ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА НИКОЛАЯ II»

Кафедра «Электропоезда и локомотивы»

Н.И.ДОЛГАЧЕВ

ИНФОРМАТИКА

Часть II

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ К ЛАБОРАТОРНЫМ РАБОТАМ

М о с к в а - 2016

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА НИКОЛАЯ II»

Кафедра «Электропоезда и локомотивы»

Н.И.ДОЛГАЧЕВ

ИНФОРМАТИКА

Часть II

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ К ЛАБОРАТОРНЫМ РАБОТАМ

для студентов I курса
специальности 23.05.03
«Подвижной состав железных дорог»
специализации «Локомотивы»

М о с к в а - 2016

УДК 007

Д-64

Долгачев Н.И. Информатика. Ч. 2: Учебно-методическое пособие. - М.: МГУПС (МИИТ), 2016. - 44 с.

Приведены основы алгоритмического языка ТУРБО-БЕЙСИК и среды программирования, порядок оформления заданий по лабораторным работам которые студенты специальности 23.05.03 «Подвижной состав железных дорог» специализации «Локомотивы» должны выполнить с использованием современных вычислительных машин. Приведены примеры составления программ с использованием файлов данных, функций пользователя, процедур, подпрограмм, операторов графики.

Рецензент: доцент кафедры «Машиноведение, проектирование, стандартизация и сертификация» МГУПС (МИИТ), к.т.н. Козлов В.В.

В В Е Д Е Н И Е

Цель лабораторных занятий по дисциплине «Информатика» - обучить студентов программированию с применением современных ПЭВМ для решения задач по специальности. Данное пособие разработано именно в этом аспекте. Так как наибольшей популярностью среди учащихся пользуются различные диалекты «Бейсика», рабочей программой предусмотрено изучение компилирующего алгоритмического языка и среды TURBO BASIC (TB).

Во 2-ой части учебно-методического пособия приведены конструкции языка TB, позволяющие использовать файлы данных, нестандартные функции, процедуры, подпрограммы, операторы графики, а также примеры составления алгоритмов и программ, выполнения и оформления расчетов на ПЭВМ. Приведен список основных сообщений об ошибках, возникающих при работе в среде TB.

1 ВВОД-ВЫВОД ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ВНЕШНИХ НОСИТЕЛЕЙ ИНФОРМАЦИИ

Файлы данных представляют собой последовательность записей, разделенных между собой запятой либо пробелом, если это числа, и запятой, если это символьные записи. Если обращение к записям данных осуществляется в произвольном порядке, такой файл называют файлом прямого доступа. Если, чтобы прочитать какую-либо запись, необходимо прочитать все предыдущие записи от начала файла, такой файл называют файлом последовательного доступа. Наиболее простой формой организации работы с файлами данных являются файлы последовательного доступа, которые и рассматриваются ниже.

Файлы данных могут создаваться на внешних носителях информации (в дальнейшем – просто "носитель") двумя способами. Первый - это так же, как и программные файлы, посредством набора информации с клавиатуры в редакторе Edit и последующей записи на носитель по команде Write to или Save меню File. Второй способ - создание файлов программным путем с помощью специальных операторов. При создании и использовании файла программным путем необходимо соблюдать 3 этапа работы с файлом:

1. Открыть файл с помощью оператора OPEN по любой из двух предлагаемых форм:

а) первая форма

```
OPEN "путь к файлу" FOR INPUT     AS #N
OPEN "путь к файлу" FOR OUTPUT AS #N
OPEN "путь к файлу" FOR APPEND AS #N
```

б) вторая форма

```
OPEN  "I",   #N, "путь к файлу"
OPEN "O",  #N, "путь к файлу"
OPEN "A",  #N, "путь к файлу"
```

Вместо непосредственного указания пути к файлу в кавычках может стоять символьная переменная без кавычек, содержащая информацию о пути к файлу. Ключевое слово FOR INPUT или символ "I" означает, что файл открывается для ввода (чтения) данных с носителя в оперативную память (ОП); FOR OUTPUT или символ "I" - для вывода (записи) данных из ОП на носитель; FOR APPEND или символ "A" - для добавления данных в конец файла; N - номер (от 1 до 7) канала ввода-вывода, ассоциированный с файлом.

2. Указать режим работы с файлом:

а) INPUT #N, список переменных (ввод данных с носителя в ОП ПК, отведенную под переменные списка);

б) PRINT #N, список выражений (управляемый вывод по зонам значений выражений списка из ОП на носитель);

в) WRITE #N, список переменных (вывод по зонам значений переменных списка через запятую из ОП на носитель).

г) PRINT #N, USING шаблон; список выражений (вывод по шаблону значений выражений списка из ОП на носитель);

3. Закрыть файл с помощью оператора CLOSE: CLOSE #N. Если необходимо закрыть одновременно несколько файлов, номера каналов можно перечислить через запятую (например: CLOSE #2, #4, #1).

1.1 Пример создания файла с клавиатуры и его использования

Допустим, что в качестве исходных данных используются значения тока $I_г$ в амперах (0, 2600, 6400, 6600) и соответствующие им значения в вольтах (750, 700, 280, 0) напряжения $U_г$ на зажимах тягового генератора тепловоза. Применяя команду Edit главного меню, перейдите в режим редактирования. В окне Edit, начиная с 1-ой позиции, с клавиатуры введите в 1-ую строку через запятую значения тока $I_г$, а с 1-ой позиции 2-ой строки значения напряжения $U_г$.

Затем выполните следующие команды: Esc, File, Write to, не забывая, где следует, нажимать Enter. В появившемся окне наберите путь к файлу, например: E:\F1.DAT. В результате значения $I_г$ и $U_г$ будут записаны на носитель "E" в корневой (главный) каталог под именем "F1.DAT". Чтобы убедиться в правильности вывода информации на носитель, очистите редактор командой New меню File и примените команду Load. В появившемся окне наберите маску: E:*.*. Выберите файл с именем "F1.DAT" и загрузите его в редактор. Файл должен иметь вид:

0, 2600, 6400, 6600
750, 700, 280, 0

Этот файл для ввода данных в ОП ПК будет использован в следующем примере (см. 1.2).

1.2 Пример выполнения задания с использованием созданного файла и возможностью создания нового файла с результатами расчётов

1.2.1 Условие и цели задачи

Составить программу расчета величины напряжения $U_г$ на зажимах тягового генератора тепловоза, если область определения этой функции по току $I_г$ генератора ограничена 3-мя зонами:

1. если $I_{г1} \le I_г \le I_{г2}$, то $U_г=U_{г2}+k\cdot(I_{г2}-I_г)$, где $k=(U_{г1}-U_{г2})/(I_{г2}-I_{г1})$;
2. если $I_{г2} < I_г < I_{г3}$, то $U_г=1835-0{,}66\cdot I_г+9{,}9\cdot 10^{-5}\cdot I_г^2-5{,}274\cdot 10^{-9}\cdot I_г^3$;
3. если $I_{г3} \le I_г \le I_{г4}$, то $U_г=U_{г4}+k\cdot(I_{г4}-I_г)$, где $k=(U_{г3}-U_{г4})/(I_{г4}-I_{г3})$.

Значения величин, входящих в описание функции $U_г(I_г)$, следующие: $I_{г1}=0$, $I_{г2}=2600$ А, $I_{г3}=6400$ А, $I_{г4}=6600$ А, $U_{г1}=750$ В, $U_{г2}=700$ В, $U_{г3}=280$ В, $U_{г4}=0$. В разделе 1.1 создан файл "F1.DAT" из этих значений.

В 1-й и 3-й зонах функция $U_г$ описана уравнениями прямой линии, и для построения ее графика достаточно задать граничные значения тока $I_г$ по зонам. Во 2-й зоне функция $U_г$ описана нелинейно (полиномом 3-й степени), и для плавного построения кривой, кроме граничных значений тока, зададим ещё два между ними ($I_г=3900$ А, $I_г=5200$ А),. Таким образом, обеспечено, как минимум, 6 значений тока $I_г$ генератора для построения графика функции $U_г=f_1(I_г)$ (см. рис. 1).

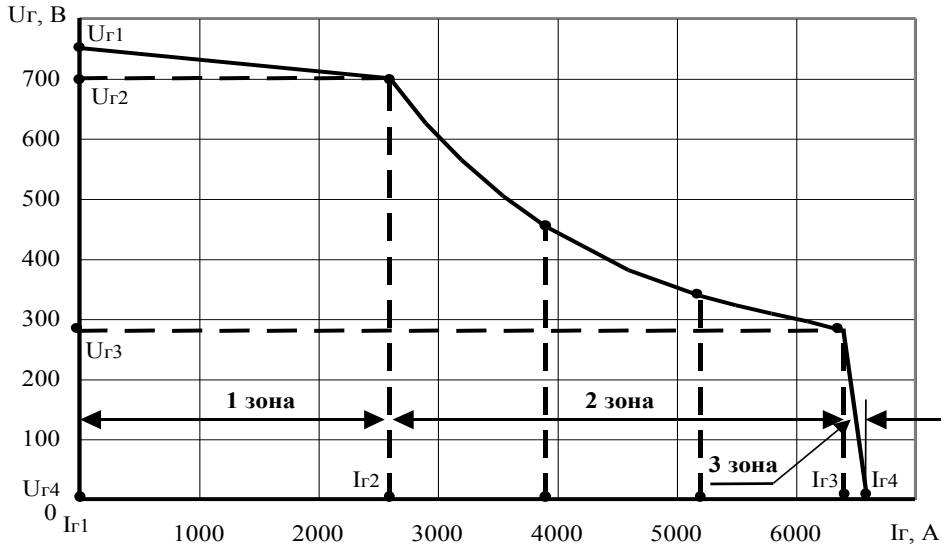


Рисунок 1 График функции $U_г(I_г)$

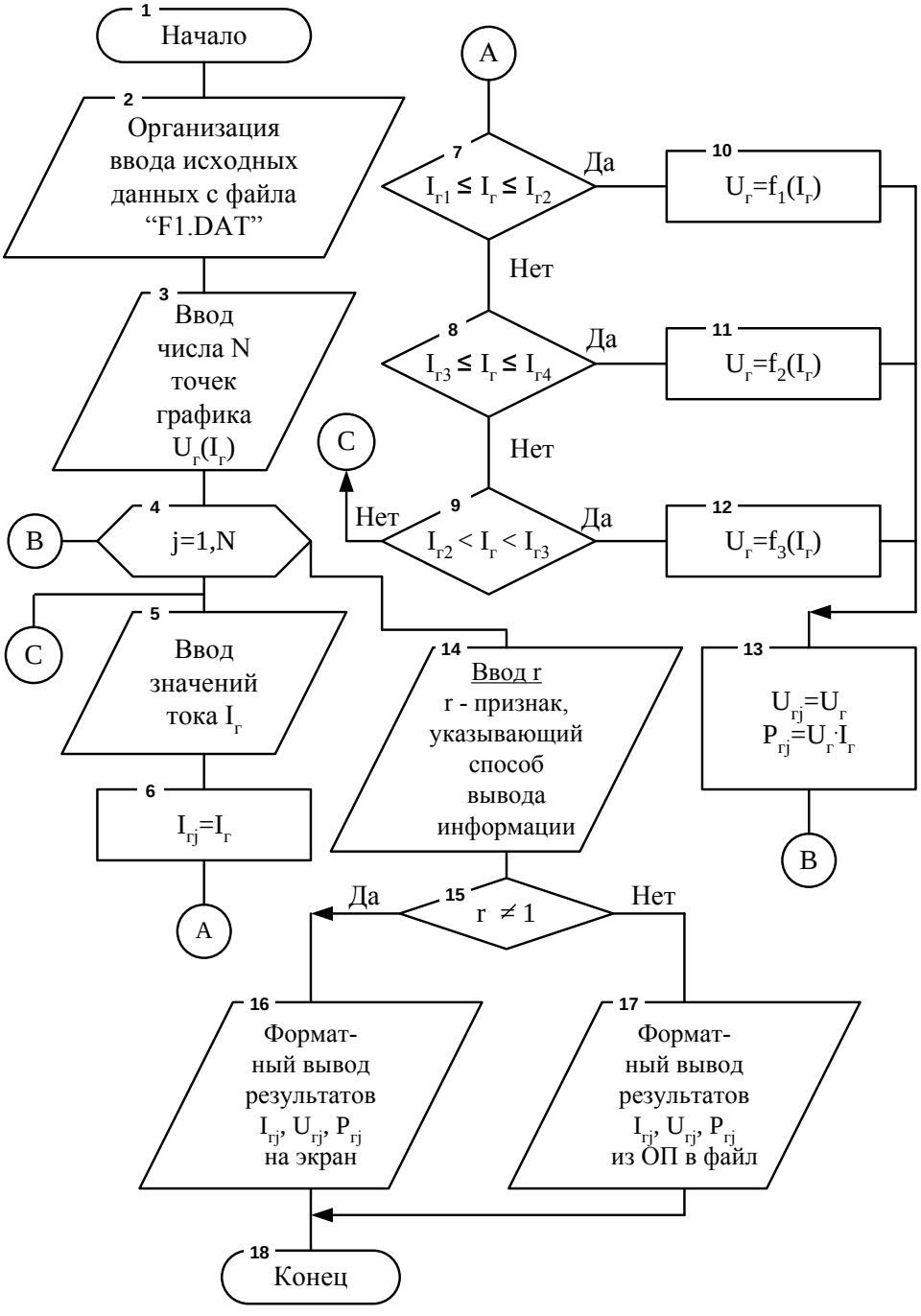
Вычислить с помощью программы значения мощности по формуле $P_г=U_г\cdot I_г$.

Записать результаты расчета (значения напряжения $U_г$ и мощности $P_г$ тягового генератора) и заданные значения тока $I_г$ на носитель, чтобы затем

использовать их для построения графиков $U_г=f_1(I_г)$ и $P_г=f_2(I_г)$. Эти три показателя при выполнении программы запоминаются в ОП ПК в виде динамических массивов размером N. Зададимся следующей формой представления массивов на носителе:

N
$I_{г1}$, $U_{г1}$, $P_{г1}$
$I_{г2}$, $U_{г2}$, $P_{г2}$
……..
$I_{гN}$, $U_{гN}$, $P_{гN}$

1.2.2 Блок-схема алгоритма решения



1.2.3 Таблица соответствия

Математические переменные	$I_г$	$U_г$	$I_{гj}$	$U_{гj}$	$P_{гj}$	j	k
Машинные переменные	I	U	I(J)	U(J)	P(J)	J	k

Математические переменные	$I_{г1}$	$I_{г2}$	$I_{г3}$	$I_{г4}$	$U_{г1}$	$U_{г2}$	$U_{г3}$	$U_{г4}$
Машинные переменные	I1	I2	I3	I4	U1	U2	U3	U4

1.2.4 Программа расчета на ТВ

```
cls
input "путь к файлу" r$     'после знака ? набрать E:\F1.DAT;
open r$ for input as #1                 'открытие файла для ввода;
input #1, I1, I2, I3, I4                 'ввод в ОП значений тока;
input #1, U1, U2, U3, U4                'ввод значений напряжения;
close #1       'закрытие файла с номером канала ввода 1
input "введем число точек графика функции" N
DIM I(N), U(N), P(N)
PRINT: PRINT "введем "; N; " значений тока"
FOR J=1 TO N  'начало цикла ввода тока и расчёта напряжения и
                  'мощности;
M1:
print "I(";J;")=";: input,I: I(J)=I
IF I>=I1 AND I<=I2 THEN 'начало оператора условного перехода
                                  'блочного типа
k=(U1-U2)/(I2-I1)
U=U2+k*(I2-I)
ELSEIF I>=I3 AND I<=I4 THEN
k=(U3-U4)/(I4-I3)
U=U4+k*(I4-I)
ELSEIF I>I2 AND I<I3 THEN
U=1835-.66*I+9.9e-5*I^2-5.274e-9*I^3
ELSE
print "недопустимое значение тока генератора; повторить ввод"
GOTO M1
END IF 'конец оператора условного перехода блочного типа
U(J)=U: P(J)=I*U/1000
NEXT J 'конец цикла расчёта
input "вывод на носитель - 1, нет - 0" r
```

```
if r<>1 then goto m3
f$="####&"
input " укажите путь к файлу" r$
open "o", #1, r$
WRITE  #1, N
FOR J=1 TO N
print #1, using f$; I(J), ",", U(J), ",", P(J)
NEXT J
close #1: goto m2
m3:
I$="ток, А": U$="напряжение, В": P$="мощность, кВт"
print: print tab(4) I$; tab(17) U$; tab(34) P$
f$="####"
FOR J=1 TO N
print"I(";J;")=";: print using f$; I(J);
print tab(18) "U(";J;")=";: print using f$; U(J);
print tab(35) "P(";J;")=";: print using f$; P(J)
NEXT J
m2:
print: print "Конец работы программы"
end
```

1.2.5 Выполнение программы

Run Enter

введем число точек графика функции? 6 Enter
введем 6 значений тока

I(1)=   0 Enter
I(2)=2600 Enter
I(3)=3900 Enter
I(4)=5200 Enter
I(5)=6400 Enter
I(6)=6600 Enter

вывод на носитель – 1, нет – 0 ? Enter

ток, А	напряжение, В	мощность, кВт
I(1)= 0	U(1)= 750	P(1)= 0
I(2)=2600	U(2)= 700	P(2)=1820
I(3)=3900	U(3)= 454	P(3)=1770
I(4)=5200	U(4)= 338	P(4)=1760

I(5)=6400 U(5)= 280 P(5)=1792
I(6)=6600 U(6)= 0 P(6)= 0

Конец работы программы

При запросе была нажата клавиша Enter, что равносильно нажатию цифры 0, поэтому произошёл форматный вывод результатов расчёта на экран. Попробуйте при запросе нажать цифру 1, что будет соответствовать выводу на носитель (предварительно подготовьте носитель и путь к файлу). Программа выведет на экран выражение «укажите путь к файлу». Укажите путь и нажмите Enter.

Чтобы убедиться в правильности создания файла на носителе, очистите редактор командой New меню File и примените команду Load. В появившемся окне наберите маску: E:*.*. Выбирайте рамкой путь к файлу. Установив окончательно имя файла, загрузите его в редактор. Файл должен иметь вид:

```
6
    0, 750,    0
2600, 700, 1820
3900, 454, 1770
5200, 338, 1760
6400, 280, 1792
6600,   0,    0
```

1.3 Пример создания файлов, имена которых формируются в процессе вычислений

1.3.1 Условие и цель задачи

Допустим, имеется массив данных $X_i=\{X_1, X_2, X_3, ..., X_n\}$, где i изменяется от 1 до n. При обработке массивов экспериментальных или статистических данных часто приходится определять:

M - среднее (арифметическое) значение величины X;

D - средний квадрат отклонений X от M (дисперсию);

σ - среднее квадратичное отклонение ($\sqrt{D}$).

$$M = (1/n)\cdot\sum_{i=1}^{n} X_i , \qquad D = (1/n)\cdot\sum_{i=1}^{n}(X_i - M)^2 , \qquad \sigma = \sqrt{D} .$$

Исходные данные по вариантам приведены в таблице 1.3.1

Таблица 1.3.1 - Исходные данные

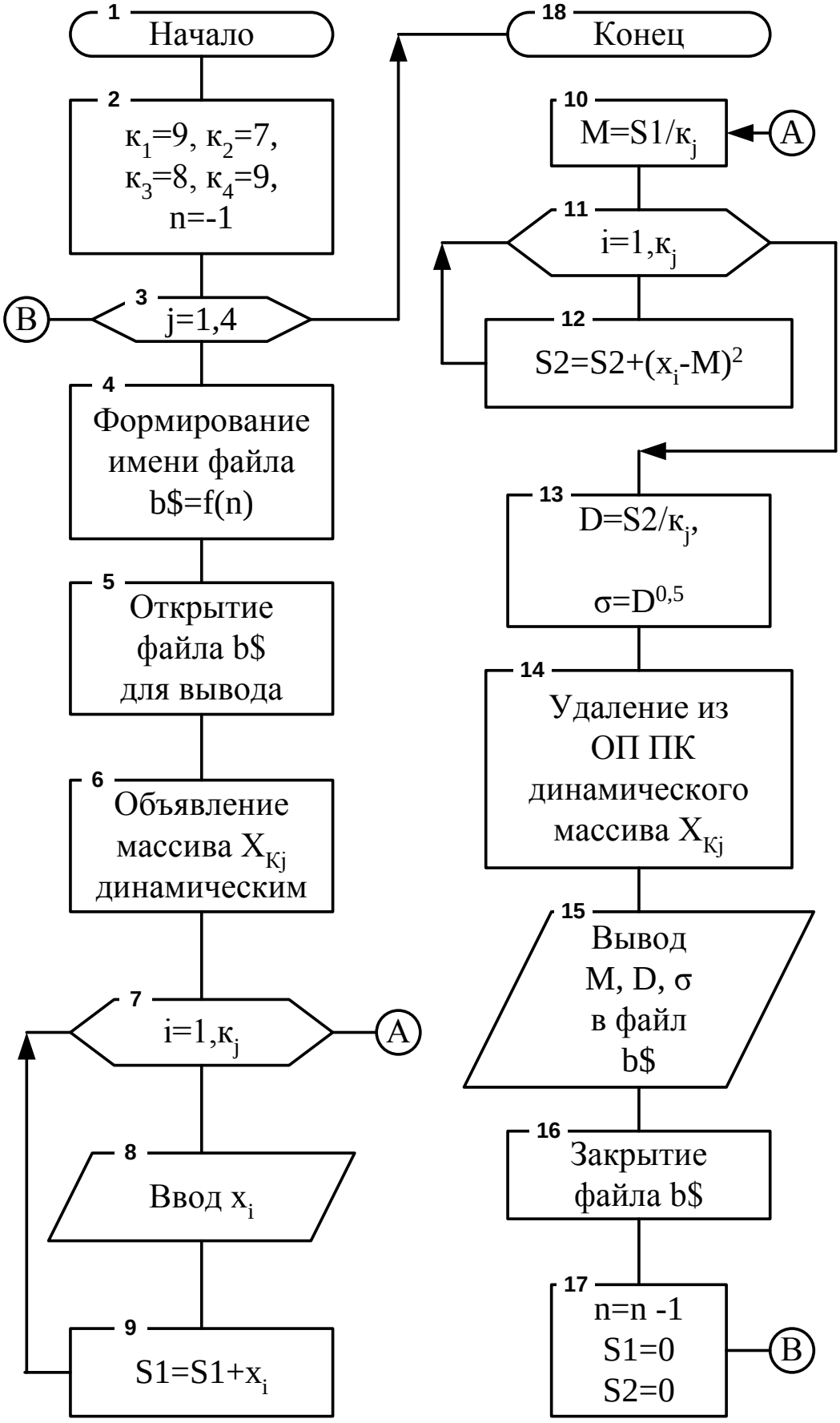
Вариант	Массив данных X_i								
1	76,8	76,1	76,8	77,4	80,2	81,4	82,8	82,7	87,6
2	15,0	17,0	19,0	21,0	20,0	18,0	20,0	-	-
3	95,0	99,0	94,0	101	97,0	95,0	96,0	98,0	-
4	1,75	1,87	1,49	1,78	1,70	1,73	1,80	1,72	1,81

Составить программу расчета M, D, σ и записи их значений в файл. Число файлов принять соответствующим числу вариантов. Файлы создать на носителе "E" в главном каталоге. Сумму $\sum_{i=1}^{n} x_i$ накапливать в ОП под именем S1, а сумму $\sum_{i=1}^{n} (x_i - M)^2$ – под именем S2. Параметр σ обозначить в программе переменной S.

1.3.2 Метод решения поставленной задачи

Обозначим файлы так, чтобы в имени файла в цикле изменялся только номер, соответствующий номеру варианта, например: F-1.DAT, F-2.DAT, F-3.DAT, F-4.DAT. Символьную переменную B$ в программе используем для формирования имени файлов. Формирование имени производится путем конкатенации (соединения) через символ «+» частей имени в одно целое. Функция STR$(N) преобразует число N, изменяющееся в цикле от -1 до -4, в его строковое представление. Обозначим через K_j массив, содержащий в качестве элементов размер массива X_i по каждому j-ому варианту расчета, т.е. K_1=9, K_2=7, K_3=8, K_4=9. Массив X_i будем объявлять для обработки и удалять после использования из оперативной памяти ПК как динамический.

1.3.3 Блок-схема алгоритма решения



1.3.4 Программа на ТВ

```
CLS
K(1)=9: K(2)=7: K(3)=8 :K(4)=9
DATA 76.8, 76.1, 76.8, 77.4, 80.2, 81.4, 82.8, 82.7, 87.6
DATA  15, 17, 19, 21, 20, 18, 20
DATA  95, 99, 94, 101, 97, 95, 96, 98
DATA 1.75, 1.87, 1.49, 1.78, 1.70, 1.73, 1.80, 1.72, 1.81
C1$="##.##&": C2$=","
n=-1
FOR J=1 TO 4
B$="E:\F"+STR$(N)+".DAT"
OPEN "O", #1, B$
DIM X(K(J))
FOR I=1 TO K(J): READ X(I): S1=S1+X(I): NEXT I
M=S1/K(J)
FOR I=1 TO K(J): S2=S2+(X(I)-M)^2: NEXT I
ERASE X: D=S2/K(J): S=SQR(D)
PRINT #1, USING C1$; M, C2$, D, C2$, S
CLOSE #1
n=n-1: S1=0: S2=0
NEXT J
```

После выполнения программы загрузить в ОП ПК поочередно созданные файлы. Их содержанием должна быть строка со значениями переменных M, D, σ:

Имя файла: "F-1.DAT"	"F-3.DAT"
80.20, 13.00, 3.61	96.88, 4.86, 2.20
"F-2.DAT"	"F-4.DAT"
18.57, 3.67, 1.92	1.74, 0.01, 0.10

2 ФУНКЦИЯ ПОЛЬЗОВАТЕЛЯ DEF FN

В разделах 2 ÷ 4 рассматривается материал, одинаковый по содержанию и различный только по форме. Речь идет о выделении из общей программы тех её частей, к которым необходимо обращаться не менее одного раза для выполнения одних и тех же операций, но с разными параметрами. Такими выделенными частями (программными модулями) могут быть функция DEF FN, процедура и подпрограмма.

DEF (DEFinition) FN (FuNction) означает определение функции пользователя, в отличие от встроенных функций. За префиксом FN без пробела

ставится имя функции. Имя задается аналогично имени переменной и используется для обращения к функции. Определение функции может стоять в любом месте программы. Функция DEF FNимя может быть однострочной и многострочной. В первом случае она имеет следующий общий вид:

DEF FNимя (список формальных параметров) = выражение

Выражение должно содержать указанные в скобках параметры и может содержать другие переменные. Тип выражения должен соответствовать типу имени функции.

Обращение к функции происходит из программы. Оно имеет следующий общий вид:

FNимя (список фактических параметров)

Число фактических параметров должно соответствовать числу формальных и не превышать 16. Формальными параметрами должны быть переменные. Фактическими параметрами могут быть переменные, константы, выражения, отдельные элементы массивов. Имена переменных в функции могут совпадать с именами переменных программы, но это разные переменные. При обращении к функции из программы происходит присвоение формальным параметрам значений фактических параметров, вычисление выражения и возвращение результата в программу в то место, откуда была вызвана функция.

Во втором случае (более универсальном при многострочности) функция DEF FNимя может быть применена в двух вариантах:

а) DEF FNимя (список формальных параметров)
операторы языка ТВ
Fnимя = выражение, содержащее формальные параметры
END DEF

При этом варианте программа должна содержать обращение к функции с обязательным списком фактических параметров.

б) DEF FNимя
операторы языка ТВ
Fnимя = выражение
END DEF

При этом варианте обращение к функции из программы не должно содержать списка фактических параметров (т.к. после имени функции нет списка формальных параметров).

В обоих вариантах функция представлена в виде блока, у которого есть начало (DEF FNимя) и конец (END DEF), а все, что между ними, называется телом функции. Наличие оператора присваивания Fnимя = ... - желательно, иначе значение, возвращаемое функцией в программу, будет неопределенным. Можно осуществить выход из тела функции с помощью оператора EXIT DEF, не дожидаясь окончания ее работы по END DEF,

Внутри функции можно, кроме обычных операторов ТВ, использовать операторы, управляющие областью действия переменных:

LOCAL, за которым следует список переменных, разделённых запятыми (область действия - внутри тела функции; при выходе из функции значения переменных теряются).

STATIC список переменных (область действия – внутри тела функции; при выходе из функции значения переменных сохраняются и могут быть использованы при повторном обращении к функции).

SHARED список переменных (область действия глобальная, т.е. не только внутри тела функции, но и в остальной части программы).

По умолчанию (при отсутствии указанных операторов) все переменные, появляющиеся в теле функции считаются глобальными. И, наконец, функции могут вызывать другие функции и сами себя. В последнем случае такой вызов называется рекурсивным. Рекурсивные вызовы в данном методическом указании не рассматриваются.

2.1 Пример использования однострочного определения функции DEF FN

2.1.1. Условие и цель задачи

Составить программу расчета основного сопротивления движению в режиме тяги поезда, состоящего из локомотива и состава, сформированного из двух типов вагонов, при изменении скорости движения V от начального значения $V_н$=10 км/ч до конечного значения $V_к$=90 км/ч с шагом ΔV=20 км/ч по формуле:

$$W_o = w_o' \cdot m_л \cdot g + w_o'' \cdot m_c \cdot g,$$

где w_o' - основное удельное сопротивление движению тепловоза, Н/кН;

w_o'' - основное удельное сопротивление движению состава, Н/кН;

$m_л$ - масса локомотива, $m_л$=258 т;

m_c - масса состава, m_c=3600 т;

g - ускорение свободного падения, g=9,81 м/с^2.

$$w_o' = 1{,}9 + 0{,}01 \cdot V + 0{,}0003 \cdot V^2;$$

$$w_o'' = \alpha_1 \cdot w_{o1}'' + \alpha_2 \cdot w_{o2}'',$$

где w_{o1}'' - удельное сопротивление движению грузовых вагонов 1-го типа;

α_1 - доля их по массе в составе, $\alpha_1 = 0{,}3$;

w_{o2}'' - то же для 2-го типа, $\alpha_2 = 0{,}7$.

$$w_{o1}'' = 0{,}7 + (3 + 0{,}1 \cdot V + 0{,}0025 \cdot V^2)/m_{во1};$$
$$w_{o2}'' = 0{,}7 + (6 + 0{,}038 \cdot V + 0{,}0021 \cdot V^2)/m_{во2},$$

где $m_{во1}$ - масса вагона 1-го типа, приходящаяся на одну ось, $m_{во1} = 20$ т;

$m_{во2}$ - то же для 2-го типа, $m_{во2} = 18$ т.

2.1.2 Метод решения поставленной задачи

Из приведенных формул видно, что полное основное сопротивление движению поезда складывается из сопротивления локомотива, сопротивления части состава, состоящей из вагонов 1-го типа, и части состава, состоящей из вагонов 2-го типа, а именно:

$W_o = \alpha \cdot w_o' \cdot m_л \cdot g + \alpha_1 \cdot w_{o1}'' \cdot m_c \cdot g + \alpha_2 \cdot w_{o2}'' \cdot m_c \cdot g$, где принять $\alpha = 1$.

Каждую из этих 3-х составляющих полного основного сопротивления движению подвижного состава можно определить по формуле общего вида: $W = \alpha \cdot w \cdot m \cdot g$, где величина w, в свою очередь, может быть представлена универсальной формулой вида: $w = a + (b + c \cdot V + d \cdot V^2)/e$. Тогда полное основное сопротивление движению поезда $W_o = \sum_{i=1}^{3} W_i$. Благодаря такому подходу появляется возможность применения функции DEF FN для расчета указанных величин. Сведем исходные данные в таблице 2.1.1.

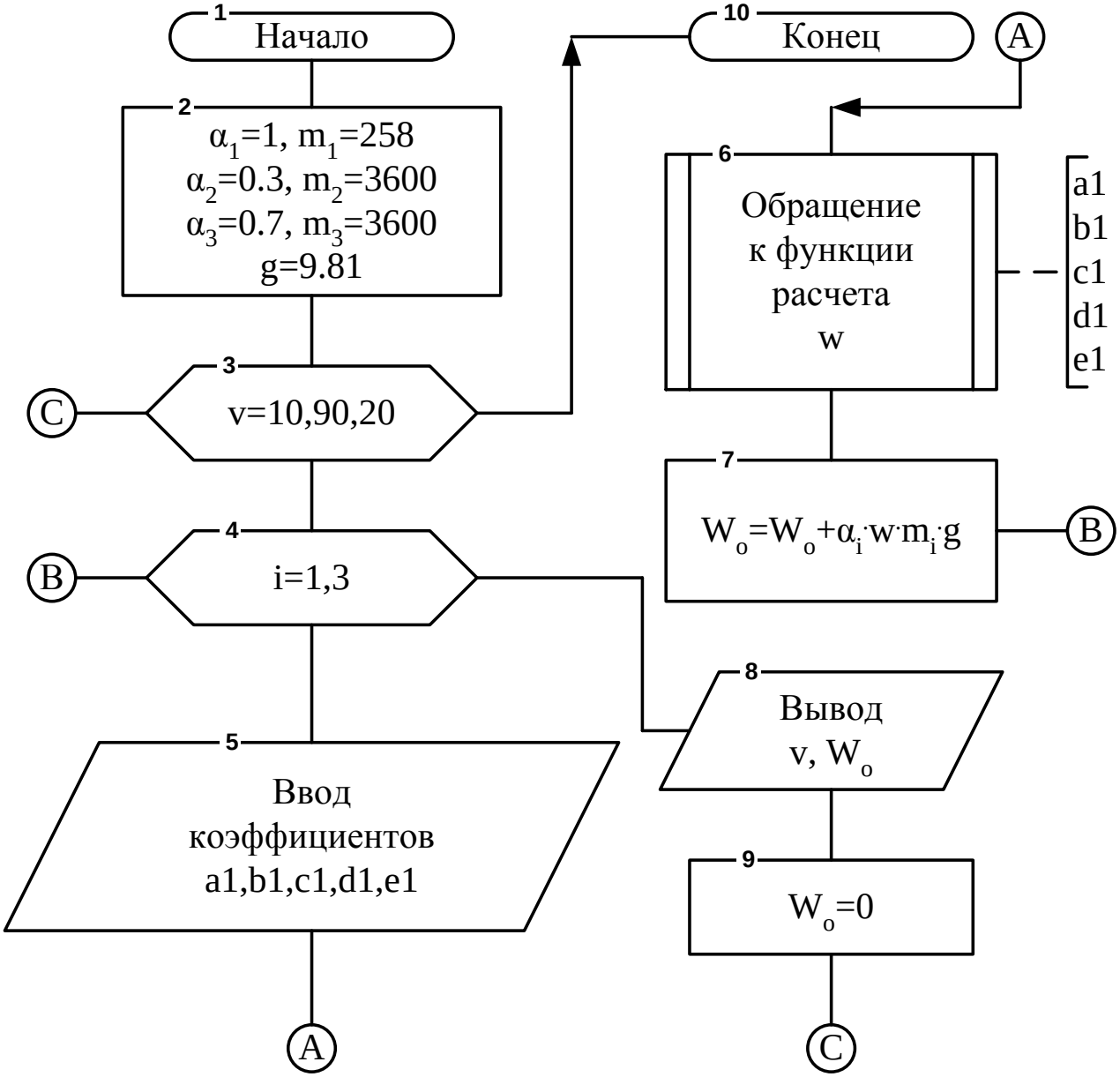
Таблица 2.1.1 - Данные для расчета w, W

Параметры / Части поезда	a	b	c	d	e	α_i	m_i	i
Тепловоз	0	1,9	0,010	0,0003	1	1	258	1
Вагоны 1 типа	0,7	3,0	0,100	0,0025	20	0,3	3600	2
Вагоны 2 типа	0,7	6,0	0,038	0,0021	18	0,7	3600	3

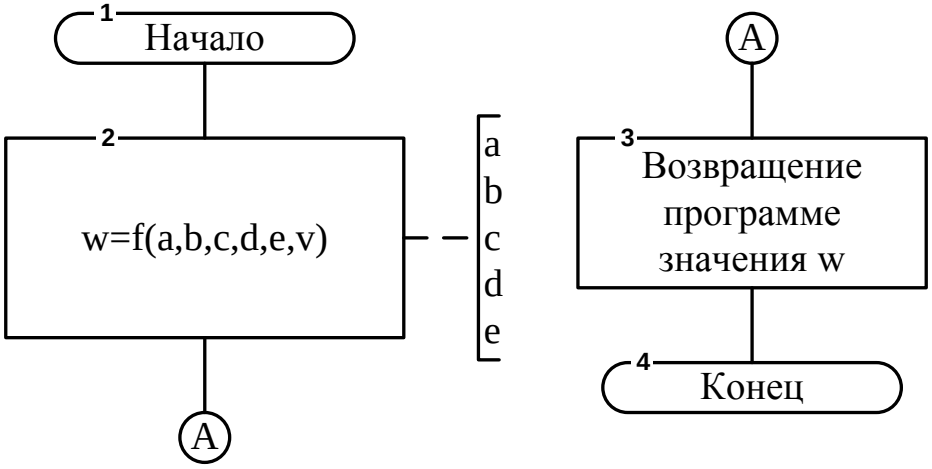
В этом подразделе используем определение функции DEF FN с именем, например, w (для расчёта величин w), в которой коэффициенты a, b, c, d, e – формальные параметры. Им в программе соответствуют фактические параметры, обозначенные, например, a1, b1, c1, d1, e1.

2.1.3 Блок-схема алгоритма решения

а) основная программа



б) функция расчёта w



2.1.4 Таблица соответствия

Математические переменные	V	$m_{во}$	$m_{л}$	m_{c}	g	w_o', w_{o1}'', w_{o2}'',	α_i	W_o
Машинные переменные	V	e, e1	M(1)	M(2) M(3)	g	w	ALF(i)	Wo

2.1.5 Программа на ТВ

```
CLS
DEF FNw(a, b, c, d, e) = a+(b+c*V+d*V^2)/e
ALF(1)=1: ALF(2)=0.3: ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
DATA 0, 1.9, 0.01, 0.0003, 1
DATA 0.7, 3, 0.1, 0.0025, 20
DATA 0.7, 6, 0.038, 0.0021, 18
FOR V=10 to 90 step 20
FOR i=1 to 3
READ a1, b1, c1 ,d1 ,e1
Wo=Wo+ALF(i)*FNw(a1, b1, c1, d1, e1)*M(i)*g
NEXT i
PRINT "V="; V,
PRINT "Wo=";: PRINT USING "#####"; Wo
Wo=0: RESTORE
NEXT V
```

2.1.6 Выполнение программы

RUN ENTER

```
V=10        Wo=41161
V=30        Wo=47745
V=50        Wo=58303
```

V=70 Wo=72835
V=90 Wo=91341

2.2 Пример использования первого варианта блочной конструкции DEF FN

Рассмотрим предыдущую задачу, только исходные данные по коэффициентам a, b, c, d, e введём с клавиатуры в окно Edit и запишем на носитель.

Путь к файлу: E:\5k.dat. Вид файла:

0, 1.9, 0.01, 0.0003, 1
0.7, 3, 0.1, 0.0025, 20
0.7, 6, 0.038, 0.0021, 18

Теперь вместо операторов READ – DATA (см. 2.1.5) для ввода коэффициентов в ОП ПК можно использовать созданный файл.

Программа на ТВ:

```
CLS
INPUT "ПУТЬ К ФАЙЛУ" R$
ALF(1)=1: ALF(2)=0.3: ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
FOR V=10 to 90 step 20
OPEN "I", #1, R$
FOR i=1 to 3
INPUT #1, a1, b1, c1, d1, e1
Wo=Wo+ALF(i)*FNw(a1, b1, c1, d1, e1)*M(i)*g
NEXT i
PRINT "V=";V,
PRINT "Wo=";
PRINT USING "#####"; Wo
Wo=0: CLOSE #1
NEXT V
END
DEF FNw(a, b, c, d, e)
FNw=a+(b+c*V+d*V^2)/e
END DEF
```

Выполнение программы:

RUN ENTER
ПУТЬ К ФАЙЛУ? E:\5k.dat ENTER
V=10 Wo=41161
V=30 Wo=47745
V=50 Wo=58303
V=70 Wo=72835
V=90 Wo=91341

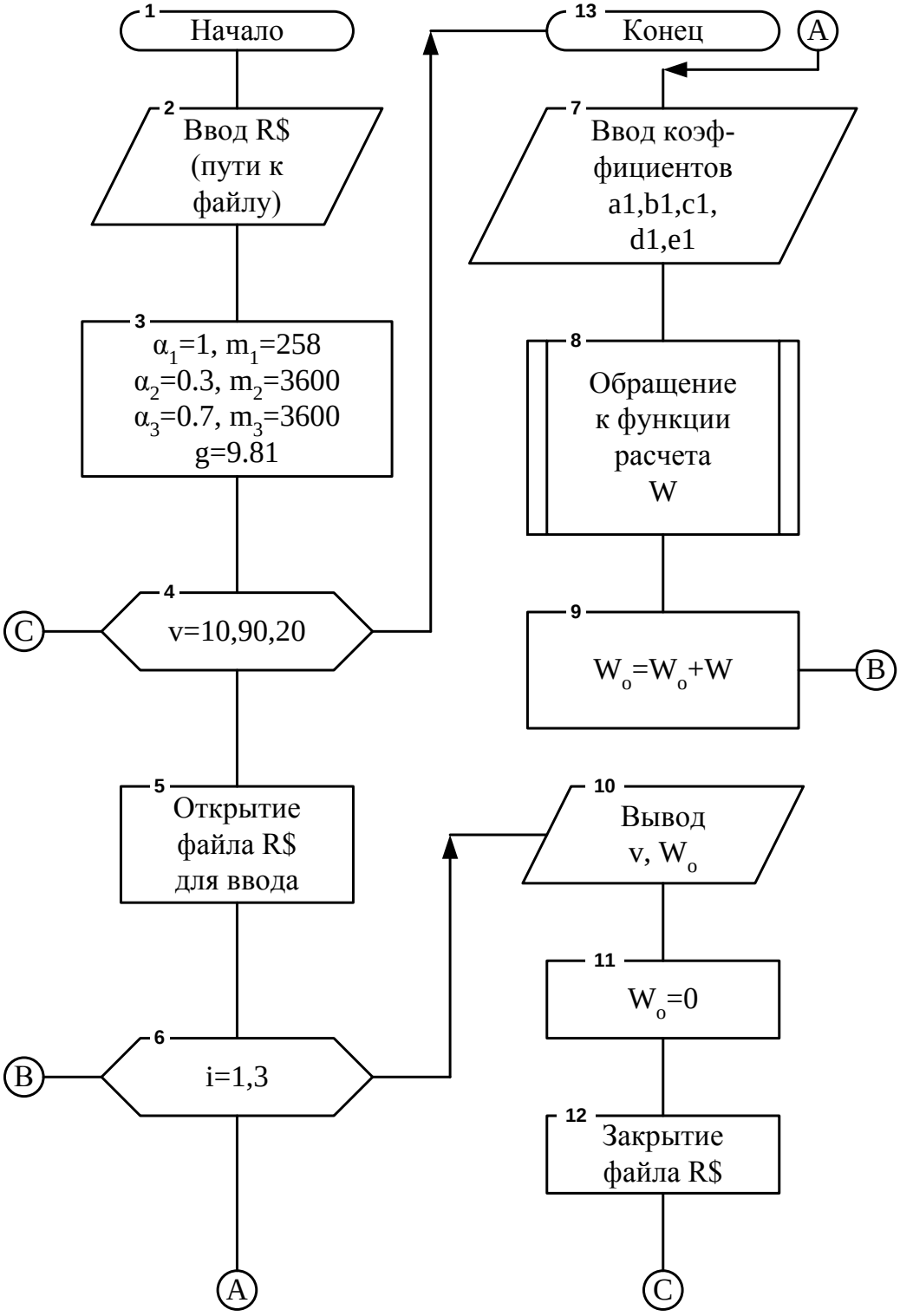
2.3 Пример использования однострочной и второго варианта блочной конструкции DEF FN

В качестве однострочной используем функцию DEF FNw (см. 2.1), а в блочной конструкции второго варианта произведем расчет сил сопро-

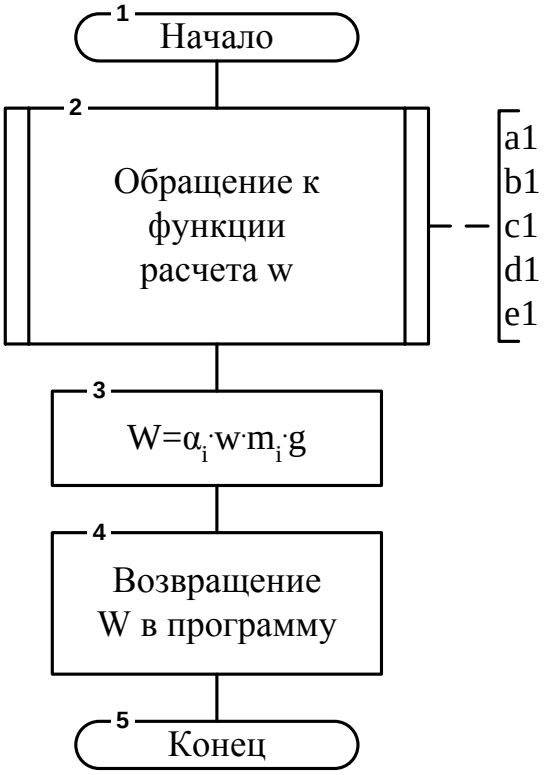
тивления движению 3-х частей поезда (см. 2.1.2) по общей формуле $W=\alpha \cdot w \cdot m \cdot g$. Обозначим в программе функцию, рассчитывающую эти силы, например, как DEF FNWO.

2.3.1 Блок-схема алгоритма решения

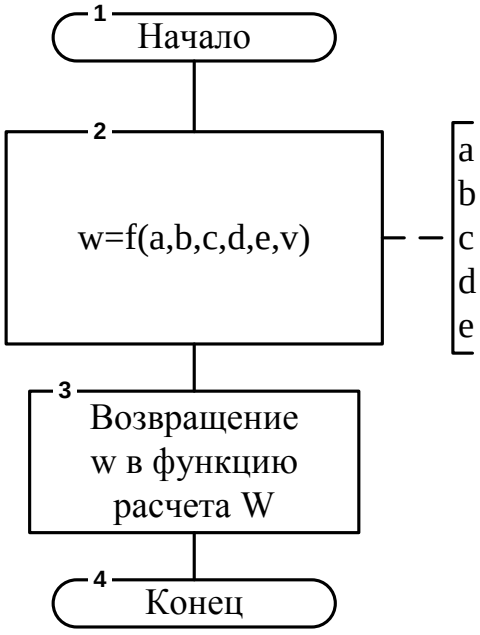
а) основная программа



б) функция расчёта W



в) функция расчёта w



2.3.2 Программа на ТВ

```
CLS
DEF FNw(a, b, c, d, e) = a+(b+c*V+d*V^2)/e
INPUT "ПУТЬ К ФАЙЛУ" R$
ALF(1)=1: ALF(2)=0.3: ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
FOR V=10 to 90 step 20
OPEN "I", #1, R$
FOR i=1 to 3
INPUT #1, a1, b1, c1, d1, e1
Wo=Wo+FNWO
NEXT i
PRINT "V=";V,
PRINT "Wo=";
PRINT USING "#####"; Wo
Wo=0: CLOSE #1
NEXT V
END
```

Выполнение программы:

RUN ENTER

ПУТЬ К ФАЙЛУ? E:\5k.dat ENTER

V=10	Wo=41161
V=30	Wo=47745
V=50	Wo=58303
V=70	Wo=72835
V=90	Wo=91341

```
DEF FNWO
FNWO=ALF(i)*FNw(a1, b1, c1, d1, e1)*M(i)*g
END DEF
```

Из приведенных примеров видно, что варианты использования функции DEF FN могут быть разные (студенты выбирают их по своему усмотрению), а результат получается один и тот же. Что касается операторов LOCAL, STATIC, SHARED (см. 2), то их применение студенты могут проработать самостоятельно.

3 ПРОЦЕДУРА

Процедура - самое мощное средство для построения модульной программы. Ограничена она операторами SUB (SUBroutine - процедура) и END SUB. Между ними размещаются операторы ТВ. Для выхода из процедуры до ее окончания служит оператор EXIT SUB. Общий вид процедуры - следующий:

```
SUB имя [(список формальных параметров)]
[LOCAL список переменных]
[STATIC список переменных]
[SHARED список переменных]
операторы ТВ
END SUB
```

Для вызова процедуры из программы используется оператор CALL (вызывать):

```
CALL имя [(список фактических параметров)]
```

Имя в операторах SUB и CALL должно быть одинаковым и может иметь длину до 31 символа. Из приведенного выше видно, что между процедурой и функцией есть много общего: они допускают вызов других процедур и функций, самих себя (рекурсию), передачу параметров, доступ к локальным, статическим и глобальным переменным. Однако есть и существенные различия. В процедуре все переменные по умолчанию являются статическими (STATIC, см. 2). Функции передаются параметры по значению и возвращается только один результат (через ее имя). Процедуре могут передаваться параметры (не более 16) как по значению, так и по ссылке

(т.е. по адресу). Возвращение параметров в процедуре осуществляется, естественно, только по ссылке (благодаря постоянству адреса параметра, в то время как содержимое параметра в процессе вычислений может изменяться).

Итак, процедура вызывается из программы по имени оператором CALL; значения фактических параметров присваиваются формальным; выполняются операторы в теле процедуры; по операторам END SUB или EXIT SUB происходит выход из процедуры, при этом значения формальных параметров присваиваются фактическим (при передаче по ссылке) и управление передается оператору, следующему за CALL. Если в списке фактических параметров переменная указана в скобках, это означает передачу по значению (как передаются константы и выражения). В этом случае при возвращении из процедуры сохраняется первоначальное значение переменной.

Особо следует сказать о массивах: если формальным параметром в процедуре SUB является массив, то в списке он должен иметь имя, а в скобках после имени указывается его размерность; в операторе CALL соответствующий этому массиву фактический параметр должен содержать имя массива и после него пустые скобки. Весь массив передается процедуре и обратно по ссылке.

3.1 Пример использования процедуры с передачей простых переменных

Условие задачи, метод решения те же, что и в 2.1÷2.2. Для показа работы процедуры используем программу, приведённую в 2.2, только обращение к функции заменим обращением к процедуре с именем, например, ww. В списке параметров кроме коэффициентов добавляется ещё один параметр, обозначенный как w. Его значение, вычисленное в процедуре по универсальной формуле как формальное, передаётся обратно фактическому параметру w в операторе CALL. В следующем после CALL операторе присваивания значение параметра w используется для расчёта сил сопротивления движению частей поезда W, их сложения и получения полного основного сопротивления движению поезда W_o. Область действия переменной v в процедуре представлена как глобальная, так как скорость v в списках передачи параметров не числится.

Программа на TB:

```
CLS
INPUT "ПУТЬ К ФАЙЛУ" R$
ALF(1)=1: ALF(2)=0.3: ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
FOR V=10 to 90 step 20
OPEN "I", #1, R$
FOR i=1 to 3
INPUT #1, a1, b1, c1, d1, e1
CALL ww (a1, b1, c1, d1, e1, w)
Wo=Wo+ALF(i)*w*M(i)*g
NEXT i
PRINT "V=";V,
PRINT "Wo=";
PRINT USING "#####"; Wo
Wo=0: CLOSE #1
NEXT V
END
SUB ww (a, b, c, d, e, w)
SHARED V
w=a+(b+c*V+d*V^2)/e
END SUB
```

Выполнение программы: RUN ENTER

ПУТЬ К ФАЙЛУ?E:\5k.dat ENTER

V=10 Wo=41161

V=30 Wo=47745

V=50 Wo=58303

V=70 Wo=72835

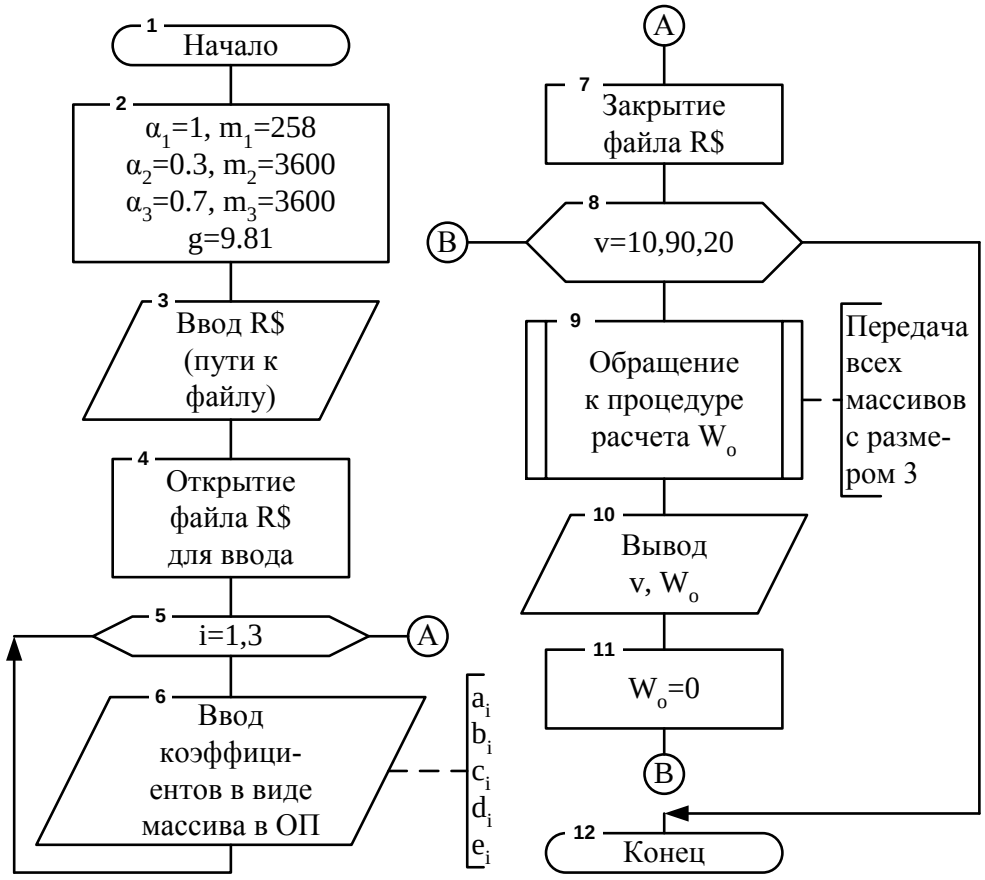
V=90 Wo=91341

3.2 Пример использования процедуры с передачей массивов

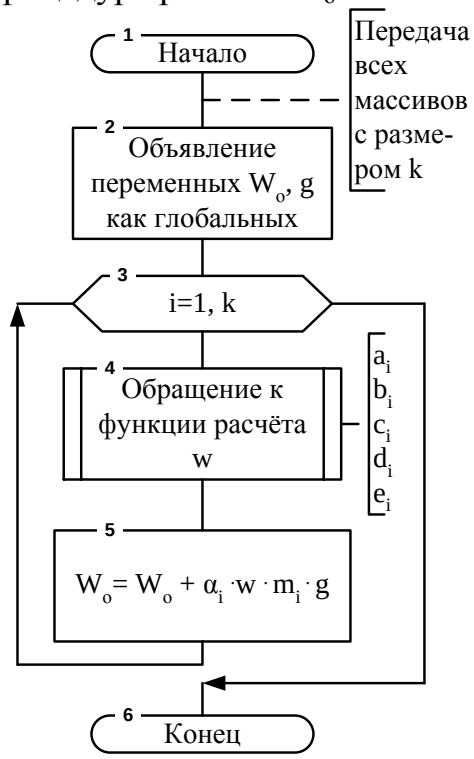
В основной программе коэффициенты, записанные в файл "E:\5k.dat" и необходимые для расчёта удельных сил основного сопротивления частей поезда w, вводятся в оперативную память только один раз, вначале (перед циклами расчётов) и запоминаются как массивы. В основной программе есть обращение к процедуре с передачей туда всех имеющихся массивов. Внутри процедуры создаётся цикл расчёта W_o с обращением к функции расчёта w. Более подробно эти процессы изображает блок-схема алгоритма решения и соответствующая ей программа (см. ниже).

3.2.1 Блок-схема алгоритма решения

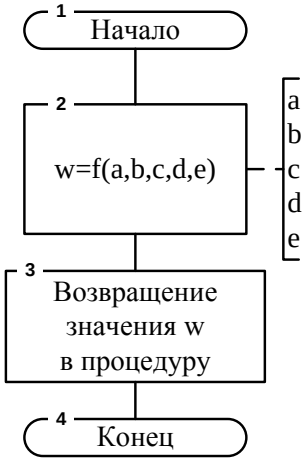
а) основная программа



б) процедура расчёта W_o



в) функция расчёта w



3.2.2 Программа на ТВ

```
CLS
DEF FNw(a, b, c, d, e)=a+(b+c*V+d*V^2)/e
ALF(1)=1: ALF(2)=0.3: ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
INPUT "ПУТЬ К ФАЙЛУ" R$
OPEN "I", #1, R$
FOR i=1 to 3
INPUT #1, a(i), b(i), c(i), d(i), e(i)
NEXT I: CLOSE #1
FOR V=10 to 90 step 20
CALL ww (ALF( ), M( ), a( ), b( ), c( ), d( ), e( ),3)
PRINT "V=";V,
PRINT "Wo=";: PRINT USING "#####"; WO
WO=0
NEXT V
END
SUB ww (ALF(1), M(1), a(1), b(1), c(1), d(1), e(1),k)
SHARED g, WO
FOR i=1 to k
WO=WO + ALF(i)*FNw(a(i), b(i), c(i), d(i), e(i))*M(i)*g
NEXT i
END SUB
```

Выполнение программы: RUN ENTER

ПУТЬ К ФАЙЛУ? A:\5k.dat ENTER

```
V=10    Wo=41161
V=30    Wo=47745
V=50    Wo=58303
V=70    Wo=72835
V=90    Wo=91341
```

4 ПОДПРОГРАММА

Подпрограммы - самый традиционный способ разделения программы на отдельные части. Они, однако, не обладают преимуществами функций и процедур, связанными с передачей параметров, использованием собственных переменных и другими возможностями. Общий вид подпрограммы - следующий:

метка (или номер строки)
операторы ТВ
RETURN

Вызов подпрограммы (переход к метке) осуществляется с помощью оператора GOSUB: GOSUB метка (или номер строки). Подпрограмма выполняется, пока не встретится оператор RETURN, после чего происходит возврат в программу к оператору, стоящему за GOSUB. Можно применять вложенные подпрограммы. Число вложений ограничено только стековой памятью, в которой хранятся адреса возврата из подпрограмм.

Следует отметить, что все переменные подпрограммы являются глобальными (имеют тип памяти SHARED). Подпрограммы не обладают рекурсивностью.

4.1 Пример использования подпрограммы с простыми переменными

Условие задачи по прежнему то же, что и в 2.1. Программу для ее решения разделим на две части: основную программу и подпрограмму. Установим назначение каждой части. Назначение основной программы: ввод исходных данных, организация цикла изменения скорости движения поезда и вывода на печать значений скорости и полного сопротивления движению, а назначение подпрограммы: открытие файла "E:\5k.dat" коэффициентов (см. 2.2), организация цикла по вводу коэффициентов в ОП ПК и расчёту удельных w и полных W сил основного сопротивления движению частей поезда (см. 2.1.2), закрытие файла и передача управления в основную программу.

Программа на ТВ:

```
CLS
INPUT "ПУТЬ К ФАЙЛУ" R$
ALF(1)=1:ALF(2)=0.3:ALF(3)=0.7
M(1)=258: M(2)=3600: M(3)=3600: g=9.81
FOR V=10 to 90 step 20
GOSUB M
PRINT "V=";V,: PRINT "Wo=";
PRINT USING "#####"; Wo
Wo=0: NEXT V
END
M:
OPEN "I", #1, R$
FOR i=1 to 3
INPUT #1, a, b, c, d, e
w=a+(b+c*V+d*V^2)/e
Wo=Wo+ALF(i)*w*M(i)*g
NEXT i: CLOSE #1
RETURN
```

Выполнение программы: RUN ENTER

ПУТЬ К ФАЙЛУ?E:\5k.dat ENTER

V=10 Wo=41161

V=30 Wo=47745

V=50 Wo=58303

V=70 Wo=72835

V=90 Wo=91341

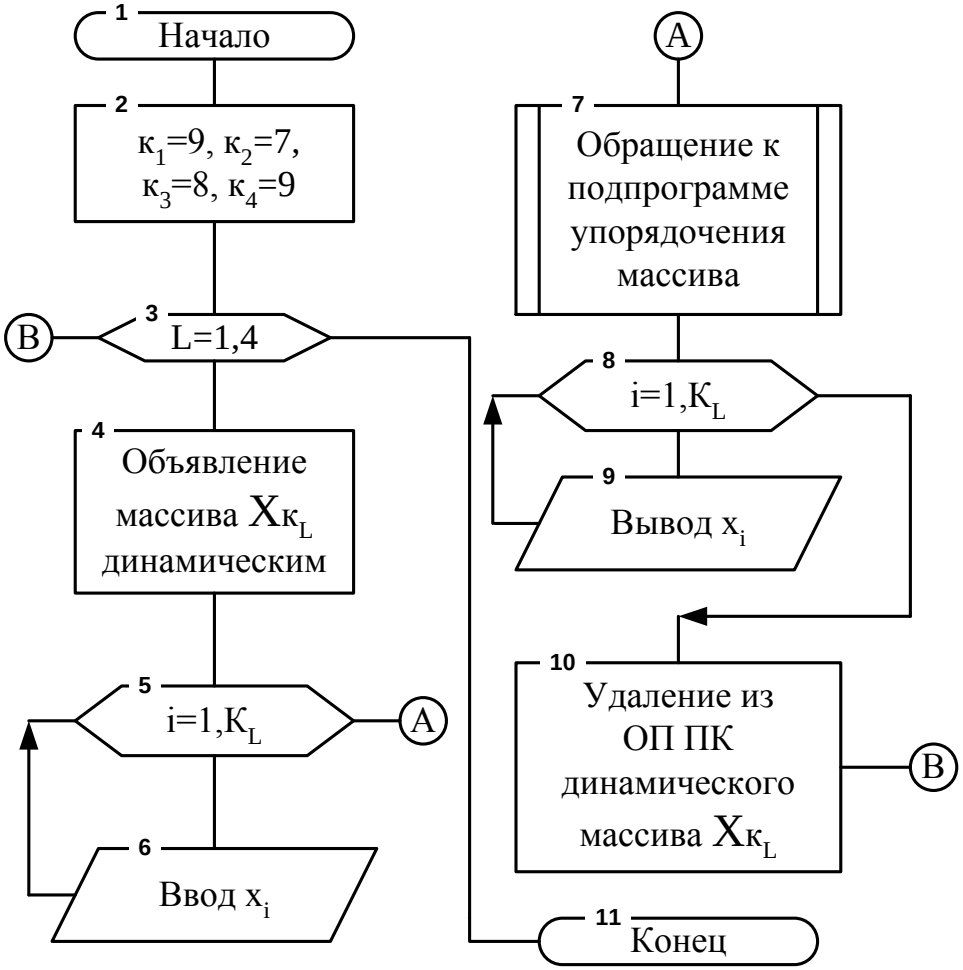
4.2 Пример использования подпрограммы упорядочения одномерного массива

Очень часто результаты расчетов представляют неупорядоченный массив, который не пригоден для построения графиков. Требуется составить подпрограмму, назначение которой: расположить все элементы неупорядоченного массива в порядке возрастания. Назначение основной программы: ввести исходный массив как динамический, обратиться к подпрограмме для его обработки и распечатать упорядоченный по возрастанию массив. В качестве неупорядоченных массивов принимаем массивы X_i, приведенные в таблице 1.3.1.

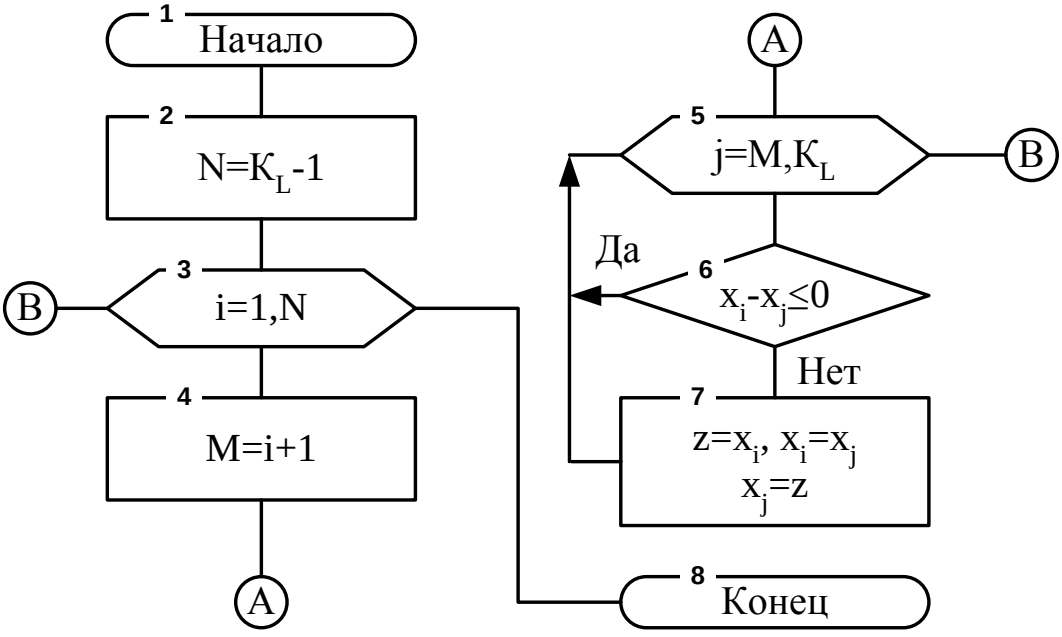
Обозначим через K_j массив, содержащий в качестве элементов размер массива X_i по каждому j-ому варианту таблицы, т.е. $K_1=9$, $K_2=7$, $K_3=8$, $K_4=9$. Массив X_i будем объявлять для обработки и удалять после использования из оперативной памяти ПК как динамический.

4.2.1 Блок-схема алгоритма решения

а) основная программа



б) подпрограмма упорядочения массива



4.2.2 Программа на TB

```
CLS
K(1)=9: K(2)=7: K(3)=8: K(4)=9
DATA 76.8, 76.1, 76.8, 77.4, 80.2, 81.4, 82.8, 82.7, 87.6
DATA  15, 17, 19, 21, 20, 18, 20
DATA  95, 99, 94, 101, 97, 95, 96, 98
DATA 1.75, 1.87, 1.49, 1.78, 1.70, 1.73, 1.80, 1.72, 1.81
C1$="###.##&": C2$=","
FOR L=1 TO 4: DIM X(K(L))
FOR I=1 TO K(L): READ X(I): NEXT I
GOSUB UM
FOR I=1 TO K(L)
IF I=K(L) THEN PRINT USING C1$; X(I);: EXIT FOR
PRINT USING C1$; X(I), C2$;: NEXT I: PRINT
ERASE X: NEXT L
END
UM:
N=K(L)-1
FOR I=1 TO N: M=I+1
FOR J=M TO K(L): IF (X(I)-X(J))<=0 THEN M
Z=X(I): X(I)=X(J): X(J)=Z
M:
NEXT J, I
RETURN
```

Результат выполнения программы и подпрограммы:

```
76.10, 76.80, 76.80, 77.40, 80.20, 81.40, 82.70, 82.80, 87.60
15.00, 17.00, 18.00, 19.00, 20.00, 20.00, 21.00
94.00, 95.00, 95.00, 96.00, 97.00, 98.00, 99.00,101.00
1.49,  1.70,  1.72,  1.73,  1.75,  1.78,  1.80,  1.81,  1.87
```

5 ОСНОВНЫЕ ОПЕРАТОРЫ ГРАФИКИ

В языке ТВ используются, как правило, те же операторы графики, которые присущи современным диалектам Бейсика. При выполнении расчетов и исследовании зависимостей физических величин в виде графиков достаточно применять операторы, приведенные ниже:

1. Переключение режимов работы графического дисплея:

SCREEN [режим][,цвет]

а) значения режима: 0 - текстовый; 1 - графический с размером экрана (320 × 200) при выводе графической информации и (25 × 40) при выводе текстовой информации; 2 - графический с размером экрана (640 × 200) и (25 × 80) соответственно или 12- 640 × 480 точек; при смене режимов работы происходит очистка экрана;

б) значение цвета: 0 - в текстовом режиме отключает цветное изображение, а в графическом режиме включает его.

2. Использование ограниченной области экрана (окон) для обработки информации в графическом режиме:

VIEW [(x_1,y_1)-(x_2,y_2)][,[цвет][,контур]]]

а) выражение (x_1,y_1)-(x_2,y_2) задает координаты левого верхнего и правого нижнего углов окна соответственно;

б) значение цвета окна, контура при разрешающей способности экрана (640 × 200): 0 - черный цвет окна, контура; 1 - белый.

Оператор VIEW без операндов определяет окно как весь экран дисплея.

3. Переопределение координат окна (использование масштабов):

WINDOW (x_1,y_1)-(x_2,y_2)

Выражение (x_1,y_1)-(x_2,y_2) задает новый диапазон изменения координат в пределах окна (новую систему координат). Оператор WINDOW (x_1,y_1)-(x_2,y_2) эквивалентен WINDOW (x_1,y_2)-(x_2,y_1). Точка с наименьшими значениями координат располагается в левом нижнем углу окна.

4. Вывод точки:

PSET (x,y)[,цвет]

Выражение (x,y) задает координаты точки, которую необходимо вывести на экран (окно) дисплея.

5. Проведение отрезка прямой:

LINE [(x_1,y_1)]-(x_2,y_2)[,цвет]

Выражение (x_1,y_1)-(x_2,y_2) задает координаты начальной и конечной точек на экране (окне) дисплея. Если координаты начальной точки (x_1,y_1) отсутствуют, используются координаты последней выведенной точки.

Для перемещения курсора на экране при выводе текстовой информации в графическом режиме можно воспользоваться оператором

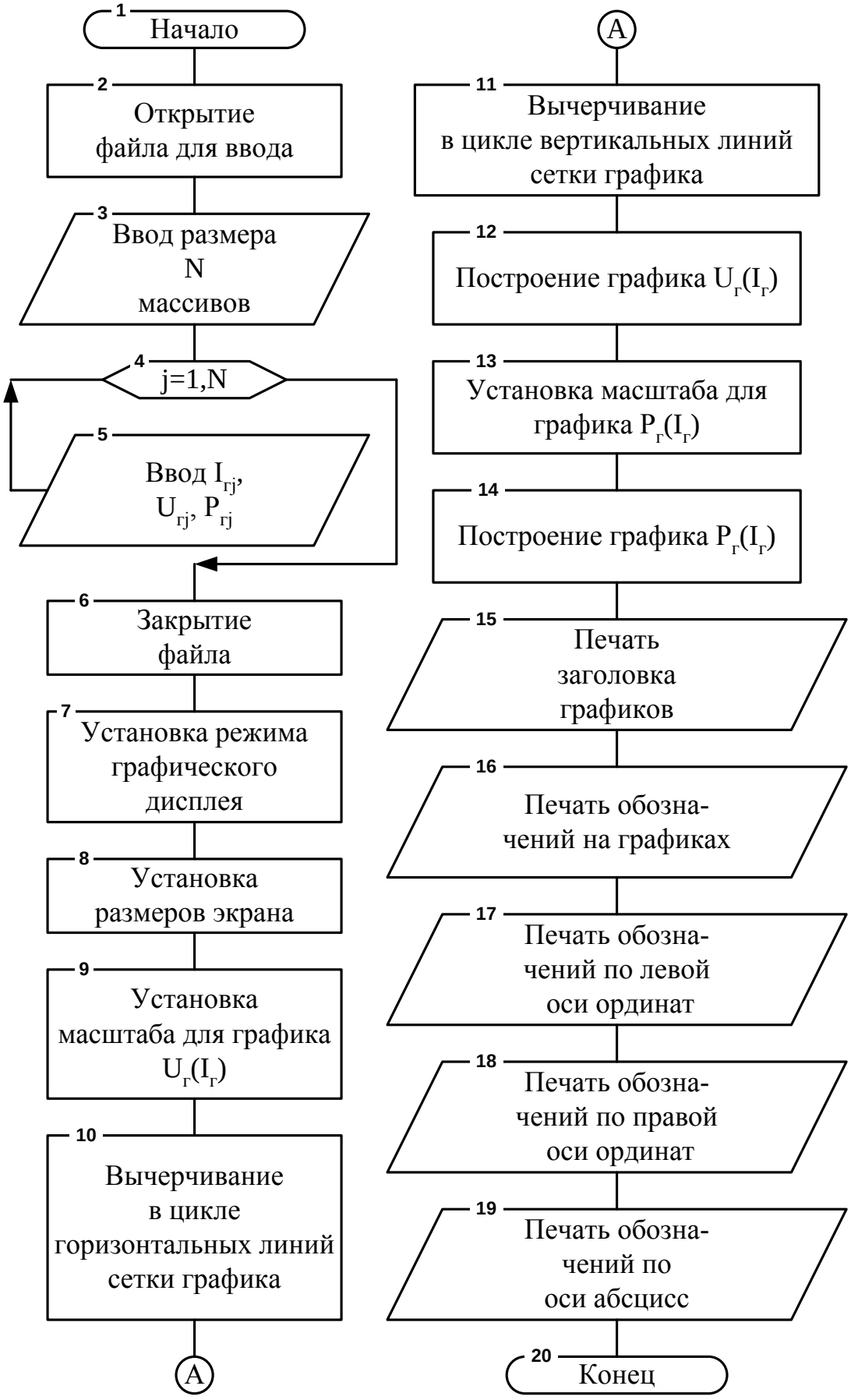
LOCATE [номер строки][,номер столбца]

5.1 Пример использования операторов графики

Построим графики $U_г=f_1(I_г)$ и $P_г=f_2(I_г)$ по данным, записанным в файл (см. 1.2.5). Допустим, что путь к файлу имеет вид: E:\GRAFIC\I-U-P.DAT. Напоминаем содержание файла:

а) в обозначениях переменных	б) в значениях переменных
N	6
$I_{г1}$, $U_{г1}$, $P_{г1}$	0, 750, 0
$I_{г2}$, $U_{г2}$, $P_{г2}$	2600, 700, 1820
.....	3900, 454, 1770
$I_{гN}$, $U_{гN}$, $P_{гN}$	5200, 338, 1760
	6400, 280, 1792
	6600, 0, 0

5.1.1 Блок-схема алгоритма решения



5.1.2 Программа на ТВ

```
cls   'очистка экрана
OPEN"I", #1, "E:\GRAFIC\I-U-P.DAT"
INPUT #1, N
FOR J=1 TO N
INPUT #1, I(J), U(J), P(J)
NEXT J
CLOSE #1
screen 2
view (100,40)-(495,165),0,1
window (0,0)-(7000,800)
for j=100 to 700 step 100
line (0,j)-(7000,j)
next j
for j=1000 to 6000 step 1000
line (j,0)-(j,800)
next j
for j=1 to n-1
line (I(j),U(j))-(I(j+1),U(j+1))
next j
window (0,0)-(7000,3200)
for j=1 to n-1
line (I(j),P(j))-(I(j+1),P(j+1))
next j
LOCATE 3,25: print "Графики функций U(I) и P(I)"
LOCATE 9,37: print "U"
LOCATE 15,24: print "P";: LOCATE,60: print "P"
U=700: P=3200: I=0: LOCATE 5,8: print "U, В"
for j=7 to 21 step 2
LOCATE j,8: print u: u=u-100
if j=9 then LOCATE j,65: print "P, кВт"
LOCATE j,64: p=p-400: if j>9 then print p
next j
for j=11 to 53 step 7
LOCATE 22,j: print i: i=i+1000
next j: LOCATE 22,60: print "I, A"
end
```

5.1.3 Выполнение программы

Исполнить команду Run, Enter. На экран выводится график: линии белого цвета на чёрном фоне. Чтобы вывести график на принтер, надо предварительно выполнить следующие операции:

1. Нажать клавишу Print Screen.
2. Выйти из среды ТВ командой Quit меню File.
3. Запустить графический редактор «Paint» (программа-рисовальщик):

а) применить команды Правка → Вставка, после чего рисунок (график) вставляется в окно редактора;

б) применить команды Рисунок → Обратить цвета, после чего чёрный фон графика поменяется на белый, а белые линии на чёрные;

в) применить команду Файл → Печать, после чего рисунок готов к выводу на принтер.

Можно не ограничиваться простым и популярным редактором «Paint», а воспользоваться более сложными пакетами программ, в том числе текстовым редактором «Word» (по усмотрению пользователя).

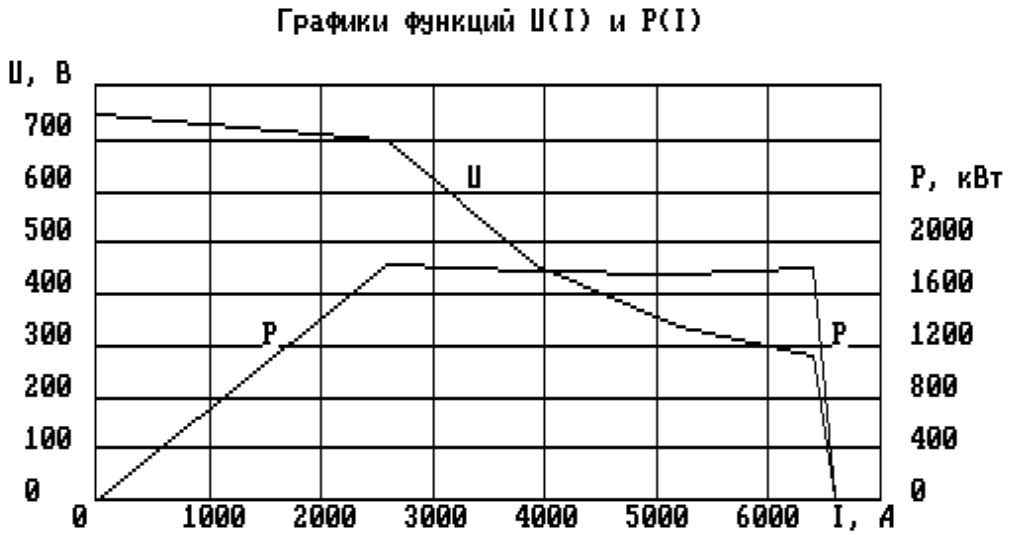


Рисунок 5.1 – График функций

6 СПИСОК СООБЩЕНИЙ ОБ ОШИБКАХ, ВОЗНИКАЕМЫХ ПРИ РАБОТЕ В СРЕДЕ TURBO BASIC

Ошибки во время выполнения

2. Синтаксическая ошибка

Обнаруживаемая во время выполнения синтаксическая ошибка в операторе READ, пытающемся прочитать символьные данные в числовую переменную.

3. RETURN без GOSUB

Выполнение оператора RETURN без соответствующего GOSUB, т.е. возвращающего управление в никуда.

4. Нет данных

Выход при выполнении оператора READ за пределы последовательности значений в DATA.

5. Неверный вызов функции

Все случаи несоответствий передаваемых аргументов некоторому оператору или функции. Возможные причины:

• слишком большой аргумент цвета или режима экрана;

• использование оператора графики или установка графического режима оператором SCREEN на компьютере без графического адаптера;

• попытка выполнить недопустимые математические операции, например, взятие квадратного корня из отрицательного числа;

• слишком большой (или отрицательный) номер записи в операторах GET и PUT;

• попытка использования оператора WIDTH для последовательного файла.

6. Переполнение

Получаемое в результате вычислений слишком большое для представления данным числовым типом значение. Например, i% = 32766+2 вызовет переполнение, поскольку число 32768 не может быть представлено как целое. Переполнение для целого типа не отлавливается, если программа откомпилирована с выключенным ключом Overflow в команде Options главного меню среды Турбо-Бейсик. Переполнение для вещественного типа обнаруживается всегда.

7. Не хватает памяти

Это сообщение порождается многими ситуациями, в том числе слишком большой размерностью массива или передачей слишком большого числа массивов в подпрограмму или функцию.

9. Индекс вне диапазона

Попытка использования индекса, большего чем максимальный номер элемента массива, объявленный в операторе DIM. Об этой ошибке не сообщается, если не включен ключ Bounds в команде Options главного меню Турбо-Бейсик.

10. Повторное определение

Попытка повторного определения динамического массива без удаления первого.

11. Деление на нуль

Попытка деления на нуль или возведения нуля в отрицательную степень.

13. Несоответствие типов

Использование символьной строки вместо числа или наоборот. Ошибка может оказаться в операторах PRINT USING, DRAW или PLAY.

14. Не хватает строкового пространства

Исчерпание пространства памяти (64К), предназначенного для хранения символьных строк.

15. Слишком длинная строка

Значение символьного выражения занимает более 32767 байт.

19. Отсутствие RESUME

Переход к физическому концу программы в подпрограмме обработки ошибки. Может возникнуть, если в обработке ошибки пропущен оператор RESUME.

20. RESUME без ошибки

Выполнение оператора RESUME без возникновения ошибки.

24. Таймаут устройства

Указанное время перерыва для связи истекло.

25. Сбой устройства

Аппаратная ошибка, например, сбой интерфейса принтера или адаптера связи.

27. Нет бумаги

Интерфейс принтера указывает на отсутствие бумаги, возможно также, что принтер просто выключен.

50. Переполнение поля

Попытка определить множество переменных полей буфера в операторе FIELD, превышающее по длине размер записи файла.

51. Внутренняя ошибка

Ошибка исполнительной программы Турбо-Бейсика.

52. Неправильный номер файла

Номер файла, указанный в операторе доступа к файлу, не соответствует номеру, данному в операторе OPEN, или вне допустимого диапазона номеров.

53. Файл не найден

Указанное имя файла не найдено на указанном диске.

54. Неправильный доступ к файлу

Использование операторов PUT или GET (PUT$ или GET$) для доступа к последовательному файлу.

55. Файл уже открыт

Попытка открыть уже открытый файл или удалить открытый файл.

57. Ошибка устройства ввода/вывода

Серьезная аппаратная ошибка при выполнении оператора ввода/вывода.

58. Файл уже существует

Попытка дать файлу оператором NAME уже существующее имя.

61. Диск полон

Недостаточно дискового пространства или сбой диска при выполнении файловой операции.

62. Пройденный конец ввода

Попытка читать файл за его границей (используйте функцию EOF для преодоления этой проблемы), или попытка читать файл, открытый для вывода.

63. Неправильный номер записи

Отрицательный или превышающий 16777215 номер записи в операторах PUT или GET доступа к файлу.

64. Неправильное имя файла

Имя файла, указанное в операторах FILES, KILL или NAME, содержит неверные символы.

67. Слишком много файлов

Недопустимое имя файла, попытка создать слишком много файлов в корневом каталоге диска, что влияет на производительность файловой системы DOS.

68. Устройство не готово

Попытка открыть файл на устройстве, которого нет.

69. Переполнение буфера связи

Выполнение оператора INPUT ввода символов в уже полный буфер связи.

70. Отказ в разрешении

Попытка записи на защищенный от записи диск.

71. Диск не готов

Дисковое устройство не закрыто или отсутствует дискета в нем.

72. Ошибка дисковой среды

Контроллер дисковода обнаруживает сбойные сектора на диске.

74. Переименование файла через диск

Попытка переименовать файл с переносом.

75. Ошибка доступа к пути/файлу

Неправильное использование пути в операторах OPEN, RENAME, MKDIR; например, попытка открыть оператором OPEN каталог.

76. Путь не найден

Указанный в операторах CHDIR, MKDIR, OPEN и т.д. путь не найден.

201. Не хватает стекового пространства

Переполнение стека (можно увеличить стековое пространство метаоператором $STACK или сократить количество рекурсивных вызовов процедур и функций).

202. Не хватает временного строкового пространства

Символьное выражение требует слишком большого пространства во временном буфере для символьных строк. Может вызываться выражениями с вложенными вызовами символьных функций, например: MID$(LEFT$ (MID$(RIGHT$(...). (В таких случаях можно упрощать символьные выражения, разбивая их на несколько выражений.)

203. Несоответствие общих переменных

Использование оператора CHAIN в двух сегментах программы с несоответствующими операторами COMMON. Проверяются типы и число переменных в операторах COMMON.

204. Несоответствие установок в параметрах

Использование оператора CHAIN в двух программных сегментах, откомпилированных с различными установками параметров среды (разные математические библиотеки, различия в требовании сопроцессора и т.д.).

205. Несоответствие в версиях программы

Использование оператора CHAIN в двух программных сегментах, созданных с помощью различных версий компилятора Турбо-Бейсик.

206. Недопустимый программный файл

Использование операторов CHAIN или RUN для программных сегментов, которые не были откомпилированы в Турбо-Бейсике.

242. Порча памяти для строк или массивов

Неправильная перезапись области строковой памяти; причиной может быть неправильное действие ассемблерной подпрограммы, доступ за пределы символьного массива или ошибка исполнительной программы Турбо-Бейсика.

243. CHAIN/RUN только из .EXE файлов

Использование операторов CHAIN или RUN из среды Турбо-Бейсика. Необходимо сначала откомпилировать программу на диск с использованием установки Compile to команды Options главного меню, а затем вызвать программу, применяющую CHAIN/RUN из DOS.

Ошибки, обнаруживаемые компилятором

401. Выражение слишком сложное

Выражение содержит слишком много операторов/операндов; необходимо разбить его на два или более простых выражений.

402. Оператор слишком сложный

Сложность оператора вызывает переполнение внутреннего буфера компилятора; необходимо разбить оператор на два или более простых операторов.

403. Переполнение вложений $IF

Блоки условной компиляции ($IF/$ELSE/$ENDIF) могут иметь глубину вложения до 16 уровней.

404. Переполнение вложений $INCLUDE

Включенные файлы могут иметь глубину вложения до 5 уровней, включая саму главную программу. Таким образом, программа может иметь вложенные включенные файлы с глубиной вложения 4.

405. Переполнение вложения блоков

В программе слишком много структурных блоков, вложенных друг в друга. Блочные структуры в Турбо-Бейсике могут иметь глубину вложения до 255.

406. Компилятору не хватает памяти

Доступная компилятору память для хранения символьных строк, буферов и т.п. исчерпана. В этом случае нужно разделить программу на меньшую главную программу, имеющую метаоператор $INCLUDE, подключающий остаток программы. Если эта ошибка возникнет при компиляции в память, нужно попытаться откомпилировать ее на диск.

407. Программа слишком большая

Программа содержит больше чем 65530 операторов.

408. Сегмент превышает 64К

Программа содержит сегмент, превышающий ограничение 64К. Включите оператор $INCLUDE для помещения программного кода в другой сегмент.

409. Переменные превышают 64К

Общее пространство для скалярных переменных ограничено 64К. Сюда включаются указатели на символьные строки, целые, длинные целые, вещественные и числа двойной точности. Необходимо избавиться от неиспользуемых переменных или разбить программу на главную и подключаемые (CHAIN).

410. Ожидается ","

Синтаксисом оператора требуется запятая (,).

411. Ожидается";"

Синтаксисом оператора требуется точка с запятой (;).

412. Ожидается "("

Синтаксисом оператора требуется левая скобка (().

413. Ожидается ")"

Синтаксисом оператора требуется правая скобка ()).

414. Ожидается "="

Синтаксисом оператора требуется знак равенства (=).

415. Ожидается "-"

Синтаксисом оператора требуется тире (-).

416. Ожидается оператор

Ожидается оператор Турбо-Бейсика. Встретилось то, что не идентифицируется ни как команда, ни как метаоператор, ни как переменная.

417. Ожидается метка/номер строки

В операторе IF, GOTO, GOSUB или ON ожидается допустимая метка или номер строки.

418. В числовом выражении требуется оператор сравнения

В месте, где должна быть числовая операция, компилятор обнаружил символьное выражение.

419. Символьное выражение требует символьного операнда

В символьном выражении встретился не символьный операнд. Например: x$=y$+n.

420. Ожидается скалярная переменная

В качестве формального параметра определяемой функции ожидается скалярная переменная: символьная строка, целое, длинное целое, вещественное или число двойной точности.

421. Ожидается массив

В операторе DIM или графических операторах GET, PUT ожидается массив.

422. Ожидается числовая переменная

В операторах INCR, DECR и в спецификации смещения CALL ABSOLUTE ожидается числовая переменная.

423. Ожидается символьная переменная

В операторах FIELD, GET$, PUT$ и LINE INPUT ожидается символьная переменная.

424. Ожидается переменная

В функциях VARPTR, VARPTR$ и VARSEG ожидается переменная.

425. Ожидается целая константа

В присваивании именованной константе или операторе условной компиляции $IF/$ELSEIF ожидается целая константа.

426. Ожидается целая положительная константа

В качестве границы массива в операторе DIM или в метаоператорах $COM, $SOUND и $STACK ожидается положительная целая константа.

427. Ожидается символьная константа

В качестве имени файла или в метаоператоре $INCLUDE ожидается символьная константа.

428. Ожидается числовая скалярная переменная

Ожидается скалярная переменная любого числового типа; например, в цикле FOR/NEXT.

429. Ожидается целая скалярная переменная

В качестве параметра оператора CALL ABSOLUTE ожидается целая переменная.

430. Ожидается массив целого типа

Ожидается массив целого типа; например, в операторе PALETTE.

431. Ожидается конец строки

Вслед за метаоператором, оператором END SUB или меткой оператора не должно быть никаких символов (кроме комментария).

432. Ожидается AS

В операторе FIELD или OPEN пропущено ключевое слово AS.

433. Ожидается DEF FN

Обнаружен оператор END FN или EXIT FN без определения функции. Оно должно начинаться с оператора DEF FN.

434. Ожидается IF

Обнаружены END IF или EXIT IF без соответствующего IF.

435. Ожидается DO LOOP

Обнаружены операторы LOOP или EXIT LOOP без начального оператора DO.

436. Ожидается SELECT

Либо в операторе SELECT CASE пропущено слово SELECT, либо компилятор попал на END SELECT или EXIT SELECT, обойдя начальные ключевые слова SELECT CASE. Это сообщение может быть выдано также, если была попытка использовать ключевое слово CASE в качестве имени переменной.

437. Ожидается CASE

В операторе SELECT CASE пропущено ключевое слово CASE. Это сообщение также может быть выдано, если была попытка использовать ключевое слово SELECT в качестве имени переменной.

438. Ожидается FOR LOOP

Обнаружен оператор EXIT FOR без начального ключевого слова FOR оператора цикла.

439. Ожидается SUB

Обнаружен оператор END SUB или EXIT SUB без определения процедуры. Оно должно начинаться с ключевого слова SUB.

440. Ожидается END DEF

Определение функции не завершено оператором END DEF.

441. Ожидается END IF

Блок IF не завершён соответствующим оператором END IF.

442. Ожидается LOOP/WEND

Цикл LOOP или WHILE не завершён соответствующими операторами LOOP или WEND.

443. Ожидается END SELECT

Оператор SELECT CASE корректно не завершён ключевыми словами END SELECT.

444. Ожидается END SUB

Определение процедуры корректно не завершено оператором END SUB.

445. Ожидается NEXT

Цикл FOR корректно не завершен оператором NEXT.

446. Ожидается THEN

В конструкции IF пропущена часть THEN.

447. Ожидается TO

В операторе FOR пропущена часть TO.

448. Ожидается GOSUB

В операторе ON пропущена часть GOSUB.

449. Ожидается GOTO

В операторе ON пропущена часть GOTO.

450. Ожидается $ENDIF

В метаоператоре $IF пропущено соответствующее ключевое слово $ENDIF. Необходимо проверить все метаоператоры $IF и выяснить, куда вставить соответствующее $ENDIF.

451. Несоответствующее $ELSE

Обнаружен метаоператор $ELSE, для которого пропущен предшествующий ему метаоператор условной компиляции $IF. Необходимо подняться от $ELSE вверх по исходному тексту и выяснить, куда вставить соответствующее $IF.

452. Несоответствующее $ENDIF

Обнаружен метаоператор $ENDIF, для которого пропущен предшествующий ему метаоператор условной компиляции $IF. Необходимо подняться от $ENDIF вверх по исходному тексту и выяснить, куда вставить соответствующее $IF.

453. Неопределенная именованная константа

Использована неопределенная именованная константа. Или определите ее, или используйте обычную константу.

454. Ссылка на неопределенную функцию

В выражении использовано имя несуществующей функции. Напишите определение функции или исправьте ошибку в имени функции.

455. Ссылка на неопределенную подпрограмму

В программе имеется вызов не определенной подпрограммы. Исправьте ошибку в имени подпрограммы или напишите ее определение.

456. Ссылка на отсутствующую метку/строку

В операторе IF, GOTO, GOSUB или ON использована метка или номер строки, отсутствующие в программе. Проверьте, нет ли ошибки в имени или номере строки или введите метку.

457. Ссылка на неопределенный массив

Используется массив, нигде не определенный оператором DIM. Проверьте, нет ли ошибки в имени массива или добавьте оператор DIM для этого массива.

458. Повторяющиеся метка/номер строки

Один и тот же номер строки или одна и та же метка использованы дважды. Проверьте текст программы и всех включенных файлов на повторяющиеся метки/номера строк и измените их.

459. Дублирование именованной константы

В определениях двух именованных констант использовано одно и то же имя. Проверьте текст программы и всех включаемых файлов на дублирующее определение именованных констант и измените их.

460. Повторное определение функции

Две функции, определенные оператором DEF FN, используют одно и то же имя. Проверьте текст программы и всех включаемых файлов на присутствие повторного определения и измените имя одной или обеих функций.

461. Повторное определение процедуры

В двух процедурах, определенных оператором SUB, использовано одно и то же имя. Проверьте текст программы и всех включаемых файлов на дублирование имени процедуры и измените одно из этих имен.

462. Дублирование общей переменной

В списке переменных в операторе COMMON дважды присутствует одно и то же имя. Проверьте текст программы и всех включаемых файлов на дублирование имени и измените имя одной из переменных.

463. Дублирование определения переменной

В операторе LOCAL, STATIC или SHARED объявлены две переменные с одним именем. Проверьте текст программы и всех включаемых файлов на присутствие дублирующих определений переменных и измените имя одной из них.

464. Дублирование определения $COM

Только один метаоператор $COM может использоваться для каждого коммуникационного порта. Проверьте текст программы и всех включаемых файлов на присутствие совпадающих определений $COM и измените одно из них.

465. Дублирование определения $SOUND

В программе может быть только один метаоператор $SOUND. Проверьте текст программы и всех включаемых файлов на присутствие метаоператоров $SOUND и оставьте только один из них.

466. Дублирование определения $STACK

В программе может быть только один метаоператор $STACK. Проверьте текст программы и всех включаемых файлов на присутствие метаоператоров **$**STACK и оставьте только один из них.

467. Неверный номер строки

Номер строки должен быть в диапазоне от 0 до 65535.

469. Метаоператор здесь не разрешен

Метаоператоры должны быть первыми операторами в строке.

470. Структурные операторы здесь не разрешены

В однострочных операторах IF не разрешены структурные операторы, такие как WHILE/WEND, DO/LOOP и SELECT CASE.

471. Неизвестный идентификатор или синтаксическая ошибка

Что-то неправильно в строке программы, но компилятор не в состоянии идентифицировать ошибку.

472. Ошибка в индексах массива

Массив определен с одним числом размерности, а используется с другим.

473. Ошибка в границах массива

Используется индекс массива, находящийся вне возможного диапазона.

474. Несоответствие типов

Неодинаковые типы переменных в операторе SWAP.

475. Несоответствие параметров

Типы или число параметров функции или процедуры не соответствуют тем, что были в определении.

476. Ошибка в операторе CLEAR - используйте MEMSET / $STACK

У оператора CLEAR нет аргументов.

477. LOCAL требует DEF FN / SUB

Объявление переменной в операторе LOCAL может находиться только в функции или процедуре.

478. SHARED требует DEF FN / SUB

Объявление переменной в операторе SHARED может находиться только в функции или процедуре.

479. STATIC требует DEF FN / SUB

Объявление переменной в операторе STATIC может находиться только в функции или процедуре.

480. COMMON-массивы должны быть динамическими

Массивы, использованные в операторе COMMON, должны быть объявлены как динамические.

481. LOCAL-массивы должны быть динамическими

Массивы, определенные как LOCAL, не могут быть определены как STATIC. Избавьтесь от ключевого слова STATIC или вынесите определение массива из процедуры или функции в главную программу.

482. Массивы-параметры нельзя модифицировать

Нельзя использовать операторы ERASE и DIM для массива, являющегося параметром функции или процедуры.

483. Массив не статический

Для определения индексов статического массива нельзя использовать переменные. Необходимо применять либо константы, либо именованные константы.

484. Массив уже статический

Если в программе встречаются два объявления одного массива, то он автоматически считается динамическим. Ошибки возникают, если после этого в другом месте программы он объявляется как STATIC, или сначала он был объявлен как STATIC, а потом в другом операторе DIM - как DYNAMIC.

485. Ожидается имя массива без "( )"

В операторе ERASE не нужно указывать скобки после имени массива.

486. Массив превышает 64К

Размер массива не может превышать 64К. Если у вас больший массив, разбейте его на несколько массивов.

487. Массивы ограничены восемью размерностями

Максимальное число размерностей в массиве - восемь. Это внутреннее ограничение компилятора

488. Неверное представление числа

Число имеет в своём представлении более 18 цифр или вещественное число в экспоненциальном представлении не имеет цифр после символа E..

489. Недопустимое имя функции / процедуры

Имя функции должно начинаться с букв FN, за которыми следует буква и за ней другие буквы и цифры. Заканчиваться имя может идентификатором типа (%, &, !, # или $). Имя процедуры должно начинаться с буквы, за которой могут следовать другие буквы и цифры. Идентификаторы типа не могут присутствовать в имени процедуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информатика. Базовый курс. Учебное пособие для вузов / Под ред. С.В. Симоновича. – 2-е изд. – СПб.: Питер, 2007. – 640 с.

2. Степанов А.Н. Информатика: учебник для вузов. – 5-е изд. – СПб.: Питер, 2008. – 766 с.

3. Торхов В.Л. Турбо-Бейсик: Справочное руководство. - М.: Гендальф, 1993. - 63 с.

4. Правила тяговых расчетов для поездной работы. - М.: Транспорт, 1985, 287 с.

СОДЕРЖАНИЕ

Учебно-методическое издание

Долгачев Николай Иванович

Информатика

Часть II

Учебно-методическое пособие к лабораторным работам

Подписано в печать 9.11.2016	Формат 60×81 1/16	Изд. № 139-16
Усл. печ. л. 2,79	Заказ 393/16	