Министерство образования Республики Беларусь
Белорусский национальный технический университет
Механико-технологический факультет
Кафедра «Материаловедение в машиностроении»

Краткий курс программирования на языке Pascal ABC

для студентов специальности
1 - 36 01 02 Материаловедение в машиностроении

Электронный учебный материал

В.В. Мельниченко

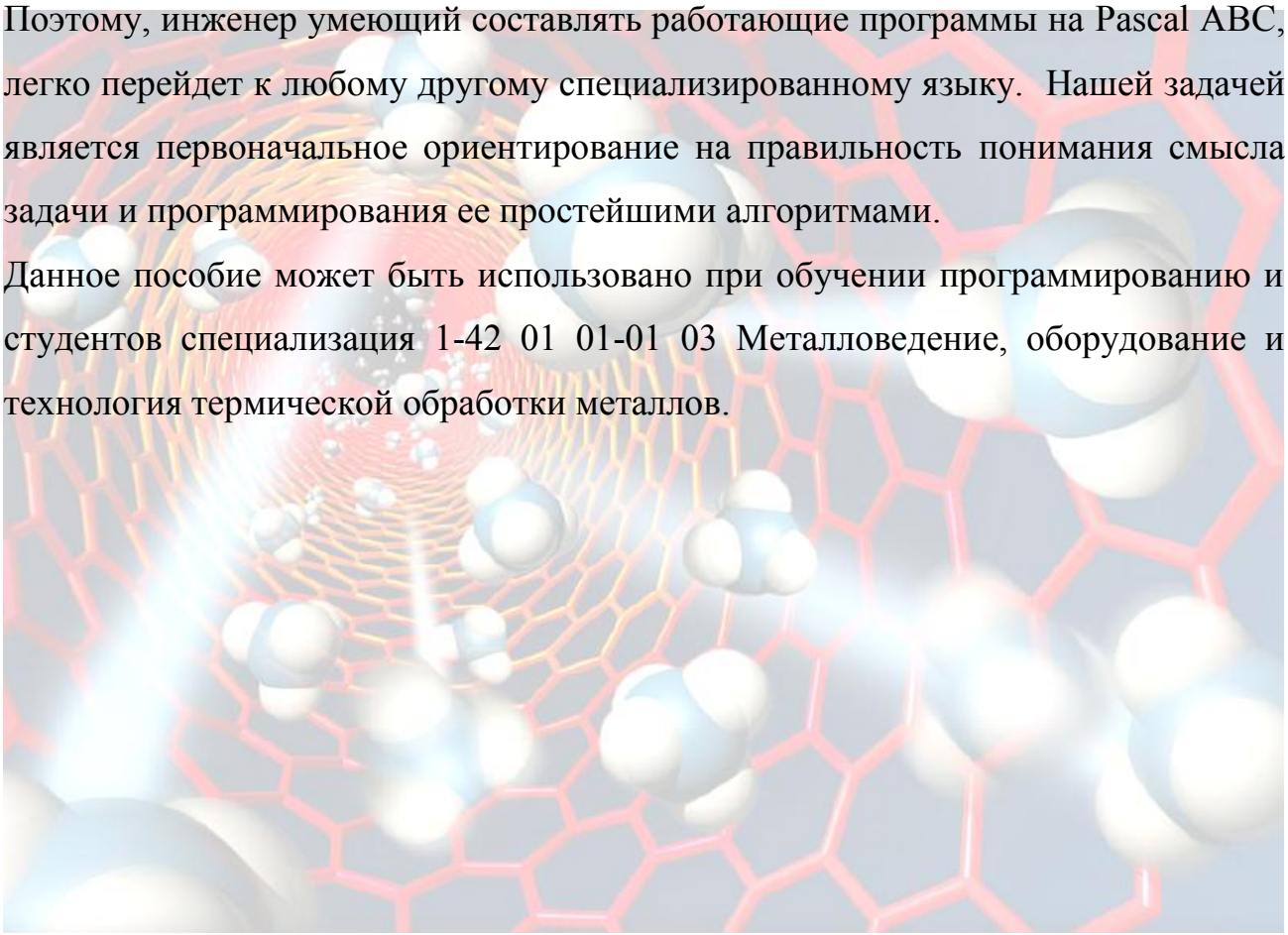
Минск 2020

Краткое введение

На вопрос почему выбран именно язык программирования Pascal, можно ответить просто. Мы готовим инженеров, а не программистов. Практически все используемые языки программирования выросли именно из языка Pascal, который постоянно развивается и дает импульс в понимании современного программирования.

Поэтому, инженер умеющий составлять работающие программы на Pascal ABC, легко перейдет к любому другому специализированному языку. Нашей задачей является первоначальное ориентирование на правильность понимания смысла задачи и программирования ее простейшими алгоритмами.

Данное пособие может быть использовано при обучении программированию и студентов специализация 1-42 01 01-01 03 Металловедение, оборудование и технология термической обработки металлов.



Оглавление

1. Среда языка программирования Pascal ABC

Первая версия языка Паскаль была разработана в 1968 году. Ее разработчиком является швейцарский ученый Никлаус Вирт. Свое название язык получил в честь создателя первой механической вычислительной машины француза Блеза Паскаля. На основе языка Паскаль в 1985 г. фирма Borland выпустила версию Turbo Pascal версии 3.0. С этого времени язык Паскаль используется во всем мире в учебных заведениях в качестве первого изучаемого языка программирования.

Система Pascal ABC основана на языке Delphi Pascal и призвана осуществить постепенный переход от простейших программ к объектно-ориентированному программированию.

Составление последовательности команд для решения конкретных задач на языке программирования называется разработкой программ, либо **программированием**.

Для вызова среды программирования Pascal ABC необходимо запустить на выполнение файл PascalABC.exe или загрузить среду посредством ярлыка, если он существует на рабочем столе.

Для создания нового файла необходимо выполнить следующие действия: **Файл/Новый** или нажать кнопку  на панели инструментов**.** На экране откроется чистое окно с именем Program1.pas. , его при сохранении файла желательно изменить.

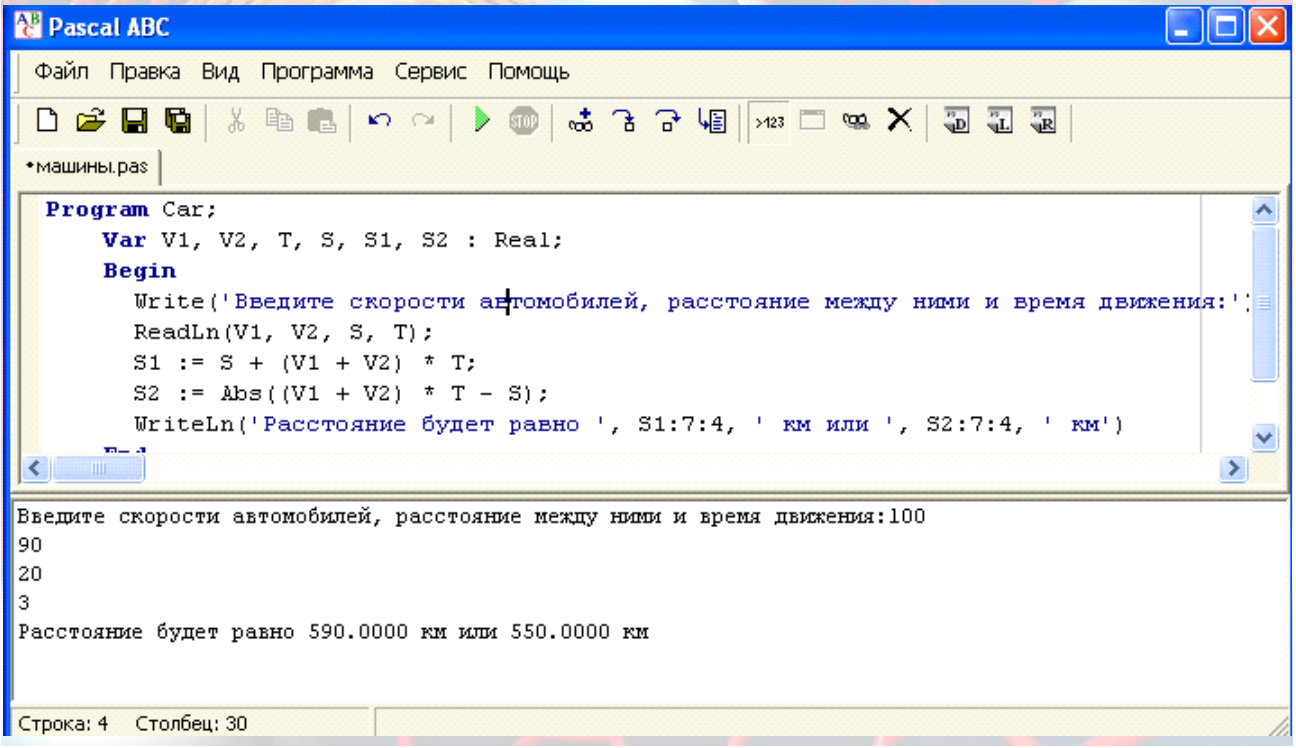
При вводе и редактировании текста программы используются такие же приемы, как и при работе в текстовом редакторе. С помощью команды **Помощь/Содержание** можно ознакомиться со справочной системой Pascal ABC.

Для сохранения программы необходимо выполнить команду **Файл/Сохранить как…**имя файла, или нажать кнопку  на панели инструментов, если необходимо сохранить существующую уже программу.

Для загрузки программы из файла необходимо выполнить команду **Файл/Открыть** выбрать файл или нажать кнопку  на панели инструментов.

Для выполнения программы необходимо выполнить команду **Программа/Выполнить**, либо нажать клавишу F9, либо нажать кнопку  .

Если в программе отсутствуют ошибки, на экране монитора можно увидеть результат выполнения программы. При обнаружении ошибок курсор устанавливается в области ошибки, а в окне вывода выдается сообщение об ошибке. В этом случае следует исправить ошибки и снова выполнить программу.



Наиболее часто встречающиеся ошибки при работе в системе программирования Pascal ABC:

- «неожиданный символ» - символ введен не с регистра английских букв;
- «ожидался символ "точка", "точка с запятой", Begin и др.»- отсутствует необходимый символ или команда;

- «неизвестное имя» - неправильно введена команда;
- «ошибка ввода» - неверно указано имя файла;
- «ожидался конец файла» - ошибки в команде end.

Структура программы

Программа, записанная на языке Pascal может содержать следующие разделы:

1. Заголовок (Program)
2. Раздел меток (Label)
3. Раздел констант (Const)
4. Раздел типов (Type)
5. Раздел переменных (Var)
6. Раздел процедур и функций (Procedure, Function).
7. Раздел операторов (Begin...End)

Все программы обязательно имеют раздел заголовка и раздел операторов, остальные составляющие могут отсутствовать. При отсутствии некоторых частей программы общий порядок их следования сохраняется. Разделы между собой разделяются знаком ";" Раздел операторов заключается в операторные скобки. Это зарезервированные слова Begin, End. Раздел операторов заканчивается точкой. Раздел "заголовок" начинается с зарезервированного слова program, за которым указывается имя программы. В качестве имени может использоваться любой набор символов алфавита с несколькими исключениями:

1. нельзя использовать зарезервированные слова;
2. нельзя начинать имя с цифры;
3. при использовании имени не используется пробел.

Программа на языке Pascal может иметь следующий вид:

Program <имя программы>;

<раздел описаний, в котором описываются данные>

Begin <раздел команд (тело программы)>;

End.

2. Алфавит, типы данных языка программирования Pascal

Алфавит языка программирования Pascal включает:

- Латинские буквы: A a B b… Z z
- Цифры: 0 1 2..9
- Знаки математических операций: +(сложение) -(вычитание) *(умножение) /(деление)
- Знаки математических отношений: < > = <=(знак меньше или равно) >=(больше или равно) <>(не равно)
- Специальные знаки: **[ ] . , ( ) : ; ^ { } $ # @**

Для программной обработки данные представляются в виде величин и их совокупностей. С понятием величины связаны следующие характеристики (атрибуты):

- имя (идентификатор) - это ее обозначение и место в памяти. Имя переменной (идентификатор) всегда должно начинаться с латинской буквы, после которой могут следовать несколько латинских букв, цифры либо символ подчеркивания «_», записанные без пробелов;
- тип - множество допустимых значений и множество применимых операций к ней;
- значение - характеристика, которая может меняться многократно в ходе исполнения программы.

Постоянной (константа) называется величина, значение которой не изменяется в процессе исполнения программы. Константы должны объявляться

в разделе констант, начиная с зарезервированного слова Const. В языке программирования Pascal константами являются любые явно заданные в программе данные.

Пример. Const year=2009;

При записи числовых констант с дробной частью эта часть отделяется от целой не запятой, а точкой.

Пример. Const Pi=3.14;

Для записи очень больших по модулю или очень малых (близких к нулю) чисел существует возможность записи их в так называемой экспоненциальной форме.

Пример: Const a=2.4567E-06

В вещественных константах (они соответствуют действительным числам в математике) присутствует точка, которая разделяет целую и дробную части числа, или буква E. Использование E приводит к представлению числа в виде с плавающей запятой (точкой): запись mEp, соответствует числу $m \cdot 10^p$.

Константы, представляющие собой строковые величины, заключаются в апострофы.

Пример. Const Name='Татьяна';

В качестве данных в программах на языке Pascal могут выступать числа, символы, целые строки символов. С этими видами информации выполняются совершенно разные действия. Например, с числовыми величинами производятся арифметические операции, чего невозможно сделать с символьными. Кроме того, разные виды данных требуют различного объема памяти для хранения. В соответствии с этими соображениями в языке Pascal введено понятие "Тип" (Type). Тип переменной указывает на то, какие данные могут быть сохранены в этом участке памяти, и в каких действиях эта переменная может участвовать.

Переменной называется величина, значение которой меняется в процессе исполнения программы. Имена переменных перечисляются в разделе описания переменных Var через запятую, затем ставится двоеточие и указывается тип данных.

Пример. Var

A,K: Real; B : Integer; C : Char;

В памяти компьютера можно хранить числовые типы, символы, слова, предложения и другие данные. Физически типы данных отличаются друг от друга количеством ячеек памяти (байтов), отводимых для хранения соответствующей переменной.

Различают переменные следующих простых типов: целые (Integer, Byte, ShortInt, Word, LongInt), вещественные (Real, Comp, Double, Single, Extended), логический (Boolean), символьный (Char).

Тип	Длина(байт)	Диапазон значений	Операции
Целые типы			
integer	2	-32768..32767	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
byte	1	0..255	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
word	2	0..65535	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
shortint	1	-128..127	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
longint	4	-2147483648..2147483647	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
Вещественные типы			

real	6	$2,9*10^{-39}$ - $1,7*10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
single	4	$1,5*10^{-45}$ - $3,4*10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
double	8	$5*10^{-324}$ - $1,7*10^{308}$	+, -, /, *, >=, <=, =, <>, <, >
extended	10	$3,4*10^{-4932}$ - $1,1*10^{4932}$	+, -, /, *, >=, <=, =, <>, <, >
comp	8	$9.2*10^{18}..9.2*10^{18}$	+, -, /, *, >=, <=, =, <>, <, >
Логический тип			
boolean	1	true, false	Not, And, Or, Xor, >=, <=, =, <>, <, >
Символьный тип			
char	1	все символы кода ASCII	+, >=, <=, =, <>, <,

3. Оператор присваивания. Арифметические операции и стандартные функции

Оператор присваивания - один из самых простых и наиболее часто используемых операторов в любом языке программирования, в том числе и в Pascal. Он предназначен для вычисления нового значения некоторой переменной, а также для определения значения, возвращаемого функцией. В общем виде оператор присваивания можно записать так: переменная:= выражение;

Оператор выполняется следующим образом. Вычисляется значение выражения в правой части присваивания. После этого переменная, указанная в

левой части, получает вычисленное значение. При этом тип выражения должен быть совместим по присваиванию с типом переменной. Тип выражения определяется типом операндов, входящих в него, и зависит от операций, выполняемых над ними.

Для операций сложения, вычитания, умножения и деления тип результата в зависимости от типа операнда будет таким:

Операнд 1	Операнд 2	Результат сложения, вычитания, умножения	Результат деления
Integer	Integer	Integer	Real
Integer	Real	Real	Real
Real	Integer	Real	Real
Real	Real	Real	Real

В Pascal существуют арифметические операции: +(сложение), -(вычитание), *(умножение), /(деление), DIV (целочисленное деление), MOD (остаток от деления).

Пример: X := (Y + Z) / (2 + Z * 10) - 1/3;

Пример. При выполнении целочисленного деления (операция DIV) остаток от деления отбрасывается: 15 div 3 = 5; 18 div 5 = 3; 7 div 10 = 0.

Пример. С помощью операции MOD можно найти остаток от деления одного целого числа на другое: 15 mod 3 = 0;18 mod 5 = 3; 7 mod 10 = 7.

Пример. Программа с использованием операций с целочисленными переменными.

```
Program summa;
Var a,b,s,c,p: integer; {раздел описания переменных}
Begin      {начало тела программы}
  a:=12;   {команда присваивания записывает в переменную a число 12}
  b:=5;    {команда присваивания записывает в переменную b число 5}
```

```
{переменные A и  B являются исходными данными}
  s:=a+b;  {команда присваивания вычисляет сумму значений переменных a и b и записывает результат в переменную s}
  Writeln('a + b=',s) {вывод значения переменной s на экран - вывод результата}
  c:=a div b;  {деление нацело (вычисление целой части  от деления a на b)}
  writeln('a div b=',c); {вывод на экран частного от деления нацело a на b}
  p:=a mod b;  {вычисление остатка от деления a на b}
  Writeln ('a mod b=',p); {вывод на экран остатка от деления a на b}
End.       {конец  программы}
```

После выполнения программы в окне вывода результата появится:

```
a + b=17
a div b=2
a mod b=2
```

Логический операнд - это конструкция, которая задает правило для вычисления одного из двух возможных значений: True или False. Чаще всего логические выражения используют в операторах присваивания или для записи условия, чтобы на некоторый вопрос получить ответ “ДА” или “НЕТ”. Составными частями логических выражений могут быть: логические значения (True, False); логические переменные; отношения, операции: Not(НЕ), And(И), Or(ИЛИ), Xor(исключающее ИЛИ).

Пример. 1) Y:=True; 2) LogPer:=A > B; 3) Log1:=(A = B) And (C <= D).

Арифметические выражения - это конструкции результатом, которых является число. В состав арифметического выражения на языке Паскаль могут входить:

- числовые константы;
- имена переменных;
- знаки математических операций;

- математические функции и функции, возвращающие число;
- открывающиеся и закрывающиеся круглые скобки.

При составлении выражений могут быть использованы следующие арифметические функции:

Имя функции	Математическое значение	Тип результата
abs (a)	|a|- модуль числа	Совпадает с типом аргумента
sqr (a)	a^2 возведение в квадрат	Совпадает с типом аргумента
sqrt (a)	$\sqrt{a}$ - квадратный корень	Вещественное
sin (a)	sin a - синус x радиан	Вещественное
cos (a)	cos a - косинус x радиан	Вещественное
arctan (a)	arctg a - арктангенс числа x	Вещественное
ln (a)	ln a- натуральный логарифм x	Вещественное
exp (a)	e^a значение e в степени x	Вещественное
trunc(x)	$[x]$ - целая часть числа x	Целое
frac(x)	$\{x\}$ - дробная часть числа x	Вещественное
Pi	π - число	Вещественное
Round (x)	Округление до ближайшего целого	Целое
Int (x)	[x] целая часть числа	Вещественное

Порядок действий при вычислении значения выражения:

1) вычисляются значения в скобках;
2) вычисляются значения функций;
3) выполняется операции смена знака, возведение в степень;
4) выполняются операции умножения и деления (в том числе целочисленного деления и нахождения остатка от деления);
5) выполняются операции сложения и вычитания.

Пример. Программа с использованием стандартных арифметических функций.

```
Program fun;
Var a, c :integer; {раздел описания переменных целого типа}
 b,:real;  {раздел описания переменных вещественного типа}
Begin      {начало тела программы}
   a:=16;       {исходные данные}
   b:= sqrt (a);  {функция sqrt извлекает квадратный корень из числа a и
присваивает его значение переменной b - тип результата вещественный}
writeln('b=',b);{вывод на экран значения переменной b}
   c:=sqr (a);  {функция sqr возводит в квадрат значение переменной a и
присваивает его  переменной c - тип результата целый}
 writeln('c=',c); {вывод на экран монитора значения переменной c}
End.
```

Аргументы функции всегда пишутся в скобках, операцию умножения опускать нельзя.

Нельзя писать выражения в виде обыкновенных дробей.

Примеры записи математических выражений:

Математическая запись	Запись на Pascal
1. x2 - 7,2x + 6	Sqr(x) - 7.2 * x + 6
2. $\frac{\lvert x\rvert - \lvert y\rvert}{1+\lvert xy\rvert}$	(Abs(x) - Abs(y)) / (1 + Abs(x * y))

4. Организация ввода и вывода данных

Обмен информацией с компьютером предполагает использование определенных средств ввода-вывода. Основным средством ввода является клавиатура, вывода - дисплея. Процедура, которая в режиме диалога с

клавиатуры присваивает значение для переменной величины, называется процедурой ввода. В языке Pascal эта команда выглядит следующим образом:

Read(список переменных);

ReadLn(список переменных);

Разница между работой процедур Read и Readln состоит в следующем: после выполнения Read значение следующего данного считывается с этой же строчки, а после выполнения Readln - с новой строки.

Выполнение операторов ввода происходит так: ход программы приостанавливается, на экран выводится курсор, компьютер ожидает от пользователя набора данных для переменных, имена которых указаны в списке ввода. Пользователь с клавиатуры вводит необходимые значения в том порядке, в котором они требуются списком ввода, нажимает Enter. После этого набранные данные попадают в соответствующие им переменные и выполнение программы продолжается.

Пример.

```
Var A : real; B : integer; C : char;
Begin
  Read(A, B, C)
End.
```

Процедура, которая выводит содержимое переменных на экран, называется процедурой вывода на экран. В Pascal эта команда выглядит следующим образом:

Write (список констант и/или переменных, разделенных запятой)

WriteLn(список констант и/или переменных, разделенных запятой)

Различие между двумя операторами вывода: после выполнения оператора Writeln происходит переход на новую строчку, а после выполнения инструкции Write, переход на новую строчку не происходит. При вызове оператора Writeln без параметров просто происходит переход на новую строчку.

Пример. Write(A, B, C);

WriteLn('Корнем уравнения является ', X);

Для управления размещением выводимых значений процедуры Write и WriteLn используются форматный вывода. Под форматом данных понимается расположение и порядок отдельных полей данных.

Процедура вывода с форматом для целого типа имеет вид:

WriteLn(A:N); где N - выражение целого типа, задающие длину поля вывода значений.

При выводе вещественных значений без указания формата - выводится вещественное число длиной 18 символов в форме с плавающей запятой. Для десятичного представления значения применяется оператор с форматами вида:

WriteLn(R:N:M); где N - выражение целого типа, задающие длину поля вывода значений, M- количество знаков в дробной части.

Пример:

A:=25; Write (A); результат 25

A:=25.367; Write (A:6:3); результат 25.367

Write (A:6:2); результат 25.37

A:=-7.4385 Write (A); результат -7.438500E+00

S:=7; Write (S,'=',5+2); - результат 7=7

Write ('S=5+2'); - вывод текста S=5+2, заключенного в апострофы.

5. Реализация линейных алгоритмов

Алгоритм, в котором команды выполняются последовательно друг за другом, называется **линейным**.

Пример: Ввод трех целых чисел, вычисление и вывод их среднего арифметического и среднего геометрического значения.

```
Program Z;
Var a1,a2,a3:integer;
  sa,sg:real;
Begin
   Writeln('Введи 3 целых числа:'); {Печать на экране просьбы о вводе}
   Read(a1,a2,a3);{Ввод данных}
   sa:=(a1 + a2 + a3) / 3;  {Вычисление среднего арифметического}
   sg:=sqrt(abs(a1*a2*a3)); {Вычисление среднего геометрического}
   Writeln ('Ср.арифм.=',sa:8:2);
   Writeln ('Ср.геом.=',sg:8:2) {Вывод данных}
End.
```

Пример: Вычисления площади круга. Программа запрашивает у пользователя значение радиуса круга, обеспечивает возможность ввести его значение, рассчитывает и выводит на экран величину площади круга с введенным радиусом. Таким образом, появляется возможность, не внося изменений в программу, вводить различные значения радиуса и получать, соответствующие им значения площади круга. Для этого достаточно несколько раз запустить программу.

```
Program Inteface;
  Var R,S: Real;
 Begin
   Write('Введите радиус круга '); {Печать на экране просьбы о вводе}
   Readln(R); {Ввод значения R}
   S:=Pi*SQR(R); {Вычисление площади руга}
   Writeln('Площадь круга радиусом ',R:5:2,' равна ',S:8:4)
End.
```

Пример. Скорость первого автомобиля v_1 км/ч, второго - v_2 км/ч, расстояние между ними s км. Какое расстояние будет между ними через t ч, если автомобили движутся в разные стороны?

Согласно условию задачи искомое расстояние $s_1=s+(v_1+v_2)t$ (если автомобили изначально двигались в противоположные стороны) или $s_2=|(v_1+v_2)t-s|$ (если автомобили первоначально двигались навстречу друг другу).

```
Program Car;
Var V1, V2, T, S, S1, S2 : Real;
Begin
 Write('Введите скорости автомобилей, расстояние между ними и время движения:');
 ReadLn(V1, V2, S, T);
 S1 := S + (V1 + V2) * T;
 S2 := Abs((V1 + V2) * T - S);
 WriteLn('Расстояние будет равно ', S1:7:4, ' км или ', S2:7:4, ' км')
End.
```

Пример. Найти сумму цифр двухзначного натурального числа.

```
Program Did;
Var A,S1,S2:integer;
Begin
 Write ('введите двухзначное число: ');
 ReadLn (A);      {ввод с клавиатуры числа}
 S1:=a div 10;    {нахождение первой цифры числа}
 S2:=a mod 10;    {нахождение последней цифры числа}
 WriteLn ('сумма цифр числа ',a, '=', S1+ S2) {вывод результата}
End.
```

Пример.Дано: c, m, S - часы, минуты, секунды. Найти общее количество секунд с начала суток.

Формула: Sec=c*3600+m*60+s

```
Program Second;
  Var c, m, s : Byte ; Sec: LongInt ;
Begin
  Write ('Сколько часов прошло от начала суток? '); Readln (c);
  Write ('Сколько минут? '); Readln (m);
  Write ('Сколько секунд? '); Readln (s);
  Sec :=c*3600+m*60+s;
  Writeln ('С начала суток прошло секунд: ', Sec)
End.
```

Пример. Дано: координаты точки (X,У). Получить сообщение TRUE, если точка принадлежит первой четверти окружности с радиусом 1, или сообщение FALSE, если точка не принадлежит области.

Формула: точка с координатами (X,У) принадлежит первой четверти окружности с радиусом 1 если одновременно выполняются условия: $X>0$, $Y>0$, $X^2+Y^2<=1$

```
Program FH;
  Var X,Y : Real;
Begin
  Writeln (‘введи координаты точки: X, Y:’); Readln (X,Y);
  Writeln ((X1>=0) AND (Y1>=0) AND (SQR(X)+SQR(Y)<=1));
End.
```

Задания для практической работы по теме линейные алгоритмы.

Составить программы на языке Pascal.

1. Даны два числа. Найти сумму и произведение этих чисел.
2. Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей.

3. Вычислить значение выражения $\frac{|x|-|y|}{1+|xy|}$, где x и y - целые числа.
4. Периметр прямоугольника равен p, одна из сторон равна a. Найти его площадь.
5. Гипотенуза прямоугольного треугольника равна c, острый угол a градусов.
6. Найдите площадь квадрата, если две его противоположные вершины заданы координатами (x1,y2) ,(x2,y2).
7. Определить сколько времени (t) затрачено на путь (s) со скоростью (v).
8. Вершины треугольника заданы точками (x1,y2) ,(x2,y2), (x3,y3). Используя формулу Герона, найдите его площадь.(AB=Sqrt((x1-x2)^2+(y1-y2)^2).
9. Сумма вклада S руб., ее вкладывают в банк под C % годовых на N лет. Какая прибыль будет на вкладе.
10. Подсчитать стоимость поездки на S км. : расход бензина на 100 км.-L литров, стоимость 1 литра бензина K руб.
11. Вычислить стоимость покупки со скидкой : покупка A руб., скидка C %.
12. Даны координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь.
13. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T1 ч, а по реке (против течения) - T2 ч. Определить путь S, пройденный лодкой.
14. Найти периметр и площадь прямоугольного треугольника, если даны длины его катетов a и b.
15. Дано целое четырехзначное число. Используя операции div и mod, найти сумму его цифр.
16. Дано целое четырехзначное число. Используя операции div и mod, найти произведение его цифр.
17. Длительность некоторого физического эксперимента измеряется в секундах; определить количество часов(полных), минут и секунд.

18. С начала месяца прошло m часов. Определить какое сейчас число.
19. С начала суток прошло k минут. Определить который сейчас час. (В часах и минутах).
20. Дано: координаты точки (X,У). Получить сообщение TRUE, если точка принадлежит 2,3,4 четверти окружности с радиусом R, или сообщение FALSE, если точка не принадлежит области.

6. Алгоритмическая конструкция ветвление

На практике решение большинства задач не удается описать с помощью программ линейной структуры. При этом после проверки некоторого условия выполняется та или иная последовательность операторов, однако происходит нарушение естественного порядка выполнения операторов. Условный оператор для реализации разветвлений в программе имеет следующую структуру:

If <логическое выражение> Then серия1 Else серия2;

Если логическое выражение, выступающее в качестве условия, принимает значение True (истинно), то выполняются операторы следующие за then (серия1), если False (ложь), то выполняются операторы расположенные после else (серия2).

Поскольку развилка может быть неполной, то возможна и неполная форма записи условного оператора:

If <логическое выражение> Then серия;

Когда выполняется последовательность команд (серия), необходимо использовать так называемые операторные скобки:

```
Begin
<Оператор 1>;
<Оператор 2>;
   ...
```

<Оператор N>

End;

Точка с запятой после Begin не ставиться.

Вариант условного оператора в этом случае:

If <условие> Then Begin <группа операторов 1> End;

Else Begin < группа операторов 2> End;

Знак "точка с запятой" не ставится перед служебным словом Else, но операторы в группах отделяются друг от друга этим знаком.

На языке Паскаль условия представляют собой выражения, значением которых является величина логического типа (True - истина или False - ложь). Это может быть как просто переменная указанного типа, так и сложная последовательность высказываний, связанных логическими операциями.

В простых условиях могут применяться знаки операций сравнения: >(больше), <(меньше), =(равно), <>(не равно), >=(больше или равно), <=(меньше или равно).

Пример. (C+D3)>=(D1*(45-2))

В сложных условиях логические выражения строятся с помощью логических операций: and (и), or (или), not (не).

Примеры: (a>b) and (a>c); (X<=1) or (Y>5) and (a>2); (4*X-1>0) and (X+2<3); not (a=b).

Операции сравнения имеют низший приоритет по сравнению с другими операциями, поэтому отношения в логических выражениях заключают в скобки.

Пример. Из двух чисел необходимо выбрать наибольшее.

```
Program Example;
Var A,B,C : Real; {A,B - для хранения чисел, C - результат}
Begin
  Writeln('Введите два числа');
  Readln(A,B);  {Ввод чисел с клавиатуры}
```

```
If A>B Then C:=A Else C:=B; {Если A>B, то результат A сохраняем в C,
```
иначе результат - B}

```
Writeln('Наибольшее число =',C);  {Выводим результат на экран}
End.
```

Пример. По заданным коэффициентам решить квадратное уравнение

```
Program Sq1;
Var A, B, C, D, X1, X2 : Real;
 Begin
  Writeln ('Введите коэффициенты квадратного уравнения');
  Readln (A,B,C);
  D:=B*B-4*A*C; {вычисление дискриминанта}
  If D<0 Then Writeln ('Корней нет! ')
  Else Begin
  X1:=(-B+SQRT(D))/2/A;
  X2:=(-B-SQRT(D))/2/A;
  Writeln ('X1=', X1:8:3, ' X2=',X2:8:3)
  End
End.
```

Условный оператор реализует разветвление вычислительного процесса по двум направлениям, одно из которых осуществляется при выполнении условия, другое - в противном случае. Для реализации разветвлений более чем по двум направлениям необходимо использовать несколько условных операторов. В этом случае говорят о вложенности условных операторов. В случае вложенных ветвлений каждое новое ключевое слово Else относится к ближайшему If.

Пример. Дано действительное число a. Вычислить f(a), если

$$f(x) = \begin{cases} 0 \text{ при } x \le 0, \\ x^2 - x \text{ при } 0 < x \le 1, \\ x^2 - \sin \pi x^2 \text{ придругихx}. \end{cases}$$

```
Program Usl1;
 Var x, F : Real;
 Begin
  WriteLn('Введите действительное число: ');  ReadLn(x);
   If x <= 0 Then F: = 0 Else
        If x <= 1 THEN F: = Sqr(x) - x  Else F = Sqr(x) - Sin(Pi * Sqr(x));
  WriteLn('Значение функции F(‘,x:5:2,’) = ', F:10:4);
 End.
```

Кроме условного оператора в качестве управляющей структуры довольно часто используется оператор выбора CASE. Эта структура позволяет переходить на одну из ветвей в зависимости от значения заданного выражения (селектора выбора). Ее особенность состоит в том, что выбор решения здесь осуществляется не в зависимости от истинности или ложности условия, а является вычислимым. Оператор выбора позволяет заменить несколько операторов развилки (в силу этого его ещё называют оператором множественного ветвления).

В конструкции CASE вычисляется выражение K и выбирается ветвь, значение метки которой совпадает со значением K. После выполнения выбранной ветви происходит выход из конструкции CASE. Если в последовательности нет метки со значением, равным K, то управление передается внешнему оператору, следующему за конструкцией CASE (в случае отсутствия альтернативы ELSE; если она есть, то выполняется следующий за ней оператор, а уже затем управление передается внешнему оператору).

Запись оператора выбора

```
Case K OF
  A1 : серия 1;
  A2 : серия 2;
   ...
  AN : серия N
```

```
Else  серия N + 1
End;
```

Любая из указанных серий операторов может состоять как из единственного оператора, так и нескольких (в этом случае, как обычно, операторы, относящиеся к одной метке, должны быть заключены в операторные скобки Begin…End).

Выражение К здесь может быть любого порядкового типа (к таким типам относятся все целые типы, Boolean, Char).

Пример. Найти наибольшее из двух действительных чисел используя, оператор выбора.

```
Program Maximum;
Var Max, X, Y : Real;
Begin
  Write('Введите два неравных числа:');
  ReadLn(X, Y);
  Case X > Y Of
  TRUE  : Max := X;
  FALSE : Max := Y
  End;
  WriteLn('Максимальное из двух есть ', Max : 12 : 6)
End.
```

Задания для практической работы.

Составить программы на языке Pascal.

1. Дано число к- целое. Определить, является ли оно четным (нечетным).
2. Выяснить, имеет ли решение уравнение вида A*x+B=0 в зависимости от параметра А.
3. Вычислить значение функции: $f(x) = \begin{cases} x*5 & \text{,если } x<=-1 \\ x/5 & \text{,если } x>=-1 \end{cases}$

4. Вычислить значение функции: $f(x) = \begin{cases} x*5 & ,\text{если } x<=-1 \\ x/5 & ,\text{если } x>=-1 \\ x^2 & ,\text{если } -1<x<1 \end{cases}$
5. Даны три числа A, B и C. Значение наибольшего из них удвоить.
6. Даны три числа A, B и C. Выяснить существует ли треугольник с длинами сторон A,B,C.
7. Дано число K- целое, трехзначное. Найти сумму цифр этого числа.
8. Даны три числа A , B и C . Выдать их в порядке возрастания.
9. Составьте алгоритм, который по номеру дня недели выдает его название.
10. Составьте алгоритм, который по номеру месяца выдает пору года.
11. Составьте алгоритм, который по кол-ву лет школьника определяет, в каком он учится классе.

7. Алгоритмическая конструкция повторение (цикл)

С помощью операторов повторения (цикла) организуется многократное выполнение повторяющихся действий. Если число повторений цикла известно или может быть вычислено, то целесообразно использовать оператор цикла "с параметром"(For). Если же момент завершения цикла зависит от выполнения некоторого условия, то применяются операторы "Пока" и "До" (While, Repeat). Друг от друга циклы отличаются структурой и используются каждый для своего класса задач.

С помощью оператора For осуществляется циклическое выполнение последовательности действий, управляемой переменной цикла, которой присваиваются последовательно возрастающие или убывающие значения.

Оператор имеет два варианта записи:

1. For <переменная цикла> := <начало > to <конец> do < оператор>

(С возрастанием переменной цикла)

После вычисления и проверки начального и конечного значений переменной цикла (начало <= конец), выполняются операторы, образующие тело цикла. Далее значение переменной цикла увеличивается на единицу, и процесс, включающий проверку и выполнение операторов, повторяется. Если переменная цикла превышает конечное значение, то происходит выход из цикла и выполняется оператор, следующий за структурой For.

2. For <переменная цикла> := <начало > downto <конец> do < оператор>

(С убыванием переменной цикла**)**

В данной конструкции отличие в том, что переменная цикла при каждом повторении уменьшается на единицу, поэтому начальное значение переменной должно быть больше конечного.

В операторе цикла Repeat используется условие логического типа, управляющее выходом из цикла, которое проверяется после выполнения операторов цикла.

Оператор имеет следующий вид:

Repeat <последовательность операторов>

Until<условие>

Последовательность операторов выполняется до тех пор, пока <условие > не выполнится. В этом случае цикл завершается, и происходит переход на следующий оператор.

В операторе While в отличие от Repeat, используется условие, служащее индикатором выхода из цикла и проверяется до выполнения последовательности операторов, содержащихся внутри цикла.

Оператор имеет следующий вид:

While <условие> do <оператор>

Оператор выполняется дот тех пор, пока условие остается верным, если условие принимает ложное значение, то цикл заканчивается и происходит переход на следующий оператор.

Пример. Найти произведение первых N натуральных чисел с использованием трех циклов.

1. For i:=1 to n do s:=s*i

Преимущество структуры(1) тогда, когда переменная цикла целого типа и изменяется с шагом 1.

2. s:=1; i:=1;
Repeat
s:=s*i;
i:=i+1;
Until i>n;

3. s:=1; i:=1;
While i<=n do
Begin
s:=s*i;
i:=i+1;
End;

В конструкции (2,3) переменная цикла может быть вещественного типа. и изменятся с любым шагом.

! Если тело цикла содержит более одного действия, то необходимо использовать операторные скобки (Begin End).

Пример. Вывести на печать все цифры введенного целого числа.

```
Program Mac;
Var a,b: longint;
Begin
```

```
  Read(a); {ввод целого числа}
  Repeat
    b:=a mod 10; {вычисление крайней правой цифры числа}
    Writeln(b); {вывод цифры}
    a:=a div 10; {переменная а без крайней правой цифры}
  Until a=0;
End.
```

Пример. Возвести число а в степень n.

```
Program Work;
 Var n,i: integer;
     L,a: real;
 Begin
 Writeln('Введите число - a  :'); Readln(a);
 Writeln('Введите показатель степени - n :'); Readln(n);
 L:=1; i:=1;
 While i<=n do Begin
                 L:=a*L;
                 i:=i+1;
               End;
  Writeln('Число ',a,' в степени ',n,' равно ',L:6:4);
 End.
```

Задания для практической работы.

Составить программы на языке Pascal.

1. Вычислить сумму первых К слагаемых ряда:

а) 1+1/2+1/3+…1/к;

б) 2+4+6+….2*к;

в) $1+1/2^2-1/3^2+1/4^2-\ldots 1/к^2$;

г) 1+(1+2)+(1+2+3)+…+(1+2+…+10).

2. Найти частное и остаток от деления натурального числа А на число В, используя только операции вычитания и сравнения двух чисел.
3. Дано натуральное число К. Найти :а) сумму цифр числа; б) кол-во цифр в числе.
4. Даны два числа А и В(положительные). Составить алгоритм нахождения:

а) наименьшего общего делителя;

б) наименьшего общего кратного.

5. Вычислить n! =1*2*3*4*…*n (факториал числа).
6. Даны два целых положительных числа А и В. Найти наибольшее число К такое, что $A+A^2+A^3+\ldots+A^K <= B$.
7. Дано натуральное число К (K>0). Определить, является ли число простым (натуральные числа, которые имеет два натуральных делителя 1 и самого себя - 2,3,5,7…).
8. Дано натуральное число К (K>0). Определить, является ли оно совершенным. (число совершенное, если оно равно сумме своих делителей, 6=1+2+3).
9. Найти кол-во всех трехзначных чисел, сумма цифр которых равна числу К.
10. Сумма в S руб. положили в банк. При этом прибыль составляет k % в год от первоначальной суммы. Через какой срок сумма вклада увеличится в x раз.

8. Структурированный тип данных: массив

До сих пор мы рассматривали переменные, которые имели только одно значение, могли содержать в себе только одну величину определенного типа. Для

сохранения и обработки данных, содержащих десятки, сотни, тысячи величин в языке Pascal существует структурированный тип данных: массив.

Массив - это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются последовательно в ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его компонентой (или элементом массива).

Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент.

Переменные, представляющие компоненты массивов, называются переменными с индексами в отличие от простых переменных, представляющих в программе элементарные данные. Индекс в обозначении компонент массивов может быть константой, переменной или выражением порядкового типа.

Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется линейным (одномерным). Количество индексов элементов массива определяет размерность массива. По этом признаку массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д. Проще всего представить себе массив в виде таблицы, где каждая величина находится в собственной ячейке. Положение ячейки в таблице должно однозначно определяться набором координат (индексов).

Одномерный массив - одномерная упорядоченная совокупность элементов некоторого типа, которые адресуются с помощью индекса.

Пример. Числовая последовательность четных натуральных чисел 2, 4, 6, ..., N представляет собой линейный массив, элементы которого можно обозначить A[1]=2,A[2]=4,A[3]=6, ..., A[K]=2*(K+1), где K - номер элемента, а 2, 4, 6, ..., N -

значения. Индекс (порядковый номер элемента) записывается в квадратных скобках после имени массива.

При решении практических задач часто приходится иметь дело с различными таблицами данных, математическим эквивалентом которых служат матрицы. Такой способ организации данных, при котором каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен, называется двумерным массивом или таблицей.

Пример. Таблица умножения. На пересечении строки и столбца содержится результат произведения соответствующих чисел: A[i,j]=i*j, A[2,2]=4, A[3,2]=6. Первый индекс в записи таблиц обозначает строку, второй - столбец.

Форматы определения массивов для одномерного массива:

Var имя массива: array [начальный индекс .. конечный индекс] of тип данных;

Для двумерного массива:

Var имя массива: array [начальный индекс .. конечный индекс, начальный индекс .. конечный индекс] of тип данных;

Пример. Var massiv1: array [1..5] of real; - одномерный массив, состоящий из 5 элементов (чисел) действительного типа.

Var massiv2: array [1..10, 1..10] of integer; -двумерный массив из 100(10*10) целых чисел.

Описание массива в виде табличной структуры делается лишь из соображений удобства программирования. Многомерные массивы, так же как и одномерные, в памяти компьютера хранятся в виде линейной последовательности своих компонент, и при составлении алгоритмов принципиальной разницы между ними нет. Однако при работе с многомерными массивами приходится использовать вложенные циклы. В языках программирования существует возможность организовать цикл внутри тела другого цикла. Такой цикл будет называться вложенным циклом. Внутри вложенного цикла в свою очередь может быть вложен еще один цикл, образуя

следующий уровень вложенности и так далее. Порядок завершения работы вложенных циклов происходит с самого низкого уровня.

Заполнить массивы можно с помощью оператора присваивания. Этот способ заполнения элементов массива удобен, когда между элементами существует какая-либо зависимость, например, арифметическая или геометрическая прогрессии, или элементы связаны между собой рекуррентным соотношением.

Пример. Заполнить одномерный массив элементами, отвечающими следующему соотношению:

a1=1; a2=1; ai=ai-2+ai-1 (i = 3, 4, ..., 20).

```
A[1]:= 1; A[2]:= 1;
For i := 3 to 20 do  A[i] := A[i-1] + A[i-2];
```

Заполнить двумерный массив элементами таблицы умножения.

```
For i := 1 to 10 do
   For j := 1 to 10 do  A[i,j] := i*j;
```

Другой вариант присваивания значений элементам массива с помощью их ввода клавиатуры.

Пример. Заполнить одномерный массив 20 целыми числами, используя ввод их с клавиатуры, а затем распечатывает их в обратном порядке.

```
Program M1;
  Var A : Array [1..20] Of Integer; i : Integer;
  Begin
    For i:=1 to 20 do Begin
    Write('Введите A[', i, '] ');
    Readln(A[i]); {значениям индексов и вводим A[i] с клавиатуры }
    For i:=20 downto 1 do Write(A[i],' ') {Распечатываем массив в обратном порядке}
End.
```

Еще один вариант присваивания значений элементам массива - с помощью датчика случайных чисел. В языке Паскаль случайные числа формирует функция Random. Числа получаются дробными, равномерно расположенными в интервале от 0 до 1. Выражение, дающее целое случайное число в интервале [-n,n] будет выглядеть так: Trunc(Random*n+1)-n.

Пример. Заполнить и вывести на печать одномерный массив из 40 чисел с помощью датчика случайных чисел из интервала [-50,50].

```
Program M2;
  Const n=40; {Константа n - количество элементов массива}
  Var A : Array [1..n] of Integer; i : Integer;
  Begin
    For i:=1 to n do Begin A[i]:= Trunc(Random*101)-50;
    {заполнение массива случайными числами из интервала [-50,50]}
       Write(A[i],' ') { вывод элементов массива в строку через пробел}
  End
End.
```

Пример. Заполнить и вывести на печать двумерный массив с помощью датчика случайных чисел из интервала [0,1].

```
Program M3;
Const max=5;
Var a: array [1..max, 1..max] of real; i,j:byte;
Begin
        For i:=1 to max do
          For j:=1 to max do Begin
            a[i,j]:=Random;
            Writeln('a[',i,',',j,']=',a[i,j]);
                              End
End.
```

9. Реализация алгоритмов с массивами

Над элементами массивами чаще всего выполняются такие действия, как поиск, подсчет, перестановка, вставка, удаление элементов в массиве, удовлетворяющих заданному условию, сортировка элементов в порядке возрастания или убывания;

Сумму элементов массива можно подсчитать по формуле s=s+a[i] первоначально задав s=0. Количество элементов массива можно подсчитать по формуле к=к+1, первоначально задав к=0. Произведение элементов массива можно подсчитать по формуле p=p*a[i], первоначально задав p=1. При работе с элементами массива часто приходится изменять значения некоторых элементов, а так же переставлять элементы в массиве. При перестановке необходимо вводить дополнительную переменную.

Рассмотрим фрагменты программ, реализующих поиск элементов с заданными свойствами:

- поиск четных элементов.

```
For i:=1 to 10 do If a[i] mod 2=0 then write(a[i],' ');
```

- поиск элементов с нечетными индексами.

```
For i:=1 to 10 do If i mod 2<>0 then write(a[i],' ');
```

- подсчет количества элементов равных 0.

```
k:=0; For i:=1 to 10 do If a[i]=0 then k:=k+1;
```

- нахождение суммы элементов массива, находящихся до первого отрицательного.

```
s:=0;  i:=1;
While a[i]>=0 do Begin
   s:=s+a[i]; i:=i+1;
```

```
End;
```

- поиск максимального элемента в одномерном массиве.

```
max :=a[1]; i_max:=1;
For i :=1 to 5 do
If a [i] > max then Begin
                    max :=a [i];
                    i_max :=i;
                    End;
```

Рассмотрим фрагменты программ, реализующих замену, перестановку, вставку, удаление элементов в массиве.

- замена положительных элементы на 1, отрицательных на 0.

```
For i:=1 to n do If a[i]>0 then a[i]:=1 else if a[i]<0 then a[i]:=0;
```

- вставка числа x после всех элементов равных 0.

```
k:=0;
For i:=n downto 1 do
 If a[i]=0 then Begin
     For j:=n+k downto i+1 do
  a[j+1]:=a[j];
 a[i+1]:=x;
  k:=к+1; End;
```

- удаление из массива всех отрицательных элементов.

```
k:=0;
For i:=n downto 1 do
  If a[i]<0 then Begin
 For j:=i to n-1 do a[j]:=a[j+1];
        a[n]:=0; k:=к+1;
              End;
```

- перестановка минимального и максимального элементов

```
max:=a[1];  {присвоение максимуму значения первого элемента}
i_max:=1;  {присвоение номеру максимального элемента 1}
For i:=1 to n do  сравнение всех элементов с максимумом}
If a[i]>max then Begin
    max:=a[i]; {запоминаем значение максимального элемента}
    i_max:=i;  {запоминаем номер максимального элемента}
                End;
min:=a[i]; i_min:=i;
 For i:=1 to n do  {поиск минимального элемента}
If a[i]<min then Begin
    min:=a[i];
    i_min:=i;
                End; {конец поиска минимального элемента}
x:=a[i_max];{перестановка минимального и максимального элементов}
a[i_max]:=a[i_min];
a[i_min]:=x;
```

Сортировка массивов

При решении задачи сортировки обычно выдвигается требование минимального использования дополнительной памяти, из которого вытекает недопустимость применения дополнительных массивов. Для оценки быстродействия алгоритмов различных методов сортировки, как правило, используют два показателя: количество присваиваний, количество сравнений. Все методы сортировки можно разделить на две большие группы: прямые методы сортировки, улучшенные методы сортировки. Прямые методы сортировки в свою очередь разделяются на три подгруппы:

1) сортировка вставкой (включением);

2) сортировка выбором (выделением);

3) сортировка обменом (так называемая "пузырьковая" сортировка).

Улучшенные методы сортировки основываются на тех же принципах, что и прямые, но используют некоторые оригинальные идеи для ускорения процесса.

Сортировка простым включением: элементы массива просматриваются по одному, и каждый элемент вставляется в подходящее место, среди ранее упорядоченных.

```
Program Work_1;
Var a: array[1..10] of integer;
Var i,d,j:integer;
 Begin
 For i:=1 to 10 do Read(a[i]);
  For j:=2 to 10 do
Begin
  i:=j-1;
  d:=a[j];
While (i>0) and (d<a[i]) do
Begin
 a[i+1]:=a[i];
 i:=i-1;
 End;
a[i+1]:=d;
End;
 For  i:=1 to 12 do Write(a[i],' ');
End.
```

Сортировка простым выбором: Выбирается элемент с наименьшей величиной и меняется местами с первым. Затем эти операции повторяются с

оставшимися элементами без первого пока не останется только один элемент - наибольший.

```
Program Work_2;
Var a: array[1..10] of integer;
Var i,k,x,j,n:integer;
 Begin
  For i:=1 to 10 do read(a[i]);
   For i:=1 to 9 do Begin
k:=i; x:=a[I];
For j:=i+1 to 10 do
If  x<a[j] then Begin
 x:=a[j]; k:=j;
End;
a[k]:=a[i]; a[i]:=x
End;
For i:=1 to 10 do Write(a[i],' ');
End.
```

Сортировка простым обменом (метод “пузырька”): Если два элемента массива расположены не по порядку, то они меняются местами. Процесс повторяется до тех пор, пока элементы не будут упорядочены.

```
Program Work_3;
Var a: array[1..10] of integer;
Var i,k,x,j,n:integer;
 Begin
  For i:=1 to 10 do Read(a[i]);
  For i:=2 to 10 do Begin
    For j:=10 downto i do
If a[j-1]>a[j] then Begin
```

```
 x:=a[j-1]; a[j-1]:=a[j];a[j]:=x;
End;
End;
 For  i:=1 to 10 do Write(a[i],' ');
End.
```

Задания для практической работы.

Составить программы на языке Pascal.

1. Организовать ввод элементов одномерного массива размерностью n=10 с клавиатуры, заменить отрицательные элементы массива на их модули, а положительные увеличить на 1.
2. Переставить элементы одномерного массива по следующей схеме:

исходный массив - a[1],a[2],a[3],a[4],a[5]

итоговый - a[5],a[4],a[3],a[2],a[1]

3. Удалить из массива элементы кратные 3 или 5.
4. Информация о средней суточной температуре воздуха за n дней (n<=30) представлена в виде массива t из вещественных чисел. Определить номер дня, когда средняя суточная температура за n дней была наибольшей, и номер дня, когда средняя температура за сутки была второй по своей величине после наибольшей.
5. Организовать ввод элементов одномерного массива размерностью n=10 с помощью функции случайных чисел, вставить число *K* в одномерный массив после первого элемента кратного 3.
6. Поменять местами минимальный и максимальный элементы в массиве размерностью n=10.
7. Даны два массива a(n) и в(m). Получить новый массив с(?), который содержит только четные числа двух массивов.

8. Имеется n грузов весом a1, …, an и платформа грузоподъемностью r. Определить, какое максимальное количество грузов можно поместить на эту платформу. (Сортировка простыми вставками).

10. Обработка символьной и строковой информации

Символьный тип Char - это тип данных, предназначенный для описания одного символа: буквы, цифры, знака или кода. В памяти компьютера переменная типа Char занимает 1 байт. Символьные переменные в языке Pascal задаются следующим образом: **Var <идентификатор>:char;**

Значения символьных переменных обычно заключаются в апострофы, например: ’A’,’+’, ’;’, ’W’. Однако они могут записываться с помощью знака решетки (#) и кода таблицы ASCII, например #67 соответствует символу ‘C’.

Пример.

```
Var Mv, kv, nv, cv:char;
rc:=’Q’; nv:=#0;{пустой символ}
```

При работе с переменными типа Char в языке Pascal используются следующие функции:

Chr(x:byte):char; - возвращает символ, соответствующий в ASCII-таблице коду числа x;

Ord(x:char):byte; - возвращает порядковый номер в ASCII-таблице символа x;

UpCase(x:char):char: - преобразует символы из строчных латинских букв в прописные;

Pred(x:char):char: - возвращает символ, который предшествует символу x в ASCII-таблице;

Succ(x:char):char: - возвращает символ, который следует за символом x в ASCII-таблице/

Пример.

Выражение	Результат
Chr(60)	’ <’
Ord(’1’)	48
Chr(55)	’7’
Pred(’9’)	’8’
Succ(’5’)	’6’

Символьные переменные можно сравнивать друг с другом. Большим считается тот символ, код которого больше по таблице ASCII.

Строковый тип данных String – структурированный тип данных, предназначенный для обработки строк. Строка - это последовательность символов. Каждый символ занимает 1 байт памяти (код ASCII). Количество символов в строке называется ее длиной. Длина строки может находиться в диапазоне от 0 до 255. Строковые величины могут быть константами и переменными. Особенностью строки в языке Pascal является то, что с ней можно работать как с массивом символов, так и с единым объектом.

Строковая константа - последовательность символов, заключенная в апострофы. Строковая переменная описывается в разделе описания переменных следующим образом:

Var <идентификатор> : string[<максимальная длина строки>];

Пример: Var Name : string[20].

Пустая строка изображается как ’’.

Тип string и стандартный тип char совместимы. Строки и символы могут употребляться в одних и тех же выражениях. Строковые выражения строятся из строковых констант, переменных, функций и знаков операций. Над строковыми данными допустимы операции сцепления и операции отношения.

Операция сцепления (+) применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и

переменные. Операции отношения: =, <, >, <=, >=, <>. Позволяют произвести сравнение двух строк, в результате чего получается логическое значение (true или false).

Пример.

```
Var S1:string[5]; S2:string[6];S3:string[11];
S1:=’инфор’;
S2:=’матика’;
S3:=S1+S3; {получим строку: ’информатика’}
```

Ввод и вывод строковых данных осуществляется с помощью стандартных процедур ввода и вывода. Обращение к отдельному символу строки выполняется с помощью индекса (номера) символа в квадратных скобках.

Строковые функции

Пусть S, S1, S2, sk - выражения строкового типа; poz, n, code - целого типа.

1. **Length(S)**: integer; - вычисляет длину строки S в символах;
2. **Concat(s1, s2,…,sk):**string; - выполняет объединение строк s1, s2, …sk;
3. **Copy(s, poz, n):string**: - выделяет из строки S подстроку , начиная с позиции pozл длиной n;
4. **Pos(s1, s2):byte;** - отыскивает первое появление в строке s2 подстроки s1. Результатом работы этой функции становится номер (индекс) позиции в строке s2, с которой начинается строка s1. Если результат равен 0, то подстрока не найдена.

Строковые процедуры

1. **Delete(S, poz, n);** - удаляет из строки S n символов, начиная с позиции poz;
2. **Insert(S1, S2, poz):** - вставляет строку S1 в строку S2, начиная с позиции poz;
3. **Str(v, S):-** преобразует числовое значение v в строковое представление; если v – целое число, то может быть указано общее число символов 1, например str(v:1:s); если v принимает вещественное значение, то

дополнительно может быть указано число m символов после десятичной точки, например str(v:1:m:s);

4. **Val(S,v,code);** - преобразует строку S в число, и переменной v присваивается это числоn; тип числа зависит от типа переменной v – целый или вещественный. Если такое преобразование выполнено успешно, то переменной code присваивается значение 0, в противном случае code содержит номер позиции первого ошибочного символа.

Пример.

Пусть st - строка типа string[6] и st=’klmn’, S – строка string[5]. x - переменная целого типа, fs – строкового типа.

Функция	Результат
X:=Leght(st)	4
Fs:=copy(st,2,2)	‘lm’
X:=pos(‘mn’,st)	3
Процедура	Результат
Delete(st,1,1)	‘lmn’
Insert(‘ab’,st,1)	‘abklm’
Str(15,s)	‘15’
Val(‘-3’, x, code)	-3 0

Пример. Дано слово ’золотопромышленник’, из которого постройте слова ’золото’, ’промышленник’, ’пленник’, ’мышление’, используя процедуры и функции обработки строк.

```
Var S, rs1, rs2, rs3, rs4, temp:string;
Begin
    S:= ’золотопромышленник’;
    Rs1:=Copy(s,1,6);   {выделим слово ‘золото’}
```

```
    Rs2:=Copy(s,length(rsl)+1,length(s)-length(rsl));{слово ‘промышленник’}
    Temp:=S;
    Delete(temp,1,12);  {оставим часть слова ’ленник’}
    Rs3:=S[7]+temp;     {построим слово ‘пленник’}
    Rs4:=S;
    Delete(rs4,1,9);        {оставим часть слова ‘мышленник’)
    Delete(rs4,7,1);        {оставим часть слова  ‘мышленик’}
    Rs4[length(rs4)]:=rs4[5];   {заменим ‘к’ на ‘е’, получим ‘мышление’}
    Writeln(rs1,’’,rs2,’’,rs3,’’,rs4,);
End.
```

Пример. Дан текст, состоящий из слов, разделенных одним пробелом. Удалите все вхождения данного слова в строку.

```
Var K, M:byte; Sl, Wr:string;
Begin
  Writeln(‘введите строку текста из слов’); Readln(Sl);
  Sl:=’ ’+ Sl +’ ’;
  Writeln(‘введите удаляемое слово’); Readln(Wr);
  Wr:=’ ‘ + Wr + ‘ ‘ ;
  K:=length(Wr);       {определяем длину слова}
  Repeat
  M:=pos(Wr,Sl);      {номер позиции удаляемого слова}
  If (M<>0) then delete(Sl,M + 1,K-1) {удаляем слово из текста}
  Until M:=0
  Writeln(Sl)
End.
```

Пример. Дана слово. Необходимо определить, является ли это слово палиндромом. (потоп, Анна)

```
Var  D, G:byte; Tet:string; F:boolean;
```

```
Begin
  Writeln(‘введите слово’); Readln(Tet);
  D:=length(Tet);     {определим длину слова}
  F:=true;     {признак, фиксирующий, что слово является палиндромом}
  G:= D div 2;
  While G>=1 do Begin
  If Tet[G]<>Tet[d – G + 1] then F:=false;
  G:=G – 1
  End;
  If F Then Writeln(‘ слово является палиндромом’)
  Else Writeln(‘слово не является палиндромом’)
End.
```

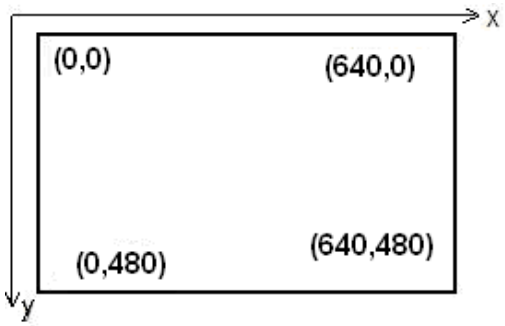
Задания для практической работы.

Составить программы на языке Pascal.

1. Подсчитать, сколько раз в заданном тексте встречается заданный символ.
2. Заменить в заданном тексте буквосочетание "min" на "ma".
3. В заданном тексте подсчитать общее количество букв "x" и "y".
4. Удвоить каждую букву в заданном тексте.
5. Вычеркнуть из заданного слова все буквы "a".
6. Заданную строку A переписать в обратном порядке в строку B.
7. В заданной последовательности слов найти все слова, начинающиеся с заданной приставки.
8. Найти самое длинное и самое короткое слово в заданном предложении.
9. В заданном тексте подсчитать наибольшее количество подряд идущих пробелов.

11. Использование графических возможностей языка программирования

При работе в графическом режиме изображение на экране строится из точек (пиксель). Каждый пиксель на экране имеет координаты (X,Y), которые образуются номерами столбцов (X) и строк (Y). Нумерация начинается в верхнем левом углу. Именно в этом углу расположен пиксель с координатами (0,0). Координата X растет вправо, координата Y - вниз.



Любая графическая картинка формируется из простых геометрических фигур. Это точки, отрезки (линии), прямоугольники, окружности и т.д. Графические координаты принимают только целочисленные значения.

Рисование различных геометрических фигур осуществляется с помощью специальных стандартных команд (процедур). Команды для работы в графическом режиме хранятся в библиотечном модуле GraphAbc, который описывается в разделе описаний с помощью зарезервированного слова Uses. В модуле GraphAbc с помощью команды setwindowsize(X,Y) можно задавать размеры графического окна. По умолчанию графическое окно будет принимать размеры экрана компьютера

Пример.

```
Program t1;
uses graphabc;  {подключение модуля GraphAbc}
Begin
setwindowsize(640,480); {устанавливает размеры графического окна}
End.
```

Команда setpixel(x,y,c) - рисует точку с координатами (x,y) цветом c. Стандартные цвета c задаются символическими константами:

clBlack	черный	clYellow	желтый
clWhite	белый	clNavy	темно-синий
clRed	красный	clMaroon	темно-красный
clGreen	зеленый	clPurple	фиолетовый
clBrown	коричневый	clCream	кремовый
clBlue	синий	clAqua	бирюзовый
clSkyBlue	голубой	clOlive	оливковый
clFuchsia	сиреневый	clTeal	сине-зеленый
clGray	темно-серый	clMedGray	серый
clLime	ярко-зеленый	clSilver	серебряный

Процедура Line(x1,y1,x2,y2) вычерчивает прямую линию из точки (x1,y1) в точку (x2,y2) цветом установленным процедурой SetPenColor(c).

Для задания стиля линии существует команда SetPenStyle (Style), где Style - стиль линии: psSolid - сплошная, psDash – штриховая, psDot – пунктирная, psClear – прозрачная.

Установить ширину текущего пера можно процедурой SetPenWidth(Width: integer).

Процедура Rectangle(x1,y1,x2,y2) рисует прямоугольник, где (x1,y1) координаты верхней левой точки, а (x2,y2) - правой нижней.

Процедура FillRect(x1,y1,x2,y2: integer) - заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.

Процедура Circle(x,y,r) рисует окружность с центром в точке (x,y) и радиусом r.

Закрасить замкнутую фигуру можно с помощью процедуры FloodFill(x,y,c), (x,y) - координата точки внутренней области фигуры, c – цвет заливки.

Процедура ClearWindow(color) устанавливает цвет графического окна. Закрасить прямоугольник и круг можно, используя команду закраски кистью SetBrushColor (color).

SetBrushStyle(Style) - устанавливает стиль текущей кисти. Константы стилей кисти: bsSolid – сплошная, bsClear – прозрачная, bsHatch – штриховая, bsGradient – градиентная.

Процедуры для ввода текста:

TextOut(x,y: integer; s: string) - выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s);

SetFontSize(size: integer) - устанавливает размер текущего шрифта в пикселях;

SetFontColor(c: Color) - устанавливает цвет текущего шрифта;

SetFontStyle(fs: integer) - устанавливает стиль текущего шрифта.

Пример. Программа рисования отрезка, прямоугольника, треугольника и окружности.

```
Program L;
 uses GraphAbc;
 Begin
SetWindowsize(640,480);{устанавливает размеры графического окна в пикселах}
ClearWindow(clYellow); {устанавливает желтый цвет фона}
SetPenColor(clBlue);{устанавливает синий цвет пера}
SetPenWidth(10); {устанавливает ширину пера}
Line(250,250,300,100);   {рисует линию}
Line (100,270,200,350); {рисует треугольник}
Line (200,350,300,270);
Line (300,270,100,270);
FloodFill(175,300,clgreen); {Закрашивает треугольник}
Setpencolor(clGreen); {устанавливает зеленый цвет пера}
```

```
SetPenWidth(5);      {устанавливает ширину пера}
Rectangle(30,30,230,130);{рисует прямоугольник}
FloodFill(50,100,clSilver);
Setpencolor (clNavy);    {устанавливает красный цвет пера}
Circle(460,260,70); {рисует окружность};
FloodFill(460,260,clSkyBlue);
SetFontStyle(fsbold); {устанавливает стиль шрифта}
SetFontSize(18);     {устанавливает размер шрифта}
SetFontColor(clFuchsia); {устанавливает оливковый цвет шрифта}
TextOut(300,30,'Геометрические фигуры'); {делает надпись}
 End.
```

С помощью геометрических фигур можно создавать любые картинки. Используя известные команды рисования геометрических фигур, нарисуйте картинки и подпишите их:

- закрашенную зеленым цветом елку;
- лодку с парусом;
- снеговика;
- дом;
- придумайте свой рисунок.

Приложение

Основные команды оболочки программирования Pascal ABC

Таблица 1. Меню Правка

Команда	Назначение
Отменить CTRL+Z	Отмена последней операции редактирования текста программы
Восстановить Shift+Ctrl+Z	Восстановление предыдущей операции редактирования текста программы
Вырезать CTRL+X	Перемещение выделенного фрагмента текста из окна редактора в буфер обмена
Копировать CTRL+C	Копирование выделенного фрагмента текста из окна редактора в буфер обмена
Вставить CTRL+V	Вставка выделенного текста из буфера обмена в окно редактора
Найти CTRL+F	Поиск текста
Заменить CTRL+R	Поиск текста и замена его новым текстом
Найти далее CTRL+L	Дальнейший поиск текста

Таблица 2. Меню Файл

Команда	Назначение
Новый CTRL+N	Открытие окна для нового файла
Открыть CTRL+O	Открытие (загрузка) файла
Сохранить CTRL+S	Сохранение файла с прежним именем
Сохранить как …	Сохранение файла с новым именем
Закрыть CTRL+F4	Закрыть текущее окно файла
Выход	Выход из системы программирования

Таблица 3. Ошибки при работе в системе программирования Pascal ABC

Ошибка	Причина ошибки

Неожиданный символ	Символ введен не с регистра английских букв
Ожидался символ "точка"	После end нет точки
ожидалось begin	Нет слова begin
Ожидалась "точка с запятой"	Нет символа «;» либо неверно расставлены скобки в команде
Ожидался идентификатор	Не указано имя программы
Ожидалась команда	Отсутствует end
Неизвестное имя	Неправильно написано название команды
Ожидадось «)»	Не закрыта скобка в конце команды
Ожидался конец файла	Неверно написано слово end либо есть лишние символы после слова end в конце программы
ожидался идентификатор, но ... обнаружено зарезервированное слово	Например, вместо val1 написано var
Ошибка ввода. Программа завершена	Неверно указано имя файла с задачей (‘begin’)

Графические примитивы Модуля GraphABC

procedure SetPixel(x,y,color: integer);

Закрашивает один пиксел с координатами (x,y) цветом color.

procedure MoveTo(x,y: integer);

Передвигает невидимое перо к точке с координатами (x,y); эта функция работает в паре с функцией LineTo(x,y).

procedure LineTo(x,y: integer);

Рисует отрезок от текущего положения пера до точки (x,y); координаты пера при этом также становятся равными (x,y).

procedure Line(x1,y1,x2,y2: integer);

Рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2).

procedure Circle(x,y,r: integer);

Рисует окружность с центром в точке (x,y) и радиусом r.

procedure Ellipse(x1,y1,x2,y2: integer);

Рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1,y1) и (x2,y2).

procedure Rectangle(x1,y1,x2,y2: integer);

Рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2).

procedure TextOut(x,y: integer; s: string);

Выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

procedure FloodFill(x,y,color: integer);

Заливает область одного цвета цветом color, начиная с точки (x,y).

procedure FillRect(x1,y1,x2,y2: integer);

Заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.

Процедуры для работы с пером

Рисование линий осуществляется текущим пером.

procedure SetPenColor(c: Color);

Устанавливает цвет текущего пера

procedure SetPenWidth(Width: integer);

Устанавливает ширину текущего пера

procedure SetPenStyle(style: DashStyle);

Устанавливает стиль текущего пера. Константы стилей пера приведены ниже.

procedure SetPenMode(m: integer);

Устанавливает режим текущего пера

Стили пера

Стили пера определены следующими константами:

psSolid = DashStyle.Solid; Сплошное перо

psDash = DashStyle.Dash; Штриховое перо

psDot = DashStyle.Dot; Пунктирное перо

psDashDot = DashStyle.DashDot; Штрихунктирное перо

psDashDotDot = DashStyle.DashDotDot; Альтернативное штрихунктирное перо

psClear = DashStyle.Clear; Прозрачное перо

процедуры для работы с графическим окном

procedure SetWindowWidth(w: integer);

Устанавливает ширину клиентской части графического окна в пикселах

procedure SetWindowHeight(h: integer);

Устанавливает высоту клиентской части графического окна в пикселах

procedure SetWindowLeft(l: integer);

Устанавливает отступ графического окна от левого края экрана в пикселах

procedure SetWindowTop(t: integer);

Устанавливает отступ графического окна от верхнего края экрана в пикселах

procedure SetWindowCaption(s: string);

Устанавливает заголовок графического окна

procedure SetWindowTitle(s: string);

Устанавливает заголовок графического окна

procedure SetWindowSize(w,h: integer);

Устанавливает размеры клиентской части графического окна в пикселах

procedure SetWindowPos(l,t: integer);

Устанавливает отступ графического окна от левого верхнего края экрана в пикселах

procedure ClearWindow;

Очищает графическое окно белым цветом

procedure ClearWindow(c: Color);

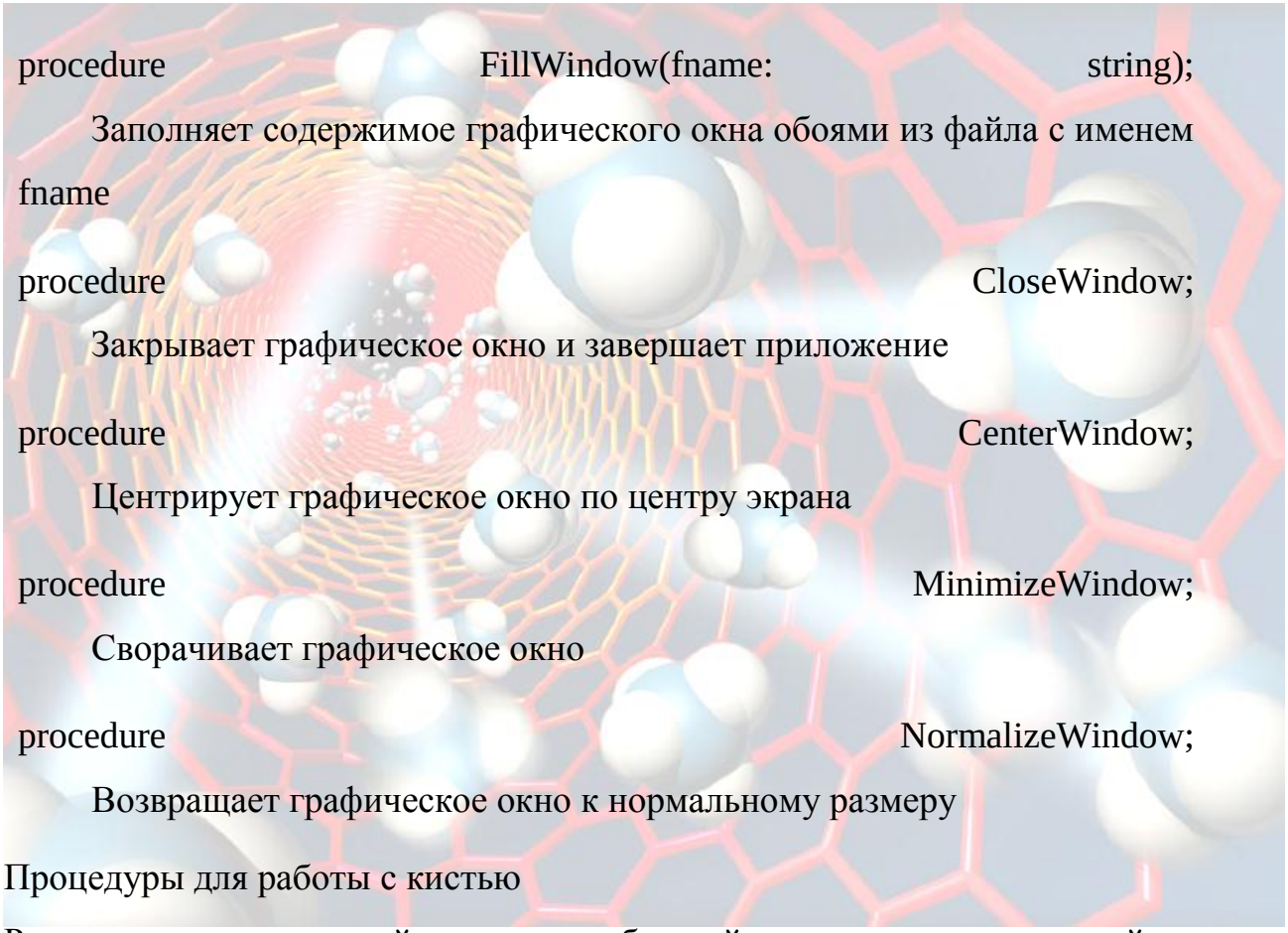
Очищает графическое окно цветом c

procedure SaveWindow(fname: string);

Сохраняет содержимое графического окна в файл с именем fname

procedure LoadWindow(fname: string);

Загружаает содержимое графического окна из файла с именем fname



procedure FillWindow(fname: string);

Заполняет содержимое графического окна обоями из файла с именем fname

procedure CloseWindow;

Закрывает графическое окно и завершает приложение

procedure CenterWindow;

Центрирует графическое окно по центру экрана

procedure MinimizeWindow;

Сворачивает графическое окно

procedure NormalizeWindow;

Возвращает графическое окно к нормальному размеру

Процедуры для работы с кистью

Рисование внутренностей замкнутых областей осуществляется текущей кистью.

Procedure SetBrushColor(c: Color);

Устанавливает цвет текущей кисти

procedure SetBrushStyle(bs: BrushStyleType);

Устанавливает стиль текущей кисти. Константы стилей кисти

приведены ниже.

procedure SetHatchBrushBackgroundColor(c: Color);

Устанавливает цвет заднего плана текущей штриховой кисти

procedure SetGradientBrushSecondColor(c: Color);

Устанавливает второй цвет текущей градиентной кисти

Стили кисти

bsSolid Сплошная кисть (по умолчанию)

bsClear Прозрачная кисть

bsHatch Штриховая кисть

bsGradient Градиентная кисть

Процедуры для работы со шрифтом

Вывод текста осуществляется текущим шрифтом.

Procedure SetFontSize(size: integer);

Устанавливает размер текущего шрифта в пикселях

Procedure SetFontColor(c: Color);

Устанавливает цвет текущего шрифта

Procedure SetFontStyle(fs: integer);

Устанавливает стиль текущего шрифта