Структурное программирование на языке Паскаль

1. Теория
2. Проект
3. Графики функций
4. Точки пересечения
5. Штриховка
6. Вычисление площади
7. Оформление отчета

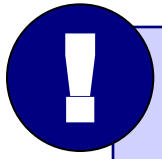
Структурное программирование на языке Паскаль

Тема 1. Теория

Этапы разработки программ

1. Постановка задачи

- определить **цель** и **категорию** программы (системная, прикладная)
- определить **исходные данные** и требуемый **результат**
- проверить, является ли задача **хорошо поставленной** (должны быть определены все связи между исходными данными и результатом)



Плохо поставленные задачи:

- не хватает исходных данных
- заданы не все связи между исходными данными и результатом
- задача не имеет решения
- задача имеет множество решений

- Зафиксировать требования к программе **в письменной форме**

Этапы разработки программ

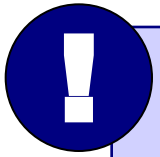
2. **Разработка модели данных**
   - формальная модель
   - типы данных (массивы, структуры, …)
   - взаимосвязь между данными
3. **Разработка алгоритма**
   - выбор существующего или разработка нового
   - возможен возврат к шагу 2
4. **Разработка программы**

   Языки: C, C++, Visual Basic, Delphi (Паскаль), `…
5. **Отладка программы** (поиск и исправление ошибок)

   debug – извлечение жучков (*bug*), 1945, МАРК-I
   - **отладчик** (точки останова, пошаговый режим, просмотр переменных)
   - **профайлер** (сколько выполняется каждая из процедур)

Этапы разработки программ

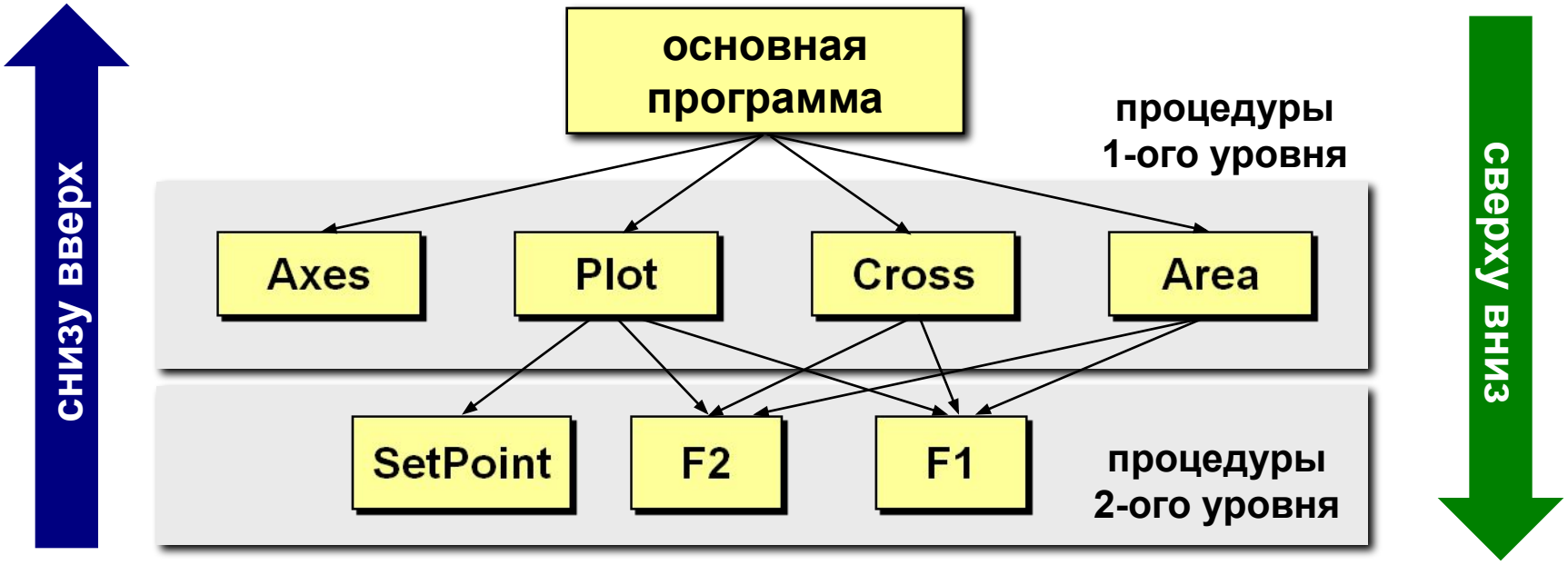
6. **Тестирование программы** (проверка на исходных данных, для которых известен результат)
   - **альфа**-тестирование: внутри фирмы (тестеры)
   - **бета**-тестирование: в других организациях, распространение через Интернет



Тестирование может показать наличие ошибок, но не их отсутствие.

7. **Разработка документации**
   - справочная система
   - руководство пользователя (*User Manual*)
   - руководство разработчика
8. **Сопровождение (техническая поддержка)**
   - исправление ошибок, найденных заказчиком
   - обучение и консультирование заказчика
   - новые версии по льготной цене

Методы проектирования программ



Проектирование «снизу вверх»

сначала составляются процедуры нижнего уровня, из которых затем «собираются» процедуры более высокого уровня.

- **легче начать** программировать
- более **эффективные** процедуры

- процедуры необходимо связывать с основной задачей («**держать в голове**»)
- при окончательной сборке может **не хватить** «кубиков»
- часто программа получается **запутанной**
- сложно распределить работу **в команде**

Проектирование «сверху вниз»

метод последовательного уточнения:

1) начинаем **с основной программы**;
2) она разбивается на подзадачи, для каждой из которых пишется процедура-«**заглушка**»;
3) реализуем каждую из процедур тем же способом.

- меньше вероятность **принципиальной ошибки** (начали с главного)
- проще **структура** программы
- удобно **распределять** работу в команде

- в разных блоках могут быть реализованы похожие операции (можно было решить одной **общей процедурой**), особенно в команде

Структурное программирование

Существовавшие проблемы:

- увеличилась **сложность** программ
- сократилось **время** на разработку

Цели:

- повысить **надежность**
- уменьшить **время** и **стоимость** разработки
- облегчить тестирование и **отладку**
- возможность **переделки** одного модуля
- улучшить **читабельность**
  - *без переходов на другую страницу*
  - *избегать трюков и запутанных приемов*

Структурное программирование

Принципы:

- **абстракции**: программу можно рассматривать на любом уровне без лишних подробностей
- **модульности**: программа разбивается на отдельные модули, которые могут отлаживаться независимо друг от друга
- **подчиненности**: связь между модулями «сверху вниз»
- **локальности**: каждый модуль использует только свои локальные переменные, глобальные переменные только в крайних случаях

Модуль

Модуль – это программный блок (процедура или функция), отделенный от кода других модулей, который полностью решает самостоятельную задачу своего уровня.

- работа модуля не зависит от того, **откуда** он вызывается, и от того, сколько раз он вызывался **до этого**
- размер модуля не более **50-60 строк** (1 страница)
- модуль имеет **один вход** и один **выход**
- модуль начинается с «шапки»-**комментария** (входные данные, результаты, какие модули использует)
- имена переменных – **смысловые**
- в одной строке – один оператор
- «трюки» – долой

Оформление текста программы

Шапка – комментарий в начале процедур и функций.

```
{----------------------------------------
  Max – максимальное из двух чисел
   Вход: a, b – исходные числа
   Выход: максимальное из a и b
----------------------------------------}
function Max(a, b: integer): integer;
begin
  ...
end;
```

Оформление текста программы

Отступы – тело цикла, условного оператора, оператора выбора и т.п. сдвигается вправо на 2-3 символа.

```
for i:=1 to n do begin j := 0; while j < i
do begin j := j + 1; k := k mod N; end; k
:= k + 1; end;
```

лесенка

```
for i:=1 to n do
begin
  j := 0;
  while j < i do
  begin
    j := j + 1;
    k := k mod N;
  end;
  k := k + 1;
```

- легче читать текст программы
- видны блоки **begin-end** (где начинаются и заканчиваются)

Скорость работы программы не меняется!

Оформление текста программы

- «говорящие» имена функций, процедур, переменных: `Sum`, `ShowMenu`, `count`, `speed`.
- пробелы в операторах

```
if(a<b)then b:=c+d;
```

```
if ( a < b ) then
  b := c + d;
```

- выделение пустыми строками и комментариями важных блоков

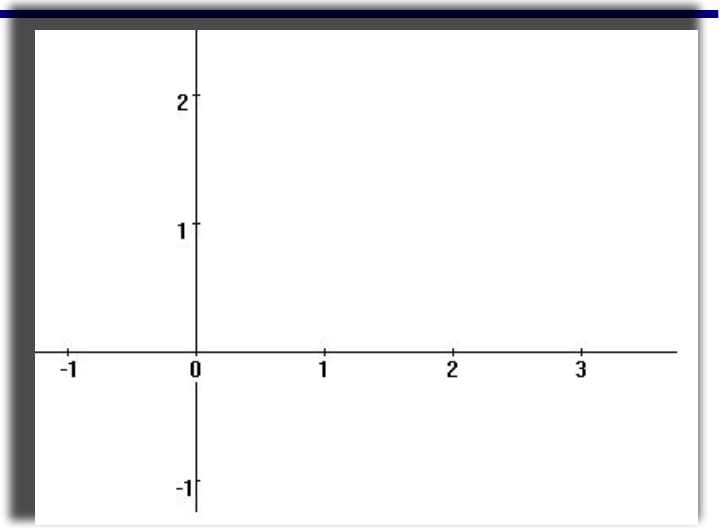
```
   { ввод данных }
writeln( 'Введите число' );
read ( n );
   { вычисления }
n2 := n*n;
   { вывод результата }
writeln ( 'Его квадрат ', n2 );
```

Структурное программирование на языке Паскаль

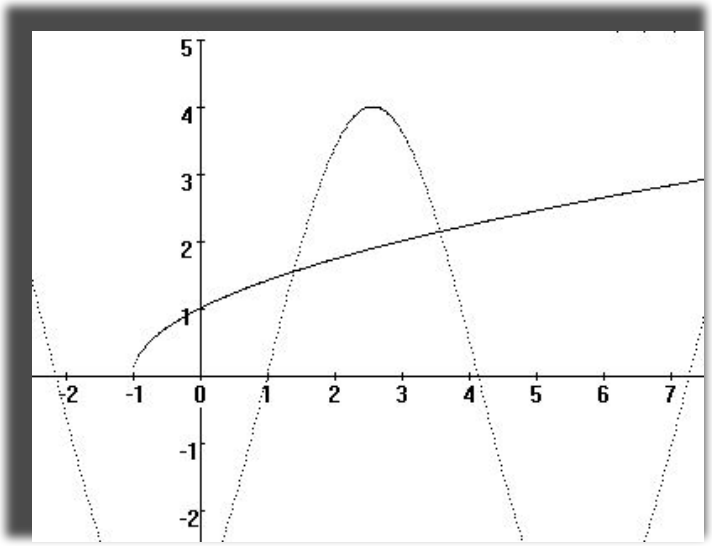
Тема 2. Проект

Проект «Графики функций»

- построить координатные **оси** и сделать их разметку

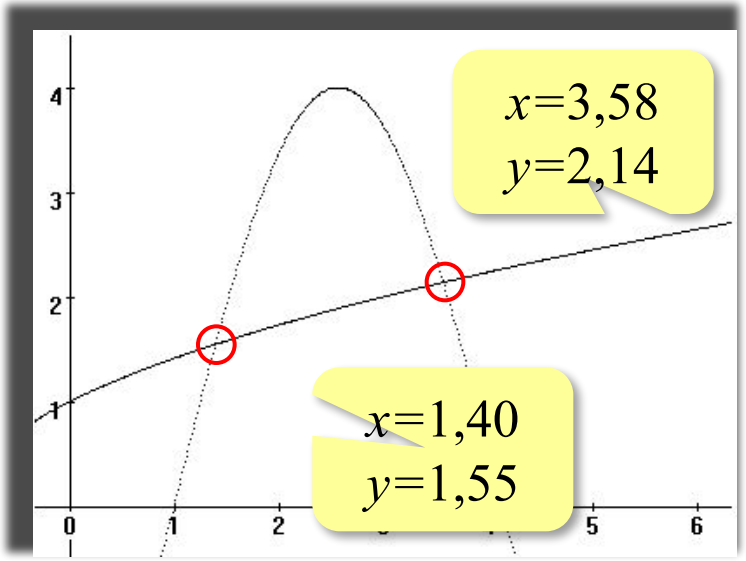


- построить **графики** заданных функций (по вариантам)

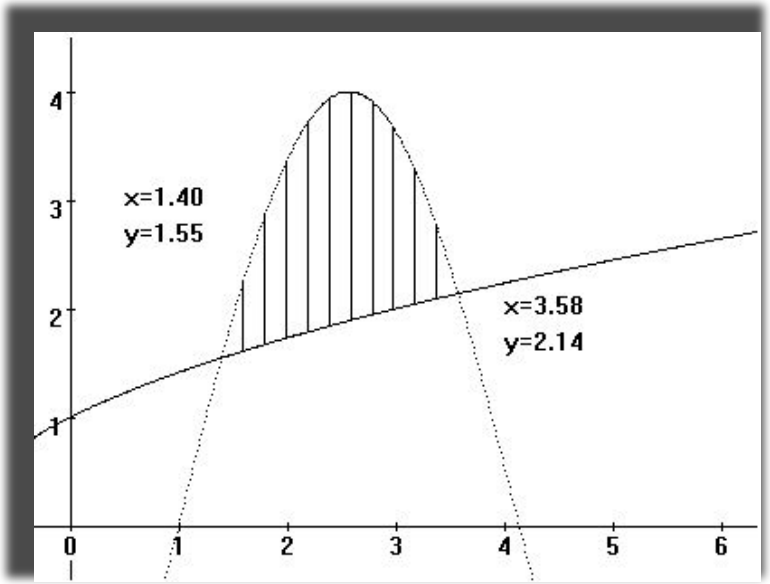


Проект «Графики функций»

- найти **точки пересечения** графиков, используя численные методы

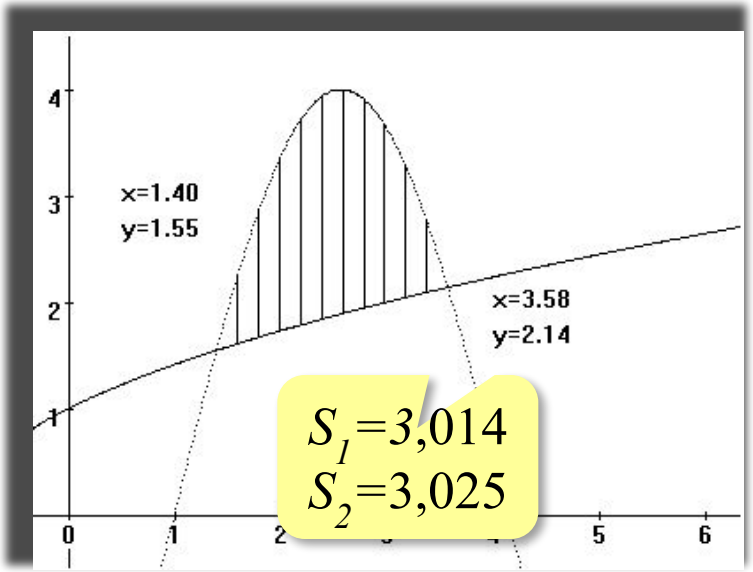


- **заштриховать** образованную замкнутую область

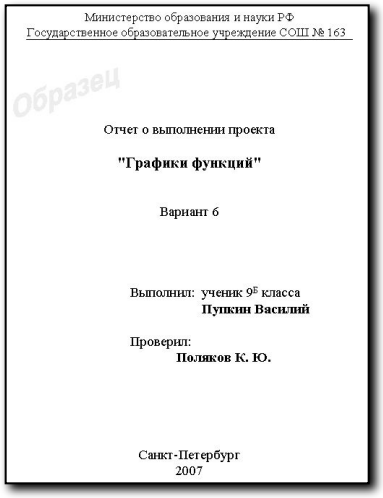


Проект «Графики функций»

- вычислить **площадь** этой области двумя способами



- оформить **отчет** по работе


Министерство образования и науки РФ
Государственное образовательное учреждение СОШ № 163

Образец

Отчет о выполнении проекта

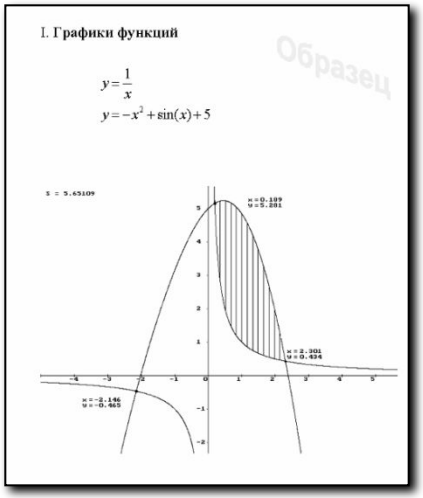
"Графики функций"

Вариант 6

Выполнил: ученик 9[Б] класса
Пупкин Василий

Проверил:
Поляков К. Ю.

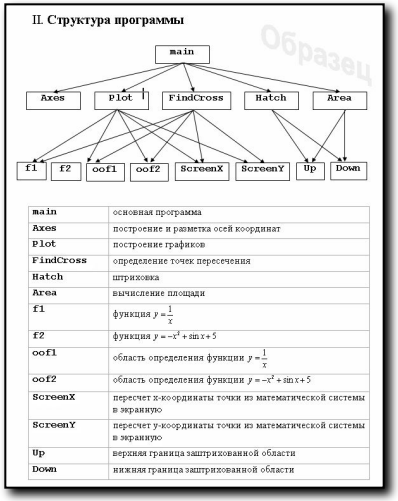
Санкт-Петербург
2007


I. **Графики функций**

Образец

$y=\frac{1}{x}$

$y=-x^2+\sin(x)+5$


II. **Структура программы**

Образец

main	основная программа
Axes	построение и разметка осей координат
Plot	построение графиков
FindCross	определение точек пересечения
Hatch	штриховка
Area	вычисление площади
f1	функция $y=\frac{1}{x}$
f2	функция $y=-x^2+\sin x+5$
oof1	область определения функции $y=\frac{1}{x}$
oof2	область определения функции $y=-x^2+\sin x+5$
ScreenX	пересчет x-координаты точки из математической системы в экранную
ScreenY	пересчет y-координаты точки из математической системы в экранную
Up	верхняя граница заштрихованной области
Down	нижняя граница заштрихованной области


III. **Текст программы**

Образец

```
program qqq;

const X0 = 100; Y0 = 400;
      k = 80;
var xc1, xc2: real;

{---------------------------------------
  ScreenX, ScreenY - переход к экранным координатам
---------------------------------------}
function ScreenX(x: real): integer;
begin
  ScreenX := round(X0 + k*x);
end;
function ScreenY(y: real): integer;
begin
  ScreenY := round(Y0 - k*y);
end;

{---------------------------------------
  Axes - оси координат
---------------------------------------}
procedure Axes;
var i, xe: integer;
begin
 line ( X0, 0, X0, 499 );
 line ( 0, Y0, 699, Y0 );
 for i:=1 to trunc((700-X0)/k) do begin
   xe := ScreenX(i);
   line ( xe, Y0-2, xe, Y0+2 );
   MoveTo(xe-4, Y0+3);
   write(i);
 end;
end;

{---------------------------------------
  SetPoint - точка графика с проверкой
---------------------------------------}
procedure SetPoint ( x, y: real; r, g, b: integer);
var xe, ye: integer;
begin
  xe := ScreenX(x);
  ye := ScreenY(y);
  if (xe >= 0)  and  (xe < 700)  and
     (ye >= 0)  and  (ye < 500) then begin
    Pen(1, r, g, b);
    Point(xe, ye);
  end;
end;

{---------------------------------------
  f1(x), f2(x) - функции
---------------------------------------}
function f1 (x: real): real;
```

Структура программы

```
program qq;
```

константы и переменные

процедуры и функции

```
begin
 Axes;   { оси координат }
 Plot;   { графики функций }
 Cross;  { точки пересечения графиков }
 Hatch;  { штриховка }
 Area;   { площадь (способ 1) }
 Area2;  { площадь (способ 2) }
end.
```

основная программа

«заглушки»

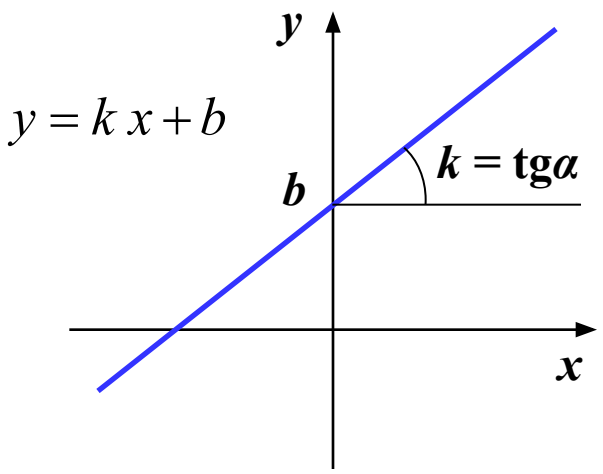
```
{-----------------------------------
  Axes: оси координат
-----------------------------------}
procedure Axes; begin end;
```

Структурное программирование на языке Паскаль

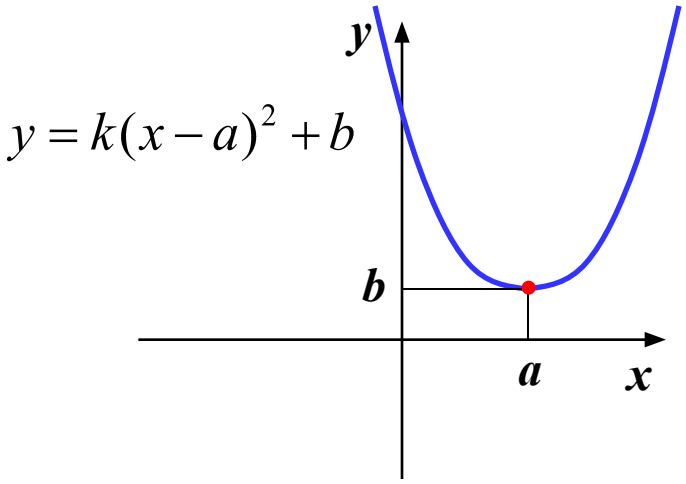
Тема 3. Графики функций

Функции, заданные в явном виде $y = f(x)$

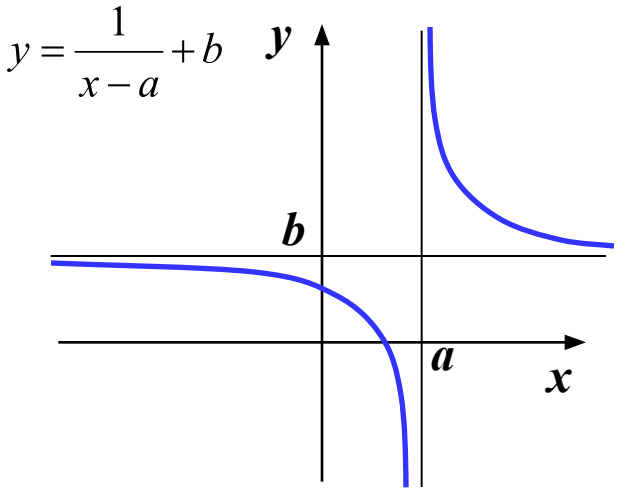
прямая



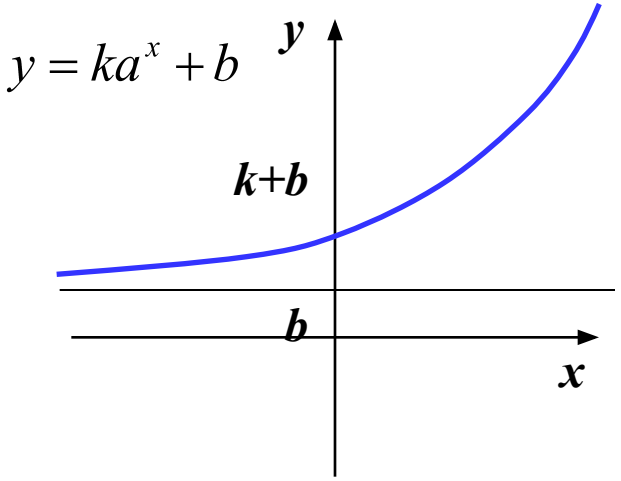
парабола



гипербола



степенная функция

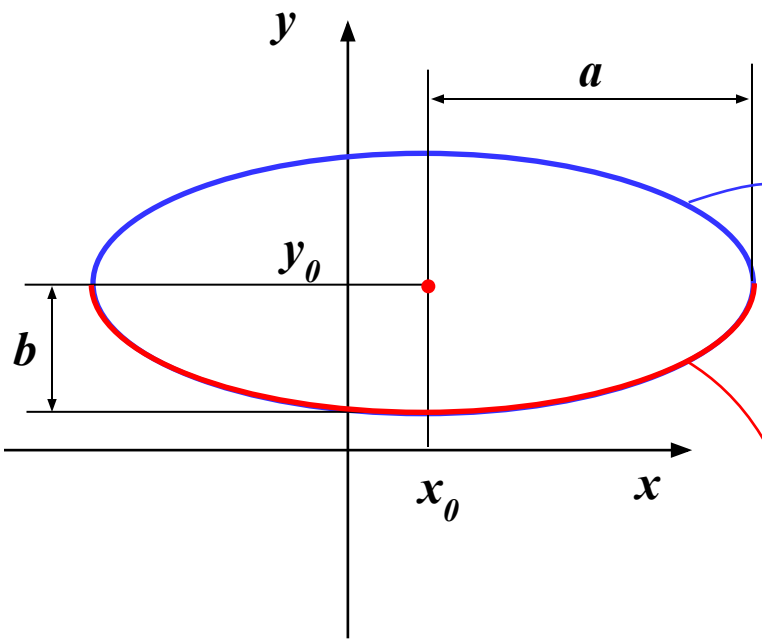


Функции, заданные в неявном виде

$f(x, y) = 0$

пример: уравнение **эллипса**

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1 \quad \Leftrightarrow \quad \frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}-1=0$$



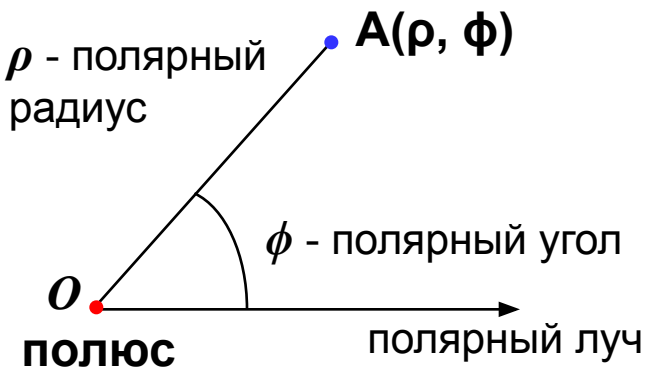
$$\frac{(y-y_0)^2}{b^2}=1-\frac{(x-x_0)^2}{a^2}$$

$$(y-y_0)^2=b^2\left(1-\frac{(x-x_0)^2}{a^2}\right)$$

$$y-y_0=\pm\, b\sqrt{1-\frac{(x-x_0)^2}{a^2}}$$

$$y=y_0\pm\, b\sqrt{1-\frac{(x-x_0)^2}{a^2}}$$

Полярные координаты

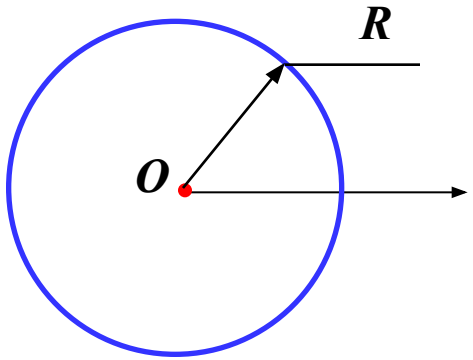


$$\rho = f(\phi)$$

Описание фигур, полученных при **вращении** объектов.

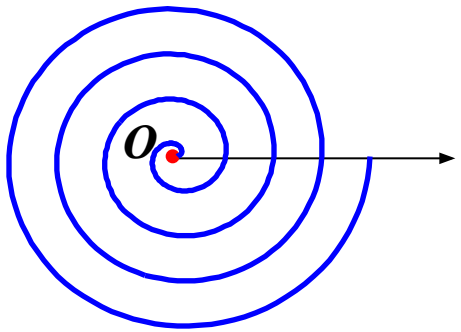
Примеры:

$$\rho = R$$



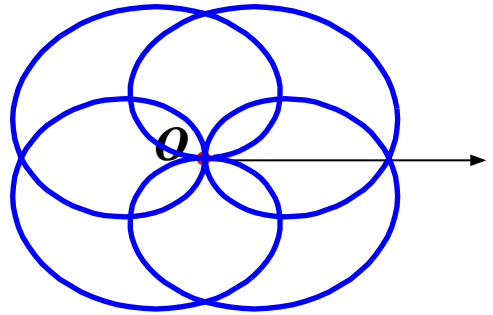
окружность

$$\rho = a \cdot \phi$$



спираль Архимеда

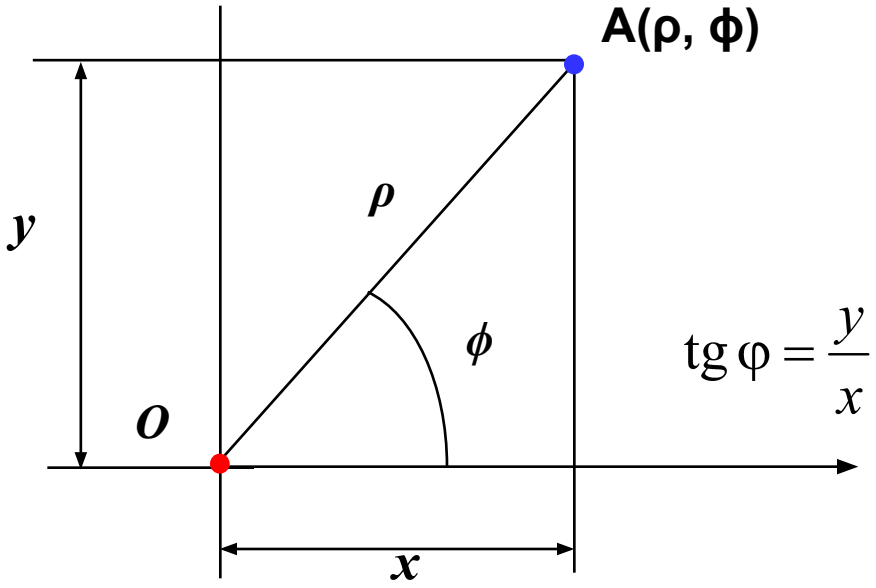
$$\rho = a \cdot \sin(2\phi/3)$$



«роза»

Полярные координаты

Переход к декартовым координатам



$$x = \rho \cdot \cos(\varphi) \qquad \rho = \sqrt{x^2 + y^2}$$

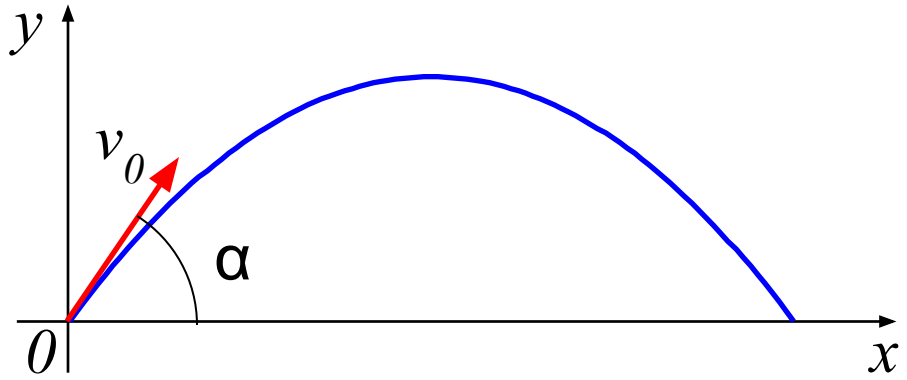
$$y = \rho \cdot \sin(\varphi) \qquad \varphi = \operatorname{arctg}\frac{y}{x}$$

Описание в параметрической форме

$$x = f_1(t)$$
$$y = f_2(t)$$

t – независимый параметр («время»)

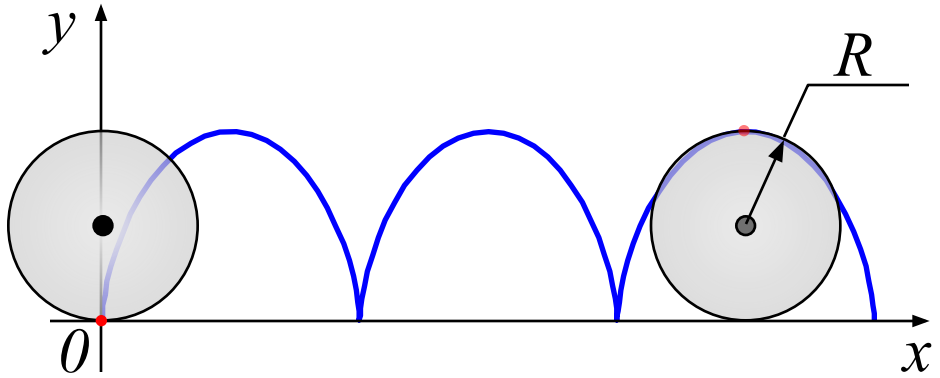
Описание фигур, полученных при сложном **движении** объектов.



$$x = v_0 \cdot \cos(\alpha) \cdot t$$

$$y = v_0 \cdot \sin(\alpha) \cdot t - \frac{gt^2}{2}$$

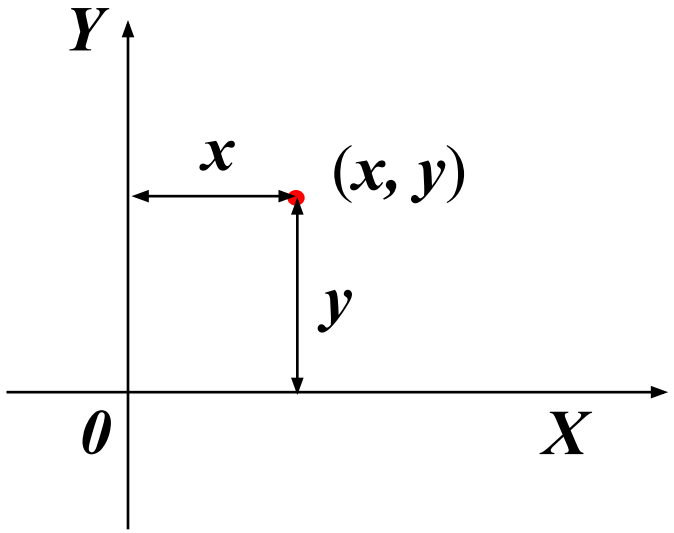
Циклоида – траектория точки на ободе колеса при вращении



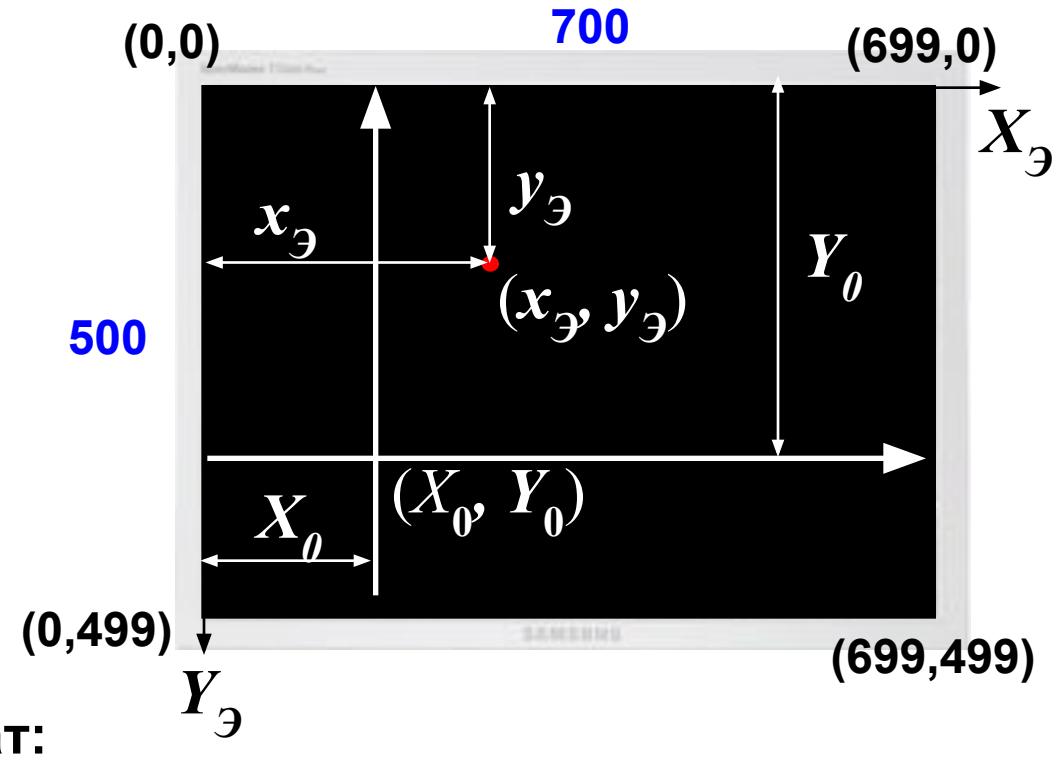
$$x = R \cdot (t - \sin t)$$
$$y = R \cdot (1 - \cos t)$$

Системы координат

Математическая



Экранная



Преобразование координат:

X_0, Y_0 – экранные координаты точки (0,0)

k – **масштаб** (во сколько раз растягивается единичный отрезок)

$$x_Э = X_0 + k \cdot x$$

$$y_Э = Y_0 - k \cdot y$$

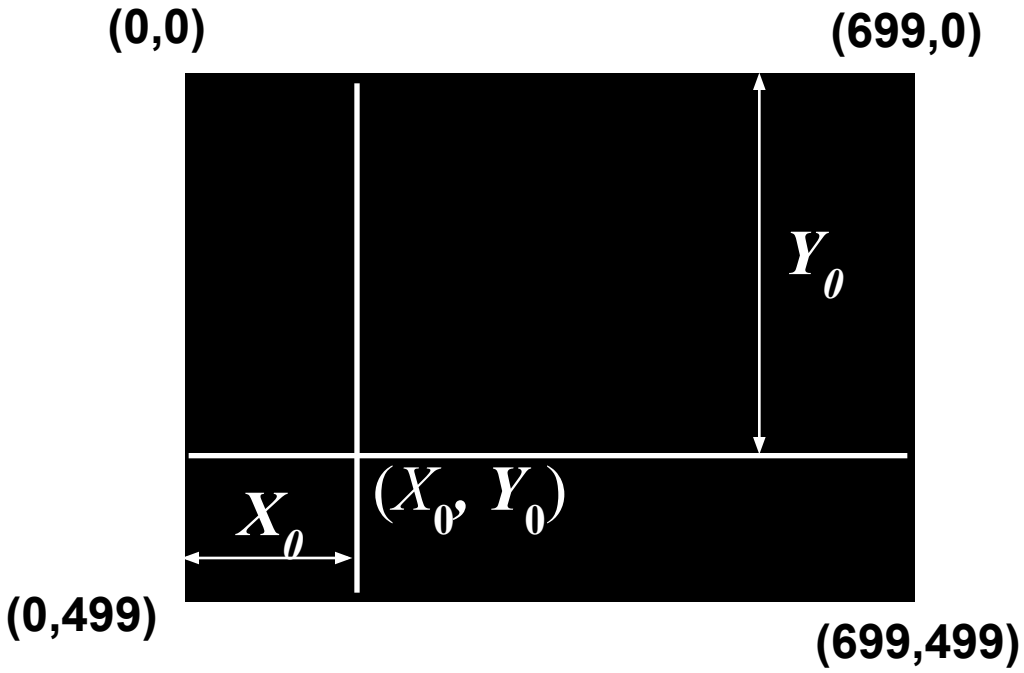
? **Почему «минус»?**

Перевод в экранные координаты

```
const X0 = 100; Y0 = 400;  { начало координат }
      k = 80;              { масштаб }
{-----------------------------------------
 ScreenX - перевод X в координаты экрана
-----------------------------------------}
function ScreenX(x: real): integer;
begin
  ScreenX := round(X0 + k*x);
end;
{-----------------------------------------
 ScreenY - перевод Y в координаты экрана
-----------------------------------------}
function ScreenY(y: real): integer;
begin
  ScreenY := round(Y0 - k*y);
end;
```

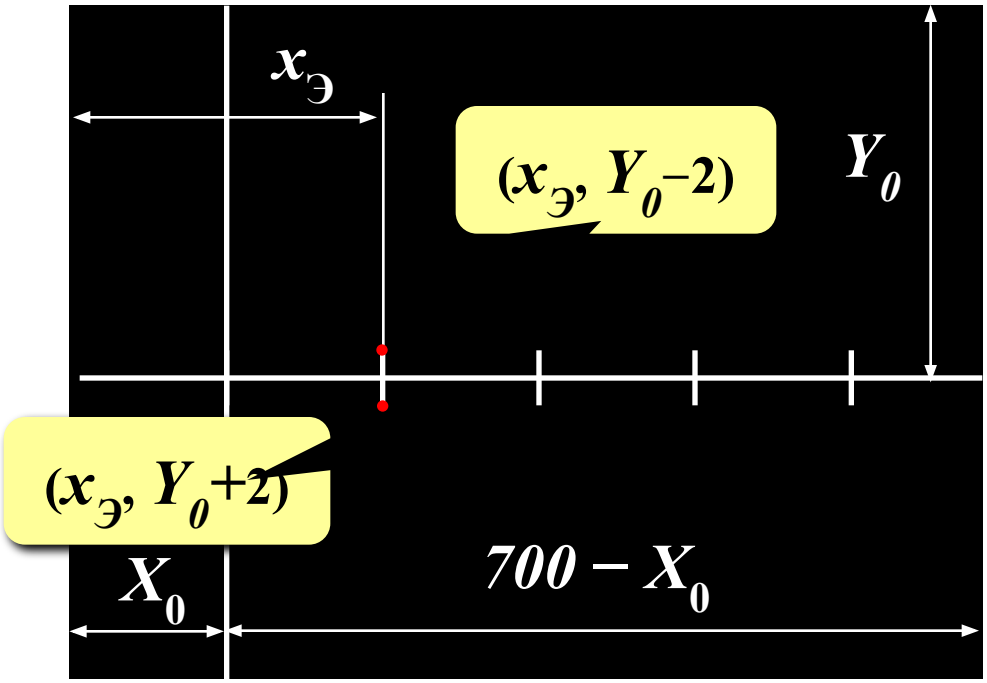
Откуда берутся X_0, Y_0 и k?

Оси координат




```
procedure Axes;
begin
 line ( X0, 0, X0, 499 );
 line ( 0, Y0, 699, Y0 );
end;
```

Разметка оси X («черточки»)



Число меток на $[0, x_{max}]$:

длина $700 - X_0$

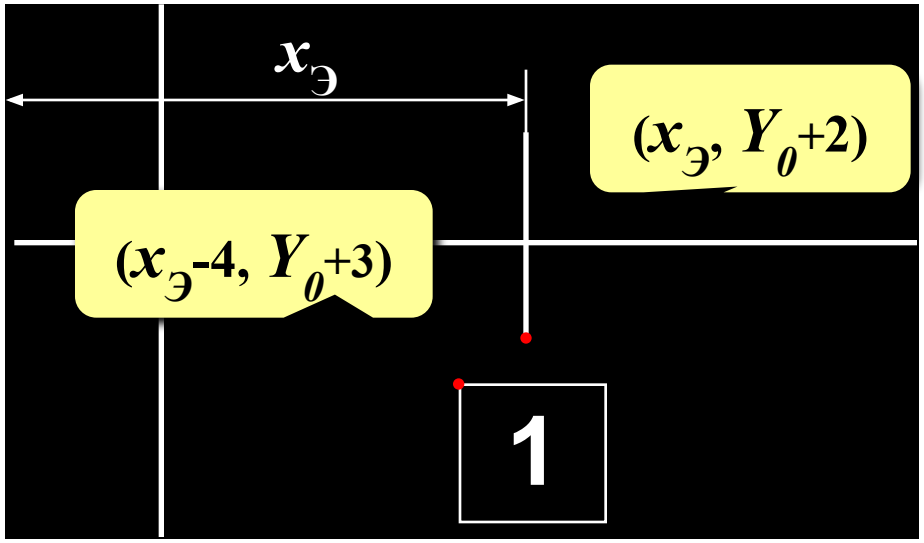
единичный отрезок k

$$N = \frac{700 - X_0}{k}$$

trunc – отбросить дробную часть

```
var i, xe: integer;
…
for i:=1 to trunc((700-X0)/k) do begin
  xe := ScreenX(i);
  line ( xe, Y0-2, xe, Y0+2 );
end;
```

Разметка оси X (числа)



Вывод в нужной точке:

```
MoveTo ( x, y );
write ( i );
```

левый верхний угол

```
var i, xe: integer;
…
for i:=1 to trunc((700-X0)/k) do begin
  xe := ScreenX(i);
  line ( xe, Y0-2, xe, Y0+2 );
  MoveTo(xe-4, Y0+3);
  write(i);
end;
```

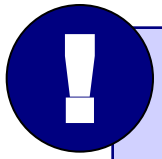
Оси с разметкой (полностью)

```
procedure Axes;
var i, xe: integer;
begin
 line ( X0, 0, X0, 499 );
 line ( 0, Y0, 699, Y0 );
 for i:=1 to trunc((700-X0)/k) do begin
   xe := ScreenX(i);
   line ( xe, Y0-2, xe, Y0+2 );
   MoveTo(xe-4, Y0+3);
   write(i);
 end;
end;
```

Задания

«4»: Сделать разметку осей полностью (не только положительной части оси X).

«5»: Сделать задание на «4», использовав только 2 цикла (1 цикл для каждой оси).

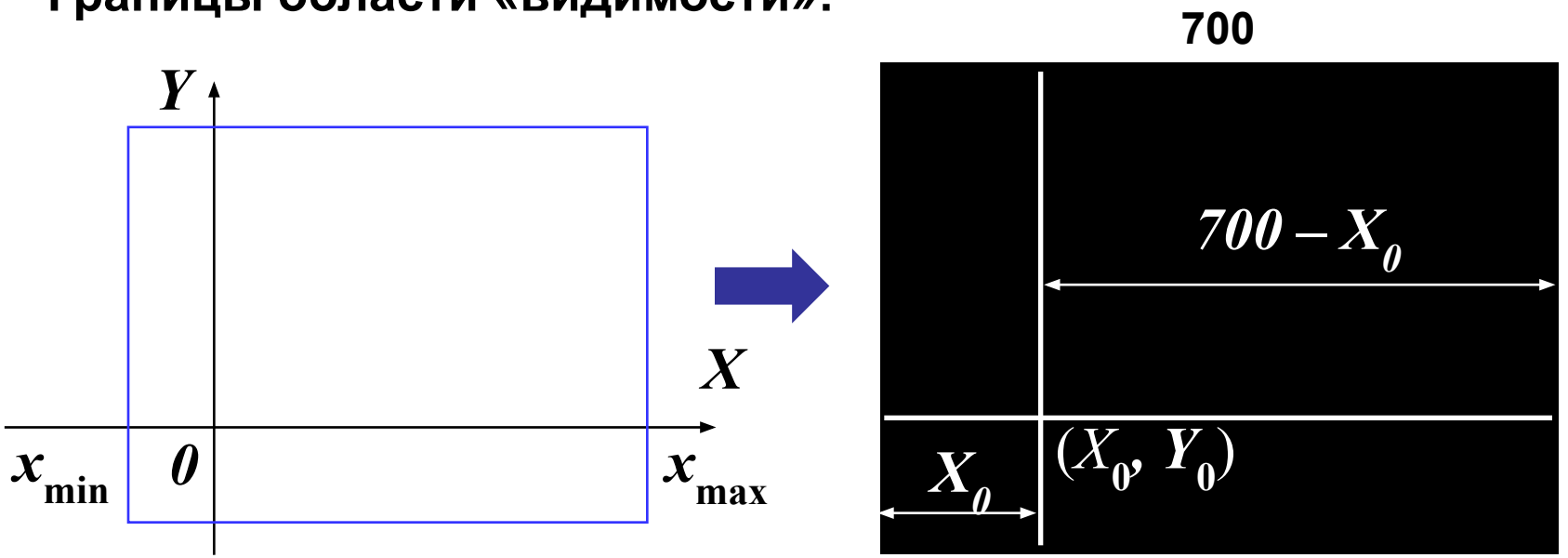


Разметка должна работать правильно при любых значениях X_0 и Y_0.

Построение графика по точкам

- Нельзя рисовать за границами экрана.
- Область определения функции (деление на ноль, корень из отрицательного числа, …)!

Границы области «видимости»:



$$x_{\min} = -\frac{X_0}{k}, \qquad x_{\max} = \frac{700 - X_0}{k}$$

Вывод точки с проверкой

```
{-------------------------------------
   SetPoint - вывод пикселя с проверкой
    и пересчетом координат
-------------------------------------}
procedure SetPoint ( x, y: real;
                     r, g, b: integer);
var xe, ye: integer;
begin
  xe := ScreenX(x);
  ye := ScreenY(y);
  if (xe >= 0)  and  (xe < 700)  and
     (ye >= 0)  and  (ye < 500) then begin
    Pen(1, r, g, b);
    Point(xe, ye);
  end;
end;
```

координаты в математической системе

цвет точки

если точка $(x_э, y_э)$ в пределах экрана…

Описание функций

```
{-----------------------------------
  F1, F2
   Вход: x
   Выход: y = f1(x), f2(x)
-----------------------------------}
function f1 (x: real): real;
begin
  f1 := sqrt(x+1);
end;
function f2 (x: real): real;
begin
  f2 := 4*sin(x-1);
end;
```

$$f_1(x) = \sqrt{x+1}$$

$$f_2(x) = 4\sin(x-1)$$

Области определения

Для $f_1(x) = \sqrt{x+1}$

```
{-------------------------------------
  ODZ1 – область определения f1(x)
   Вход:  x
   Выход: True, если x входит в ОДЗ
          False, если x не входит в ОДЗ
-------------------------------------}
function odz1(x: real): Boolean;
begin
  odz1 := (x >= -1);
end;
```

Для $f_2(x) = 4\sin(x-1)$ не нужно!

Вывод графика функции

```
{-------------------------------------
   PLOT вывод графиков функций
-------------------------------------}
procedure Plot;
var xmin, xmax, x, h: real;
begin
   xmin := - X0 / k;
   xmax := (700 - X0) / k;
   h := (xmax - xmin) / 1000;
   x := xmin;
   while x <= xmax do begin
     if odz1(x) then
       SetPoint(x, f1(x), 255, 0, 0);
     x := x + h;
   end;
end;
```

границы видимой части

шаг по x

Почему не for?

Что плохо?

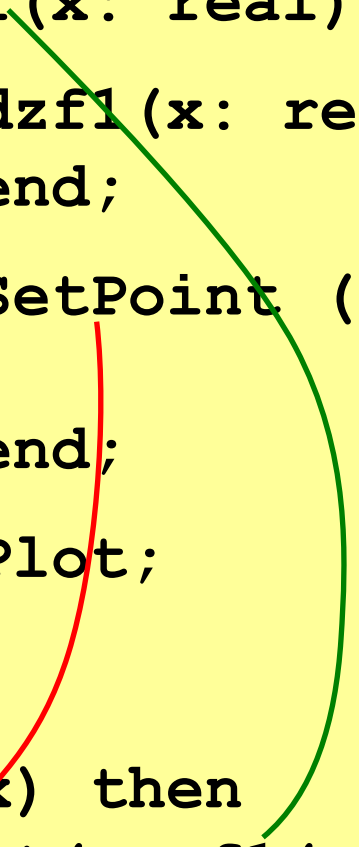
Общее расположение

```
function f1(x: real): real; begin ... end;

function odzf1(x: real): Boolean;
begin ... end;

procedure SetPoint ( x, y: real;
                     r, g, b: integer);
begin ... end;

procedure Plot;
begin
  ...
  if odz1(x) then
    SetPoint(x, f1(x), 255, 0, 0);
 ...
end;
```



Задания

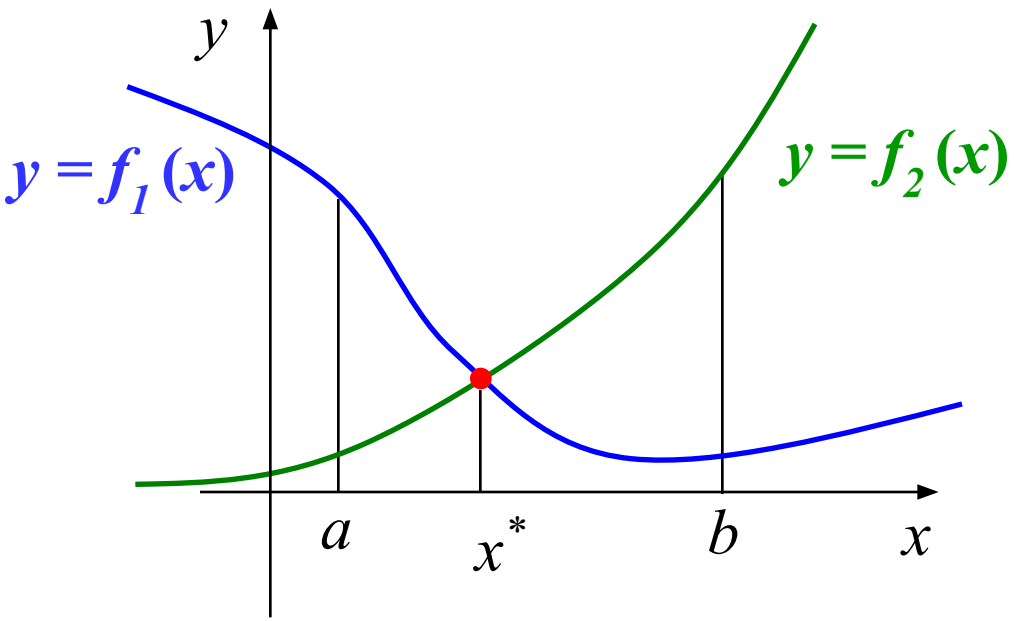
«4»: Построить графики в соответствии с заданием.

«5»: Построить графики, соединив точки линиями.

Структурное программирование на языке Паскаль

Тема 4. Точки пересечения

Точки пересечения



Точка пересечения:

$$f_1(x^*) = f_2(x^*)$$

$$f_1(x^*) - f_2(x^*) = 0$$

$$f(x^*) = 0$$

Пример:

$$f_1(x) = \sqrt{x+1}$$

$$f_2(x) = 4\sin(x-1)$$

$$\sqrt{x+1} - 4\sin(x-1) = 0$$

Проблема:

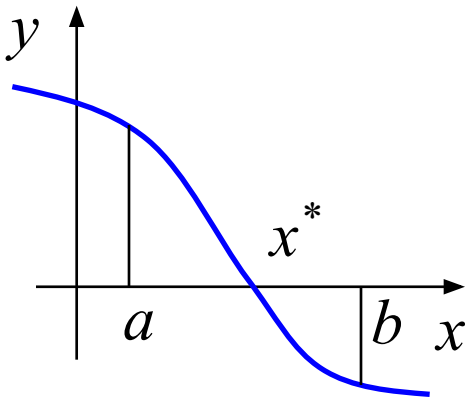
уравнение сложно (или невозможно) решить аналитически (получить формулу для x^*)

Методы решения уравнений $f(x) = 0$

- **Точные (аналитические)**

$$\sin x = 0 \quad \Rightarrow \quad x = \pi k,\ k \in Z$$

- **Приближенные**
  - **графические**



  - **численные**

(методы последовательного приближения):

1) по графику найти интервал $[a, b]$, в котором находится x^* (или одно **начальное приближение** x_0)
2) по некоторому алгоритму уточнить решение, сужая интервал, в котором находится x^*
3) повторять шаг 2, пока не достигнута требуемая точность:

$$b - a < \varepsilon$$

Численные методы

Применение: используются тогда, когда точное (аналитическое) решение неизвестно или очень трудоемко.

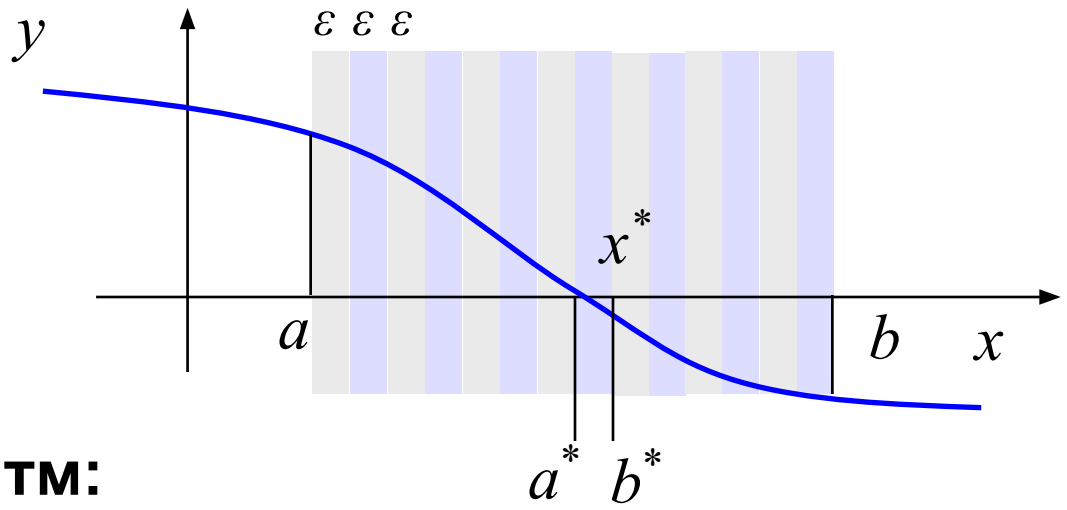
- дают хотя бы какое-то **решение**
- во многих случаях можно оценить ошибку и найти решение **с заданной точностью**

- решение всегда приближенное, **неточное**

$$\sqrt{x+1} - 4\sin(x-1) = 0 \quad \Rightarrow \quad x = 1,3974 \qquad x \approx 1,3974$$

Метод прямого («тупого») перебора

Задача: найти решение уравнения $f(x) = 0$ на интервале $[a, b]$ с заданной точностью ε (чтобы найденное решение отличалось от истинного не более, чем на ε).



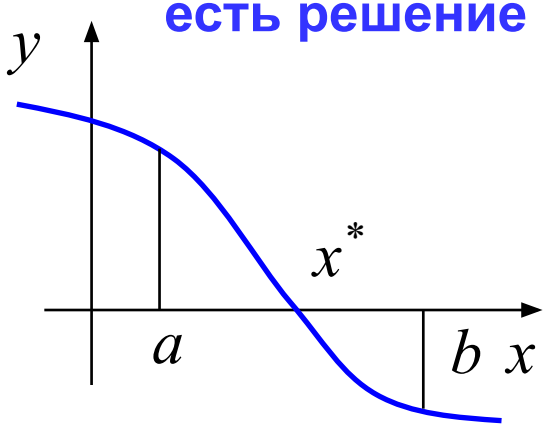
Алгоритм:

- разбить интервал $[a, b]$ на полосы шириной ε
- найти полосу $[a^*, b^*]$, в которой находится x^*
- решение – a^* или b^*

Как улучшить решение?

Есть ли решение на $[a, b]$?

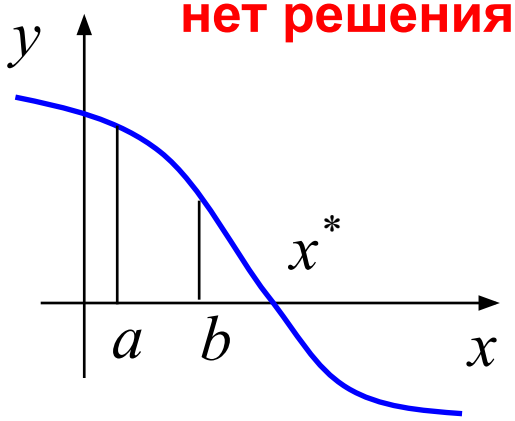
есть решение



$$f(a) > 0$$
$$f(b) < 0$$

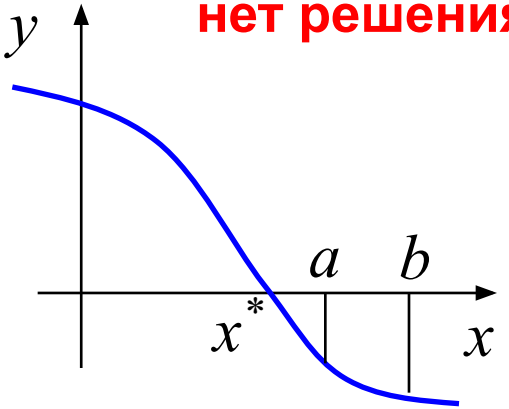
$$f(a)f(b) < 0$$

нет решения



$$f(a) > 0$$
$$f(b) > 0$$

нет решения

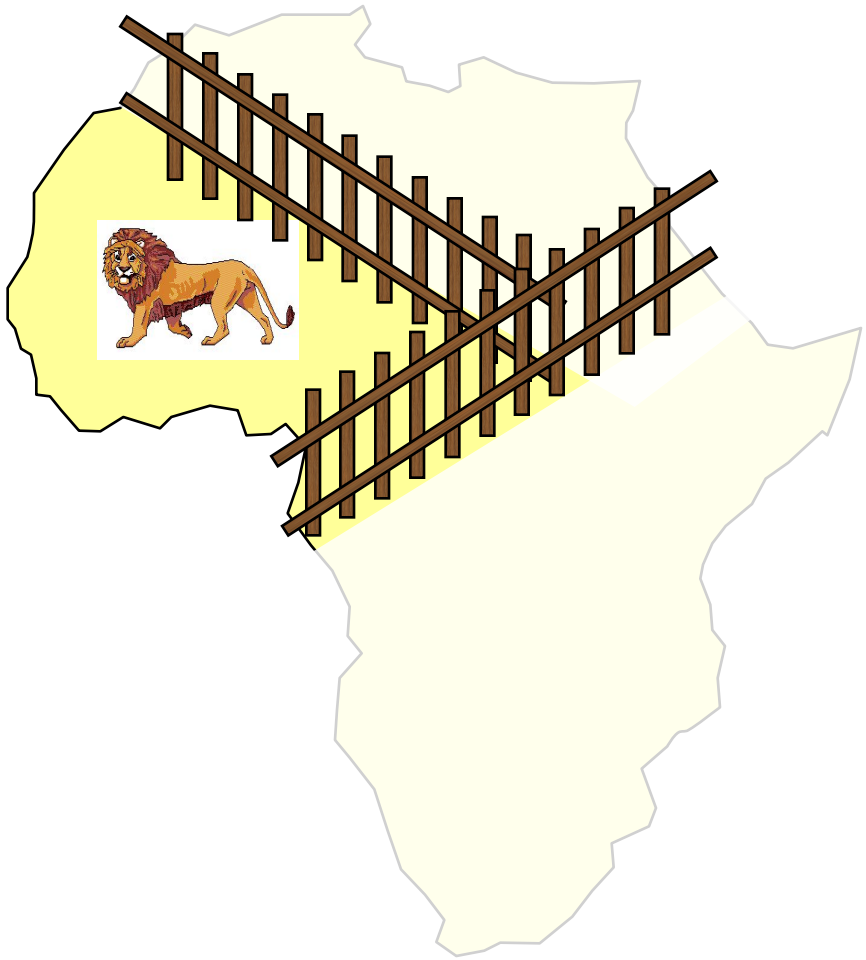


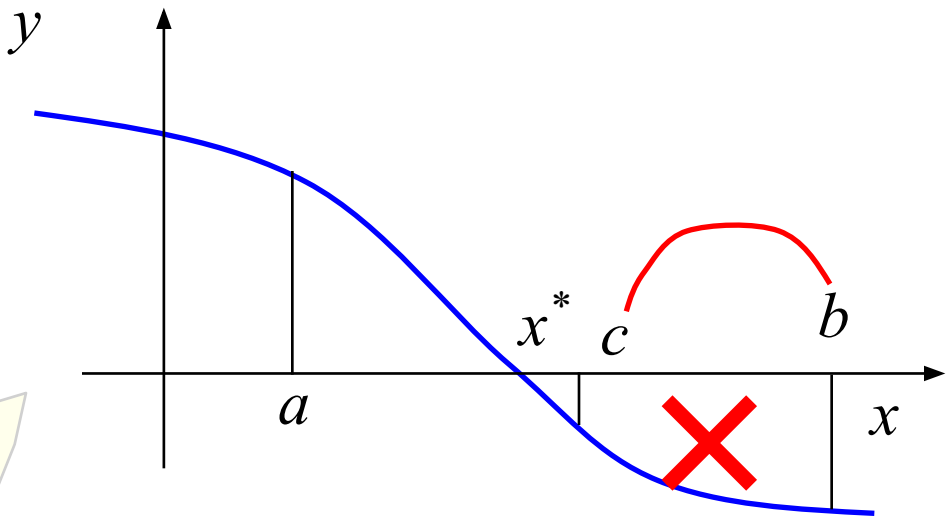
$$f(a) < 0$$
$$f(b) < 0$$

$$f(a)f(b) > 0$$

Если *непрерывная* функция $f(x)$ имеет разные знаки на концах интервала $[a, b]$, то в некоторой точке x^* внутри $[a, b]$ она равна 0, то есть $f(x^*) = 0$!

Метод дихотомии (деление пополам)





1. Найти середину отрезка `[a,b]`:
   ```
   c = (a + b) / 2;
   ```
2. Если `f(c)*f(a)<0`, сдвинуть правую границу интервала
   ```
   b = c;
   ```
3. Если `f(c)*f(a)≥0`, сдвинуть левую границу интервала
   ```
   a = c;
   ```
4. Повторять шаги 1-3, пока не будет `b – a ≤ ε`.

Метод дихотомии (деления пополам)

- **простота**
- можно получить решение с **любой** заданной **точностью**

- нужно знать **интервал** $[a, b]$
- на интервале $[a, b]$ должно быть только **одно** решение
- **большое число шагов** для достижения высокой точности
- только для функций **одной** переменной

Метод дихотомии (в программе)

```
{-----------------------------------------
 Solve находит точку пересечения на [a,b]
  Вход:  a, b – границы интервала,  a < b
         eps  - точность решения
  Выход: x – решение уравнения f1(x)=f2(x)
-----------------------------------------}
function Solve(a, b, eps: real): real;
var c, fa, fc: real;
begin
  while b - a > eps do begin
    c := (a + b) / 2;
    fa := f1(a) - f2(a);
    fc := f1(c) - f2(c);
    if  fa*fc < 0  then b := c
    else                a := c;
  end;
  Solve := (a + b) / 2;
end;
```

$$f(a) = f_1(a) - f_2(a)$$
$$f(c) = f_1(c) - f_2(c)$$

Метод дихотомии (в программе)

```
var xc1, xc2: real; { глобальные переменные }
                    { абсциссы точек пересечения
}
...
function Solve(a, b, eps: real):
begin... end;

procedure Cross;
be
   xc1 := Solve ( 1, 2, 0.0001 );
   xc1 := Solve(1, 2, 0.0001);
   MoveTo(150, 220);
   write(xc1:5:2);
   MoveTo(150, 240);
   write(f1(xc1):5:2);
   ...
end;
```

найти решение на интервале [1,2] с точностью 0,0001

вывод на экран значения x …

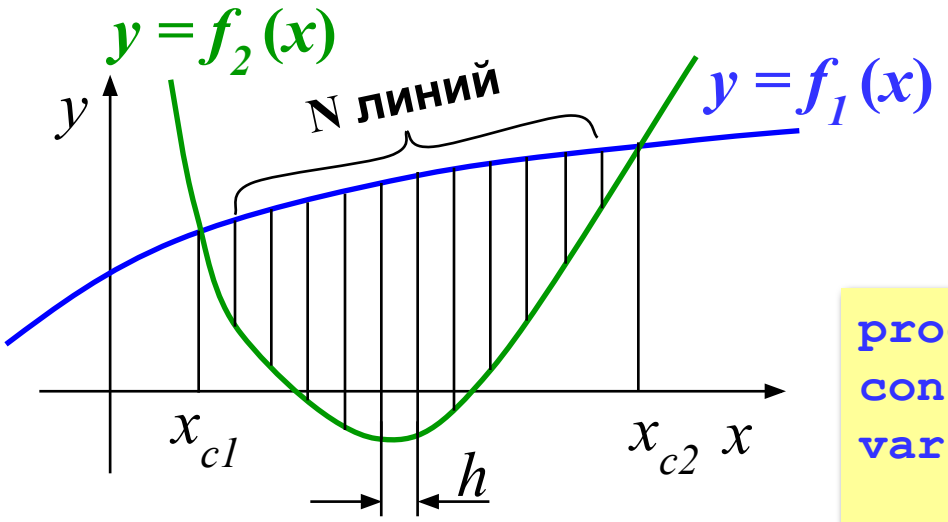
… и значения y!

то же самое для остальных точек

Структурное программирование на языке Паскаль

Тема 5. Штриховка

Штриховка (две функции)



$$h = \frac{x_{c2} - x_{c1}}{N+1}$$

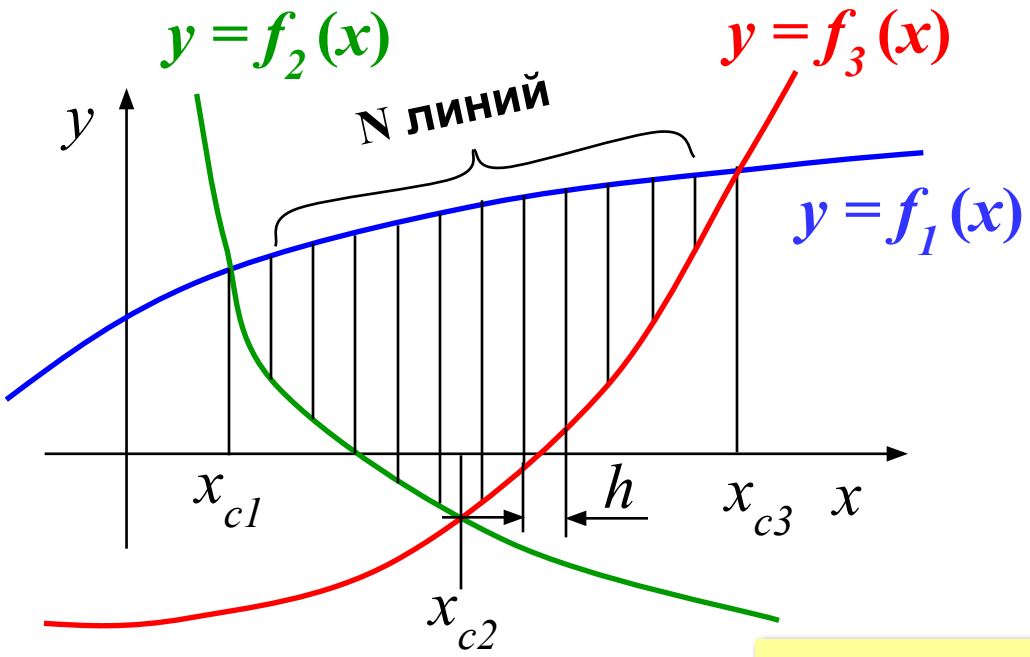
шаг по x

экранная координата x

экранные координаты границ области по оси y

```
procedure Hatch;
const N = 10;
var xe, yUp, yDown: integer;
    x, h: real;
begin
 h := (xc2 - xc1) / (N + 1);
 x := xc1 + h;
 while x < xc2 do begin
   xe := ScreenX ( x );
   yUp := ScreenY ( f1(x) );
   yDown := ScreenY ( f2(x) );
   line ( xe, yUp, xe, yDown );
   x := x + h;
 end;
end;
```

Штриховка (составная нижняя граница)



$$h = \frac{x_{c3} - x_{c1}}{N+1}$$

```
procedure Hatch;
begin
...
h = ( xc3 - xc1) / (N + 1);
...
yDown := ScreenY( FDown(x)
);
...
end;
```

```
(----------------------------
  FDown нижняя граница области
----------------------------}
function FDown(x: real): real;
begin
  if x < xc2 then
       Fdown := f2(x)
  else Fdown := f3(x);
end;
```

Штриховка (общий случай)

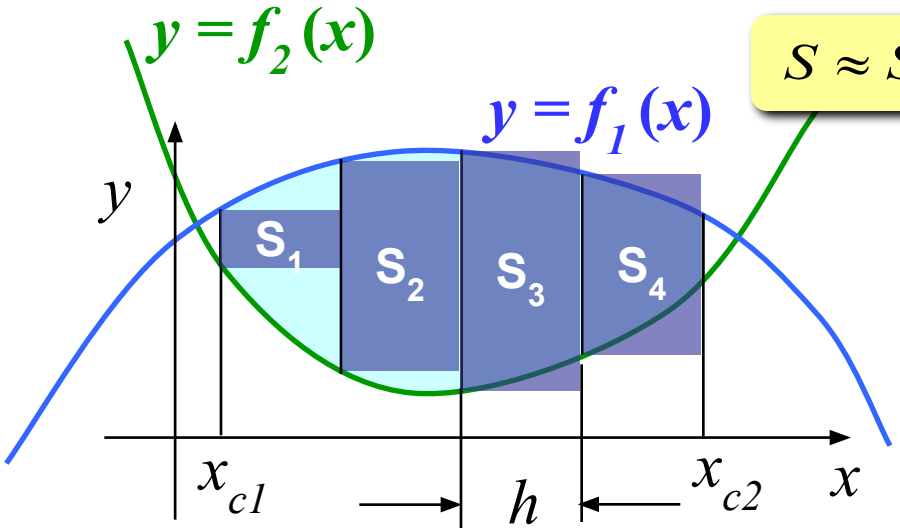
```
function FUp (x: real): real;
...
function FDown (x: real): real;
...
procedure Hatch;
...
 h := ( xc3 - xc1 ) / (N + 1);
 x := xc1 + h;
 while x < xc3 do begin
   xe := ScreenX ( x );
   yUp := ScreenY ( FUp(x) );
   yDown := ScreenY ( FDown(x) );
   line ( xe, yUp, xe, yDown );
   x := x + h;
 end;
end;
```

у всех по-разному…

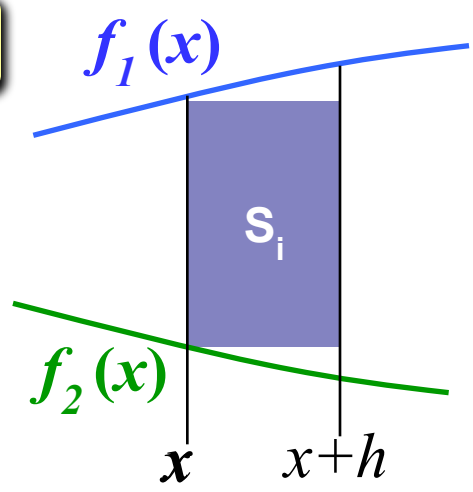
Структурное программирование на языке Паскаль

Тема 6. Вычисление площади

Метод (левых) прямоугольников



$$S \approx S_1 + S_2 + S_3 + S_4$$



$$S_i = (f_1(x) - f_2(x)) \cdot h$$

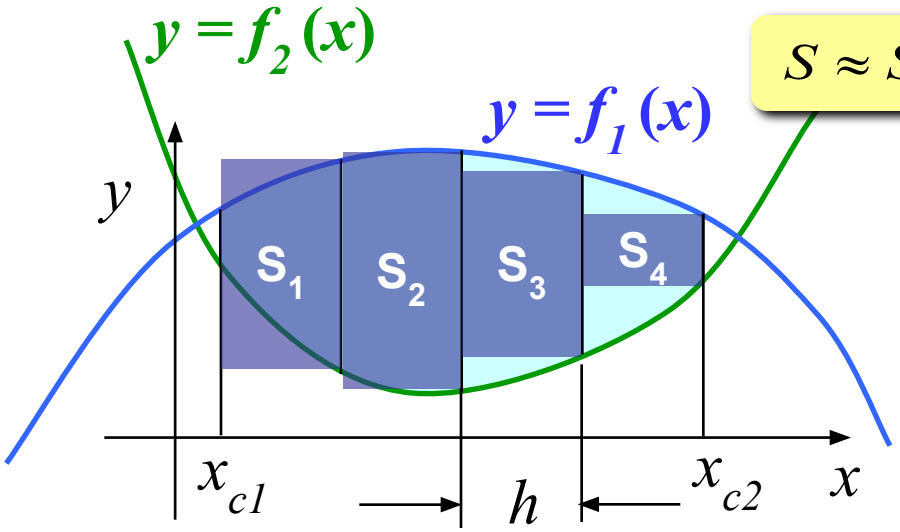
```
procedure Area;
var x, S, h: real;
begin
  S := 0; h := 0.001; x := xc1;
  while x < xc2 do begin
    S := S + f1(x) - f2(x);
    x := x + h;
  end;
  MoveTo ( 250, 320 );
  write ( 'S = ', S:0:2 );
end;
```

```
S := S * h;
```

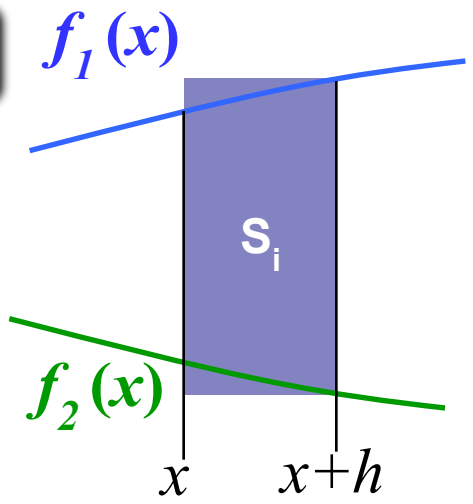
Почему не x <= xc2?

Как улучшить решение?

Метод (правых) прямоугольников



$$S \approx S_1 + S_2 + S_3 + S_4$$

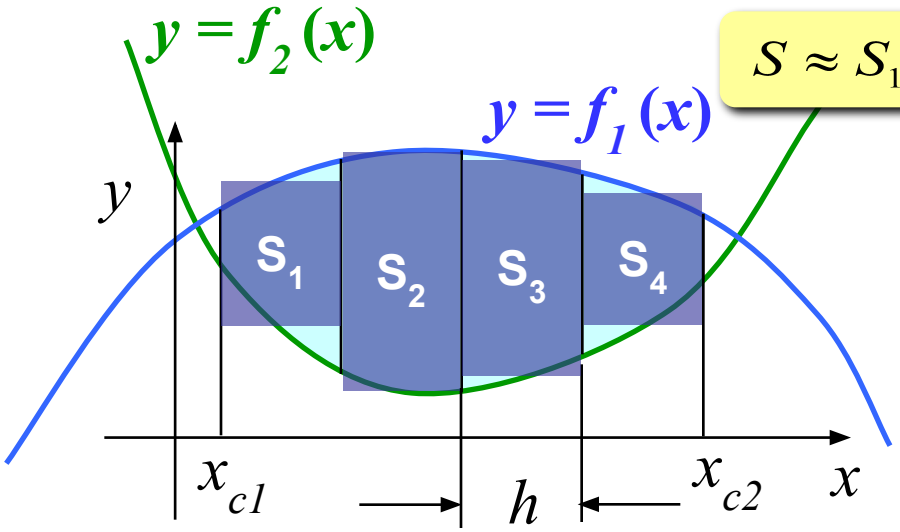


$$S_i = (f_1(x+h) - f_2(x+h)) \cdot h$$

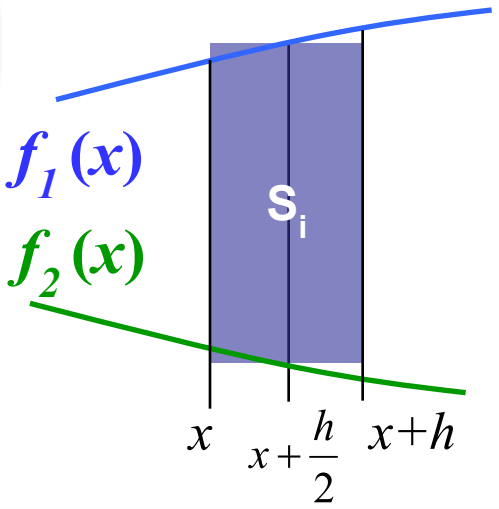
```
procedure Area;
var x, S, h: real;
begin
  S := 0; h := 0.001; x := xc1;
  while x < xc2 do begin
    S := S + f1(x+h) - f2(x+h);
    x := x + h;
  end;
  MoveTo ( 250, 320 );
  write ( 'S = ', S:0:2 );
end;
```

```
S := S * h;
```

Метод (средних) прямоугольников



$$S \approx S_1 + S_2 + S_3 + S_4$$




```
procedure Area;
var x, S, h: real;
begin
  S := 0; h := 0.001; x := xc1;
  while x < xc2 do begin
    S := S + f1(x+h/2) - f2(x+h/2);
    x := x + h;
  end;
  MoveTo ( 250, 320 ); S := S * h;
  write ( 'S = ', S:0:2 );
end;
```

$$S_i = \left[f_1(x + \frac{h}{2}) - f_2(x + \frac{h}{2}) \right] \cdot h$$

Какой метод точнее?

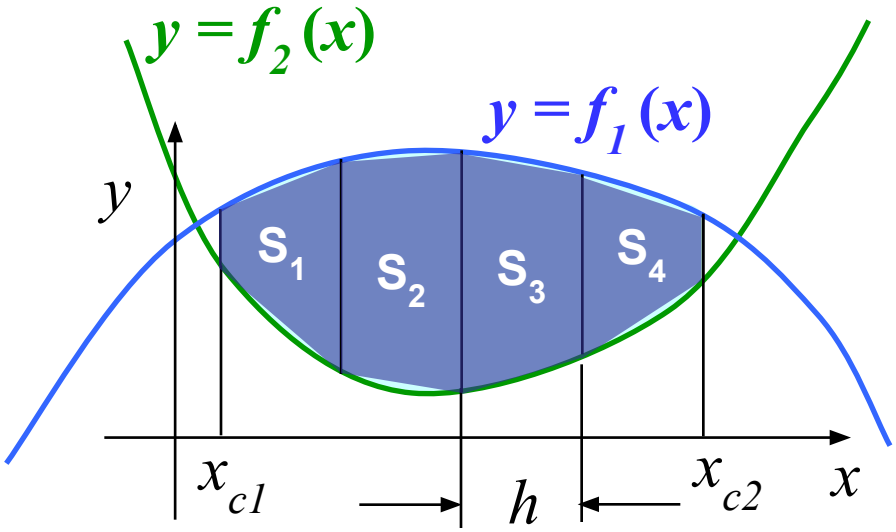
левые (правые):

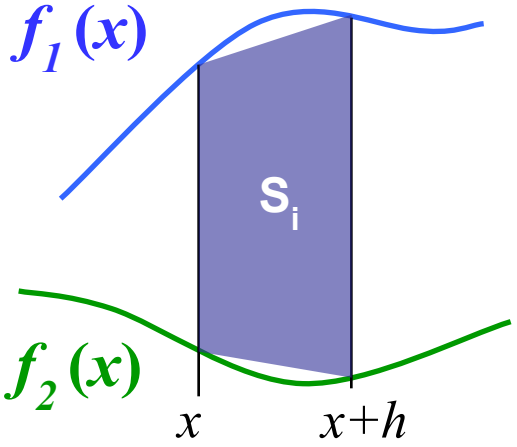
$$\varepsilon = O(h)$$

средние

$$\varepsilon = O(h^2)$$

Метод трапеций





$$S_i = \frac{f_1(x) - f_2(x) + f_1(x+h) - f_2(x+h)}{2} \cdot h$$

```
x = xc1;
v
```

```
S:=( f1(xc1)-f2(xc1)+f1(xc2)-f2(xc2) )/2;
x := xc1 + h;
while x < xc2 do begin
  S := S + f1(x) - f2(x);
  x := x + h;
end;
S := S*h;
```

Как улучшить?

Ошибка

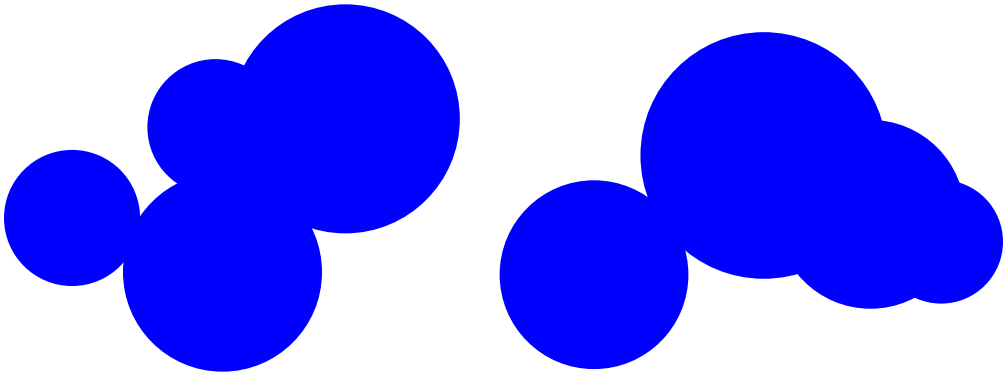
$$\varepsilon = O(h^2)$$

Метод Монте-Карло

Применение: вычисление площадей сложных фигур (трудно применить другие методы).

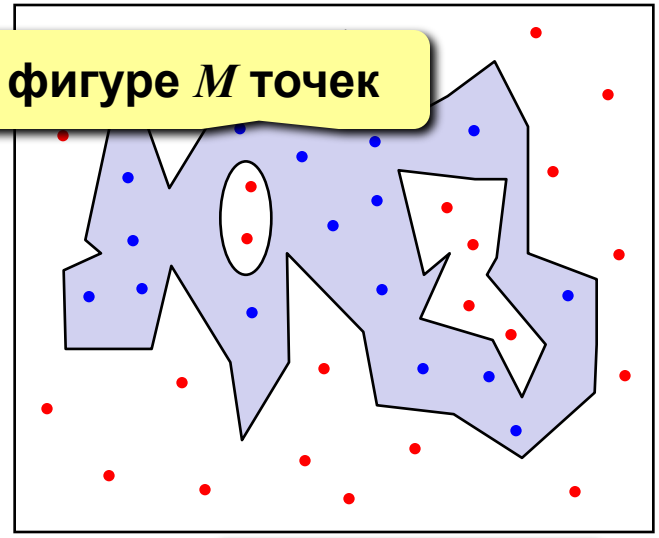
Требования: необходимо уметь достаточно просто определять, попала ли точка (x, y) внутрь фигуры.

Пример: заданы 100 кругов (координаты центра, радиусы), которые могу пересекаться. Найти площадь области, перекрытой кругами.



Метод Монте-Карло

1. Вписываем сложную фигуру **в другую фигуру**, для которой легко вычислить площадь (прямоугольник, круг, …).
2. **Равномерно** N точек со случайными координатами внутри прямоугольника.
3. Подсчитываем количество точек, **попавших на фигуру**: M.
4. Вычисляем **площадь**: $\frac{S}{S_0} \approx \frac{M}{N} \Rightarrow S \approx S_0 \cdot \frac{M}{N}$



1. **Метод приближенный.**
2. **Распределение должно быть равномерным.**
3. **Чем больше точек, тем точнее.**
4. **Точность ограничена датчиком случайных чисел.**

Случайное число в заданном интервале

```
random                  [0,1)
(b-a)*random            [0,b-a)
(b-a)*random + a        [a,b)
```

```
{-------------------------------------
  rand – случайное вещественное число
         в заданном интервале
-------------------------------------}
function rand(a, b: real): real;
begin
  rand := (b-a)*random + a;
end;
```

Проверка точки (внутри или нет?)

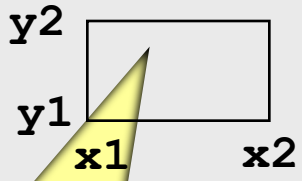
```
{----------------------------------------
  Inside - определяет, находится ли точка
           внутри фигуры
   Вход:  x, y - координаты точки
   Выход: True, если точка внутри фигуры,
          False, если точка вне фигуры
----------------------------------------}
function Inside(x, y: real): Boolean;
begin
  if (FDown(x) <= y) and (y <= FUp(x)) then
       Inside := True
  else Inside := False;
end;
```

```
function Inside(x, y: real): Boolean;
begin
  Inside := (FDown(x) <= y) and (y <= FUp(x));
end;
```

Метод Монте-Карло (реализация)

```
{-----------------------------------------------
  Area2 - вычисление площади методом Монте-Карло
-----------------------------------------------}
procedure Area2;
var i, N, M: integer;
    x1, x2, y1, y2, x, y, S: real;
begin
  N := 200000;  M := 0;
  x1 := xc1; x2 := xc2; y1 := 1; y2 := 4;
  for i:=1 to N do begin
    x := rand ( x1, x2 );
    y := rand ( y1, y2 );
    if Inside(x,y) then  M := M + 1;
  end;
  S := (x2-x1)*(y2-y1)*M/N;
  Moveto(250, 340);
  write('S = ', S:0:2);
end;
```

границы прямоугольника (у каждого свои!)



если на фигуре, увеличить счетчик

вычисление площади

Структурное программирование на языке Паскаль

Тема 7. Оформление отчета

Титульный лист

Министерство образования и науки РФ
Государственное образовательное учреждение СОШ № 163

Отчет о выполнении проекта

"Графики функций"

Вариант 6

Выполнил: ученик 9Б класса
Пупкин Василий

Проверил:
Поляков К. Ю.

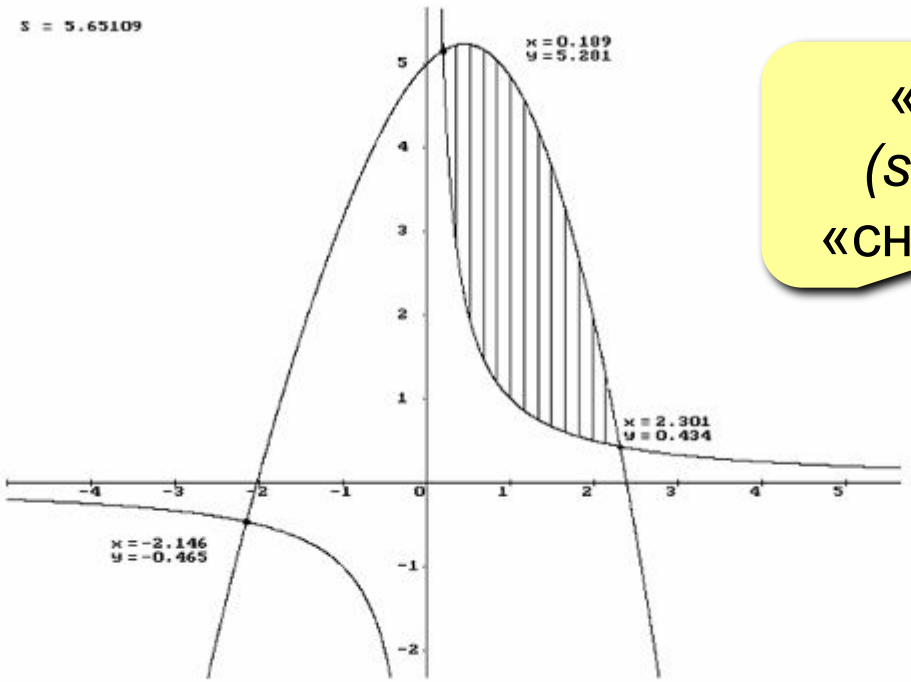
Санкт-Петербург
2007

Графики функций

I. Графики функций

$$y = \frac{1}{x}$$

$$y = -x^2 + \sin(x) + 5$$



через Редактор формул (**Вставка – Объект – Microsoft Equation**)

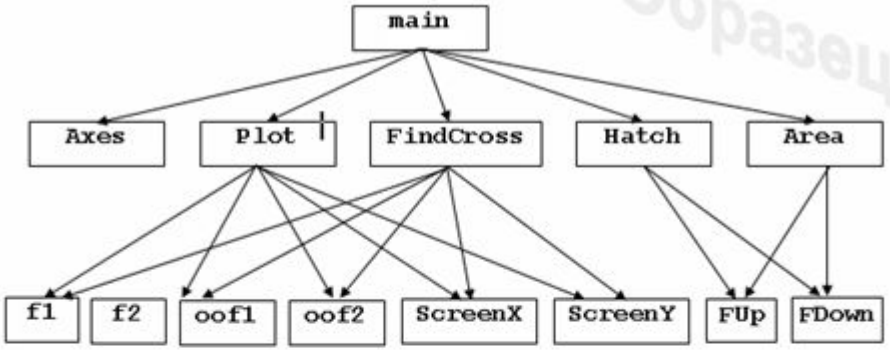
«скриншот» *(screenshot)* – «снимок» экрана

Как получить копию экрана?

1. Поменять цвета так, чтобы все линии и текст были белые.
2. Запустить программу (она должна все нарисовать).
3. Нажать клавишу **PrtScr** (*Print Screen* – «снимок» экрана) на клавиатуре или комбинацию **Alt+PrtScr** («снимок» активного окна).
4. В графическом редакторе (*Paint*): **Правка – Вставить**.
5. Выделить нужную часть рисунка. 
6. Вставить в отчет через буфер обмена (**Ctrl+C, Ctrl+V**).

Структура программы

II. Структура программы



main	основная программа
Axes	построение и разметка осей координат
Plot	построение графиков
FindCross	определение точек пересечения
Hatch	штриховка
Area	вычисление площади
f1	функция $y = \frac{1}{x}$
f2	функция $y = -x^2 + \sin x + 5$
oof1	область определения функции $y = \frac{1}{x}$
oof2	область определения функции $y = -x^2 + \sin x + 5$
ScreenX	пересчет x-координаты точки из математической системы в экранную
ScreenY	пересчет y-координаты точки из математической системы в экранную
Up	верхняя граница заштрихованной области
FDown	нижняя граница заштрихованной области

Текст программы

III. Текст программы


```
program qq;

const X0 = 100; Y0 = 400;
      k = 80;
var xc1, xc2: real;

{---------------------------------------
  ScreenX, ScreenY - переход к экранным координатам
---------------------------------------}
function ScreenX(x: real): integer;
begin
  ScreenX := round(X0 + k*x);
end;
function ScreenY(y: real): integer;
begin
  ScreenY := round(Y0 - k*y);
end;

{---------------------------------------
  Axes - оси координат
---------------------------------------}
procedure Axes;
var i, xe: integer;
begin
 line ( X0, 0, X0, 499 );
 line ( 0, Y0, 699, Y0 );
 for i:=1 to trunc((700-X0)/k) do begin
   xe := ScreenX(i);
   line ( xe, Y0-2, xe, Y0+2 );
   MoveTo(xe-4, Y0+3);
   write(i);
 end;
end;

{---------------------------------------
  SetPoint - точка графика с проверкой
---------------------------------------}
procedure SetPoint ( x, y: real; r, g, b: integer);
var xe, ye: integer;
begin
  xe := ScreenX(x);
  ye := ScreenY(y);
  if (xe >= 0)  and  (xe < 700)  and
     (ye >= 0)  and  (ye < 500) then begin
    Pen(1, r, g, b);
    Point(xe, ye);
  end;
end;

{---------------------------------------
  f1(x), f2(x) - функции
---------------------------------------}
function f1 (x: real): real;
```

шрифт **Courier New**, (моноширинный) размер 10 пт

Конец фильма