Утверждаю
Главный инженер завода "Экситон"
Скоморохов И.М.

МИКРО-ЭВМ
"ЭЛЕКТРОНИКА БК 0010-01"
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
Фокал
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
00009-01.90.01

Руководитель подразделения,
нач. отделения 2 — Косенков С. М.
Руководитель темы,
вед. инженер — Врублевский Ю. Н.
Исполнитель,
вед. инженер — Хамитов А. С.
Нормоконтролер

1984

Аннотация

В настоящем документе содержится описание приёмов работы с микро-ЭВМ "Электроника БК 0010-01" (и её модификациями).

В документе описаны конструкции языка "Фокал": символы, числа, переменные, операторы. По мере чтения пользователь обучается приёмам использования этих конструкций при написании программ.

Приводятся средства и рекомендации по отладке программ.

В конце документа в приложении 2 дано краткое описание языка "Фокал", в приложении 1 – коды ошибок и краткие сведения по их устранению.

Оглавление

АННОТАЦИЯ 2
1. ВВЕДЕНИЕ 6
2. ОСНОВНЫЕ КЛАВИШИ ЭВМ. 6
2.1. Клавиатура микро-ЭВМ 6
2.1.1. Цветовая маркировка 6
2.9. Итоги главы 2 9
2.10. Вопросы к главе 2 9
3. КАК ОБЩАТЬСЯ С ЭВМ? 10
3.1. Очистка экрана. 10
3.2. Курсор. 10
3.3. Первое распоряжение. 11
3.5. Что такое программирование на ЭВМ. 12
3.5.1. Так на каком же языке общаться с ЭВМ? 12
3.6. Итоги главы 3. 13
4. ПЕРВОЕ ЗНАКОМСТВО С "ФОКАЛОМ". 13
4.2. Когда можно вводить команды – операторы в ЭВМ? 14
4.8. Итоги главы 4. 17
4.9. Вопросы к главе 4 18
5. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ, ОПЕРАТОР SET. 18
5.1. Умеет ли ЭВМ считать? 18
5.5. Итоги главы 5. 22
5.6. Вопросы к главе 5 23
6. ОБ ОШИБКАХ 23
6.1.1. Логические ошибки 23
6.1.2. Синтаксические ошибки. 24
6.2. Как исправлять выявленные ошибки? 25
6.3. Редактирование текста. 25
6.4. Итоги главы 6. 29
6.5. Вопросы к главе 6 30
7. ОПЕРАТОРЫ ASK, ERASE, WRITE. 30
7.5. Итоги главы 7. 33
7.6. Вопросы к главе 7 33
8. ВЫВОД ДАННЫХ. ОПЕРАТОР TYPE 34
8.3. Пример печати числа в экспоненциальной форме 37
8.4. На какие элементы списка оператора "T" действует заданный формат? 37
8.6. Итоги главы 8. 39

8.7. Вопросы к главе 8.....39
9. ПОНЯТИЕ ВСТРОЕННОЙ ФУНКЦИИ. 40
9.2. Определение количества целых часов.....40
9.3. Определение количества целых минут.....40
9.4. Вычисление количества секунд.....41
9.5. Вывод результатов.....43
9.7. Итоги главы 9.....44
9.8. Вопросы к главе 9.....44
10. ПРОГРАММНЫЙ РЕЖИМ РАБОТЫ. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ. 45
10.5. Пример 1.....47
10.6. Пример 2.....50
10.7. Пример 3.....50
10.8. Итоги главы 10.....51
10.9. Вопросы к главе 10.....52
11. ПОДПРОГРАММЫ. ОПЕРАТОРЫ DO, RETURN, QUIT.....52
11.6. Пример.....56
11.9. Итоги главы 11.....59
11.10. Вопросы к главе 11.....60
12. ОПЕРАТОР ДЛЯ ОРГАНИЗАЦИИ ЦИКЛОВ FOR 61
12.4. Что является областью действия цикла? 63
12.6. Итоги главы 12.....66
12.7. Вопросы к главе 12.....66
13. УПРАВЛЕНИЕ МАГНИТОФОНОМ, ОПЕРАТОР LIBRARY 66
13.6. Запись информации на МЛ.....67
13.7. Чтение информации с МЛ.....69
13.9. Вывод данных на МЛ.....70
13.10. Чтение данных с МЛ в память ЭВМ. 71
13.11. Итоги главы 13.....72
13.12. Вопросы к главе 13.....72
14. ВСТРОЕННЫЕ ФУНКЦИИ. ОПЕРАТОР X.....73
14.2. Математические функции.....74
14.3. Оператор X. Функция случайных чисел FRAN.....77
14.4. Функция ввода-вывода символьной информации FCHR 78
14.5. Функция управления общей шиной (FX).....81
14.6. Функция, программируемая пользователем (FSBR). 82
14.7. Функция управления положением курсора (FK).....84
14.8. Функции для работы с графической информацией FT и FV. 84
14.9. Функция работы с портом ввода/вывода FP.....85

15. ОПЕРАТОРЫ KILL, VACANT, PASS 88
15.1. Оператор сброса внешних устройств KILL. 88
15.2. Оператор VACANT 88
15.3. Операторы группы PASS 89
16. КАК РЕШАТЬ ЗАДАЧУ НА ЭВМ? 89
16.1. Этап 1. Разработка алгоритма решения 90
16.2. Этап 2. Отладка программы. 92
16.3. Итоги главы 16. 96
16.4. Вопросы к главе 16 96
17. ОТВЕТЫ НА ВОПРОСЫ 97
ПРИЛОЖЕНИЕ 1 100
ПРИЛОЖЕНИЕ 2 102
1. Алфавит 102
2. Ограничители. 102
3. Режимы работы и структура программ. 103
4. Числа 104
5. Переменные 105
6. Массивы. 106
7. Арифметические выражения 107
8. Общий формат оператора языка "Фокал". 108
1. Перечень операторов языка. 108
2. Невыполняемые операторы. 108
3. Вычислительные операторы. 108
4. Операторы управления. 108
5. Операторы ввода – вывода. 108
6. Операторы – директивы 109
7. Встроенные функции. 109
ПРИЛОЖЕНИЕ 3 111
ПРИЛОЖЕНИЕ 4 115
ПРИЛОЖЕНИЕ 5 116

1. Введение

Прежде, чем начинать чтение настоящего руководства, вам необходимо ознакомиться с руководством по эксплуатации, входящим в комплект документов, поставляемых с изделием (документ М1.492.000 РЭ).

Данное руководство рассчитано на людей, не имеющих опыта работы с вычислительной техникой.

Если вы уже составляли небольшие программы для других ЭВМ, но ещё не имеет достаточного опыта в этом, мы рекомендуем вам читать руководство в следующей последовательности:

- главу 2;
- приложение 2;
- итоги каждой из глав руководства.

Если вы уже имеете большой опыт решения задач на ЭВМ и работали с языками программирования высокого уровня, мы рекомендуем вам прочесть:

- главу 2;
- приложение 2

и приступить к решению задач, пользуясь по мере надобности другими материалами, изложенными в руководстве.

При чтении материалов пользуйтесь словарём технических терминов. Словарь приведён в приложении 3.

Руководство содержит 16 глав и 5 приложений, порядок, в котором их необходимо читать, определяется степенью вашей подготовки.

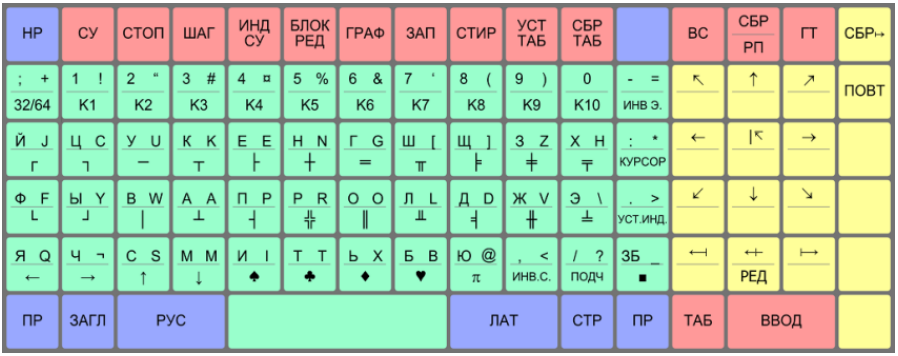
2. Основные клавиши ЭВМ.

2.1. Клавиатура микро-ЭВМ

Клавиатура вашей, микро-ЭВМ может быть плёночной или клавишной. В данном руководстве приведено описание плёночной клавиатуры. Для работы с клавишной клавиатурой ознакомьтесь с таблицей соответствия клавиш в приложении 4.

2.1.1. Цветовая маркировка

Клавиши окрашены в красный, жёлтый, зелёный и синий цвета. Окрасили их для того, чтобы примерно одинаковые по функциональному назначению были одного цвета.



При включении ЭВМ в верхней части экрана появится горизонтальная черта и над ней буквы ЛАТ. Называется она информационной строкой.

2.2. Нажмите на клавишу "РУС" – синяя клавиша внизу клавиатуры.

В информационной строке появились буквы:

```
РУС.
```

2.3. Смысл этой информации такой:

Если в строке изображено слово

```
РУС,
```

то при нажатии на буквенные клавиши в ЭВМ вводятся символы, соответствующие буквам русского алфавита (можно сказать, вводятся русские буквы).

Если в информационной строке содержится

```
ЛАТ,
```

это значит, что ЭВМ воспринимает буквы латинского алфавита.

2.4. Рассмотрим назначение ещё нескольких клавиш синего цвета.

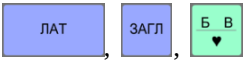
Нажмите клавиши:

\- состояние экрана не меняется

\- на экране появилась заглавная буква "Б"

\- состояние экрана не меняется

\- на экране появилась строчная (прописная) буква "б"



\- на экране появилась латинская заглавная буква "B"

Клавиши "СТР" и "ЗАГЛ" настраивают пульт на ввод либо строчных, либо заглавных букв.

Клавиши синего цвета настраивают клавиатуру на ввод из той или иной группы. Эти клавиши переключают регистры (аналогично регистрам пишущей машинки, которые переключает машинистка).

Поэтому клавиши синего цвета носят название группы регистровых клавиш.

2.5. Об остальных клавишах регистровой группы мы расскажем по ходу дальнейшего изложения, а теперь давайте познакомимся с клавишами зелёного цвета, некоторые из которых мы уже нажимали. Это так называемые алфавитно-цифровые клавиши. В отличии от клавиш предыдущей группы нажатие на них вызывает ввод символа в ЭВМ.

С помощью этих клавиш в ЭВМ вводятся буквы русского алфавита, буквы латинского алфавита, цифры, знаки, специальные символы.

2.6. Рассмотрите внимательно какую-либо из клавиш этой группы, например, клавишу 

Видно, что она состоит из трёх частей:

- верхней левой;
- верхней правой;
- нижней.

Будем их для краткости называть "левой", "нижней", "правой" аналогично, содержащие эти символы регистры будем называть "нижним", "левым", "правши".

2.7. Попытайтесь вывести на экран символ "π". Нажмите клавишу " 

".

не получается?

действительно, выводится символ "@ ".

как быть?

Дело в том, что сейчас клавиатура настроена на ввод символов с "левого" регистра (верхней части клавиши). Чтобы получить нужный вам символ, необходимо переключиться на нижний регистр с помощью клавиши

но не торопитесь!

При работе с этой клавишей есть одна особенность. Клавиша действует (нижний регистр включён) только в "нажатом" состоянии. Стоит отпустить её и произойдёт переключение с нижнего регистра на левый.

Упражнение 1:

запишите на экран символы:

<-, ->, ■

2.8. Попробуем вывести на экран знак "=", Найдите нужную клавишу в правом верхнем углу зелёного поля клавиатуры.

Мы видим, что знак "=" находится в правой части клавиши. Следовательно, как и в предыдущем примере (см. п. 2.7), для его ввода необходимо нажатие регистровой клавиши. Только в этот раз регистровой клавишей будет не "НР", а "ПР", указывающая ЭВМ, что надо читать символы с правой части клавиши.

Клавишу "ПР", как и в предыдущем примере необходимо удерживать во время нажатия символа.

2.9. Итоги главы 2

2.9.1. Клавиши синего цвета – это регистровые клавиши. Они служат для переключения регистров клавиатуры ЭВМ.

2.9.2. Клавиши зелёного цвета – это алфавитно-цифровые клавиши. Они служат для ввода букв, цифр, знаков.

2.9.3. Для ввода любого символа достаточно выбрать регистр, нажав для этого необходимую регистровую клавишу.

ПОМНИТЕ:

2.9.4. В нормальном положении, т.е. без нажатия какой-либо регистровой клавиши, включён левый регистр.

2.9.5. При включении нижнего или правого регистров регистровую клавишу необходимо удерживать до окончания ввода символа. Для других регистров достаточно нажать и отпустить соответствующую регистровую клавишу.

2.9.6. Для ввода буквенных символов пользуйтесь буквенными регистрами:

- "ЛАТ" – латинский регистр;
- "РУС" – русский регистр;
- "ЗАГЛ" – заглавные буквы;
- "СТР" – строчные буквы.

2.10. Вопросы к главе 2.

2.10.1. В какой цвет окрашены регистровые клавиши на пульте ЭВМ?

2.10.2. Каково назначение регистровых клавиш?

2.10.3. Какие особенности при работе с регистровыми клавишами "ПР", "НР"?

2.10.4. В какой цвет окрашены буквенно-цифровые клавиши?

2.10.5. Каковы функции буквенно-цифровых клавиш?

2.10.6. Какими регистрами управляются буквенные клавиши?

2.10.7. Повлияет ли переключение с русского регистра на латинский (или наоборот) на ввод цифрового символа, например, на ввод символа "8"?

2.10.8. Какие клавиши необходимо нажать, чтобы пульт был настроен на ввод:

а) заглавных латинских букв?
б) строчных русских букв?
в) строчных латинских букв?
г) заглавных русских букв?

2.10.9. Какой символ отобразится на экране, если нажать клавишу "ЛАТ" и одновременно

ПР	А	?
НР	Ю	?
ЛАТ	Д	?
РУС	Г	?

3. Как общаться с ЭВМ?

Вам уже хочется, чтобы ЭВМ выполнила какое-нибудь ваше распоряжение? давайте заставим ЭВМ напечатать что-нибудь на экране. Но сначала очистим экран от ранее введённых нами символов. Для этого нам придётся пользоваться группой клавиш красного цвета, которые называются клавишами управления.

3.1. Очистка экрана.

Последовательно нажмите клавиши "СТОП" на экране высветится надпись:

```
? 25 АТ 0.00
* ОСТАНОВ ПО КЛАВИШЕ СТОП
```

затем 

экран очистится и под информационной строкой высветится символ

```
*
```

Запомните последовательность действий по очистке экрана.

3.2. Курсор.

Прежде чем начать описание вашего первого распоряжения ЭВМ, обратите внимание на знак ▉, стоящий перед свободной строкой на экране. Это – курсор – указатель места, куда будет выведен следующий символ на экране. При выводе какого-либо символа курсор передвигается по строке.

Если он вам мешает, курсор можно убрать, нажав один раз в нижнем регистре клавишу: 

При повторном её нажатии индикация курсора восстанавливается. Пока не советуем убирать курсор, т.к. он служит для того, чтобы вам удобно было контролировать место, куда будет выведен очередной нажатый вами символ.

3.3. Первое распоряжение.

3.3.1. Очистите экран (см. п. 3.1). Наберите на клавиатуре символы:

ЛАТ – латинский регистр;

Т – символ Т;

" " – пробел, ввод которого просто передвигает курсор на одну позицию вправо (любая из четырёх зелёных клавиш без маркировки);

ПР, " – правый регистр и кавычки;

РУС – русский регистр;

БК-0010 заявляет о себе – текст, который мы хотим вывести на экран;

ПР, " – правый регистр и кавычки.

На экране появилась введённая вами информация

```
Т "БК-0010 заявляет о себе"
```

подобные строки называются командными строками.

3.4. Теперь проверьте, правильно ли вы занесли всю информацию, на месте ли все кавычки и пробелы, не забыли ли вы поставить латинскую букву Т в начале строки.

3.4.1. Нажмите клавишу управление – "Ввод", нажатие этой клавиши равноценно приказу ЭВМ: выполняй то, что тебе предписано! на экране напечатается текст:

```
БК-0010 заявляет о себе.
```

В этом тексте уже нет ни буквы Т, ни кавычек, они нам потребовались лишь для оформления нашего приказа, команды ЭВМ.

3.4.2. А если нужный текст не напечатается? в этом случае повторите все ваши действия, начиная с п. 3.3.1. Внимательно проверьте, те ли вы набрали символы. Особенно обратите внимание на символ Т, пробелы и кавычки.

3.4.3. Обратите ваше внимание на один существенный момент. Нажмите ещё раз клавишу "Ввод". На экране появится ещё один символ "*" и больше ничего.

Команда, которую выдавали машине, действуя согласно пунктам 3.3.1 – 3.4.2, "забыта". Для того, чтобы заставить ЭВМ ещё раз напечатать тот же текст, необходимо опять повторить все действия по вводу команды. Происходит то же самое, что и при счёте на калькуляторе, вводятся какие-то

числа, команды, калькулятор выполняет предписанные ему действия и "забывает" о них.

Поэтому такой режим работы ЭВМ называется режимом калькулятора. Его ещё называют командным режимом. Он далеко не ограничивает всех возможностей ЭВМ. Об этих возможностях расскажем в следующих главах.

3.5. Что такое программирование на ЭВМ.

Это деятельность по составлению программ.

Программа – это набор указаний, команд, заставляющих выполнять ЭВМ нужные целенаправленные действия.

3.5.1. Так на каком же языке общаться с ЭВМ?

Обычный разговорный язык для этого не подходит. ЭВМ понимает только точные указания, не допускающие каких-либо двояких толкований. Поэтому, для выдачи ЭВМ приказов созданы специальные языки –

языки программирования.

Один из таких языков, а именно, "Фокал – БК 0010" "понимает" Ваша ЭВМ. Слово "понимает" взято в кавычки неслучайно. Дело в том, что, когда Вы общаетесь с ЭВМ на языке "Фокал", в ней работает достаточно сложная программа-интерпретатор языка "Фокал". Эта программа всегда содержится в постоянном запоминающем устройстве (ПЗУ) Вашей ЭВМ. После включения ЭВМ именно интерпретатор сообщает о готовности к диалогу с Вами:

```
ГОТОВНОСТЬ К РАБОТЕ
*
```

интерпретатор разбирает (интерпретирует) Ваши указания, изложенные на языке "Фокал". Результатом такой интерпретации является последовательность команд, которые машина выполняет непосредственно, такие команды называют ещё машинными кодами.

А зачем же тогда нужен язык? – спросите Вы. Не проще ли сразу излагать программы в машинных кодах, без всяких посредников – интерпретаторов.

Действительно, такое возможно. Именно в машинных кодах излагались программы на заре развития вычислительной техники. Но дело это чрезвычайно трудоёмкое, рутинное, и современные программисты в подавляющем большинстве случаев пользуются языками программирования высокого уровня.

Будем пользоваться одним из них – "Фокалом" – и мы с Вами.

> Специальное сообщение
>
> для людей, которые не могут удержаться и не перескакивать через главы этой книги (такие всегда есть!). Возможно, что контроль над ЭВМ будет потерян, если Вы некстати нажмёте не

те управляющие клавиши. В этом случае ЭВМ не будет выдавать Вам сообщение:

```
* ГОТОВНОСТЬ К РАБОТЕ.
```

Для восстановления управления ЭВМ нажмите на клавишу "СТОП", 

.

Если это не поможет, выключите и включите блок питания тумблером, находящимся на его передней панели. Учтите, что выключение питания влечёт за собой исчезновение введённой Вами информации из памяти ЭВМ.

3.6. Итоги главы 3.

3.6.1. Мы узнали, что последовательное нажатие управляющих клавиш "СТОП" и 

очищает экран.

3.6.2. Программа – это набор команд, заставляющих ЭВМ выполнять целенаправленные действия.

3.6.3. Программирование – деятельность по составлению программы.

3.6.4. Для составления программ в Вашей ЭВМ служит язык программирования – "Фокал".

3.6.5. Интерпретатор языка "Фокал" – это программа, переводящая команды, выданные на языке "Фокал", в машинные команды.

4. Первое знакомство с "Фокалом".

Вы уже немного знакомы с языком "Фокал". Первое распоряжение на нём Вы уже сделали (см. п. 3.3), введя в ЭВМ командную строку:

```
Т "БК-0010 заявляет о себе" ВВОД
```

здесь и в дальнейшем изложении " " означает пробел (4 пустых клавиши в зелёной зоне клавиатуры).

Давайте подробно разберём эту строку. Все распоряжения, которые мы даём ЭВМ, пишутся в виде, определённым образом построенных фраз языка "Фокал", такие фразы называются операторами.

Обычно оператор состоит из имени оператора и операндов (т.е. той информации, с которой работает оператор). Схематично оператор можно представить так:

```
<имя оператора><пробел>[<операнд1>,<операнд2>,...]
```

В угловых скобках расположены обязательные (т.е. те, которые не могут отсутствовать, в операторе) элементы оператора, в квадратных скобках

– те, которые могут в некоторых случаях отсутствовать. Такое схематичное изображение операторов называется форматом оператора.

В нашем примере:

"T" – это имя оператора печати, которое ЭВМ расшифровывает примерно так:

- Напечатай на экране текст,
  который заключён в кавычки.

Текст можно называть операндом. Повторите ещё раз действия по п. 3.3.1. Убедитесь, что, действительно, на экран было выведено (напечатано) сообщение:

```
БК-0010 заявляет о себе.
```

Буква "T" – это начальная буква английского слова TYPE. "Фокал" позволяет писать не только начальную букву в операторе, а всё английское слово, результат будет один и тот же. Но согласитесь, написать одну букву проще, чем целое слово; кроме того, целое слово занимает больше места в памяти ЭВМ, чем одна буква. Поэтому в дальнейшем мы советуем писать Вам операторы не полным словом, а одной начальной буквой.

У Вас, видимо, возник ещё один вопрос:

- почему для операторов языка выбраны английские слова, а не русские? Ведь писать операторы на родном языке нам было бы удобнее.

Такое неудобство вызвано тем, что языки программирования, как правило, международные. Есть договорённость (стандарт) на каждый язык, в том числе и на "Фокал". Этот стандарт предписывает использовать в языке "Фокал" английские слова, хотя это нам и не совсем удобно. В утешение хочется сказать, что при достаточной практике в написании операторов. "Фокала" это языковое неудобство исчезает само собой. Вы о нём просто забудете.

Кроме оператора "T" язык "Фокал" содержит ещё 11 операторов. Все они подробно описаны в настоящем руководстве. Их перечень и краткая характеристика приведены в приложении 2.

4.2. Когда можно вводить команды – операторы в ЭВМ?

Выключите и включите блок питания. Под информационной строкой Вы увидите на экране:

```
? 00 AT 0.00
Готовность к работе
*
```

появление после текстового, сообщения символа "*" означает готовность к приёму Ваших команд.

4.3. Наберём на клавиатуре пульта следующую командную строку:

```
10.01 T "БК-0010 заявляет о себе"
```

Нажмём клавишу "Ввод".

На экране ничего не происходит, только символ "*" и курсор переместились на следующую строку. Проверьте, правильно ли занесена информация в ЭВМ. Если в строке есть какая-либо ошибка или описка, повторите ввод заново, без ошибок.

Введите команду, наберите на пульте символ "G" и нажмите клавишу "Ввод". На экране должен появиться текст:

```
БК-0010 заявляет о себе.
```

4.4. Ещё раз наберите символ "G", затем "Ввод"; текст ещё раз появился на экране, несмотря на то, что мы не вводили его ещё раз, как в п. 3.3.1. ЭВМ запомнила данную в п. 4.3 команду и по следующей команде "G" выполнила её.

Почему же ЭВМ команду п. 4.3 "запомнила", а команду п. 3.3.1 – 3.3.3 "забыла", чем отличаются эти командные строки?

Напишем их рядом:

```
      T "БК-0010 заявляет о себе" (п. 3.3.1).
10.01 T "БК-0010 заявляет о себе" (п. 4.3).
```

Очевидно, что эти строки отличаются только числом 10.01, стоящим перед запомненной командной строкой. Это число называется номером командной строки.

В качестве номера в языке "Фокал" можно использовать десятичные числа от 1.01 до 99.99 за исключением тех, у которых дробная часть равна нулю. Указанные номера должны различаться не менее, чем на 0.01. Разрешается также использовать номера с 100.1 до 127.9. Номера этой группы должны различаться не менее, чем на 0.1.

Обычно командные строки нумеруют не с минимальным шагом (0.01 или 0.1), а с промежутками, чтобы была возможность между двумя командными строками вставить ещё одну или несколько строк.

Ниже приведены примеры написания номеров.

```
1.10
1.2
100.1
100.20
3.00
5.00
```

Два последних номера написаны неправильно, т.к. они оканчиваются на 00.

Пронумерованные командные строки выполняются не сразу после нажатия клавиши "Ввод" (после этого ЭВМ их только запоминает), а после вызова их на исполнение.

Последний раз мы вызывали на исполнение такую пронумерованную строку командой "G".

Обращаем Ваше внимание на то, что каждая командная строка и с номером, и без номера должна завершаться нажатием клавиши "Ввод".

4.5. Командные строки без номера перед оператором будем называть просто – командами или простыми командами, команды выполняются сразу же после нажатия клавиши "Ввод".

Командные строки с номером перед оператором будем называть косвенными командами.

Косвенные команды запоминаются в памяти ЭВМ и выполняются только после передачи им управления либо командами, либо другими косвенными командами.

4.6. Передача управления.... Что это значит? Поясним на примере.

В памяти нашей ЭВМ сейчас есть строка с номером 10.01. Если Вы выключали ЭВМ, то введите эту строку ещё раз:

```
10.01 T "БК-0010 заявляет о себе" ВВОД
```

введём в память ЭВМ ещё одну косвенную команду:

```
10.06 G 10.01 ВВОД
```

рассмотрим эту косвенную команду:

мы уже знаем, что номер 10.06, стоящий перед оператором "G", отличает косвенную команду от простой команды.

Символ " G ", стоящий за номером, – это, как и обычно, имя оператора, начальная буква английской фразы GO TO, которую в нашем случае можно понимать как приказ – перейти к выполнению строки.

Далее следует номер командной строки, которую должна выполнить ЭВМ после того, как только она закончит выполнять командную строку 10.06. (Это строка с номером 10.01). Этот номер указан в строке 10.06 после имени оператора "G".

Результатом выполнения командной строки

```
10.06 G 10.01
```

будет автоматический переход к выполнению строки 10.01, т.е. автоматически произойдёт передача управления от строки 10.06 к строке 10.01.

4.7. Вы правильно ввели командные строки 10.01 и 10.06? Ещё раз проверьте и, если найдёте ошибки, повторите ввод ошибочной строки сначала.

Введите команду:

```
G ВВОД.
```

Эта команда начинает выполнение первой строки нашей программы. Правильно также будет сказать: командой "G" управление передаётся на строку с номером 10.01, т.е. на вывод знакомого нам текста.

Но что происходит на экране?

Вывод текстовой строки непрерывно повторяется!

Всё правильно, это мы наблюдаем действие косвенной команды, которую мы ввели строкой 10.06. Пусть пока ЭВМ поработает, а мы ещё раз подробно разберём, как функционирует наша программа.

После того, как мы последний раз нажали на клавишу "Ввод", управление было передано на строку:

```
10.01 T "БК-0010 заявляет о себе".
```

По этой команде ЭВМ воспроизводит на экране текст.

Как только ЭВМ закончила, выполнение указанной командной строки, она просматривает есть ли в программе ещё командные строки, номера которых больше, чем у данной строки. Если такие строки есть, то управление передаётся строке со следующим номером. Такой строкой в нашем примере является:

```
10.06. G 10.01
```

выполняя эту косвенную команду, ЭВМ передаёт управление на строку 10.01, по которой на экран опять будет выдано сообщение ... и т.д. ЭВМ выполняет в нашем случае непрерывный вывод текстовой информации. В этом случае, как говорят программисты, ЭВМ "зациклилась" на выводе текстовой информации. Из этого состояния сама по себе она выйти уже не сможет.

Если Вам уже наскучило мелькание одной и той же информации, нажмите на клавишу "СТОП". ЭВМ опять выдала символ "*", который, как мы знаем, приглашает нас к общению с ЭВМ.

4.8. Итоги главы 4.

4.8.1. При нажатии на клавишу "СТОП" ЭВМ прекращает выполнение предписанных ей ранее действий и приглашает вас к диалогу, высвечивания на экране символ *

4.8.2. Мы узнали новые операторы:

- "G" (GO TO), предписывающий ЭВМ перейти к выполнению строки с номером, записанным за оператором в данной строке.
- "T" (TYPE) -вывести на экран информацию, можно также сказать: напечатать на экране указанную в операторе информацию.

4.8.3. Если за оператором "G" не следует никакого номера, управление передаётся на командную строку с минимальным номером.

4.8.4. ЭВМ понимает командные строки двух типов: команды и косвенные команды

- команды пишутся в строке, начиная с оператора, а строка не нумеруется;
- косвенные команды начинают писать с номера, через пробел пишется имя оператора и через пробел вся остальная информация;
- косвенные команды располагаются в памяти в порядке возрастания их номеров.

4.8.5. Каждая командная строка должна заканчиваться нажатием клавиши "Ввод".

4.8.6. Управление от одной косвенной команды к другой передаётся автоматически, либо по возрастанию номеров косвенных команд, либо с помощью команд передачи управления, одной из которых является косвенная команда, содержащая оператор "G".

4.8.7. Режим, в котором выполняются команды, называется диалоговым режимом.

4.8.8. Режим, в котором выполняются косвенные команды, называется программным режимом.

4.9. Вопросы к главе 4.

4.9.1. Чем отличается команда от косвенной команды?

4.9.2. Как осуществить передачу управления на косвенную команду?

4.9.3. Как передать управление на команду?

4.9.4. Что делает приведённая ниже программа?

```
10.10 T "AAA" ввод      (A)
10.20 G 10.10 ввод      (Б)
G ввод                  (В)
```

4.9.5. Какие командные строки, приведённые в программе упражнения 4.9.4 – команды, какие – косвенные команды?

4.9.6. Напишите программу, которая бы вывела на экран вашу фамилию, имя, отчество. (Воспользуйтесь оператором "T").

5. Арифметические выражения, оператор SET.

5.1. Умеет ли ЭВМ считать?

Само название ЭВМ – электронная вычислительная машина говорит, что её прямое назначение – считать.

Продемонстрируем это на примере перемножения трёх чисел: 29, 37, 396. С этой целью наберём на клавиатуре команду:

```
T %8.02 26*37*396 ВВОД.
```

На экране появилось число 380952.00; это результат.

Рассмотрим подробнее процесс выполнения команды. В командной строке нет номера перед оператором "T", а это значит, что наша ЭВМ начнёт выполнять такую команду сразу же после нажатия клавиши "Ввод", т.е. ЭВМ работает в диалоговом режиме. "T" – это оператор печати. В нашем примере он нужен для того, чтобы вывести на экран результат вычисления арифметического выражения, стоящего после оператора "T". Присмотримся к арифметическому выражению.

Мы должны перемножить три числа, но вместо привычного нам знака умножения стоит знак "*" – звёздочка. Вас это не должно смущать, просто разработчики языка приняли за знак умножения – знак "*", за знак деления – "/" – наклонная черта. Только знаки вычитания – "-" и сложения – "+" не претерпели изменений.

Кроме этих четырёх арифметических действий вы можете пользоваться действием возведения в степень. Для возведения в степень некоторого числа (основания) между основанием и степенью необходимо поставить знак "^"

Например, для того, чтобы возвести 2 в десятую степень, необходимо написать командную строку:

```
T 2^10 ВВОД.
```

Введите эту строку в ЭВМ. Результат – число 1024.0000 обратите на него внимание. Сосчитайте количество цифр в результате – оно равно 8. При работе "Фокала" в стандартном режиме печати в результате обработки командных строк ЭВМ выдаёт восьмизначные десятичные числа, только в этих числах целую часть от дробной отделяет не запятая, а точка. Она носит название "десятичной точки".

ПРИМЕЧАНИЕ. При возведении в степень дробная часть степени во внимание не принимается, например:

```
2^2=2^2.9=4
```

5.2. Решим пример из школьного задачника: возвести в куб следующее выражение

```
[(3,141-1,256) (2,25+5,62)]
```

в этом арифметическом выражении присутствуют скобки, они определяют порядок действий при вычислении значения выражения.

При написании арифметических выражений на языке "Фокал", вы тоже можете пользоваться скобками, почти также, как и в привычной Вам записи выражений в школьной тетради, помня при этом об общепринятом правиле:

Каждой "открывающей" скобке в выражении должна соответствовать "закрывающая" скобка такого же типа.

Кроме "круглых" и "квадратных" скобок, присутствующих в данном примере, вы можете использовать ещё "угловые" скобки – "< >".

Если выражение содержит скобки внутри скобок, то сначала вычисляются выражения, заключённые во внутренние скобки, а затем – во внешние, причём скобки любого типа – круглые (), квадратные [], угловые <> равноправны при определении порядка операций.

Введём в ЭВМ команду, реализующую необходимые нам вычисления, учитывая, что вместо знака умножения используется знак "*", знак "^" используется для операции возведения в степень, а вместо запятой, отделяющей целую часть числа от дробной, используется десятичная точка. Итак,

```
T [(3.141-1.256)*(2.25+5.62)]^3 ввод.
```

На экране напечатается значение вычисленного арифметического выражения: 3264.8200

А как определяется порядок арифметических операций, если скобки отсутствуют?

5.3. При отсутствии скобок последовательность выполнения операций следующая:

- возведение в степень (^);
- умножение (*);
- деление (/);
- сложение (+);
- вычитание (-);

В ситуациях, когда "Фокал" имеет дело с операциями одного приоритета, следует помнить, что операции выполняются слева направо.

В сомнительных случаях, чтобы обеспечить ту очерёдность выполнения операций, которая вам необходима, не стесняйтесь пользоваться скобками.

Маленький сюрприз:

нажмите одновременно на клавиши "НР" и 

. Курсор на экране стал шире в два раза.

В строке теперь помещается не 64 символа, а только 32. В прежний режим (64 символа в строке) можно перейти повторным нажатием указанных клавиш.

5.4. До сих пор все вычисления мы делали в диалоговом режиме. Такой режим полезен в случае, когда мы имеем дело, с небольшими одноразовыми вычислениями.

Наибольший эффект в выполнении вычислительных работ достигается только в программном режиме, когда ЭВМ выполняет хранящуюся в памяти программу, автоматически передавая управление от одной командной строки

к другой. Только в программном режиме наиболее полно раскрываются возможности, заложенные в ЭВМ.

Попробуем решить на ЭВМ задачу:

Расстояние между городом равно L. За сколько времени можно преодолеть это расстояние на самолёте, летящем со скоростью V ?

Эта задача вполне посильна школьнику, знакомому с алгеброй. Примерный ход его рассуждений таков: обозначим через X искомую величину – время полёта. Известно, что время равно пути, поделённому на скорость, т.е.

X=L/V;

это и есть решение.

Где X, L, V – переменные величины задачи.

Но вот парадокс, для любой, даже самой совершенной ЭВМ, эта задача представляет пока почти непреодолимую сложность.

Дело заключается в том, что большинство ЭВМ – цифровые машины, их часто даже называют ЦВМ (цифровая вычислительная машина). Они оперируют с числами, а в нашей задаче пока нет ни одного, числового значения.

Как только мы дополним задачу и дадим переменным L и V (как говорят специалисты – присвоим переменным L и V) конкретные числовые значения, вычисление ответа задачи (переменной X) не представит никакой сложности. Дополним нашу задачу, т.е. присвоим числовые значения переменным. Пусть,

L=15000

V=650

т.е. путь равен 15000 км и самолёт летит со скоростью 650 км/час.

Напишем программу, вычисляющую ответ нашей задачи:

```
10.10 S L=15000 ввод
10.20 S V =650 ввод
10.30 S X= L/V ввод
10.40 T X ввод.
```

Разберём подробно каждую строку этой программы: в строках 10.10, 10.20, 10.30 сразу после номера командной строки стоит латинская буква "S ", это – имя оператора SET – присвоить.

В строке 10.10 программы за оператором "S" стоит переменная L, которой мы хотим присвоить значение, отделённое от переменной знаком равенства.

В результате выполнения этого оператора переменной будет присвоено значение 15000.

В строке 10.20 в результате выполнения этого оператора переменной V будет присвоено значение 650.

В строке 10.30 за переменной X после знака "=" стоит выражение L/V (переменная L делится на переменную V). В результате выполнения оператора результат деления будет присвоен переменной X.

Понятие переменной в языке "Фокал" имеет общепринятый в математике смысл: это числовая величина, имеющая имя. Подробнее переменные описаны в пункте 9.4.

Особо подчеркнём, что знак равенства "=" в выражениях, используемых в языке "Фокал", имеют смысл, отличающийся от смысла, наделённого ему математиками: "левая" часть равна "правой" части.

Интерпретатор "Фокала" воспринимает знак равенства в смысле: переменной в "левой" части присвоить значение выражения, стоящего в "правой" части.

Например, недопустимое для элементарной математики "равенство"

```
X=X+1,
```

в языке "Фокал" вполне допустимая конструкция, имеющая смысл приказа – увеличить значение переменной X на 1. Выражения приведённого выше типа называются рекурсивными и широко используется в программировании.

Занесите каждую строку программы в память ЭВМ, проверьте правильно ли вы её занесли.

Если в результате выполнения программы появится результат, отличный от ожидаемого, или сообщение об ошибке – увы ... введите программу вновь, без ошибок, и выполните. Добейтесь, чтобы появился правильный результат – число 23.0769.

5.5. Итоги главы 5.

5.5.1. Чтобы вычислить значение какого-либо выражения, достаточно записать это выражение в операторе "T" (TYPE), или в операторе " S " (SET).

5.5.2. При написании выражения можно пользоваться – знаками арифметических действий:

(^) – возведение в степень

(*) – умножение

(/) – деление

(+) – сложение

(-) – вычитание.

– скобками:

( ) – круглыми

[ ] – квадратными

< > – угловыми.

5.5.3. Порядок выполнения операций: сначала в скобках, причём внутренние скобки выполняются ранее, чем включающие их внешние. При

отсутствии скобок (или внутри скобок) операции выполняются в порядке, приведённом в п. 5.5.2.

5.5.4. Оператор "S" (SET) – оператор присваивания, Он служит для присваивания значения выражения, стоящего справа от знака "=", переменной, стоящей слева от знака "="

5.6. Вопросы к главе 5.

5.6.1. Оператор " S " содержит выражение:

```
([(X-4)*(X+4)-(5.25+A)]*100)^2
    1  2  3  4     5   6    7
```

в котором пронумерованы цифрами (1-7) все операции. Напишите номера операций в последовательности, в которой они будут выполняться.

5.6.2. Какое число напечатается в результате выполнения оператора:

```
T (5-3)^4*6/8+3 ?
```

5.6.3. Какое значение присвоится переменной Z в результате выполнения программы:

```
10.10 S X=5
10.20 S Y=10
10.30 S Z=(Y-X)^2/5 ?
```

6. Об ошибках

6.1. Вы, наверное, уже столкнулись с неприятным для вас фактом – при написании программ и их вводе в ЭВМ вы иногда допускаете ошибки.

Не огорчайтесь, ошибки допускают почти все специалисты, занимающиеся программированием. Считается, что даже опытный специалист не может без ошибок написать программу, состоящую более, чем из 50 операторов.

Программирование – это вид человеческой деятельности, который требует огромного внимания, глубокой сосредоточенности, усидчивости и терпения.

Ошибки, которые допускают программисты, можно поделить на две группы:

6.1.1. Логические ошибки.

Это ошибки, вызванные неверным переводом хода решения задачи на язык машины. Они вызывают такую ситуацию: каждый отдельно взятый оператор в программе написан без ошибок, однако, вся программа не работает. Причина этого нарушения логической взаимосвязи между самими операторами или между операторами и данными, с которыми они оперируют.

Выявление логических ошибок – процесс довольно сложный, требующий определённого опыта от программиста.

В этой главе мы не будем обсуждать вопрос о том, как выявляются логические ошибки, об этом будет сказано в главе 15.

6.1.2. Синтаксические ошибки.

Это – ошибки, вызванные отступлением от правил написания элементов языка "Фокал".

Вы, наверное, знаете, как сложно без ошибок написать, например, школьное сочинение на родном языке, которым вы от рождения пользуетесь и свободно владеете.

Не обойдётся без ошибок и написание командных строк, которые вы пишете на языке "Фокал". Только ситуация здесь для вас более благоприятная. Дело в том, что интерпретатор "Фокала", прежде чем начать выполнять любую командную строку, как школьный учитель, сам ищет ошибки, которые вы могли в ней допустить. Если какая-либо строка содержит хотя бы одну ошибку, выполнение программы приостанавливается, на экран выдаётся сообщение об ошибке, и интерпретатор переходит в режим диалога.

6.1.3. Предположим, в вашей программе допущена такая ошибка: количество открывающих скобок больше, чем количество закрывающих, например:

```
40.10 T [(5-3)+(2-1)
```

после передачи управления этому оператору будет выдано сообщение:

```
? 03 AT 40.10
непарные скобки
*
```

такое сообщение называется диагностическим, т.к. оно не только указывает на наличие ошибки, но и определяет её причину.

6.1.4. В начале верхней строки сообщения стоит знак вопроса, который мы с вами должны расценивать, как признак неопределённости (ошибки).

Далее стоят две цифры (в примере это 03) – это код ошибки. По этому коду, посмотрев в приложение 1 настоящего руководства, вы можете определить, какая ошибка вами допущена и найти рекомендацию, как её устранить.

За кодом следует две буквы – "AT" – это признак того, что данное сообщение вызвано ошибкой в операторе, номер которого стоит последним в строке (в примере это 40.10).

6.1.5. Во второй строке сообщения содержится вид ошибки, смысл его понятен. За сообщением следует символ "*", это приглашение к диалогу, которое следует расценивать, как предложение приступить к исправлению ошибки.

6.2. Как исправлять выявленные ошибки?

6.2.1. Самый простой способ – это заново переписать всю строку, в которой допущена ошибка. Если ошибка допущена в косвенной команде, то переписывать строку необходимо с номера этой косвенной команды (номера строки).

За номером (если он есть) записывается вся информация, содержащаяся в командной строке. После набора всей информации, для её ввода необходимо нажать на клавишу "Ввод", как это делается при вводе любой командной строки.

6.2.2. А как быть, если строка ещё не введена, а вы увидели в ней ошибку? в этом случае вы можете воспользоваться возможностями редактирования текста. Редактированием можно пользоваться и в других случаях.

6.3. Редактирование текста.

6.3.1. Редактирование текста – это работа по изменению информации в редактируемой строке. К таким изменениям относятся:

- удаление символов из строки;
- замена символов на другие;
- вставка в строку новых символов.

6.3.2. Редактируемой считается текстовая строка, которая вами набрана, но не введена (клавиша "Ввод" ещё не нажата), или строка, которая вызвана на редактирование оператором "M" (MODIFY). Оператор "M" будет описан в пункте 6.3.5.

6.3.3. Наберите командную строку, в которую умышленно вставьте ошибки.

Предупреждение: после набора приведённого ниже примера не нажимайте клавишу "Ввод".

Итак, наберите на клавиатуре командную строку, печатающую фамилии: Иванов, Петров, Сидоров.

```
20.90 T ИвановПетровСидороввв.
```

допущены следующие ошибки:

- синтаксическая ошибка: выводимый на экран текст должен быть ограничен символом (") – кавычками. Их необходимо вставить в строку.
- в конце строки, в фамилии Сидороввв два лишних символа вв. Эти символы необходимо удалить из строки.
- фамилии не разделены, необходимо вставить запятые.

6.3.4. Для редактирования строки служат клавиши редактирования, они окрашены в жёлтый цвет и расположены в правой части клавиатуры. Рассмотрим их.

6.3.4.1. Начнём с клавиш  и  . Эти клавиши служат для управления положением курсора. Однократное нажатие любой из них передвигает курсор на одну позицию. Учтите, что само передвижение курсора ничего не меняет в редактируемой строке, оно лишь указывает место в строке, в котором необходимо что-то изменить. Сейчас курсор стоит в конце строки, именно на том месте, где нам необходимо поставить символ " – кавычки. Не передвигая курсора, поставьте кавычки (на правом регистре нажмите клавишу "  "). В нужном месте появится символ ".

6.3.4.2. Теперь вставим кавычки в начало строки. С этой целью передвиньте курсор влево (нужное количество раз нажмите клавишу "  ") и установите его на первой букве (буква И).

Именно в этом месте согласно правилам написания текстов в операторе "Т" должны стоять открывающие кавычки ("). Мы их можем поставить, но в этом месте стоит буква "И". Обратите внимание, она сейчас темнеет "внутри" курсора. Если мы сразу нажмём на символ ("). он появится в строке вместо символа "И", а нам его надо сохранить.

Для того чтобы и текст сохранился, и кавычки в нём стояли на месте, нам необходимо раздвинуть весь текст вправо от курсора, освободив место, на котором у нас сейчас стоит буква "И".

6.3.4.3. Эта операция (раздвижка) осуществляется нажатием на клавишу "  " (не перепутайте символ "  " с символом "  "). Нажмите указанную клавишу. Как видите, все буквы в строке, начиная с буквы "И", которая находилась "в курсоре" сдвинулись вправо на одну позицию. Курсор остался на месте, но позиция под ним освободилась. Поставим теперь в освободившейся позиции символ " – кавычки.

6.3.4.4. Теперь удалите стоящие в конце текста две лишние буквы в. Для этого передвиньте курсор в самый конец нашего текста – установите его в позиции, в которой стоят закрывающие текст кавычки. Нажмите клавишу "  " столько раз, сколько необходимо для того, чтобы "внутри" курсора оказались стоящие справа от текста кавычки. (В дальнейшем мы не будем писать подробно, как вам необходимо передвигать курсор, будем только указывать: передвиньте курсор).

Нажмите клавишу "  ". Эта клавиша служит для того, чтобы удалить из строки символ, стоящий слева от курсора. Ещё раз нажмите "  ". Обе мешавшие нам буквы "в" заменены пробелами. Но эти пробелы нам не желательны, их неплохо было бы убрать из строки.

Эту операцию выполним с помощью клавиши "  " сжатие строки. Нажмите на эту клавишу два раза. Символ (") сдвинется на две позиции в направлении стрелки на клавише (влево), символы, которые находились в строке левее курсора не изменились, "исчезли" два пробела.

Итак, на экране имеем сейчас текст:

```
20.90 T "ИвановПетровСидоров"
```

6.3.4.5. Вставьте между фамилиями запятые. Установите курсор на букву "П". Раздвиньте текст клавишей "  " Позиция под курсором освободилась, запищите на этой позиции запятую (,). Запятая теперь стоит на нужном месте между фамилиями Иванов, Петров. Аналогичным образом вставьте запятую между фамилиями Петров и Сидоров. Передвиньте курсор на букву "С", раздвиньте текст клавишей "  " напишите запятую (,). Всё, что мы хотели исправить, исправлено! Нажмите клавишу "Ввод" введите отредактированную строку в память.

Возможности вашей ЭВМ позволяют отредактировать косвенную команду, которая уже находится в памяти ЭВМ. Для этого служит оператор "M" (MODIFY).

6.3.5. Попробуем с помощью оператора M вызвать на редактирование только что записанную нами в память строку (20.90). Наберите на пульте команду:

```
M 20.90 ввод.
```

- на экране появится текст косвенной команды 20.90:

```
T "Иванов, Петров, Сидоров"
```

Заметим, что номер в этом тексте отсутствует. (В командной строке он, конечно, есть). Оператором "M" можно отредактировать любую часть косвенной команды, за исключением номера.

Вы уже, наверное, догадались, что при вызове какой-либо командной строки на редактирование с помощью оператора вам необходимо за оператором "M" указать номер этой строки.

Выведенный оператором "M" текст можно редактировать с помощью клавиш редактирования, как это мы делали раньше. В жёлтом поле клавиш редактирования находятся и другие клавиши.

Следует знать, что редактирующие клавиши управляют положением курсора в строке, но сами коды редактирующих клавиш в памяти не записываются. Коды клавиш

|↖ , ↑ , ↓ , ↖ , ↗ , ↘, ↙

вообще игнорируются, т.е. эти клавиши не управляют положением курсора, и их коды не записываются в память.

В "Фокале" имеется возможность записи кодов вышеуказанных клавиш в память.

Для этого необходимо нажать одновременно клавиши "НР" и " 

". Условно обозначим эту комбинацию (НР/ 

).

Эта процедура вызовет индикацию в информационной строке вверху экрана слова "РЕД", что означает переход в режим ввода "РЕД". Для перевода "Фокала" в обычный режим необходимо нажать "НР/ 

", при этом слово "РЕД" в информационной строке исчезнет.

В режиме "РЕД" коды редактирующих клавиш и клавиш

|↖ , ↑ , ↓ , ↖ , ↗ , ↘, ↙

записываются в память ЭВМ и изменяют положение курсора на экране, но их изображение на экране не высвечивается. Это позволяет рисовать на плоскости экрана различные фигуры, что может быть полезно при составлении текстов примечаний в операторах "Т" и "А".

Вы должны иметь в виду, что в режиме "РЕД" никакая редакция введённой информации не возможна. Для коррекции введённой информации необходимо выйти из режима "РЕД", скорректировать строку и, при необходимости, вернуться в режим "РЕД". Максимальное количество символов в строке равно 80.

Пример:

на экране дисплея необходимо нарисовать следующую картину:

```
      AAA
BBB      CCC
```

Для этого необходимо использовать оператор "Т" и приведённую картину оформить в виде примечания:

```
1.1 T "AAA
BBB    CCC"
```

При наборе этой строки после первых кавычек необходимо перейти в режим "РЕД" и для получения нужной картины набрать следующую последовательность символов:

```
A, A, A, ↓, ←, ←, ←, ←, ←, ←,
B, B, B, →, →, →, C, C, C.
```

После этого надо выйти из режима "РЕД" нажатием НР/ 

, закрыть текст картинки кавычками и нажать клавишу "Ввод". При выполнении данной строки на экране дисплея появится вышеприведённая картинка.

6.4. Итоги главы 6.

6.4.1. Все допускаемые при программировании ошибки можно условно разделить на две группы:

- логические ошибки,
- синтаксические ошибки.

6.4.1.1. Логические ошибки вызваны нарушением взаимосвязи между структурными элементами программы, операторами, данными.

6.4.1.2. Синтаксические ошибки – это ошибки, вызванные отступлением от правил написания элементов языка "Фокал":

- операторов;
- цифр;
- символов.

6.4.2. Допущенные синтаксические ошибки в большинстве случаев автоматически выявляются интерпретатором "Фокала".

6.4.3. Исправление ошибок можно осуществлять либо, переписывая весь оператор, содержащий ошибку, либо с помощью средств редактирования.

6.4.4. Редактирование – это работа по изменению информации в редактируемой строке:

- удаление символов;
- замена символов;
- вставка символов.

6.4.5. Редактирование осуществляется с помощью клавиш редактирования:

- "—> " – сдвиг курсора вправо на одну позицию;
- " <—" – сдвиг курсора влево на одну позицию:
- "|—>" – размыкание (раздвигание) строки вправо с позиции, где расположен курсор;
- "<—|" – смыкание строки справа от позиции, где расположен курсор;
- "<—+" – удаление последнего введённого символа;
- "СБР |->" – стирание строки, начиная с позиции курсора;
- " 

" – сброс экрана;
- "ВС" – возврат курсора в начало строки;
- "ГТ" – горизонтальная табуляция (сдвиг курсора на 8 позиций вправо).

6.4.6. Режим редактирования заканчивается нажатием клавиши "Ввод".

6.4.7. Для вызова на редактирование содержащихся в памяти косвенных команд служит оператор "M" (MODIFY), за которым стоит номер вызываемой строки.

6.5. Вопросы к главе 6.

6.5.1. Можно ли исправить средствами редактирования номер уже введённого оператора?

6.5.2. Какая строка считается редактируемой?

6.5.3. Чем заканчивается процесс редактирования?

6.5.4. Как можно удалить подчёркнутые символы в приведённой строке?

```
40.03 T "РЕДАКТИРРОВАНИЕ".
```

6.5.5. Можно ли для удаления символов пользоваться возможностями клавиши " <—| " – смыкание?

7. Операторы ASK, ERASE, WRITE.

Глава содержит сведения о программных запросах данных, удаление программных строк и переменных, выводе текста программ на экран.

Давайте теперь вернёмся к программе о вычислении времени полёта самолёта по формуле X=L/V (см. п. 5.4).

```
10.10 S L=15000
10.20 S V=650
10.30 S X=L/V
10.40 T X
```

У этой программы есть существенный недостаток: она рассчитывает время только для пути, равного 15000 км, и скорости, равной 650 км. Для того чтобы дать этим величинам другие значения, необходимо переделывать строки программы 10.10 и 10.20, а это неудобно.

А нельзя ли сделать так, чтобы программа сама спрашивала, какие нам необходимы расстояние и скорость? напишем такую программу:

```
10.10 A "расстояние", L   ввод
10.20 A "скорость", V  ввод
10.30 S X=L/V   ввод
10.40 T X    ввод
```

она лишена упомянутого выше недостатка.

7.2. Прежде чем вводить программу в память, давайте очистим её от операторов, которые мы с вами ввели ранее, введём команду:

```
E A ввод.
```

Оператор "E" (ERASE) предназначен для уничтожения информации, в оперативной памяти, относящейся к программе.

Оператор имеет следующие модификации:

`E N` уничтожает строку или группу строк с номером N и переменные.

`E T` уничтожает текст программы, оставляя переменные.

`E A` уничтожает и текст, и переменные.

`E` уничтожает только переменные, оставляя текст.

Группа строк – это совокупность строк, имеющих номера с одинаковой целой частью.

Если мы хотели бы уничтожить только оператор 20.90, оставив в памяти все остальные операторы, мы должны бы были написать команду:

```
E 20.90 ввод
```

Если бы мы хотели уничтожить не все операторы, а только те, у которых номера имеют в целой части число 10, достаточно было бы выполнить команду:

```
E 10 ввод.
```

7.3. Введите последнюю модификацию программы о времени полёта самолёта.

Проконтролируйте, правильно ли вы ввели программу, с этой целью выполните в диалоговом режиме команду:

```
W ввод
```

на экране отпечатывается вся только что введённая программа. Оператором "W" (WRITE) можно отпечатать на экране любую командную строку, написав через пробел её номер или номер группы строк (группа строк задаётся номером группы, так же, как в операторе "E").

7.4. Выполните программу. Вы помните, что для этого необходимо выполнить команду: G ввод. На экране появится сообщение (запрос)

```
расстояние:
```

понятно, что у нас спрашивают, какое расстояние мы хотим задать в задаче. Используя "старые" данные, ответим:

```
15000 ввод
```

на экране появится:

```
скорость:
```

ответим:

```
650 ввод
```

на экране появится ответ 23.0769.

Рассмотрим подробно, как работает эта программа. В строках 10.10 и 10.20 программы мы видим, что после номера строки стоит латинская буква "A". Это сокращённое написание имени нового оператора ASK, его интерпретатор понимает как приказ: ввести данные. "Фокал" позволяет вводить данные с клавиатуры по запросу оператора "A".

При вводе любых данных всегда возникают по крайней мере два вопроса:

- какие данные надо вводить?
- каким переменным присвоятся введённые данные?

Ещё раз запустите нашу программу. На экране появится

```
расстояние:
```

Посмотрите внимательно на строку 10.10. Слово "расстояние" записано после оператора "A", в кавычках. Кавычки указывают интерпретатору, что прежде чем приступить к вводу данных, необходимо вывести на экран слово, стоящее в кавычках.

И слово, и кавычки в операторе можно опустить, но в этом случае на экране появится только двоеточие (:), и нам будет труднее разбираться, какую именно информацию требует от нас программа. Слово в кавычках как раз и служит для того, чтобы напомнить нам о ней. За словом в кавычках, через запятую, следует переменная, которой присвоится значение, введённое нами с клавиатуры. Переменная должна стоять без кавычек.

В числе, вводимом с клавиатуры, может присутствовать десятичная точка, отделяющая целую часть от дробной. Число также может быть задано в экспоненциальной форме. О такой форме представления чисел будет сказано в главе 8.

Одним оператором "A" можно ввести несколько данных сразу.

Командная строка

```
12.15 A "расстояние,L,"скорость",V ввод
```

будет выполнять точно такие же действия, как две строки нашей программы (10.10 и 10.20). Такая строка даже предпочтительнее, она занимает меньше места в памяти ЭВМ. Введём данные для программы.

На запрос "расстояние:", введём число, например, 2000. На запрос "скорость:", введём число 1000. Получим результат 2.0000.

Можете "поиграть" с вашей программой, задавая ей различные расстояния и скорости, а для того, чтобы это делать было удобнее, введём в программу ещё один оператор:

```
20.1 G 10.1 ввод
```

запустите программу и поупражняйтесь во вводе данных, давая различные значения расстоянию и скорости.

Теперь оператор 20.10 не позволит программе закончиться. После очередного расчёта или, как говорят программисты, прохода, управление будет передаваться опять на начало программы. Выйти из такого решения можно с помощью клавиши "СТОП".

7.5. Итоги главы 7.

7.5.1. Оператор "E" (ERASE) предназначен для уничтожения в оперативной памяти либо только всех переменных, либо только текста программы (всего или заданных строк), либо и всего текста, и всех переменных.

7.5.2. Оператор "W" (WRITE) предназначен для вывода на экран командных строк, записанных в памяти ЭВМ.

Если после оператора "W" нажать клавишу "Ввод", на экран будут выведены все строки, хранящиеся в памяти.

Если после оператора "W" через пробел стоит номер строки (группы строк), выводится только данная строка (группа строк).

7.5.3. Оператор "A" (ASK) служит для ввода информации в ЭВМ. В операторе "A" через пробел следуют переменные, которым необходимо присвоить значения этих данных. Переменные должны разделяться запятыми. В операторе "A" могут также присутствовать текстовые сообщения, которые должны заключаться в кавычки. От других операндов оператора "A" сообщение должно быть отделено запятой.

7.5.4. На запрос оператора "A" (:) необходимо набрать нужную информацию. Ввод информации происходит после нажатия клавиши "Ввод".

7.6. Вопросы к главе 7.

7.6.1. Выведите на экран текст программы, записанной в памяти.

7.6.2. Напишите оператор, который выведет на экран все операторы программы, начинающиеся с номера 10.

7.6.3. Каким оператором можно уничтожить программу, введённую в память ЭВМ?

7.6.4. Можно ли с помощью оператора "A" выводить текстовые сообщения?

7.6.5. Какие числа можно вводить с помощью оператора "A"?

7.6.6. Напишите программу, которая по оператору "A" вводила бы число, вычисляла и печатала квадрат, куб, четвертую степень этого числа и само исходное число.

7.6.7. Сделайте так, чтобы программа пункта 7.6.6 после разового выполнения опять выходила на запрос числа.

8. Вывод данных. Оператор TYPE.

Глава содержит сведения о том, как выводить данные в удобной форме на экран телевизора.

8.1. Ещё раз воспользуемся программой, которую мы написали в пункте 7.1., которой и дальше будем пользоваться.

Выполните её несколько раз. Вы обратили внимание на изъяны в форме данных, которые выдаёт эта программа?

Перечислим их:

Данные печатаются не одни под другими, а "где придётся", т.е. в тех местах, где к моменту выдачи данных остановился курсор. Иногда даже часть цифр не умещается в строке и переносится на следующую строку. Читать такие данные утомительно.

У результирующих данных нет поясняющего названия. Стоит определённое число и всё. Несведущему человеку трудно догадаться, что это – время полёта самолёта.

Видимо, необходимо перед числом написать, например, "время полёта:".

В дробной части результата слишком много значащих цифр (4). Такая точность вычисления излишняя. Одной – двух цифр после запятой было бы вполне достаточно.

Попытаемся избавить нашу программу от перечисленных недостатков.

Для того, чтобы значения переменной (или числа) печатались с новой строки, достаточно перед ней (ним) поставить восклицательный знак. В нашем примере достаточно изменить строку 10.40 следующим образом:

```
10.40 T !, X
```

интерпретатор, встретив в строке !, переведёт курсор в начало следующей строки.

Введите модифицированную строку 10.40 и выполните программу. Убедитесь, что теперь ответ постоянно печатается с новой строки.

Вопрос: пусть X=l, Y=2, Z=3, A=4, B=5, как будет выполняться вывод, осуществляемый командной строкой:

```
60.50 T !,X,!,Y,!,Z,A,B,! ?
```

Ответ:

Перед выводом значений переменных, независимо от того, где был до начала выполнения оператора 60.50 курсор, он будет установлен в начало

строки (согласно символу "!"). На новой строке выведется значение переменной X, курсор переместится на новую строку (по второму символу "!"), выведется значение Y, строка опять будет переведена (по третьему "!") и на ней будут напечатаны значения переменных Z, A, B, после чего курсор будет установлен в начало следующей строки (последний "!"). На экране это выгладит так:

```
1.0000
2.0000
3.0000    4.0000    5.0000
```

8.1.1. Для того, чтобы перед значением переменной стояла надпись, необходимо поставить эту надпись перед переменной. Надпись должна быть заключена в кавычки. Надпись отделяется от других операндов оператора "T" запятыми.

Для того, чтобы перед ответом в нашей программе стояла надпись "время полёта:", достаточно написать командную строку 10.40 в виде:

```
10.40 T !,"время полета:",X
```

Замените в программе оператор 10.40 написанным выше оператором и выполните программу. Убедитесь, что ответ всегда печатается с новой строки, а перед ответом стоит нужная надпись.

8.1.2. Для того, чтобы управлять количеством выводимых цифр числа и положением запятой в нём, в операторе "T" предусмотрен указатель формата. Продемонстрируем его действие, вставив в оператор 10.40. Договоримся оставить в ответе две цифры после запятой.

Неразумно также считать, что время полёта превысит 99 часов. Поэтому 4 цифры для изображения ответа задачи вполне достаточно.

Итак, оставим 4 цифры: 2 до запятой и 2 после неё.

Введём указатель формата в командную строку 10.40. После этого она будет выглядеть так:

```
10.40 T !,%4.02,"время полета:",X
```

В хорошо знакомой нам командной строке 10.40 появился новый элемент (%4.02) – это и есть указатель формата. Рассмотрим его подробнее.

В начале указателя стоит символ %, он нужен для того, чтобы интерпретатор мог отличить указатель формата от обычного десятичного числа. Далее, до десятичной точки стоит цифра 4, она указывает количество знаков в результате вычисления. После точки стоит число 02, оно указывает интерпретатору, сколько цифр необходимо оставить после запятой, а остальные, после округления результата, отсечь.

Введём модифицированную строку 10.40 в программу. (Вы также можете воспользоваться оператором MODIFY и вставить указатель формата в строку 10.40, которая уже находится в памяти).

Выполним программу, давая скорости значение 1000 км/час, расстоянию 5000 км. В результате выполнения на экране появится ответ:

```
время полета: 5.00
```

но что это?

Мы просили вывести результат, который должен был содержать 4 цифры, а в нашем результате стоят только 3.

Никакого недоразумения здесь нет. Действительно, казалось бы, результат должен был быть таким: 05.00. Но нуль, стоящий перед цифрой 5, "незначащий". Поэтому интерпретатор, как только встречает нули, стоящие впереди (незначащие), заменяет их пробелами. По этой причине в нашем ответе вместо числа 05.00 появилось число 5.00.

Введём ещё одно понятие, которое нам пригодится в дальнейшем, – список оператора. Список – это то, что пишется справа от оператора, через пробел от него. Элементы списка отделяются друг от друга запятыми.

В данной главе и предыдущих главах мы достаточно подробно описали все элементы списка оператора "T" (TYPE). Есть списки и у других операторов, например, у оператора "A". Например:

```
100.1 A "данные", X,Y,Z,T,M,L
```

Список оператора "A" в командной строке 100.1 – это "данные", X, Y, Z, T, M, L.

8.2. Рассмотрим несколько примеров, демонстрирующих действие указателя формата на форму выходных данных. Выше упоминалось, что числа перед записью на экран округляются таким образом, чтобы, по крайней мере, их целая часть могла разместиться в предписанном формате.

Выполним в диалоговом режиме команду:

```
T !,%5.02,13495.72,3495.79,9.999
```

ввод печатаются с новой строки 3 числа:

```
13496.    3495.8    10.00
```

все они оказались непохожими на свои прообразы в списке оператора "T".

Первое число напечатано без дробной части по той причине, что в указателе формата для размещения числа отведено 5 позиций для цифр, а в исходном числе – 7 цифр. Для того, чтобы результат вписался в формат, интерпретатор отсек всю дробную часть, а чтобы такое отсечение наименьшим образом исказило результат, произошло округление дробной части до целых.

Во втором примере произошло отсечение сотых долей с округлением до десятых. Третье число только округлено.

А что произойдёт, если даже целая часть числа не уместится в предписанном формате? В этом случае интерпретатор представит это число в экспоненциальной форме.

8.3. Пример печати числа в экспоненциальной форме

Введите команду:

```
T !,%,5349.25
```

на экране число в экспоненциальной форме:

```
 +0.534925E+04
```

как его следует читать?

Перед буквой "E" стоит число +0.534925 – это мантисса числа. После буквы E расположен порядок числа (+04). Само число вычисляется из арифметического выражения +0.534925*10^4. Иными словами, чтобы получить число, необходимо запятую в мантиссе сдвинуть на столько знаков, каково значение порядка числа.

Если порядок положительный, запятая передвигается вправо, если порядок отрицательный, запятая передвигается влево. Знак результата такой же, как у мантиссы.

В качестве указателя формата стоит символ (%), без, казалось бы, необходимых цифр, указывающих формат и положение запятой. (В предыдущем примере указатель формата был %4.02). Отсутствие цифр в формате воспринимается интерпретатором, как приказ напечатать результат в экспоненциальной форме.

Если в списке оператора "T" поставить "одинокий" символ %, то все результаты, описанные в списке за таким указателем формата, будут выводиться в экспоненциальной форме.

8.4. На какие элементы списка оператора "T" действует заданный формат?

Рассмотрим ещё пример: продемонстрируем оператор "T", в списке которого содержится указатель формата. Напишем программу:

```
100.1 S X=0.52; S Y=53.62 ;S Z=0.549
100.2 T !,%4.02,X,%5.03,Y,%04.04,Z
```

в этой программе непривычно для вас записана строка 100.1.

В ней записано сразу несколько операторов. "Фокал" допускает такую запись. Строка 100.1 выполняет такие же действия, как последовательность строк, приведённых ниже:

```
S X=0.52
S Y=53.62
```

```
S Z=0.549
```

просто эти три оператора "S" объединены в одну строку 100.1. Такая запись операторов предпочтительнее, т.к. она менее трудоёмка и занимает меньше места в памяти ЭВМ.

Теперь рассмотрим оператор 100.2. В списке этого оператора указатель формата встречается 3 раза:

- перед переменной X, как %4.02;
- перед переменной Y, как %5.03;
- перед переменной Z, как %4.04.

Каждый из перечисленных форматов действует на те переменные, перед которыми он расположен. Если бы мы выполнили программу, приведённую в начале настоящего пункта, на экране появился бы такой результат:

```
0.52       53.620     0.5490
```

из этого примера видно, что область действия каждого указателя формата ограничена теми переменными, которые расположены между ним и следующим указателем формата в списке данного оператора или любого далее встречающегося оператора "T".

8.5. Вас, наверное, уже занимает такой вопрос: а как интерпретатор устанавливает формат, когда в списке оператора "T" совсем не присутствует указатель формата? такие операторы (без указателя формата) нам действительно встречались и достаточно часто в предшествующих главах.

Интерпретатор прекрасно в них разбирается, т.к. действует правило: если в списке отсутствует указатель формата, то "по умолчанию" интерпретатор выводит на экран данные так, как будто в начале списка установлен формат %8.04. (Заметим, что формат %8.04 действует "по умолчанию" только в том случае, если установка других форматов не производилась. В противном случае "по умолчанию" действует последний установленный формат). Например, в списке оператора

```
T 2.5,3.4,%2.00,6.75,8.3,7.09
```

перед данными 2.5, 3.4 нет указателя формата. "По умолчанию" эти данные будут выведены по формату %8.04. Затем, в списке встречается указатель формата %2.00, поэтому остальные данные: 6.75, 8.3, 7.09 будут выведены согласно этому формату. Выполните последний оператор в диалоговом режиме. На экране появятся данные:

```
2.5000    3.4000    7         8         7
```

последние три числа округлены до целых согласно формату %2.00.

В заключении укажем ещё один элемент формата, который вы можете включить в список оператора "T".

$ – этот элемент выводит на экран значения всех переменных программы. Все элементы списка, стоящие после элемента $, игнорируются.

Оператор T $ бывает очень полезен при отладке программы (см. п. 15.2).

8.6. Итоги главы 8.

8.6.1. Введены новые элементы списка оператора "T":

- ! – переводит строку и устанавливает курсор в её начало;
- $ – выводит на экран значения всех переменных программы;
- %W.0N – указатель формата, где

  W – общее количество выводимых цифр

  N – количество цифр после запятой.

Помните, что все элементы списка оператора "T" должны отделяться друг от друга запятыми.

Если в списке стоит текстовое сообщение, заключённое в кавычки, это сообщение выводится на экран.

8.7. Вопросы к главе 8.

8.7.1. Что такое формат выводимых данных?

8.7.2. В какой форме записывается указатель формата?

8.7.3. Какой указатель формата подразумевается "по умолчанию"?

8.7.4. Перечислите все элементы, которые можно включать в список оператора "T".

8.7.5. `T !,"текст 1",!,"текст 2",!,"текст 3"` не выполняя этой команды на ЭВМ, запишите выходную информацию этого оператора на листе бумаги.

8.7.6. Напишите оператор, который, ничего не выводя на экран, переводит курсор в начало следующей строки.

8.7.7. Какая информация должна выводиться в результате выполнения следующих команд:

```
T !,"число с округлением", !,%8.02,1010.3993
T !,"целая часть", !,%8.00,0.325
T !,"результат вычисления:", !,%4.02,25*0.1
```

8.7.8. Какие числа отпечатаются на экране в результате выполнения программы?

```
90.10 S X=1* 100+0.9+0.09+0.009
90.30 T !,%4.01,X,100.999
```

9. Понятие встроенной функции.

9.1. Вернёмся к программе вычисления времени полёта. Приведём ещё раз текст программы со всеми теми изменениями, которые мы вносили в предыдущих главах.

```
10.10 A "расстояние",L
10.20 A "скорость",V
10.30 S X=L/V
10.40 T !,%4.02, "время полета",X
10.50 G 10.10
```

Обратим Ваше внимание ещё на один недостаток этой программы. Она вычисляет время полёта в часах и десятых долях часа, тогда как обычно принято отсчитывать время в часах, минутах, секундах.

Время у нас хранится в переменной X. Пусть для определённости X = 14.3840 часа.

9.2. Определение количества целых часов.

Для нас с вами количество целых часов очевидно. Достаточно взглянуть на число, как мы сразу скажем, что количество целых часов (обозначим его буквой N) равно 14. Для машины процесс выделения целой части числа не так прост, как для нас с вами.

Для этой цели в составе интерпретатора существует программа, которая выполняет функцию выделения целой части числа. Эти и другие подобные программы часто называют встроенными функциями, или просто функциями, как в математике.

Так же, как и у математических, у встроенных функций есть название и аргумент, заключённый в скобки. Даже записываются они так же, как это принято в математике. Имя функции "целая часть числа" – FITR(), в скобках записывается арифметическое выражение, из которого необходимо выделить целую часть.

Напомним, что арифметическим выражением может быть число, переменная, числа и переменные, соединённые знаками арифметических действий.

Например, командная строка 20.10 присвоит "N" целую часть "X".

```
20.10 S N=FITR(X)
```

В нашем случае FITR(14.3840) равно 14.

9.3. Определение количества целых минут.

Обозначим количество целых минут буквой "M". Командная строка 20.20 вычисляет значение "M".

```
20.20 S M=FITR([X-N]*60)
```

Строка 20.20 похожа на предыдущую строку, только в качестве аргумента функции FITR в скобках стоит арифметическое выражение. Прежде чем начать вычислять функцию, интерпретатор вычисляет значение выражения, стоящего в скобках. Это правило справедливо не только для функции FITR, но и для всех остальных встроенных функций интерпретатора "Фокала".

Рассмотрим по частям выражение, стоящее в скобках.

(X-N) – это дробная часть количества часов. Для нашего примера X – N = 14.3840 – 14 = 0.3840 (часов). (X – N)*60 – количество минут; (X-N)* 60 = 0.3840*60 = 23.0400 (мин).

После того, как завершится вычисление арифметического выражения, стоящего в качестве аргумента функции FITR, вычисляется значение функции. Для нашего примера значение функции FITR(23.0400) равно 23 (M=23).

9.4. Вычисление количества секунд.

Обозначим буквой "S" количество секунд. Командная строка 20.30 вычисляет значение "S".

```
20.30 S S=(X-N)*3600-M*60
```

Рассмотрим строку 20.30 подробнее. Сначала обратим внимание на то, что латинская буква S присутствует в строке дважды. Один раз – в качестве оператора, другой раз – в качестве имени переменной.

Никакой ошибки в этом нет. В "Фокале" допускается использовать название операторов в качестве переменных.

Интерпретатор легко разбирается, в каком случае имеется в виду оператор, а в каком – переменная, по их положению в командной строке.

Но если бы мы хотели различить написание оператора от написания переменной, как быть в этом случае?

"Фокал" не ограничивает нас в длине имени переменной, но мы должны помнить, что переменные различаются только по двум первым символам. Например, переменные SEKUND и SE21 для интерпретатора одинаковы (не различимы), для них будет зарезервировано одно и то же место в памяти; для интерпретатора они – одна и та же переменная, поэтому, если Вы хотите написать разные переменные, они должны различаться первыми двумя символами.

Есть ещё ограничения при написании имён переменных: их имена могут состоять из латинских заглавных букв или латинских заглавных букв и цифр, но первым символом должна быть буква.

Переменные не должны начинаться с буквы "F". С буквы "F" начинаются имена функций, например, FITR, FSIN, FCOS, FLOG и т.д. как принято говорить, имена, начинающиеся с буквы "F", зарезервированы.

Приведём несколько имён переменных:

```
A
AI29
IVAN
MAIA
MI10
FEDOR
2HALT
ЯША
```

Последние три имени переменных написаны с ошибками:

- FEDOR – начинается с буквы "F";
- 2HALT – начинается не с буквы;
- ЯША – написана русскими буквами.

В дальнейшем Вы можете назначать переменным имена, состоящие из нескольких символов. Старайтесь, чтобы эти имена были "говорящими", т.е. несли Вам информацию о тех величинах, которые под ними подразумеваются.

В "Фокале" могут применяться кроме простых (скалярных) имён, описанных выше, индексированные имена. Индексированное имя – это имя с индексом, например, X[1]; YAP[7,2]; Z(I,J); T<A*B,D>. Индекс – арифметическое выражение, стоящее в скобках справа от переменной. Если переменная содержит два индекса (двумерная), то индексы должны разделяться запятой. Количество индексов – не должно быть больше двух.

Упорядоченные по индексам (например, по возрастанию индексов) наборы индексных переменных программ часто называют массивами, например, набор переменных

X[1,1]; X[1,2]; X[1,3]

X[2,1]; X[2,2]; X[2,3]

образуют двумерный массив X.

Изменение индекса в "Фокале" ограничено. Индекс может изменяться от -128 до +127 для двумерного массива и от -32768 до +32767 для одномерного массива.

Имена как простых, так и индексных переменных в "Фокале" не требуют специального объявления. Переменная считается объявленной (т.е. ей отводится место в памяти ЭВМ), как только её имя встретится в тексте программы.

Итак, вернёмся к рассмотрению строки 20.30 арифметическое выражение, расположенное после знака равенства, представляет из себя разность между всем количеством секунд, содержащимся в дробной части (X-N)*3600 (в часу – 3600 сек.) и количеством секунд в. целых минутах (м). Таким образом, переменная содержит остаток (в секундах) после вычитания

из времени, затраченного на полет, количества секунд в целых часах и целых минутах.

Для нашего примера:

```
S = (14.3840-14) * 3600 – 23 * 60 =
= 1382.4000 – 1380.000 = 2.4000 (сек).
```

9.5. Вывод результатов.

Для завершения программы нам необходимо вывести переменные N, M, S на экран с помощью приведённых ниже операторов:

```
20.40 T !, "ВРЕМЯ ПОЛЕТА:"
20.50 T %2.00, N, "час", M, "мин", S, "сек"
```

Первый оператор напечатает сначала отроки "ВРЕМЯ ПОЛЕТА:". Второй – выведет переменные N, M, S с соответствующими надписями "час", "мин", "сек".

Итак, приведём нашу программу полностью.

```
10.10 A "РАССТОЯНИЕ", L
10.20 A "СКОРОСТЬ", V
10.30 S X=L/V
10.40 T !, %4.02 "ВРЕМЯ ПОЛЕТА", X
10.50 G 10.10
20.10 S N = FITR (X)
20.20 S M = FITR ([X-N]*60)
20.30 S S = [X-N]*3600 – M*60
20.40 T !, "ВРЕМЯ ПОЛЕТА :"
20.50 T %2.00,N, "час", M, "мин", S, "сек"
20.60 G 10.10
```

Введите и выполните эту программу, программа выполняется так, как прежде, будто в ней нет операторов группы 20.

Где же ошибка?

Да, в программе есть логическая ошибка. Обратите внимание на операторы 10.40, 10.50. Они здесь явно не нужны. Мы просто "забыли" их убрать.

При наличии оператора 10.50 управление по логике программы никак не может быть передано на операторы группы 20.

Удалите операторы 10.40 и 10.50 с помощью оператора "E". Выполните в диалоговом режиме команды:

```
E 10.40 ВВОД
E 10.50 ВВОД
```

Выполните ещё раз программу. Теперь печатается нужный результат:

```
ВРЕМЯ ПОЛЕТА: 14час.23мин.2сек.
```

9.7. Итоги главы 9.

9.7.1. Интерпретатор "Фокал" содержит программы, к которым можно обращаться по именам. Такие программы называются внутренними функциями "Фокала".

9.7.2. Функция "ЦЕЛАЯ ЧАСТЬ ЧИСЛА" – FITR (X) принимает значение целой части абсолютного значения арифметического выражения, заключённого в скобки, со знаком арифметического выражения.

9.7.3. Переменные в "Фокале" могут обозначаться не только одной латинской буквой, но и целыми словами – именами.

Имя переменной должно начинаться с буквы и состоять из латинских букв и латинских букв и цифр.

Имя переменной не должно начинаться с буквы "F".

Имена различаются только двумя первыми символами.

9.8. Вопросы к главе 9.

9.8.1. Проверьте, правильно ли то, что значение арифметического выражения:

```
FITR (2.5+[FITR (3.3)]/2)
```

Равно 4?

9.8.2. Среди имён переменных, приведённых ниже, несколько имён записано с синтаксическими ошибками. Найдите эти имена.

```
SYMVOL, A21, B32, 2SA, SS-4,
WAR, AA1, B79K, SУ36, SУ37, SУ38,
A296, AAA, XXX, FXX, XFX.
```

9.8.3. Найдите среди приведённых в пункте 9.8.2 имён переменных одинаковые (неразличимые) для интерпретатора переменные.

9.8.4. В команде

```
T FITR 2.49
```

При написании функции допущена ошибка. Найдите её.

9.8.5. Какая информация появится на экране после выполнения каждой из команд?

а) `T FITR(1.5^2) ВВОД`
б) `T FITR(1.5)^2 ВВОД`
в) `T 1.^2 FITR(2) ВВОД`
г) `T 1.5^2 ВВОД`

9.8.6. Одинаков ли результат вычисления арифметических выражений?

а) `FITR (2.5*6)`

б) `FITR [FITR (2.5*6)]`

10. Программный режим работы. Циклические программы.

10.1. Вполне возможно, что прочитав предыдущие главы, Вы захотите задать вопрос: "А где же преимущество перед калькулятором?"

Усложним нашу задачу о вычислении времени полёта самолёта и постараемся продемонстрировать на ней одну из главных возможностей вычислений в программном режиме. Это – возможность автоматически повторять нужное количество раз одни и те же действия –

возможность выполнения циклических программ.

Циклические программы позволяют с помощью небольшого количества косвенных, команд выполнять огромное количество операций.

10.2. Существуют два способа автоматической передачи управления от одной косвенной команды на другую.

Первый способ передачи управления – это, так называемый, естественный способ, при котором после выполнения одной команды управление передаётся следующей за ней команде в естественном порядке их следования.

Второй способ – это передача управления с помощью команд передачи управления. Одну из таких команд – команду "G " (GO TO) – мы уже знаем.

Поскольку в циклических программах необходимо неоднократно повторять нужные нам группы косвенных команд, в число команд, образующих эту группу, должны быть включены команды, изменяющие естественный порядок их выполнения команды передачи управления.

Косвенная команда программы: 20.60 G 10.10 изменяет естественный порядок выполнения операторов и передаёт управление на строку с номером 10.10.

Таким образом, все операторы нашей программы образуют цикл. У этого цикла – существенный недостаток: он выполняется бесконечно. ЭВМ не может автоматически закончить его выполнение и перейти к другим действиям. Например, закончить выполнение программы.

Чтобы завершить выполнение этого цикла, Вы нажимали клавишу "СТОП". Нажмите её ещё раз.

Попробуем заставить ЭВМ автоматически завершить выполнение программы после заданного нами условия – числа повторений (переходов) или, как говорят специалисты, организуем цикл с конечным числом проходов.

Выберем число проходов равным 3 и после завершения цикла выведем сообщение "КОНЕЦ ЦИКЛА".

10.3. С этой целью введите одну косвенную команду в начале программы:

```
10.05 S K = 3
```

и замените строку 20.60 тремя строками:

```
20.60 S K = K-1
20.70 I (K) 20.80, 20.80, 10.10
20.80 T ! , "КОНЕЦ ЦИКЛА"
```

Теперь вся программа выглядит так:

```
10.05 S K = 3
10.10 A "РАССТОЯНИЕ", L
10.20 A "СКОРОСТЬ", V
10.30 S X = L /V
20.10 S N = FITR(X)
20.20 S M = FITR([X-N]*60)
20.30 S S = [X-N]*3600-M*60
20.40 T !, "ВРЕМЯ ПОЛЕТА:"
20.50 T %2.0, N, "час", M, "мин ", S, "сек"
20.60 S K = K-1
20.70 I (K) 20.80, 20.80, 10.10
20.80 T !, "КОНЕЦ ЦИКЛА"
```

Введите и выполните эту программу. С её помощью заполните недостающие данные в таблице (время полёта).

№	Расстояние, км L	Скорость, км/час V	Время полёта, час.
1	1500	300	?
2	1500	400	?
3	1500	500	?

Вы обратили внимание на тот факт, что программа выполнила все расчёты только 3 раза, как мы и хотели, и закончилась сообщением "КОНЕЦ ЦИКЛА".

Выведите текст программы:

```
W ВВОД
```

10.4. Обратите внимание на строку 20.70. В этой строке стоит новый оператор "I". Это сокращение имени оператора IF – оператор передачи управления по условию. Через пробел от оператора в скобках записывается арифметическое выражение. В зависимости от знака выражения в скобках, управление будет передано на командную строку:

- номер которой стоит сразу же за скобками, если выражение в скобках отрицательно;

- номер которой стоит вторым от скобки, если выражение равно нулю;
- номер которой стоит третьим от скобки, если выражение больше нуля.

Номера строк, присутствующие в операторе "I", разделяются запятыми.

При написании оператора "I" можно опустить (не писать) третий (или одновременно второй и третий), считая от скобок, номер. При этом переход происходит на следующий после IF оператор.

Для нашего примера в скобках стоит переменная "К". (Посмотрите на оператор с номером 20.70). Если значение "К" станет отрицательным или равно нулю, управление будет передано на строку с номером 20.80, т.е. на завершающую программу командную строку, выдающую на экране сообщение "КОНЕЦ ЦИКЛА" (условие завершения цикла: К <= 0).

Если значение переменной "К" больше нуля, управление передаётся командной строке 10.10 – на начало области действия цикла.

Областью действия цикла называются операторы, которые повторяются многократно в данном цикле.

Цикл в нашем примере, мы в этом убедились, выполняется ограниченное число раз. Почему?

Потому, что при каждом проходе выполнение оператор 20.60 приводит к уменьшению переменной "К" на 1 (К=К-1).

Через три прохода в результате действия этого оператора переменной "К" приводится значение, равное нулю.

Почему именно через три прохода?

Потому, что начальное значение переменной "К" установлено равным 3 оператором, находящимся в строке 10.05.

Обратим Ваше внимание и на то, что этот оператор находится вне области действия цикла. Стоит только включить его в эту область, как цикл станет выполняться бесконечное число раз, т.к. никогда не наступит условие окончание цикла (К<=0).

Изменяя строку 10.05, мы можем присвоить переменной "К" любое нужное нам значение, и, тем самым, установить нужное нам количество повторений. Переменные, выполняющие в цикле роль, аналогичную "К", т.е. переменные, влияющие на количество проходов цикла, называются ПАРАМЕТРАМИ ЦИКЛА.

Перед входом в область действия цикла необходимо задать начальное значение параметрам цикла.

10.5. Пример 1.

Напишем программу, вычисляющую сумму (обозначим её именем "SYM") квадратов первых N членов натурального ряда:

```
SYM=1^2+2^2+3^2+...+ N^2
```

пусть, для определённости, N=80.

С целью демонстрации эффективности применения циклических программ приведём решение этой задачи двумя способами – без применения цикла и с применением цикла.

Самое, казалось бы, простое решение такое:

```
10.10 S SYM=0
20.01 S SYM=SYM+1*1
20.02 S SYM=SYM+2*2
20.03 S SYM=SYM+3*3
20.04 S SYM=SYM+4*4
------------------
------------------
------------------
20.80 S SYM=SYM+80*80
30.10 T SYM
```

программа последовательно суммирует с предварительно "обнулённым" значением "SYM" квадраты нужных нам членов натурального ряда, затем печатает результат.

Прочерками обозначены недостающие операторы (20.05 – 20.79). Согласитесь, что писать их все полностью довольно утомительно, ведь всего их более 80.

Кроме того, программу придётся увеличивать или уменьшать, как только мы дадим другое значение "N", отличное от 80.

Приведём программу, которая более лаконична и лишена упомянутого недостатка:

```
10.10 A "N", N
10.20 S SYM=0
10.30 S K=1
20.10 S SYM= SYM+K*K
20.20 S K=K+1
20.30 I (N-K) 30.10,20.10,20.10
30.10 T !,"сумма равна", SYM
```

очевидно, что эта программа короче предыдущей. Давайте её подробно разберём.

Командные строки группы 10 не включены в область действия цикла, но они выполняют важные действия по подготовке к нему.

Оператор в строке 10.10 даёт нам возможность ввести значение переменной N – числа повторений цикла, установим его равным 80.

Оператор в строке 10.20 присваивает значение, равное нулю, переменной "SYM".

Переменная "SYM" используется в программе в качестве накопителя, иными словами, в ней последовательно накапливается вычисляемая нами сумма квадратов. Это означает, что при каждом проходе операторов цикла к этой переменной будет добавляться квадрат очередного члена натурального рада чисел.

Если "SYM" не сделать равной нулю, то в накопителе перед началом суммирования может быть записано какое-то число. Это число сложится с накопленной нами суммой, исказит её, и программа выдаст неверный результат.

Одно из технических устройств, которое обязательно присутствует в каждой ЭВМ, это – память. В памяти ЭВМ запоминаются и сами программы, и данные, с которыми они работают. Для удобства работы память ЭВМ разбивается на "кусочки", с которыми машине легко оперировать. Такие элементы памяти в вашей ЭВМ называются байтами.

Для того, чтобы легко можно было отличать байты и найти любой из примерно 60000 байтов, имеющихся в памяти ЭВМ, все байты пронумерованы.

Номер, который "достался" байту при нумерации, называют адресом байта, а все возможные в ЭВМ адреса называют адресным пространством.

Замечание: адреса принято записывать в восьмеричной системе счисления. В этой системе используются только цифры 0-7, а 8 и 9 не используются.

Как только "Фокал" встречает в программе новую переменную (первый раз встречающуюся в тексте программы), он резервирует для хранения значения этой переменной нужное количество байтов (по 8 байтов для каждой переменной).

Заметим, что "Фокал" только резервирует память под размещение переменной и ничего в эти байты не записывает, вся информация, которая была в этих байтах до того, как они были назначены для хранения нашей переменной, сохраняется.

Когда мы начинаем выполнять нашу программу, мы, как правило, ничего не знаем, как ЭВМ использовала эти только что зарезервированные байты раньше.

Очень вероятно, что их совокупность соответствует вовсе не нулевому значению переменной. Чтобы избежать ошибки при использовании такой переменной в качестве накопителя, программист обязан предварительно занести нуль в эту ячейку памяти.

Продолжим разбор программы. Оператор строки 10.30 присваивает переменной "К" начальное значение равное 1, точно так же, как мы поступили в программе о вычислении времени полёта самолёта.

Рассмотрим область действия цикла. В нашей программе это командные строки группы 20. Оператор строки 20.10 суммирует содержимое накопителя (SYM) с очередным квадратом члена натурального ряда. При первом проходе использовались начальные значения (SYM=0, K=1) переменных, установленные нами при подготовке к циклу. После первого прохода значение SYM будет равным 1.

Оператор в строке 20.20 увеличивает на 1 значение переменной "K", тем самым этот оператор как бы конструирует очередной член натурального ряда чисел, подготавливая его для следующего прохода. При первом проходе значение "K" увеличится на 1 и станет равным 2. Оператор в строке 20.30 проверяет условие окончания цикла (K > N). Если это условие не выполнено, управление будет передано строке 20.10, где к "SYM" добавится квадрат очередного члена натурального ряда чисел и т.д., до того момента, когда "K" станет больше "N".

Как только это произойдёт, цикл закончится, и будет напечатана итоговая сумма.

10.6. Пример 2

Напишем программу, вычисляющую сумму квадратов нечётных членов натурального ряда, не превышающих M .

Решение:

```
10.10 A "N",N
10.20 S SYM=0
10.30 S K=1
20.10 S SYM=SYM+K*K
20.20 S K=K+2
20.30 I (N-K) 30.10,20.10,20.10
30.10 T !,"сумма равна", SYM
```

программа от предыдущей отличается только оператором строки 20.20, который увеличивает "K" не на 1, как в примере 1, а на 2.

К нечётному числу прибавляется чётное, таким образом конструируются нечётные члены натурального ряда чисел.

10.7. Пример 3

Напишем программу, вычисляющую сумму квадрата чётных членов натурального ряда чисел, не превышающих " N ". От программы примера 2 эта программа отличается только оператором 10.30, который записывается так:

```
10.30 S K=0
```

теперь в переменную "K" оператором 20.20 будут записываться только чётные числа натурального ряда. Поскольку в области действия цикла к переменной

"K" будет прибавляться чётное число 2, суммироваться будут только чётные числа.

10.8. Итоги главы 10.

10.8.1. В главе 10 рассмотрен оператор перехода по условию – оператор "I" (IF).

Оператор "I" служит для передачи управления на одну из командных строк, номера которых перечислены в операторе. Перед выполнением оператора вычисляется значение арифметического выражения, заключённого в скобки. В зависимости от того, меньше, равно или больше нуля значение выражения, управление передаётся на командную строку, номер которой присутствует первым, вторым или третьим по порядку в составе оператора.

Если в операторе отсутствует третий или одновременно второй и третей по порядку номера строк, то в их отсутствии оператор работает так, как будто вместо каждого отсутствующего номера стоит номер строки, следующей сразу за строкой с оператором "I".

Отметим, что номера строк, на которые передаёт управление оператор "I", могут изображаться в этом операторе не только десятичными числами, но и переменными, значения которых должны соответствовать имеющимся в программе строкам. Если такая переменная не соответствует номеру какой-либо строки программы, выдаётся сообщение – "несуществующий номер строки", например, программа

```
100.1 S A=100.30; S B=100.40; S C=100.50
100.2 I (X) A,B,C
100.3 T "X меньше нуля"; G 100.6
100.4 T "X равно нулю"; G 100.6
100.5 T "X больше нуля"
100.6 T !, "конец программы"
```

сработает так же, как будто бы в операторе строки 100.2 вместо переменных "A", "B", "C" стояли номера 100.3, 100.4, 100.5.

10.8.2. Цикл – это такой способ организации выполнения программы, когда одна и та же последовательность операторов выполняется многократно. Такая циклически повторяемая последовательность операторов называется областью действия цикла.

10.8.3. Перед передачей управления в область действия цикла необходимо установить начальные значения параметров цикла.

Операторы, которые устанавливают начальные значения параметров цикла, должны находиться вне области действия цикла.

10.8.4. Параметры цикла – это переменные, которые используются при проверке условий окончания цикла.

10.8.6. Память ЭВМ – это техническое устройство, в котором хранятся (запоминаются) программы и данные.

10.8.7. Вся память ЭВМ разбита на элементы – байты. Все байты пронумерованы, присвоенный байту, номер называется адресом байта.

10.8.8. Совокупность всех возможных в ЭВМ адресов байтов называется адресным пространством.

10.9. Вопросы к главе 10.

10.9.1. Что такое адресное пространство?

10.9.2. Почему перед использованием переменной в качестве накопителя необходимо присвоить этой переменной значение, равное нулю?

10.9.3. Укажите номера командных строк, на которые будет передано управление оператором "I" в фрагментах программ, приведённых ниже, если (X-Y)=0,

А)
```
..............................
88.10 I (X-Y) 10.10,88.20,88.20
88.20 T "пример А"
```
Б)
```
.....................
88.10 I (X-Y ) 10.10
88.20 T "пример В".
```

10.9.4. Что такое область действия цикла?

10.9.5. Почему недопустимо включать в область действия цикла операторы, устанавливающие начальные значения параметров цикла?

10.9.6. Напишите программу вычисления суммы чисел, не превышающих числа N и делящихся на 5, т.е. сумму:

```
0+5+10+15+20+...
```

10.9.7. Напишите программу вычисления суммы третьих степеней чисел, оканчивающихся на 5, но не превышающих N, т.е. сумму:

```
5^3+15^3+25^3+...
```

11. Подпрограммы. Операторы DO, RETURN, QUIT.

Ещё одна возможность экономить память ЭВМ и затраты труда при программировании.

11.1. Вы уже убедились в преимуществах вашей ЭВМ перед калькулятором.

Такое преимущество обеспечено возможностью многократно выполнять фрагменты (последовательности операторов) программ, хранящихся в памяти ЭВМ. Такое повторение обеспечивалось путём организации цикла. Расскажем ещё об одной возможности организации выполнения нужного нам фрагмента программы.

11.1.1. Рассмотрим пример: вычислить время полёта пули, зная расстояние до цели (L) и скорость пули (V). Для определённости, пусть L=200 м, V=300 м/сек.

Вы, наверное, уже готовы задать вопрос – что же нового в этой задаче, она "как две капли воды" похожа на задачу о вычислении времени полёта самолёта.

Действительно, этой программой в принципе можно воспользоваться, введя по запросу программы, приведённой выше, значения V и L. В результате мы получим на экране итоговую строку:

Время полёта: 0 час 0 мин 0 сек.

Выполните программу пункта 10.4 и убедитесь, что результат её работы именно такой.

Нас этот результат не устраивает по следующим причинам:

- нам не нужно вычислять часы и минуты. Ясно, что реальная пуля летит до цели меньше секунды;
- приведена только целая часть секунд, а нас это не устраивает. В данной задаче нам необходимо время в секундах с дробной частью. Можно, конечно, изменить эту программу и устранить указанные недостатки, но тогда мы создадим новую программу и уничтожим старую, а она нам может понадобиться. Можно сохранить старую программу и написать почти такую же новую, но при этом мы впадаем в другую крайность, будем хранить одновременно две почти одинаковые программы, что, конечно, не целесообразно в плане экономии памяти.

11.2. Выход видится в том, чтобы дополнить уже существующую программу операторами, выполняющими нашу задачу, причём, написать их так, чтобы использовались какие-то операторы или группы операторов, ранее написанной программы. Предложим один из возможных вариантов этого решения:

```
10.05 S K=3
10.10 A "расстояние",L
10.20 A "скорость",V
10.30 S X=L/V
20.10 S N=FITR(X)
20.20 S M=FITR([X-N]*60)
20.30 S S=[X-N]*3600-M*60
20.40 T !, "время полета:"
20.50 T %3.0,N, "час", M, "мин", S, "сек"
20.60 S K=K-1
20.70 I (K) 20.80,20.80,10.10
20.80 T !, "конец цикла"
30.10 C начинаются дополнительные операторы!
30.20 R
```

```
40.10 D 10
40.20 D 20.40
40.30 T X
```

Операторы группы 10 и 20 мы уже комментировали в предыдущей главе. Для нас представляют интерес операторы групп 30 и 40. Разберём сначала операторы группы 40.

11.3. В командной строке 40.10 записано сокращённое имя оператора DO – оператора безусловной передачи управления с возвратом.

Оператором D 10 мы даём указание ЭВМ выполнять, начиная с первого по порядку, операторы, группы 10, а после окончания их выполнения – передать управление на оператор, следующий за оператором "D", – на оператор строки 40.20.

Оператор "D" позволяет как бы временно включить в последовательность выполняемых операторов группу операторов (или даже один оператор), написанных совсем в другом месте программы.

Но и с помощью оператора "G" (или оператора "I") мы могли бы заставить выполняться любую группу операторов.

В чём же тогда разница между оператором "D" и, например, оператором "G"?

Разница между операторами "G" и "D" весьма ощутима. Оператор "G" только передаёт управление на какой-либо оператор группы, не "заботясь" о том, куда и как в дальнейшем будет передаваться управление операторами группы. Оператор "G" никак не гарантирует того, что в конце концов управление вернётся к программе, передавшей управление с его помощью.

Аналогичная ситуация возникает и после передачи управления с помощью оператора "I".

Другое дело, когда управление передаётся оператором "D".

Какой бы ни была последовательность передач управления в группе, (если, конечно, там не начнёт выполняться бесконечный цикл) управление будет обязательно передано на оператор, следующий за вызвавшим передачу управления оператором "D". Более того, если в программе нет циклов с бесконечным числом проходов, управление из принявшей его от оператора "D" группы никуда не может быть передано без гарантии "вернуться" к оператору, следующему за оператором "D". Вы сразу же спросите:

А что произойдёт, если группе операторов, которой передано управление оператором "D", в свою очередь встретится оператор перехода "G" или оператор "I"? ведь мы только что утверждали, что эти операторы не "гарантируют" возврата управления.

Работа оператора DO имеет некоторые особенности.

Если в строке или группе строк, на которые передаёт управление оператор DO, находятся операторы перехода GOTO или IF, передающие управление за пределы группы, то они вызывают передачу управления только

на одну строку, после выполнения которой управление перейдёт к оператору, следующему за DO.

Включение в последовательность выполняемых операторов группы операторов из другого места программы называется обращением к подпрограмме, а саму группу вызванных операторов – подпрограммой.

В большинстве случаев программисты выделяют в подпрограммы одинаковые последовательности операторов, часто встречающиеся в разных местах программы.

Создание и использование подпрограмм позволяет экономить время разработки программ, память ЭВМ, делает чтение программы более удобным, а текст программы – более доходчивым.

Вернёмся к нашей программе, Оператор в строке 40.20 D 20.40 передаёт управление не группе операторов, как это происходило при выполнении предыдущего оператора, а только единственному оператору, содержащемуся в строке 20.40, который напечатает текст "время полёта:", после чего управление перейдёт к оператору 40.30.

Почему же оператор "D", стоящий в строке 40.10, выполнил целую группу операторов, а тот же оператор в следующей строке выполнил только единственный оператор?

Потому что в операторе строки 40.10 D 10 за именем оператора, "D" указан номер группы операторов – 10, а в строке 40.20 D 20.40 указан номер конкретного оператора в группе 20, а не вся группа 20.

Напомним, что группа операторов нумеруется целым числом, а любой конкретный оператор в группе – дробным десятичным числом, поэтому, если вы хотите какие-то операторы выделить в подпрограмму, они должны быть объединены в единую группу.

Для этого достаточно, чтобы все номера строк в этой группе имели одинаковую целую часть (число до десятичной точки).

Оператор в строке 40.30 вычисляет время полёта пули в секундах и выдаёт его на экран.

11.4. В программу включены ещё два новых для вас оператора. Посмотрите на строку 30.10, там стоит оператор "C" (COMMENT) (не спутайте его с оператором "G" GOTO). Этот оператор не оказывает никакого влияния на ход выполнения программы, а нужен он только для того, чтобы ввести в текст программы пояснения (комментарии) к ней.

Если вы будете писать программы со сложной логической структурой, необходимо использовать комментарии.

В текст комментариев могут входить любые символы, доступные ЭВМ (в том числе и русские буквы).

11.5. Рассмотрим, строку 30.20 нашей программы. В ней записан оператор "R" (RETURN) – оператор завершения подпрограммы.

Как только ЭВМ встретит оператор "R" в выполняемой ею последовательности операторов, выполнение программы (подпрограммы) прекращается, и управление передаётся на программу, которая вызвала эту подпрограмму. В данном случае наша программа вызвалась на выполнение интерпретатором языка "Фокал", ему же и передаёт управление оператор "R", стоящий в строке 30.20. Для интерпретатора это будет равнозначно тому, что программа завершила свою работу. Интерпретатор после этого выдаёт приглашение к диалогу – *. Чаще всего оператор "R" используется совместно с оператором "D" или функцией FSBR – о ней мы расскажем в главе 14 – и служит для возврата "досрочно" к оператору, следующему за оператором DO или за оператором, содержащим обращение к функции FSBR. Операторы, следующие за оператором "R" в этой же строке, никогда не выполняются, поэтому после него есть смысл размещать только комментарии. Если в группе строк встречается оператор "R", то строки группы, следующие за оператором "R", не выполняются.

Точно такого же результата можно добиться, если в строку 30.20 вставить оператор "Q" (QUIT) – оператор останова выполнения программ.

Разница между операторами "Q" и "R" заключается в том, что оператор. "Q" передаёт управление интерпретатору независимо от того, какой программой вызвана группа операторов, в которой находится оператор "Q".

Оператор "R" передаёт управление той программе (подпрограмме), которая вызвала группу операторов, содержащую оператор "R". Такой программой может оказаться вовсе не интерпретатор.

Оператор "Q" обычно используется для аварийного останова работы программы, например, при отладке. При встрече с ним работа программы останавливается с сохранением всех значений переменных, и на экране печатается символ "*".

Работа программы может быть продолжена с необходимой строки с помощью оператора " G ":

```
G N
```

здесь N – номер строки, с которой следует продолжить выполнение.

Операторы "Q" и "R" не имеют операндов.

11.6. Пример

```
1.1 A Y ; D 3
1.2 T X; Q
3.1 I (Y) 3.2,3.3,3.4
3.2 S X=-1; R
3.3 S X=0; R
3.4 S X=1; R
```

В данном примере помещена программа, печатающая одну цифру из трёх: -1, 0, +1, в зависимости от того меньше, равно или больше нуля переменная Y. Оператор D 3 передаёт управление группе с номером 3.

Оператор в строке 1.2 печатает результаты и завершает программу с помощью оператора "Q".

В группе 3 могут быть выполнены только 2 командные строки – строка 3.1 и одна из строк 3.2, 3.3, 3,4. После выполнения любой из них управление обязательно будет передано на оператор, следующий за DO, т.к. любая из строк 3.2, 3.3, 3.4 содержит оператор "R".

Обратите внимание, оператор "R" стоит в каждой из этих строк после оператора "S", операторы разделены точкой с запятой. Напомним, что одна командная строка может содержать несколько операторов, разделённых точкой с запятой. В программах, которые мы писали до сих пор, мы редко пользовались этой возможностью в целях простоты комментирования этих программ. В дальнейшем при написании, командных строк мы будем записывать в одной строке несколько операторов, специально не оговаривая этого момента.

11.7. Для чего же нам понадобился в данном месте нашей программы оператор "R"? помните, в начале этой главы мы ставили задачу написать программу, которая бы сохранила все функции программы вычисления времени полёта самолёта и смогла бы, при надобности, рассчитать время полёта пули. Представьте себе, что этого оператора нет, т.е. за строкой 30.10 сразу же следует строка 40.10. Пусть мы только хотим, как и в предыдущей главе, вычислить три раза время полёта самолёта. Для этого, как и раньше, мы должны дать команду:

```
G ввод (или G 10.05 ввод).
```

Программа проделает все необходимые нам действия. Напечатает с помощью оператора строки 20.80 сообщение "конец цикла", но не завершится, как это было в предыдущей главе, а начнёт выполнять операторы группы 40.

Но нам не надо вычислять время полёта пули!

Чтобы избежать этих, пока не нужных нам, действий в программе, мы и ввели оператор 30.20 R, который завершает программу, а как же тогда вычислить время полёта пули?

Для этого достаточно ввести команду:

```
G 40.1 ввод,
```

по которой управление будет передано на оператор 40.10. Группу операторов с номером 40 мы уже разбирали.

Итак, чтобы вычислить время полёта самолёта, мы должны передать управление на оператор 10.05, чтобы вычислить время полёта пули – на оператор 40.10. Как говорят, наша программа имеет две точки входа – это командные строки с номерами 10.05 и 40.10.

11.8. Операторы "D" могут быть вложенными. Это означает, что во время выполнения подпрограммы, к которой обратились с помощью оператора "D", в ней может встретится ещё один оператор "D", вызывающий, в свою очередь, другую подпрограмму, и т.д.

Подпрограммы в "Фокале" также могут обращаться сами к себе. Такое обращение называется рекурсивным. Для того, чтобы продемонстрировать вложенные операторы DO и рекурсивное обращение, рассмотрим программу, вычисляющую факториал числа N (N!), т.е. произведение чисел от 1 до N .

```
10.10 A  "N?", N ; C      вводится само число
10.20 D  20; C            обращение к группе 20
10.30 T  !,N,"!=",P,!,C   печать результатов
10.40 R; C                окончание программы
20.10 S  P=1;S X=1;C      установка начальных значений
20.20 D  40;R;C           обращение к группе 40 выход из
                          группы 20
40.10 S  P=P*X;S X=X+1;C  вычисление очередного
                          факториала
40.20 I  (X-N-1) 40.40;C  проверка условия окончания
                          расчётов
40.30 R; C                окончание расчёта
40.40 D  40;R;C           рекурсивное обращение к
                          подпрограмме
```

Все строки этой программы приведены с комментариями, поясняющими функцию командной строки в программе.

Рассмотрим эту программу.

В строке 10.10 расположен оператор "A", с помощью которого вводится значение числа, для которого надо вычислить факториал. Командная строка 10.30 выводит на экран значение результата, а строка 10.40 завершает программу.

В командной строке 10.20 следует обращение к подпрограмме, записанной командными строками группы 20. В строке 20.20 этой группы следует обращение к подпрограмме – группе 40. Можно сказать, что оператор "D" строки 20.20 вложен в оператор "D" строки 10.20. Все предыдущие операторы групп 10 и 20 как бы ведут подготовку к выполнению расчёта, сам же расчёт выполняется подпрограммой группы строк 40.

Оператор строки 40.10 формирует очередное число (X) и умножает на произведение, накопленное при предыдущих проходах – (P). Оператор строки 40.20 обеспечивает окончание расчёта по исчерпании всех необходимых для него чисел.

Новой для нас будет роль, которую в этой программе выполняет оператор "D" в строке 40.40.

Пусть для примера N=3. Первое обращение к подпрограмме осуществляется оператором "D", расположенном в строке 20.20. Но что это?

подпрограмма не успевает завершиться, а её уже опять вызывают на выполнение оператором, который находится в ней же самой – оператором, стоящим первым в строке 40.40. Никакого нарушения логики здесь нет, это и есть рекурсивное обращение к подпрограмме.

Поясним только фразу "не успевает завершиться". Выполнение подпрограммы завершается при одном из условий:

- если выполнены все операторы подпрограммы, подлежащие выполнению согласно её логике;
- если в числе выполняемых операторов встретился оператор завершения подпрограммы – оператор "R" или "Q".

Поскольку при первом обращении к подпрограмме ни то, ни другое условие не выполнено, мы имеем право написать применительно к подпрограмме: "не успела завершиться".

В нашем примере будет ещё, по крайней мере, один такой же "незавершённый" вызов на исполнение группы командных строк 40.

Завершится ли когда-нибудь наша программа?

Да. Завершится, когда выполнится условие (X-N-1)=0 (см. строку 40.20). Только процесс завершения рекурсивной подпрограммы состоит не из одного шага, как у обычной подпрограммы, а из целой серии шагов. В нашем примере к подпрограмме, прежде чем она завершится первый раз, будет произведено три обращения, три же раза будет осуществлён и выход из неё.

Действительно, после того, как сработает оператор "R", управление будет передано на оператор, следующий за оператором "D", вызвавшим эту программу последним по времени.

Таким оператором будет оператор "R", стоящий в строке 40.40. После его выполнения управление опять вернётся к нему же, поскольку и перед этим подпрограмму вызвал оператор "D", стоящий в той же командной строке.

В конце концов наступит момент, когда последним оператором, вызвавшим подпрограмму, окажется оператор, расположенный в строке, которым завершится выполнение командных строк группы 20. Это будет означать, что наша рекурсивная программа завершилась.

Многократное вложение, операторов "D" и количество рекурсий ограничено в "Фокале" – 44, если же вы превысите это число, будет выдано сообщение – "переполнение стека". (Стек – это область памяти для запоминания служебной и иной информации о работе программы).

11.9. Итоги главы 11.

11.9.1. Рассмотрен оператор "D" (DO) – оператор перехода с возвратом.

Оператор предназначен для передачи управления на заданную командную строку или группу строк с последующим возвратом на оператор, следующий за DO. Оператор может быть использован в режиме прямых и косвенных команд.

В качестве номера строки или группы строк, куда передаётся управление оператором "D", может быть использована переменная, имеющая значение соответствующего номера.

Например, оператор D N передаёт управление на строку 50.10, если N=50.10, т.е. он сработает так же, как оператор D 50.10.

Если в строке или группе строк, на которые передаётся управление оператором "D", находятся операторы "G" или "I", то они вызывают передачу управления только на одну строку, после выполнения которой управление перейдёт к оператору, следующему за "D".

11.9.2. Рассмотрена техника создания и выполнения подпрограмм.

В подпрограмму объединяются операторы, принадлежащие одной группе, т.е. любая последовательность косвенных команд, имеющих одну и ту же целую часть в числах, изображающих номера строк этих команд. Обращаться к такой последовательности операторов (вызывать её на исполнение) можно с помощью оператора "D".

11.9.3. Рассмотрен оператор "C" (COMMENT), который не выполняется интерпретатором "Фокала", а служит только для того, чтобы снабдить комментариями текст программ, написанных на "Фокале".

11.9.4. Рассмотрен оператор "R" (RETURN) – оператор выхода из подпрограммы. Оператор служит для досрочного возврата из подпрограммы к оператору, следующему за оператором "D" или за оператором, содержащим обращение к функции FSBR.

11.9.5. Рассмотрен оператор "Q" – оператор останова программы. После выполнения оператора "Q" управление будет передано интерпретатору, на экране появится символ "*".

11.10. Вопросы к главе 11.

11.10.1. Перечислите функции и возможности оператора "D".

11.10.2. Что такое подпрограмма?

11.10.3. Зачем нужны подпрограммы? .

11.10.4. Как можно "досрочно" вернуться из подпрограммы к оператору, следующему за оператором "D", вызвавшим эту подпрограмму на исполнение?

11.10.5. Опишите особенности работы операторов передачи управления – "G" и "I", если они располагаются в группе, вызванной на исполнение с помощью оператора "D"?

11.10.6. Какой результат выведет на экран программа, приведённая ниже?

```
70.10 D 80; T !,X,R
80.10 G 90.10
80.20 T !,"X="
```

```
90.10 S X=5
90.20 S X=X*X
90.30 S X=X+75
100.1 R
```

11.10.7. Какой результат выведет на экран эта программа?

```
70.10 D 90;T !,"X=",X;R
90.10 S X=5
90.20 S X=X*X
90.30 S X=X+75;R
90.40 S X=X-100
```

12. Оператор для организации циклов FOR

12.1. Вернёмся ещё раз к написанию циклических программ. Важность циклов для составления программ мы уже не раз подчёркивали. Все современные языки программирования содержит специальные операторы для организации циклов.

Продемонстрируем его действие на программе о времени полёта самолёта. В главе 10 мы рассматривали усложнённый вариант этой задачи (см. п. 10.3): зная расстояние – 1500 км, нам необходимо было рассчитать время полёта самолёта для скоростей 300, 400, 500 км/час. Приведём более простое решение, чем то, которое мы приводили в пункте 10.3.

```
10.10 A "расстояние",L ;C ввод расстояния
10.20 F V =300,100,500;D 20
10.30 T !,"конец цикла";R
20.05 S X=L/V
20.10 S N=FITR(X)
20.20 S M=FITR([X-N]*60)
20.30 S S=[X-N]*3600-M*60
20.40 T !,"время полета"
20.50 T %3.0, N, "час", M, "мин", S, "сек"
```

12.2. Введите эту программу в ЭВМ. Если в памяти ЭВМ была какая-либо другая программа, не забудьте удалить ненужные вам операторы.

Выполните эту программу

12.3. Рассмотрим подробно введённый нами в программу новый оператор "F" (FOR) – оператор цикла. Именно оператор F позволил нам сократить и саму программу, и количество манипуляций, которые мы производили с пульта, выполняя программу (по сравнению с программой п. 10.3).

Выведите хранящуюся в памяти программу на экран командной W "Ввод".

Присмотритесь к оператору "F", расположенному в строке 10.20.

Через пробел от имени оператора помещено равенство V=300, оно интерпретируется как задание начального значения параметру V. В правой части равенства может быть помещено не только число, правая часть может быть любым допустимым в языке "Фокал" арифметическим выражением.

Выражениями могут быть и все следующие операнды оператора.

Через запятую от только что рассмотренного равенства расположено число 100. Это – шаг изменения параметра цикла (переменной V). Откройте ваше руководство в том месте, где представлена программа п. 10.3. В этой программе параметром цикла была переменная "K". Мы её изменяли в операторе строки 20.60 S K=K-1.

В примере, который мы сейчас разбираем, тоже есть свой параметр – переменная "V ". Как уже указывалось в предыдущих главах, с каждым проходом значение параметра должно меняться.

Оператор "F" автоматически меняет значение параметра цикла при каждом новом проходе на величину, равную значению выражения, которым представлен шаг изменения параметра цикла.

В нашей программе такое выражение представлено числом 100.

При первом проходе области действия цикла переменная "V" будет иметь начальное значение, равное 300. Перед вторым проходом значение переменной автоматически будет увеличено на величину шага изменения параметра цикла – 100 и станет равным 400, перед третьим проходом значение опять увеличится и станет равным 500 и т.д.

Выражение, определяющее шаг изменения параметра цикла, может быть опущено. По "умолчанию" интерпретатор установит его равным 1.

А когда значение параметра цикла станет больше, чем конечное значение параметра, цикл закончится.

Конечным значением параметра нашего цикла является число 500, которое расположено сразу после шага через запятую. Допускается, как и предыдущие операнды записывать конечное значение параметра цикла в виде арифметического выражения. Например, командная строка

```
10.30 S Z=250;F V=Z+50, (Z-50)/2, Z*2;D 20
```

будет выполнять такие же действия, как командная строка 10.20, и может вполне, заменить её в нашей программе. Действительно, значение "Z" установлено равным 250 и все выражения, записанные в операндах оператора "F" строки 10.30, имеют те же значения, что и соответствующие им операнды оператора "F" строки 10.20.

Следует отметить, что значения всех переменных, входящих в выражения операндов, записанных в операторе "F", должны быть определены до его выполнения.

У оператора "F" есть особенность, которую необходимо помнить: цикл, образованный этим оператором, не может быть закончен (например, с помощью передачи управления за область действия цикла операторами "G" или "F") раньше, чем исчерпается параметр цикла. Иными словами, для окончания цикла необходимо, чтобы параметр цикла стал больше, чем его конечное значение.

А как же "досрочно" завершить цикл, если возникла такая необходимость?

Можно "обмануть" оператор "F".

Дело в том, что параметр цикла – это обычная для нас переменная, с которой мы в области действия цикла можем обращаться по своему усмотрению. Мы знаем, что для окончания цикла необходимо, чтобы параметр цикла стал больше конечного значения. При необходимости внутри области действия цикла мы можем установить значение параметра цикла, заведомо большее его конечного значения, и тем самым "спровоцировать" окончание цикла.

12.4. Что является областью действия цикла?

Областью действия цикла являются операторы, которые расположены в той же самой командной строке, что и оператор "F", но после него.

В строке 10.20 после оператора "F" расположен оператор D 20. Этот оператор выполняет подпрограмму, записанную операторами группы 20, которые формально не входят в область действия цикла.

Фактически операторы группы 20 выполняют основные действия в нашем цикле – делают необходимые расчёты и печатают результаты. Этим самым мы ещё раз подчеркнём те удобства, которые нам представляет использование оператора "D". Операторы группы 20 подробно описаны в предыдущих главах (6, 7, 8).

Если вам в этих операторах что-то непонятно, обязательно вернитесь и ещё раз прочтите указанные главы. Добейтесь того, чтобы у вас не было непонятных мест.

Продемонстрируем действие оператора "F" на задачах, которые мы решали в примерах главы 10.

Пример 1.

Напишем программу, вычисляющую сумму квадратов первых членов натурального ряда чисел:

```
1^2+2^2+3^'2+...+N^2.
```

```
10.1 A "введите значение N",N;C с пульта вводится N
10.2 S SYM=0; F K=1,N;S SYM=SYM+K*K
10.3 T !, "сумма равна:", SYM
```

В командной строке 10.1 вводится значение N. Весь расчёт осуществляется командной строкой 10.2. Первый оператор этой строки "обнуляет" переменную SYM, которая используется в качестве накопителя.

Второй оператор – "F" организует цикл. Начальное значение параметра цикла равно 1 (K=1). Шаг изменения параметра в операторе опущен, "по умолчанию" он равен 1. Конечное значение параметра цикла – значение, хранящееся в переменной "N".

Область действия цикла – оператор S SYM=SYM+K*K, который при каждом проходе добавляет к накопителю возведённое в квадрат значение переменной "K". Поскольку шаг равен 1 и его начальное значение равно 1, "K" будет с каждым проходом последовательно пробегать значения 1,2,3, ..., N, N+1.

На (N+1)-ом шаге значение "K" станет равным N+1, т.е. больше, чем конечное значение параметра цикла. Как только это произойдёт, цикл закончится. Это означает, что управление будет передано следующему оператору, т.е. оператору строки 10.3, который напечатает результат.

Пример 2.

Вычислить сумму квадратов нечётных членов натурального ряда чисел, не превышающих N: 1^2+3^2+5^'2+...+N^2. Программа от предыдущей отличается только тем, что шаг изменения параметра будет равен не 1, а 2. Строка программы 10.2 запишется так:

```
10.2 S SYM=0; F K=1,2,N; S SYM=SYM+K*K
```

Пример 3.

Вычислить сумму квадратов чётных членов натурального ряда чисел. От примера 2 программа будет отличаться только начальным значением параметра, оно будет равным 0. Строка 10.2 будет выглядеть так:

```
10.2 S SYM=0; F K=0,2,N; S SYM=SYM+K*K
```

Приведём пример с вычислением тригонометрических функций.

Пример 4.

Составить таблицу функций SIN и COS, начиная от 0 до π/2 радиан. Шаг таблицы – 0.1. Задачу решает командная строка:

```
10.10 F X=0,0.1,3.141/2;T !,X,FSIN(X),FCOS(X)
```

Поясним программу примера 4. Первым оператором строки 10.10 является оператор F X=0,0.1,3.141/2. Смысл его понятен – организовать цикл, начальное значение параметра ("X") которого равен 0, шаг – 0.1, конечное

значение – π/2 (π=3,141). Областью действия этого цикла является оператор T .... который печатает результаты.

12.5. В предыдущей главе мы рассмотрели возможности вложенных операторов "D" (см. п. 11.8).

А можно ли вкладывать циклы?

Вложение циклов, т.е. написание таких циклических программ, у которых в область действия одного цикла включается другой цикл, в область действия которого, в свою очередь, может входить третий цикл и т.д., часто используется при разработке программ.

Пример 5.

Сосчитаем значения функции SIN от 0 до 89 градусов через 1 градус и выведем эти значения в виде таблицы:

Таблица от 0 до 89 градусов

	0	10	20	30	40	50	60	70	80
0	0.00	0.17	0.34	0.50	0.64	0.77	0.84	0.87	0.9
1									
2									
3									
4									
5				0.57 = SIN(30+5) = SIN(35)					
6									
7									
8									
9	0.16	0.33	0.48	0,63	0.75	0.86	0.93	0.98	1.0

Приведённая ниже программа даёт необходимые расчёты и печатает данную таблицу.

```
10.10 T !,"    таблица SIN(X) от 0 до 89 градусов",!
10.20 F I=1,64;T "-"; C печатается черта
10.30 T "   "
10.40 F I=0,10,80; T "   ",%2.00,I; C шапка таблицы
10.50 T !
10.60 D 10.2; C печатается горизонтальная черта
10.70 F Y =0,9; T !,%1.00, Y, " : "; D 20
10.80 T !,D 10.2;R
20.10 F X=0,10,80;T %3.02,FSIN[(X+Y)*3.14/180]
```

В данной программе цикл, изображённый в строке 10.7, включает в себя с помощью оператора D 20 цикл, изображённый в строке 20.10. Это и есть вложение циклов. Мы предлагаем вам разобрать самостоятельно пример 5.

12.6. Итоги главы 12.

12.6.1. Рассмотрен оператор цикла "F" (FOR).

В операторе "F" за именем оператора следует через пробел операнды, устанавливающие:

- имя и начальное значение параметра цикла, имя отделено от параметра знаком "=";
- шаг изменения параметра цикла;
- конечное значение параметра цикла.

Все значения могут быть представлены в виде арифметических выражений. Все операнды, присутствующие в операторе "F", должны бить разделены запятыми. Операнд – шаг изменения параметра – может быть опущен. По "умолчанию" значение шага равно 1.

Предупреждение: все переменные, которые включены в арифметические выражения оператора "F", должны быть определены до обращения к оператору.

12.6.2. Областью действия цикла являются операторы, расположенные после оператора "F" в той же командной строке.

12.6.3. В области действия цикла можно использовать параметр цикла как обычную переменную.

12.7. Вопросы к главе 12.

12.7.1. Для чего предназначен оператор "F"?

12.7.2. Как в операторе "F" задаются параметр цикла, шаг изменения параметра, конечное значение параметра?

12.7.3. Что является областью действия оператора "F"?

12.7.4. Каковы будут значения переменных X и Z после окончания программы, приведённой ниже?

```
100.1 F Z=0,0.01,6.28,D 110
100.2 R
110.1 S X=SIN(Z)
110.2 I (X-0.5) 110.4
110.3 Z=100;R
110.4 T !,"SIN", Z ,"=",X
```

13. Управление магнитофоном, оператор LIBRARY

13.1. Есть возможность надолго запомнить информацию, которую вы храните в оперативной памяти ЭВМ. Для этой цели служит подключаемый к ЭВМ магнитофон.

13.2. В руководстве по эксплуатации перечислены типы магнитофонов, совместных с ЭВМ. Воспользуйтесь инструкцией по эксплуатации

магнитофона и руководством по эксплуатации ЭВМ и подключите магнитофон к ЭВМ.

Включите ЭВМ и магнитофон. Вставьте в магнитофон исправную чистую кассету.

Информация, которая может храниться на магнитной ленте может быть двух видов:

- текст программ;
- данные, с которыми работают программы.

13.3. Введите с пульта ещё раз программу вычисления времени полёта самолёта. Текст которой приведён в п. 12.1. Выполните программу и убедитесь, что она введена правильно.

13.4. Установка уровня записи.

Наберите на клавиатуре ЭВМ команду, не нажимая клавиши "Ввод":

```
L S PROG
```

Нажмите на передней панели магнитофона клавиши: (пуск) и (запись) (клавиши одновременно должны находиться в нажатом состоянии!).

Установите на магнитофоне уровень записи.

С этой целью нажмите на клавиатуре ЭВМ клавишу "Ввод" и пока идёт запись, по индикатору, установите в соответствии с инструкцией по эксплуатации магнитофона необходимый уровень записи.

Помните: пока уровень записи не установлен, нет гарантии, что информация правильно запишется на магнитную ленту (МЛ).

На экране появился символ "*"?

Если нет, подождите, пока первая попытка записи информации закончится, и ЭВМ нам выдаст приглашение к диалогу. Если вы правильно подключили магнитофон и установили уровень записи, ЭВМ готова к записи информации на МЛ.

13.5. Наберите на пульте команду:

```
L M ввод
```

смысл её будет понятен в дальнейшем.

Перемотайте МЛ таким образом, чтобы магнитная головка находилась на свободном от записи участке ленты.

13.6. Запись информации на МЛ.

Нажмите клавиши магнитофона (пуск) и (запись)

Ещё раз введите команду:

```
L S PROG ввод.
```

На МЛ записывается текст вашей программы, после завершения записи на экране появится символ "*". Запись закончена!

Оператор "L" (LIBRARY) осуществляет обмен информации между ЭВМ и магнитофоном.

При всяком обмене информацией между внешним устройством (магнитофоном) и ЭВМ необходимо указать направление обмена, т.е. записывать информацию из памяти ЭВМ на МЛ, или же, наоборот, читать информацию с МЛ в памяти ЭВМ.

Указание о направлении обмена задаётся стоящим через пробел от "L" подоператором. В нашей команде таким подоператором является буква "S" (не путайте подоператор "S" с именем оператора "S" – присвоить).

Подоператор "S" (SAVE) указывает ЭВМ, что предстоящий обмен информацией – это запись на МЛ текста программы, хранящейся в момент обмена в памяти ЭВМ.

Мы присвоили записанной на МЛ программе имя PROG. Саму информацию, которую мы записали с помощью одного оператора "L" принято называть

файлом,

а имя, записанное операндом в операторе, -

именем файла.

После выполненной нами команды L S PROG , на МЛ хранится файл с именем PROG, содержащий текст программы вычисления времени полёта. Имя файла вы можете задавать произвольно, следует только помнить, что его длина не должна превышать 15-ти символов, а каждый из символов должен быть либо буквой, либо цифрой.

Буквы в имени могут быть как латинские, так и русские, как заглавные, так и прописные.

На ленте могут быть записаны несколько файлов с одним и тем же именем. Обычно одним и тем же именем именуют одинаковые файлы, записанные (продублированные) на МЛ с целью повышения надёжности хранения информации.

Если вы неверно зададите имя файла, ЭВМ выведет сообщение "неправильное имя файла".

Продублируйте ещё раз запись вашей программы на МЛ. Поскольку программа занимает сравнительно немного ленты, а гарантия того, что мы сможем прочесть записанную информацию, резко повышается. Рекомендуем вам и в будущем записывать на МЛ несколько дублей информации (от 2 до 4).

Итак, ещё раз выполним команду:

```
L S PROG.
```

13.7. Чтение информации с МЛ.

Попробуем прочесть информацию, которую только что записали на МЛ. Нажмите на панели магнитофона клавишу перемотки "назад"

Лента не перематывается.

Не волнуйтесь, ваш магнитофон исправлен, просто сейчас он контролируется ЭВМ, которая после окончания предыдущей записи автоматически выключила мотор магнитофона.

Для того, чтобы перемотка работала, нам необходимо дать ЭВМ команду на включение мотора. Такой командой является:

```
L M ввод.
```

"M" (MOTOR) – подоператор оператора "L", который предусмотрен для включения мотора магнитофона.

Отметим, что при записи информации или при её чтении не обязательно выдавать ЭВМ команду "L M", в этих случаях мотор магнитофона включается и выключается автоматически.

В противоположность команде, запускающей мотор магнитофона, существует специальная команда, останавливающая мотор: "L R".

"R" (RESET) – подоператор оператора "L", останавливающий мотор магнитофона.

С помощью клавиши перемотки МЛ на панели магнитофона перемотайте ленту таким образом, чтобы искомая информация оказалась дальше того участка ленты, на котором будет установлена магнитная головка.

Отожмите все клавиши на панели магнитофона. Наберите на пульте ЭВМ команду:

```
L G PROG
```

нажмите на панели магнитофона клавишу "пуск", а на клавиатуре ЭВМ – "Ввод". Теперь ждите, когда ЭВМ найдёт и прочтёт файл с заданным именем PROG.

Надеемся, вы догадались, что операндом команды L G PROG является имя искомого файла. Первый подоператор – "G" (GET), как и в предыдущем случае, задаёт тип обмена – чтение с МЛ в память ЭВМ.

Как происходит чтение?

ЭВМ читает имена всех встречающихся на МЛ файлов и сравнивает их с именем, записанным в операторе "L". Если имя прочитанного файла с ним не совпадает, файл не читается, на экране отпечатывается его имя.

Если же встретился искомый файл, он читается в память, осуществляется контроль правильности хранения и чтения информации.

В случае верного чтения на экран выводится имя искомого файла и символ "*". Если же информация считана неверно, выдаётся сообщение:

```
ОШИБКА КОНТРОЛЬНОЙ СУММЫ.
*
```

В случае появления указанного сообщения, вам необходимо повторить снова ввод информации, и если она опять не будет читаться правильно, попытайтесь считать её с дубля, если такой есть.

13.8. Продолжим описание оператора "L". Предположим, что вы забыли, какие файлы написаны на кассете. Для того, чтобы это узнать, достаточно прочитать имена этих файлов. Такое чтение можно осуществить с помощью команды

```
L F (имя)
```

"F" (FGET) – это подоператор оператора "L", предписывающий машине осуществить фиктивное чтение с МЛ (имя) – любое имя.

ЭВМ не читает информацию, которая хранится в файле, она читает и выводит на экран только имена файлов, которые встречаются ей на МЛ.

При этом мы не рискуем, как в случае чтения имён файлов с помощью оператора L G (имя), что потеряем информацию, хранящуюся в памяти в момент чтения. В то же время имена всех файлов на МЛ будут выведены на экран, и вы получите представление о том, что это за кассета.

Во время автоматического поиска информации на МЛ вы можете вмешаться в этот процесс и перемотать ленту в любом направлении, не прерывая работы ЭВМ.

Это можно делать в том случае, если вы знаете, где, примерно, хранится информация на МЛ, и можете с помощью перемотки кассеты ускорить поиск.

13.9. Вывод данных на МЛ.

Для программиста вопрос сохранения данных часто бывает очень важен.

Предположим, вы обрабатываете какие-либо длинные, наборы данных и вводите их с пульта по запросу программы. У вас могут возникнуть следующие трудности:

- вы не успеете по каким-либо причинам ввести все данные, вынуждены будете выключить ЭВМ и затереть таким образом всё, что вы ввели до с сих пор в память;
- данные, которые вы вводите, не помещаются в памяти, ведь она не безгранична;
- необходимо сохранить введённые данные для повторного их использования.

Эти трудности можно преодолеть, используя операторы ввода и вывода данных. Известно, что всем данным, которые мы ввели в ЭВМ или получили в результате расчётов, соответствуют имена, присваиваемые им "Фокалом" –

имена переменных. Для запоминания значений всех переменных программ достаточно написать оператор

```
L O (имя)
```

латинская буква "O" – это имя подоператора "O" (OUTPUT) оператора " L ". Который предписывает ЭВМ записать на МЛ значения всех переменных программы.

(Имя) – это имя, которое вы хотите присвоить файлу, содержащему переменные программы.

13.10. Чтение данных с МЛ в память ЭВМ.

Чтобы прочесть записанные нами данные в память, выполните следующие действия:

- перемотайте МЛ таким образом, чтобы магнитная головка была установлена на участке ленты, предшествующем участку, на котором имеется нужный нам файл;
- наберите на клавиатуре ЭВМ команду L I (имя);
- отожмите все клавиши, а затем нажмите клавишу "пуск" на панели магнитофона;
- нажмите на клавиатуре ЭВМ клавишу, "Ввод".

Данные читаются с МЛ и записываются в память ЭВМ. При правильном завершении записи данных ЭВМ выдаёт на экран символ "*". При этом автоматически устанавливается связь между переменными программы и прочитанными данными.

В операторе L I (имя):

"I" – подоператор, предписывающий ЭВМ читать с МЛ в память ЭВМ данные.

(имя) – имя файла, в котором расположены данные.

Ещё раз отметим, что для надёжной записи файлов на МЛ необходимо пользоваться исправными магнитофоном и кассетами и записывать на ленту несколько дублей файла.

Упражнение 1.

Выполните оператор E A ввод. Это необходимо, чтобы затереть программу, которая находилась в памяти. Попробуем ввести ещё раз с МЛ в память ЭВМ программу, записанную в файле с именем PROG.

13.10.1. Включите мотор магнитофона командой:

```
L M ввод
```

13.10.2. Перемотайте ленту, установив головку магнитофона перед искомым файлом.

13.10.3. Наберите на пульте ЭВМ:

```
L G PROG
```

13.10.4. Нажмите на панели магнитофона клавишу "пуск", все остальные клавиши должны находиться в отжатом состоянии.

13.10.5. Нажмите клавишу ЭВМ "Ввод". Если информация прочиталась неверно, попытайтесь прочесть её с дубля. Для этого, не перематывая ленту, повторите действия по пунктам 13.10.3, 13.10.4, 13.10.5. В случае корректного завершения чтения ЭВМ должна выдать символ "*" без диагностического сообщения.

Храните разрабатываемые Вами программы на МЛ, они могут Вам понадобиться в будущем. Со временем у Вас накопится целая библиотека программ. С помощью кассет Вы можете обмениваться программами с другими пользователями микро-ЭВМ "Электроника БК 0010".

ПРЕДУПРЕЖДЕНИЕ:

Дублирование программ и перезапись их с кассеты на кассету допускается только через память ЭВМ и не допускается непосредственной перезаписью с магнитофона на магнитофон.

13.11. Итоги главы 13.

13.11.1. Рассмотрен оператор, служащий для обмена информацией между магнитофоном и ЭВМ – "L" (LIBRARY). Рассмотрены следующие возможности применения этого оператора:

L	S <имя файла>	- запись текста программ на МЛ;
L	G <имя файла>	- чтение текста программ с МЛ в память ЭВМ;
L	O <имя файла>	- запись значений переменных, на МЛ;
L	I <имя файла>	- чтение с МЛ в память ЭВМ значений переменных;
L	M	- включение мотора магнитофона;
L	B	- выключение мотора магнитофона.

Где:

<имя файла:> – имя, которое присваивается набору данных, хранящихся на МЛ, для их идентификации. Это – любая последовательность из букв и цифр, начинающаяся с буквы. Длина имени не должна превышать 15 символов. В имя файла могут входить как заглавные, так и строчные буквы латинского или русского алфавитов, например,

A12345T78Б – имя состоит из букв и цифр;

АБВГ – имя состоит из букв русского алфавита.

13.12. Вопросы к главе 13.

13.12.1. Перечислите основные функции оператора "L".

13.12.2. Что такое имя файла? Какие ограничения существуют в "Фокале" на написание имён файлов?

13.12.3. Опишите, как производится запись на МЛ с помощью подоператора "S" оператора "L".

13.12.4. Опишите, как происходит чтение текста программы с МЛ с помощью подоператора "G" оператора "L".

13.12.5. Опишите операторы запуска и останова монитора магнитофона.

13.12.6. Напишите оператор, вызывающий фиктивное чтение с МЛ.

13.12.7. Напишите оператор, осуществляющий запись данных на МЛ.

14. Встроенные функции. Оператор X.

14.1. Вы уже сталкивались в предыдущих главах с встроенными функциями FITR (X), FSIN(X), FCOS (X).

Встроенные функции предназначены для вычисления часто встречающихся математических функций, преобразования кодов, работы с графической информацией и выполнения других операций. Встроенные функции составляют библиотеку стандартных программ языка "Фокал".

Обращение к встроенным функциям осуществляется по именам. Аргументом функции является выражение, стоящее в скобках справа от имени функции. Сами функции тоже могут входить в состав арифметических выражений. Допускается рекурсивное обращение к функции.

Библиотека стандартных функций языка "Фокал" включает 21 функцию.

FSIN (X)	- синус (аргумент в радианах);
FCOS (X)	- косинус (аргумент в радианах);
FTAN (X)	- тангенс (аргумент в радианах);
FASIN (X)	- арксинус (|X| 4=1);
FACOS (X)	- арккосинус (|X| <=1);
FATAN (X)	- арктангенс;
FLOG (X)	- натуральный логарифм (X> 0);
FEXP (X)	- показательная функция;
FSGN (X)	- знаковая часть числа;
FITR (X)	- целая часть числа;
FABC (X)	- абсолютная величина числа;
FRAN (X)	- генератор случайных чисел;
FSQT (X)	- корень квадратный;
FCHR (X1,X2,...Xn)	- ввод/вывод символьной информации;
FX (КОП,АДРЕС,ДАННЫЕ)	- управление общей шиной;

FSBR (ГРУППА,АРГУМЕНТ) - функция, программируемая пользователем;
FK (КОП,X,Y) - установка курсора по координатам X и Y;
FT (КОП,X,Y) - формирование точки по координатам X и Y;
FV (КОП,X,Y) - формирование вектора по координатам X и Y;
FP (КОП,МАСКА) - работа с портом ввода/вывода.

Другие функции, не вошедшие в библиотеку стандартных программ, можно применять как функции пользователя.

14.2. Математические функции.

Для вычисления синуса угла предназначена функция FSIN.

Формат:

```
FSIN(X)
```

где "X" – значение аргумента в радианах. Может быть арифметическим выражением. Пример:

```
* T FSIN (3.14159/4) ввод
0.7071 *
```

Если угол задан в градусах, то для преобразования аргумента используется множитель (π/180), где π – число Архимеда, π=3.14159. Пример:

```
* T FSIN (45*3,1415/180) ввод
0.7071 *
```

Для вычисления косинуса угла предназначена функция FCOS.

Формат:

```
FCOS(X)
```

где "X" – значение аргумента в радианах. Может быть арифметическим выражением. Пример:

1)
```
T FCOS (2x3.14159) ввод
1.0000 *
* T FCOS (0.5000) ввод
0.8776 *
```

2) косинус угла в градусам :

```
* T FCOS (45*3.14159/180) ввод
0.7071 *
```

Для вычисления тангенса угла предназначена функция FTAN.

Формат:

```
FTAN (X)
```

где "X" – значение аргумента в радианах. Может быть арифметическим выражением. Примеры:

1)
```
T FTAN (0.87) ввод
1.1853 *
```

2) тангенс угла в градусах:
```
* T FTAN (45*3.14159/180) ввод
1.0000 *
```

Для вычисления арксинуса предназначена функция FASIN. Если аргумент по модулю больше 1, выдаётся сообщение об ошибке 20 – "В функциях FASIN и FACOS аргумент по модулю > 1". Функция FASIN получает вычисленное значение угла в радианах.

Формат:

```
FASIN (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FASIN (0.5) ввод
0.5236 *
```

Для вычисления арккосинуса предназначена функция FACOS. Если аргумент по модулю больше 1, выдаётся сообщение об ошибке 20. FACOS получает вычисленное значение угла в радианах.

Формат:

```
FACOS (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FACOS (0.5) ввод
1.0472 *
```

Для вычисления арктангенса предназначена функция FATAN. Угол, полученный при вычислении, выражен в радианах.

Формат:

```
FATAN (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FATAN (3.91) ввод
1.3204 *
```

Для вычисления натурального логарифма предназначена функция FLOG. Если аргумент меньше или равен нулю, выдаётся сообщение об ошибке 19 – "Логарифм нуля или отрицательного числа".

Формат:

```
FLOG (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FLOG (5.17) ввод
1.6429 *
```

Для вычисления десятичного логарифма предназначена функция FLOG10. Если аргумент меньше или равен нулю, выдаётся сообщение об ошибке 19.

Формат:

```
FLOG10 (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FLOG10(10) ввод
1.0000 *
```

Для вычисления экспоненты предназначена функция FEXP.

Формат:

```
FEXP (X)
```

где "X" – арифметическое выражение. Пример:

```
* T FEXP (5) ввод
148.4130 *
```

Для получения знаковой части числа, являющегося аргументом функции, предназначена функция FSGN. Если аргумент меньше нуля, то функция FSGN получает значение -1, если равен нулю, FSGN получает нулевое значение, если больше нуля, то функция получает значение +1.

Формат:

```
FSGN (X)
```

где "X" – арифметическое выражение. Примеры:

1)
```
T FSGN (6-4) ввод
1.0000 *
```
2)
```
T FSGN (0) ввод
0.0000 *
```

Для получения целой части выражения, являющегося аргументом функции, предназначена функция FITR.

Формат:

```
FITR(X)
```

где "X" – арифметическое выражение. Функция принимает значение целой части абсолютного значения аргумента со знаком аргумента. Примеры:

```
1)      T FITR (55.27) ввод
        55.0000 *
2)      T FITR (7.86) ввод
        7.0000 *
3)      T FITR (-4.1) ввод
        -4.0000 *
```

Для получения абсолютной величины выражений предназначена функция FABS.

Формат:

```
FABS (X)
```

где "X" – арифметическое выражение. Пример:

```
1)      T FABS (-17) ввод
        17.0000 *
2)      T FABS (30.05) ввод
        30.05 *
```

Для вычисления квадратного корня из арифметического выражения применяется функция FSQT. Если выражение имеет отрицательное значение, то выдаётся сообщение об ошибке 17 – "Корень квадратный из отрицательного числа".

Формат:

```
FSQT (X),
```

где "X" – арифметическое выражение. Пример:

```
X T FSQT (625) ввод
25.0000 *
```

14.3. Оператор X. Функция случайных чисел FRAN.

Прежде, чем начать дальнейшее описание встроенных функций, введём ещё один оператор "Фокала" – X (XECUTE).

Оператор "X" производит выполнение функций из библиотеки стандартных подпрограмм "Фокала" или арифметического выражения. При этом вывод на экран вычисленного значения не производится.

Оператор "X" может использоваться как в прямых, так и в косвенных командах в любом месте строки. Обычно оператор "X" используется для вычисления функций.

Примеры.

```
1.   * FCHR (65,66)
```

на экране появится символы AB.

```
2.    * FCHR(FCHR(-1))
```

эхо-печать – вводимый символ отображается на экране. Функция FCHR описана ниже.

Функция FRAN предназначена для генерации псевдослучайных чисел, распределённых равномерно в интервале (-1,1).

Формат:

```
FRAN ( ).
```

Пример:

```
FOR I= 1,100 ; SET R(I)=FRAN()
```

– программа генерирует сто случайных чисел.

```
* T FRAN() ввод
-0.3916 *
* FRAN() ввод
0.1659 *
```

При повторном применении FRAN "перетасовывает" числа и выдаёт, таким образом, непредсказуемую последовательность случайных чисел.

Если необходимо, чтобы последовательность случайных чисел была каждый раз одинаковой, в качестве аргумента функции берётся 1 – FRAN (1).

FRAN (1) полезно использовать при отладке программ, так как одна и та же последовательность случайных чисел производится каждый раз, когда программа повторяется.

14.4. Функция ввода-вывода символьной информации FCHR

Функция FCHR предназначена для приёма и (или) печати символов, соответствующих кодам КОИ-7.

Формат:

```
FCHR (список аргументов).
```

Основное назначение этой функции состоит в преобразовании символов, вводимых с клавиатуры, из кода КОИ-7 в десятичную форму, или, наоборот, из десятичной формы в коды КОИ-7 с выводом символа на экран.

Для преобразования символов, вводимых с клавиатуры, из кода КОИ-7 в десятичную форму аргумент функции FCHR должен быть отрицательным. Например, FCHR (-1). При выполнении такой функции "Фокал" будет ожидать ввода символа с клавиатуры ЭВМ. После нажатия нужной клавиши сама функция примет десятичное значение кода КОИ-7 введённого символа. Пример:

```
T FCHR (-1).
```

После нажатия клавиши "Ввод" машина ожидает ввода символа. Нажимаем, например, клавишу "A" (не печатается на экране), и на экране выдаётся её десятичное значение:

```
65.0000 *.
```

Таким образом, происходит перевод из кода КОИ-7 в десятичную форму.

Для преобразования из десятичной формы в код КОИ-7 с выводом символа на экран аргумент функции FCHR должен быть положительным. Целая часть величины аргумента, трактуемая как десятичное число, будет преобразована в соответствующий символ КОИ-7 и передана на экран. Значение функции FCHR при этом есть целая часть значения аргумента. Примеры:

```
1)      T FCHR(65.8) ВВОД
        A 65.0000 *
2)      FCHR(66) ВВОД
        B *
```

В случае, когда используется несколько аргументов, функция примет десятичное значение последнего введённого или выведенного символа, примеры:

```
3)      *T FCHR(65,66) ВВОД
        AB 66.0000*
4)      *T FCHR(65,66,-1) ВВОД
        AB 67.0000*
```

В примере 4 печатаются символы A и B, соответствующие восьмеричным кодам КОИ-7, равным 101 и 102, или десятичным числам 65 и 66. После этого с клавиатуры считывается следующий символ (пусть "C"), десятичное значение которого (67) и становится значением самой функции. Функция FCHR может использоваться рекурсивно, например,

```
* SET Z=FCHR(FCHR(-1)).
```

В процессе выполнения этой командной строки будет принят один символ с клавиатуры. FCHR(-1) принимает соответствующее десятичное значение (например, если вводимый символ "A", то FCHR(-1) принимает значение 65 в десятичном исчислении).

Далее, т.к. происходит выполнение FCHR(65), То будет напечатан символ "A" (для нашего примера).

Оператором SET будет присвоено переменной "Z" значение 65. Последнее объясняется тем, что оператор SET присваивает переменной значение арифметического выражения, стоящего в правой части.

Следует помнить, что при положительных аргументах код символа, выводимого на устройство вывода, вычисляется по модулю 256. Например:

```
* S X=FCHR(65); T !,X,! Ввод
A
65.0000*
```

Воспользуемся предыдущим примером и продемонстрируем, что в случае, когда аргументом функции FITR является число 321, на экране снова появится символ A, поскольку 321 по модулю 256 равно 65.

```
* S X=FCHR(321); T !,X,! Ввод
A
321.0000*
```

Функция FCHR может быть использована в обучающей программе для анализа ответов обучаемого. Например: ответ (A, B или C) на поставленный вопрос может анализироваться так:

```
1.01 S REQ=FCHR(-1)
1.03 I (REQ-65) 3.10,4.10,3.10
3.10 T "неверно, попробуйте ещё раз", !; G 1.01
4.10 T "верно",!
```

Ответ "A" является верным.

Для определения десятичных значений символов, расположенных на клавиатуре ЭВМ, можно воспользоваться следующей короткой программой:

```
* 1.1 T FCHR(-1);G Ввод
```

После её запуска на экран терминала будут выводиться десятичные значения символов, вводимых с клавиатуры ЭВМ. Следует быть внимательным при использовании функции FCHR с положительным аргументом.

Например, команда X FCHR (12) выполнит очистку экрана, т.к. код 12 есть десятичный код клавиши "СБР". Выполнение строки

```
1.1 FOR I=1,20; X FCHR(25)
```

вызовет сдвиг курсора на 20 позиций вправо, т.к. код 25 есть десятичный код клавиши "→".

Пример:

```
1.01 C заполнение экрана шахматной сеткой
1.02 C С помощью функции FCHR
1.1 X FCHR (12)
1.2 F K=1.3; D 2
1.3 Q
2.1 F I=1,40; D 3
```

```
2.2 F I=1,40;D 3.2; D 3.1
3.1 F J=1,40;X FCHR(127)
3.2 F J=1.40;X FCHR(32)
```

Если ваша машина имеет цветной выход, то функция FCHR поможет Вам работать с цветным изображением. Коды 245, 246, 247 и 248 переключают ЭВМ в режимы формирования красного, зелёного, синего и чёрного цветов соответственно (см. таблицу кодов в приложении 4).

14.5. Функция управления общей шиной (FX)

Функция FX используется для выполнения дополнительных действий с периферийными устройствами и для обращения к ячейкам памяти. Формат:

```
FX(код операции, адрес общей шины, данные).
```

Первый аргумент "код операции" может принимать значения -1, 0, +1 и определяет тип выполняемой операции:

+1 – чтение слова;

0 – чтение слова логическое с конъюнкцией;

-1 – запись слова по адресу общей шины.

Второй аргумент "адрес общей шины" должен быть или восьмеричным числом, или названием переменной и должен быть чётным. "Адрес общей шины" есть адрес ячейки памяти или регистра периферийного устройства, к которому производится обращение.

Третий аргумент "данные" должен быть десятичным числом в интервале от -32768 до +32767 или арифметическим выражением.

При операции чтения (1) выполняется чтение слова по "адресу общей шины". Третий аргумент не нужен. Функция принимает десятичное значение считанного слова. При операции запись (-1) выполняется запись слова (данных) по адресу общей шины.

"Фокал" обладает способностью самозащиты, т.е. запрещает запись в ячейки оперативной записи, используемые интерпретатором "Фокала" для своей работы (0-2000). При попытке записи в запрещённую область оперативной памяти выдаётся сообщение об ошибке 13 – "запрещённый адрес шины в функции FX". При операции "запись" функция FX принимает десятичное значение записываемого слова. При операции чтение слова с конъюнкцией (0) выполняется операция конъюнкции слова памяти и данных. Функция получает десятичное значение результата операции конъюнкции.

Если в функции FX указан несуществующий адрес на общей шине, то выдаётся сообщение об ошибке 21 – "несуществующее устройство". Примеры:

1) `T FX(1,1000) ввод`
   Данные считываются из ячейки с адресом 1000.
2) `*X FX (-1,7550,1) ввод`
   по адресу 7550 записывается 1.

3) `*T FX (0,4320,AH) ввод`
выполняется операция конъюнкции "И" над содержимым адреса 4320 и значением переменной АН и выводится на экран.

В нижеприведённой таблице 1 содержится справочная информация для работы с функцией FX. В левой графе таблицы записаны десятичные значения чисел для обращения к битам слова. В средней графе – восьмеричные значения чисел. В правой – номера битов слова, соответствующие числам слева.

ТАБЛИЦА 1

Десятичное число	Восьмеричное число	Номера битов в слове!
-32768	100000	15
16384	40000	14
8192	20000	13
4096	10000	12
2048	4000	11
1024	2000	10
512	1000	9
256	400	8
128	200	7
64	100	6
32	40	5
16	20	4
8	10	3
4	4	2
2	2	1
1	1	0
-32640	100200	15 и 7
192	300	7 и 6
255	377	7,6,5,4,3,2,1,0
-256	177400	15,14,13,12,11,10,9,8

14.6. Функция, программируемая пользователем (FSBR).

Программируемая пользователем функция FSBR используется для выполнения нумерованной группы строк (или одной строки) как подпрограммы.

Подпрограмма представляет собой функцию одной специальной переменной, обозначаемой "&". Формат обращения к функции FSBR:

```
S V=FSBR (N,ARG)
```

или

```
X FSBR (N,ARG)
```

где V - название переменной, которой присваивается вычисленное значение функции;
N - номер группы, реализующей алгоритм программируемой функции (число или переменная);
ARG - аргумент функции FSBR (число, переменная или арифметическое выражение).

Интерпретатор "Фокала", встретив в тексте программы обращение к FSBR, вычисляет значение аргумента функции и присваивает полученное значение специальной переменной "&". Затем осуществляется передача управления на первую строку программы с номером N. Возврат из подпрограммы, реализующей алгоритм программируемой функции, осуществляется по команде RETURN, или по исчерпанию строк в группе. После завершения выполнения подпрограммы последнее вычисленное значение и становится значением функции.

Пример вычисления экспоненты:

```
1.10 A "Введите значение аргумента", ARG; C EXP
1.20 S Y=FSBR(7,ARG)
1.30.T !, Y; QUIT
7.10 I (&^2-0.01) 7.2;S &=&/2; D 7; S & = &^2; R
7.20 S &=l+&+&^2/2+&^3/6+&^4/24+&^5/120+& ^6/720
```

В группе строк (подпрограмме) под номером 7 используется специальная переменная "&", которой автоматически присваивается начальное значение, равное ARG. Последнее значение становится окончательным значением функции FSBR.

Функция FSBR может использоваться рекурсивно.

Если строк группы недостаточно для записи подпрограммы, то можно использовать строки других групп, передавая им управление оператором DO. Это объясняется тем, что FSBR, передавая управление группе, работает как оператор DO и, исчерпав данную группу, возвращает управление команде, следующей за вызовом FSBR. Если алгоритм, реализующий программируемую функцию, имеет несколько точек выхода, то командная строка, содержащая точку выхода, должна оканчиваться командой "R".

Пример вычисления факториала:

```
1.10 A N
1.20 T !,FSBR(5,N)
1.30 Q
5.10 I(1-&) 5.2; R
5.20 S &=&*FSBR(5,&-1)
```

14.7. Функция управления положением курсора (FK).

Функция FK предназначена для установки курсора по координатам X и Y плоскости экрана. Формат функции:

```
FK (X,Y)
```

где X - десятичное значение координаты X, X < 64;
Y - десятичное значение координаты Y, Y < 24.

X и Y могут быть любым арифметическим выражением. Изменение положения курсора на экране функцией FK осуществляется относительно верхнего левого угла экрана. Функция FK принимает десятичное значение координаты Y.

Пример:

```
* X FK(32,12)
```

Выполнение данной команды поместит курсор в центр экрана, т.е. сместит курсор на 32 позиции вправо по оси X и на 12 позиций вниз по оси Y относительно верхнего левого угла экрана.

Выполнение команды FK(-32,-12) также поместит курсор в центр экрана, но он будет смещаться относительно нижнего правого угла на 32 позиции влево по оси X и на 12 позиций вверх по оси Y.

Отметим, что экран имеет следующие размеры: 24 позиции по вертикали и 64 – по горизонтали (в режиме "32 символа в строке" экран имеет 32 позиции по горизонтали).

14.8. Функции для работы с графической информацией FT и FV.

Для формирования точки на экране используется функция FT.

Формат:

```
FT(КОП,X,Y)
```

где КОП = 0 - стирание точки на экране
1 - запись точки на экране

X < 512, Y < 256; X и Y – десятичные значения координат, по которым формируется точка на экране. X и Y могут быть любым арифметическим выражением. Функция FT принимает десятичное значение координаты Y. Координаты X и Y отсчитываются от верхнего левого угла. Пример: построить синусоиду

```
1.1 F I=0,360; X FT(1,1,100-50*FSIN(1*3.14159/180))
```

Для формирования вектора на экране терминала используется функция FV. Формат:

```
FV(КОП,X,Y)
```

где КОП = 0 - стирание вектора
1 - запись вектора

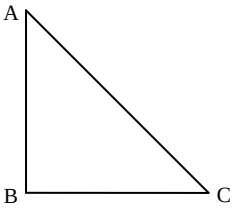
X и Y – десятичные значения координат конца вектора. X и Y могут быть любым арифметическим выражением. Функция FV принимает десятичное значение координаты Y. Перед выполнением функции FV должны быть определены координаты начала вектора. Это можно сделать, например, функцией FT. Примеры:

```
1)      X FT(0,0,0); X FV (1,400,200)
```

Будет построен вектор с координатами начала вектора (0,0) и конца вектора (400,200).

```
2)      X FT (0,100,100)
        1.2 X FV (1,100,200)
        1.3 X FV (1,200,200)
        1.4 X FV (1,100,100)
```

Данная программа построит прямоугольный треугольник ABC



Строка 1.1 устанавливает координаты точки A.

Строка 1.2 строит катет AB, строка 1.3 строит катет BC, а строка 1.4 строит гипотенузу CA

14.9. Функция работы с портом ввода/вывода FP.

Порт – это техническое устройство ЭВМ, предназначенное для управления какими-либо приборами, оборудованием, подключёнными к ЭВМ через порт. Порт имеет разъём, находящийся на задней стенке корпуса справа.

С контактами разъёма "УП" (кроме некоторых) соединены выводы двух регистров порта:

- входного 16-ти разрядного регистра;
- выходного 16-ти разрядного регистра.

Регистр порта – это устройство ЭВМ, позволяющее, как и память, записывать и хранить в нём информацию. Разряд (БИТ) регистра – это элемент этого устройства, который может принимать значения нуля или единицы. Комбинация нулей и единиц в разрядах регистра порта представляет из себя

двоичное число. Все разряды в регистрах пронумерованы от 15 до 0. Схематично регистр можно представить так:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

В квадратиках показаны номера разрядов. 15-тый разряд называют старшим, нулевой – младшим, так как в них хранятся старший и младший разряды двоичного числа соответственно. В дальнейшем будем называть двоичные 16-ти разрядные числа, словами.

Информацию во входном регистре ЭВМ можно только прочитать, (как говорят, этот регистр доступен только по чтению).

Информацию в выходной регистр можно только записать и нельзя прочитать (регистр доступен только по записи).

Для работы с портом необходимо знать, каким контактам выходного разъёма соответствуют разряды каждого из двух регистров. Такое соответствие представлено в таблице 2.

Таблица 2. Соответствие разрядов входного (выходного) регистров контактам разъёма порта.

Номера разрядов регистров порта		Контакт разъёма порта
Входной	Выходной	
	0	A16
	1	A13
	2	B12
	3	B10
	4	B5
	5	B7
	6	B6
	7	A7
	8	A28
	9	B28
	10	A27
	11	B27
	12	A26
	13	B26
	14	A25
	15	B25
0		B24
1		A24
2		B23
3		B17
4		B20
5		A20
6		B22
7		A23

8		B31
9		A31
10		B32
11		A32
12		B30
13		A29
14		B29
15		A30

Контакты разъёма "УП" A11, B11, A18, B18, A19, B19 общие. Остальные контакты использовать запрещается. Подключаемые к порту устройства должны иметь входы-выходы, управляемые сигналами уровня ТТЛ.

```
   0 В < U(1) <= +0,5 В.
5,25 В > U(0) >= +2,4 В.
```

Необходимая последовательность обмена сигналами между устройством пользователя, подключённым к порту, и ЭВМ задаётся с помощью программы, разрабатываемой пользователем. Для написания таких программ на языке "Фокал" служит встроенная функция FP. Формат:

```
FP(КОП,МАСКА)
```

где КОП - код операции, может принимать значения 0, 1, 2, 3.
0 – чтение регистра ввода по маске;
1 – очистка битов регистра вывода по маске;
2 – установка битов регистра вывода по маске;
3 – чтение регистра вывода по маске.

МАСКА - восьмеричное число в диапазоне от 0 до 177777 или переменная с десятичным значением в диапазоне от -32758 до +32767.

При операциях чтения с регистра ввода (КОП=0) и регистра вывода (КОП=3) выполняется побитная поразрядная конъюнкция (операция логическое "И") между считываемыми из регистра данными и маской. Если маска равна 177777, то читаются все 16 битов слова. Если маска равна 377, то читается младший байт слова в регистре, если 177400, то читается старший байт.

Если КОП=1, то выполняется очистка (установка равными нулю) битов регистра вывода по маске. Очистка битов регистра вывода происходит в соответствии с теми битами маски, которые установлены в единицу. Для очистки всего слова регистра маска должна быть равна 177777, для очистки младшего байта – 377, для старшего байта – 177400.

Если КОП=2, то выполняется установка равными единице битов регистра вывода по маске. Установка битов регистра вывода происходит в

соответствии с теми битами маски, которые установлены в единицу. Для установки всех битов регистра вывода маска должна быть равна 177777, для установки в единицу битов младшего байта – 377, для старшего байта – 177400.

После выполнения функция FP принимает десятичное значение считанного или записанного слова. Пример:

```
*1.1 X FP(2,177777); T FP(3,10) ввод
8.0000*
```

Первый оператор строки 1.1 выполнит установку всех единиц в регистре вывода порта. Второй оператор читает бит [3] в регистре вывода порта и т.к. он был установлен в единицу, то функция принимает значение 8.

15. Операторы KILL, VACANT, PASS

15.1. Оператор сброса внешних устройств KILL.

Оператор KILL предназначен для установки всех внешних устройств в исходное состояние. При работе оператора KILL выполняется команда ассемблера RESET.

Оператор KILL не требует операндов. При его выполнении прекращается работа всех внешних устройств, работавших в режиме прерывания. Например, после выполнения оператора KILL в программном режиме запрещается прерывание от клавиатуры, и прервать выполнение программы становится невозможным до её окончания.

Оператор KILL может использоваться как в прямых, так и в косвенных командах в любом месте строки.

15.2. Оператор VACANT

Оператор VACANT предназначен для определения оперативной памяти, не занятой текстом программы на "Фокале".

Оператор VACANT не требует операндов. Выполнение оператора приводит к выдаче на экран телевизора следующей информации:

```
"СВОБОДНО XXXXXX Б.ОЗУ"
```

где XXXXXX есть восьмеричное число, определяющее количество оставшейся свободной памяти для текста программы, в байтах.

ПРЕДУПРЕЖДЕНИЕ!!!

Оператор VACANT справедлив только для микро-ЭВМ "Электроника БК 0010".

15.3. Операторы группы PASS

Операторы группы PASS предназначены для передачи управления системным программам микро-ЭВМ: системному монитору и тестовому монитору.

Формат:

```
PASS <ПОДОПЕРАТОР>
```

Подоператор определяет системную программу, которой требуется передать управление. Оператор PASS включает 2 подоператора: MONITOR и TEST.

Оператор PASS MONITOR передаёт управление системному монитору компьютера.

Оператор PASS TEST передаёт управление тестовому монитору.

Для возвращения в "Фокал" из системного монитора необходимо набрать F и <ВВОД>, из тестового монитора – К.

При возвращении в "Фокал" текст программы и переменные в памяти стираются.

16. Как решать задачу на ЭВМ?

В предыдущих главах Вы уже решали много задач на ЭВМ, и у Вас могло сложиться впечатление, что вопрос, которым начинается эта глава, излишен – напишите программу решения задачи и никаких проблем.

Действительно, это так, но написать программу, ведающую правильный ответ задачи, без глубокого предварительного анализа пути её решения на ЭВМ, для многих задач чрезвычайно трудно, а для некоторых – невозможно. Следует отметить, что на ЭВМ можно решать далеко не все задачи, а только те, для которых может быть разработан алгоритм решения.

Рассмотрим пример и на нём продемонстрируем этапы решения задачи на ЭВМ.

Пример 1.

Разработать программу, вычисляющую функцию SIN (X), не используя встроенной функции.

В настоящем примере Вы увидите сам процесс вычисления функции SIN (X), который при использовании встроенной функции остался для Вас, как бы "за кадром", что, разумеется, не уменьшает удобств использования встроенных функций.

16.1. Этап 1. Разработка алгоритма решения.

Алгоритм – это точно определённое правило действий, для которого задано указание, как и в какой последовательности это правило необходимо применять к исходным данные задачи, чтобы получить решение.

Перед составлением алгоритма необходимо провести анализ задачи, в результате которого должен быть выбран метод её решения. Если такой метод ещё не найден, бесполезно приступать к составлению алгоритма.

Для решения нашей задачи применим метод разложения функции SIN (X) в ряд Маклорена, который имеет вид:

SIN(X)=X-X^3/3!+X^5/5!-X^7/7!+...+(-1)^(N-1)*X^(2N-l)/(2N-l)

полученный ряд имеет бесконечное число членов. Такое положение нас не устраивает, поскольку даже самая быстродействующая ЭВМ никогда не вычислит сумму бесконечного числа слагаемых. Нам необходимо получить решение за конечное число шагов.

С этой целью при расчёте принимают во внимание не все члены ряда, а только несколько его первых членов. Как говорят, ряд усекается, при этом в результате мы получим не точное значение нашей функции, а приближенное, заведомо допуская ошибку, равную сумме отсечённых членов ряда.

Очень важно бывает выбрать эту ошибку (а по ней – число членов ряда, принимаемых в расчёт) таким образом, чтобы ошибка была приемлемой.

Выберем для нашей задачи ошибку 0.00001. Объясним, почему мы выбрали ошибку именно такой.

Дело в том, что "Фокал" оперирует с числами, в которых запоминается только 6 значащих цифр, и при арифметических операциях с числами мы не сможем гарантировать большей точности. Требовать от программы точности большей, чем мы задали, не имеет смысла, т.к. это только увеличит время расчёта и не уменьшит ошибки.

По заданной ошибке можно вычислить, сколько надо взять членов ряда, чтобы обеспечить заданную точность вычислений.

Для ряда Маклорена, представляющего функцию SIN(X), известно, что ошибка, вызванная усечением, не превышает по абсолютной величине последнего по порядку члена усечённого ряда.

Как только абсолютная величина очередного вычислительного члена ряда станет меньше ошибки, установленная точность будет достигнута.

16.1.1. Теперь приступим к составлению алгоритма решения задачи.

Алгоритм можно описывать разными способами. Существуют даже специализированные алгоритмические языки для описания алгоритмов, например, Алгол-60. Программа, написанная на "Фокале", даёт вполне ясное, исключающее двоякое толкование представление об алгоритме решения задачи. Если Вам понятен путь решения, нет необходимости описывать

алгоритм решения задачи на каком-то алгоритмическом языке, достаточно написать программу на "Фокале".

Если Вы ещё не имеете достаточного опыта, позволяющего Вам выразить алгоритм сразу в виде программы на "Фокале", мы рекомендуем Вам записать его в понятной Вам "словесной" форме. Для нашей задачи алгоритм может быть представлен в виде словесного описания, описания, как показано ниже:

16.1.2. Словесное описание алгоритма вычисления функции SIN (X).

БЛОК 1. Головная (вызывающая) программа.

БЛ.1.1. Ввести аргумент X (в градусах) с клавиатуры и перевести X из градусной меры в радианную:

```
X=3.14159*X/180
```

БЛ.1.2. Обратиться к подпрограмме вычисления SIN(X)

БЛ.1.3. Напечатать результат, выйти из программы.

БЛОК 2. Подпрограмма вычисления SIN(X).

БЛ.2.1. Подготовить переменные, в которых будут накапливаться суммы (обнулить их).

```
SIN=0
N=0
```

БЛ.2.2. Установить текущий номер члена ряда.

```
N = N +1
```

БЛ.2.3. Вычислить факториал.

```
MF=(2N-1)!
```

БЛ.2.4. Вычислить очередной член ряда, стоящий на N-ом месте.

```
RN=(-1)^(N+1)*X^(2N-1)/(2N-1)!,
```

или

```
RN=(-1)^(N+1)*X^(2N-1)/MF
```

БЛ.2.5. Сложить очередной член ряда с суммой ряда

```
SIN=SIN+RN
```

БЛ.2.6. Если ещё не достигнута точность (0.00001), то есть ещё выполняется условие

```
(0.00001-!RN!) <0,
```

перейти к блоку 2.2.

БЛ.2.7. Выйти из подпрограммы.

Предложенный "словесный" вариант далеко не единственный, Вы можете списывать алгоритм любым доступным Вам способом, важно только, чтобы он был для Вас понятен, а последовательность действий, которую он определяет, не допускала двояких толкований и за конечное число шагов приводила к решению задачи.

Детализация действий при описании алгоритма зависит от конкретной задачи, опыта программиста, языка программирования, системы машинных команд ЭВМ и может быть в разных случаях различной.

Большая степень детализации алгоритма, в принципе, навредить не может, но требует больших затрат времени на разработку и описание алгоритма. В дальнейшем приобретённый Вами опыт подскажет, какая степень детализации Вам необходима для составления алгоритма. А для начала старайтесь составлять подробные алгоритмы.

Воспользуемся описанием алгоритма и напишем программу:

```
1.1 A !,"X=",X;S X=3.1415926*X/180; C ВВОД X
1.2 D 2; C обращение к вычислению SIN(X)
1.3 T !,%7.06, SIN; R; C печать результата
2.1 S SIN=0; S N=0; C подготовка накопителей
2.2 S N=N+1; C установка номера члена
2.3 S MF=1; F I=1,2*N-1; S MF=MF*1; C факториал
2.4 S RN=(-1)^(N+1)*X^(2*N-1)/MF; C очередной член
2.5 S SIN=SIN+RN; C накопление синуса
2.6 I (0.00001-FABS(RN)) 2.2; C анализ ошибки
2.7 R; C выход из подпрограммы
```

Номера строк в программе соответствуют номерам блоков в словесном описании алгоритма.

16.2. Этап 2. Отладка программы.

Отладка – это процесс поиска и устранения ошибок в программе. Мы сейчас будем говорить только о логических ошибках. Считается, что, пользуясь диагностикой "Фокала", Вы уже умеете устранять синтаксические ошибки в программе.

Перед отладкой текст программы должен быть занесён в память ЭВМ. Отладка программы – это процесс глубоко индивидуальный, его сложно описать каким-либо алгоритмом. В рамках нестоящего руководства мы сможем лишь привести следующие отдельные рекомендации.

16.2.1. Перед отладкой программы или блока программы подготовьте отладочные данные.

Отладочные данные – это набор исходных данных, для которых заранее известно правильное решение (ответ) задачи.

Например, для нашей задачи известно, что SIN(30)=0,5. Если мы зададим перед входом в нашу программу значение переменной X, равное 30,

то после её правильного выполнения значение переменной SIN должно быть равным 0.500000 (с учётом допустимой погрешности – 0.000001). Если такой результат не получен, следует искать ошибку.

16.2.2. Не старайтесь отладить целиком всю программу, отлаживайте её по блокам и подпрограммам.

16.2.3. При отладке старайтесь получить как можно больше информации об изменениях переменных программы. Это достигается вставкой в определённые вами мест программы дополнительных операторов.

Обычно это – оператор печати, позволяющий выдавать на экран значения переменных на каком-либо шаге выполнения программы. По динамике изменения переменных Вы сможете определить место ошибки.

Отладочные операторы убирают из текста программы после её отладки.

16.2.4. При отладке пользуйтесь трассировкой, т.е. таким режимом работы программы или её отдельный частей, при котором виден путь, по которому происходит передача управления в Вашей программе.

Чтобы выполнить трассировку, нужно вставить знак трассировки – "?" – в любое место строки (кроме текста, заключённого в кавычки).

"Фокал" печатает каждый следующий за знаком трассировки символ до тех пор, пока не встретится следующий знак "?", или управление не будет передано интерпретатору.

16.2.5. Внимательнее следите за именами переменных, а также за их использованием.

Часто одной и той же переменной присваиваются разные имена, а разным переменным – одно и то же имя. Например, Вы где-то в программе вставили отладочный оператор T$, который напечатал перечень и значения всех переменных Вашей программы, и при этом Вы обнаружили наличие имён каких-либо переменных, которые Вы не использовали. Это говорит о том, что Вы эти переменные ввели где-то ошибочно.

16.2.6. Анализируйте условия, при выполнении которых передаётся управление. Часто эти условия неверно сформулированы или неверно записаны в программе и являются источником ошибки.

16.2.7. Помните об особенностях работы операторов перехода в группах, работающих под управлением оператора "D".

16.2.8. Внимательно следите за параметрами циклов, условиями окончания циклов и областью их действия.

Занесите написанную Вами программу вычисления синуса в память ЭВМ. Данная программа, как и все остальные программы, которые приведены в настоящем руководстве, предварительно отлажена. Если Вы её занесли без ошибок, программа будет сразу работать. Добейтесь того, чтобы программа работала и задавала верные результаты. Только после этого приступайте к дальнейшему чтению изложения.

Предположим, что Вы всё-таки ошиблись. Например, в операторе командной строки 2.50 вместо

```
2.50 S SIN=SIN+RN
```

написали

```
2.50 S STN= SIN +R
```

(вместо SIN – STN)

Такую ошибку диагностика интерпретатора обнаружить не может, т.к. ошибочная для нашей программы переменная STN написана с соблюдением всех правил записи переменных на языке "Фокал".

Занесите эту ошибку в программу и посмотрите, как программа будет работать. Теперь вместо верного результата SIN(30)=0.5 она выдаёт результат 0.000. Убедитесь в этом.

Как мы стали бы искать ошибку, если бы не знали о ней заранее? Для этого введём в программу оператор:

```
2.52 T !,$
```

который нам напечатает перечень всех переменных программы и их значения. Выполните, задав аргументу значение 30, программу.

На экране несколько раз промелькнут и остановятся столбцы с информацией. Почему столбцов несколько?

Потому, что вставленный нами отладочный оператор находится в области действия цикла, вычисляющего значение синуса.

Поскольку вывод столбцов прекратился, мы можем сделать вывод, что этот цикл закончился.

Приглядимся внимательнее, что за информация расположена в последнем выведенном столбце:

```
S MF()=5040.000
S I()=3.000000
s SI()=0.000000
S N()=4.000000
S RN()=0.000002
S ST()=0.000002
S X()=0.523590
```

Рассмотрим подробно первую строку. Слева напечатана буква S – это признак того, что информация выводится по элементу списка оператора T – $. Далее стоит имя MF() со скобками, в которых ничего нет. Имя – это имя переменной MF, а за знаком равенства напечатано её значение при последнем проходе цикла. Если в скобках имеется только пробел, это означает, что это скалярная (простая) переменная.

Если в скобках расположены какие-либо числа, то это – индексы переменной, а сама переменная – индексная.

В нашем примере все переменные столбца – скалярные.

Имена переменных в столбце содержат не более двух символов.

В этом отражён тот факт, что интерпретатор "Фокала" различает имена только по двум первым символам, игнорируя все остальные.

Рассмотрим внимательнее имена переменных с целью ответа на вопрос: все ли нужные нам имена включены в список и нет ли каких-либо "лишних" имён;

Проведём краткий анализ имён переменных и их значений.

Согласно алгоритму, MF=(2N-1)! проверим, так ли это на самом деле. Найдём в столбце значение N, равное 4, и вычислим значение факториала

(2x4-1) ! = 7*6*5*4*3*2*1 = 5040.00.

В нашем столбце значение MF – такое же. Есть все основания полагать, что факториал вычисляется верно.

Поскольку мы пришли к такому выводу, нет смысла подробно рассматривать переменную I, с помощью которой вычислялся факториал.

Следующая переменная – N. Её значение равно 4, т.е. для расчёта SIN(30) с заданной нами точностью 0.000001, достаточно было 4-х членов ряда. Это вполне допустимый результат, и нам нет оснований сомневаться в том, что ЭВМ правильно вычислила N.

Следующая переменная в столбце – SIN, SI=0, но это неверно, т.к. SIN(30) отличен от 0. Есть, по крайней мере, два варианта хода событий, в результате которых переменная SIN оказалась равной 0:

- либо к ней ничего не прибавлялось в ходе расчётов, и она оказалась равной своему начальному значению;
- либо мы нечаянно "обнулили" её где-то в ходе вычислений.

Пока оставим выяснение этого вопроса и перейдём к рассмотрению. переменной RN. RN – это значение последнего вычисленного нами члена ряда. Как только оно станет меньше, чем 0.00001, мы должны прекратить добавление к SIN последующих членов. Значение RN равно 0.000002. Это – вполне возможное значение. У нас нет основания сомневаться в том, что RN считается верно.

ST – это начальные буквы какой-то переменной, которую мы по алгоритму не должны были использовать. Есть надежда, что поиск этой переменной приведёт к нахождению ошибки. Очень подозрительно то, что значение этой переменной равно значению переменной RN. Видимо, где-то вместо SIN мы суммируем RN в ST.

Последним в столбце расположена переменная X. Её значение равно 3,14...*30/180=0.52, что соответствует значению X. Таким образом, X вычисляется верно.

Выведем на экран текст программы и внимательно рассмотрим ту её часть, где формируется сумма SIN = SIN + RN. В операторе 2.50 мы сразу обнаружим ошибку: вместо SIN стоит STN. Исправьте эту ошибку.

В завершении главы следует сказать, что даже самая тщательная отладка программы во многих случаях не может гарантировать того, что не встретятся такие сочетания данных, для которых программа будет выдавать неверный результат.

К примеру, выполним ещё раз нашу программу вычисления синуса и вычислим SIN (900). Вычислили?

Нет, верного ответа не получили, поскольку программа не выполнилась до конца, – возникло переполнение. Этот пример показывает, что надо очень осторожно подходить к проверке правильности работы программы. На данные для нашей программы необходимо наложить ограничения, например, вычислить SIN(X) только для тех X, которые удовлетворяют условию |X|<=360. Одним из способов проверки правильности работы программы является проверка её работы на границах допустимых значений входных данных.

При наших ограничениях следовало бы ещё вычислить SIN(0), SIN(-360), SIN(360) и, если их значения находятся в пределах заданной точности, то можно считать, что наша программа верно работает и для других значений, не выходящих за допустимые границы.

16.3. Итоги главы 16.

16.3.1. Для разработки программы, реализующей какую-либо задачу, необходим предварительный анализ, в результате которого должен быть выбран метод её решения и составлен алгоритм, реализующий этот метод.

16.3.2. Алгоритм – это точно определённое правило действий, для которого точно задано указание, как и в какой последовательности это правило необходимо применять к исходным данным, чтобы получить её решение. Алгоритм можно выражать в виде блок – схемы, словесного описания, в виде программы и т.д.

16.3.3. На примере вычислений функции SIN (X) путём разложения её в ряд Маклорена разобран пример отладки, и приведены рекомендации по отладке.

Отладка – это процесс поиска и устранения ошибок в программе.

16.4. Вопросы к главе 16

16.4.1. Что такое алгоритм решения задачи?

16.4.2. Какими способами можно записывать алгоритм?

16.4.3. Что такое отладка программы?

16.4.4. Какие приёмы используются при отладке?

17. Ответы на вопросы

К ГЛАВЕ 2.

2.10.1. Синий.

2.10.2. Настройка ЭВМ на ввод соответствующей группы символов, расположенных на левом, правом, нижнем, русском, латинском, заглавном, строчном регистрах.

2.10.3. Клавиши следует нажимать и удерживать.

2.10.4. Зелёный.

2.10.5. Ввод цифр, букв, специальных знаков.

2.10.6. Латинским, русским, строчным, заглавным.

2.10.8. А) "ЛАТ"

Б) "СТР", "РУС"

В) "СТР", "ЛАТ"

Г) "ЗАГЛ", "РУС"

2.10.9. А, π, D, Г.

К ГЛАВЕ 4.

4.9.1. Перед косвенной командой стоит номер командной строки, перед командой же номер строки отсутствует.

4.9.2. Либо командами Передачи управления, либо естественным образом.

4.9.3. Нажатием клавиши "Ввод".

4.9.4. Печатает символы AAA.

4.9.5. 1,2 – косвенные команды; 3 – команда.

4.9.6. `20.1 T "Иванов Иван Иванович".`

К ГЛАВЕ 5.

5.6.1. 1,3,5,2,4,6,7.

5.6.2. 15.

5.6.3. 5.

К ГЛАВЕ 6.

6.6.1. Нельзя.

6.6.2. Строка, которая ещё не введена (не нажата клавиша "Ввод") или вызвана на редактирование оператором "М".

6.6.3. Нажатием клавиши "Ввод".

6.6.4. С помощью клавиши "СМЫКАНИЕ".

6.6.5. Можно.

К ГЛАВЕ 7

7.6.1. Нажмите клавиши "W", "Ввод".

7.6.2. W 10 ввод

7.6.3. E A ввод

7.6.4. Можно.

7.6.5. Десятичные, в экспоненциальной форме.

7.6.6.
```
10.1 A, X
10.2 T !,X,X^2,X^3,X^4
```

7.6.7.
```
10.3 G 10.1
```

К ГЛАВЕ 8.

8.7.1. Элемент списка выводимых данных, начинающийся с символа %, указывающий, в какой форме необходимо вывести число.

8.7.2. `%W.0N`.

8.7.3. `%8.04`.

8.7.4. Текст, числа, переменные, !,$,%W.0N.

8.7.5. Текст 1
Текст 2
Текст 3

8.7.6. `T !`

8.7.7. Число с округлением 1010.40
Целая часть 0
Результат вычисления 2.50

8.7.8. `101.0 101.0`

К ГЛАВЕ 9.

9.8.1. Правильно

9.8.2. 2SA, шар, FXX, SS-4.

9.8.3. SY36, SY37, SY38.

9.8.4. Аргумент не заключён в скобки.

9.8.5. А) 2.0000;
Б) 1.0000;
В) 2.2500;
Г) 2.2500.

9.8.6. Результат одинаков.

К ГЛАВЕ 10.

10.9.1. Совокупность всех возможных в ЭВМ адресов байтов.

10.9.2. Поскольку она может иметь значение, отличное от нуля, при суммировании это значение исказит результат.

10.9.3. А) 88.20; Б) 88.20

10.9.4. Совокупность операторов, которые выполняются при работе цикла.

10.9.5. Параметры цикла должны вполне определённым образом изменяться в ходе его выполнения. Если же мы будем им присваивать при каждом проходе цикла одни и те же значения, они практически не будут изменяться, а поэтому никогда не выполнится условие окончания цикла.

10.9.6.
```
10.1 S SM=0
10.2 F K=0,5,N; S SM=SM+K
```

10.9.7.
```
10.1 S SM=0
10.2 F K=0,5,N; S SM=SM+K^3
```

К ГЛАВЕ 11.

11.10.1. Смотри п. 11.9.1.

11.10.2. Смотри приложение 3.

11.10.3. Для выделения общих мест программы, а также для её структурирования, с целью сделать более чётким и понятным алгоритм программы.

11.10.4. Передав управление оператору R.

11.10.5. Управление передаётся только на один оператор, после чего возвращается на оператор, следующий в группе за 1 или 6

11.10.6. X=5.0000

11.10.7. X=100.0000

К ГЛАВЕ 12.

12.7.1. См. п. 12.6.

12.7.2. См. п. 12.6.

12.7.3. Операторы, следующие за оператором F в той же самой командной строке.

Если в области действия оператора F находится оператор D, то все операторы, вызываемые им на выполнение, так же включаются в область действия оператора F.

12.7.4. SIN0=0

К ГЛАВЕ 13.

13.12.1. См. п. 13.11.1.

13.12.2. См. п. 13.11.1.

13.12.3. См. п. 13.6.

13.12.4. См. п. 13.7.

13.12.5 12.12.7. См. п. 13.11.1.

К ГЛАВЕ 16.

16.4.1. См. п. 16.3.2.

16.4.2. См. п. 16.3.2.

16.4.3. См. п. 16.3.3.

Приложение 1

ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ

Код ошибки	Текстовое сообщение о причине ошибки	Рекомендуемые действия
00	Готовность к работе	Система ожидает любых действий пользователя
01	Неправильный номер строки	Проверьте метку строки
02	Неправильное имя функции или переменной	Проверьте имя функции или переменной (должны быть только латинские буквы, а переменная не должна начинаться с буквы F)
03	Непарные скобки	Проверьте соответствие левых и правых скобок (каждой открывающей скобке должна соответствовать закрывающая скобка того же типа)
04	Неправильная команда	Проверьте имя и формат команды (должны быть использованы только латинские буквы)
05	Несуществующий номер строки	Проверьте наличие строки с указанным в операторе номером
06	Несуществующий номер группы или номер строки в операторе DO	Проверьте наличие строки и группы строк, указанных в операторе
07	Неправильный формат SET или FOR	Проверьте формат операторов SET или FOR
08	Двойной или отсутствующий оператор в выражении	Проверьте арифметическое выражение, соответствие открывающих и закрывающих скобок
09	Переполнение стека	Проверьте логику программы, число рекурсий
10	Переполнение памяти текстом программ	Сократите объём Вашей программы
11	Нет места для переменных	Сократите количество переменных и констант Вашей программы
12	Порядок больше 38	Проверьте значение переменных и констант Вашей программы
13	Запрещённый адрес шины в функции FX	Проверьте аргумент функции, замените адрес запрещённой ячейки памяти

Код ошибки	Текстовое сообщение о причине ошибки	Рекомендуемые действия
14	Попытка деления на нуль	Проверьте значение делителей в выражении
15	Попытка возведения в отрицательную или слишком большую степень	Проверьте значения показателей степени в выражении
16	Слишком много символов во входных данных	Сократите количество символов, вводимых в ответ на запрос оператора ASK, до 23-х
17	Корень квадратный из отрицательного числа	Проверьте значение аргумента функции FSQT
18	Переполнение входного буфера	При вводе строки быть набрано более 80-ти символов
19	Логарифм нуля или отрицательного числа	Проверьте значение аргумента функций FLOG или FLOG10
20	В функциях FASIN или FACOS аргумент по модулю > 1	Проверьте значение аргумента функции FASIN или FACOS
21	Ошибка в имени файла	Проверьте имя файла в операторе LIBRARY
22	Ошибка контрольной суммы	Повторите чтение или запись файла
23	Ошибка по длине файла	Повторите чтение или запись файла
24	Останов операций магнитофона по прерыванию оператора	Вами была нажата клавиша на клавиатуре ЭВМ во время работы оператора группы LIBRARY
25	Останов по клавише СТОП	Вами была нажата клавиш СТОП
26	Несуществующее устройство	Произошло обращение по несуществующему адресу общей шины. Измените адрес в программе
27	Неправильный код операции в функции FP	Проверить и изменить код операции в функции FP

Приложение 2

Краткие сведение о языке "Фокал"

1. Алфавит

В качестве алфавита языка. "Фокал" используются буквы, цифры, ограничители.

Буквы. Латинский алфавит:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

"Фокал" позволяет комментарии и текст примечаний записывать русскими заглавными и строчными буквами.

Цифры. Только арабские:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Цифры, служат для образования чисел и номеров строк.

2. Ограничители.

В качестве ограничителей используются:

Знаки арифметических операций:

^ - возведение в степень;

* - умножение;

/ - деление;

+ - сложение;

- - вычитание;

А) Знаки, используемые как разделители:

= - знак присваивания

" " - пробел.

Ввод - ввод информации в ЭВМ,

. – точка,

, - запятая,

; - точка с запятой,

: - двоеточие;

Б)

[]
< > - скобки для записи выражений;
( )

В) Знаки, используемые как управляющие:

" - кавычки для печати текстовой информации,

% - процент для задания формата вывода чисел,

? - знак трассировки,

! - переход в начало новой строки,

$ - вывод на печать таблицы значений переменных;

Г)

→ - сдвиг курсора вправо,

← - сдвиг курсора влево,

↦ - размыкание,

↤ – смыкание,

↤- - удаление последнего введённого символа,

СБР↦ - удаление строки с позиции курсора,

СБР - сброс (очистка) экрана,

ВС - возврат в начало строки,

ГТ - горизонтальная табуляция,

ЗБ - сброс набранной строки.

Д) Знаки, управляющие действиями "Фокала":

КТ (пустая синяя клавиша) - сброс выполнения строки, удаление всех переменных;

НР/СБР - вызывают коррекцию экрана и предоставляемой пользователю оперативной памяти;

СТОП - прерывание любых действий "Фокала", вызывает сообщение об ошибке 25 – "ОСТАНОВ ПО КЛАВИШЕ СТОП".

Язык позволяет дополнительно использовать специальные символы (↑,↓,↖,↗,↘,↙,|↖) вводятся "ФОКАЛОМ" только в режиме редактирования "РЕД", в противном случае игнорируются.

3. Режимы работы и структура программ.

Язык "Фокал" позволяет работать в двух режимах:

- диалоговом,
- программном.

В диалоговом режиме операторы вводятся без номера строки непосредственно после звёздочки (символ *) и выполняются немедленно

после нажатия клавиши "Ввод". Выполнившись один раз, прямая команда не может выполниться вторично, пока не будет набрана снова.

Если данная строка операторов должна быть сохранена в оперативной памяти для дальнейшего использования, то перед ней необходимо поместить номер строки.

Строки операторов, помеченные номерами, называются косвенными строками операторов или косвенными командами.

Строка может содержать несколько операторов, разделённых ограничителем ";".

ФОРМАТ КОСВЕННОЙ КОМАНДЫ:

```
* ТТ.ПП <операторы>,
```

где ТТ.ПП - номер строки,

ТТ - номер группы строк

ПП - номер шага

Косвенная команда запоминается в памяти после нажатия клавиши "Ввод" и не выполняется до тех пор, пока ей не будет передано управление.

Последовательность косвенных команд образует программу на языке "Фокал", которая выполняется в порядке увеличения номеров строк, если нет дополнительных указаний, организуемых операторами DO, GOTO, IF, RETURN. Операторы, составляющие строку, реализуются в порядке их следования, начиная с первого; часть операторов строки может остаться нереализуемой вследствие передачи управления другой строке.

Запуск программы на "Фокале" выполняется только после введения прямой команды GOTO или DO ALL. Подобный режим работы называется программным. После завершения выполнения программы или прямой команды выдаётся символ к. Это означает, что система вышла в диалоговый режим и готова выполнять дальнейшие команды пользователя.

Для номеров строк в программах можно использовать любые номера с 1.01 до 99.99 с шагом 0.01 за исключением тех, которые оканчиваются на .00. После точки необязательно указывать два знака, т.е. 2.1 эквивалентно 2.10.

Разрешается также использовать номера от 100.1 до, 127.9 с шагом 0.1 за исключением тех, которые оканчиваются на 0.

4. Числа

В "Фокале" могут быть использованы любые десятичные числа в диапазоне от 10^{-38} до 10^{38}. Числа могут быть со знаком "+" или дробные числа – с десятичной точкой или в экспоненциальном формате. "Фокал" преобразует все числа в экспоненциальный формат с шестью значащими цифрами. Если указано больше 6 цифр, число будет округлено до 6 знаков. Константа в экспоненциальном формате состоит из двух частей: собственно константы и

порядка (целой степени числа). Порядок записывается целым числом, перед которым стоит буква E. Он может иметь знак "+" или "-".

Следующие ниже представления чисел идентичны:

```
70
70.00
7E+01
700E-01
70.00003
```

Заметим, что цифра "0" перед латинской заглавной буквой или набором букв информирует "Фокал" о том, что следующие после нуля буквы, за исключением буквы "E", следует рассматривать как цифры, равные порядковому номеру буквы в алфавите, а не как имя переменной, например:

```
*T 0NO
155.0000*
```

где

```
0NO=10^1*N+10^0*0=10*14+15=155
```

Стандартный формат, используемый "ФОКАЛОМ" для ввода и вывода чисел, включает 8 десятичных знаков: 4 до точки и 4 после. Первые нули не печатаются, последние печатаются. Существенны, однако, только первые 6 знаков, причём 6-ой может быть неточным вследствие ошибок округления.

5. Переменные

Язык "Фокал" позволяет непосредственно обрабатывать только арифметическую информацию. Поэтому в нём нет средств для описания типов данных. Для работы с символьной информацией используется встроенная функция FCHR.

Переменные в "Фокале" описываются их именами и, если переменная имеет индексы, то и значениями индексов.

Пример:

```
X(1), TOP(I), KE(5,6), Y, ZYX
```

Имена переменных могут состоять из одного или нескольких

символов. Первый символ должен быть буквой, но не буквой F, которая используется для имён функций. Остальные символы могут быть буквами или цифрами. Пользователь может написать имя переменной, состоящее более чем из двух символов, но "Фокал" использует только первые 2 символа для идентификации переменной.

Переменные, имена которых не начинаются с букв A и F, могут быть использованы в качестве номеров строк в операторах. Причина ограничения

состоит в том, что буква A используется в операторах DO ALL, WRITE ALL, ERASE ALL, а буква F – в названиях функций

Пример:

```
* 1.10 S X=3
* 1.20 D X
```

Эти же операторы можно записать и в таком виде:

```
* 2.2 D 3
```

В "Фокале" используются простые переменные и переменные с индексами. Переменные с индексами могут иметь не более двух индексов. Индексы заключаются в круглые скобки и разделяются запятой (в двухиндексных переменных).

Наряду с константами в качестве индексов могут использоваться арифметические выражения, но в качестве значения индекса будет принята целая часть значения выражения. Индексы могут принимать значения в диапазонах от -128 до +127 для двухиндексной переменной и от -32768 до +32767 для одноиндексной переменной. При превышении этих диапазонов значения берутся по модулю соответствующей границы.

Переменные в "Фокале" не требуют объявления. Пользователь должен знать, что переменные, появляющиеся в тексте программ или в прямых командах, интерпретируются "ФОКАЛОМ" следующим образом. Если переменная уже была введена ранее, то интерпретатор обращается к таблице переменных и отыскивает её там, устанавливая указатель на ячейку, содержащую значение переменной. Если же переменная в таблице не обнаружена, то она заносится в память в том смысле, что отводится место для размещения её имени и очищаются ячейки для размещения её значения.

Если переменная "X" ранее не была введена, то команда

```
*T X
```

приводит к распечатке

```
0.0000*
```

Переменная занимает в памяти 4 слова: одно слово для имени переменной, одно слово для индексов (для простых переменных это слово равно 0), два слова для значения. Простые переменные отыскиваются интерпретатором в том случае, когда их имена начинаются с разных символов, по сравнению с индексными переменными, имеющими одно имя, но разные индексы.

6. Массивы.

"Фокал" позволяет использовать одно и двумерные массивы (или иначе, одно – и двухиндексные переменные).

Индекс может быть числом, названием переменной или арифметическим выражением.

Пределы изменения индекса:

от -128 до +127 для двумерного массива;

от -32768 до +32767 для одномерного массива.

Индексы заключаются в круглые скобки, а в случае двумерного массива разделяются запятой.

Например: X(137), P0(I, J).

Массивы в "Фокале" не требуют объявления.

7. Арифметические выражения.

Арифметические выражения в "Фокале" аналогичны математическим формулам. Арифметическое выражение может состоять из следующих элементов: чисел, переменных, функции, знаков арифметических операций и скобок. Следует иметь в виду, что отдельное число, отдельная переменная, отдельная функция также считаются арифметическим выражением и являются его частным случаем.

Арифметические выражения вычисляются в соответствии с приоритетом операций и наличием скобок, т.е. с учётом следующих обстоятельств:

1. Выражения, заключённые в скобки, имеют наивысший приоритет; это означает, что они вычисляются прежде выражений, стоящих вне скобок;
2. При отсутствии скобок порядок выполнения операций следующий:
   - возведение в степень (^);
   - умножение (*);
   - деление (/);
   - сложение (+);
   - вычитание (-);
3. В конфликтных ситуациях, когда п. 1 и п. 2 недостаточны для определения порядка вычислений, следует помнить, что интерпретатор вычисляет выражение слева направо.

Если выражение содержит скобки внутри скобок, то сначала вычисляются внутренние скобки, а затем внешние.

В "Фокале" можно использовать три вида скобок: круглые (), квадратные [], угловые < >. Интерпретатором они воспринимаются одинаково, но следует помнить, что каждой открывающей скобке должна соответствовать закрывающая скобка того же типа.

8. Общий формат оператора языка "Фокал".

Числовая метка Имя оператора Текст оператора

1. Перечень операторов языка.

№	Название оператора	Имя оператора	Функция	Пример	Страницы описания
	2. Невыполняемые операторы.				
1	COMMENT	C	Комментарий	1.1 C начало	53, 55
	3. Вычислительные операторы.				
2	SET	S	Оператор присваивания	2.1 S A=I	21
	4. Операторы управления.				
3	DO	D	Выполнение группы операторов	3.1 D 10	54
4	FOR	F	Цикл	4.1 F =1,10; D 10	61
5	GO TO	G	Безусловный переход	5.1 G 10.1	16
6	IF	I	Переключатель	6.1. I (M) 3.1,3.2,2.1	46
7	QUIT	Q	Конец вычислений	7.1 Q	56
8	RETURN	R	Возврат из подпрограммы	8.1 R	55
9	XECUTE	X	Вызов встроенной функции	9.1 X FRAN()	77
	5. Операторы ввода – вывода.				
10	ASK	A	Ввод значений с клавиатуры	10.1 A A, B, C	32
11	TYPE	T	Вывод на экран	11.1 T A, B, C	34
12	LIBRARY	L G	Ввод программы с МЛ	L G RITM	69
		L S	Вывод программы на МЛ	L S RITM	67
		L F	Вывод на экран имён файлов с МЛ	L F	70

		L O	Вывод данных на МЛ	L O ПЕРЕМ.	70
		L I	Ввод данных с МЛ	L I ПЕРЕМ.	71
6. Операторы – директивы					
1	GO	G	Выполнить программу сначала	G	16
2	ERASE	E N	Удалить строку с номером N	E 20.90	30
		E N	Удалить группу строк с номером N	E 20	30
		E A	Уничтожить текст и переменные программы	E A	30
		E T	Уничтожить текст программы	E T	30
		E	Уничтожить переменные программы	E	30
3	MODIFY	M N	Скорректировать текст строки с номером N	M 15.05	27
4	WRITE	W N.M	Вывести строку N.M на экран	W 1.30	31
		W N	Вывести группу строк N на экран	W 30	31
		W	Вывести весь текст программы на экран		31

7. Встроенные функции.

1	FSIN	Вычисление синуса (аргумент в радианах)
2	FCOS	Вычисление косинуса
3	FTAN	Вычисление тангенса
4	FATAN	Вычисление арктангенса
5	FASIN	Вычисление арксинуса
6	FACOS	Вычисление арккосинуса
7	FEXP	Вычисление экспоненты

8	FLOG	Вычисление логарифма натурального
9	FLOG10	Вычисление логарифма десятичного
10	FX	Обращение к общей шине
11	FCHR	Обработка текстовой информации
12	FRAN	Функция случайных чисел
13	FABS	Вычисление абсолютной величины
14	FSGN	Вычисление знака числа
15	FITR	Вычисление целой части числа
16	FSQT	Вычисление квадратного корня
17	FSBR	Функция, программируемая пользователем
18	FK	Установка курсора
19	FT	Формирование точки
20	FV	Формирование вектора
21	FP	Работа с портом ввода/вывода

Приложение 3

СЛОВАРЬ ТЕХНИЧЕСКИХ ТЕРМИНОВ

АДРЕС -

цифровое или буквенно-цифровое обозначение элемента памяти

АДРЕСНОЕ ПРОСТРАНСТВО -

совокупность возможных для данной ЭВМ адресов.

АЛГОРИТМ -

совокупность правил, определяющих эффективную процедуру решения любой задачи из некоторого заданного класса задач.

АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ -

числа и переменные, соединённые знаками арифметических операций.

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ -

совокупность арифметических преобразований числовых операндов.

БАЙТ -

единица количества информации, соответствующая группе из восьми соседних двоичных битов, которой ЭВМ может оперировать как единым целым.

БИТ -

двоичная единица измерения количества информации.

БЛОК-СХЕМА -

условное обозначение программы для ЭВМ или алгоритма решения некоторой задачи в виде условных графических обозначений.

ВВОД-

чтение информации с внешнего устройства в память ЭВМ.

ВНЕШНЕЕ УСТРОЙСТВО -

устройство, подключаемое к ЭВМ и работающее под её управлением.

ВСТРОЕННАЯ ФУНКЦИЯ -

программа, написанная в кодах ЭВМ, вызываемая на выполнение по имени, например, FSIN (X).

ВЫВОД -

запись информации из памяти ЭВМ на внешнее устройство.

ДАННЫЕ -

информация, представленная в формализованном виде, позволяющем передавать или обрабатывать её при помощи некоторого процесса.

ДИАГНОСТИКА ОШИБОК -

процесс обнаружения синтаксических ошибок, который выполняется интерпретатором.

ДИАЛОГОВЫЙ РЕЖИМ -

режим работы интерпретатора языка "Фокал", при котором командная строка выполняется сразу же после нажатия клавиши "Ввод".

ИНТЕРПРЕТАТОР ЯЗЫКА "ФОКАЛ" -

программа, позволяющая переводить операторы и директивы, изложенные на языке "Фокал", в команды, непосредственно выполняемые ЭВМ.

КОМАНДА -

предписание языка "Фокал", выполняемое сразу же после нажатия клавиши "ВВОД".

КОНСТАНТА -

неизменные в ходе выполнения программы данные, с которыми оперирует программа.

КОНТРОЛЬНАЯ СУММА -

число, служащее для контроля правильности хранения данных.

КОСВЕННАЯ КОМАНДА -

оператор языка "Фокал" (группа операторов), перед которым помещён номер строки.

ЛОГИЧЕСКИЕ ОШИБКИ -

ошибки, вызванные неверным алгоритмом программы.

МАШИННОЕ СЛОВО -

последовательность двоичных единиц информации, состоящая из двух байтов. С машинным словом ЭВМ может оперировать, как с единым целым.

ОБЛАСТЬ ДЕЙСТВИЯ ЦИКЛА -

многократно используемые в процессе выполнения цикла участки программы.

ОПЕРАНД -

часть оператора, следующая через пробел от имени оператора.

ОПЕРАТОР -

допустимое в данном языке программирования предписание, предназначенное для определения некоторого шага процесса обработки информации на ЭВМ.

ОТЛАДКА -

процесс проверки правильности программы, обнаружения и исправления ошибок, допущенных при её составлении.

ОТЛАДОЧНЫЙ ОПЕРАТОР -

любой из операторов, вставляемый в программу для облегчения отладки. После отладки отладочный оператор удаляется из программы.

ПЕРЕДАЧА УПРАВЛЕНИЯ -

процесс перехода от выполнения одного оператора к выполнению другого.

ПЕРЕМЕННАЯ -

числовая величина в программе, значение которой изменяется в ходе её выполнения. Переменная имеет в языке "Фокал" имя.

ПОДПРОГРАММА -

часть программы, реализующая определённый алгоритм, оформленная таким образом, что допускает обращение к ней из любой части программы.

ПОЛЬЗОВАТЕЛЬ -

человек, который использует возможности ЭВМ в своей деятельности.

ПОРТ ВВОДА-ВЫВОДА -

техническое устройство ЭВМ, служащее для подключения к ней устройств пользователя.

ПОСТОЯННОЕ ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО -

техническое устройство пользователя, позволяющее сохранять информацию без источников анергии.

ПРОГРАММА -

описание алгоритма решения задачи, заданное на языке ЭВМ.

РЕДАКТИРОВАНИЕ ДАННЫХ -

преобразование формы представления данных к виду, удобному для использования.

СЛОВО -

смотри "МАШИННОЕ СЛОВО".

ФАЙЛ -

поименованный набор данных, с которым оперирует ЭВМ, как с единым целым.

ЦИКЛ -

организация программы, при которой отдельные её участки многократно повторяются.

ЯЗЫК ПРОГРАММИРОВАНИЯ -

язык общения человека с ЭВМ, предназначенный для описания совокупности инструкций, выполнение которых обеспечивает правильное решение требуемой задачи.

Приложение 4

ТАБЛИЦА КОДОВ СИМВОЛОВ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		СБР ТАБ *	ПРО БЕЛ	0	@	P	`	p		ШАГ	¶	┤	ю	п	Ю	П
1			!	1	A	Q	a	q	ПОВТ **	К	┴	←	а	я	А	Я
2		↖	"	2	B	R	b	r	ИСУ *	З	♥	╬	б	р	Б	Р
3	КТ	↑	#	3	C	S	c	s		С	┐	↑	ц	с	Ц	С
4		↓	$	4	D	T	d	t	БЛР *	Ч	╡	♣	д	т	Д	Т
5		↓	%	5	E	U	e	u		ГРАФ	├	─	е	у	Е	У
6		|←	&	6	F	V	f	v		ЗАГ	└	╫	ф	ж	Ф	Ж
7	ЗВ	→|	'	7	G	W	g	w		СТИР	═	│	г	в	Г	В
8	←	←+	(	8	H	X	h	x			╤	♦	х	ь	Х	Ь
9	З	→	)	9	I	Y	i	y	ТАБ **	⇨|	♠	┘	и	ы	И	Ы
10	ВВОД	↑	*	:	J	Z	j	z		КУР СОР *	┌	╪	й	з	Й	З
11		↓	+	;	K	[	k	{		32/64 *	┬	╥	к	ш	К	Ш
12	СБР	↖	,	<	L	\	l	|	РП	ИНВС	╨	╧	л	э	Л	Э
13	УСТ ТАБ	↗	-	=	M	]	m	}		ИНВЭ *	↓	╞	м	щ	М	Щ
14	РУС **	↘	.	>	N	^	n	~		УСТ ИНД *	┼	→	н	ч	Н	Ч
15	ЛАТ **	↙	/	?	O	_	o	ЗБ		ПОДЧ	╟	▒	о	ъ	О	Ъ

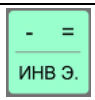
П_р_и_м_е_ч_а_н_и_е. * – коды передаются с драйвера клавиатуры на драйвер ТВ-приёмника, минуя внешнюю программу. ** – коды используются только драйвером клавиатуры.

Приложение 5

ТАБЛИЦА СООТВЕТСТВИЯ КЛАВИШ ПЛЁНОЧНОЙ
И КЛАВИШНОЙ КЛАВИАТУРЫ

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
НР	АР2
СУ	СУ
СТОП	СТОП
ШАГ	ШАГ
ИНД СУ	ИНД СУ
БЛОК РЕД	БЛОК РЕД
ГРАФ	СУ / АР2 / У U *
ЗАП	СУ / АР2 / Ж V
СТИР	СУ / АР2 / В W
УСТ ТАБ	СУ / М M
СБР ТАБ	СУ / П P

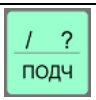
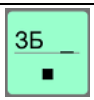
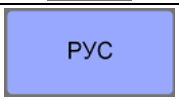
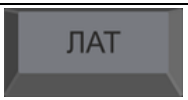
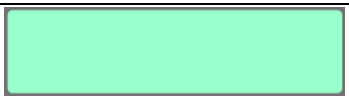
КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
ВС	ВС
СБР / РП	СБР
ГТ	СУ / Т Т
СБР↦	
; + / 32/64	; +
1 ! / К1	1 !
2 “ / К2	2 ”
3 # / К3	3 #
4 ¤ / К4	4 ¤
5 % / К5	5 %
6 & / К6	6 &
7 ‘ / К7	7
8 (/ К8	8 (

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
	
	
	
	
	
	
	
	
	
	
	

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
↘	СУ / Ч ¬
↤	⇤
↔ РЕД	⇐
↦	⇒
СБР↦	⇸
ПОВТ	ПОВТ
Й J ┌	Й J
Ц C ┐	Ц C
У U –	У U
К K ┬	К K
Е E ├	Е E
Н N ┼	Н N
Г G =	Г G

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
Ш [/ ╥	Ш [
Щ] / ╞	Щ]
З Z / ╪	З Z
Х H / ╤	Х H
: * / КУРСОР	: *
Ф F / └	Ф F
Ы Y / ┘	Ы Y
В W / │	В W
А A / ┴	А A
П P / ┤	П P
Р R / ╬	Р R
О O / ║	О O
Л L / ╨	Л L

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
Д D ╡	Д D
Ж V ╫	Ж V
Э \ ╧	Э \
. > УСТ.ИНД.	. >
Я Q ←	Я Q
Ч ¬ →	Ч ¬
С S ↑	С S
М M ↓	М M
И I ♠	И I
Т T ♣	Т T
Ь X ♦	Ь X
Б B ♥	Б B
Ю @ π	Ю @

КЛАВИШИ ПЛЁНОЧНОЙ КЛАВИАТУРЫ	КЛАВИШИ КЛАВИШНОЙ КЛАВИАТУРЫ
	
	
	
	
	
	
	
	
	
	
	
	

* Запись вида СУ/АР2/У обозначает одновременное нажатие клавиш СУ, АР2 и У.

Заказ 1623 Тираж 10000
Орехово-зуевская типография