --- | --- | --- |
| M | - | выражение; |
| K1,K2,K3 | - | метки выполняемых операторов. |
| Арифметический оператор условного перехода. Передает управление оператору с номером K1, K2 или K3 в зависимости от значения выражения.  Если значение выражения меньше нуля, то выполняется переход на "K1".  Если значение выражения равно нулю, то выполняется переход на "K2".  Если значение выражения больше нуля, то выполняется переход на "K3". | | |

IF (E) S

|  |  |  |
| --- | --- | --- |
| E | - | выражение; |
| S | - | любой выполняемый оператор, за исключением оператора "DO" и логического условного оператора "IF". |
| Логический условный оператор. Выполняет оператор "S" при условии истинности выражения. | | |

IMPLICIT T1(D1),T2(D2),...

|  |  |  |
| --- | --- | --- |
| Тi (i=1,2,...) | - | тип данного; |
| Di (i=1,2,...) | - | список букв, разделенных запятыми или знаком минус. |

PAUSE [N]

|  |  |  |
| --- | --- | --- |
| N | - | последовательность от 1 до 6 любых символов. |
| Приостанавливает выполнение программы и выводит на консоль строку "N", если "N" задано. | | |

PROGRAM NAM

|  |  |  |
| --- | --- | --- |
| NAM | - | идентификатор. |
| Определяет идентификатор программы. | | |

READ (U,F[,END=S1] [,ERR=S2])[K]

READ F[,K]

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа; |
| F | - | метка оператора "FORMAT" или идентификатор массива; |
| S1,S2 | - | метки выполняемых операторов; |
| K | - | список ввода. |
| Форматный последовательный ввод. Считывает одну или более логических записей с устройства "U" и присваивает элементам списка ввода значения, преобразованные в соответствии со спецификацией формата в операторе "F". | | |

READ (U[,END=S1][,ERR=S2])[K]

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа; |
| S1,S2 | - | метки выполняемых операторов; |
| K | - | список ввода. |
| Неформатный последовательный ввод. Считывает одну неформатную запись с устройства "U" и присваивает значения элементам списка. | | |

RETURN

|  |
| --- |
| Возвращает управление в вызывающую программу из выполняемой подпрограммы. |

REWIND U

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа. |
| Вызывает возврат к началу файла, открытого в настоящий момент на логическом устройстве "U". | | |

STOP [N]

|  |  |  |
| --- | --- | --- |
| N | - | последовательность от 1 до 6 символов. |
| Прекращает выполнение программы и выводит на консоль строку "N", если "N" задано. | | |

SUBROUTINE NAM [(P1,P2,...,Pn)]

|  |  |  |
| --- | --- | --- |
| NAM | - | идентификатор; |
| Pi (i=1,2,...,n) | - | идентификатор. |
| Начинает подпрограмму, указывая ее идентификатор и идентификаторы формальных параметров "Pi". | | |

TYPE[[2]](#footnote-2)

|  |  |
| --- | --- |
| - | см. WRITE, форматный последовательный ввод; |
| - | см. WRITE, ввод под управлением списка. |

TYP V1,V2,...,VN

|  |  |  |
| --- | --- | --- |
| TYP | - | указатель типа данных; |
| Vi (i=1,2,...,n) | - | идентификатор переменной, идентификатор массива, идентификатор функции или точки входа в функцию, или описатель массива. За идентификатором может следовать необязательный указатель длины типа данных (\*n). |
| Идентификаторам "Vi" в данной программной единице присваивается указанный тип. Тип может быть может быть одним из следующих: DOUBLE PRECISION, REAL, REAL\*4, REAL\*8, INTEGER, INTEGER\*1, INTEGER\*2, INTEGER\*4, BYTE, LOGICAL, LOGICAL\*1, LOGICAL\*2 | | |

WRITE (U,F[,ERR=S])[K]

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа; |
| F | - | метка оператора "FORMAT" или идентификатор массива; |
| S | - | метка выполняемого оператора; |
| K | - | список вывода. |
| Форматный последовательный вывод. Записывает на устройство "U" одну или несколько записей, содержащих значения элементов списка, преобразованных согласно спецификации формата "F". | | |

WRITE (U[,ERR=S])[K]

|  |  |  |
| --- | --- | --- |
| U | - | целая переменная или целая константа; |
| S | - | метка выполняемого оператора; |
| K | - | список вывода. |
| неформатный последовательный вывод. Записывает на устройство "U" одну неформатную запись, содержащую значения элементов списка. | | |

# Приложение 2 Библиотечные функции языка ФОРТРАН

| **Имя функции** | **Определение** | **Тип** | |
| --- | --- | --- | --- |
| **аргумента** | **функции** |
| **Функции преобразования** | | | |
| ABS(X) | Вещественное абсолютное значение | вещественный | вещественный |
| IABS(X) | Целое абсолютное значение | целый | целый |
| DABS(X) | Абсолютное значение с двойной точностью | с двойной точностью | с двойной точностью |
| FLOAT(I) | Преобразование целой величины в вещественную | целый | вещественный |
| IFIX(X) | Преобразование вещественной величины в целую. IFIX(X) эквивалентна INT(X) | вещественный | целый |
| SNGL(X) | Преобразование величины с двойной точностью в вещественную величину | двойной точности | вещественный |
| DBLE(X) | Преобразование вещественной величины в величину с двойной точностью | вещественный | с двойной точностью |
| **Функции выделения целой части**  Результатом функций выделения целой части является наибольшее целое число, не превышающее по абсолютной величине аргумент функции. Функции присваивается знак аргумента. | | | |
| AINT(X) | Выделение целой части вещественной величины | вещественный | вещественный |
| INT(X) | вещественный | целый |
| IDINT(X) | Выделение целой части величины с двойной точностью | двойной точности | двойной точности |
| **Функции остатка**  Результатом функций остатка является остаток от деления первого аргумента на второй. | | | |
| AMOD(X,Y) | Остаток вещественный | вещественный | вещественный |
| MOD(I,J) | Остаток целый | целый | целый |
| DMOD(X,Y) | Остаток с двойной точностью | двойной точности | двойной точности |
| **Функции выбора максимального значения**  Результатом функции выбора максимального значения является наибольшее значение из аргументов списка, состоящего из двух или более аргументов. | | | |
| AMAX0(I,J,...) | Максимальное вещественное значение из списка целых аргументов | целый | вещественный |
| AMAX1(X,Y,...) | Максимальное вещественное значение из списка вещественных аргументов | вещественный | вещественный |
| MAX0(I,J,...) | Максимальное целое значение из списка целых аргументов | целый | целый |
| MAX1(X,Y,...) | Максимальное целое значение из списка вещественных аргументов | вещественный | целый |
| DMAX1(X,Y,...) | Максимальное значение двойной точности из списка аргументов с двойной точностью | с двойной точностью | с двойной точностью |
| **Функции выбора минимального значения**  Результатом функций выбора минимального значения является наименьшее значение из списка аргументов, состоящего из двух или более аргументов. | | | |
| AMIN0(I,J,...) | Минимальное вещественное значение из списка целых аргументов | целый | вещественный |
| AMIN1(X,Y,...) | Минимальное вещественное значение из списка вещественных аргументов | вещественный | вещественный |
| MIN0(I,J,...) | Минимальное целое значение из списка целых аргументов | целый | целый |
| MIN1(X,Y,...) | Минимальное целое значение из списка вещественных аргументов | вещественный | целый |
| DMIN1(X,Y,...) | Минимальное значение с двойной точностью из списка параметров с двойной точностью | двойной точности | двойной точности |
| **Функции присваивания знака**  Результатом функции присваивания знака является абсолютное значение первого аргумента с присвоенным ему знаком второго аргумента. | | | |
| SIGN(X,Y) | Присваивание знака вещественной величине | вещественный | вещественный |
| ISIGN(I,J) | Присваивание знака целой величине | целый | целый |
| DSIGN(X,Y) | Присваивание знака величине с двойной точностью | с двойной точностью | с двойной точностью |
| **Функции положительной разности**  Результатом функции положительной разности является разность первого аргумента и минимального из двух аргументов. | | | |
| DIM(X,Y) | Положительная разность вещественных величин | вещественный | вещественный |
| IDIM(I,J) | Положительная разность целых величин | целый | целый |
| **Экспоненциальные функции**  Результатом экспоненциальной функции является значение "e", возведенное в степень, равную аргументу. | | | |
| EXP(X) | e\*\*x | вещественный | вещественный |
| DEXP(X) | с двойной точностью | с двойной точностью |
| **Логарифмические функции** | | | |
| ALOG(X) | Натуральный логарифм вещественного аргумента | вещественный | вещественный |
| ALOG10(X) | Десятичный логарифм вещественного аргумента | вещественный | вещественный |
| DLOG(X) | Натуральный логарифм аргумента с двойной точностью | с двойной точностью | с двойной точностью |
| **Квадратный корень** | | | |
| SQRT(X) | Корень квадратный из вещественного аргумента | вещественный | вещественный |
| DSQRT(X) | Корень квадратный из аргумента двойной точности | с двойной точностью | с двойной точностью |
| **Тригонометрические функции** | | | |
| SIN(X) | Синус вещественного аргумента | вещественный | вещественный |
| DSIN(X) | Синус аргумента с двойной точностью | с двойной точностью | с двойной точностью |
| COS(X) | Косинус вещественного аргумента | вещественный | вещественный |
| DCOS(X) | Косинус аргумента с двойной точностью | с двойной точностью | с двойной точностью |
| ATAN(X) | Арктангенс вещественного аргумента | вещественный | вещественный |
| DATAN(X) | Арктангенс аргумента с двойной точностью | с двойной точностью | с двойной точностью |
| ATAN2(X,Y) | Арктангенс вещественного аргумента от "X/Y" | вещественный | вещественный |
| DATAN2(X,Y) | Арктангенс аргумента с двойной точностью от "X/Y" | с двойной точностью | с двойной точностью |
| TANH(X) | Тангенс гиперболический | вещественный | вещественный |

1. Оператор ACCEPT не является стандартным. Оператор является расширением языка в версии FORTRAN DEC и ряда других производителей. [↑](#footnote-ref-1)
2. Оператор TYPE не является стандартным. Оператор является расширением языка в версии FORTRAN DEC и ряда других производителей. [↑](#footnote-ref-2)