МИНИСТЕРСТВО ОБРАЗОВАНИЯ
КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СОПУЕВ А., АРКАБАЕВ Н.К.

ПРАКТИКУМ НА ЭВМ

ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ РАБОТ ПО ЯЗЫКУ ПАСКАЛЬ

Ош – 2004

ББК 32.973-01
С - 64

Рецензенты: Сатыбаев А., зав. каф. "Прикладной математики"
КУУ, к.ф.-м.н., доцент

Алымкулов К., Зав. каф. "Общая информатика"
ОшГУ, д.ф.-м.н., профессор

Сопуев А., Аркабаев Н.К.
С – 64 Практикум на ЭВМ. Задания для практических работ по языку Паскаль.– Ош: Изд. центр ОшГУ, 2009. – 67с.

ISBN 9967-03-181-6

Приведены наиболее сложные понятия языка Паскаль и методы решения как стандартных, так и нестандартных задач, задания для практических и лабораторных работ, касающихся изучения базовых конструкций и принципов программирования на языке Паскаль. Материалы пособия могут быть использованы при проведении занятий по предметам "Язык программирования Паскаль", "Языки программирования и методы трансляции", "Практикум на ЭВМ", "Введение в вычислительную математику", "Вузовский компонент", а также при подготовке учащихся средних школ и студентов к олимпиаде.

Данное пособие предназначено для студентов и преподавателей вуза.

Печатается по решению Ученого Совета ОшГУ

С 2404090000-04 **ББК 32.973-01**

ISBN 9967-03-181-6

Содержание

Введение 4

1. Идентификация объектов 5

2. Оператор выбора 8

3. Вывод таблиц 10

4. Расчет конечных сумм 11

5. Расчет бесконечных сумм 13

6. Расчет функциональных рядов 16

7. Расчет банковских вкладов 18

8. Расчет произведений 20

9. Работа с массивами 22

10. Работа со строками 25

11. Преобразования массивов 27

12. Текстовые файлы 30

13. Зашифрование текстовых файлов 33

14. Работа с блочными файлами 35

15. Управление текстовым режимом 37

16. Управление звуком 40

17. Динамические текстовые эффекты 41

18. Типовые задачи для текстового режима 45

19. Построение заполненных фигур (модуль Graph) 47

20. Работа с линиями 49

21. Перемещение фигуры 52

22. Масштабирование фигуры 57

23. Построение изображений 60

24. Построение графиков функций 63

Литература 67

Введение

Для приобретения навыков программирования не достаточно прочтения книги, посвященной языку программирования, необходимо составлять программы, решать конкретные задачи. В учебниках, как правило, приводятся типовые, стандартные задачи, в основе которых лежит расчет по формулам. Такие задачи полезны, но они не всегда интересны.

В данной книге приведены наиболее сложные понятия языка Паскаль и методы решения как стандартных, так и нестандартных задач, задания для практических и лабораторных работ, касающихся изучения базовых конструкций и принципов программирования на языке Паскаль.

Чтобы получить максимальную пользу от книги, надо активно с ней работать, решать задачи, изучать приведенные решения. Вводить их в компьютер. По чаще экспериментировать – вносить изменения в программы.

Перед тем как приступить к решению задач, нужно изучить соответствующую тему – прочитать раздел учебника. Если решение задачи дается не сразу, то можно посмотреть образец решения аналогичного примера и затем ещё раз попытаться решить задачу самостоятельно. Составлять программу лучше сначала на бумаге, а уже затем вводить в компьютер. Задача считается решенной, если программа работает так, как сказано в условии.

Важно, чтобы решенная задача соответствовала правильному оформлению, включающему в себя использование:

- осмысленных имен переменных, констант, функций и процедур;
- отступов при записи инструкций;
- комментариев.

Правильно оформленную программу легче отлаживать, кроме того, она производит хорошее впечатление.

Материалы пособия могут быть использованы при проведении занятий по предметам "Язык программирования Паскаль", "Языки программирования и методы трансляции", "Практикум на ЭВМ", "Введение в вычислительную математику", "Вузовский компонент", а также при подготовке учащихся средних школ и студентов к олимпиаде.

Книга представляет собой задачник по программированию на языке Паскаль и предназначено для студентов и преподавателей вуза.

1. Идентификация объектов

Для **идентификации** (распознавания) **объекта** применяется условный оператор по определенным признакам составляющих его элементов. Например, если объектом является треугольник, то **элементами объекта** могут быть: 1) три его угла (α, β, γ); 2) три его стороны (a, b, c); и т. д.

Признаками являются значения элементов, по которым производится идентификация, например, для углов:

1) один угол > 90 - (один признак);

2) три угла < 180 - (три признака); и т. д.

В результате идентификации объект получает имя. Например, треугольник - остроугольный, либо тупоугольный и т. д.

Если идентификация проводится **по одному признаку для нескольких элементов**, то несколько условий связываются служебным словом **"or"**, например:

If (a >90) or (b >90) or (c >90) then writeln ('Треугольник-тупоугольный');

Если идентификация проводится по нескольким признакам, число которых равно числу элементов, то несколько условий связываются служебным словом **"and"**, например:

If (a<90) and (b <90) and (c <90) then writeln ('Треугольник-остроугольный');

Если **имя объекта составное**, то добавляются признаки для идентификации второй части имени и применяются вложенные условные операторы, например, для равнобедренного треугольника:

If (a<90) and (b<90) and (c<90) then If (a=b) or (b=c) or (a=c) then writeln('Треугольник - остроугольный и равнобедренный') else writeln('Треугольник - остроугольный');

Напомним, что условный оператор можно применять для **контроля правильности вводимых данных**, например:

If (a+b+c)<>180 then begin writeln('Сумма углов<>180'); Halt end;

Если для идентификации объекта достаточно меньшего числа признаков, чем число элементов, то условия, связанные **"and"** группируются, а группы соединяются служебным словом **"or"**. Например, четырехугольник имеет элементами четыре стороны (a, b, c, d), а его имя устанавливается по двум признакам (равенство двух пар сторон), тогда можно использовать операторы:

If ((a=b) and (c=d)) or ((a=c) and (b=d)) or ((a=d) and (b=c)) then writeln('Параллелограмм');

Таким образом, если при идентификации объекта число признаков меньше, чем число элементов, то условия группируются.

Пример 1. Пусть для идентификации зайца достаточно трех признаков для четырех элементов: YXO - длинное ('D'), XBOCT - короткий ('K'), LAPA - передние меньше задних ('PmZ'), HOC - короткий ('K'). Составьте программу идентификации зайца по ее элементам.

Решение. Для решения задачи можно использовать операторы:

If ((YXO='D') and (XBOCT='D') and (LAPA='PmZ')) or
((YXO='D') and (XBOCT='D') and (HOC='K')) or
((YXO='D') and (HOC='K') and (LAPA='PmZ')) then writeln('Это ЗАЯЦ');

Тогда программа решения задачи выглядит так:

```
Program primer1;
uses crt;
  Var YXO, XVOST, NOS, LAPA: char;
begin  clrscr;
  writeln('Введите ухо. Если длинное/короткое тогда D/K');
  readln(YXO);
  writeln('Введите хвость. Если длинное/короткое тогда D/K');
  readln(XVOST);
  writeln('Введите нос. Если длинное/короткое тогда D/K');
  readln(NOS);
  writeln('Введите лапа. Если передные меньше/больше задних
PmZ/PbZ');
  readln(Lapa);
If ((YXO='D') and (XVOST='D') and (LAPA='PmZ')) or
 ((YXO='D') and (XVOST='D') and (NOS='K')) or
 ((YXO='D') and (NOS='K') and (LAPA='PmZ')) then writeln('Это
заяц')
  else writeln('Это не заяц');
end.
```

Задачи для самостоятельной работы

В приведенных ниже задачах необходимо составить программу идентификации геометрической фигуры по ее элементам. Величины указанных элементов фигуры генерируются случайно как целые числа в допустимых диапазонах (например, углы - в диапазоне от 1 до 180).

1.1. Идентификация треугольника по двум его углам U1 и U2. Определяемое свойство: остроугольный, прямоугольный, тупоугольный. Следует учесть, что для выполнения условия U1+U2<180 при генерации значения второго угла необходимо учитывать величину первого угла (а для корректного анализа не надо забывать и о

величине третьего угла).

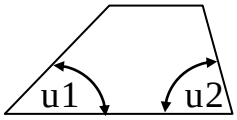
1.2. Идентификация треугольника по трем его сторонам. Определяемое свойство: прямоугольный или нет.

Значения сторон генерировать в диапазонах от 1 до 20, причем для третьей стороны нужно, чтобы ее размер не превышал суммы и не был меньше модуля разности первых двух сторон.

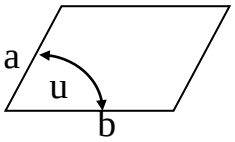
1.3. Идентификация треугольника по двум сторонам и углу между ними. Определяемое свойство: равносторонний, равнобедренный или прямоугольный (второе и третье может быть одновременно).

Для корректного анализа нужно определить еще третью сторону, например, по теореме косинусов $c^2=a^2+b^2-2abcos(U))$.

1.4. Идентификация трапеции по двум прилежащим углам. Определяемое свойство: обыкновенная, прямоугольная, равнобедренная, прямоугольник.



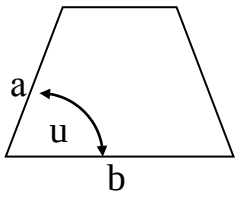
1.5. Идентификация параллелепипеда по прилежащим сторонам и углу между ними. Определяемое свойство: ромб, прямоугольник, квадрат или обыкновенный параллелограмм.



1.6. Идентифицировать треугольник: остроугольный, тупоугольный, прямоугольный, равнобедренный, равносторонний по трем элементам - углам (сторонам). Причем, имя равнобедренного треугольника может быть составным. Контролировать допустимые значения элементов при вводе данных.

1.7. Идентифицировать четырехугольника: ромб, квадрат по пяти элементам - сторонам и углу.

1.8. Идентификация равнобедренной трапеции по двум прилежащим сторонам и углу между ними. Определяемое свойство: обыкновенная, прямоугольная, квадрат. Первая генерируемая сторона трапеции (a) должна быть не меньше противолежащей стороны (и эта противолежащая сторона не должна вырождаться при генерации угла и второй стороны).



1.9. Идентификация эллипса по двум его осям. Определяемое свойство: вертикальный, горизонтальный и окружность.

1.10. Идентификация параллелепипеда по двум прилежащим углам. Определяемое свойство: квадрат, ромб.

1.11. Идентифицировать зверьков: заяц, кенгуру, белка по трем признакам для четырех элементов.

1.12. Идентифицировать летающий объект: самолет, вертолет, ракета, тарелка по трем признакам для четырех элементов. Например: для элементов: крылья, хвост, двигатель, длинный корпус при-

знаками могут быть значения "Y" или "N".

Возможная модификация заданий 1.1–1.8: вводить величины элементов для идентификации фигур с клавиатуры, предусмотрев защиту от неверного ввода данных и для неопознанного объекта должно выдаваться соответствующее сообщение на экран.

2. Оператор выбора

Оператор служит для одного из помеченных вариантов действия (операторов), в зависимости от значения "параметра". Оператор имеет вид:

Case "параметр" Of
"список помеченных операторов"
Else "оператор" End;

Здесь "параметр" - выражение или переменная порядкового типа.

Из "списка помеченных операторов" выполняется оператор с меткой, включающей значение "параметра", иначе оператор после слова Else. Конструкция Else "оператор" может отсутствовать.

Пример 2. Составьте программу случайного предсказания одного из десяти вариантов ближайшего будущего с вероятностью 1/20, в остальных случаях - вы "неудачник".

Решение. Здесь мы можем использовать функцию Random(x), которая генерирует случайное число с равномерной плотностью распределения на заданном интервале. Для инициализации распределения в начале программы необходимо вызвать процедуру Randomize. Ниже приводится программа решения задачи.

```
Program primer2;
Uses crt;
var N: word;
Begin clrscr;
  writeln('ПРЕДСКАЗАНИЕ БУДУЩЕГО');
  Randomize; N:=Random(20)+1; {N - случайное число от 1 до 20}
  writeln; write('Вас ожидает _');
  case N of
  1: writeln('счастье'); 6: writeln('здоровье');
  2: writeln('пятерка'); 7: writeln('деньги');
  3: writeln('дорога'); 8: writeln('любовь');
  4: writeln('двойка'); 9: writeln('встреча');
  5: writeln('болезнь'); 10: writeln('дети')
  else writeln('неудача')
end;
End.
```

Задачи для самостоятельной работы

2.1. Составить программу случайного выбора места летнего отдыха из семи предлагаемых туристическим агентством курортов, причем с вероятностью 3/10 придется отдыхать на даче.

2.2. Составить программу случайного выбора дежурного из списка, в котором 4 мальчика и 4 девочки, причем для девочек вероятность выбора в два раза ниже, чем для мальчиков.

2.3. Составить программу случайного выбора трех дисциплин, по которым придется сдавать экзамены, из предлагаемых на выбор четырех (всего возможно 4 варианта выбора).

2.4. Составить программу, анализирующую нажатую клавишу на ее принадлежность к определенной группе клавиш. Соответствующие сообщения выдаются для цифровых клавиш, больших букв и малых букв латинского алфавита.

2.5. Составить программу, анализирующую код символа на принадлежность символа к определенной группе. Соответствующие сообщения выдаются для псевдографики, больших букв и малых букв русского алфавита. Символ генерируется по коду как случайному числу в диапазоне 33..255.

2.6. Составить программу-анализатор вводимого с клавиатуры целого числа по двум признакам - его разрядности и знака.

2.7. Вывести на экран сообщение в зависимости от введенного значения температуры воздуха (от -50 до +50 0 С), например: -50..-20: очень холодно, -19..-10: холодно, и т. д., иначе - неправильный ввод данных.

2.8. Вывести на экран сообщение в зависимости от введенного значения оценки (по десятибалльной системе), например:1..2: плохо, 3..5: удовлетворительно, и т.д., иначе - неправильный ввод данных.

2.9. Дано натуральное число N. Если оно делится на 4, вывести на экран ответ N=4k (где k-соответствующее частное); если остаток от деления на 4 равен 1, N=4k+1; если остаток от деления на 4 равен 2, N=4k+2; если остаток от деления на 4 равен 3, N=4k+3.

2.10. Составить программу, печатающую возможные значения координат x и y в зависимости от номера квадранта.

2.11. Составьте программу, которая определяет, является ли введенный символ гласной или согласной буквой английского алфавита.

2.12. Составьте программу, определяющую нажатую клавишу: функциональный, основной, управляющий.

3. Вывод таблиц

При выводе данных можно задавать формат вывода. Для этого после имени переменной ставится знак двоеточие ":", например: Writeln(a:5:2); - при выводе значения переменной вещественного типа отводится 5 позиций (включая отрицательный знак и точку), из них 2 позиции отводится для вывода цифр в дробной части. При выводе значения переменной целого типа задается количество позиций для числа (включая отрицательный знак), например: Writeln(i:8);

При выводе значений символьных и строковых переменных формат определяет число позиций для вывода значения переменной (текста).

При выводе число или текст выравниваются по правому краю отведенного поля, например: если a:=5. 02;, то оператор Writeln('a=', a:6:2); выведет на экран: a=_ _ 5. 02, если n:='116'; s:='Школа N';, то оператор Writeln(s:7, n:6);
выведет на экран надпись: Школа N _ _ _ 116.

Для выравнивания числа или текста по левому краю отведенного поля первый знак формата задается отрицательным, например:
оператор Writeln('a=', a:-8:2); выведет на экран надпись: a=5. 02
оператор Writeln(s:7, n:-6); выведет на экран надпись: Школа N116.

Пример 3. Найти все пятизначные числа, делящиеся на произведения своих цифр. Вывести эти числи в 10 колонок.

Решение. Для решения задачи достаточно использовать следующую программу:

```
Program primer3;
uses crt;
Var p,j,t,code: integer;
    s:string;
    i:longint;
begin  clrscr;
 for i:=10000 to 99999 do
  begin
   str(i,s);p:=1;
   for j:=1 to 5 do
   begin val(s[j],t,code);p:=p*t; end;
   if (p<>0) and (i mod p=0) then write(i:8);
  end;
 readln;
end.
```

Задачи для самостоятельной работы

В задачах 1–6 (анализ чисел по цифрам) необходимо вывести список указанных чисел в "K" колонок.

3.1. Шестизначные четные числа, делящиеся без остатка на сумму своих цифр. K=10.

3.2. Трехзначные числа, делящиеся без остатка на произведение своих цифр. Поставить защиту от возможного деления на ноль. K=5.

3.3. Симметричные пятизначные нечетные числа, (например 34543 или 70507). K=5.

3.4. Шестизначные "счастливые" числа (сумма первых трех цифр равна сумме трех последних), кратные семи. K=10.

3.5. Четырехзначные числа, сумма цифр которых нечетна. K=3.

3.6. Семизначные числа, сумма цифр которых четна. K=12.

3.7. Напечатать все натуральные четырехзначные числа, в которых нет двух одинаковых цифр. K=6.

В задачах 3.8–3.11 (таблицы функций) необходимо вывести полностью оформленную таблицу в рамке и с элементами, указывающими содержание строк и столбцов. Точность – "Z" знаков после десятичной точки.

3.8. Целые степени N для чисел **π, ε** и их отношения **π/ε** N=1..6, Z=4.

3.9. Логарифмы целых N=2..20 по целому основанию M=2..10 (LogMN =Ln(N) / Ln(M)). Z=3.

3.10. Корни N-й степени (N=2..5) чисел 10K (k=2..5). Z=6. Тригонометрический и гиперболический синус, косинус, тангенс для 20 значений аргумента, взятых равномерно в диапазоне от 0 до **2π** Z=5.

3.11. Функции $\frac{N!}{10^N}$ и $\frac{N!}{N^{\sqrt{N}}}$ для целых N=10..20. Здесь важна методика вычисления функций для избежания переполнения разрядной сетки типов данных. Z=3.

3.12. Составить программу для получения таблицы из N значений функции Y=F(x) в равноудалённых точках интервала [a,b]. Исходные значения: N=5; A=-2; B=0.

4. Расчет конечных сумм

Сумма членов последовательности величин a_1, a_2, a_3,..., a_N называется конечной суммой $S_N=a_1+a_2+a_3+...+a_N$. Для некоторых последовательностей известны формулы расчета конечных сумм, например:

при $a_N=a_{N-1}+d$; $S_N=(a_1+a_N)\cdot N/2$; - арифметическая прогрессия,

при $a_N=a_{N-1}\cdot q$; $S_N=(a_1-a_N\cdot q)/(1-q)$; - геометрическая прогрессия,

где d и q - постоянные числа.

Здесь N-ый член последовательности выражается через (N-1)-ый член. Такие зависимости называются реккурентными.

Конечная сумма последовательности может быть неизвестна, тогда для ее расчета применяется алгоритм суммирования членов последовательности в цикле от 1 до N.

Пример 4. Составит программу расчета конечной суммы последовательности: 1·2+3·2+5·2+...+(2N-1)·2;

Решение. **В п**рограмме для расчета суммы используем формулу: $\mathbf{S_N = N(4N^2-1)/3.}$

```
Program primer4; {расчет конечной суммы}
 var a, S, Sn, i, N: word;
Begin
   write('Введите число членов суммы N='); readln(N);
   S:=0;
   For i:=1 to N do begin {цикл суммирования}
   a:=Sqr(2*i-1); S:=S+a; end;
   Sn:=N*(4*N*N-1) div 3;
   Writeln('Конечная сумма S=', S:-10:2);
   Writeln('Расчет конечной суммы по формуле Sn=', Sn:-10:2);
   Writeln('Нажмите Enter'); readln;
End.
```

Задачи для самостоятельной работы

В приводимых задачах необходимо составить программу расчета конечной суммы и сравнения полученного результата с контрольным значением. Число членов суммы вводится с клавиатуры с защитой от возможного неверного ввода данных.

№	Вид суммы	Контрольное значение
1.	$1+2+3+4+...+N$	$\frac{N(N+1)}{2}$
2.	$1+3+5+7+...+(2N-1)$	N^2
3.	$2+4+6+8+..+2N$	$N(N+1)$

4.	$1^2+2^2+3^2+4^2+...+N^2$	$\frac{N(N+1)(2N+1)}{6}$
5.	$1^2+3^2+5^2+...+(2N-1)^2$	$\frac{N(4N^2-1)}{3}$
6.	$1^3+2^3+3^3+4^3+...+N^3$	$\frac{N^2(N+1)^2}{4}$
7.	$1^3+3^3+5^3+...+(2N-1)^3$	$N^2(2N^2-1)$
8.	$1^4+2^4+3^4+4^4+...+N^4$	$\frac{(N^2+N)(2N+1)(3N^2+3N-1)}{30}$
9.	4+8+12+16+20+…+(2N-2)·2	$\frac{4N^2}{2}$
10.	3+9+15+21+27+…+(2N-1) ·3	$\frac{6N^2}{2}$
11.	$4^2+8^2+12^2+16^2+...+((2N-2)*2)^2$	$\frac{(16+(4N-4)^2)*N}{2}$
12.	$3^2+9^2+15^2+21^2+...+((2N-1)*3)^2$	$\frac{(9+(6N-3)^2)*N}{2}$

5. Расчет бесконечных сумм

Сумма членов бесконечной последовательности a_1, a_2, a_3, ..., a_N, … называется ***бесконечным рядом*** и записывается в виде:

$a_1+a_2+a_3+...+a_N+...$

Здесь a_N - общий член ряда. Сумма конечного числа членов ряда называется частичной суммой и обозначается "S_N". Если сумма членов бесконечного ряда имеет конечный предел "S", то ряд называется сходящимся. Для некоторых рядов получены формулы расчета суммы членов ряда. Например, сумма членов числового ряда:

1/(1·2)+1/(3·2)+1/(5·2)+...+1/((2N-1)·2)+...

имеет предел S =Pi2/8; и общий член a_N=1/((2N-1)·2), где N =1,2,3,...

Для сходящегося ряда вычисляется последовательность частичных сумм с заданной погрешностью. Абсолютная погрешность расчетов определяется по формуле Eps=abs(S-S_N), либо Eps=abs(a_N), если значение S неизвестно. Относительная погрешность расчетов

определяется по формуле Eps0=abs((S-S_N)/S), либо Eps0=abs(a_N/S_N). Частичные суммы вычисляются по формуле: $S_N=S_{N-1}+a_N$;

Для знакопеременного ряда следует добавить $k_1=-1$, а в цикле: $k_1:=-k_1$, $a_N=k_1 \cdot a_N$. В некоторых случаях "N"-ый член ряда выражается через "N-1"-ый, например, для ряда:

1+1/2!+1/4!+1/6!+...+1/(2N)!+...; N =0, 1, 2,...

общий член ряда вычисляется по формуле: $a_N = a_{N-1} \cdot k$;

Параметр $k=a_N/(a_{N-1})$ - коэффициент роста вычисляется предварительно (до написания программы). Для данного ряда

$a_N = 1/(2N)! = 1/(1 \cdot 2 \cdot ... \cdot (2N-2) \cdot (2N-1) \cdot 2 \cdot N)$;

$a_{N-1} = 1/(2(N-1))! = 1/((2N-2))! = 1/(1 \cdot 2 \cdot ... \cdot (2N-2))$;

$k=a_N/a_{N-1} = 1/((2N-1) \cdot 2 \cdot N)$.

Здесь $N! = 1 \cdot 2 \cdot 3 \cdot ... \cdot N$; - вычисление факториала числа "N", причем $0! = 1$.

Пример 5. Составить программу расчета бесконечной суммы: 1+1/2!+1/4!+1/6!+...+1/(2N)!+...; N =0, 1, 2,...

Решение. Для нахождения суммы будем использовать оператор цикла repeat…until.

```
Program primer5; {Сумма бесконечной последовательности}
   uses crt;
   const eps=0.001; {Погрешность
   var k,s,sn,e,a:real;
       n:integer;
     begin
      N:=0; a:=1; SN:=1; e:=2.7182828; S:=(e+1)/e;
      repeat N:=N+1; k:=1/((2*N-1)*2*N); a:=a*k;
      SN:=SN+a;
      Writeln('Частичная сумма Sn=', Sn:-11:6, '_ _ n=', n:2);
      until abs(S-Sn) < eps; {eps-допустимая погрешность расчетов}
      Writeln('   Сумма ряда S =', S :-11:6);
     end.
```

Задачи для самостоятельной работы

В приводимых задачах необходимо составить программу расчета бесконечной суммы обратных степеней числового ряда. Суммирование проводить, пока очередной член ряда по модулю не станет меньше заданной точности **ε**. Результат сравнить с точным значением ST., а погрешность сопоставить с величиной **ε**.

№	Вид суммы	N	Вид ряда	ST	□
1.	$\sum_{i=1}^{\infty} i^{-N}$	2	$1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots$	$\frac{\pi^2}{6}$	10-4
2.		4	$1+\frac{1}{2^4}+\frac{1}{3^4}+\ldots$	$\frac{\pi^4}{90}$	10-6
3.	$\sum_{i=1}^{\infty} (-1)^{i-1} i^{-N}$	2	$1-\frac{1}{2^2}+\frac{1}{3^2}-\ldots$	$\frac{\pi^2}{12}$	10-5
4.		4	$1-\frac{1}{2^4}+\frac{1}{3^4}-\ldots$	$\frac{7\pi^4}{720}$	10-7
5.	$\sum_{i=1}^{\infty} (2i+1)^{-N}$	2	$1+\frac{1}{3^2}+\frac{1}{5^2}+\ldots$	$\frac{\pi^2}{8}$	10-4
6.		4	$1+\frac{1}{2^4}+\frac{1}{3^4}+\ldots$	$\frac{\pi^4}{96}$	10-5
7.	$\sum_{i=1}^{\infty} \frac{(-1)^i}{(2i+1)^N}$	1	$1-\frac{1}{3}+\frac{1}{5}-\ldots$	$\frac{\pi}{4}$	10-4
8.		3	$1-\frac{1}{3^3}+\frac{1}{5^3}-\ldots$	$\frac{\pi^3}{32}$	10-5

В следующих задачах определить число членов ряда, необходимых для расчета с заданной погрешностью суммы членов ряда:

№	**Вид ряда**	**Общий член**	**Сумма**	**N=**
9.	$\frac{1}{(1*2)}+\frac{1}{(2*3)}+\frac{1}{(3*4)}+\ldots$	$\frac{1}{N*(N+1)}$	1	1,2,3, …
10.	$\frac{1}{(1*3)}+\frac{1}{(3*5)}+\frac{1}{(5*7)}+\ldots$	$\frac{1}{((2N-1)*(2N+1))}$	$\frac{1}{2}$	1,2,3, …
11.	$1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\ldots$	$\frac{(-1)^N}{N!}$	$\frac{1}{e}$	0,1,2,…
12.	$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots$	$\frac{1}{N!}$	e	0,1,2,…

6. Расчет функциональных рядов

Если члены ряда являются функциями аргумента "x", то ряд называется ***функциональным***. Расчет многих функций производится разложением функции в степенной ряд.

Пример 6. Составить программу расчета значения функции y=sin(x) при некотором значении "x" с использованием представления функции в виде ряда:

$$y_1=x-x^3/3!+...+(-1)^{(N+1)}x^{(2N+1)}/(2N+1)!+...$$

где a0=x, $a_N=k\cdot a_{N-1}$, $k=(-x^2)/(2N(2N+1))$, N=0, 1, 2,...

Решение. Каждый член ряда a_N при N>0 можно получить умножением предыдущего члена ряда a_{N-1} на коэффициент k. Приближенное значение функции y=sin(x) находится как частичная сумма "N" членов ряда. Погрешность вычисления значения функции "y" при некотором значении "x" зависит от количества членов ряда и значения "x", поэтому расчет заканчивается при $|a_N| < eps$, где eps-допустимая погрешность расчетов. В программе используем цикл с условием.

```
Program primer6;
Var y, y1, x, eps, a, k: real; n: Word;
Begin
 Writeln('Вычисление y=sin(x) с использованием ряда !');
 Write('Введите значение аргумента x='); readln(x);
 Write('Введите значение погрешности eps='); readln(eps);
 Writeln; y:=sin(x);
n:=0; a:=x; {a-первый член ряда}
y1:=a; {y1-первая частичная сумма ряда}
While abs(a) > eps do begin
n:=n+1; k:=-x*x/(2*n*(2*n+1)); a:=a*k; y1:=y1+a;
Writeln('Приближенное значение функции y1=',y1:-11:8,'_при n=', n)
end;
Writeln('Контрольное значение функции y=sin(x)=', y:-11:8);
Writeln('Нажмите Enter'); readln;
End.
```

Применение оператора Repeat ... в данном примере имеет вид:

```
Repeat "операторы" Until abs(a)<eps;
```

Операторы цикла с условием могут применяться для анализа правильности вводимых данных. Контроль входных данных обязателен для рядов в области сходимости. Например, функция arctg(x) разлагается в сходящийся степенной ряд в области |X|<1. Чтобы

учесть это обстоятельство мы будем использовать следующий фрагмент:

```
Repeat
   Write('введите значение |x|<1;  x=');
   readln(x)
until abs(x)<1;
```

Задачи для самостоятельной работы

Ряды с факториалами

Составить программу, содержащую функцию вычисления F(x) в виде бесконечного ряда с точностью 10-9.

В основной программе организовать вычисление этого ряда для двух значений x, запрашиваемых с клавиатуры, и проверку получаемых результатов путем сравнения с системной функцией F(x).

Распечатать таблицу значений функции для x, изменяющегося в диапазоне 0...A с шагом 0.1. Таблица должна содержать не более S строк.

№	F(x)	Разложение в ряд	A	S
1.	$\sin(x)$	$x-\frac{x^3}{3!}+\frac{x^5}{5!}-...+(-1)^N\frac{x^{2N+1}}{(2N+1)!}+..$	3	4
2.	$\exp(x)$	$1+\frac{x^1}{1!}+\frac{x^2}{2!}+...+\frac{x^N}{N!}+..$	4	5
3.	$\cos(x)$	$1-\frac{x^2}{2!}+\frac{x^4}{4!}-...+(-1)^N\frac{x^{2N}}{(2N)!}+..$	5	6
4.	$sh(x)$	$x+\frac{x^3}{3!}+\frac{x^5}{5!}+...+\frac{x^{2N+1}}{(2N+1)!}+..$	3	4
5.	$ch(x)$	$1+\frac{x^2}{2!}+\frac{x^4}{4!}+...+\frac{x^{2N}}{(2N)!}+..$	5	6
6.	$\frac{\sin(x)}{x}$	$1-\frac{x^2}{3!}+\frac{x^4}{5!}-...+(-1)^N\frac{x^{2N}}{(2N+1)!}+..$	3	4

Примечание: гиперболические синус и косинус вычисляются через экспоненту как $sh(x)=\frac{e^x-e^{-x}}{2}$, $ch(x)=\frac{e^x+e^{-x}}{2}$

Разложение функций в ряд Фурье

Составить программу с расчетом функции F(x) в виде ее разложения в ряд Фурье с точностью 10–4 (задача гармонического анализа). Определить, какие количества членов ряда Фурье

необходимо просуммировать для достижения указанной точности для значений аргумента X1=0.05, X2=0.5, X3=3.

№	F(x)	Разложение в ряд Фурье	Диапазон аргумента
7.	x	$2(\frac{\sin x}{1}-\frac{\sin 2x}{2}+\frac{\sin 3x}{3}-\dots)$	$-\pi < x < \pi$
8.	$\lvert x \rvert$	$\frac{\pi}{2}-\frac{4}{\pi}(\cos x+\frac{\cos 3x}{3^2}+\frac{\cos 5x}{5^2}+\dots)$	$-\pi \le x \le \pi$
9.	$\lvert \sin(x) \rvert$	$\frac{2}{\pi}-\frac{4}{\pi}(\frac{\cos 2x}{1\cdot 3}+\frac{\cos 4x}{3\cdot 5}+\frac{\cos 6x}{5\cdot 7}+\dots)$	$-\pi \le x \le \pi$
10.	$\frac{\pi - x}{2}$	$\sum_{i=1}^{\infty}\frac{\sin(ix)}{i}$	$-\pi < x < \pi$
11.	$\frac{\pi^2-3x^2}{12}$	$\sum_{i=1}^{\infty}(-1)^{i+1}\frac{\cos(ix)}{i^2}$	$-\pi \le x \le \pi$
12.	$\frac{\pi^2 x-\pi x^2}{8}$	$\sum_{i=1}^{\infty}\frac{\sin((2i+1)x)}{(2i+1)^3}$	$0 \le x \le \pi$

7. Расчет банковских вкладов

При расчете суммы последовательностей в некоторых случаях "N"-ый член последовательности определяется через сумму предыдущих членов, например, $a_N=pS_{N-1}$, тогда $S_N=S_{N-1}+a_N=S_N(1+p)$, и конечную сумму можно рассчитать по формуле:

$S_N=S_0(1+p)N$, где "S_0" - начальная сумма.

Пример 7. Составить программу вычисления конечной суммы денежного вклада в банк через N месяцев при ежемесячной процентной ставке "pr" (5% соответствует pr=5).

Решение. Для решения задачи удобно будет использовать цикл с параметром for, так как параметр цикла известен.

```
Program primer7; {расчет конечной суммы вклада в банк}
 var S, Sn, pr: real; i, N: integer;
Begin
      Write('Введите начальную сумму вклада S='); readln(S);
```

```
        Write('Введите процент по вкладу pr='); readln(pr);
        Write('Введите количество месяцев вклада N='); readln(N);
        For i:=1 to N do S:=S*(1+pr/100.); {цикл произведений}
        Writeln('Конечная сумма вклада S=', S:-10:2);
        {Оператор для расчета "Sn" напишите самостоятельно}
Writeln('Расчет конечной суммы вклада по формуле Sn=', Sn:-10:2);
       Writeln('Нажмите Enter');
        readln
End.
```

Задачи для самостоятельной работы

Составить программу расчета роста банковского вклада по месяцам со сроком в полтора года. Программа запрашивает с защитой от неверного ввода указанные данные и выводит таблицу роста вклада по месяцам. Также рассчитывается указанная дополнительно информация.

7.1. Ввести:

- начальный размер вклада (1000…10000),
- размер периодических платежей (от 1% до 10% от начального вклада),
- размер процентной ставки (0.5% … 4% в месяц).

В таблицу роста вклада по месяцам включить дополнительный столбец роста вклада в предположении отсутствия периодических платежей.

7.2. Ввести:

- начальный размер вклада (2000…20000),
- размер процентной ставки по вкладу (1%…3% в месяц),
- размер периодических платежей (от 0 до размера начального вклада).

Дополнительно определить количества месяцев, необходимые для роста вклада в полтора и в два раза.

7.3. Ввести:

- начальный размер вклада (3000…30000),
- размер процентной ставки (1% … 4% в месяц),
- размер периодических платежей (от 3% до 30% от начального вклада),

Дополнительно вывести таблицу, показывающую влияние размера периодических платежей на количество месяцев, необходимое для роста вклада в 3 раза (варьировать периодические платежи от 5% до 50% от начального вклада с шагом 5%).

7.4. Вычислить конечную сумму денежного вклада в рублях при открытии валютного счета в банке через несколько месяцев при ежемесячной процентной ставке 1% и росте курса валютной единицы по отношению к рублю 2% в месяц. Программа запрашивает ввод суммы и срока вклада, и выводит на экран конечную сумму в рублях. Проверить расчет вычислением по формуле: $S_N=S_0\cdot(1+pv)\cdot N\cdot(1+pi)\cdot N$,
где pv, pi - банковский и инфляционный коэффициенты.

7.5. В п. 7.4 сделать ежемесячный ввод банковской процентной ставки и роста курса валюты при подсчете конечной суммы.

7.6. В п. 7.4 сделать ежегодный ввод банковской процентной ставки и роста курса валюты при подсчете конечной суммы.

7.7. Определить число месяцев, через которое начальная сумма вклада в банк увеличится более чем в три раза. Процентная ставка равна 5% в месяц. Программа выводит на экран ежемесячное значение конечной суммы.

7.8. Определить число месяцев, через которое начальная сумма валютного вклада в банк увеличится в рублях более чем в три раза. Процентная ставка равна 0. 3% в месяц, а курс валюты растет по отношению к рублю 1% в месяц. Программа выводит на экран ежемесячное значение конечной суммы в рублях.

7.9. Рассчитать зарплату купюрами достоинством 1, 3, 5 рублей. Программа запрашивает ввод величины зарплаты и выводит на экран всевозможные комбинации числа купюр, сумма которых равна заработной плате. Начальные значения параметров циклов равны нулю, конечные значения параметров циклов находятся делением величины зарплаты на 1, 3, 5.

7.10. Составить программу вычисления величины дохода по вкладу. Процентная ставка (в процентах годовых) и время хранения (в днях) задаются во время работы программы.

7.11. Составить программу вычисления стоимости покупки с учетом скидки. Скидка 10% представляется, если сумма покупки представляется больше 1000 сом.

7.12. Составить программу вычисления стоимости покупки с учетом скидки. Скидка 3% представляется в том случае, если сумма покупки больше 500 сом, в 5% - если сумма больше 1000 сом.

8. Расчет произведений

Для вычисления значения функции или выражения целесообразно организовать цикл, в котором не только вычисляется текущие значение, но и накапливается их произведения путем умножение полученного множителя к произведению предыдущих. Формула ис-

пользуемые для накопления, имеет вид $z_i=z_{i-1}y_i$, а начальное значение произведения должно быть равно единице.

Пример 8. Написать программу, следующих произведений:

$$1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n-1) = \prod_{i=1}^{n} (2i-1)$$

Решение. В качестве начального значения произведения берем p=1, а шаг – 2. Для этого будем использовать цикл While … Do.

```
Program primer8;
  uses crt;
  var p:longint;
      i,n:integer;
  begin clrscr;
     p:=1;i:=1;
     readln(n);
     while i<2*n-1 do
       begin
         p:=p*i;
         i:=i+2;
       end;
  writeln(p);
  end.
```

Задачи для самостоятельной работы

В следующих задачах необходимо составить программу расчета произведений с помощью циклических программ.

8.1. $p = \prod_{n=1}^{8} \frac{1}{n+1}$

8.2. $s = \prod_{k=1}^{15} \frac{i}{(i+1)^2}$

8.3. $p = a \cdot b \cdot c - \prod_{n=1}^{20} \frac{1}{n^2}$

8.4. $p = \prod_{n=1}^{12} (e^n + \ln n)$

8.5. $p = \prod_{i=1}^{30} \frac{1}{e^i}$

8.6. $p = \prod_{i=1}^{10} \frac{1}{e^i} + 1$

8.7. $p = \prod_{i=1}^{7} \frac{1}{1+i} + 1$

8.8. $p = \prod_{i=1}^{12} \frac{1}{i^2 + 1} + 1$

8.9. $p = \prod_{i=1}^{10} \frac{1}{e^i + 2}$

В задачах 8.10. – 8.12. составить программу, содержащую функцию вычисления F(x) в виде разложения как бесконечного произведения с точностью 10-4.

№	F(x)	Разложение	x1	x2
8.10	$\cos(x)$	$\prod_{N=1}^{\infty}(1 - \frac{4x^2}{(2N-1)^2\pi^2})$	0.05	10
8.11	$sh(x)$	$x \cdot \prod_{N=1}^{\infty}(1 + \frac{x^2}{N^2\pi^2})$	0.1	20
8.12	$ch(x)$	$\prod_{N=1}^{\infty}(1 + \frac{4x^2}{(2N-1)^2\pi^2})$	0.2	30

В основной программе организовать вычисление этого ряда для двух значений x и проверку получаемых результатов путем сравнения с системной функцией F(x). При этом вывести количество сомножителей, требуемое для достижения необходимой точности расчетов.

Распечатать таблицу значений функции для аргумента, изменяющегося в диапазоне ***0..2π-π/50*** с шагом ***π/50*** (размер таблицы 5x20 чисел).

9. Работа с массивами

Массив - упорядоченные данные одного типа. Возможно создание массива, включающего массив другого типа. Массивом часто обозначают характеристики объектов одного типа, имеющих одинаковые единицы измерения. Массив состоит из элементов, имеющих порядковые номера, т. е. элементы массива упорядочены. Таким образом, если объекты одного типа обозначить именем, например "A", то элементы объекта будут A[1], A[2] и т. д.

Присвоение случайные значения к элементам массива в диапазоне от -30 до+40 производятся следующем виде:

Randomize; for i:=1 to 100 Do R[i]:=-30+Random(71);

Присвоим значения семи элементам массива "A" оператором Readln:

```
For i:=1 to 7 Do begin Write('Введите A[', i, ']='); Readln(A[i]) end;
```

При выводе массива на экран удобно размещать данные в виде таблицы - в несколько колонок. Для вывода обозначений переменных ("шапки таблицы") можно использовать операторы вывода символов в цикле, например:

```
For j:=1 to 66 do Write('-'); Writeln;
For j:=1 to 3 do Write('|_ _ Фамилия _ _| _ оценка _|'); Writeln;
For j:=1 to 66 do Write('-'); Writeln;
```

- шапка для вывода в три пары колонок значений переменных "S" и "A". Шапка занимает 66 позиций (по ширине экрана в текстовом режиме размещается 79 символов и пробел). Оператор Writeln; переводит курсор на новую строчку.

Вывод значений ста элементов массивов "S" и "A" в три пары колонок, произведем операторами:

```
For i:=1 to 100 do begin Write('|', s[i]:11,'|', a[i]:8, '|');
        if (i mod 3) =0 Then Writeln;
        if (i mod 60) =0 then readln end;
```

В этом случае данные таблицы полностью не умещаются на экране и можно задержать прокрутку экрана при выводе данных, применяя оператор Readln после вывода, например, 20 строк.

Пример 9. Определить массив первых ста натуральных чисел, сумма цифр которых делится на 15, и распечатать его в виде матрицы 10x10.

Решение. Для решения задачи сначала определим числа, сумма цифр которых делятся на 15, и одновременно эти числа присваиваем в массив. После каждого найденного такого числа, значения счетчика увеличиваем на 1. Когда значения счетчика достигнет 100, завершаем цикл и распечатаем массив как таблица (см. раздел 3).

```
Program primer9;
Uses crt;
const n=100;
var a:array[1..n] of integer;
   i,j,t,k,p,code:integer;
   s:string;
begin clrscr;
i:=69; {первое натуральное число, сумма которого делится на 15
это число 69}
 t:=1; {счетчик чисел}
 repeat
  str(i,s);
```

```
  k:=0;
  for  j:=1 to length(s) do
   begin
    val(s[j],p,code);
    k:=k+p;
   end;
  if k mod 15=0 then begin a[t]:=i;t:=t+1;end;
  i:=i+1;
  until t>n;
  for i:=1 to n do
     write(a[i]:8);
end.
```

Задачи для самостоятельной работы

9.1. Определить массив кубов первых ста натуральных чисел и распечатать его в виде матрицы 10x10.

9.2. Определить массив первых 196 натуральных нечетных чисел, не кратных трем, и распечатать его в виде матрицы 14x14.

9.3. Определить массив первых 120 натуральных чисел, сумма цифр которых кратна 10, и распечатать его в виде матрицы 10x12.

9.4. Определить и вывести массивы чисел X и Y, где X=0, 0.2, 0.4,...20, Y=X–20cosX. Затем вывести в 10 колонок с заголовками сначала положительные элементы массива Y, а затем отрицательные. После таблицы вывести значения Y_min и Y_max.

9.5. Определить массив Y=sinX–cosX, где X=0, 0.2, 0.4,...60. Распечатать в 10 колонок с заголовками сначала номера отрицательных элементов массива, а затем положительных. После таблицы вывести значения Y_min и Y_max.

9.6. Определить массу тела в форме куба со стороной L, плотность которого равномерно убывает от центра к краям. В центре куба плотность равна P_c, а на вершинах куба – $P_k<P_c$.

9.7. Решить задачу 9.1 для прямоугольной пластины размером L_x на L_y, причем плотность (как масса на единицу площади) убывает обратно пропорционально расстоянию до центра пластины.

9.8. Сгенерировать и вывести на экран массив из 500 случайных чисел со значением в диапазоне от 1 до 9. Определить массив из 9 элементов, где будут записаны количества пар одинаковых соседних элементов сгенерированного массива.

9.9. В случайном массиве 100 реальных чисел от 0 до 1 найти минимум и максимум суммы трех элементов.

9.10. Слить два массива A и B по 100 элементов в массив C из 200 элементов по одному из следующих вариантов:

а) элементы массива A имели в C нечетные номера;

б) элементы массива A имели номера от 51 до 150;

в) элементы A и B чередовались по 10 штук;

г) вначале шли элементы меньше среднего значения по всему массиву C.

9.11. Сгенерировать и вывести на экран массив A размером 10x10 из случайных реальных чисел в диапазоне от 1 до 9. Определить и вывести массив B также размером 10x10 по одному из следующих вариантов:

а) элементы в последней строке определялись как суммы элементов по соответствующим столбцам,

б) элементы в последнем столбце определялись как произведение элементов по соответствующим строкам,

в) элементы главной диагонали определялись как суммы по соответствующим строкам,

г) элементы главной диагонали определялись как произведение по соответствующим столбцам,

д) элементы главной диагонали определялись как произведение по соответствующему столбцу и строке,

9.12. Сгенерировать и вывести на экран массив 9x10 случайных целых чисел в диапазоне от 1 до 9. Определить и вывести массив B 10x10 (с дополнительными элементами десятого столбца или главной диагонали) по одному из вариантов п.9.11.

10. Работа со строками

Переменная строкового типа (String) может рассматриваться как массив элементов символьного типа (Char). Например, если в программе определены переменные

S: string; C: char; и задано S:='Москва', то S[1]='M', S[2]='o' и т. д. и возможно присвоение, например: C:=S[1]; Таким образом, строка может рассматриваться как линейный массив символов. Элементы массива, составляющие строку можно переставлять местами и получать новые слова, например:

```
for i:=1 to N div 2 do begin C:=S[i]; S[i]:=S[N-i+1]; S[N-i+1]:=C end;
Writeln(S); {исходное слово выведется справа налево: "авксоМ"}
```

Здесь N:=ord(S[0]);-число символов в переменной "S" хранится в переменной S[0]. Функция "ord" преобразует символьный тип в целый. N div 2-количество перестановок для слова из "N" символов. В переменной "C" запоминается значение i-го элемента, который меняется с элементом, симметричным относительно середины строки.

Можно производить поиск и замену заданного символа в строке, например:

```
for i:=1 to N do if S[i]='_' then writeln('найден символ пробел');
for i:=1 to N do if S[i]='/' then S[i]:='\'; {замена символа "/" на "\"}
```

Заменяя или переставляя символы в строке по определенной схеме (закону) можно зашифровать строку. Для дешифровки используется, как правило, схема обратной перестановки или замены символов. Например:

```
for i:=1 to N do S[i]:=chr(ord(S[i])+2); {преобразование исходных
символов в символы с кодом большим на две единицы}
```

Напомним, что все используемые в MS-DOS символы имеют ASCII коды от 0 до 255.

Здесь удобно также использовать функции Pred(C); и Succ(C);

Пример 10. Составить программу, которая удаляет начальные пробелы из строки, введенной с клавиатуры.

Решение. Для решения сначала определим длины строки и организуем цикл до первого найденного символа, отличного от пробела (' '). Затем в цикле удаляем пробелы.

```
Program primer10;
 Uses crt;
 Var  st:string[80]; {строка}
   Begin clrscr; {очистит экран}
      Writeln('Удаление начальных пробелов строки.');
      Writeln('Введите строку.');
      Readln(st);
   While (pos(' ', st)=1) and (Length(st)>0) do
     Delete(st,1,1);
    Writeln('Строка без начальных пробелов.');
Readln;
End.
```

Задачи для самостоятельной работы

10.1. Зашифровать введенную с клавиатуры строку заменой исходных символов на символы с кодом, большим на три единицы. Провести дешифровку.

10.2. Зашифровать введенную с клавиатуры строку, поменяв местами первый символ с третьим, второй с четвертым и т. д. Провести дешифровку.

10.3. Зашифровать введенную с клавиатуры строку, поменяв местами первый символ со вторым, третий с четвертым и т. д. Затем провести дополнительную шифровку результата смещением кода. Провести дешифровку.

10.4. Зашифровать введенную с клавиатуры строку смещением кода, которое зависит от номера символа в строке. Для коротких строк можно использовать линейную зависимость, для длинных – комбинации функций MOD и DIV. Провести дешифровку.

10.5. Найти и заменить определенный символ в строке, введенной с клавиатуры. Программа должна запрашивать заменяемый и заменяющий символы, а также подтверждение каждой замены символа с сообщением о номере его позиции в строке.

10.6. Определить и вывести на экран номера позиций и количество повторений запрашиваемого символа в строке, введенной с клавиатуры.

10.7. Определить количество слов в строке, введенной с клавиатуры (за слова принимать части строки, отделяющиеся друг от друга одним или несколькими пробелами).

10.8. Определить самое короткое и самое длинное слово во введенной строке.

10.9. Подсчитать в строке число букв A и B, если букв A больше, чем B, то удалить в строке все символы B.

10.10. Определить, сколько раз в строке встречается данное слово.

10.11. Дана символьная строка, заканчивается точкой. Найти длину самого длинного и самого короткого слова.

10.12. Вводится последовательность латинских букв, признаком конца которой является пробел. Напечатать эту последовательность, упорядоченную по алфавиту.

11. Преобразования массивов

В цикле удобно определять сумму элементов массива, наибольший (наименьший) элемент и создавать новые массивы, удовлетворяющие некоторому условию, например:

```
s:=0; for i:=1 to 100 do s:=s+a[i]; {s-сумма элементов массива}
a_max:=a[1]; for i:=1 to 100 do {поиск наибольшего элемента a[j]}
if a[i] > a_max then begin a_max:=a[i]; j:=i end;
j:=0; k:=0;
for i:=1 to 100 do {создание новых массивов с элементами: b[j] >=0,
c[k] <0}
if a[i] >=0 then begin j:=j+1; b[j]:=a[i] end
 else begin k:=k+1; c[k]:=a[i] end;
j:=0; k:=8;
for i:=1 to 100 do {создание массива номеров "M" для элементов:
a[i]>a[k]}
```

if a[i] > a[k] then begin j:=j+1; M[j]:=i end;

Пример 11. Написать программу, которая выводит минимальный элемент массива целых чисел, введенного с клавиатуры.

Решение. Представляем следующий вид экрана во время работы программы:

Поиск минимального элемента массива.

Введите в одной строке элементы массива (5 целых чисел) и нажмите <Enter>

23 0 45 -5 12

Минимальный элемент массива: -5

Для определения минимального элемента массива в цикле используем операции сравнения.

```
Program primer11; {Поиск минимального элемента}
 Uses crt;
 Const r=5; {размер массива}
 Var a:array[1..r] of integer;
       Min: integer; {минимальный элемент массива}
       I:integer;
Begin clrscr; {очистить экран}
      Writeln('Поиск минимального элемента массива.');
      Writeln('Введите в одной строке элементы массива', r, 'целых чисел) и нажмите <Enter>');
      Write('- >');
For I:=1 to r-1 do
     Read(a[i]);
  Readln(a[r]);
  Min=a[1];
For i:=2 to r do
     If a[i] < Min then Min:=a[i];
     Writeln(' минимальный элемент массива:', Min);
Readln;
End.
```

Задачи для самостоятельной работы

11.1. Определить массив Y=X2–X3, где X=-1,-0.9,-0.8,...2. Выделить из него массив положительных значений Yp и вывести этот массив на экран с сортировкой по возрастанию в 4 колонки.

11.2. Определить массив Y=cosX–cos(X2), где X=-5,-4,-3,...10. Выделить из него массив отрицательных значений Ym и вывести этот массив на экран с сортировкой по убыванию в 5 колонок.

11.3. Определить массив Y=X2–7cosX, где X=1.0, 1.2, 1.4,...10. Выделить из него массив положительных значений Yp и вывести этот массив на экран с сортировкой по возрастанию в 10 колонок.

11.4. Сгенерировать и вывести на экран массив 10x10 из нулей и единиц так, чтобы нулей было в несколько раз больше. Определить и вывести массив B как одно из геометрических преобразований массива A:

а) разворот на 90 градусов по часовой стрелке;

б) разворот на 90 градусов против часовой стрелки;

в) разворот на 180 градусов;

г) зеркальное отражение по горизонтали;

д) зеркальное отражение по вертикали;

е) зеркальное отражение по главной диагонали;

ж) зеркальное отражение по побочной диагонали.

11.5. В массиве ста случайных реальных чисел найти максимальный и минимальный элементы и переставить их на первое и последнее места соответственно.

11.6. Расширить массив из п. 11.5 – добавить по новому элементу между каждой парой соседних элементов как их сумму–всего 99 штук.

11.7. Среди тех строк целочисленной матрицы, которые содержат только нечётные элементы, найти строку с максимальной суммой модулей элементов.

11.8. Дан одномерный массив чисел, состоящий только из нулей и единиц. Последовательность из не менее чем K рядом стоящих единиц называется "K-батареей". Для K = 2, 3, 4, 5 подсчитать количество K - батарей

11.9. В одномерном массиве размещены: в первых n элементах значения аргумента, в следующих - соответствующие им значения функции. Напечатать элементы этого массива в виде двух параллельных столбцов.

11.10. Если в массиве нет повторяющихся элементов, то упорядочить его по возрастанию.

11.11. В массиве X из n элементов каждый из элементов равен 0, 1 или 2. Переставить элементы массива так, чтобы сначала располагались нули, затем единицы и двойки. Дополнительный массив не использовать.

11.12. Вводится два массива символов A и B из n и m<n элементов. Определить является ли массив B подстрокой массива A. Если является, то с какого элемента, а если не является то вывести 0.

12. Текстовые файлы

Текстовые файлы представляют собой совокупность строк переменной длины с последовательным доступом к данным, т. е. данные записываются на диск и считываются только последовательно. Информация в текстовых файлах хранится в символьном (текстовом) виде. При записи числовых или логических значений происходит автоматическое преобразование данных в символьный тип, а при считывании данные автоматически преобразуются в машинные коды.

Файловая переменная " f " описывается оператором Var f: Text;

В программе файловая переменная " f " связывается с именем файла на диске, оператором:

Assign(f, 'Name_f');

где Name_f - имя файла.

Например, переменная "f" связывается с файлом "file.dat" оператором Assign(f, 'file. dat'); если файл находится в текущем каталоге, иначе к нему указывает дорожка, например: 'C:\Pascal\Work\file.dat'. Связывание файловой переменной "f" с именем файла на диске аналогично присвоению "f" значения.

Для записи данных в файл его необходимо открыть оператором **ReWrite(f)**; При этом на диске создается новый файл.

Имя файла указано в операторе Assign(f,‘Name_f’); Данные записываются в файл оператором Write(f,"сп"); или Writeln(f,"сп"); Причем, оператор Writeln(f,"сп"); устанавливает в конце данных управляющие символы: #13, #10. Здесь обозначено "сп"-список переменных. Повторное применение оператора ReWrite(f); стирает содержимое файла и устанавливает указатель на начало файла.

Для считывания данных из файла его необходимо открыть оператором Reset(f);

После окончания работы с файлом его необходимо закрыть процедурой Close(f);

При работе с числовыми данными необходимо учитывать, что числа в файле должны отделяться друг от друга хотя бы одним пробелом. Следовательно, при записи числовых данных в файл необходимо использовать формат с достаточным количеством позиций для вывода.

Приведем примеры операторов записи и считывания данных.

```
Assign(f1,'File1.dan'); {назначить переменой f1, имя файла: File1. dan}
ReWrite(f1); {открыть файл для записи в первой программе}
Writeln(f1,'Значения "X","Y"'); {начать запись}
    For i:=1 to N do begin
```

```
 X:=0.5*i; Y:=Ln(X); {пример расчета значений переменных}
 write(f1, X:6:2, Y:10:4); {записать данные в файл File1. dan}
 If i mod 5 =0 then writeln(f1) {записать символ #13}
 end;
Close (f1); {закрыть файл в первой программе}
Assign(f2,'File1.dan');
Reset(f2); {открыть файл для чтения во второй программе}
 Readln(f2); {пропустить первую строчку}
 For i:=1 to N do begin
 read(f2, a[i], b[i]); {считать данные в массивы "A" и "B"}
 If i mod 5 =0 then readln(f2) {считать символ #13}
 end;
Close (f2); {закрыть файл во второй программе}
```

При работе со строковыми данными необходимо указывать длину переменной типа String в операторе описания типов переменных, иначе оператором Read(f, S); в строковую переменную "S" считывается до 255 символов, а оператором Readln(f, S); считываются все символы до #13, но не более 255, причем пробелы в конце строки игнорируются.

Пример 12. Составить программу для считывания строковых и числовых данных из файла и их записи в другой файл.

Решение. Для решения задачи будем использовать процедуры assign, reset, rewrite, read и write.

```
Program primer12;
Uses crt;
var c: char; j, i: word;
 s: array[1..10] of string[12];
 a: array[1..10, 1..6] of word;
 f1, f2: text;
BEGIN clrscr;
 assign(f1, 'F1.txt'); reset(f1);
 assign(f2, 'F2.txt'); rewrite(f2);
 for i:=1 to 10 do begin read(f1, s[i]); {считывание строки}
 for j:=1 to 6 do read(f1, a[i,j]); {считывание шести чисел}
 readln(f1) {считывание символа конца строки}
 end;
 for c:='A' to 'Я' do {цикл по перебору символов}
 for i:=1 to 10 do
 if s[i,1] =c then begin
```

```
        write(f2, s[i]); {запись строк в алфавитном порядке первых символов}
        for j:=1 to 6 do write(f2, a[i,j]:2); {запись шести чисел}
        writeln(F2)
        end;
        close(f1); close(f2);
  end.
```

Задачи для самостоятельной работы

12.1. Записать в новый файл f1.pas 100 реальных случайных чисел (от –100 до 100) в 5 колонок с точностью 6 знаков после десятичной точки. Файлу установить атрибут "Read-Only".

12.2. Считать из файла f1.pas (см. задачи 12.1) числа и вывести их в файл f2.pas-сначала отрицательные, а затем положительные в 10 колонок с точностью 2 знака после десятичной точки. Файлу f2.pas установить атрибут "Hidden".

12.3. Определить массив Y=X[2]-X[1], где X =-2.0, -1.8, ... ,1.2. Выделить массив положительных значений Y_p и вывести его в файл с сортировкой по возрастанию в 5 колонок.

12.4. Определить массивы чисел X и Y, где X=-5, -4, -3, ... 10, Y=cosX[1]-cosX[2]. Выделить из Y массив отрицательных значений Y_m и вывести этот массив в файл с сортировкой по убыванию в 5 колонок. Следом вывести массив соответствующих Y_m значений X.

12.5. Функция Бесселя порядка N имеет вид

$$J_N(x) = (\frac{x}{2})^N \sum_{K=0}^{\infty} (-1)^K \frac{(x/2)^{2K}}{K!(K+N)!};$$

Создать текстовый файл, содержащий таблицу функции Бесселя 5-го порядка при x =3.3, 3.6, 3.9,...12 с точностью 8 знаков после десятичной точки.

12.6. Создать текстовый файл с таблицами функции Бесселя порядка от 0 до 4 (всего 5 функций) для x =0.5, 1.0, 1.5, ... , 10 с точностью 6 знаков после десятичной точки.

12.7. Рассчитать значения функции Y=sin(x) при изменении "x" с шагом 0. 01 в диапазоне от 0 до 3. Записать в файл F1. txt значения "x" и "y". Во второй программе считать из файла F1.txt значения "x" и "y", рассчитать значения функций $Z_1=y_2$, $Z_2=y_3$ и добавить значения Z_1, Z_2 в конец файла F1.txt.

12.8. Выполнить задачи 12.7 для функции $Y=e^x$.

12.9. Записать в файл F1. d массив отрицательных целых чисел "A" по убыванию, а в файл F2. d массив положительных целых чисел "A" по возрастанию. Массив "A" из 25 целых чисел задается в диапазоне от -10 до+10 функцией Random.

12.10. Записать в файл F1.dat массив четных целых чисел "A" по убыванию, а в файл F2.dat массив нечетных целых чисел "A" по возрастанию. Массив "A" из 30 целых чисел задается в диапазоне от 0 до 20 функцией Random.

12.11. Записать в конец файла F1.txt список из фамилий (в алфавитном порядке) с оценками по пяти предметам. Список фамилий (в произвольном порядке) с оценками считывается из файла F1.txt, предварительно набранного в редакторе текста.

12.12. Составить программу, записывающую в файл простые числа из интервала [m, n].Переписать в другой файл те из них, которые оканчиваются на 9.

13. Зашифрование текстовых файлов

Заменяя или переставляя символы в тексте по определенной схеме (закону) можно зашифровать текст в файле. Для дешифровки файлов как мы выше говорили, используется, схема обратной перестановки или замены символов.

Пример 13. Составить программу, которая зашифрует текстовый файл Asan.txt заменой исходных символов на символы с кодом большим на две единицы.

Решение. Для зашифрования данного текстового файла процедурой eof(f) организуем цикл до конца файла, затем в цикле одновременно считываем каждый символ и увеличим его код на две единицы. Полученный символ перепишем на новый файл.

```
Progpam primer13;
Uses crt;
Var f,g:text; {f - зашифруемый файл, g – зашифрованный файл}
      Ch:char;
Begin clrscr;
Assign(f,'Asan.txt'); Assign(g,'Uson.txt');
Reset(f); {открытие зашифруемого файла}
Rewrite(g); {создание зашифрованного файла}
While not eof(f) do {цикл до конца файла}
 Begin
    read(f, ch); {чтение символа из файла f}
    ch:=chr(ord(ch)+2); {преобразование исходных
        символов в символы с кодом большим на две единицы}
    write(g, ch);{запись символа в файл g}
```

```
        end;
      close(f); close(g); {закрытие файлов}
end.
```

Задачи для самостоятельной работы

13.1. Дан текстовый файл. Зашифровать тексты, заменой исходных символов на символы с кодом большим на три единицы. Провести дешифровку.

13.2. Зашифровать текстовый файл, заменой символов на символы с кодом меньшим на две единицы. Провести дешифровку.

13.3. Зашифровать текстовый файл, поменяв местами первый символ со вторым, третий с четвертым и т. д. Провести дешифровку.

13.4. Зашифровать текстовый файл, поменяв местами первый символ с третьим, второй с четвертым и т. д. Провести дешифровку.

13.5. Зашифровать текстовый файл, заменяв английскими буквами на русский. Провести дешифровку.

13.6. Зашифровать текстовый файл состоящих из цифр, заменяв цифры на его двоичными кодами. Например, 0 – 0000, 1 - 0001, 2 – 0010, 3 – 0011, …, 9 – 1001. Провести дешифровку.

13.7. Зашифровать файл заменяв буквы на его порядковыми номерами. Например, а – 01, б – 02, в – 03, …, я – 32. Провести дешифровку.

Примечание к задачам 13.1-13.7: Записывайте зашифрованный и дешифрованный текст на новый файл.

13.8. Зашифровать текст, считанный из файла F1.txt, предварительно набранного в редакторе текста и записать в конец файла F1.txt. Во второй программе дешифровать текст и добавить в конец файла F1.txt. Алгоритм шифровки разработать самостоятельно.

13.9. Составить программу, которая подсчитывает количество символов в заданном текстовом файле.

13.10. Дан файл целых чисел преобразовать его так, чтобы сначала в нем шли числа кратные трем, а затем дающие при делении на три единицу.

13.11. В текстовом файле хранятся вещественные квадратные матрицы порядка n. Преобразовать файл, удалив из матриц последнюю строку и столбец.

13.12. Дан текстовый файл на русском языке. Записать в новый файл все согласные буквы в алфавитном порядке.

14. Работа с блочными файлами

С помощью ***блочных файлов*** возможно выполнение небуферизованных операций ввода-вывода, осуществляемых непосредственно между переменными программы и внешней дисковой памятью. По умолчанию элементами блочного файла являются блоки по 128 байт. Блочный файл может представлять любой дисковый набор данных. Поэтому такие операции, как Erase и Rename, могут быть выполнены с помощью блочных файлов.

Описание типа блочного файла состоит из ключевого слова file.

В паскале имеется два типа блочного файла:

Типизированные файлы. Файлы этого вида состоят из элементов одинакового типа, то есть в них нельзя записывать (или читать) значения переменных разных типов, в отличие от текстовых файлов.

Объявляются типизированные файлы так:

var f: file of тип_элемента;

В качестве типа элемента можно использовать как простые типы, так и структурированные (массивы, записи и т.п.).

Нетипизированные файлы. Нетипизированный файл, в отличие от типизированного, используется для хранения разнородной информации, а не одинаковых элементов. В него можно записывать (а также читать) значения переменных практически любого типа (простых типов, массивов, записей, и т. п.). Описываются переменные, соответствующие нетипизированным файлам, следующим образом:

var f: file;

Для чтения и записи процедуры read и write не подходят. Используются такие процедуры:

BlockRead(var f: file; var buf; count: word [;var result: word]); – читает в переменную **Buf count** записей из файла, переменная result показывает, сколько записей было скопировано в действительности. Под записью понимается «кусок» файла в несколько байт, размер записи можно установить при открытии файла, например: reset(f,1).

BlockWrite(var f: file; var buf; count: word [; var result: word]); – записывает указанное количество записей в файл. Если для открытия используется rewrite, то во втором её параметре также можно указать размер записи.

Пример 14. Составить программу, производящую запись числовых данных в текстовый и типизированный файл.

Решение. Запись чисел в текстовый файл рассмотрен в разделе 12. Для записи чисел в типизированный файл объявляем файл с помощью служебного слова Var, как типизированный файл.

```
Program primer14;
Uses crt;
var Extfile: file of Extended;
     Textfile: text;
     x,y: Extended;
     i: word;
Begin clrscr;
    Assign(Textfile, 'table.txt');
    Rewrite(Textfile);
    x:=0.0;
    for i:=1 to 100 do
      begin
        y:=sin(x);
        Writeln(Textfile, y);
      End;
      Close(Textfile);
   Assign(Extfile, 'table.ext');
    Rewrite(Extfile);
    x:=0.0;
    for i:=1 to 100 do
      begin
        y:=sin(x);
        Writeln(Textfile, y);
      End;
      Close(Extfile);
   Readln;
End.
```

Задачи для самостоятельной работы

14.1. Определить файл f1.pas (создаваемый по задаче 12.1) как типизированный файл из символов. Подсчитать, сколько в нем встречается семерок, а также определить и вывести в текстовый файл массив номеров позиций этих семерок.

14.2. Составить программу создания типизированного файла чисел типа double из текстового файла f1.pas (задачи 12.1).

14.3. Составить программу создания четырех типизированных файлов из записей типа "число и символ". Типы чисел – integer, word, real, double, а в качестве символа записывать #14 (перевод строки). Это позволит при просмотре созданных файлов текстовым редактором оценить принцип кодирования чисел в типизированных файлах (каждая кодировка числа пойдет с новой строки).

14.4. Составить программу сцепления двух файлов с созданием третьего файла (аналог команды Copy MS-DOS), причем имена файлов задаются как параметры командной строки.

14.5. Составить программу подсчета контрольных сумм файла, например, суммы кодов символов на четных и на нечетных позициях.

14.6. Составить программу шифровки и дешифровки файла методом простой символьной подстановки. Для этого сгенерировать таблицу из N смещений кодов символов и записать ее в файл. Использовать эту таблицу при генерации и расшифровке символов файла пачками по N штук.

14.7. Составить программу поиска в файле строки символов, задаваемой с клавиатуры. Программа должна определять количество найденных экземпляров строки и их позиции от начала файла

14.8. Составить программу, удаляющую в файле текст после первой точки (использовать процедуру Truncate). Проверить работу этой программы над собственным текстом.

14.9. Имеется типизированный файл, элементами которого являются отдельные символы. Все цифры этого файла записать во второй файл, а остальные символы – в третий файл.

14.10. Имеется два типизированных файла одинакового размера, элементами которых являются отдельные символы. Выяснить, совпадают ли их элементы. Если нет, то получить номер первого компонента, в котором эти файлы отличаются друг от друга.

14.11. Составить программу, копирующую набор данных с произвольным типом элементов.

14.12. Имеется типизированный файл с целыми числами. Вставить число 100 после первого числа ста. При отсутствии такого числа вставить число 100 в конце файла. Результат записать в другой файл.

15. Управление текстовым режимом

Модуль CRT служит для управления экраном в текстовом режиме, а также для управления клавиатурой и звуковыми сигналами. Модуль содержит библиотеку процедур (подпрограмм) и функций, которые выполняются при их вызове. Модуль подключается в начале раздела описания основной программы оператором Uses CRT;

Пример 15. Составить программу, выводящую на экран в различных текстовых режимах надпись в виде ступеньки с заданным шагом "dx" по горизонтальном направлении, начиная с позиции (1,1). Шаг dx задается в символах.

Решение. Для составления программы решения задачи используем следующие процедуры модуля CRT: TextBackGround, TextColor, TextMode, gotoXY.

```
Program primer15;
Uses CRT;
 var N : word; f, dx, x, y, i, j, xm, ym : byte;
BEGIN
 for i:=0 to 9 do begin {режимы работы монитора}
 if i<4 then N:=i else N:=256+i-4; if i=9 then N:=7;
 textMode(N);
 xm:=lo(WindMax)+1; ym:=hi(WindMax)+1;
 write('xm=',xm, '_ym=',ym, '_N=',N, '_Нажмите Enter'); readln;
 TextBackGround(1); clrscr; TextColor(14); x:=1;
 f:=8; dx:=3; {f-длина фамилии+курсор, dx-приращение отступа}
 for j:=1 to ym-1 do begin y:=j;
 if (xm-x-f)<0 then x:=1; {контроль выхода надписи за экран}
 gotoXY(x, y); write('ФАМИЛИЯ'); x:=x+dx; end;
 Writeln; write('Нажмите Enter'); readln end;
 TextMode(3)
end.
```

Задачи для самостоятельной работы

15.1. Вывести на экран в различных текстовых режимах надпись в виде ступеньки с заданным шагом "dx" по оси "x", в каждой строке "y", начиная с позиции (1, Ym) с направлением вверх, вправо.

15.2. Вывести на экран в различных текстовых режимах надпись в виде ступеньки с заданным шагом " dx<0 " по оси " x ", в каждой стороке " y ", начиная с позиции: (Xm-f, Ym) с направлением вверх, влево.

15.3. Модифицировать программы п. 15.1/15.2, выводя надписи в окнах разного цвета, с различным цветом символов. Размер окна определяется по оси "x" количеством символов в надписи плюс 2, по оси "y" размер равен 3 строчкам. После оператора GotoXY(x, y); следует оператор Window(x, y, xx, yy); и т. д.

15.5. В режимах N=1 и N=3 вывести на экран окна разного цвета с уменьшающимся размером (окно в окне). В первой позиции окон выводить номер окна. Использовать оператор цикла с условием

ограничения размеров наименьшего окна, например: (x2-x1=2) or (y2-y1=0).

15.6. В режимах N=1 и N=3 вывести на экран окна разного цвета с координатами, определяемыми функцией Random с ограничением по размеру экрана, например: x_1:=Random(x_{n-1})+1; x_2:=x_1+Random(x_n-x_1); Окна выводятся в операторе цикла с условием: до нажатия любой клавиши.

15.7. Определить массив первых 196 натуральных четных чисел и распечатать его в виде матрицы 15x15 желтым цветом в центре экрана в рамке синего цвета. При этом выделить красным цветом указанные ниже по одному из вариантов элементы:

а) элементы с последней цифрой "0" или "2",

б) элементы главной диагонали, кратные четырем,

в) элементы пятой колонки и шестой строки,

г) элементы четных столбцов и седьмой строки

д) столбцы матрицы должны быть разного цвета.

е) диагонали матрицы должны быть разного цвета.

15.8. Составить программу, организующую перемещение окна 8x8 по экрану. Движение начинается по нажатию клавиши и заканчивается либо по нажатию клавиши, либо при достижении окном края экрана. Варианты движения:

а) из левого верхнего угла в правый нижний угол. При неточном "попадании" в нижний угол смещать окно по одной из сторон до точной остановки в углу.

б) из левого нижнего угла в правый верхний с условиями по задачи 15.8а.

в) из центра экрана к одной из боковых сторон. При достижении края размер окна по направлению движения должен уменьшаться до минимального.

15.9. Составить программу, организующую пульсирующее окно в центре экрана. Размер окна периодически увеличивается от 2x1 до максимума, а затем уменьшается с сохранением пропорций. Цвет окна в каждом цикле устанавливается случайно.

15.10. Составить программу рисование окон случайным образом.

15.11. С помощью CRT формировать меню. Организовать контролирование и управление процессом его использования.

15.12. Дан текст. Напечатать каждый абзац разными цветами и яркостями.

16. Управление звуком

Свое знакомство с программированием звука на Паскале мы начнем с использования встроенного динамика. Модуль CRT содержит две процедуры, предназначенные для работы с динамиком. Первое из них **Sound(N)** - включение звука с частотой тона N (измеряется в герцах). Динамик генерирует звук до тех пор, пока он не будет отключен вызовом процедуры **NoSound**, не имеющей параметров.

Delay(M) - задержка выполнения программы на M миллисекунд.

Управление звуковым устройством компьютера осуществляется последовательностью операторов:

Sound(F); Delay(N); NoSound;{***NoSound*** выключение звука.}
где F - частота звука в Гц.

Для программирования мелодии удобно применять известную формулу расчета частоты звука:

F=Round(440*Exp(Ln(2)*(No-(10-Nn)/12)));
где No-номер октавы -3<=No <=4; Для основной октавы No=0. Nn-номер ноты 1-До, 2-До#, 3-Ре, и т. д. 12-Си.

При сквозной нумерации нот принять для No<0-Nn<1, для No>0-Nn>12.

Пример 16. Составить программу, генерирующую гамму с нарастающей продолжительностью звучание каждой ноты.

Решение. Для решения задачи сначала определим массив кодов семи нот от «до» до «си». Далее, в цикле осуществляем проигрывания массива нот с помощью звуковых операторов Sound и NoSound в программном режиме.

```
Program primer16;
 Uses crt;
Const M: array[1..7] of integer=(262,294,330,349,440,494); {Ноты}
        T: array[1..7] of integer=(10,11,12,13,14,15,16); {продолжительность звучания}
 Var i: byte;
 Begin clrscr;
   While not KeyPressed do begin
    For i:=1 to 7 do begin
      Sound(M[i]);
      Delay(T[i]);  NoSound;
     end;
   end;
end.
```

Задачи для самостоятельной работы

16.1. Составить программу, имитирующую звук телефонных звонков.

16.2. Составить программу, генерирующую последовательности прямоугольных звуковых импульсов.

16.3. Создать "бегущую строку", со звучанием мелодии. После вывода каждой группы символов вместо процедуры delay(200); ставятся операторы: Sound(F[k]); Delay(N[k]); NoSound; где F, N-массивы частот и длительности звучания нот, определенные в начале программы по приведенной выше зависимости согласно исполняемой мелодии. Параметр "k" наращивается до числа нот в мелодии, затем обнуляется и т. д.

16.4. Создать мелодию с частотой "F" и длительностью звучания нот "N", изменяющихся по выбранным вами формулам. Например: F=1000sin(x)+100, N=random(2)+1, где x=x+0.01 от x=0 до x=Pi.

16.5. Составить программу, генерирующую звук одной из 12 нот в октаве по нажатию цифровых клавиш верхнего ряда (0–"до",1–"до#"…"+" -"си").

16.6. Модернизировать программу задачи 16.3, добавив отображение клавиш фортепиано в виде набора 12 узких окон и высвечивая нужное окно (соответствующее нажимаемой клавише) инверсным цветом в процессе звучания.

16.7. Оформить задачу 16.4 в виде процедуры, отображающей визуально проигрывание мелодии, считанной из файла.

16.8. Создать модуль для проигрывания музыки, играющей в процессе выполнения программы.

16.9. Составить программу для игры в музыкальную викторину. Программа должна последовательно воспроизводить несколько популярных мелодий, играющих угадывание проигранной мелодии.

16.10. Разработать и реализовать на экране поздравительную надпись с мелодиями.

16.11. Озвучить клавиши: цифровые, либо буквы латинского алфавита.

16.12. Воспроизвести до-мажорное трезвучие (до - ми - соль) последовательно, по одной ноте.

17. Динамические текстовые эффекты

Процедуры DelLine;, InsLine; позволяют прокручивать надписи в окне по аналогии с прокруткой текста, не вмещающегося на экране. Процедура Delay(M);-задерживает выполнение программы на M миллисекунд (N и M-тип Word).

Приведем пример операторов, прокручивающих текст в окне:

```
 Window(5,5,30,9); TextBackGround(1); Clrscr; TextColor(14);
repeat delay(500); GotoXY(1, 5);
 write('Всем большой привет !');
 gotoXY(1, 1);
 for k:=1 to 5 do begin {прокрутка вверх}
 delay(300); delLine end
until keyPressed ;
```

Для вывода на экран символа по номеру его кода можно использовать операторы:

C:=chr(N); write(C); где N-номер кода символа (тип Byte), chr-функция преобразования целого типа в символьный. Напомним, что символы, используемые в MS-DOS, имеют коды с номерами от 0 до 255. Некоторые управляющие символы при выводе производят определенные действия, например: #7 - звуковой сигнал, #13 - перевод курсора на новую строку.

В рекламных надписях применяется прием "бегающих" символов. Заставить бегать символ по строке можно с помощью следующих операторов:

```
 y:=5; Xm:=lo(WindMax)+1;
 c:=readKey; {ввод символа с клавиатуры}
repeat
 for i:=1 to Xm-1 do begin {i-номер позиции в строке}
 gotoXY(i, y); write(c); {вывести на экран символ}
 delay(100); gotoXY(i,y); write('_') end {вывести на экран пробел}
until keyPressed;
```

При выводе "бегающих" символов изображение курсора портит картинку. Избавиться от курсора можно, поместив его в "дальний угол" перед задержкой программы, например: GotoXY(1, 1); delay(100); Можно также изменить размер курсора до нуля с помощью следующих операторов:

В разделе описания программы: Uses DOS; Var r: registers; В разделе выполнения программы: r.ah:=1; r.ch:=$20; intr(16,r);.

Пример 17. Составить программу "падающие символы" для цифр от 0 до 9, имеющих кода с номерами от 48 до 57. При нажатии на клавишу с “падающей цифрой”, генерируется другая “падающая цифра” и т. д. до нажатия клавиши ESC.

Решение. В этой программе используем операторы модуля CRT TextBackGround, TextColor, TextMode, gotoXY и регистры модуля DOS: r.ah:=1; r.ch:=$20; intr(16,r).

```
Program primer17;
Uses CRT, DOS;
```

```
 var r: registers;
 x, y, i, xm, ym: byte;
c: array [41..57] of char; cha:char;
BEGIN
 randomize; textMode(3);
 xm:=lo(WindMax)+1; ym:=hi(WindMax)+1;
 textBackGround(blue); ClrScr; TextColor(14);
 r.ah:=1; r.ch:=$20; intr(16,r);
for i:=48 to 57 do begin
 c[i]:=chr(i); write(c[i]) end; {вывод символов с позиции (1, 1)}
repeat gotoXY(10, 10); Write('ПРИГОТОВЬТЕСЬ -нажмите ENTER');
cha:=readKey
until cha=#13; {вывод надписи}
 delay(10000); gotoxy(10, 10); clreol; {стирание надписи}
 GOTOXY(1, YM);
Write('НАЖМИТЕ КЛАВИШУ С ПАДАЮЩИМ СИМВОЛОМ');
 Write(' ДЛЯ ОКОНЧАНИЯ -нажмите ESC');
repeat
 i:=random(10)+48; c[i]:=chr(i); {выбор случайной цифры}
 x:=i-48+1; for y:=2 to ym-1 do begin
 gotoXY(x,y); write(c[i]); delay(10000); {вывод цифры}
 gotoXY(x,y); write(' '); {стирание цифры}
 if keypressed then cha:=readkey;{при нажатии клавиши-считать циф-
ру}
 if cha=#27 then Break;
 if cha=c[i] then Break end {досрочный выход из цикла "for... "}
until cha=#27; {выход при нажатии ESC}
 textMode(3);
end.
```

Рассмотрим последовательность операторов, позволяющих создать на экране "бегущую строку" символов. Введем с клавиатуры надпись с выводом на экран и запомним строку символов в массиве cha.

```
 i:=0; repeat i:=i+1; read(cha[i]) until cha[i]=#13; n:=i-1;
```

Ввод надписи заканчивается нажатием клавиши SpaceBar (последним символом надписи должен быть пробел), затем Enter. Число символов в надписи запоминается в переменной "n", символ с кодом #13 не включается.

Принцип создания "бегущей строки" символов заключается в выводе групп символов строки с позиции от 1 до "n" (первый символ, два первых символа строки и т. д.) и смещением влево позиции вывода первого символа группы. Затем выводятся группы символов

от "n-1" до 1 (строка без первого символа, строка без второго символа и т. д.) без изменения места вывода первого символа группы. После вывода каждой группы символов следует задержка программы, позволяющая "увидеть" группу символов. При выводе каждая новая группа символов затирает предыдущую, причем последний символ (SpaceBar) затирает символы при уменьшении группы выводимых символов. Приведем пример операторов для создания "бегущей строки" символов.

Здесь n1 - номер первого выводимого символа, n2 - номер последнего выводимого символа.

```
repeat
 for i:=1 to 2*n-1 do begin
 if i<=n then begin n1:=1; n2:=i; gotoXY(n-i+1,5) end
 else begin n1:=i-n+1; n2:=n; gotoXY(1,5) end;
 for j:=n1 to n2 do write(cha[j]); {вывод группы символов}
 delay(200); end
until keyPressed ;
```

Задачи для самостоятельной работы

17.1. Создать "бегущую строку". Строку выводить в окне, размером n+1 (n-число символов).

17.2. Создать "бегущую строку". Выводить одновременно две надписи одинаковой длины в разных строках одного окна.

В следующих задачах составить программу, циклически перемещающую вашу фамилию по экрану по одному из следующих направлений:

17.3. по вертикали снизу вверх;

17.4. по горизонтали слева направо;

17.5. по горизонтали справа налево;

17.6. по горизонтали с отражением от границ;

17.7. по диагоналям с отражением от границ;

17.8. по краю экрана по часовой стрелке;

причем при смене направления движения должен генерироваться короткий звуковой импульс, а в начале нового цикла должен случайно меняться цвет надписи.

17.9. Составить программу перемещения по экрану вводимой предварительно с клавиатуры текстовой строки. Перемещение осуществляется по восьми направлениям, причем каждому направлению должен соответствовать определенный цвет вывода этой строки.

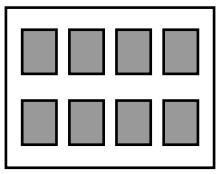
17.10. Составить программу, перемещающую строку по горизонтали слева направо. При достижении края экрана строка "распа-

дается" на отдельные символы, движущиеся по экрану с отражением от его границ (при "ударе" необходимо сгенерировать для массива символов случайные начальные скорости).

17.11. Составить программу, перемещающую строку по вертикали снизу вверх. При достижении края экрана перемещайте обратно.

17.12. Вывести на экран в различных текстовых режимах надпись в виде ступеньки с заданным шагом "dx" по оси "x", в каждой строке "y", начиная с позиции (1, Ym) с направлением вверх, вправо.

18. Типовые задачи для текстового режима



Используя операторы MOD и DIV можно создать несколько рядов одинаковых окон (карточек) на экране в одном цикле:

```
Randomize;
 N:=Nx*Ny; Nc:=3; {Nc-число цветов в колонке}
for i:=0 to N-1 do begin
 x1:=1+hx*(i MOD Nx); x2:=x1+hx-1; {чередование столбцов}
 y1:=1+hy*(i DIV Nx); y2:=y1+hy-1; {наращивание строк}
 window(x1, y1, x2, y2);
 j:=1+(i mod Nx); {номер столбца с карточкой}
 c[i+1]:=1+(j-1)*Nc+random(Nc); {номер цвета карточки}
 TextColor(c[i+1]);
 for k:=1 to hx*hy do write('_'); {заполнение окна символами}
end;
```

Здесь каждое окно (карточка) заполняется символами разного цвета, причем номер цвета выбирается случайно из трех значений, различных для каждой колонки. При заполнении последним символом происходит прокрутка окна.

Пример 18. Составить программу, выводящую на центр экрана "всплывающий" надпись.

Решение. При составлении программы используем процедуры модуля CRT: Window, TextBackGround, TextColor. Для передвижения и задержки строки использован оператор цикла и delay. Чтобы вывести надпись в центр экрана, выбран параметр экрана в процедуре Window i>4.

```
Program primer18;
Uses crt;
Var x1,y1:integer;
```

```
   xm,ym:integer;
   i,a:integer;
 begin
 x1:=xm div 2; y1:=ym div 2; {центр окна}
 a:=15; {2*a-размер окна}
for i:=0 to a do begin Window(x1-i, y1-i, x1+i, y1+i);
 TextBackGround(5); ClrScr;
 if i-4>0 then begin GotoXY(i*2, i); {вывод надписи}
 TextColor(14); Write('ФАМИЛИЯ') end; {в координатах окна}
 delay(10000);
end; readln;
end.
```

Задачи для самостоятельной работы

18.1. Построить вертикальную гистограмму функции Y=420 div X+X2 в диапазоне X=1..55.

18.2. Построить горизонтальную гистограмму функции Y=200 div X в диапазоне X=1..20.

18.3. Вывести на экран окна разного цвета с уменьшающимся размером (окно в окне, не менее пяти штук). В первой позиции окон выводить их номера.

18.4. Разделить экран на два окна разного цвета так, чтобы в каждом окне можно было вводить с клавиатуры текст. Переход между окнами – клавишей Tab, окончание ввода – Esc.

18.5. Составить программу, "распахивающую" на экране несколько рядов окон (например, 2 ряда по 4 окна в каждом) разного цвета с надписями. Варианты распахивания окон -

по горизонтали, по вертикали, из центра. Усложненный вариант задачи – вводить число рядов и окон в ряду с клавиатуры.

18.6. Закодировать в файле элемент орнамента 4x4,-как цвет соответствующих знакопозиций. В программе считать данные из файла и заполнить орнаментом экран в режиме 50x80.

18.7. Составить процедуру создания текстового окна, окаймленного рамкой из псевдографических символов. В параметры процедуры ввести координаты левого верхнего угла, размеры и цвет окна, а также цвет рамки.

18.8. Создать на экране (или в окне) "бегущую строку" с вашей фамилией, которая появляется с левого края посимвольно и также плавно "уходит" за правый край. Усложненный вариант задачи – уходящие за правый край символы строки тут же появляются слева, то есть одновременно видны все символы строки.

18.9. С помощью приведенной таблицы символов с ASCII кодами создать четырехугольник с заданным размером.

Символ	ASCII код	Описание
┌	218	Левый верхний угол
┐	191	Правый верхний угол
└	192	Левый нижний угол
┘	217	Правый нижний угол
│	179	Вертикальная линия
─	196	Горизонтальная линия

18.10. Вывести на экран в различных текстовых режимах надпись в виде ступеньки с заданным шагом "dx<0" по оси "x", в каждой строке "y", начиная с позиции: (Xm-f, Ym) с направлением вверх, влево.

18.11. В режимах N=1 и N=3 вывести на экран окна разного цвета с уменьшающимся размером (окно в окне). В первой позиции окон выводить номер окна. Использовать оператор цикла с условием ограничения размеров наименьшего окна, например: (x2-x1=2) or (y2-y1=0).

18.12. В режимах N=1 и N=3 вывести на экран окна разного цвета с координатами, определяемыми функцией Random с ограничением по размеру экрана, например: x1:=Random(Xm-1)+1; x2:=x1+Random(Xm-x1); Окна выводятся в операторе цикла с условием: до нажатия любой клавиши.

19. Построение заполненных фигур (модуль Graph)

Ряд графических процедур выполняет построение заполненных фигур - фигур с замкнутым контуром, автоматически заполняемых сразу после построения. По умолчанию заполнение производится сплошным белым цветом. Цвет и стиль (орнамент) заполнения можно устанавливать из стандартного набора BGI или определять самим.

SetFillStyle(P, N);-процедура установки орнамента P=0,1,...,12 и цвета с номером "N" для заполняемых фигур.

P =0-сплошное заполнение цветом фона, при этом значение "N" игнорируется,

P =1-сплошное заполнение цветом с номером "N",

P =2...11-стандартный набор орнаментов BGI,

P =12-орнамент и цвет определяет пользователь.

Пример 19. Составить программу, выводящую два прямоугольника с орнаментами пользователя (bukva_Y и Red_50), а затем демонстрирующую набор стандартных орнаментов на передней грани параллелепипедов.

Решение. Для создание орнаментов пользователя используем тип данных FillPatternType. После чего процедурой SetFillPattern закрашиваем прямоугольник созданным орнаментом. Стандартные орнаменты выводятся процедурой SetFillStyle.

```
Program primer19;
uses Crt, Graph;
Const bukva_Y: FillPatternType=($81, $C3, $66, $3C, $18, $18, $18,
$18);
 Red_50: FillPatternType=($AA, $55, $AA, $55, $AA, $55, $AA, $55);
var i, x1, y1, x2, y2, Gd, Gm : integer;
Begin Gd:=Detect; InitGraph(Gd, Gm, 'D:\BP\BGI');
 SetFillPattern(Red_50, Red); {орнамент-50% красных пикселов}
 Bar(250, 10, 350, 110);
 SetFillPattern(bukva_Y, Blue); {орнамент-синяя буква "Y"}
 Bar(340, 30, 440, 130);
 {стандартный набор из 12 орнаментов BGI выводим цветом с номером "11"}
 for i:=0 to 11 do begin SetFillStyle(i, 11);
 if i<6 then begin x1:=90*i; y1:=150 end
 else begin x1:=90*(i-6); y1:=270 end;
 x2:=x1+70; y2:=y1+80;
 Bar3d(x1, y1, x2, y2, 10, TopOn) end;
Readkey; CloseGraph
End.
```

Заполняя не черный экран орнаментом Red_50, можно получить новые цвета фона.

Выбранным из стандартных или определенным орнаментом можно заполнить любую замкнутую область с границей цвета "N" оператором

```
FloodFill(X, Y, N);
```

Заполнение начинает производиться из точки X, Y и ограничивается при достижении границы цвета с номером "N". Например: Rectangle(x1, y1, x2, y2); FloodFill((x1+x2) div 2, (y1+y2) div 2, Red); Если область не замкнута или цвет границы не равен "N", то заполнение "разольется" по экрану.

Задачи для самостоятельной работы

С использованием оператора цикла нарисовать на экране 12 одинаковых заполненных различными стандартными орнаментами фигур (в 3 ряда по 4 фигуры). По нажатию клавиши должен меняться вариант фигуры:

19.1. Прямоугольник.
19.2. Треугольник.
19.3. Параллелепипед.
19.4. Ромб.
19.5. Эллипс.
19.6. Круг.
19.7. Трапеция.

Выполнить эти задания с использованием процедур рисования фигур без автоматического заполнения, а затем заполнить их.

19.8. Нарисовать 5 вложенных прямоугольников с увеличением размера на 20 пикселов в каждом направлении. Стиль заполнения для каждой фигуры определить буквами Вашего имени.

19.9. Нарисовать 5 со-осьных кругов разного цвета с уменьшением радиуса на 10 пикселов. Стиль заполнения для каждого круга определить его номером, т. е. цифрами 1, 2, 3, 4, 5.

19.10. Определить три орнамента заполнения: "снежинка", "иголки" с наклоном влево и вправо. Нарисовать из треугольников пять елок, и заполнить их орнаментом "иголки". Заполнить экран орнаментом "снежинка".

19.11. Создать элемент орнамента с единицами (4*4) в центре. Нарисовать на экране картину "ночной город", используя фрагменты "звездного неба" и пять прямоугольников, заполненных орнаментом 4*4 разного цвета.

19.12. Используя исходные орнаменты из одной линии составить с использованием логических операций орнаменты цифр: 1, 6, 7 и букв: B, F, E.

20. Работа с линиями

В графическом режиме курсор невидим, его положение можно определить функциями, возвращающими значения координат:

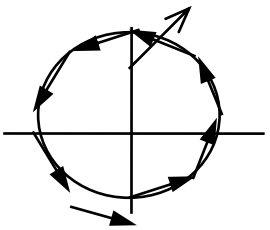
GetX;-по оси "X", GetY;-по оси "Y".

Следующие процедуры перемещают курсор без рисования:
MoveTo(x, y);-переместить курсор в точку с координатами (x, y),
MoveRel(dx,dy);-сместить курсор на расстояние dx, dy от текущего положения.

Для построения многоугольников и ломаных линий удобно использовать процедуры:

LineTo(x, y);-провести отрезок прямой линии от текущего положения курсора до точки с координатами X, Y.

LineRel(dX, dY);-провести отрезок прямой линии от текущего положения курсора до точки, смещенной на расстояние dX, dY по соответствующим осям.



В отличие от процедуры Line(x1, y1, x2, y2); процедуры LineTo(x, y); и LineRel(dX, dY); при своем исполнении перемещают текущий указатель.

Пример операторов для построения восьмиугольника:

```
R:=100; {расстояние от вершин до центра xc, yc}
xc:=GetMaxX div 2; yc:=GetMaxY div 2;
for i:=1 to 8 do begin alfa:=i*pi/4; {значение угла в рад.}
x[i]:=xc+round(R*cos(alfa)); {координаты вершин}
y[i]:=yc+round(R * sin(alfa))
end;
MoveTo(x[8], y[8]); {исходная позиция для рисования}
for i:=1 to 8 do LineTo(x[i], y[i]); {рисование линий}
```

Для отрезков прямых линий и процедур с их использованием можно задать режимы построения прямых линий оператором:

SetWriteMode(N);

N=0-замещение линией изображения на экране (режим CopyPut) используется по умолчанию,

N=1-изображение комбинируется (режим XorPut). Работа функции состоит в изменении согласно логической операции "исключающее ИЛИ" исходного значения цвета пиксела (числа "1" или "0"). Логическая функция Xor, примененная к одной переменной дважды, восстанавливает ее исходное значение: (J xor I) xor I=J. Следовательно, при повторном проведении линии в режиме XorPut изображение этой линии уничтожается, а цвет пикселов экрана становится исходным. На этом правиле основаны некоторые программы построения движущихся изображений.

Пример 20. Составить программу, рисующую движущийся прямоугольник.

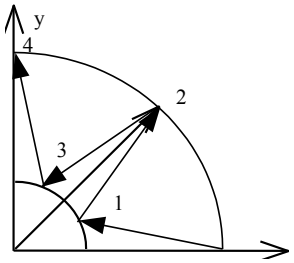
Решение. Для рисования прямоугольника используем процедуру Rectangle. Чтобы привести в движение прямоугольника, образуем цикл.

```
Program primer20;
uses Graph;
Var Gd,Gm:integer;
    a, b, x1, y1: word;
    i, N: integer;
Begin
```

```
Gd:=Detect; InitGraph(Gd, Gm, '_');
SetWriteMode(1);
 a:=100; b:=50; {стороны прямоугольника}
 x1:=0; y1:=GetMaxY div 2;
 N:=GetMaxX-x1-a; {N-число перемещений}
for i:=0 to N do begin
 Rectangle(x1+i, y1, x1+a+i, y1+b); {рисование прямоугольника}
 delay(10); Rectangle(x1+i,y1,x1+a+i,y1+b);{стирание прямоугольни-
ка}
end;
end.
```

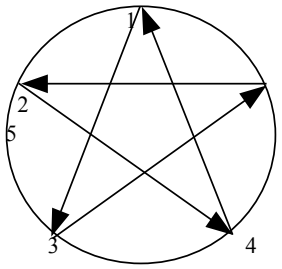
Задачи для самостоятельной работы

20.1. Составить процедуру рисования N-угольной звезды. В параметры процедуры включить число лучей звезды, радиусы вписанной и описанной окружностей, цвет линий и координаты центра звезды. Перемещать две звезды разного цвета в пределах экрана случайным образом.



20.2. Составить процедуру рисования N-угольной звезды, где N=3, 5, 7,... Операторы рисования имеют вид:

```
M:=N div 2; Moveto(x[1],y[1]);
For i:=1 to N do begin j:=(M*i) mod N+1;
LineTo(x[j],y[j]) end;
```



20.3. Вывести на экран толстые горизонтальные линии с двоичным представлением:
1111000001100000, 1111000011110000, 0111101111011110,
1100110011001100, 1001100110011001, 1111100011111000.

20.4. Вывести на экран толстые вертикальные линии с двоичным представлением:
0101010101010101, 1100011000110001, 1111110011111100,
0111011101110111, 1110001110001110, 1111000000001111.

20.5. Вывести на экран линии разных форм, заданных параметром PN =PN-1+2 * N,
где P0=1; N=1,.. , 150. Линии располагать вертикально.

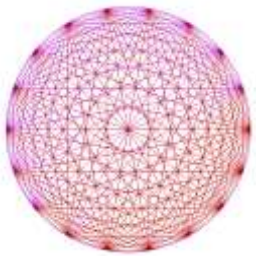
20.6. Вывода на экран линий разных форм, заданных параметром PN =PN-1+2N,
где P0=1; N=1,.. , 15. Линии располагать горизонтально.

20.7. Нарисовать расходящийся из центра экрана по спирали ромбический лабиринт шириной 6 пикселов из отрезков наклонных прямых. Очищать экран и менять толщину и форму линии F=0,.. , 3.

20.8. Нарисовать расходящийся из центра экрана по спирали прямоугольный лабиринт шириной 5 пикселов из отрезков вертикальных и горизонтальных прямых. Очищать экран и менять толщину и форму линии.

20.9. Создать эффект "бегущих огней" перемещением на один пиксел набора из трех касающихся толстых пунктирных линий (перерисовка в режиме XorPut). Крайние линии стиля "P", средняя - стиля "not P".

20.10. Создать штриховые стили "P1", "P2" и рассчитать с использованием всех логические операций приведенные выше стили P3. Вывести на экран исходные и расчетные линии.

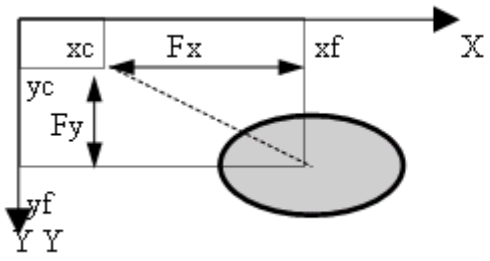


20.11. Начертить узор, показанной на рисунке. (См. демонстрирующий файл … BP \ Examle \ DOS \ BGI \ BGIDEMO.PAS)

20.12. Построить треугольник по заданным вершинам. Точки экрана, являющиеся вершинами треугольника, указываются с клавиатуры.

21. Перемещение фигуры

Если фигуру перемещать, не вращая относительно своего "центра", то получим плоскопараллельное движение тела (любой отрезок прямой на фигуре остается параллельным самому себе). За "центр" фигуры может быть принята любая точка жестко связанная с фигурой. Обычно "центр" фигуры (xf, yf) перемещают относительно центра узора (xc, yc) по определенному закону:




```
xf = xc + Fx("параметры"),
yf = yc + Fy("параметры"),
```

где Fx, Fy – функции от параметров.

Приведем пример задания закона движения "центра" фигуры относительно центра узора:

```
for i:=1 to Nc do begin
  alfa:=2 * pi * i/Nc; {угол поворота "центра" фигуры}
 Lx:=FLx(i); Ly:=FLy(i); {функции расстояний}
```

```
 R:=FR(i); S:=FS(i); {функции радиуса и цвета окружности}
 xf:=xc+round(Lx * cos(alfa)); {координаты "центра" фигуры}
 yf:=yc+round(Ly * sin(alfa));
 SetColor(S); Circle(xf, yf, R) end;
```

В этом фрагменте Nc-раз рисуется окружность с центром, поворачивающимся на угол alfa вокруг центра узора. Расстояние от центра i-й окружности до центра узора задается функциями Flx(i), Fly(i), радиус окружности - функцией FR(i), цвет - функцией FS(i). Подбором этих функций и числа окружностей Nc можно добиться разнообразных декоративных эффектов. Вместо окружностей можно строить любые фигуры, используя процедуры их рисования с заданием "центра" фигуры и других параметров в системе координат экрана.

В общем случае фигура может перемещаться, вращаясь относительно своего "центра" и деформироваться. При этом параметры процедуры рисования фигуры должны включать все координаты точек, которые соединяются линиями. Координаты i-ой точки фигуры определяются по формулам:

xxi =xf+Kxi * ((xi-xf) * cos(A)-(yi-yf) * sin(A)),

yyi =yf+Kyi * ((yi-yf) * cos(A)+(xi-xf) * sin(A)),

где A-угол поворота фигуры относительно своего "центра", отсчитываемый в левой системе координат экрана по часовой стрелке относительно оси X,

xi, yi-исходные координаты i -ой точки фигуры,

xxi, yyi-новые координаты i -ой точки фигуры,

Kxi, Kyi-коэффициенты масштабирования координат i -ой точки по осям X и Y.

Приведем пример задания закона движения линии относительно своего "центра":

```
for j:=1 to Nf do begin
  A:=2 * pi * j/Nf; {угол поворота линии вокруг своего "центра"}
       Kx1:=FKx1(j); Ky1:=FKy1(j); Kx2:=FKx2(j); Ky2:=FKy2(j);
{координаты 1-ой точки фигуры}
       xx1:=xf+round(Kx1 * ((x1-xf)*cos(A)-(y1-yf)*sin(A)));
       yy1:=yf+round(Ky1 * ((y1-yf)*cos(A)+(x1-xf)*sin(A)));
          {координаты 2-ой точки фигуры}
       xx2:=xf+round(Kx2* ((x2-xf)*cos(A)-(y2-yf)*sin(A)));
       yy2:=yf+round(Ky2* ((y2-yf)*cos(A)+(x2-xf)*sin(A)));
 SetColor(14); line(xx1, yy1, xx2, yy2); delay(100);
end;
```

Здесь x1, y1, x2, y2 - исходные координаты точек фигуры, xx1, yy1, xx2, yy2-координаты 1-ой и 2-ой точек фигуры на i-ом шаге рисования.

В этом фрагменте многократно (Nf-раз) рисуется линия, вращающаяся на угол "A” относительно своего центра xf, yf. Фигура может искажаться (деформироваться), если не соблюдаются равенства: Fkx1(j)=Fky1(j)=Fkx2(j)=Fky2(j)=K=1.

Если центр узора перемещается, то изменение его координат необходимо задать во внешнем цикле.

Пример 21. Написать процедуру, которая рисует на экране кораблик. В качестве параметров процедуру используется координаты базовой точки и цвет, которым следует рисовать. С помощью этой процедуру, напишите программу, которая выводит на экран движущийся кораблик.

Решение. Сначала создадим процедуру с параметром выводящим на экран кораблик. Далее изменяя параметры этой процедуры можно добиться передвижению кораблика.

```
Program primer21;
Uses crt,graph;
Var GraphDriver, GraphMode : Integer;
    X,Y : Integer;
    Color, Bkcolor: Word;
Procedure Korabl(x,y:integer; Color: Word);
    const dx=5; dy=5;
    var oldcolor: Word;
 begin
  oldcolor:=GetColor; {сохранение текущего цвета}
  Setcolor(color); {установить новый цвет}
  {Корпус}
  MoveTo(x,y);
  LineTo(x,y-2+dy);
  LineTo(x+10*dx,y-2*dy);
  LineTo(x+11*dx,y-3*dy);
  LineTo(x+17*dx,y-3*dy);
  LineTo(x+14*dx,y);
  LineTo(x,y);
  {Надстройка}
  MoveTo(x+3*dx,y-2*dy);
  LineTo(x+4*dx,y-3*dy);
  LineTo(x+4*dx,y-4*dy);
  LineTo(x+13*dx,y-4*dy);
  LineTo(x+13*dx,y-3*dy);
  Line(x+5*dx,y-3*dy,x+9*dx,y-3*dy);
  Setcolor(oldcolor); {Востановит текуший цвет}
end;
```

```
Begin
 GraphDriver:=Detect;
 InitGraph(GraphDriver, GraphMode, 'C:\LANGUAGE\BP\Bgi');
 x:=10; y:=200;
 color:=LightGray;
 SetBkcolor(Blue);
 Bkcolor:=GetBkcolor;
 Repeat
  Korabl(x,y,color); {Нарисовать корабль}
  Delay(100);
  Korabl(x,y,Bkcolor); {Стереть корабль}
  PutPixel(x,y,color); {След от корабя}
   x:=x+2;
 Until x>500;
 readln;
 CloseGraph;
end.
```

Задачи для самостоятельной работы

21.1. Нарисовать узор из 30-ти эллипсов с центром узора в середине экрана. Радиусы каждого эллипса (Rx, Ry) и расстояние от "центра" эллипсов до центра узора увеличивать на один пиксел.

21.2. Нарисовать узор из 21-ти прямоугольников с центром узора в середине экрана. Длины сторон прямоугольников и расстояние от центра узора до "центра" фигуры (например, левого верхнего угла прямоугольника) уменьшать на один пиксел. Для рисования прямоугольника использовать оператор: Rectangle(xf, yf, xf+a-i, yf+b-i); где a и b-стороны прямоугольника, i-параметр цикла вращения вокруг центра узора.

21.3. Нарисовать узор из отрезка прямой линии, вращающегося вокруг своего "центра" (N-раз) за один оборот вокруг центра узора.

21.4. Нарисовать узор из отрезка прямой линии, вращающегося вокруг своего "центра" (N-раз) за один полупериод движения по синусоиде.

yf =yc+Af * sin((xf-xc)/100), где Xf =xc+10 * Pi * j; j=1, 2,.. , 10,
Af-задать исходя из размеров экрана.

21.5. Составить процедуру рисования самолета (задавать координаты узлов, которые соединяются прямыми линиями). Нарисовать самолет, движущийся вокруг центра узора по эллиптической траектории (Lx<>Ly). Самолет должен поворачиваться вокруг своего "центра" в соответствии с траекторией (cos(A)=Fx/L, sin(A)=Fy/L, где L=/Fx2+Fy2).

21.6. Составить процедуру рисования машины (задавать координаты узлов, которые соединяются прямыми линиями). Нарисовать машину, движущуюся синусоиде. Машина должна поворачиваться вокруг своего "центра" в соответствии с траекторией (A=arctg(d(yf)/d(xf)), для данного случая: A:=arctan(Af/100*cos((xf-xc)/100);

Примечание к п. 21.5, 21.6: Фигура перерисовывается в режиме SetWriteMode(1);

21.7. Составить программу для управления размерами окружности и ее положением на экране. Исходная окружность имеет центр в точке (100,100) и радиус r=20.

Управление выполняется клавишами:

«>» - увеличивает радиус окружности на 5 точек;

«<» - уменьшает радиус окружности на 5 точек;

клавиши управления курсором вызывают перемещение окружности в соответствующем направлении;

«ввод» завершает работу программы.

21.8. Составить программу для управления размерами прямоугольника и его положением на экране. Левый верхний уголь исходного прямоугольника расположен в точке (50,50), правый нижний в точке (100,100).

Управление выполняется клавишами:

«>» - увеличивает ширину прямоугольника на 5 точек;

«<» - уменьшает ширину прямоугольника на 5 точек;

«+» - увеличивает высоту прямоугольника на 5 точек;

«-» - уменьшает высоту прямоугольника на 5 точек;

клавиши управления курсором вызывают перемещение прямоугольника в соответствующем направлении;

«ввод» завершает работу программы.

21.9. Изобразить на экране прямую, вращающуюся в плоскости экрана вокруг одной из своих точек.

21.10. Задачу 21.9 усложните следующими дополнительными требованиями, чтобы цвет прямой изменялся при переходе от предыдущего положения к следующему.

21.11. Изобразить на экране отрезок, вращающийся в плоскости экрана вокруг

а) своей середины

б) своего конца

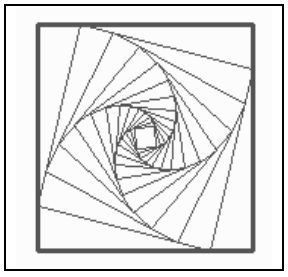
в) точки, делящей отрезок в отношении 1:3.

21.12. Усложните условие задачи 21.9; в плоскости экрана должны вращаться, каждая вокруг своей точки, две прямые.

22. Масштабирование фигуры

Рассмотрим случай уменьшения размеров фигуры делением ее сторон.

```
N:=7; R:=170; xc:=GetMaxX div 2; yc:=GetMaxY div 2;
 for i:=1 to N do begin alfa:=i*2. *pi/N;
    x[i]:=xc+round(R*cos(alfa));  {координаты вершин}
    y[i]:=yc+round(R*sin(alfa))  {исходного N-угольника}
 end;
      MoveTo(x[N], y[N]);
 for i:=1 to N do LineTo(x[i], y[i]);    {рисуем N-угольник}
      ch:=ReadKey;        {нажать клавишу}
Repeat {найдем середины сторон многоугольника и запишем их в
массивы новых координат многоугольника}
       x1:=x[1]; y1:=y[1];
 for i:=1 to N-1 do begin x[i]:=(x[i]+x[i+1]) div 2;
       y[i]:=(y[i]+y[i+1]) div 2 end;
       x[N]:=(x[N]+x1) div 2;
       y[N]:=(y[N]+y1) div 2;
                  {строим многоугольник по новым точкам}
     MoveTo(x[N], y[N]);
 for i:=1 to N do LineTo(x[i], y[i]);
       ch:=ReadKey;      {нажать клавишу}
Until ch=#27;
```



При нажатии клавиши внутрь фигуры будет "убегать" ее уменьшенная копия до нажатия клавиши Esc. Стороны можно делить не пополам, а в каком-либо соотношении. Для стирания фигуры необходимо перерисовать ее в режиме XorPut.

Масштабирование фигур можно проводить используя зависимости, приведенные выше для вращения фигуры относительно своего "центра", изменяя Kx и Ky, при постоянных параметрах xf, yf, A.

Теперь посмотрим **симметричное отображение фигуры.** Пусть xz-ось зеркальной симметрии, тогда координата "yzi" i-ой точки фигуры в системе координат (xz, yz) определяется по формуле:

yzi=(yi-yf)*cos(A)-(xi-xf)*sin(A),

где xf, yf - координаты любой точки на оси зеркальной симметрии,
xi, yi-координаты отображаемой точки в системе координат экрана,
A - угол наклона оси симметрии относительно оси "x" по часовой стрелке.

Тогда координаты отраженной точки в системе координат (X0Y) определяются по формулам:

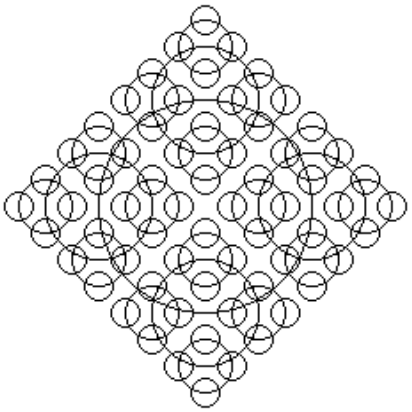
```
xoi=xi+2*yzi*sin(A);
yoi=yi-2*yzi*cos(A);
```

Приведем пример операторов для расчета координат xo[i], yo[i] точек фигуры зеркально отображенной относительно оси, наклоненной под углом "A" и проходящей через точку с координатами xf, yf.

```
for i:=1 to N do begin
        yzi:=round((y[i]-yf)*cos(A)-(x[i]-xf)*sin(A));
        xo[i]:=x[i]+2*round(yzi*sin(A));
        yo[i]:=y[i]-2*round(yzi*cos(A))
 end;
```

Пример 22. Составить программу, которая выводит на экран приведенный ниже узор.

Решение. Узоры создаются с помощью процедуры, параметрами которого является координаты центра и радиуса окружности и порядка узора.



```
Program Primer22;
Uses crt,graph;
Var GraphDriver, GraphMode : Integer;
      X,Y : Integer;
 {Рисует элемент узора}
 Procedure Elem(x,y,r,p: integer);
  {x,y,r - координаты центра и радиуса основного элемента узора. P
  - порядок узора}
   Begin
   if p>=0 then
    begin
     circle(x,y,r); delay(100);
     Elem(x+r,y,round(r/2),p-1);
     Elem(x,y-r,round(r/2),p-1);
     Elem(x-r,y,round(r/2),p-1);
     Elem(x,y+r,round(r/2),p-1);
    end;
   end;
Begin
 GraphDriver:=Detect;
 InitGraph(GraphDriver, GraphMode, 'C:\LANGUAGE\BP\Bgi');
 Elem(320,240,60,3); {Рисуем узор третьего порядка}
```

```
 readln;
CloseGraph;
end.
```

Задачи для самостоятельной работы

22.1. Создать уменьшающийся треугольник (деление сторон 9:1) и пульсирующий треугольник.

22.2. Создать уменьшающийся прямоугольник (деление сторон 3:1) и пульсирующий прямоугольник.

Примечание к п. 22.1 и п. 22.2: пульсация фигуры достигается перерисовкой ее в режиме XorPut, при многократном увеличении и последующем уменьшении коэффициента масштабирования, например, по закону: K=K+i*0. 02. Где i-параметр цикла.

22.3. Нарисовать елку с основанием в центре экрана и получить три зеркальных отображения относительно осей, проходящих через центр экрана под углом +450, 00, -450 к оси "x".

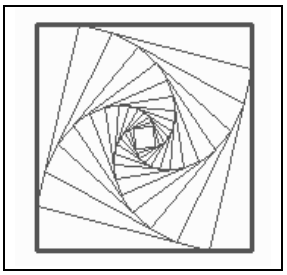
22.4. Выполнить п. 22.3 для подсвечника со свечой.

22.5. Пусть две точки заданы своими координатами (x_1,y_1) и (x_2,y_2). Прямая, проходящая через эти две точки, может быть описаны следующими параметрическими уравнениями:

$x=x_1+(x_2-x_1)t$, $y=y_1+(y_2-y_1)t$,

При $0<t<1$ точка (x,y) лежит внутри отрезка и делит его в отношении $t/(1-t)$; при $t=0$ достигается конец отрезка (x_1,y_1), при $t=1$ – конец (x_2,y_2). При $t>1$ точка (x,y) лежит на прямой вне отрезке с той же стороны от (x_1,y_1) и (x_2,y_2); при $t<0$ – противоположной стороны.

Даны натуральные числа x_1,y_1,x_2,y_2, действительное число μ $(0\le\mu<1)$. Построить отрезок с координатами концов (x_1,y_1), (x_2,y_2) и точку, делящую отрезок в отношении $\mu/(1-\mu)$.

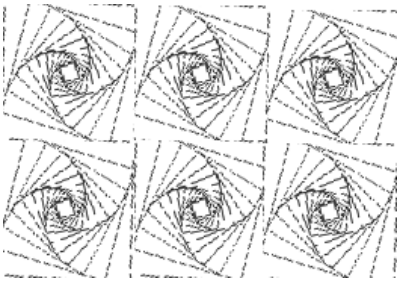


22.6. Начертить узор, показанный на рисунке. Узор образован 20 вложенными квадратами. Стороны первого квадрата параллельны осям координат экрана и равны 60. Вершины каждого последующего квадрата – это точки на сторонах предыдущего квадрата, делящие эти стороны в отношении $\mu=0.008$ (см. предыдущую задачу).

22.7. Начертить узор, повторяющий узор, описанный в предыдущей задаче, на составлении из треугольников.

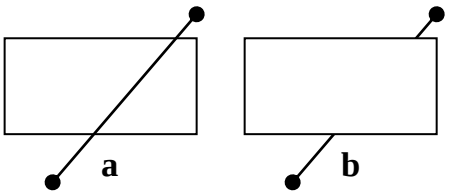
22.8. Начертить узор, повторяющий узор, описанный в предыдущей задаче, на составлении из пятиугольников.

22.9. Начертить узор, повторяющий узор, описанный в предыдущей задаче, на составлении из шестиугольников.

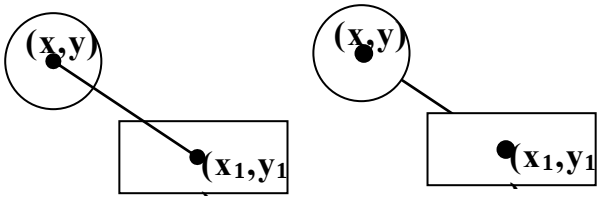


22.10. Построить узор, показанный на рисунке используя алгоритм, описанный в задаче 22.5.

22.11. Прямоугольник задан координатами левого верхнего и правого нижнего угла, отрезок – координатами концов. Предполагается, что отрезок пересекает противоположные стороны прямоугольника так, как показано на рисунке **a**. Построить фигуру, изображенную на рисунке **b**.



22.12. Даны натуральные числа x, y, r, x_1, y_1, h, ω. Построить окружность радиуса r с центром в точке (x,y), прямоугольник с центром в точке (x_1, y_1), высотой h и шириной ω, а так же отрезок, соединяющий центр окружности с центром прямоугольника. (см. рис.)



23. Построение изображений

Графические возможности Турбо-Паскаля позволяют построить уникальные изображения различных фигур и узоров. Для этого обычно используются различные процедуры графического режима. Графики функций строятся обычно в декартовой системе координат. Но система координат экрана отличается от декартовой системы координат, так как начало координат в графическом режиме находятся в левом верхнем углу, ось абсцисс направлена слева направо, а ось ординат – сверху вниз. Поэтому это обстоятельство надо учесть в каждой программе.

Для изменения графических координат экрана в Турбо-Паскале предусмотрено задание графического окна процедурой:

SetViewPort(xG1, yG1, xG2, yG2, Cl);

где (xG1, yG1), (xG2, yG2) - координаты левого верхнего и правого нижнего вершин прямоугольника, образующего графическое окно (тип Integer).

Cl - признак рисования за границами окна (тип Boolean) задается в модуле Graph константами: ClipOn =True - рисование только в пределах окна,

ClipOff=False - рисование в пределах экрана.

После задания графического окна изменяется начало системы координат (рисунок на экране сохраняется) и можно рисовать кривые с отрицательными значениями координат точек, например, установив начало координат в центре экрана:

xG2:=GetMaxX; yG2:=GetMaxY; xG1:=xG2 div 2; yG1:=yG2 div 2;

и задав Cl:=ClipOff; Направление осей при этом не меняется, график не масштабируется, а процедура SetBkColor(N); изменит цвет всего экрана. Графическое окно можно очистить процедурой

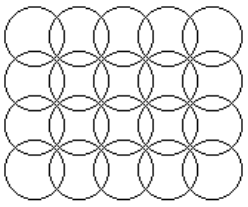
ClearViewPort;

При этом восстанавливается система координат экрана, а изображение в области прямоугольника с координатами вершин (xG1, yG1), (xG2, yG2) затирается цветом фона.

Построение графиков на экране монитора имеет свои особенности, связанные с пиксельным изображением и системой координат экрана. Поскольку для некоторых режимов работы монитора отношение ширины к высоте экрана не равно (GetMaxX+1)/(GetMaxY+1), то при построении по точкам вместо окружности получается эллипс. Для рисования правильных геометрических фигур по точкам необходимо подключить процедуру:

GetAspectRatio(xx, yy);

возвращающую значения xx, yy - параметры (тип Word), определяющие коэффициент сжатия изображения k =xx/yy. При построении графиков или рисовании фигур по точкам значения координат "y" необходимо умножить на "k". Отметим, что для монитора VGA в режиме Gm=2 значение k=1.



Пример 23. Составить программу, которая выводит вышеуказанный рисунок на экран.

Решение. Изображение создается с использованием стандартного оператора Circle в цикле For.

```
Program primer23;
Uses crt,graph;
Var GraphDriver, GraphMode : Integer;
              y,x:integer; r:integer;
              l:integer; i,j:integer;
Begin
 GraphDriver:=Detect;
InitGraph(GraphDriver, GraphMode, 'C:\LANGUAGE\BP\Bgi');
 y:=100; r:=20; l:=30;
```

```
 for i:=1 to 4 do
begin   x:=100;
for j:=1 to 5 do
  begin  Circle(x,y,r); x:=x+l; end;
  y:=y+l;
end;
readln; CloseGraph;
end.
```

Задачи для самостоятельной работы

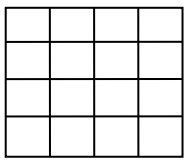
Задать графическое окно с началом в центре экрана. Для драйвера VGA в режимах 0, 1, 2 при k=1 и k =xx/yy выполнить следующие действия.

23.1. Процедурой Circle(x, y, r); нарисовать окружность радиусом 50 с центром в начале координат. Процедурой Line(x1, y1, x2, y2); нарисовать квадрат, описанный вокруг окружности. Очистить графическое окно.

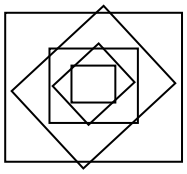
23.2. Процедурой Rectangle(x1, y1, x2, y2); нарисовать квадрат со стороной 100 с центром в начале координат. Процедурой Line(xi,yi,xj,yj); нарисовать окружность (в виде многоугольника), вписанную в квадрат. Очистить графическое окно.

Примечание. Вывести надпись - номер режима и значение k. При k=1 и режимах 0, 1 окружность не будет вписанной в квадрат.

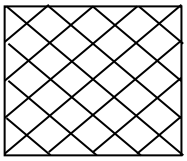
Построить изображение, приведенное на рисунке.



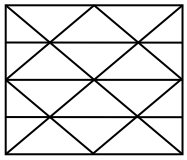
23.3



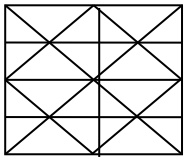
23.4



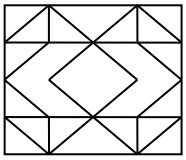
23.5



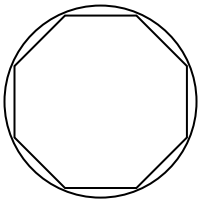
23.6



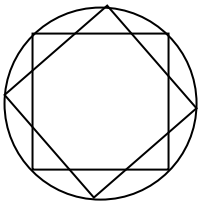
23.7



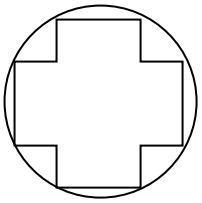
23.8



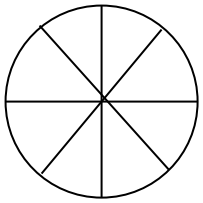
23.9



23.10



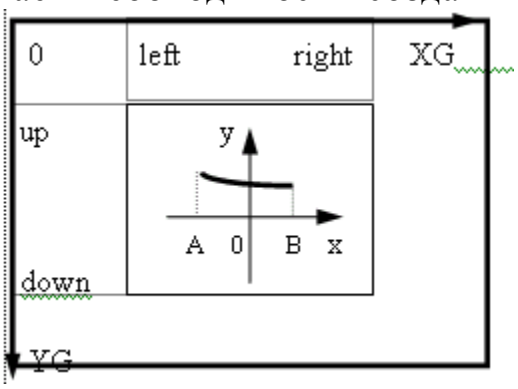
23.11



23.12

24. Построение графиков функций

При построении графиков функций на экране монитора необходимо преобразовывать расчетные координаты в графические с соблюдением определенных пропорций, а также предусмотреть возможность масштабирования графика по осям координат. Это вызывает необходимость создания специального алгоритма и процедур, обеспечивающих универсальность программирования графических изображений. Ниже приводится алгоритм построения графиков в правой системе координат, расположенной в заданной области экрана, с возможностью автоматического масштабирования.



Пусть задана непрерывная функция F(x) в диапазоне изменения аргумента x=[A..B]. Требуется построить по N точкам график функции Y=F(x) в прямоугольной области экрана left, up, right, down. 0 ≤ left, right ≤ GetMaxX
0 ≤ up, down ≤ GetMaxY

Алгоритм построения графика функции Y=F(x).

1). Определяем массивы значений аргумента и функции: x[i], Y[i]=F(x[i]), где i=1... N. При равномерном разбиении интервала [A..B] массивы можно задавать операторами:

```
Dx:=(B-A)/(N-1);          {шаг разбиения по "x"}
for i:=1 to N do begin
x[i]:=A+round(Dx*(i-1));  Y[i]:=F(x[i]) end;
```

2). Определяем наибольшее (Y_MAX) и наименьшее (Y_MIN) значения функции в заданном интервале изменения аргумента:

```
Y_MAX:=Y[1];  Y_MIN:=Y[1];
for i:=1 to N do begin
        IF Y_MAX < Y[i] THEN Y_MAX:=Y[i];
        IF Y_MIN > Y[i] THEN Y_MIN:=Y[i]   end;
```

В случае явного задания функции, для аргумента "x" имеем наибольшее значение X_MAX:=B и наименьшее значение X_MIN:=A.

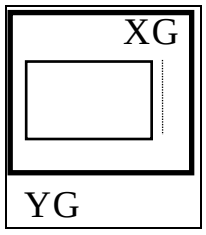
Наибольшее и наименьшее значения функции и аргумента необходимы для полного размещения графика в расчетной области. Эти значения можно изменять с целью уменьшения размеров графика или увеличения его отдельных частей.

3). Определяем коэффициенты масштабирования при построении графика в заданной области экрана:

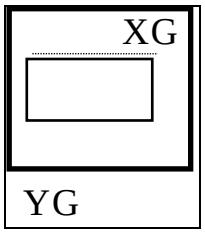
```
Kx:=(right-left)/(X_MAX-X_MIN);
```

Ky:=(down-up)/(Y_MAX-Y_MIN);

Если параметры функции "x" и "y" имеют одинаковую размерность или оба безразмерны, то может появиться искажение естественной формы кривой вследствие разного масштабирования по осям координат: чрезмерное растяжение/сжатие по одной из осей. Например, при рисовании по точкам вместо окружности будет нарисован эллипс. Кроме этого следует учитывать искажение формы графика, регулируемое параметрами процедуры GetAspectRatio(xx, yy). Для вывода графика без искажения формы кривой следует переназначить координаты области экрана так, чтобы получить Ky=K*Kx, где K=xx/yy.



1: pr =1. Пусть заданы нижняя, верхняя и левая границы области построения графика: down, up, left. Необходимо найти значение right при условиях: Ky =K * Kx и right<=GetMaxX. Если условие ограничения графика по длине экрана не выполняется, то полагается right:=GetMaxX; и значение "up" корректируется (уменьшается).



2: pr =2. Пусть заданы левая, правая и нижняя границы области построения графика: left, right, down. Необходимо найти значение up при условиях: Kx =Ky / K и up>=0. Если условие ограничения графика по высоте экрана не выполняется, то полагается up:=0; и значение "right" корректируется (уменьшается).

4). Строим оси координат (начало координат x =0, y =0).

5). Строим график в виде последовательных отрезков, используя аналоги графических процедур BGI:

```
moveto_G(x[1], y[1]);
for i:=2 to N do lineto_G(x[i], y[i]);
```

Пример 24. Написать программу, которая выводит на экран точечный график функции $y=0,5x^2+4x-3$. Диапазон изменения аргумента – от -15 до 5, шаг аргумента – 0,1. График вывести на фоне координатных осей, точка пересечения которых должна находится в центре экрана.

Решение. Созданная программа выводит на экран координатные оси и точечный график заданной функции с использованием PutPixel.

```
Program primer24;
Uses crt,graph;
Var GraphDriver, GraphMode : Integer;
      y,x,dx:real;
      x1,x2:real;
      i,mx,my:integer;
      x0,y0:integer;
      px,py:integer;
Begin
 GraphDriver:=Detect;
      InitGraph(GraphDriver, GraphMode, 'C:\LANGUAGE\BP\Bgi');
 x0:=320; y0:=240;
 my:=20; my:=20;
 {оси координат}
 Line(10,y0,630,y0);
 Line(x0,10,x0,270);
 {график}
 x1:=-15; x2:=5; dx:=0.1; x:=x1;
 While x<x2 do
   begin
      y:=0.5*x*x+4*x-3;
      px:=x0+Round(x*mx);
      py:=y0-Round(y*my);
      PutPixel(px,py,10);
      x:=x+dx;
    end;
 readln;
 CloseGraph;
end.
```

Задачи для самостоятельной работы

Вывести на экран график функции y=f(x), приведенной в таблице, в заданном диапазоне изменения аргумента x от a до b.

№	Y=f(x)	a	b
24.1.	Sin(x)	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
24.2.	Cos(x)	0	$\frac{3\pi}{2}$
24.3.	|sinx+cosx|	0	π

24.4.	|sinx-cosx|	0	π
24.5.	2sinx+3cosx	-π	π
24.6.	sinx+cos2x	-π	+π
24.7.	2sin2x+1	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
24.8.	sinx+cosx+1	-π	+π
24.9.	$\sqrt{x^2+2}$	-3	5
24.10	$\frac{10}{(1+x^2)}$	-3	3
24.11	$\sqrt[3]{x^2}+1$	-1	1
24.12	$\frac{1}{\sqrt[3]{x^2}+1}$	0	10

Литература

1. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. - М.: Наука, 1988. – 320 с.

2. Файсман А.В. Профессиональное программирование на Турбо Паскале. - Ташкент: Инфомех, 1992. – 270 с.

3. Фаронов В.В. Турбо Паскаль 7.0 – М.: Нолидж, 2000.- 416 с.

4. Ставровский А.Б. Турбо Паскаль 7.0. – Киев: BHV, 2000. 400 с.

5. Епанешников А.М. Епанешников В.А. Программирование в среде Turbo Pascal 7.0. - М.: ДИАЛОГ-МИФИ. - 2001. - 367 с.

6. Зуев Е.А. Turbo Pascal. Практическое программирование. - М.: ПРИОР, 1997. – 336 с.

7. Климова Л.М. Pascal 7.0 Практическое программирование. Решение типовых задач. - М.: КУДИЦ-ОБРАЗ. - 2000. – 528 с.

8. Пильщиков В.Н. Сборник задач по языку Паскаль. – М.: Наука, 1989. – 160 с.

9. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию. – М.: Наука, 1988. – 224 с.

10. Юркин А.Г. Задачник по программированию.–СПб.: Питер, 2002.–192 с.

11. Культин Н.Б. Turbo Pascal в задачах и примерах.–СПб.: БХВ–Петербург, 2001. – 256 с.

Подписанов в печать 20.04.2004 г.

Формат: 60x84 1/16 Объем: 4,25 п.л.
Заказ: 22 Тираж: 500 экз.

ОшГУ, Издательский центр «Билим».
г.Ош, ул.Ленина, 331, каб.135.,тел.:7.20.61