Кафедра конструирования и производства радиоаппаратуры

УТВЕРЖДАЮ
Заведующий кафедрой КИПР
_____В.Н. ТАТАРИНОВ
“___” _____2012 г.

Линейные программы

Лабораторная работа по дисциплинам **«**Информатика**»** для студентов специальности 211000.62 (бакалавриат) и «Информатика и информационные технологии» специальности 162107.65 (специалитет)

Разработчик:
Доцент кафедры КИПР
_____**Ю.П. Кобрин**

Томск 2012

СОДЕРЖАНИЕ

1 Цели работы

✓ Изучение способов представления арифметических данных

✓ Знакомство с правилами записи арифметических выражений с использованием встроенных арифметических функций

✓ Изучение работы операторов вызова стандартных процедур, операторов присваивания, операторов ввода и операторов вывода.

✓ Освоение приемов программирования и отладки простейших линейных алгоритмов.

✓ Закрепление навыков работы в интегрированной среде ***Borland Pascal.***

2 Порядок выполнения работы

1) Ознакомиться с разделами «Представление и преобразование информации в программах на языке Pascal», «Методика разработки линейных программ» а также, при необходимости, а также с дополнительной литературой [1] [2] [3] [4] [5] [6].
2) Ответить письменно на контрольные вопросы.
3) Получить у преподавателя индивидуальное задание. Разработать программу в соответствии с индивидуальным заданием.
1) Войти в свой личный каталог и загрузить и настроить систему ***Borland Pascal 7.0***.
2) Ввести шаблон программы, сделанный на прошлом занятии. Сделать его копию, записав под новым именем (линейной программы) с помощью команды ***Save as*** ...
4) Ввести разработанную Вами программу, корректируя и дополняя свой шаблон.
5) Добиться с помощью отладки, чтобы программа давала правильные результаты
6) Оформить отчет и защитить его у преподавателя.

3 Контрольные вопросы

Ответьте на следующие контрольные вопросы:

1) Перечислите стандартные типы языка ***Borland Pascal***.
2) Как записываются операторы заголовка и конца программы? Является ли обязательным заголовок программы в языке ***Borland Pascal***?

1) Каковы диапазоны допустимых значений для целых и вещественных типов данных?
2) В каком порядке выполняются операции в выражениях?
3) Как работает оператор присваивания?
4) Как ввести в программу данные с клавиатуры?
5) Как записываются операторы вывода на экран в языке Паскаль?
6) В чем преимущества использования подпрограмм?

4 Защита отчета

Отчет должен быть выполнен в соответствии с [9] и состоять из следующих разделов:

- ✓ Тема и цель работы.
- ✓ Индивидуальное задание.
- ✓ Ответы на контрольные вопросы.
- ✓ Текст программы и вводимые тестовые исходные данные.
- ✓ Результаты выполнения программы.
- ✓ Выводы.

При защите отчета по работе для получения зачета студент должен:

- ✓ уметь отвечать на контрольные вопросы;
- ✓ обосновать структуру выбранного алгоритма и показать его работоспособность;
- ✓ уметь пояснять работу программы;
- ✓ продемонстрировать навыки работы в среде ***Borland Pascal***.

5 Типы данных в языке *Borland Pascal*

5.1 Общие понятия

Решение любой задачи достигается обработкой информации или данных. Поэтому, как программисту, Вам необходимо знать, ***как***:

✓ ***осуществить ввод данных*** - т.е. ввести информацию (данные) в программу с клавиатуры, диска или из порта ввода/вывода;

✓ ***сохранить данные*** (***информацию***) во внутренней (оперативной) или внешней (экран, магнитные и оптические диски и другие устройства ввода-вывода) памяти. Это достигается использованием констант, переменных и структур, содержащих числа (целые и вещественные), текст (символы и строки) или указатели - адреса переменных и структур;

✓ ***правильно задать команды обработки данных*** (***операторы***, ***инструкции***). С их помощью осуществляют присваивание значений, вычисление выражений, сравнение значений (равно, не равно, больше и т.д.);

✓ ***получить данные из программы*** (***вывод данных***) на экран, на диск или в порт ввода/вывода.

Вы можете написать и упорядочить свои команды так, чтобы:

✓ некоторые из них выполнялись при выполнении некоторого условия или ряда условий (условное выполнение);

✓ другие выполнялись определенное число раз (циклы), пока истинно некоторое условие, или пока условие не станет истинным;

✓ другие собирались в отдельные части, объединенные именем (подпрограммы), которые могут быть выполнены в нескольких местах программы, где есть вызов их по имени.

5.2 Данные

Язык Паскаль является строго-типизированным языком программирования. Это означает, что для всех переменных в программе должен быть определен их тип данных.

Понятие **типа данных** является ключевым в языке Паскаль.

Типы данных характеризуют:

- способ их внутреннего (машинного) представления (кодирования),
- множество допустимых значений, которые они могут принимать,
- набором операций, которые могут быть выполнены с ними.

Программа на языке ***Borland Pascal*** оперирует всевозможными данными - числами, символами, строками символов, массивами и т.д.

Данные - это конкретные значения, которые обрабатываются во время выполнения программы.

Обязательное описание типа приводит к избыточности в тексте программ, но такая избыточность является важным вспомогательным средством разработки программ и рассматривается как необходимое свойство современных алгоритмических языков высокого уровня. Поэтому любые данные должны быть **явно** отнесены к определенным *типам.*

Кроме ***стандартных типов данных*** (определенных в самом языке), в ***Borland Pascal***. предусмотрен механизм создания новых *(**пользовательских**)* типов данных, заданных самим программистом. Благодаря этому общее количество типов, используемых в программе, может быть сколь угодно большим.

Borland Pascal характеризуется разветвленной структурой типов данных (*рис. 5.1*).

Рисунок 5.1 - Типы данных в языке Borland Pascal

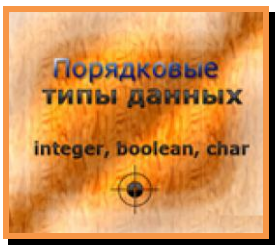
Рассмотрим основные стандартные типы данных, используемые в программировании. Но предварительно ознакомимся с одним из допустимых способов описания конструкций языков программирования.

5.3 Простые типы данных

К ***простым типам (скалярным типам) данных*** относят типы данных таких величин, значения которых не содержит составных частей.

К простым типам (см. *рис. 5.1*) относятся ***порядковые*** и ***вещественные*** типы.

Порядковые типы отличаются тем, что каждый из них имеет конечное число возможных значений. Эти значения можно определенным образом упорядочить (отсюда - название типов) и, следовательно, с каждым из них можно сопоставить некоторое целое число - *порядковый номер значения*.



Как видно из *рис. 5.1* , к порядковым типам относятся: ***целые***, ***логический***, ***символьный***, ***перечисляемый*** и ***тип-диапазон***. К любому из них применима функция ***Ord(X)***, которая возвращает порядковый номер значения выражения ***X*.**

Вещественные типы (с дробной частью), строго говоря, тоже имеют конечное число значений, которое определяется форматом внутреннего представления вещественного числа. Однако количество возможных значений вещественных типов настолько велико, что сопоставить с каждым из них целое число (его номер) не представляется возможным.

5.3.1 Целочисленные типы чисел

В ***Borland Pascal*** каждый целочисленный тип обозначает определенное подмножество целых чисел (чисел без дробной части). Диапазон возможных значений целых типов зависит от их внутреннего представления, которое может занимать один, два или четыре байта, как это показано в следующей таблице.

Таблица 5.1 - Стандартные целочисленные типы данных

Тип	Название	Диапазон	Формат
Shortint	короткое целое	-128 .. 127	8 бит со знаком
Integer	целое	-32768 .. 32767	16 бит со знаком
Longint	длинное целое	-2147483648 .. 2147483647	32 бита со знаком
Byte	длиной в байт	0 .. 255	8 бит без знака
Word	длиной в слово	0 .. 65535	16 бит без знака

5.3.2 Перечисляемый тип

Перечисляемый тип задается перечислением тех значений, которые он может получать. Это тип, определяемый пользователем. Каждое значение именуется некоторым идентификатором и располагается в списке, обрамленном круглыми скобками, например:

```
type
  tColors = (red, green, blue);
  tMonth = (jan, feb, mar, may, jun, jul, aug, sep, oct, nov, dec);
```

Соответствие между значениями перечисляемого типа и порядковыми номерами этих значений устанавливается порядком перечисления: первое значение в списке получает порядковый номер 0, второе - 1 и т.д. Максимальная мощность перечисляемого типа составляет 65536 значений.

Применение перечисляемых типов делает программы нагляднее.

5.3.3 Тип-диапазон (интервальный)

Тип-диапазон есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона. Тип-диапазон задается границами своих значений внутри базового типа:

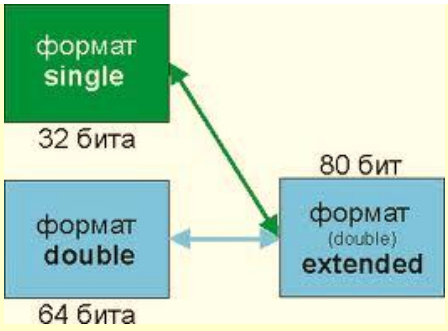
<***мин.знач***.> **..** <***макс.знач***.>

Здесь <***мин.знач***.> - минимальное значение типа-диапазона, <***макс.знач***.> - максимальное его значение. Например:

```
type
  digit = '0' .. '9';
  MonthDay = 1 .. 31;
```

Чаще всего тип-диапазон употребляется для указания допустимых границ изменения индексов у элементов массивов.

5.3.4 Вещественные (действительные) типы чисел



К вещественному типу относится подмножество вещественных чисел, которые могут быть представлены в машинном формате с плавающей точкой *с фиксированным числом цифр.* Вещественные типы в ***Borland Pascal*** различаются диапазоном и точностью связанных с ними значений (см. *Табл. 5.2*).

Таблица 5.2 - Диапазоны представления и точность представления (числа значащих десятичных цифр) для вещественных типов данных

Тип	Название	Диапазон	Число значащих цифр	Размер в байтах
Real	вещественное	$2.9 \cdot 10^{-39}$ .. $1.7 \cdot 10^{38}$	от 11 до 12	6
Single	с одинарной точностью	$1.5 \cdot 10^{-45}$ .. $3.4 \cdot 10^{38}$	От 7 до 8	4
Double	с двойной точностью	$5.0 \cdot 10^{-324}$ .. $1.7 \cdot 10^{308}$	от 15 до 16	8
Extended	с повышенной точностью	$1.9 \cdot 10^{-4951}$ .. $1.1 \cdot 10^{4932}$	от 19 до 20	10
Comp	сложный тип[1]	$-2^{63} + 1$ .. $2^{63} - 1$	от 19 до 20	8

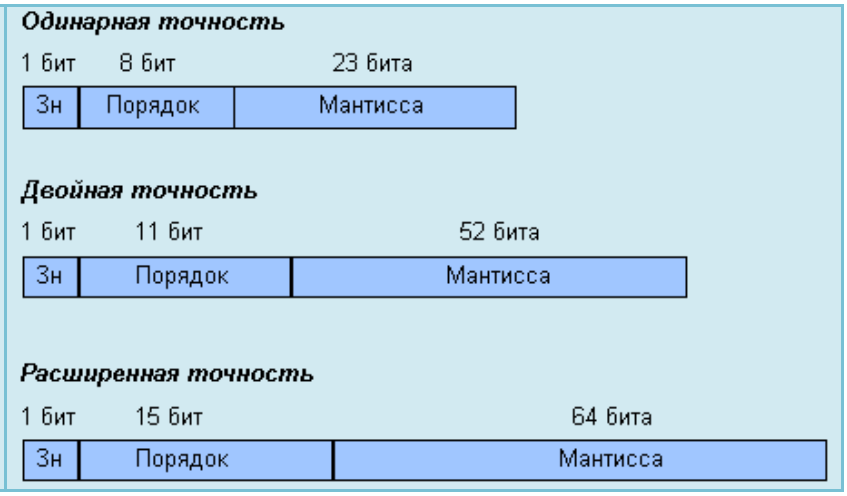
Вещественные типы являются упорядоченными, но не порядковыми.

Внутреннее машинное представление различных типов вещественных чисел.

В любом представлении старший бит **Зн** определяет знак вещественного числа:

0 - положительное число;

1 - отрицательное число.



Так как при любом варианте кодирования число битов в мантиссе ограничено, вещественные числа хранятся неточно. Каждый из имеющихся вещественных типов гарантирует правильное хранение только определенного количества значащих (**верных)** цифр.

Неточности в хранении вещественных чисел могут привести к тому, что при вычитании близких чисел может произойти потеря значимости. Это же объясняет, почему следует избегать сравнения вещественных величин на точное равенство.

5.3.5 Символьный тип

Значением символьного типа (***char***) является множество всех символов персонального компьютера, упорядоченных в соответствии с расширенным набором символов кода ***ASCII*** (*American Standard Code for Information Interchange* - американский стандартный код для обмена информацией).

---

[1] Сложный тип содержит только целочисленные значения в диапазоне $-2^{63}+1$ … $2^{63}-1$, что приблизительно соответствует диапазону $-9.2 \cdot 10^{18}$ … $9.2 \cdot 10^{18}$ в десятичной системе счисления.

При таком подходе любой символ кодируется одним байтом, а это всего 256 возможных значений. Естественно, все возможные символы в этот диапазон не уложить. Поэтому разработчики кодовой таблицы выбирают, какие алфавиты охватить. Первая половина таблицы символов ПК с кодами 0...127 соответствует стандарту ***ASCII*** (*табл. 5.3*) и может использоваться в программах ***Borland Pascal***. Вторая половина символов с кодами 128...255 предназначена для представления символов национальных алфавитов и применяется лишь в комментариях к программе и в качестве значений текстовых строк.

Таблица 5.3 - Кодировка символов в соответствии со стандартом ASCII

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	NUL	32	BL	64	®	96	'
1	ЗОН	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	k	107	k
12	FF	44	,	76	L	108	i
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	0
16	DEL	48	0	80	p	112	P
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	V
23	ETB	55	7	87	w	119	w
24	CAN	56	8	88	X	120	X
25	EM	57	9	89	У	121	У
26	SUB	58	:	90	z	122	z
27	ESC	59	/	91	[	123	{
28	FS	60	<	92	\	124	I
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	n

Символы с кодами **0 .. 31** относятся к служебным кодам. Если эти коды используются в символьном тексте программы, они считаются пробелами. В операциях ввода-вывода некоторые из них могут иметь самостоятельное значение (*табл. 5.4*).

В Windows популярна кодовая таблица - Unicode. В ней каждый символ обозначен двумя байтами. В 65536 позициях новой таблицы умещаются все современные алфавиты, слоговые системы письма, специальные знаки (вроде интегралов и нот), даже китайские иероглифы. Но Pascal создавался во времена DOS и кодировку Unicode не поддерживает.

Таблица 5.4 - Кодировка служебных символов в соответствии со стандартом ***ASCII***

Символ	Код	Значение
BEL	7	Звонок; вывод на экран этого символа сопровождается звуковым сигналом
HT	9	Горизонтальная табуляция; при выводе на экран смещает курсор в позицию, кратную 8, плюс 1 (9, 17, 25 и т.д.)
LF	10	Перевод строки; при выводе его на экран все последующие символы будут выводиться, начиная с той же позиции, но на следующей строке
VT	11	Вертикальная табуляция; при выводе на экран заменяется специальным знаком
FF	12	Прогон страницы; при выводе на принтер формирует страницу, при выводе на экран заменяется специальным знаком
CR	13	Возврат каретки; вводится нажатием на клавишу ***Enter*** (при вводе с помощью ***READ*** или ***READLN*** означает команду «Ввод» и в буфер ввода не помещается; при выводе означает команду «Продолжить вывод с начала текущей строки»)
SUB	26	Конец файла; вводится с клавиатуры нажатием ***Ctrl-Z***; при выводе заменяется специальным знаком
SSC	27	Конец работы; вводится с клавиатуры нажатием на клавишу ESC; при выводе заменяется специальным знаком

Каждому символу приписывается целое число в диапазоне **0 .. 255**. При вызове функции ***Ord(Ch)***, где ***Ch*** - значение символьного типа, возвращается порядковый номер внутреннего представления символа ***Ch***.

5.3.6 Логический тип

Логическое высказывание (выражение) может быть либо истинно, либо ложно. Значениями логического типа (***Boolean***) может быть одна из предварительно объявленных констант ***False*** (Ложь) или ***True*** (Истина).

Переменные типа ***Boolean*** занимают 1 байт.

5.4 Формулы Бэкуса–Наура

Для определения элементов языка и синтаксиса конструкций объектов программы на языке ***Borland Pascal*** обычно употребляются *формулы Бэкуса–Наура*, в которых придерживаются в которой одни синтаксические категории последовательно определяются через другие категории. В этих формулах придерживаются следующих соглашений:

✓ ***определяемое понятие*** отделяется от своего ***определения***, находящегося в правой части формулы, знаком ***::=*** , который заменяет выражение «определяется как» или «это есть»;

✓ определяемые понятия, обозначения и неделимые единицы текста заключаются в ***треугольные скобки < >.*** Текст внутри угловых скобок характеризует элемент, однако не описывает синтаксис этого элемента;

✓ допускаются ***рекурсивные определения*** (определение выражается через само себя в правой части);

✓ **необязательные элементы** заключаются в квадратные скобки **[ ]**;

✓ элементы списков или альтернативные варианты конструкций (ИЛИ) отделяются друг от друга вертикальной чертой |;

✓ ***стандартные множества*** задаются своими граничными значениями, отделенными друг от друга многоточием **..** (т.е. задают диапазон значений).

5.5 Константы

Константами называются данные программы, значения которым присваиваются компилятором *перед началом* выполнения программы и не могут быть изменены до завершения ее работы.

> ***Константа*** характеризуется фиксированным именем, фиксированным типом и фиксированным значением[2]. Так, **1947** есть *имя* константы целого типа с фиксированным значением **1947**.

В качестве констант в ***Borland Pascal***, могут использоваться целые, вещественные и шестнадцатеричные числа, логические константы, символы, строки символов, конструкторы множеств и признак неопределенного указателя ***Nil***.

В языке ***Borland Pascal***, можно использовать значение константы непосредственно в явном виде (как операнд выражения), например:

```
100
'A'
(2.5 + 1) / (2.5 - 1)
'Borland' + 'Pascal'
```

Тип констант автоматически распознается компилятором без предварительного описания.

---

[2] Кроме типизированных констант, значения которых можно изменять в ходе выполнения программы.

Значения констант можно задать в описательной части программы. Можно дать константе имя и затем обращаться к ней по имени.

Именованная константа - это имя (идентификатор), которое в программе используется вместо самой константы.

Все именованные константы должны быть описаны в специальном разделе, который начинается зарезервированным словом **const** (англ. constant – константа).

<**объявление константы**> **::=** ***const*** <**список**>***;***

<**список**> ::= <**элемент**>[;<**список**>]

<**элемент**> ::= <**имя**> **=** <**значение**>

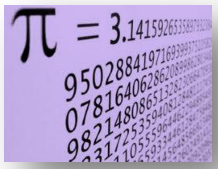
Например, если вы используете в программе номер Вашего телефона, то его лучше всего задать константой, так как он не меняет своего значения.

В правой части допускается использовать не только значения константы, но и выражения[3], в котором могут быть использованы ранее объявленные константы:

<**элемент**> **::=** <**имя**> = <***выражение***>

В выражениях-константах допускается использовать следующие стандартные функции:

Abs, Chr, Hi, High, Length, Lo, Low,
Odd, Ord, Pred, Ptr, Round, SizeOf, Succ, Swap, Trunc.



В Pascal имеется ряд констант, к значениям которых можно обращаться без предварительного определения. Их называют **зарезервированными константами**. Например, число π

Примеры использования выражений-констант в описаниях констант:

```
const {Раздел объявления констант}
    {Числовые константы - записываются обычным образом}
    Gamma = 5.6712E-3;
    Eps = 1e-5;
    Ln10 = 2.302585092994095684;
    Ln10R = 1 / Ln10;
    Min = 0; Max = 100;
    Center = (Max - Min) div 2;
```

[3] Значения всех выражений в разделе объявлений констант вычисляются один раз на шаге компиляции, перед выполнением программы.

```
Const {Раздел объявления констант}
    {Символьные константы}
    Char7 = '7';          {один символ заключается в апострофы}
    CharCr = #13;         {задается символ с соответствующим номером по
таблице ASCII в десятичной системе счисления}
    Beta  = Chr(187); {функция Chr преобразовывает целое значение аргумента в
символ с соответствующим номером по таблице ASCII}
    Cod = $124;  {задается символ с соответствующим номером по таблице ASCII
в шестнадцатеричной системе счисления}
    {Строковые константы }
    Address = 'г. Томск, проспект Ленина, ТУСУР, кафедра КИПР';
    Message = 'Out of memory';
    ErrStr = 'Error: ' + Message + '.' + CharCr;
    Alpha = '2.4';  {Это именно строковая константа, т.е. строка символов, которая
изображает число "две целые четыре десятых", а не число 2,4}
```

5.6 Переменные

Переменными называются данные, которые в ходе выполнения программы могут принимать различные значения.

К переменным обращаются по имени. Они получают значения только в ходе выполнения программы при выполнении операций присваивания или чтения. Имя переменной подобно ящичку, который можно заполнить различными значениями, чего нельзя сделать с константой.

Переменная характеризуется именем, типом и переменным значением из множества допустимых значений данного типа.

Как и в большинстве языков программирования, в Паскале все переменные, используемые в программе, описываются в ее начале в специальном **разделе описания переменных** после зарезервированного слова **Var**. С помощью ее объявления устанавливается не только факт существования переменной, но и задается ее тип (стандартный или предварительно определенный пользователем). Таким образом компилятор получает информацию о том, сколько байт необходимо выделить для хранениянданного, диапазон допустимых значений, способ кодирования и набор допустимых операций.

В общем виде раздел объявления переменных выглядит так:

```
Var  {Раздел объявления переменных}
   <список> : <тип>;
   <список> ::= <имя>[,<список>]
```

где:

- ✓ ***имя*** - имя переменной;
- ✓ ***тип*** - тип данных, для хранения которых предназначена переменная.

Пример объявления переменных:

```
Var
  a: Real;
  b: Real; i : Integer;
```

В приведенном примере объявлены две переменные типа ***Real*** и одна переменная типа ***Integer***.

Таким образом, чтобы программа могла обратиться к переменной (области памяти), например, для того, чтобы получить исходные данные для расчета по формуле или сохранить результат, переменная должна иметь имя.

Имя переменной придумывает программист.

В качестве имени переменной можно использовать последовательность из букв латинского алфавита, цифр и некоторых специальных символов. Первым символом в имени переменной должна быть буква. Пробелы и другие специальные символы в имени переменной использовать нельзя.

Следует обратить внимание на то, что компилятор языка ***Borland Pascal*** не различает прописные и строчные буквы в именах переменных, отчего имена ***SUMMA***, ***Summa*** и ***summa*** обозначают одну и ту же переменную.

Разумно, чтобы имя переменной было логически связано с ее назначением!

Например, если в программе есть переменные, предназначенные для хранения суммы покупки и величины скидки, то этим переменным можно присвоить имена

TotalSumm и ***Discount*** или ***ObSumma*** и ***Skidka***.

В тексте программы объявление каждой переменной, как правило, помещают *на отдельной строке*. Это дает возможность сделать комментарий к этой переменной: указать ее назначение, размерность, диапазон значений и т.п.

Если в программе имеется несколько переменных, относящихся к одному типу (список переменных), то имена этих переменных можно перечислить через запятую, а тип переменных указать после имени последней переменной через двоеточие, например:

```
var
  Result,          {результат вычислений}
  SummaDay,        {сумма  выручки за день}
   Power: Real;    {Мощность, потребляемая РЭС, вт}
  Flag,            {флаг события: True – свершилось, False - нет}
  Error: Boolean; {ошибка: True – случилась, False - нет}
```

5.7 Типизированные константы

Исключением являются «типизированные константы», описываемые в разделе объявления констант (***Const***). Типизированная константа равнозначна переменной с заранее инициализированным значением, и в программе действия с ней могут производиться так же, как с переменной.

Типизированные константы являются своеобразным промежуточным звеном между переменными и константами. Слово «**константа**» означает, что данные этого типа описываются в разделе const, а слово «типизированная» указывает, что для них должен указываться и тип, как у переменных:

const

<идентификатор>:<тип>=<значение>;

Пример:

```
const
   VideoSeg : word = $B800;
   Ocenka : byte=4;
   Predmet : string='Информатика';
```

5.8 Выражения

С помощью выражений задаются правила вычисления новых значений на основе ранее уже известных. Арифметические выражения в ***Borland Pascal*** составляются из имен переменных и функций, констант, знаков операций и скобок. К моменту выполнения операций всем переменным, входящим в выражение, тем или иным способом должны быть присвоены конкретные значения.

5.8.1 Последовательность выполнения операций

Последовательность выполнения операций в выражениях определяется:

- ✓ приоритетом операций;
- ✓ порядком расположения операций в выражениях;
- ✓ использованием скобок.

Для задания нужного порядка выполнения операций в выражении можно использовать скобки, например:

(r1 + r2 +r3) / (r1 * r2 * r3)

Выражение, заключенное в скобки, трактуется как один операнд. Это означает, что операции над операндами в скобках будут выполняться в обычном порядке, но раньше, чем операции над операндами, находящимися за скобками.

При записи выражений, содержащих скобки, должна соблюдаться парность скобок, т.е. число открывающих скобок должно быть равно числу закрывающих скобок.

Нарушение парности скобок - наиболее распространенная ошибка при записи выражений!

Операции более высокого ранга (умножение и деление) выполняются раньше, чем операции более низкого ранга (сложение и вычитание). Операции одного ранга выполняются *слева направо*.

Наивысший приоритет имеют *унарные* (содержащие один операнд) операции:

1) **+** (сохранение знака)
2) **-** (отрицание знака)
3) ***not*** (логическое отрицание).

Например,

-7 →-7 -(-6) → 6 not False → True

Затем выполняются *бинарные* (содержащие два операнда) операции *типа умножения:*

4) ***** (умножение);
5) **/** (деление);
6) ***div*** (целочисленное деление);
7) ***mod*** (остаток от деления целых чисел);
8) ***and*** (логическое И);
9) ***shl*** (поразрядный сдвиг влево, операция над двоичным кодом);
10) ***shr*** (поразрядный сдвиг вправо, операция над двоичным кодом).

Например,

3.14 * 2 → 6.28 5 / 2 →2.5 5 div 2 →2 23 mod 5 →3

Вслед за тем выполняются бинарные операции *типа сложения:*

11) **+** (сложение);
12) **–** (вычитание);
13) ***or*** (логическое ИЛИ, операция над двоичным кодом);
14) ***xor*** (логическое исключающее ИЛИ, операция над двоичным кодом).

Например,

5 + 7 → 9 5.37 – 2.15 → 3.22 True or False → True

Самый низший приоритет имеют операции типа *бинарных отношений:*

15) **=** (равно);
16) **<>** (не равно);
17) **>** (больше);
18) **<** (меньше);
19) **>=** (больше или равно);
20) **<=** (меньше или равно);

Например,

2 = 2 →True 2 <> 2 →False 3 > 2 →True 3 <= 4 →True

5.8.2 Результаты вычисления выражений

В следующей таблице приведены типы операндов и результаты для бинарных арифметических операций:

Таблица 5.5 - Бинарные арифметические операции

Операция	Действие	Типы операндов	Тип результата
+	Сложение	Целый Вещественный	Целый Вещественный
-	Вычитание	Целый Вещественный	Целый Вещественный
*******	Умножение	Целый Вещественный	Целый Вещественный
/	Деление	Целый Вещественный	Целый Вещественный
div	Целочисленное деление	Целый	Целый
mod	Остаток от деления	Целый	Целый

Конкретный тип результата можно определить, руководствуясь следующими правилами:

- ✓ если оба операнда в операциях +, -, *, ***div*** или ***mod*** являются операндами целого типа, то тип результата будет таким же, как общий тип обоих операндов;
- ✓ если один или более операндов в операциях +, -, или * имеют вещественный тип, то тип результата будет *вещественным*;
- ✓ если при использовании унарной операции сохранения знака или операции отрицания знака операнд имеет целый тип, то результат будет тоже целого типа. Если операнд вещественного типа, то тип результата будет вещественным или типом с повышенной точностью (*extended*).
- ✓ значение выражения ***x / y всегда*** будет вещественного типа (*real*) или с повышенной точностью (*extended*), независимо от типов операндов. Если y равно 0, то результат будет ошибочным;
- ✓ значение выражение ***i div j*** (деление нацело) представляет собой математическое частное от ***i / j***, округленное в меньшую сторону до значения целого типа. Если ***j*** равно ***0***, результат будет ошибочным;
- ✓ операция ***mod*** возвращает остаток, полученный путем деления двух ее операндов, то есть:

i mod j = i - (i div j) * j

Знак результата операции ***mod*** будет тем же, что и знак ***i***. Если ***j*** равно нулю, то возникает ошибка.

5.8.3 Некоторые встроенные функции

Для выполнения часто встречающихся вычислений и преобразований язык ***Borland Pascal*** предоставляет программисту ряд стандартных подпрограмм-функций (*табл. 5.4*).

Значение функции связано с ее именем. Поэтому функцию можно использовать в качестве операнда выражения, например в инструкции присваивания. Так, чтобы вычислить квадратный корень, достаточно записать ***k := Sqrt(X)***, где ***Sqrt*** - функция вычисления квадратного корня, ***X*** — переменная, которая содержит число, квадратный корень которого надо вычислить.

Функция характеризуется типом значения и типом параметров. Тип переменной, которой присваивается значение функции, должен соответствовать типу функции. Точно так же тип фактического параметра функции, т.е. параметра, который указывается при обращении к функции, должен соответствовать типу формального параметра. Если это не так, компилятор выводит сообщение об ошибке.

Таблица 5.6 - Арифметические функции

Заголовок функции	Назначение	Тип результата	Примеры использования
Abs(X: real/integer): real/integer	Вычисление абсолютного значения аргумента ***X***	Совпадает с типом ***X***	***Abs(-2.3) ⇒ 2.3*** ***Abs(-157) ⇒ 157***
ArcTan(X : real): real	Вычисление угла, тангенс которого равен ***X*** радиан	Вещественный	***ArcTan(1) ⇒*** ***⇒ 7.854e-01***
Cos(X: real): real	Вычисление косинуса ***X***; параметр ***X*** задает значение угла в радианах	Вещественный	***Cos(Pi) ⇒*** ***⇒ -9.9999e-01***
Exp(X: real): real	Вычисление экспоненты ***X***, т.е. значение e^X (***e*** = ***2.718282*** – основание натурального логарифма)	Вещественный	***Exp(1) ⇒*** ***⇒2.71828e-01***
Frac(X: real): real	Вычисление дробной части ***X***	Вещественный	***Frac(123.456) ⇒*** ***⇒0.456*** ***Frac(-123.456) ⇒*** ***⇒ -0.456***
Int(X: real): real	Вычисление целой части ***X***	Вещественный	***Int(123.456) ⇒ 123.0*** ***Int(-123.456) ⇒*** ***⇒ -123.0***
Ln(X: real): real	Вычисление натурального логарифма ***X*** (по основанию ***e***)	Вещественный	***Ln(1) ⇒*** ***⇒ 2.718282e+00***

Заголовок функции	Назначение	Тип результата	Примеры использования
Pi: real	Возвращает значение числа ***π*** (***π = =3.14159265358979323 85***)	Вещественный	***Pi ⇒ ⇒3.141592653e+00***
Sin(X: real): real	Вычисление синуса ***X***; параметр задает значение угла в радианах	Вещественный	***Sin(Pi) ⇒ ⇒-0.0000e+00***
Sqr(X)	Вычисление X^2	Совпадает с типом ***X***	***Sqr(5) ⇒ 25*** ***Sqr(2.5) ⇒ 6.25***
Sqrt(X: real): real	Вычисление $\sqrt{X}$	Вещественный	***Sqrt(2) ⇒ ⇒1.41421356***
Random: real	Генерирует значение случайного числа[4] из диапазона ***0*** .. ***0.99***	Вещественный	***Random⇒ 0.4876*** (случайное число)
Random(Range: word): word	Генерирует значение целого случайного числа из диапазона ***0*** .. ***Range***	Целый	***Random***(***1000***) ⇒ ⇒ ***745*** (случайное число)
Round(X: real): longint	Возвращает значение ***X***, округленное до ближайшего целого числа	Целый	***Round***(***1.4***) ⇒ ***1*** ***Round***(***1.5***) ⇒ ***2*** ***Round***(***-1.4***) ⇒ ***-1*** ***Round***(***-1.5***) ⇒ ***-2***
Trunc(X: real): longint	Возвращает ближайшее целое число, меньшее или равное ***X***, если ***X ≥ 0***, и большее или равное ***X***, если ***X < 0***.	Целый	***Trunc***(***1.4***) ⇒ ***1*** ***Trunc*** (***1.5***) ⇒ ***1*** ***Trunc*** (***-1.4***) ⇒ ***-1*** ***Trunc*** (***-1.5***) ⇒ ***-1***

6 Линейные программы

6.1 Общие понятия

Программы с линейной структурой являются простейшими и используются **в чистом виде** на практике достаточно редко: при расчете обычных арифметических и алгебраических выражений, при расчете по формулам, при решении ряда бытовых задач.

При любых значениях исходных данных в линейном алгоритме все действия ***A1, A2, …. AN*** выполняются однократно, строго последовательно, одно за другим (*Рис. 5.1*). Такой порядок выполнения действий называется *естественным*.

Алгоритм линейной структуры представляет собой последовательность действий и не содержит каких-либо **условий.**

---

[4] Для инициализации генератора случайных чисел случайным значением можно предварительно воспользоваться процедурой ***Randomize***.

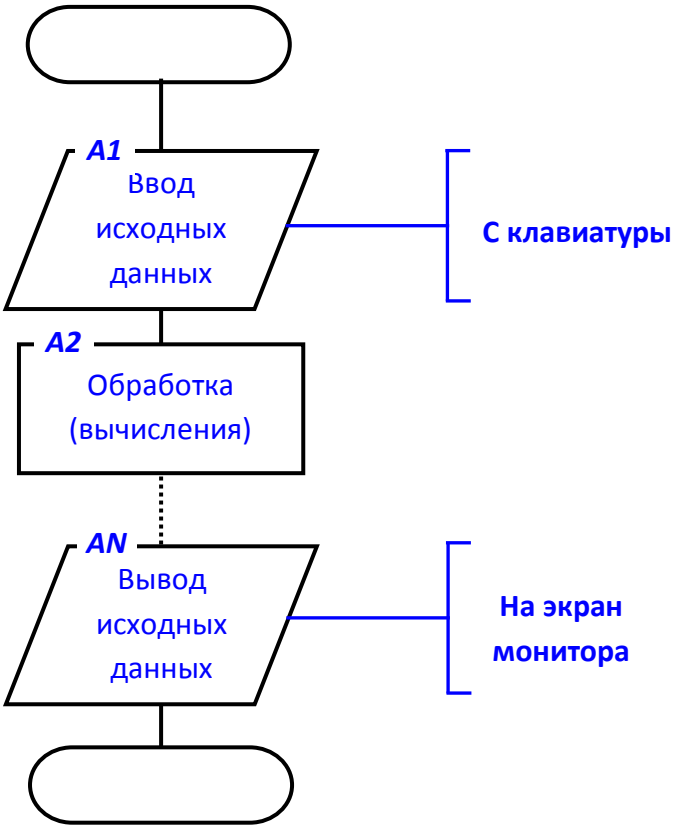


Рисунок 5.1 – Линейный алгоритм

В линейных программах могут применяться только:

- ✓ операторы (процедуры) ввода,
- ✓ операторы (процедуры) вывода,
- ✓ операторы присваивания (изменения значения переменных),
- ✓ операторы обращения к подпрограммам.

Рассмотрим важнейшие элементы программы на языке ***Borland Pascal***, используемые при создании линейных программ.

6.2 Операторы

Для преобразования информации в программе на языке ***Borland Pascal*** используют операторы. **Оператором** называется элемент программы, задающий некоторое законченное действие. Это элементарный шаг алгоритма, реализуемого программой. В программе операторы отделяются друг от друга символом «точка с запятой» (;).

Операторы выполняются строго последовательно, в том порядке, в котором они записаны в тексте программы в соответствии с синтаксисом и правилами пунктуации. В программе на языке Pascal раздел операторов является основным, так как именно в нем выполняются действия с предварительно описанными переменными, константами, значениями функций, что и позволяет получить результат, ради которого создавалась программа.

В простейших линейных программах могут использоваться операторы присваивания, ввода и вывода, а также вызова подпрограмм-процедур.

6.3 Оператор присваивания

Оператор присваивания является основной вычислительной инструкцией. Если в программе надо выполнить вычисление, то нужно использовать оператор присваивания.

В результате выполнения оператора присваивания значение переменной меняется, ей присваивается новое значение.

В общем виде инструкция присваивания выглядит так:

<***Имя*** > ***:=*** <***выражение***>

где:

Имя – имя переменной, значение которой изменяется в результате выполнения оператора присваивания;

: = - символ присваивания.

Выражение - выражение, значение которого присваивается переменной, имя которой указано слева от символа оператора присваивания.

Оператор присваивания выполняется следующим образом:

1) Сначала вычисляется значение выражения, которое находится справа от символа присваивания (***: =***).

2) Затем вычисленное значение записывается в переменную, имя которой стоит слева от символа присваивания[5].

Например, в результате выполнения операторов:

I := 0; {значение переменной ***i*** становится равным нулю}

a := b + c; {значением переменной ***a*** будет число, равное сумме значений переменных ***b*** и ***c***}

j := j + 1; {значение переменной ***j*** увеличивается на единицу}

Done := (I > 0) and (I < 100); {Присвоение результата проверки условия ***0 < I <100*** логической переменной ***Done***}

Выражение должно быть *совместимо по присваиванию с типом переменной.* Оператор присваивания считается верным, если тип выражения соответствует или может быть приведен к типу переменной, получающей значение. Например, переменной типа ***real*** можно присвоить значение выражения, тип которого ***real*** или ***integer***, а переменной типа ***integer*** можно присвоить значение выражения только типа ***integer***.

Так, например, если переменные ***i*** и ***n*** имеют тип ***integer***, а переменная ***d*** - тип ***real***, то

I := n / 10; i := 1.0; {операторы неправильные}

---

[5] Старое значение переменной слева от символа присваивания будет при этом утеряно.

d := i +1; {операторы записаны правильно}

6.4 Использование подпрограмм

Подпрограмма - это некоторая последовательность операторов, к которой можно обратиться по имени. Всякий раз, когда мы называем имя подпрограммы, инициируется последовательность запрограммированных в ней действий.

Когда следует использовать подпрограммы?

Логично применить процедуру или функцию как средство экономии памяти. Если Вы обнаружите, что в Вашей программе одни и те же действия выполняются *многократно, но с разными исходными данными*, то целесообразно один раз записать эту последовательность действий в процедуре или функции, а затем многократно вызывать ее из разных точек программы.

При создании программы для решения сложной задачи возникают те же проблемы, что и при разработке больших и сложных систем. Пути решения этих проблем для систем хорошо известны. Это, прежде всего, *декомпозиция* (разделение) сложной задачи на отдельные подзадачи, подзадач – на еще более мелкие подзадачи и так далее, пока подзадача не станет понятной и легко доступной для реализации небольшим числом простейших операторов. Каждая функционально законченная подзадача может и должна быть реализована подпрограммой.

Подпрограммы (процедуры и функции), используемые в языке ***Borland Pascal***, как раз и служат для разделения логики большой программы на обозримые и управляемые фрагменты. Любая подпрограмма может в свою очередь обращаться как к другим подпрограммам, так и к самой себе (рекурсия).

Программы, состоящие из процедур и функций, называют ***модульными***. Они имеют отчетливую *иерархическую структуру*. Опытные программисты обоснованно считают, что модульные программы гораздо легче для разработки и проще для понимания и тестирования, чем ***монолитные***.

Как обращаться к подпрограмме-функции в выражениях Вы уже знаете. Подпрограмма-процедура активизируется с помощью оператора процедуры, в котором содержатся имя процедуры, после которого в круглых скобках приводят список необходимых (фактических) параметров. Некоторые процедуры списка параметров могут и не иметь.

В языке ***Borland Pascal*** существует возможность хранить хорошо зарекомендовавшие себя процедуры и функции в модулях-библиотеках (***unit***). При необходимости Вы можете подключать к разрабатываемой программе на ***Borland Pascal*** описания данных и подпрограммы как из своих собственных *библиотек-модулей*, так и из *библиотек-модулей*, разработанных другими программистами.

Для подключения одного или более модулей с нужными подпрограммами необходимо в раздел описаний любой программы на ***Borland Pascal*** ввести предложение

```
Uses
  <список модулей-библиотек>;
```

К примеру, если Вы хотите изменять цвета выводимого на экран текста, то в Вашей программе необходимо использовать *стандартный* модуль ***Crt***, являющийся частью *библиотеки* ***Borland Pascal***. Для этого в модуле ***Crt*** имеются нужные подпрограммы. Строка ***uses Crt*** подключит модуль ***Crt*** к Вашей программе, и станет допустимым, например, следующий фрагмент программы:

```
Uses
  Crt; {Подпрограммы модуля Crt обеспечивают контроль над текстовыми режимами экрана, расширенными кодами клавиатуры, цветами, окнами и звуком}

  …
{Подготовка экрана дисплея}
{Вызываем процедуры из модуля CRT для настройки режима экрана:}
TextColor(Blue);          {установка цвета символов на экране (синего) }
TextBackGround(LightGray); {установка цвета фона экрана (светло-серого)}
ClrScr;                   {очистка экрана от предыдущих результатов}
```

6.5 Операторы ввода с клавиатуры

Интересно, что в ***Borland Pascal*** нет специальных операторов ввода-вывода. Для обмена информацией с окружающим миром в программах, написанных на языке ***Borland Pascal***, используются специальные стандартные процедуры ***Read*** и ***Write***, при обращении к которым допускается использование произвольного числа параметров. Параметры передаются этим процедурам в виде списка, располагающегося в круглых скобках сразу за именем процедуры.

Для считывания одного или нескольких значений текстовой информации с клавиатуры и преобразования их в значения одной или более переменных в соответствии с их объявленным типом в языке ***Borland Pascal*** предусмотрены процедуры ***Read*** и ***Readln***:

```
Read(<список переменных ввода>);
ReadLn(<список переменных ввода>);
```

где:

<***список переменных ввода***> ***::=*** <***переменная***>[,<***список ввода***>]

<***переменная***> ***::= имя переменной***[6]

Отличие работы оператора ***Readln*** от ***Read*** заключается в том, что после выполнения ***Readln*** осуществляется пропуск до начала следующей строки исходных данных (в отличие от ***Read***).

Рассмотрим особенности использования процедур ***Readln*** и ***Read*** для ввода данных наиболее распространенных типов.

---

[6] Имя переменной может быть любого числового, символьного или строкового типа. Число имен переменных может быть любым (в том числе и пустым).

6.5.1 Ввод данных символьного типа

Пусть на клавиатуре при выполнении программы ***Prim1*** на клавиатуре последовательно нажаты следующие клавиши: ***@, <Пробел>, 5, <Enter>***

```
Program Prim1; {Изучение ввода данных символьного типа}
Var
  Cod, Probel, Number: Char; {объявлены переменные символьного типа, значения которых вначале не определены}
Begin {Начало основного блока программы Prim1}
  Read(Cod, Probel, Number); {чтение значений переменных Cod, Probel, Number с клавиатуры}
end. {Конец основного блока программы Prim1}
```

Переменные, записанные в списке ввода оператора ***Read***, получат значения:

Cod будет равно ***'@'*** (номер символа по таблице ***ASCII - 96***)
Probel будет равно ***' '*** (номер символа по таблице ***ASCII - 32***)
Number будет равно ***'5'*** (номер символа по таблице ***ASCII - 85***)

6.5.2 Ввод данных целого или вещественного типа.

Пусть на клавиатуре при выполнении программы ***Prim2*** на клавиатуре в двух строках набрана следующая информация:

66, <Пробел>, -374, <Enter>
73.4e-3, <Пробел>, -123456, <Enter>

```
Program Prim2; {Изучение ввода данных целого или вещественного типа}
Var
  I, J: Integer;          {объявлены две переменные целого типа}
  A, B: real;             {объявлены две переменные вещественного типа}
begin {Начало основного блока программы Prim2}
  Read(I);          {I  будет равно 66}
  Readln(J);        {J  будет равно -374}
  Read(A);          {A  будет равно 0.0734}
  Readln(B);        {B  будет равно –123456.0}
end. {Конец основного блока программы Prim1}
```

Как видно из примера, выделение числа выполняется до обнаружения первого пробела, символа табуляции, признака конца строки (***Enter***) или файла. Если выделенная последовательность символов не соответствует числовым форматам, то происходит ошибка ввода-вывода.

6.6 Операторы вывода на экран дисплея

Для вывода информации на экран дисплея в языке ***Borland Pascal*** предусмотрены процедуры ***Write*** и ***Writeln***:

```
Write(<список элементов вывода>);
WriteLn(<список элементов вывода>);
```

где:

- ✓ <***список элементов вывода***> ***::=*** < ***элемент вывода*** >[,<***список элементов вывода***>]
- ✓ < ***элемент вывода***> ***::= Expr*** **[*: MinField* [*: DecDigits*]]**
- ✓ ***Expr*** – выводимое выражение символьного, целого, вещественного строкового или булевского типа.

Необязательный параметр, позволяющий отформатировать любой элемент вывода:

- ✓ ***MinField*** **–** выражение целого типа, задающее минимальную ширину поля вывода, которая должна быть больше нуля.

Необязательный параметр, позволяющий отформатировать выводимое число вещественного типа:

- ✓ ***DecDigits*** **-** выражение целого типа, задающее *число десятичных знаков, выводимых на экран после десятичной точки*. ***DecDigits*** указывается только для ***Expr*** вещественного типа, если указан параметр ***MinField*****.**
- ✓ Если ***DecDigits*** указывается, то число выводится в формате с фиксированной точкой, а если не указывается, то в формате с плавающей точкой.

Формат вывода с фиксированной точкой:

- ✓ [<***пробелы***>] [-] <***цифры***> [**.**<***цифры дробной части***>]

Формат вывода с плавающей точкой:

- ✓ [-] <***цифра***> [**.**<***цифры дробной части***>] ***E*** [***+***|***-***<***показатель степени***>]

Отличие работы оператора ***WriteLn*** от ***Write*** заключается в том, что после выполнения ***WriteLn*** осуществляется переход на следующую строку (в отличие от ***Write***).

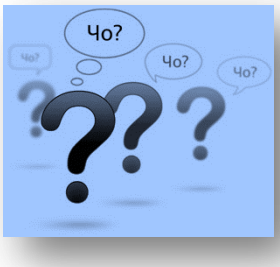
Пример**.**

```
Program TestWrite;  {тестирование возможностей форматирования вывода}
Var
   I: integer;          {объявлена переменная целого типа}
   R: real;             {объявлена переменная вещественного типа}
Begin {Начало основного блока программы TestWrite}

   {присвоение переменным I и R тестовых значений}
   I := 12345;
   R := -123.1234567;
```

```
    Writeln('Печать без форматирования');
    Writeln(I, R);
    Writeln; {Пропуск строки}

    Writeln('Форматированная печать');
    Writeln(I:10, R:10:3);
    Writeln; {Пропуск строки}

    Writeln('Печать вещественных чисел в фиксированном формате');
    Writeln(R:3:0);
    Writeln(R:5:3);
    Writeln(R:10:3);
    Writeln(R:11:7);
    Writeln(R:15:8);
    Writeln(R:25:8);
    Writeln; {Пропуск строки}

    Writeln('Печать вещественных чисел в плавающем формате');
    Writeln(R:3);
    Writeln(R:5);
    Writeln(R:11);
    Writeln(R:17);
    Writeln(R:25);

end. {Конец основного блока программы TestWrite}
```

Результаты выполнения программы ***TestWrite***[7]:

Печать без форматирования



1	2	3	4	5	-	1	.	2	3	1	2	3	4	5	6	7	0	0	0	0	E	+	0	0	0	2

[7] Клеточка обозначает позицию на экране или печатающем устройстве - знакоместо для вывода одного символа.

Форматированная печать

					1	2	3	4	5			-	1	2	3	.	1	2	3

Печать вещественных чисел в фиксированном формате

-	1	2	3																
-	1	2	3	.	1	2	3												
		-	1	2	3	.	1	2	3										
-	1	2	3	.	1	2	3	4	5	6	7								
		-	1	2	3	.	1	2	3	4	5	6	7	0					
												-	1	2	3	.	1	2	3	4	5	6	7	0

Печать вещественных чисел в плавающем формате

-	1	.	2	E	+	0	0	0	2															
-	1	.	2	E	+	0	0	0	2															
-	1	.	2	3	E	+	0	0	0	2														
-	1	.	2	3	1	2	3	4	5	6	7	E	+	0	0	0	2							
-	1	.	2	3	1	2	3	4	5	6	7	0	0	0	0	6	5	0	E	+	0	0	0	2

7 Методика разработки линейных программ

7.1 Пример разработки программы

Последовательность стандартных шагов: *ввести, сохранить, отредактировать и выполнить* - определяет общий сценарий разработки практически любой программы. Рассмотрим, как можно выполнить любой из этих шагов.

Пример.

Заданы стороны треугольника ***a, b, c***. Вычислить его высоты по формулам

$$h_a = \frac{2}{a}\sqrt{p(p-a)(p-b)(p-c)}, \quad h_b = \frac{2}{b}\sqrt{p(p-a)(p-b)(p-c)},$$

$$h_c = \frac{2}{c}\sqrt{p(p-a)(p-b)(p-c)}, \qquad p = 0.5\cdot(a+b+c).$$

Решение.

Для исключения повторений вычисления одного и того же выражения целесообразно ввести промежуточную переменную *R:*

$$R = 2\sqrt{p(p\quad a)(p\quad b)(p\quad c)}, \quad \text{и тогда} \quad h_a = R/a,\ h_b = R/b,\ \ h_c = R/c.$$

Программа на языке ***Borland Pascal*** может иметь следующий вид:

```
Program  DemoLine; {Расчет высот треугольника по его сторонам}
Uses
   CRT; {Подключаем к программе модуль CRT с процедурами работы с экраном:
TextColor, TextBackGround, ClrScr}
{Все данные - константы, переменные и т.п., а также процедуры и функции, использу-
емые программой, необходимо предварительно определить в разделах описания
данных перед использующим их программным блоком}
Var   {Начало раздела объявления переменных:}
   a, b, c,                    {значения сторон треугольника}
   p, R,                       {вспомогательные переменные}
   Ha, hb, hc: real;    {высоты треугольника}
Begin {Начало основного блока программы DemoLine }

  {Подготовка экрана дисплея}
  {Вызываем процедуры из модуля CRT для настройки режима экрана:}
  TextColor(Yellow);          {установка цвета символов на экране (желтого)}
  TextBackGround(Blue); {установка цвета фона экрана  (синего)}
  ClrScr;                     {очистка экрана от предыдущих результатов}

  {вывод назначения программы}
  WriteLn('   Расчет высот  треугольника по его сторонам');
  WriteLn; {пропуск строки после вывода назначения программы }

 {Ввод исходных данных:}
  WriteLn('Введите стороны треугольника:'); {вывод приглашения к вводу}
  Write('a : '); ReadLn(a);                 {диалог при вводе стороны a}
  Write('b : '); ReadLn(b);                 {диалог при вводе стороны b}
  Write('c : '); ReadLn(c);                 {диалог при вводе стороны c}

  {Решение:}
  p := 0.5 * (a + b + c); {расчет значения полупериметра p}
  R := sqrt(p * (p - a)*(p - b) * (p - c));       {расчет вспомогательной переменной}
 Ha := R / a; Hb := R / b; Hc := R / c;           {расчет высот треугольника}

  {Результаты:}
  WriteLn ('Высоты треугольника:');    {вывод результатов}
  WriteLn ('Ha = ', Ha:10:3, '    Hb = ', Hb:10:3, '    Hc = ', Hc:10:3)

end.   {конец основного блока программы DemoLine }
```

7.2 Ввод программы

После запуска ***IDE Borland Pascal*** сделайте текущим Ваш подкаталог с помощью команды ***Change Dir*** (Изменить директорию) меню ***File.*** Затем загрузите ранее сделанную Вами заготовку программы или выполните команду ***New*** меню ***File*** для организации нового окна. Далее с помощью уже известных команд редактора вводите исходный текст Вашей программы.

7.3 Сохранение программы

Для сохранения исходного текста следует использовать команду ***Save*** (Сохранить) меню ***File,*** либо нажать "горячую" клавишу команды ***Save - F2.*** На экране отображается окно, где следует ввести имя сохраняемого файла в виде обычной строки. Для выполнения команды после ввода имени файла следует нажать ***Enter*** либо установить мышь на поле ***OK*** и нажать кнопку. Новый файл с заданным именем будет сформирован в текущем каталоге, после чего изменится полоса заголовка у окна редактирования. Если не вводить новое имя, то файлу по умолчанию присваивается стандартное имя ***Noname00.pas***.

Для сохранения нового файла можно использовать и другие каталоги. Для этого следует ввести полное имя файла с явным указанием каталога, где он должен быть расположен. С помощью команды ***Save as ..*** (*Сохранить как...*) можно сохранить один и тот же файл под разными именами.

7.4 Компиляция программы

После того как программа сохранена, следует перейти к шагу компиляции. Во время компиляции происходит проверка на присутствие в тексте программы возможных синтаксических ошибок, и выполняются некоторые другие действия. Чтобы откомпилировать программу, следует нажать ***Alt-F9*** либо перейти к команде ***Compile*** *(Компилировать)* главного меню. Далее на экране появится окно ***Compiling*** *(Компиляция),* которое содержит вспомогательную информацию о ходе выполнения компиляции.

Если в тексте программы обнаружена ошибка, то на экране в первой строке будет отображено соответствующее сообщение об ошибке с указанием ее возможной причины. В этом случае курсор автоматически устанавливается либо в строку, приведшую к ошибке, либо в строку, следующую за ней. Текст такой программы с ошибками необходимо исправить и выполнить команду ***Compile*** еще раз. Если синтаксических ошибок больше нет, то компилятор создает образ *выполнимого (**executable**)* файла.

Команда ***Destination*** *(Назначение)* меню ***Compile*** поможет определить, где хранить выполнимый файл: в *памяти (**memory**)* или на *диске (**disk**)* в виде *.**EXE*** файла. При выполнении команды происходит переключение между этими двумя установками. Сохраните измененный текст программы с помощью клавиши ***F2*** и перейдите к выполнению программы.

7.5 Выполнение программы

Чтобы выполнить программу, следует нажать ***Ctrl-F9*** либо перейти к команде ***Run*** *(Выполнить)* главного меню и вызвать команду ***Run***. Перед стартом команды ***Run*** автоматически проверяется - проводилась ли компиляция программы. Если предварительной

компиляции не было, то программа сначала компилируется, а только затем выполняется. На нормальный ход выполнения программы может повлиять наличие *ошибок времени выполнения (**run-time errors**).* В этом случае компилятор выводит на экран соответствующее сообщение и автоматически позиционирует курсор на строку текста программы, приведшую к ошибке. После внесения соответствующих изменений такую программу следует опять отправить на выполнение. Этот процесс продолжается до тех пор, пока не удастся получить программу без ошибок.

Для проверки правильности работы программы, как правило, используют тестовый пример (или несколько тестовых примеров). Это наборы исходных данных, которых известны результаты решения задачи. Кроме конечных результатов при поиске логических ошибок, как правило, нужно знать и некоторые промежуточные результаты.

Во время работы программы привычное изображение ***IDE*** уступает место стандартному экрану DOS, содержимое которого аналогично тому, что было на экране перед запуском компилятора. После окончания работы программы управление передается обратно в ***IDE***.

7.6 Просмотр результатов

Чтобы просмотреть возможные видимые результаты работы вашей программы, следует воспользоваться командой ***Output*** *(Вывод)* меню ***Window***. На экране при этом появится новое активное окно с соответствующим порядковым номером. Окно будет содержать копию текущего экрана DOS, включая вместе результаты работы и текст любых командных строк DOS.

Для полноэкранного просмотра достаточно раскрыть окно *"**Output**"* на весь экран. Дополнительно ту же возможность предоставляет специальная команда ***User screen*** *(Экран пользователя)* меню *Window* («горячая» клавиша ***Alt+F5***). В отличие от команды ***Output***, отображающей текст в обычном окне ***Borland Pascal***, команда ***User screen*** приводит к полному отображению стандартного экрана DOS, и для возврата в ***IDE*** вам понадобится нажатие кнопки мыши либо нажатие любой клавиши.

7.7 Повторное обращение к программе

С помощью команды ***Open*** *(Открыть)* меню ***File*** ("горячая" клавиша - ***F3***) можно загрузить в окно редактора текст любой программы из списка программ, имеющихся на диске. Во время выполнения команды на экране появится список всех файлов в текущем каталоге. Используя обычные действия, с помощью мыши либо клавиатуры можно легко выбрать файл с требуемой программой.

Дополнительно в ***IDE*** по умолчанию предусмотрена возможность автоматического сохранения на диске списка всех редактируемых файлов после завершения сеанса работы. Если это происходит, то в начале следующего сеанса работы ваша последняя программа также автоматически будет загружена в память.

8 Индивидуальные задания

8.1 Рекомендации по составлению программы

Составление программы целесообразно начать с открытия файла, содержащего заготовку программ, разработанную на предыдущем занятии. Целесообразно откорректировать эту программу-заготовку, добавив комментарии к каждому разделу, поясняющие, что должен содержать этот раздел, а также операторы, формирующие вспомогательные свойства программы (очистку экрана, цвет фона и текста и т.п.). Откорректированную таким образом программу следует сохранить под тем же именем. После этого, чтобы не испортить заготовку программы, следует сохранить ее еще раз в своем каталоге под новым именем, которое Вы пожелаете дать разрабатываемой программе.

При составлении программы обязательно предусмотреть:

- ✓ разумный выбор идентификаторов:
- ✓ многократный ввод данных при исполнении программы, т.е. возможность повторного счета при других исходных данных;
- ✓ простейший диалог типа «запрос-ответ» при вводе данных;
- ✓ необходимые комментарии в тексте программы;
- ✓ вывод результатов в удобном для пользователя виде (отформатированные результаты, размерность, цвет и т.п.);
- ✓ подготовку тестового примера, позволяющего доказать правильность работы Вашей программы.

Внимание! В большинстве случаев вычисления ведутся *не в системе СИ,* а в *согласованной системе единиц*! И в формулах ничего менять не нужно. В то же время, чтобы получить значения реактивных сопротивлений в Омах, необходимо в формулы для расчета индуктивного и емкостного сопротивлений ***все данные подставлять в системе СИ!***

8.2 Варианты заданий для составления линейных программ

1. Рассчитать индуктивность L_0 [нГн] круглого витка со средним диаметром D [см] и диаметром провода d [см] и его реактивное сопротивление X_{L_0} на частоте f по формулам

$$L_0 = 2\cdot\pi\cdot D\cdot\left(ln(8D/d)-1.75\right),\ X_{L_0} = 2\cdot\pi\cdot f\cdot L_0.$$

2. Рассчитать число витков w однослойной тороидальной катушки круглого сечения с индуктивностью L_0 [нГн] и ее реактивное сопротивление X_{L_0} на частоте f по формулам

$$w = \sqrt{L_0 \Big/ \left(2\cdot\pi\cdot\left(D-\sqrt{D^2-D_1^2}\right)\right)},$$ где

D [см] – средний диаметр тора, а D_1 [см] – диаметр сечения тора.

3. Рассчитать число витков *w* однослойной тороидальной катушки прямоугольного сечения с индуктивностью L_0 [нГн] и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = \sqrt{L_0 / (2 \cdot h \cdot ln(D_2 / D_1))}, \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

D_1 [см] – внутренний диаметр тора, а D_2 [см] – внешний диаметр тора, *h* [см] - сторона квадрата сечения тора.

4. Рассчитать число витков *w* многослойной тороидальной катушки круглого сечения с индуктивностью L_0 [нГн] и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = \sqrt{L_0 / (2 \cdot \pi \cdot D \cdot (ln(8 \cdot D / D_1) - 1.75))}, \; X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

D [см] – средний диаметр тора, а D_1 [см] – диаметр сечения тора.

5. Рассчитать индуктивность L_0 [мкГн] однослойной катушки круглого сечения (без сердечника) и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$L_0 = 0.394 \cdot r^2 \cdot w^2 / (9 \cdot r + 10 \cdot l), \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

w - число витков, *r* [см] – радиус катушки, *l* [см] – длина намотки ($l > 0.666 \cdot r$)

6. Рассчитать число витков *w* однослойной катушки круглого сечения (без сердечника) и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = \sqrt{L_0 \cdot (9 \cdot r + 10 \cdot l) / 0.394} / r, \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

L_0 [мкГн] - индуктивность, *r* [см] – радиус катушки индуктивности, *l* [см] – длина намотки ($l \geq 0.666 \cdot r$).

7. Рассчитать число витков *w* однослойной катушки круглого сечения (без сердечника) с индуктивностью L_0 [мкГн] и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = 12.7 \cdot L_0 \cdot \left(1 + \sqrt{\cdot 0.14 \cdot r^3 \cdot p^2 / L_0}\right) \Big/ \left(p \cdot r^2\right), \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

r [см] – радиус катушки, $p = w / l$ – количество витков на единицу длины намотки катушки, *l* [см] (определяется размерами провода и толщиной изоляции).

8. Рассчитать индуктивность L_0 [мкГн] короткой катушки (без сердечника) по формуле и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$L_0 = 0.394 \cdot r^2 \cdot w^2 / (r \cdot (9 - 0.2 \cdot r / l) + 10 \cdot l), \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

w - число витков, *r* [см] – радиус катушки, *l* [см] – длина намотки ($0.1 \cdot r \leq l \leq 0.666 \cdot r$).

9. Рассчитать число витков *w* короткой катушки (без сердечника) по формуле:

$$w = \sqrt{(r \cdot (9 - 0.2 \cdot r / l) + 10 \cdot l) \cdot L_0 / 0.394} / r, \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

L_0 [мкГн] - индуктивность, *r* [см] – радиус катушки, *l* [см] – длина намотки катушки ($0.1 \cdot r \leq l \leq 0.666 \cdot r$).

10. Рассчитать индуктивность L_0 [мкГн] катушки с многослойной обмоткой (без сердечника) и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$L_0 = 0.315 \cdot r^2 \cdot w^2 / (6 \cdot r + 9 \cdot l + 10 \cdot d), \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

w - число витков, *r* [см] – средний радиус катушки, *l* [см] – длина намотки, *d* [см] – радиальная толщина обмотки.

11. Рассчитать число витков *w* катушки с многослойной обмоткой (без сердечника) и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = \sqrt{(6 \cdot r + 9 \cdot l + 10 \cdot d) \cdot L_0 / 0.315} / r, \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

L_0 [мкГн] - индуктивность, *r* [см] – средний радиус катушки, *l* [см] – длина намотки, *d* [см] – радиальная толщина обмотки.

12. Рассчитать число витков *w* катушки с замкнутым сердечником и ее реактивное сопротивление X_{L_0} на частоте *f* по формулам:

$$w = \sqrt{L_0 l_m (1 + \mu d_l / a l_m) / 4\pi \mu F_c}, \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0, \text{ где}$$

L_0 [мкГн] - индуктивность, d_l[см] – средний радиус катушки, l_m [см] – длина средней силовой линии, *a* [см] – толщина сердечника, F_c. [см^2]- площадь поперечного сечения сердечника, μ - магнитная проницаемость.

13. Рассчитать индуктивность L_0 [мкГн] прямого провода длиной *l* [см] и диаметром *d* [см] и его реактивное сопротивление на частоте *f* по формулам:

$$L_0 = 0.002 \cdot l \cdot [2.3 log(4 \cdot l / d - 0.75)], \quad X_{L_0} = 2 \cdot \pi \cdot f \cdot L_0.$$

14. Определить емкость *C* [пФ] бумажного конденсатора с параллельными прямоугольными пластинами и его реактивное сопротивление X_C на частоте *f*:

$$C = \varepsilon \cdot S / (11.3 \cdot d), \quad X_C = 1 / (2 \cdot \pi \cdot f \cdot C), \text{ где}$$

$S = a \cdot b$ [см^2] - площадь пластины, *a* и *b* [см] – размеры пластины, *d* [см] – расстояние между ними, ε - относительная диэлектрическая проницаемость (для бумаги $\varepsilon = 4$).

15. Определить емкость *C* [пФ] коаксиального кабеля длиной *l* [м] и его реактивное сопротивление X_C на частоте *f*:

$$C = 24.5 \cdot \varepsilon \cdot l / log(D/d), \quad X_C = 1 / (2 \cdot \pi \cdot f \cdot C), \text{ где}$$

D [мм] – наружный диаметр кабеля, *d* [мм] – внутренний диаметр кабеля, ε - относительная диэлектрическая проницаемость ($\varepsilon = 2.23$).

16. Определить реактивное X_C и полное Z_C сопротивления конденсатора емкостью *C* с внутренним сопротивлением $R_{посл}$ на частоте *f* (все параметры заданы в системе Си), а также его затухание *d* и добротность *Q*:

$$X_C = 1 / (2 \cdot \pi \cdot f \cdot C), \quad Z_C = \sqrt{R_{посл}^2 + X_C^2}, \quad d = \frac{R_{посл}}{X_C} \cdot 100\%, \quad Q = \frac{1}{d}.$$

17. Определите резонансную частоту f_P, добротность *Q*, ширину полосы пропускания Δf, ток *I* и напряжения $U_{Rпосл}$, U_L и U_C при резонансе на элементах *L* (индуктивность),

C (емкость) и $R_{посл}$ (внутреннее сопротивление) последовательного резонансного контура, питающегося от генератора с напряжением $U_Г$:

$$f_p = \frac{1}{2\pi\sqrt{LC}},\ I = \frac{U_Г}{R_{посл}},\ Q = \frac{2\pi fL}{R_{посл}},\ U_L = U_C = U_Г \cdot Q,\ \Delta f = \frac{f_p}{Q}.$$

18. Вычислить медианы треугольника со сторонами *a, b, c* по формулам:

$$m_a = 0.5 \cdot \sqrt{2b^2 + 2c^2 - a^2},\ m_b = 0.5 \cdot \sqrt{2a^2 + 2c^2 - b^2},$$
$$m_c = 0.5 \cdot \sqrt{2a^2 + 2b^2 - c^2}.$$

19. Вычислить биссектрисы треугольника со сторонами *a, b, c* по формулам:

$$\beta_a = 2 \cdot \sqrt{b \cdot c \cdot p \cdot (p-a)} / (b+c),\ \beta_b = 2 \cdot \sqrt{a \cdot c \cdot p \cdot (p-b)} / (a+c),$$
$$\beta_c = 2 \cdot \sqrt{a \cdot b \cdot p \cdot (p-c)} / (b+a),\ \text{где}\ p = (a+b+c)/2.$$

20. Вычислить координаты центра тяжести трех материальных точек с массами m_1, m_2, m_3 и координатами (x_1, y_1), (x_2, y_2), (x_3, y_3) по формулам:

$$x_c = (m_1 \cdot x_1 + m_2 \cdot x_2 + m_3 \cdot x_3) / (m_1 + m_2 + m_3),$$
$$y_c = (m_1 \cdot y_1 + m_2 \cdot y_2 + m_3 \cdot y_3) / (m_1 + m_2 + m_3).$$

21. Вычислить координаты точки, делящей отрезок a_1a_2 в отношении n_1: n_2 по формулам:

$$x = (x_1 + \lambda \cdot x_2)/(1+\lambda),\ y = (y_1 + \lambda \cdot y_2)/(1+\lambda \cdot),\ \text{где}\ \lambda = n_1/n_2.$$

22. Вычислить площадь поверхности $S = \pi \cdot (R+r) \cdot l + \pi \cdot R^2 + \pi \cdot r^2$ и объем $V = \pi \cdot h \cdot (R^2 + r^2 + Rr)/3$ усеченного конуса.

23. Составить программу для вычисления расстояний между двумя точками с координатами (x_1, y_1, z_1) и (x_2, y_2, z_2) в трехмерном пространстве по формуле:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

24. Составить программу для вычисления значений функций:

$$y = \frac{e^{-x_1} + e^{-x_2}}{2} \text{ и } z = \frac{a\sqrt{x_1} - b\sqrt{x_2}}{c},\ \text{где}$$
$$x_1 = \left(b + \sqrt{|b^2 - 4ac|}\right)/(2a),\ x_2 = \left(b - \sqrt{|b^2 - 4ac|}\right)/(2a).$$

9 Список литературы

1. **Рапаков Г.Г., Ржеуцкая С.Ю.** *Программирование на языке Pascal.* — СПб. : БХВ-Петербург, 2004. - 480 с.

2. **Фаронов , В.В.** *Программирование на персональных ЭВМ в среде Турбо-Паскаль. :, .* - М. : Изд-во МГТУ, 1990.-580 с.

3. **Павловская , Т.А.** *Паскаль. Программирование на языке высокого уровня: Учебник для вузов.* — СПб. : Питер, 2007. — 393 с.

4. **Фаронов , В.В.** *Турбо Паскаль 7.0. Начальный курс. Учебное пособие.* -М. : ОМД Групп, 2003. - 616 с.

5. **Рютген Т., Франкен Г.** *Турбо Паскаль 7.0 .* — К. : Торгово-издательское бюро BHV, 1996-448 с.

6. **Попов, В.Б.** *Паскаль и Дельфи. Самоучитель.* — СПб. : Питер, 2004. — 544 с.