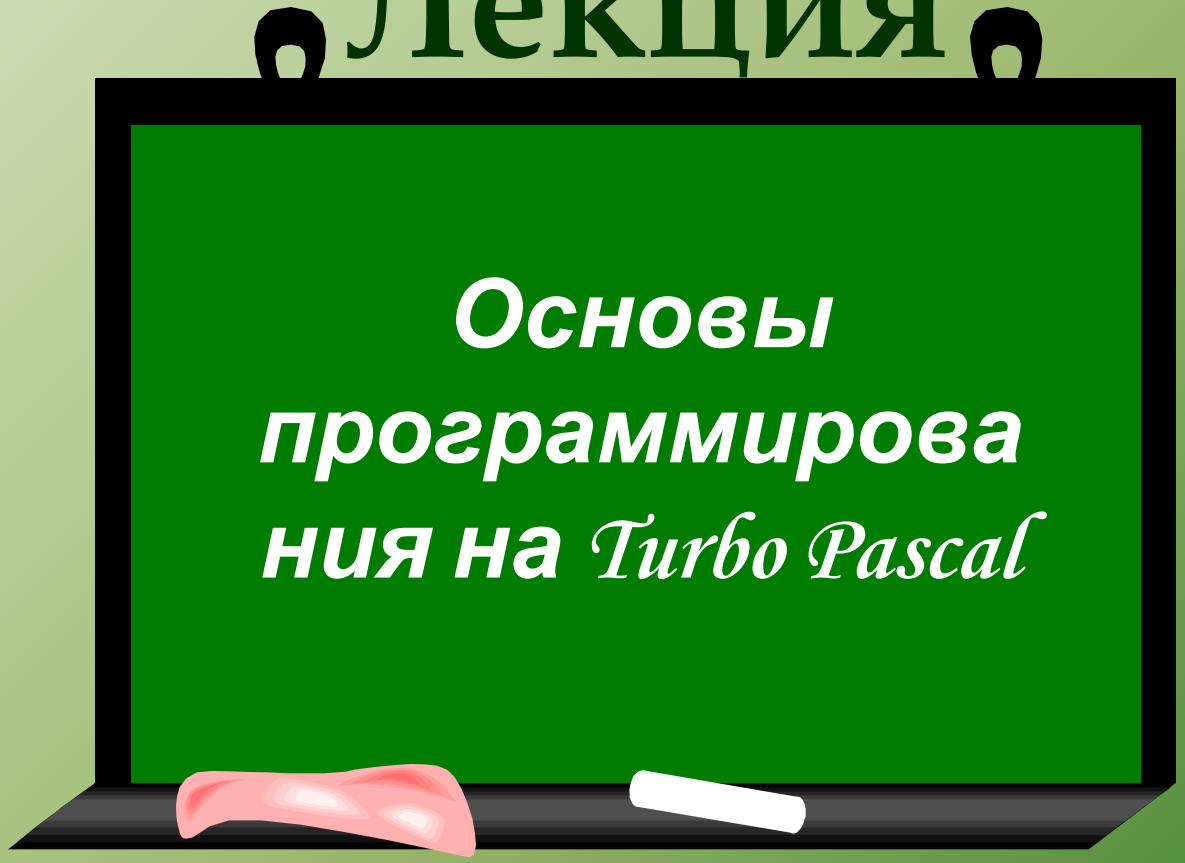
Лекция



Содержание:

Работа в среде Turbo Pascal

Загрузочный файл: **turbo.exe**

Вход в меню: **F10**

Выход из системы: **<Alt – X>**

Операции с файлами, меню **File**:

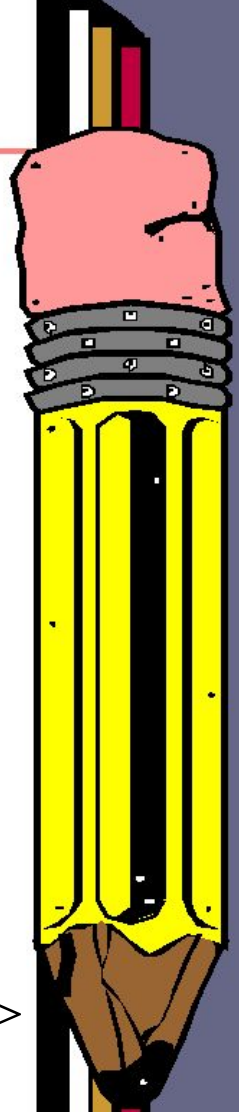
Open – открытие файла (указать путь нахождения файла Pascal-программой)

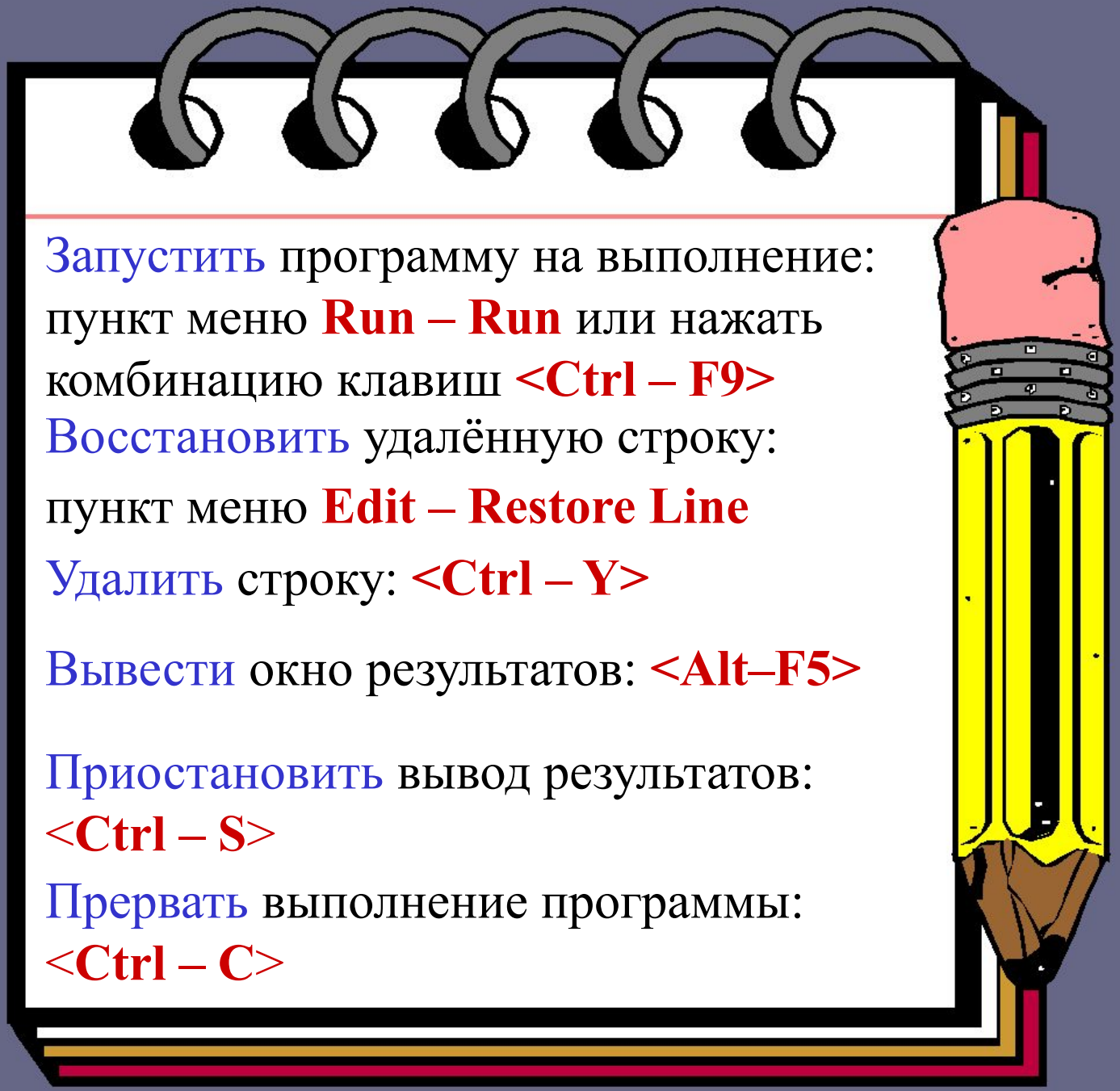
New – создание нового файла

Save – сохранение файла, либо **F2**

Print –печать файла

Exit – выход из системы, либо**<Alt–X>**





Запустить программу на выполнение: пункт меню **Run – Run** или нажать комбинацию клавиш **<Ctrl – F9>**

Восстановить удалённую строку: пункт меню **Edit – Restore Line**

Удалить строку: **<Ctrl – Y>**

Вывести окно результатов: **<Alt–F5>**

Приостановить вывод результатов: **<Ctrl – S>**

Прервать выполнение программы: **<Ctrl – C>**

Выделить блок: начало - **<Ctrl –K – B>**

конец - **<Ctrl –K – K>**

Копировать блок: **<Ctrl –K – C>**

Перенести блок: **<Ctrl –K – V>**

Удалить блок: **<Ctrl –K – Y>**

Снять выделение: **<Ctrl –K – H>**

Основные стандартные функции

Функция	Описание	Тип результата
Abs (x)	**Абсолютное значение, модуль**	Совпадает с типом x
Arctg (x)	**Арктангенс**	Вещественный
Cos (x)	**Cos x**	**-//-**
Exp (x)	**Экспонента e^x**	**-//-**
Frac (x)	**Дробная часть числа**	**-//-**
Int (x)	**Целая часть числа**	**-//-**
Ln (x)	**Натуральный логарифм**	**-//-**

Основные стандартные функции

Функция	Описание	Тип результата
Sin (x)	Sin x	-//-
Sqr (x)	Квадрат аргумента x^2	Совпадает с типом x
Sqrt (x)	Корень квадратный	Вещественный
Random	Случайное число 0<=x<1	Вещественный
Random (n)	Случайное целое число 0<=x<n	Целый
Randomize	Инициализация генератора случайных чисел от таймера ПК	
Round (x)	Округление x до ближайшего целого	Целый

Основные стандартные функции

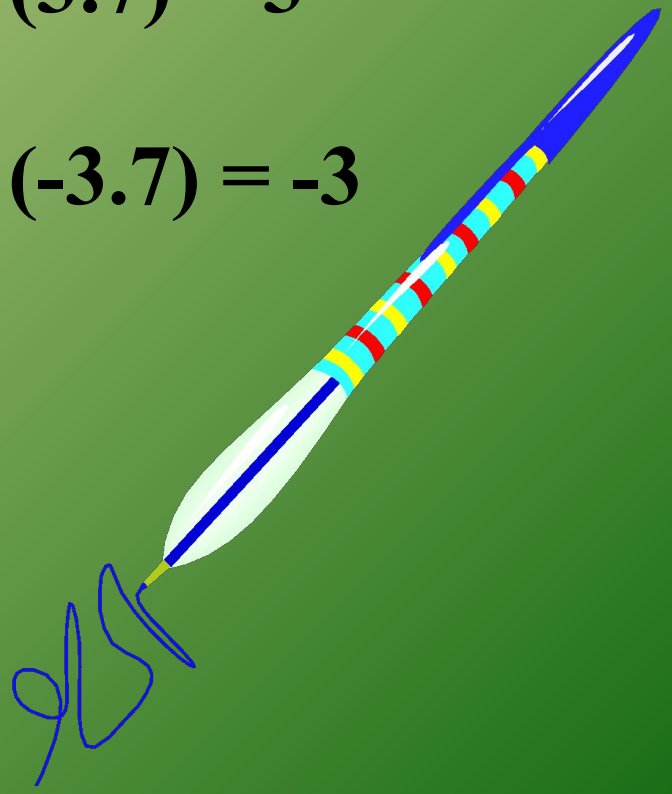
Функция	Описание	Тип результата
Trunc (x)	**Отсечение дробной части x**	целый
Pi	***Пи***	3.1415925635

Приме
р:

Round (3.2) = 3 **Trunc (3.2) = 3**

Round (3.7) = 4 **Trunc (3.7) = 3**

Round (-3.7) = -4 **Trunc (-3.7) = -3**



Математические операции:

- + * - /
- Div - целочисленное деление
- Mod - остаток от целочисленного деления

Операции сравнения:

- = равно
- < > не равно
- < меньше
- <= меньше или равно
- > больше
- >= больше или равно

Знаки пунктуации

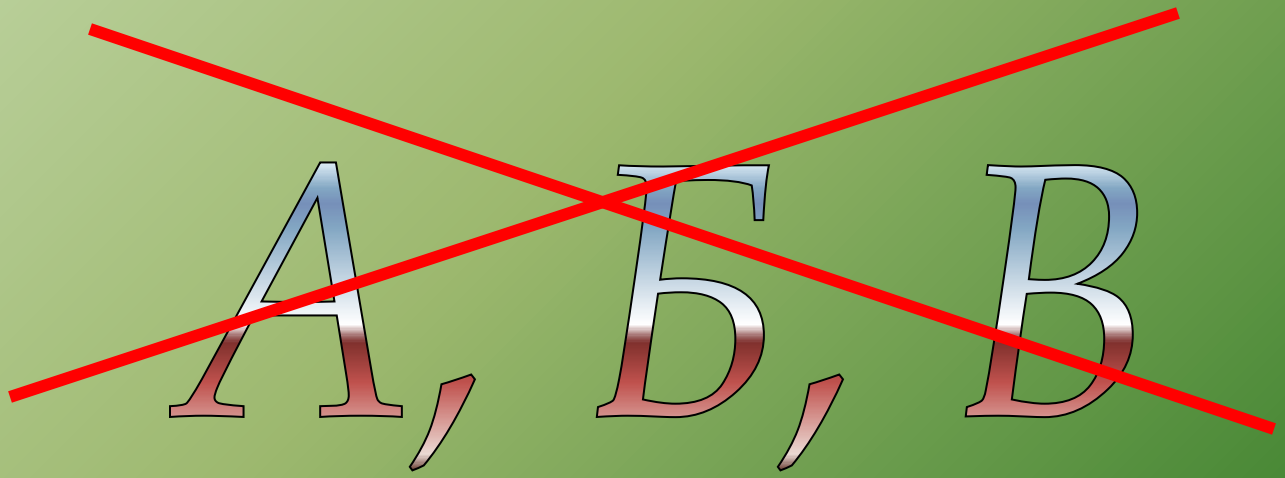
Знак	Название	Функция
[ ]	Квадратные скобки	Индексы массивов, размер строк
( )	Круглые скобки	Математические скобки, список, параметры
.	Точка	Конец программы , десятичный разделитель, отделение полей записи
,	Запятая	Список
;	Точка с запятой	Разделитель операторов
:	Двоеточие	Отделение метки, переменной и типа
'	Апостроф	Обозначение строки или символа
=	Равно	Отделение имени типа от описания типа
#	Решетка (диез)	Обозначение символа по его коду

Знаки пунктуации

Знак	Название	Функция
Двойные знаки:		
(**) { }	Фигурные скобки	Комментарии
:=	Присваивание	Оператор присваивание значения
. .	Две точки	Границы диапазона

Неиспользуемые символы

К неиспользуемым символам относят русские символы. Русскими буквами можно писать только комментарии и строки (в апострофах).



Идентификатор

Идентификатор - это имя любого объекта программы или просто имя; может содержать последовательность латинских букв (от A до Z), цифры от 0 до 9, символ подчёркивания (_) без пробелов. Начинается идентификатор с буквы, максимальная длина имени 63.

Комментарии

это пояснения к программе (на русском языке) в фигурных скобках.

Комментарии при выполнении программы игнорируются. Количество открывающих фигурных скобок должно совпадать с количеством закрывающих скобок.

Порядок выполнения выражений

Приоритет	Тип операции	Выражение
1	Вычисления в скобках	( )
2	Вычисление функций	функции
3	Унарные операции	Not, смена знака -
4	Операции типа умножения	*, /, div, mod, and
5	Операции типа сложения	+, -, or
6	Операции отношения	=, < >, >=, <, <=

Математические выражения выполняются в порядке очерёдности приоритетов, при одинаковом приоритете - слева на право, этот порядок можно изменить только круглыми скобками.

Основные стандартные типы переменных

Паскаль для каждой переменной в программе требует предварительного описания - указания типа. Любая переменная может быть только одного типа. Существует несколько стандартных типов, не требующих предварительного описания:

Основные стандартные типы переменных

Тип переменной	Название типа
INTEGER	целые
REAL	вещественные
BOOLEAN	логические. При сравнении используется логические (булевы) переменные, которые могут принимать только два значения: Истина – True – верно и Ложь – False - не верно
CHAR	символьные
STRING	строковые

Программа

это последовательность инструкций, оформленная по правилам данного языка, которая управляет работой компьютера по заданному алгоритму.

Отладка программы - выявление ошибок и неточностей, как программы, так и алгоритма.

Ошибки программы называются синтаксическими, если они нарушают алфавит языка. Эти ошибки чаще всего обнаруживает среда программирования при компиляции программы.

Ошибки программы называются алгоритмическими, если они приводят к неправильной последовательности действий компьютера. Их должен обнаружить и устранить сам программист.

Порядок решения задачи

- Постановка задачи
- Составление математической модели
- Составление алгоритма программы
- Написание текста программы
- Отладка программы
- Получение результата

Любая программа начинается с заголовка PROGRAM NAME;
где NAME – это имя программы (любой идентификатор).
После этого следуют два раздела:

Раздел описания меток

Раздел операторов (текст самой программы или тело программы).

Раздел описаний состоит из 5 секций:

- Секция описания констант;
- Секция описания переменных;
- Секция описания типов;
- Секция описания меток;
- Секция описания процедур и функций.

Число секций и порядок их следования может быть произвольными. Любая из секций может отсутствовать. Раздел описания заканчивается **BEGIN** и начинается тело программы, которое, в свою очередь, заканчивается словом **END**.

В Паскале существует строгое правило: любой объект программы перед своим использованием должен быть указан в разделе описания. Нарушение этого правила приводит к сообщению об ошибке: "Unknown identifier"

Секция описания констант

Константы - это объект программы, который не может изменять своё значение. Начинается словом Const.

Форма записи:

CONST список констант с указанием значений;

Каждый элемент списка констант состоит из идентификатора, знака = и значения (числа, выражения, слова).

Пример:

```
CONST x = 100;
      Name = ‘Иван Иванович’;
      y = x + 2;
```

Секция описания переменных и типов

Переменная - это объект программы, который может менять своё значение при выполнении программы. Любая переменная до первого своего использования должна быть описана. Все переменные указываются после служебного слова Var.

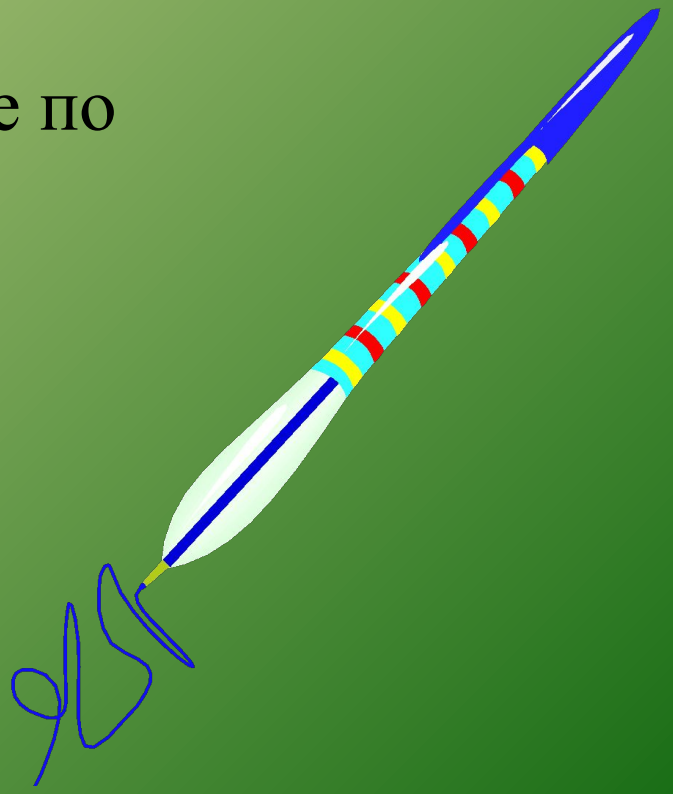
Форма записи:

Var имя1, имя2, … : тип;

Пример:

```
Var a, b, c, d, x: real;
    f          : char;
```

Все переменные получают значение по умолчанию равное 0.



Секция описания меток

В программе любой оператор может быть отличен с помощью метки для перехода при необходимости в эту часть программы.

Метка- это любой идентификатор или число от 0 до 9999. В теле программы метка определяется двоеточием. Ключевое слово Label.

Метки используются для изменения последовательного сверху вниз выполнения операторов и перехода на нужный оператор в любом месте программы. Если метка перечислена в данной секции, она обязательно должна быть использована в программе.

Пример:

```
Label     9999, label_1, metka_10;

Begin
     ..........
   9999:          x:=
   label_1:       d:=
   metka_10:      z:=
```

Секция описания функций

Любую функцию в Паскале перед тем как её использовать в программе необходимо описать и задать. Секция начинается с ключевого слова **FUNCTION**.

Форма записи:

```
FUNCTION   f(x: тип): тип;
begin
f:= выражение
end;
```

Далее в программе к функции обращаются по f(x)

Операторы Turbo Pascal

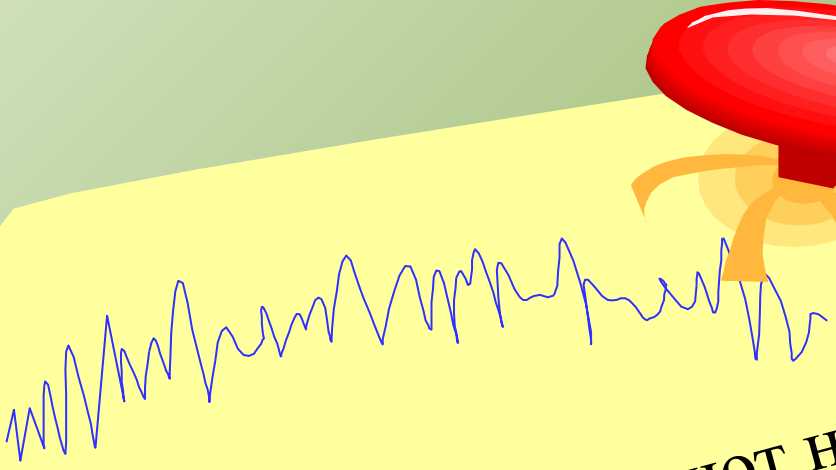
Операторы – это синтаксические конструкции, которые позволяют согласно алгоритму задач обработать исходные данные и получить требуемый результат. Операторы состоят из служебных слов и слов пользователя. Тело программ представляет собой набор операторов.

Любую программу можно отнести к одному из трёх видов:

Последовательное · выполнение операторов (сверху вниз и слева направо)

Циклические · или повторяющиеся действия. Количество повторений может быть или не быть заранее известными.

Разветвленные · действия. Согласно некоторому условию выполняется та или другая часть программы.



Операторы разделяют на ***простые*** и ***структурные***.

В состав структурных могут входить другие операторы.

Простые операторы:

- Оператор присваивания
- Оператор процедуры
- Оператор перехода на метку

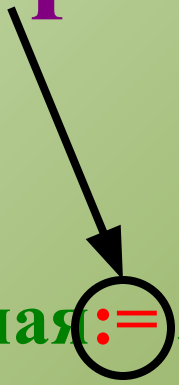
Структурные операторы:

- Составной
- Условный
- Оператор варианта
- Оператор процедуры
- Оператор цикла с параметром
- Оператор цикла с предусловием
- Оператор цикла с постусловием

В конце оператора ставится ; Операторы можно записывать отдельно в каждой строке или несколько операторов в одной строке через точку с запятой.

Оператор присваивания

Форма записи:



Переменная := значение;

Переменной присваивается значение или выражение.

Пример:

```
a: = 1;
b: = 1.5;
c: = a+b;
st : = 'пример текста';
```

Оператор процедуры

Имя процедуры записывается в отдельной строке и заканчивается **;** .

Если процедура имеет параметры, то они записываются в круглых скобках.

Процедура очистки экрана:

второй строкой после Program должна стоять запись **Uses crt;**

Сама процедура очистки экрана **ClrScr;** (Clear Screen)

Процедура ввода с клавиатуры **Read** и процедура вывода на экран **Write.**

Чаще используется вторая форма **Readln** и **Writeln.**

Окончание **Ln** (Line) означает переход на следующую строку.

Если нужно ввести несколько данных, то параметры **Readln** записываются через запятую. При вводе данных на экран, в круглых скобках указывается список вывода. В список могут входить ***переменные*** через запятую ***строки текстов*** в апострофах, ***выражения***, ***имена функций.***

По умолчанию имена выводятся в экспоненциальной форме (с плавающей точкой), 11 знаков после запятой. Для более наглядного результата используется ***формат вывода:***

а) для вещественных чисел:

Writeln (x: n1: n2);

Где **n1** - общее число знаков результата, включая + - и десятичную точку,

n2 - число цифр после запятой, результат при этом округляется.

Пример:

1,23456 формат :5:3 на экран выведен **1,235.**

Формат вида x:0:0 автоматически определяет число знаков округлённого целого числа.

б) формат вывода для целых чисел

Writeln (x: n1);

где **n1** - общее число знаков.

в) для строк текста

Write ('Пример текста': 40,'с форматом':10)

Для вывода текста в апострофах отводится указанное число знаков с выравниванием по правому краю. В одной строке помещается 80 знаков.

Оператор перехода на метку

Этот оператор используется для перехода в нужный пункт программы на оператор с той же меткой.

Форма записи:

Goto метка;

В разделе описания указывается метка.

Приме

р:

```
Label 10;
Begin
    a:=1;
    …….;
10:   b:=3;
    ……..;
    Goto 10;
End.
```

Составной оператор

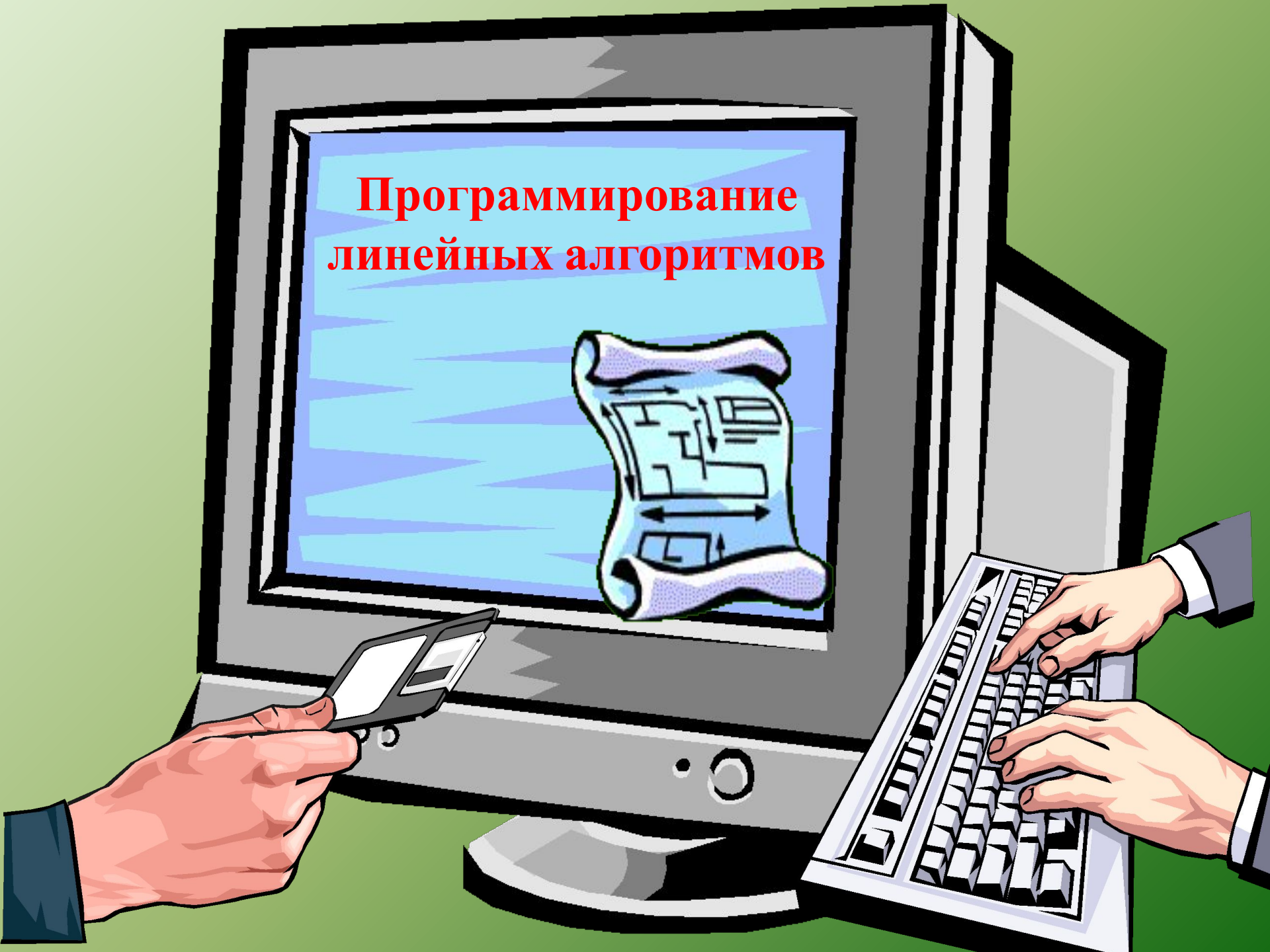
Когда необходимо, чтобы несколько операторов выполнялись как одно целое, то используется составной оператор **Begin … End;**

Эти два слова иначе называют операторными скобками. Всё, что находится между этими словами считается одной группой.

Составные операторы могут быть вложенными друг в друга, например:

```
Begin
    begin … end;
    …
End;
```

Программирование линейных алгоритмов



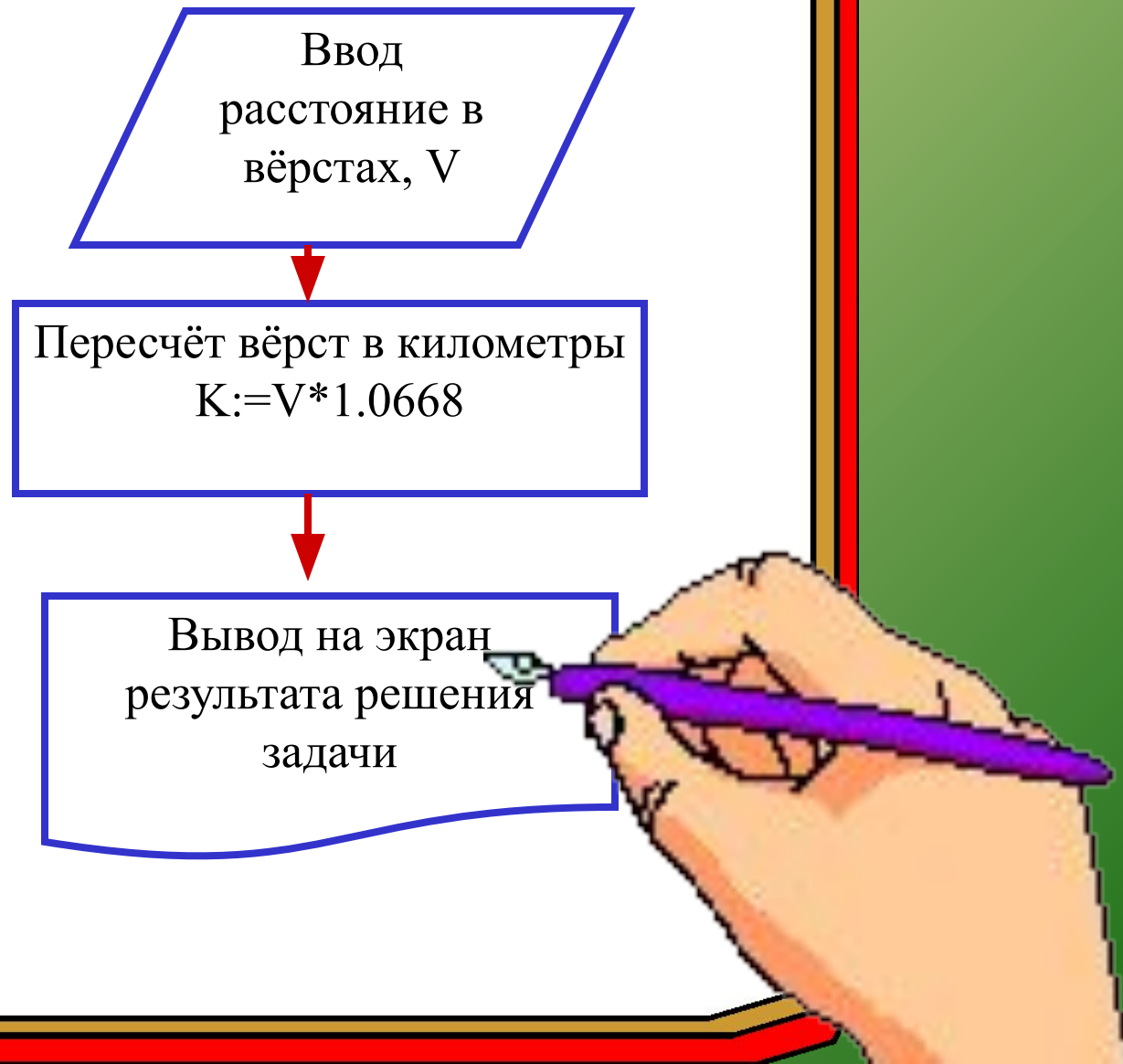
Пример решения задачи:

Постановка задачи:

Составить программу пересчёта расстояния из вёрст в километры (1 верста = 1066,8 м).

Пример решения задачи:

Составление алгоритма:



Пример решения задачи:

Составление программы:

```
PROGRAM Versta_Km;
VAR v: real;  {расстояние в вёрстах}
      k: real; {расстояние в км}
BEGIN
   writeln ('Пересчёт расстояния из вёрст в км');
   writeln ('Введите расстояние в вёрстах');
   readln (v);
   k:=v*1.0668;
   writeln(v:6:2,' верст – это ' ,k:6:2, ' км');
   readln;
END.
```

Задачи для решения

Составить программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к ним:

Вычисление стоимости покупки.
Введите исходные данные:
Цена тетради (руб.): **2.75**
Цена обложки (руб.): **0.5**
Количество комплектов (шт.): **7**

Стоимость покупки: 22.75 руб.

ПЕРЕМЕНА

Условный оператор

Этот оператор используется для выполнения одного из двух возможных вариантов программы.

Форма записи:

if логическое_условие
then оператор_1
else оператор_2;

если **логическое_условие** верно, то выполняется **оператор_1** , иначе **оператор_2**;

Перед else точка с запятой не ставится!

Существует вторая упрощенная форма этого оператора:

if логическое_условие then оператор_1;

Вначале проверяется **логическое_условие**, если оно верно, то выполняется **оператор_1**, после этого выполняется оператор, расположенный ниже, если **логическое_условие** - ложно, то **оператор_1**, не выполняется, выполняется оператор ниже.

Логическое_условие может содержать одно или несколько условий, заключенных в круглые скобки и разделенными функциями And, Not, Or.

Оператор_1 и **оператор_2** могут быть простыми и составными.

Приме
р:

```
1)   If  D<0      then writeln ('Корней нет')
          else  begin
                  x1:=(-b+sqrt(d))/2/a;
                        x2:= (-b-sqrt (d))/2/a;
                end;
  2) If x>=x0 then
                      begin  b:=1;
                             c:=b*x
                      end
                else
                      begin
                         b:=-1;
                              c:=x;
                      end;
```

Условные операторы могут быть вложены друг в друга.

Оператор варианта

Если в программе надо выбрать один вариант выполнения из трёх возможных и более, то нужно использовать оператор варианта (выбора) Case.

Форма записи:

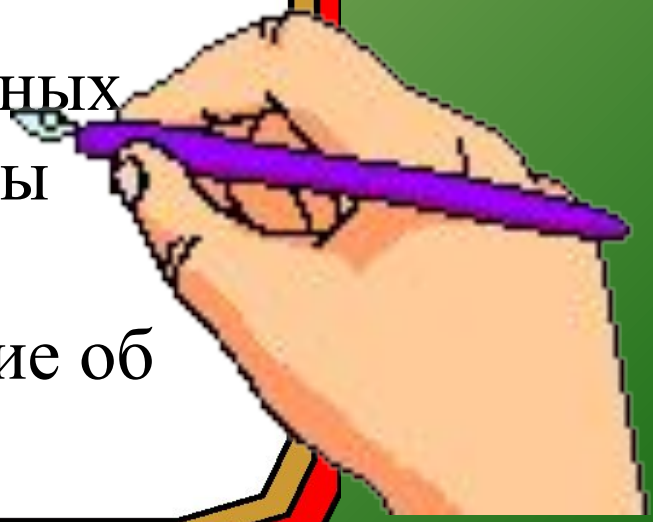
```
Case  n of
Значение_1 : оператор_1;
Значение_2 : оператор_2;
Значение_3 : оператор_3;
...
Else оператор_else
End;
```

Программирование разветвляющихся алгоритмов

Пример решения задачи:

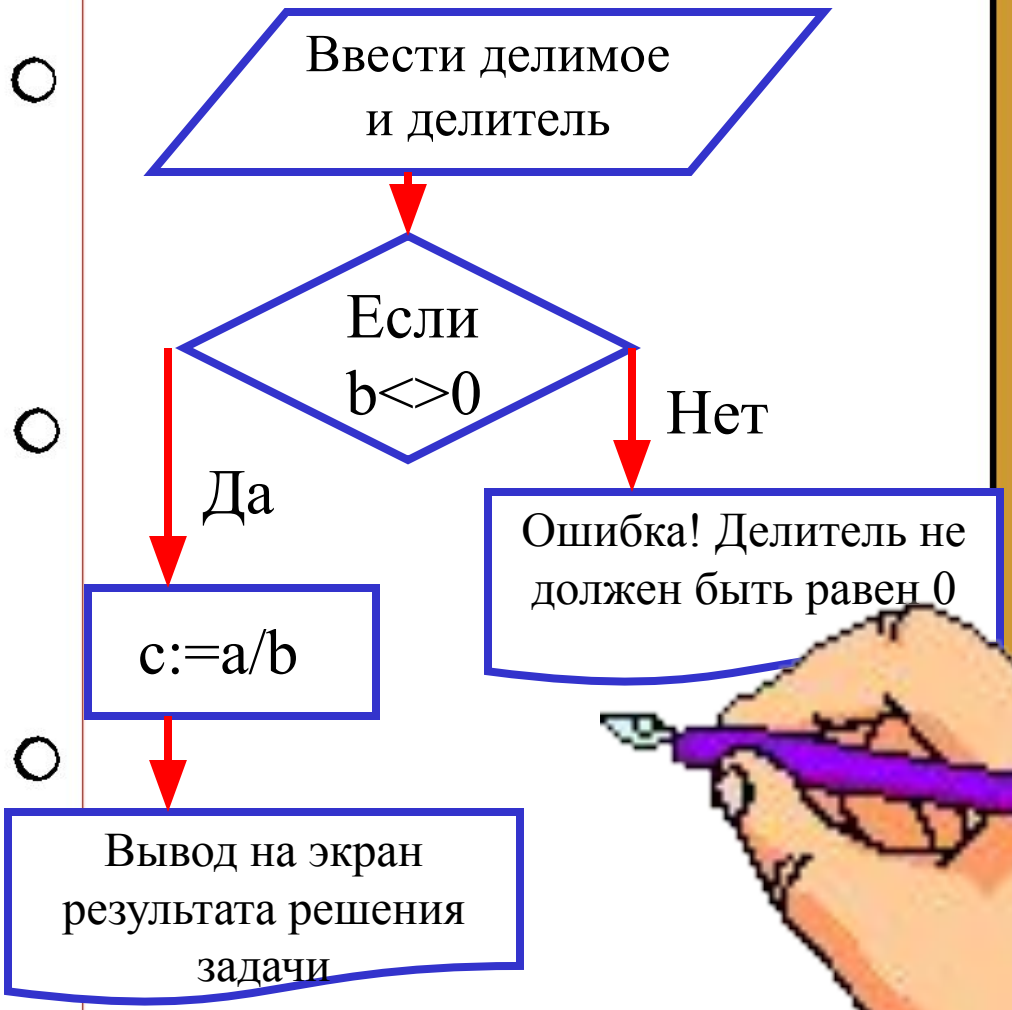
Постановка задачи:

Составить программу, которая вычисляет частное от деления двух чисел. Программа должна проверять правильность введённых пользователем данных и, если они неверны (деление на 0), выдавать сообщение об ошибке.



Пример решения задачи:

Составление алгоритма:



Пример решения задачи:

Составление программы:

```
PROGRAM Chastnoe;
VAR a,b,c: real;  {делимое, делитель и частное}
BEGIN
   writeln (’Вычисление частного’);
   writeln (’Введите в одной строке делимое и делитель’);
    readln (a,b);
     if b<>0 then
                    begin
                        c:=a/b;
                         writeln (’частное от деления  ’ ,a:6:2,
                                                ’ на’,b:6:2, ’равно  ‘,c:6:2);
                     end
                else
  writeln(’Ошибка! Делитель не должен быть равен 0’);
      readln;
END.
```

Задачи для решения

Составить программу вычисления стоимости покупки, с учётом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., 5% - если сумма больше 1000 руб.

Вычисление стоимости покупки с учётом скидки.

Введите сумму покупки: **640**

Вам предоставляется скидка 3%

Стоимость покупки с учётом скидки: 620.80 руб.

Пример решения задачи:

Постановка задачи:

Составить программу для вычислений суммы, произведения, разности и деления двух чисел (калькулятор).

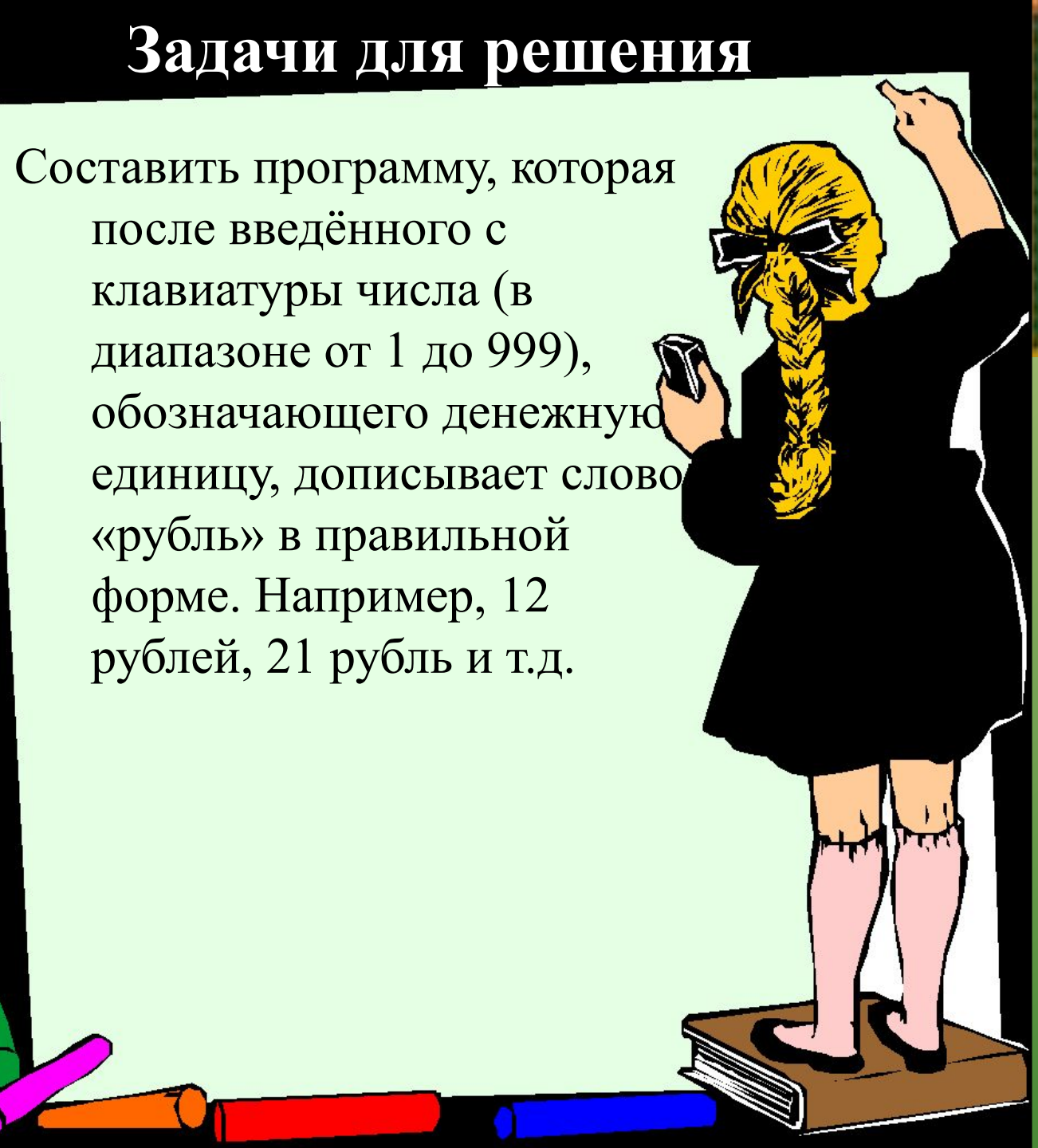
Пример решения задачи:

Составление программы:

```
Program Calculator;
uses crt;
VAR OP:        char;      {операция}
       X,Y, Z:   real;      {числа}
BEGIN
      clrscr;
writeln ('Введите исходные данные');
writeln ('Введите число X');
readln (X);
writeln ('Введите число Y');
readln (Y);
writeln ('Введите операцию OP');
readln (OP);
   Case OP of
        '+'  :  Z:=X+Y;
        '*'  :  Z:=X*Y;
        '- '   : Z:=X-Y;
        '/ '   : Z:=X/Y;
        '^ '  : Z:=exp(Y*ln(X));
     End;
writeln ('Z=', Z);
END.
```

Задачи для решения

Составить программу, которая после введённого с клавиатуры числа (в диапазоне от 1 до 999), обозначающего денежную единицу, дописывает слово «рубль» в правильной форме. Например, 12 рублей, 21 рубль и т.д.



ПЕРЕМЕНА

Оператор цикла с параметром

Цикл – это многократное повторение некоторой части программы до выполнения заданного условия. Оператор For используется в том случае, если заранее число повторений цикла.

Форма записи:

for параметр: = Начальное_значение to конечное_значение do оператор;

При этом **параметр, Начальное_значение** и **конечное_значение** должны быть целого типа Integer.
Начальное значение должно быть меньше конечного.
При этом шаг параметра равен 1.

Цикл работает следующим образом:

1) Параметр принимает **начальное_значение**

2) Выполняется **оператор**

3) **Параметр** увеличивается на единицу

4) Если **параметр** больше **конечного_значения**, то происходит выход из цикла, если меньше, то повторяется пункт 2.

Замечания: Внутри цикла изменять параметр нельзя! Изменять шаг нельзя!

Существует вторая форма записи оператора:

for параметр: = Большее_значение down to Меньшее_значение do Оператор;

При этом шаг **параметра** равен -1.

Оператор цикла с предусловием

Этот оператор повторяется пока истинно некоторое условие. Условие проверяется в начале, перед циклом поэтому, если условие сразу же ложно, то цикл не повторяется ни разу.

Форма записи:

```
While условие do
    Begin
        Операторы;
    End;
```

После do ставить точку с запятой нельзя!

Этот оператор часто используется для организации паузы в программе до нажатия любой клавиши.

```
Uses crt ;
Begin
        …
      While not Key_Pressed do;      {пауза}
        …
End.
```

Оператор цикла с постусловием

Этот оператор повторяется до тех пор, пока не выполнится определённое условие.

Форма записи:

```
Repeat
    Оператор_1;
    Оператор_2;
    …
    Оператор_n
Until  условие;
```

Пример:

```
S:=0;
K:=1;
Repeat
        S:=S+K;
        K:=K+1;
Until K>10;
```

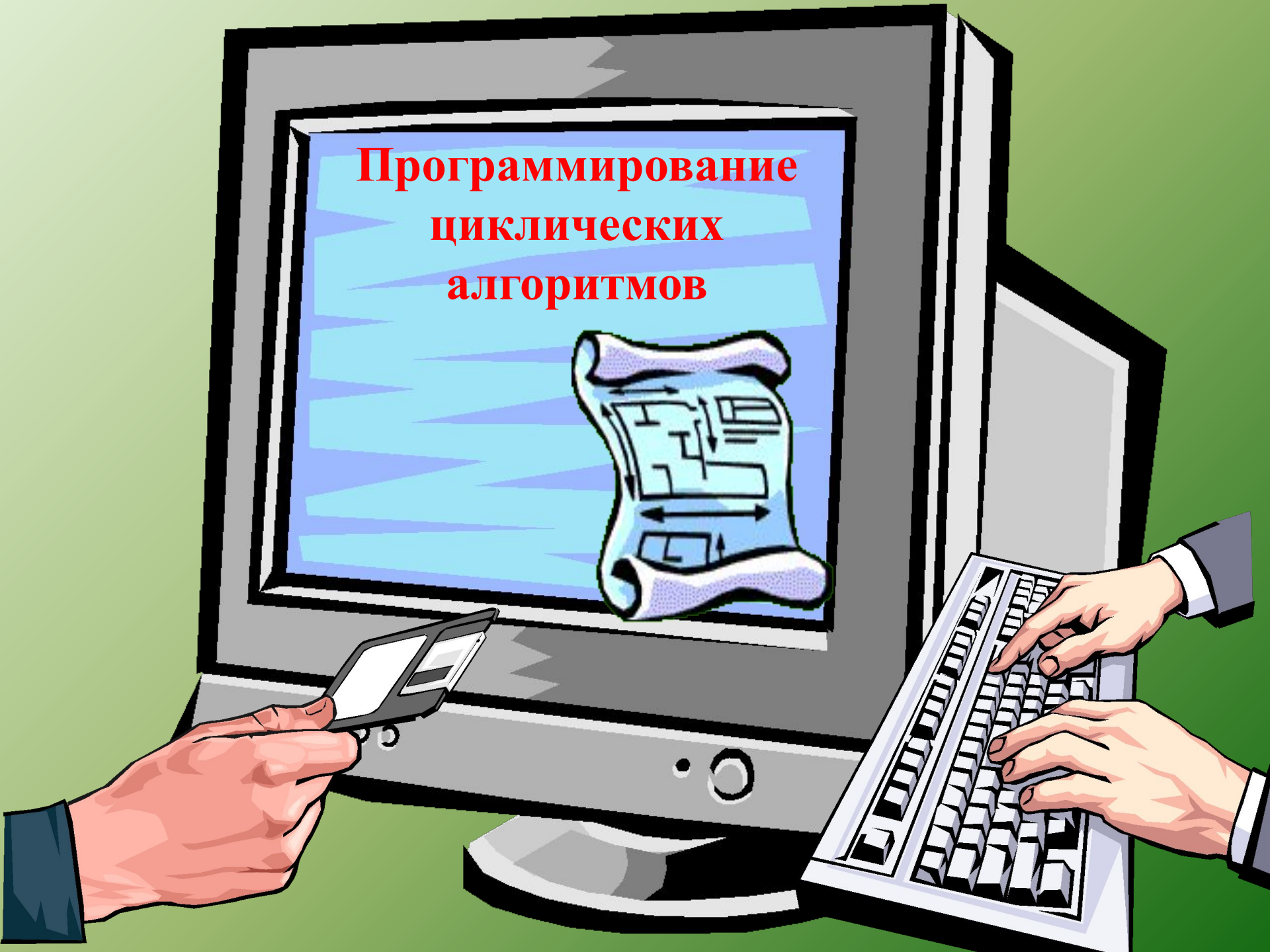
Стандартные процедуры Break и Continue

В Turbo Pascal 7.0 для циклов for, while, repeat введены две специальные процедуры:

1) Процедура досрочного выхода из цикла **Break**

2) Процедура перехода на следующий шаг до полного окончания выполнения текущего шага **Continue**.

Программирование циклических алгоритмов



Пример решения задачи:

Постановка задачи:

Напечатать таблицу с шагом 5° от 0° до 60°.

X, град	X, рад	Sin(x)	Cos(x)	Tan(x)

Пример решения задачи:

Составление алгоритма:

Пример решения задачи:

Составление программы:

```
Program Tablica;
uses crt;
VAR xg:integer;
      xr,a,b,c,h:real;
BEGIN
      clrscr;
      xg:=0;
      h:=pi/180;  {коэффициент перевода из градусов в радианы}
      writeln('_____');
      writeln('  xg    xr    sin(x)  cos(x)  tan(x)');    {шапка таблицы}
      writeln('_____');
      repeat
            xg:=xg+5;
            xr:=xg*h;
            a:=sin(xr);
            b:=cos(xr);
            c:=sin(xr)/cos(xr);
            writeln(xg:5,xr:8:2,a:8:2,b:8:2,c:8:2);
      until xg>=60;
 END.
```

Задачи для решения

1. Ввести натуральное число N. Получить все его натуральные делители.

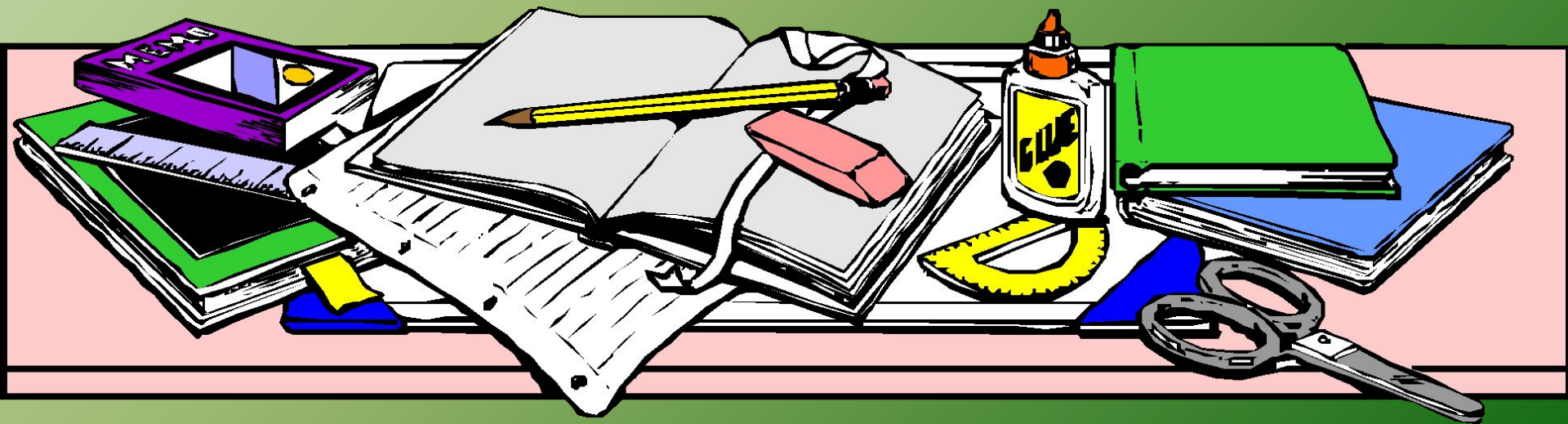
2. Составить программу для нахождения наибольшего общего делителя двух натуральных чисел **M** и **N** по алгоритму Евклида:

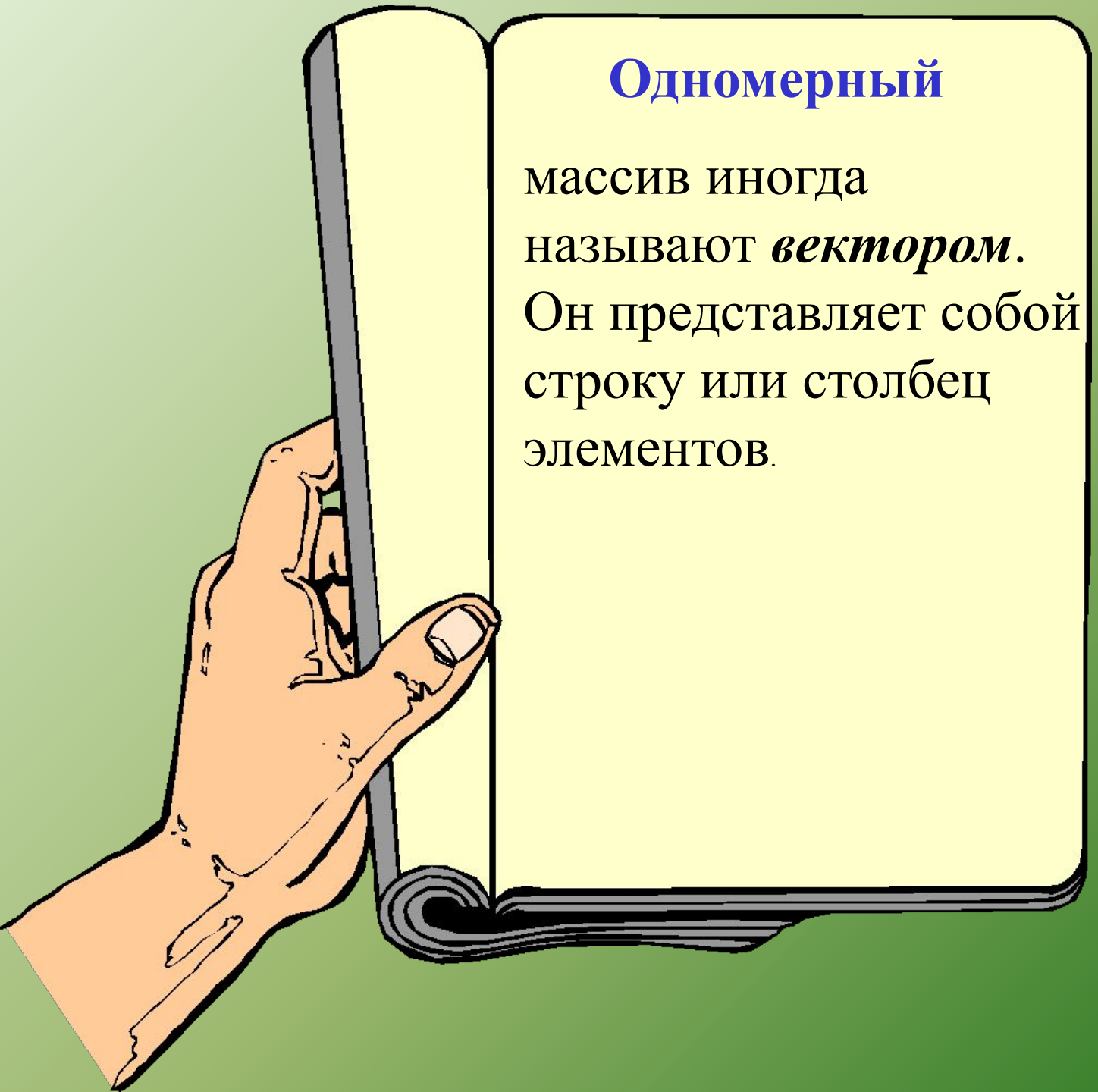
НОД=M, если M=N;
если M>N, то M=M-N,
иначе N=N-M

ПЕРЕМЕНА

Массивы

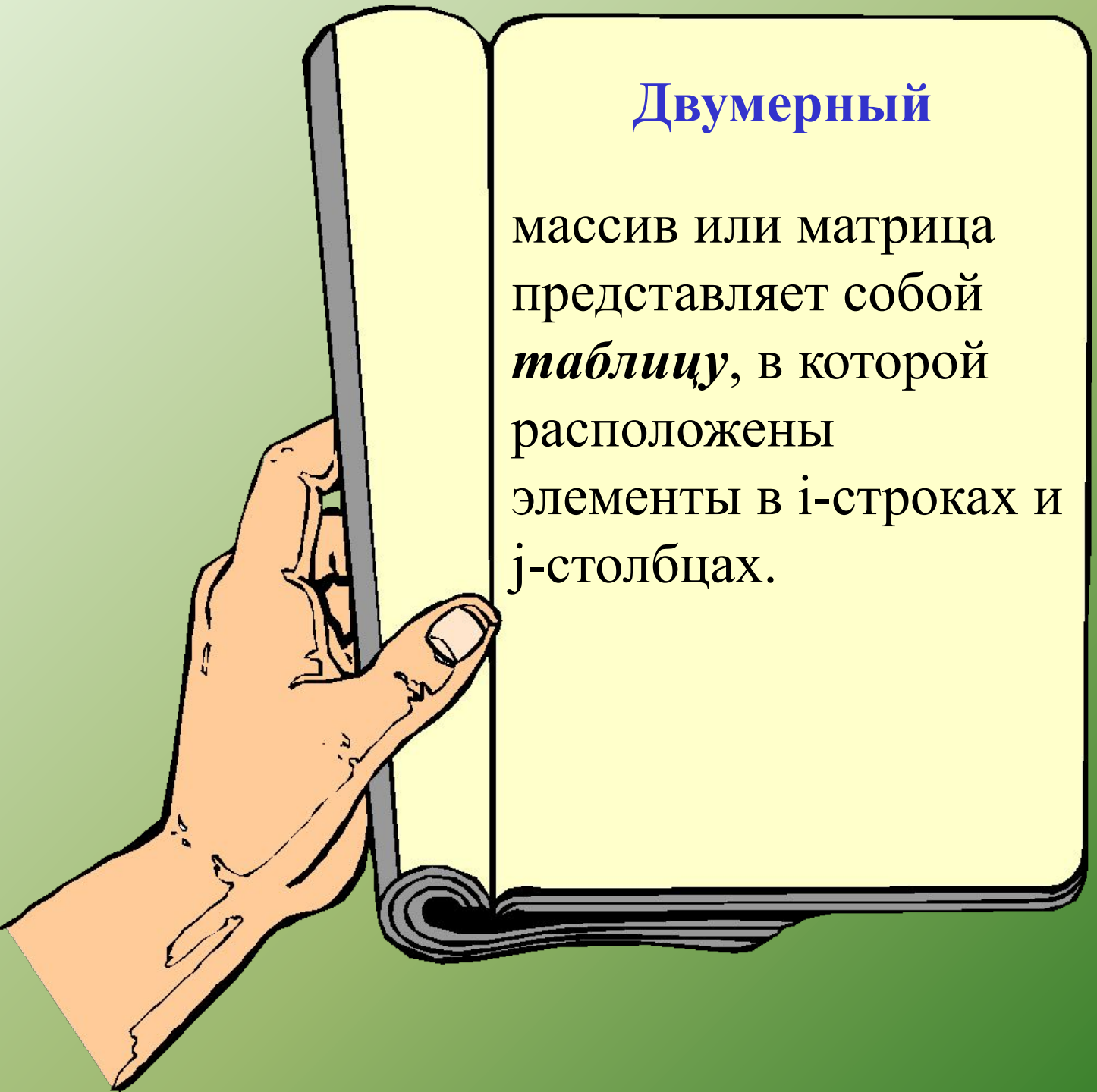
Массив - это упорядоченный набор однотипных элементов, каждый из которых имеет свой индекс. Обращение к элементу производится по индексу. Имя элемента состоит из имени массива с индексом. Массив в программировании аналогичен матрице в математике.





Одномерный

массив иногда называют ***вектором***. Он представляет собой строку или столбец элементов.



Двумерный

массив или матрица представляет собой ***таблицу***, в которой расположены элементы в i-строках и j-столбцах.

Перед тем как использовать массивы в программе его необходимо описать в секции описания переменных.

Форма записи одномерного массива:

Var Имя_массива: array [1..n] of тип;

Пример:

Var A: array [1..10] of Real;

Форма записи двумерного массива:

Var Имя_массива : array [1..n, 1..k] of тип;

Пример:

Var B: array [1..3, 1..4] of Integer;

Для того, чтобы обратиться к конкретным элементам массива, нужно указать имя массива и в квадратных скобках его конкретный индекс. Индексы указывают через запятую.

Массивы

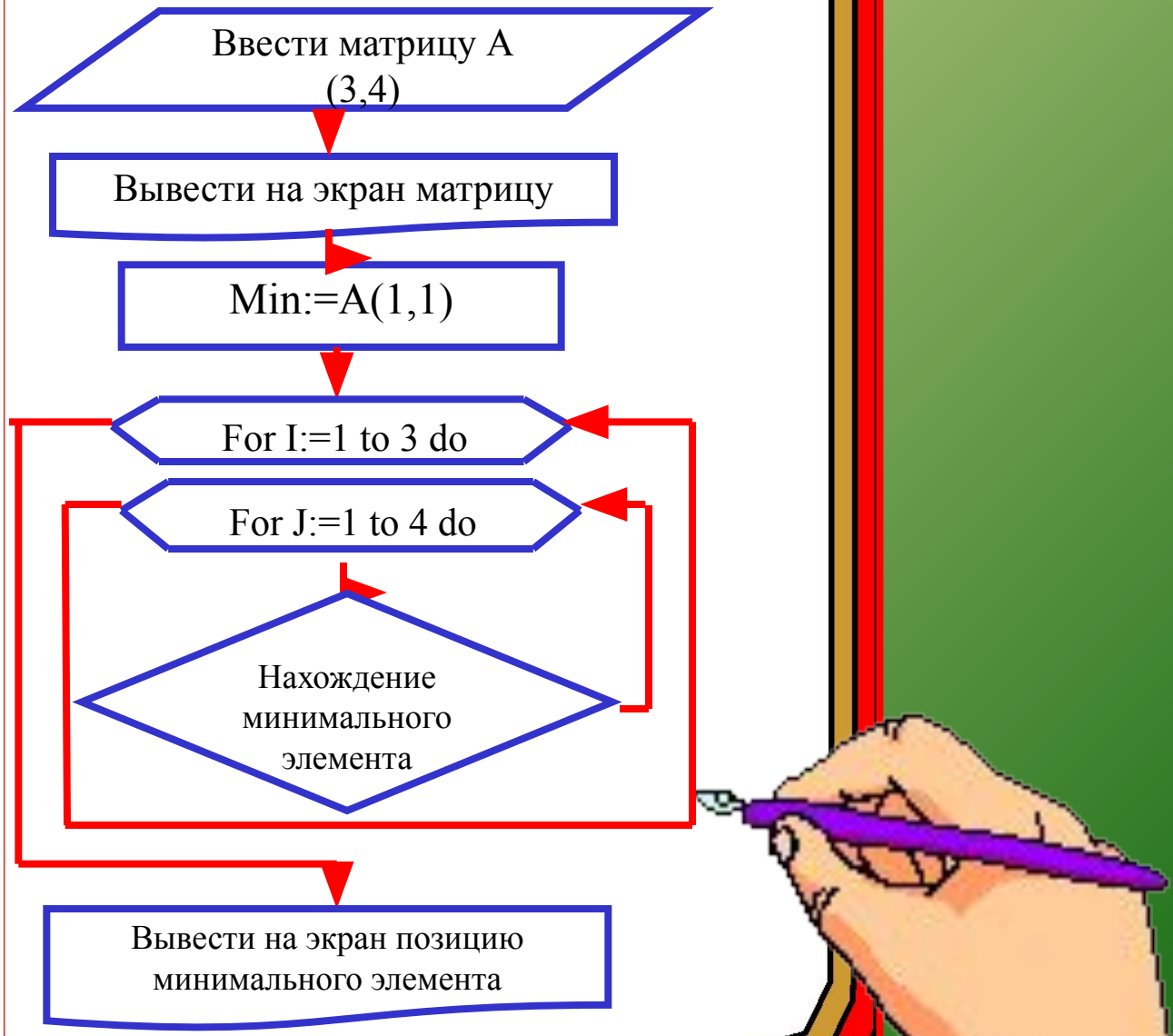
Пример решения задачи:

Постановка задачи:

Ввести матрицу A(3,4).
Найти минимальный элемент и напечатать его позицию.

Пример решения задачи:

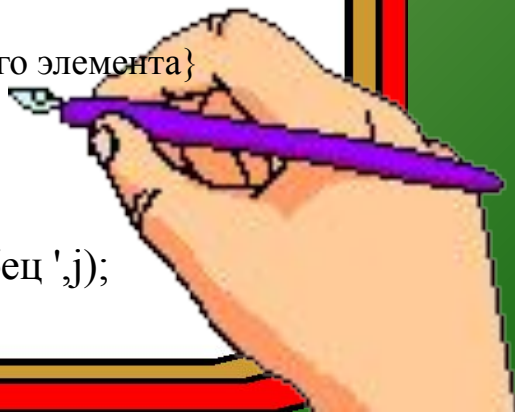
Составление алгоритма:



Пример решения задачи:

Составление программы:

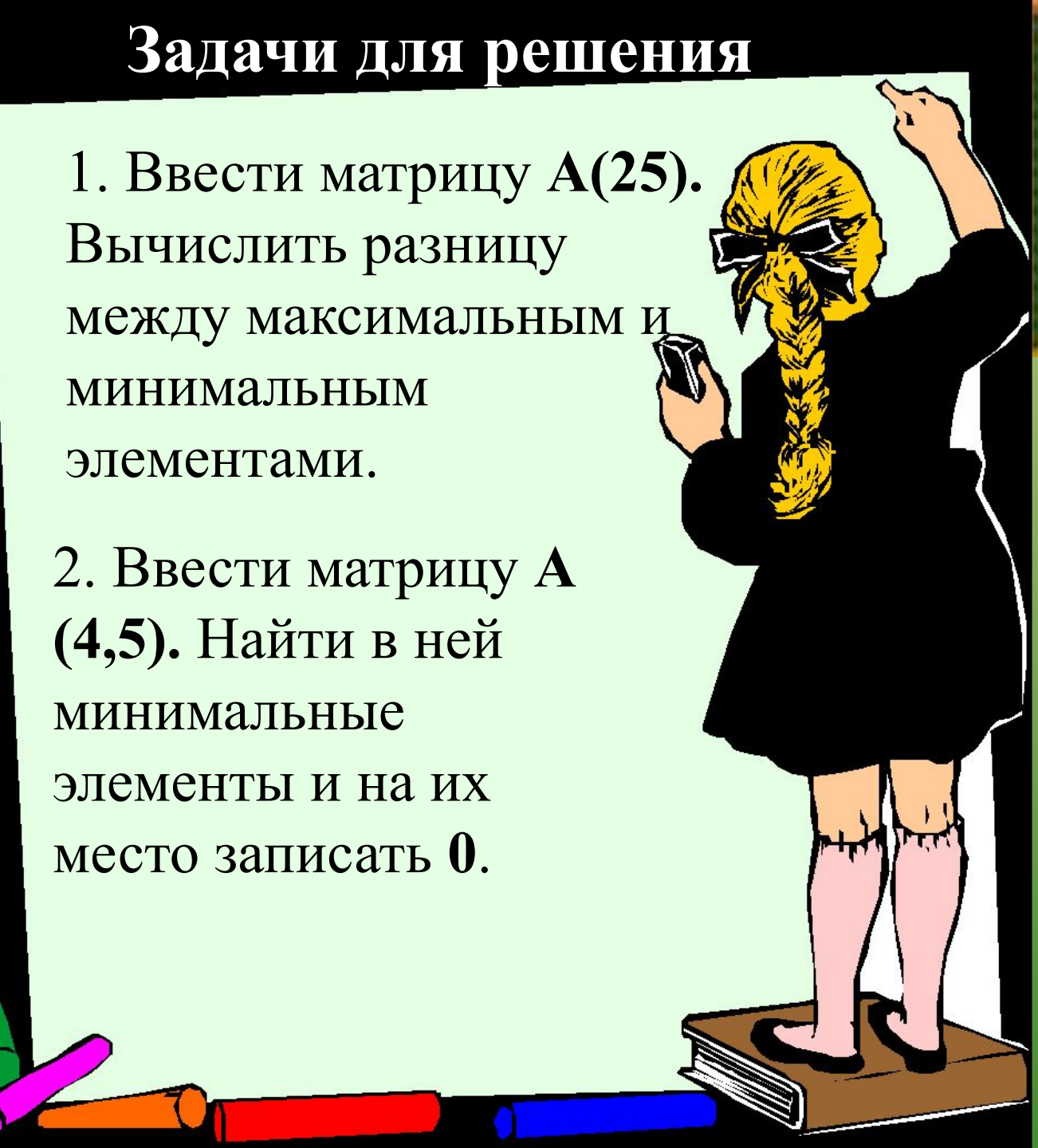
```
Program Matrica;
uses crt;
VAR A:array[1..3,1..4] of integer;
        i,j,min:integer;
BEGIN
    randomize;
    for i:=1 to 3 do    {генератор случайных чисел}
    for j:=1 to 4 do
    A[i,j]:=random(50);
 clrscr;
   for i:=1 to 3 do
        begin
           for j:=1 to 4 do {вывод исходной матрицы на экран}
           write (A[i,j]:5);
           writeln;
         end;
    writeln;
    min:=A[1,1]; {нахождение минимального элемента}
    for i:=1 to 3 do
    for j:=1 to 4 do
        if A[i,j]<=min then min:=A[i,j];
    writeln(' min=',min,' строка ',i,'  столбец ',j);
END.
```



Задачи для решения

1. Ввести матрицу **A(25).** Вычислить разницу между максимальным и минимальным элементами.

2. Ввести матрицу **A (4,5).** Найти в ней минимальные элементы и на их место записать **0**.



ПЕРЕМЕНА

Строковый тип String

Тип строка во многом похож на одномерный массив, его элементами являются отдельные символы.

Размер строки заранее может быть неограниченным (<255).

Кроме того, размер строки может быть задан явно в квадратных скобках.

Пример:

```
Var St: String[10];
```

При попытке ввести более 10 символов, все "лишние" символы игнорируются.

Для определения фактической длины строки имеется стандартная функция Length (st).

Значение строковых переменных записывается в апострофах.

Приме
р:

```
Program Print_String;
Uses crt;
Const st =  ’Turbo-Pascal’;
Var     i : integer;
Begin
     clrscr;
     for i := 1 to Length ( st ) do
     begin
          write ( st[i] ) ;
          delay (100)
     end;
     readln
End.
```

Запись информации в файл

Файл записанный из программы может хранить произвольный набор информации.

Обязательные условия для работы с файлом:

1) В программе должно быть специальная файловая переменная, связывающая программу с файлом на диске.

2) Файл вначале должен быть открыт для записи, в конце программы файл должен быть закрыт.

Основным файлом является **текстовый**. В программе для указания имени файла используется файловая переменная, которая объявляется следующим образом:

```
Var   F : text;
```

Организация записи в файл:

1. объявить файловую переменную как текстовую;
2. подключить файловую переменную к имени файла;
3. открыть файл для записи;
4. записать фрагмент файла, оператор записи **Rewrite (f);**
5. закрыть файл оператором **Close(f).**

Примечание: для добавление информации к уже созданному файлу используется оператор **Append (f).**

Рекомендации: первую и вторую операцию обычно указывают в начале программы после основного **Begin**, 3-ю – перед первым оператором записи в файл write (f,…), 5-ую операцию **close (f)** обычно указывают в конце программы.

Пример: записать в файл введённую матрицу

```
Program File_Matriv;
Var M:array [1..4, 1..3] of Real;
      i, j: integer;
      f: text;
BEGIN
    assign (f,' a:\massiv.txt');
    rewrite (f);
    writeln (' Введите матрицу');
     for i:=1 to 4 do
     for j:=1 to 3 do
     readln (M[i,j]);
     for i:=1 to 4 do
        begin
             for j:=1 to 3 do
                  write (f, M[i,j]);
             writeln (f);
        end;
      close (f);
END.
```

Чтение из файла

Операция чтения из файла подобна записи в файл, отличие только в том, что открытие файла для записи **Rewrite (f)** заменяется на открытие файла для чтения **Reset** (f).

Чтение производится оператором read (f,…)

Замечания:

1) Файл записывается по умолчанию в ту же папку, где хранится файл программы.

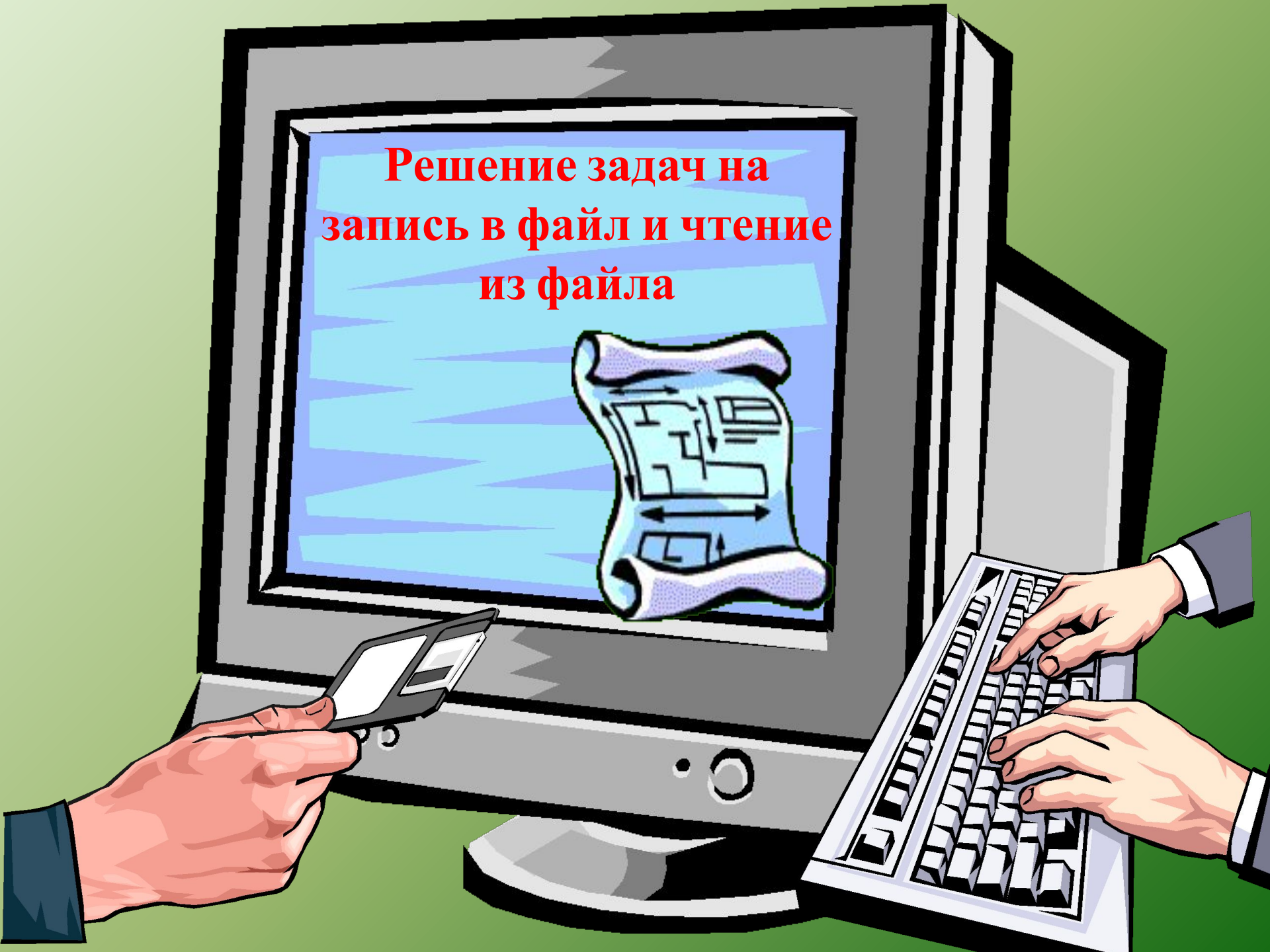
2) Если файл не закрыть, то он может остаться пустым или частично записанным.

Пример:

прочитать из файла ранее записанную в него матрицу.

Самостоятельно!

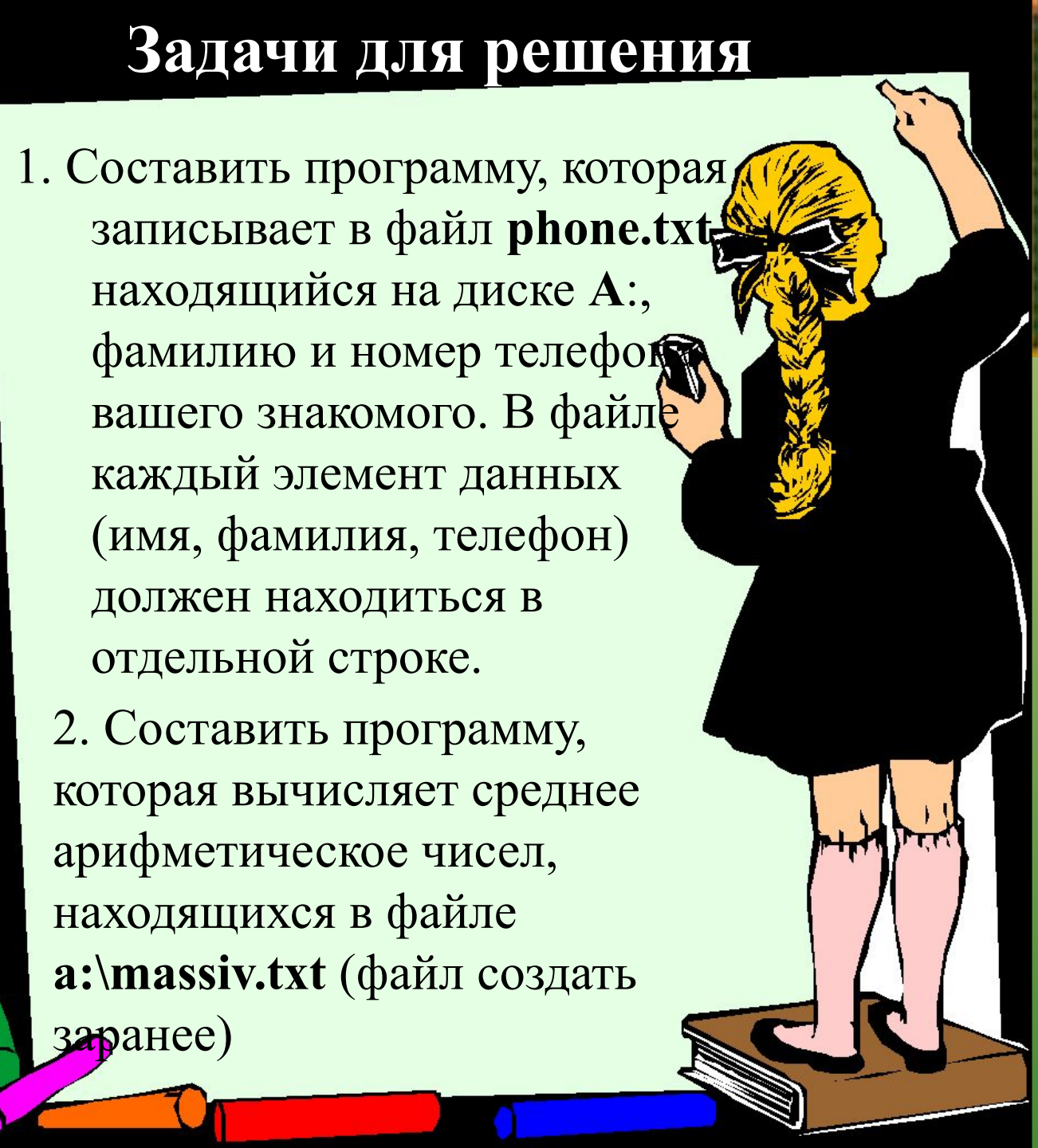
Решение задач на запись в файл и чтение из файла



Задачи для решения

1. Составить программу, которая записывает в файл **phone.txt**, находящийся на диске **A**:, фамилию и номер телефона вашего знакомого. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке.

2. Составить программу, которая вычисляет среднее арифметическое чисел, находящихся в файле **a:\massiv.txt** (файл создать заранее)



Подпрограммы, процедуры и функции

Как и в любом другом языке программирования в Pascal можно некоторые относительно самостоятельные фрагменты программы оформить в подпрограммы.

Подпрограммы имеют своё имя, к ним обращаются по имени из основной программы или другой подпрограммы, при этом подпрограммы могут иметь параметры в круглых скобках, которые определяют вариант её выполнения. Подпрограммы делятся на ***процедуры*** и ***функции***.

Любая функция похожа на программу: у неё есть заголовок, начало, тело функции, конец. В функции могут быть метки, константы, переменные, свои подпрограммы. Функция должна быть описана до того как она будет использована и помещается всегда перед основным **BEGIN**.

Если в некоторой подпрограмме нужно вычислить несколько значений или одно значение сложного типа (матрица), то нужно использовать не функции, а **процедуры**.

Процедура в чем-то похожа на функцию, также имеет имя, параметры, тело подпрограммы. Отличия от функции заключаются в описании и вызове.

Форма записи:

```
FUNCTION   f(x: тип): тип;
begin
f:= выражение
end;
```

Далее в программе к функции обращаются по f(x).

Пример:

```
FUNCTION   tan ( x : real) : real;
begin
    tan := sin(x) / cos(x)
end;
```

Форма записи процедуры:

Procedure Имя (x, y: real; var z: boolean; var r: real);

Параметры делятся на два вида:

1. Невозвращаемые (входные) (без var - x, y)
2. Возвращаемые (выходные) (с var - z, r)

Возвращаемые параметры - это результат процедуры.

Отличие процедуры и функции различаются при вызове.

Функции вызываются внутри **Writeln (имя_ функции)** или справа от оператора присваивания

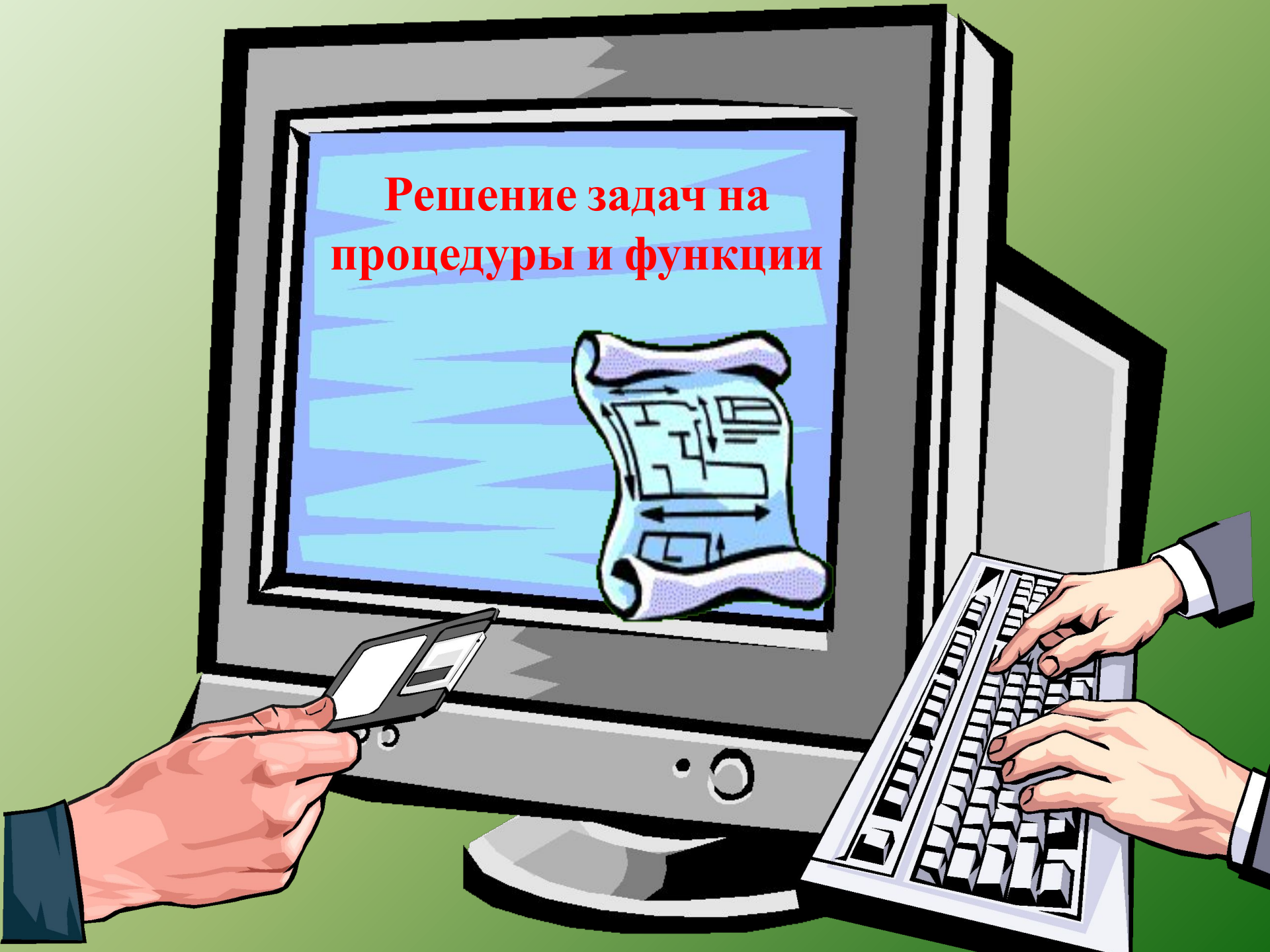
y: = имя_ функции (x);

процедура всегда вызывается отдельной строкой.

x:= 2; y:=4;

имя_ процедуры (x, y, z);

Решение задач на процедуры и функции



Пример решения задачи:

Постановка задачи:

Написать процедуру, которая выводит на экране строку, состоящую из звёздочек. Длина строки (количество звёздочек) является параметром функции.

Пример решения задачи:

Составление программы:

```
Program Star_Trek;
Uses crt;
Procedure StarLine (len: integer);
Var i: integer;
begin
      clrscr;
      for i:=1 to len do
            write (‘*’);
end;
Begin
      clrscr;
      StarLine (25);
      readln
End.
```

Пример решения задачи:

Постановка задачи:

Написать функцию, которая вычисляет значение $\mathbf{a}^{\mathbf{b}}$. Числа **a** и **b** могут быть любыми дробными положительными числами.

Степень числа вычисляется по формуле:

$$a^b = e^{b \ln(a)}$$

Пример решения задачи:

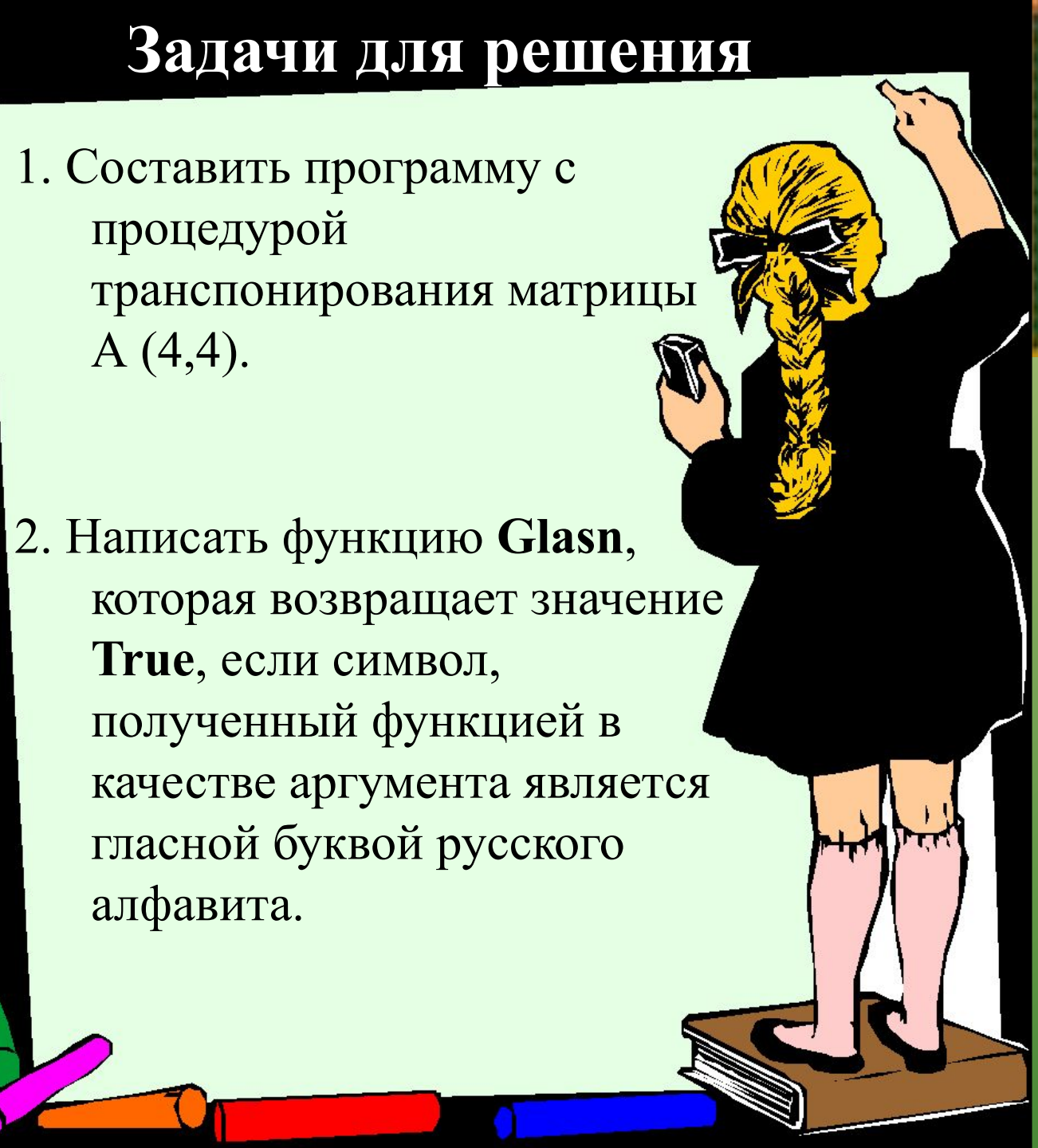
Составление программы:

```
Program Stepen;
Function InStep (a, b:real):real;
    begin
          InStep:=exp(b*ln(a));
    end;
Var a: real;    { число}
      b: real;    { степень}
      c: real;    { число в степени}
BEGIN
  writeln (' Введите число и показатель степени');
  readln (a,b);
  c:= InStep (a,b);
  writeln (a:6:3,'  в степени ', b:6:3,' = ',
c:6:3);
   readln;
END.
```

Задачи для решения

1. Составить программу с процедурой транспонирования матрицы A (4,4).

2. Написать функцию **Glasn**, которая возвращает значение **True**, если символ, полученный функцией в качестве аргумента является гласной буквой русского алфавита.



Локальные и глобальные переменные

В программе, имеющей в своём составе подпрограммы, все переменные можно разделить на две группы:

- **локальные** - **глобальные**

Локальные переменные существуют только внутри подпрограммы и считаются неизвестными вне её.

Глобальные переменные действуют как в основной программе, так и во всех её подпрограммах.

Локальные переменные - это параметры подпрограммы и переменные, описанные в разделе описания переменных этой подпрограммы.

Глобальные переменные всегда записываются в разделе описания переменных программы.

ПЕРЕМЕНА

Графика

1. Подключить модуль **uses graph**;
2. Ввести дополнительные переменные **GD** и **GM** типа integer;
3. Настроиться на тип монитора;
4. Включить графику;
5. Использовать её;
6. Выключить графику.

Пример:

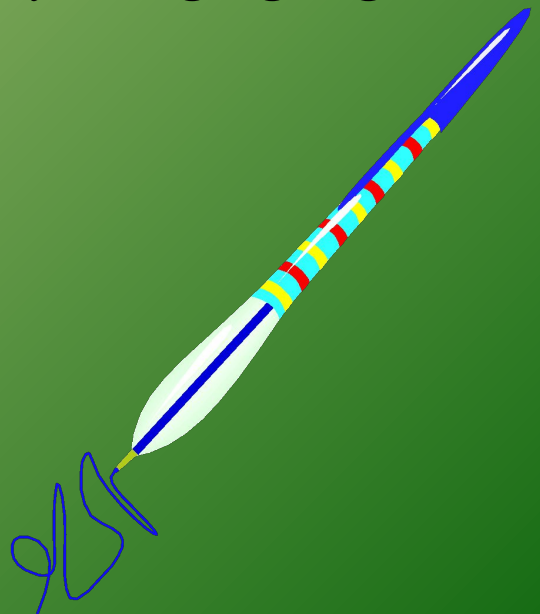
1) Uses graph;
2) Var GD, GM: integer;

…

BEGIN

3) GD:= detect;
4) Initgraph (GD,GM,’c:\prog\turbopas.60\bgi’); Путь к **egavga.bgi**
5) Circle (100,100,50)
6) Closegraph;

END.



- в графическом режиме экран представляет собой совокупность точек, каждая из которых может быть окрашена в один из 16 цветов;

- координаты точек возрастают слева направо и сверху вниз; левая верхняя точка имеет координаты (0,0), а правая нижняя – (639, 479);

- для того, чтобы программа могла выводить на экран графические примитивы (линии, окружности, прямоугольники), необходимо инициализировать графический режим.

Цвета

0 - чёрный	6 - коричневый	12 - розовый
1 - синий	7 – светло-серый	13 - малиновый
2 - зелёный	8 – тёмно-серый	14 - жёлтый
3 - голубой	9 – ярко-синий	15 - белый
4 - красный	10 – ярко-зелёный	
5 - фиолетовый	11 – ярко-голубой	

Графические функции

Точка

PutPixcel (координаты, цвет);

PutPixcel (x,y: integer, color: word);

Например: PutPixcel (100,100,1);

Отображается точка синим цветом

Графические функции

Линия

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

Координаты должны быть целыми числами

Окружность

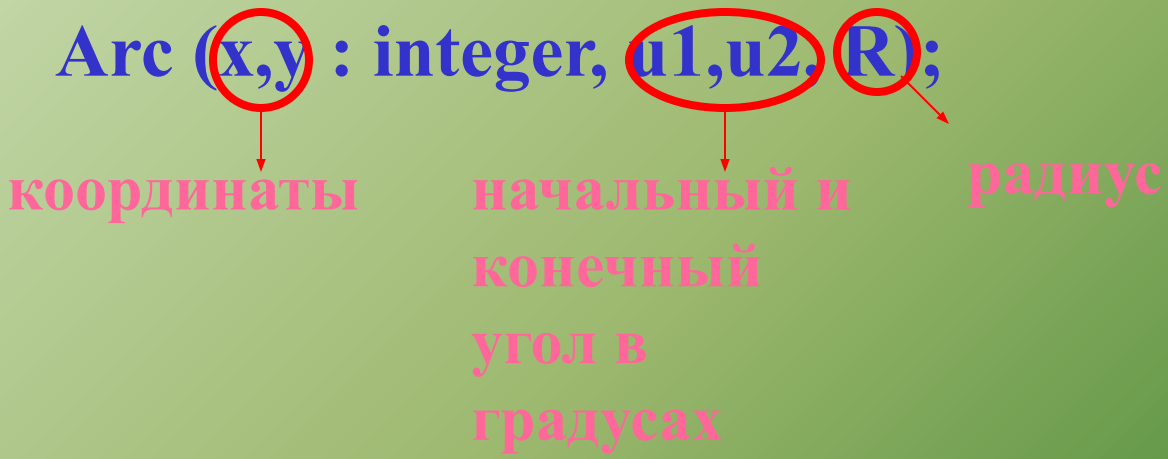
Circle (x,y: integer, R: word);

координаты **радиус**

Графические функции

Дуга

Arc (x,y : integer, u1,u2, R);



Прямоугольник

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

Графические функции

Эллипс

Ellipse (x,y, u1,u2, R1, R2);

координаты

начальный и конечный угол в градусах

Радиус по x и радиус по y

Параллелепипед

Bar3D (x1,y1, x2,y2 : integer, t: word);

координаты передней стенки

глубина в % от ширины

При необходимости можно установить тип линии.

SetLineStyle (L, P, T: word);

L – тип линии;

0 – сплошная;
1 – точечная;
2 – штрих пунктирная;
3 - пунктирная

P – цвет;
T - толщина

Управление цветом:

SetColor (color: word); Установка основного цвета

SetBkColor (color: word); Установка цвета фона

Управление текстом:

OutText (st: string);

OutTextXY (x, y: integer, st: string); Выводит текст в позицию с указанными координатами

Тип текста устанавливается процедурой

SetTextStule (Font, D, Size: word);

шрифт

направление текста

размер


Графика

Пример решения задачи:

Постановка задачи:

Составить программу, которая выводит на экран 200 окружностей разного размера и разных цветов.

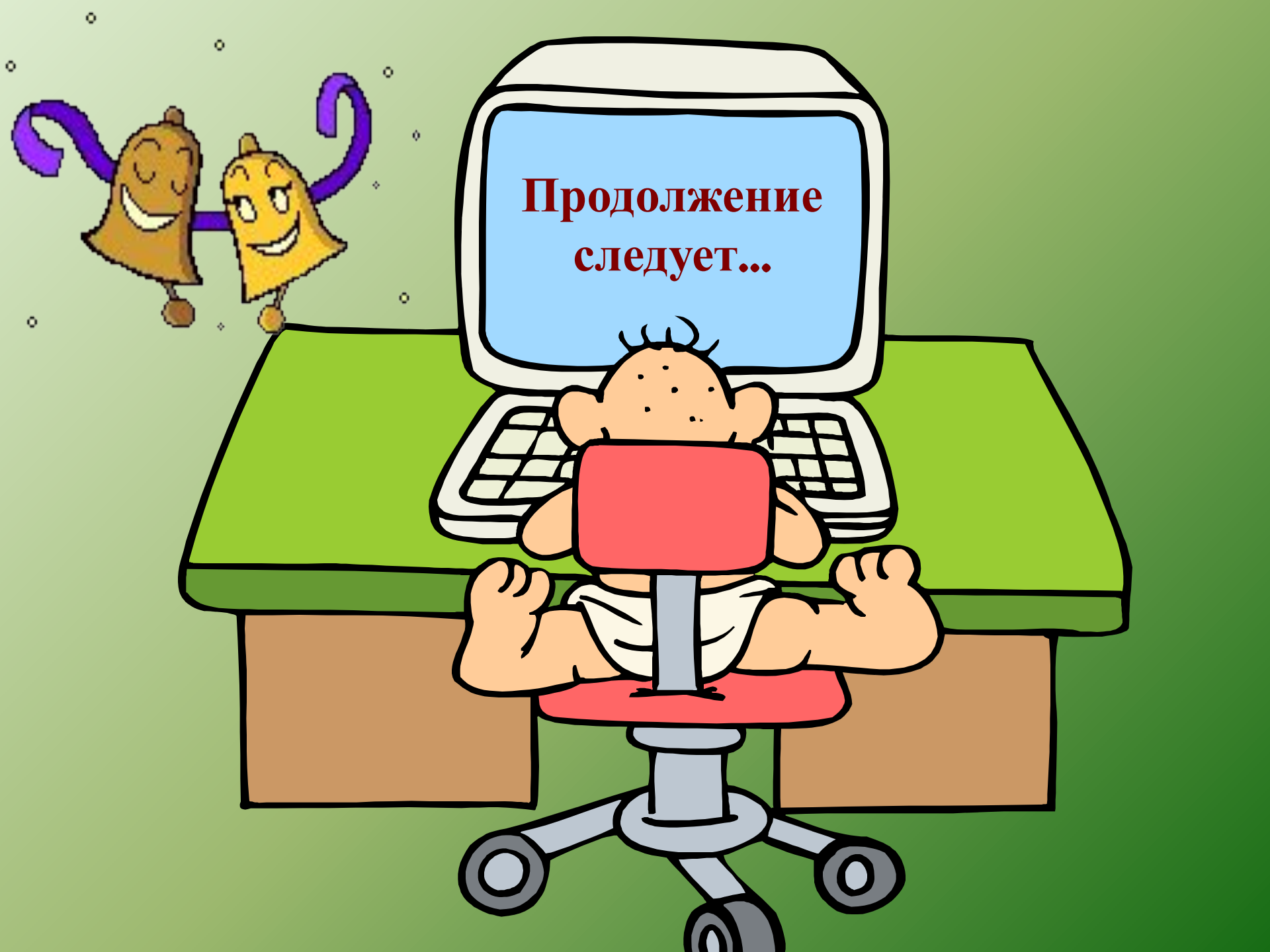
Пример решения задачи:

Составление программы:

```
Program Kover;
Uses graph,crt;
Uses crt;
Var i, GD,GM: integer;
BEGIN
    GD:= detect;
    initgraph (GD,
GM,'c:\prog\turbopas.60\bgi');
    for i:= 1 to 200 do
          begin
                SetColor (i);
                circle (300, 300, i*5);
          end;
     delay (2000);
     Readln;
     closegraph
END.
```

Задачи для решения

Составить программу, которая рисует на экране весёлую рожицу.


Продолжение
следует...