Вспомогательные алгоритмы и подпрограммы.

Процедуры

Программирование обработки информации

Вспомогательные алгоритмы и подпрограммы

1

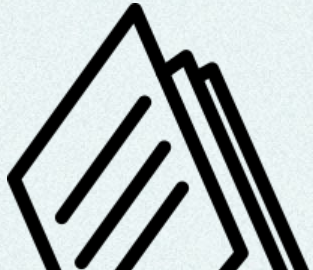
Вспомогательные алгоритмы и подпрограммы.

2

Процедуры в языке Pascal.

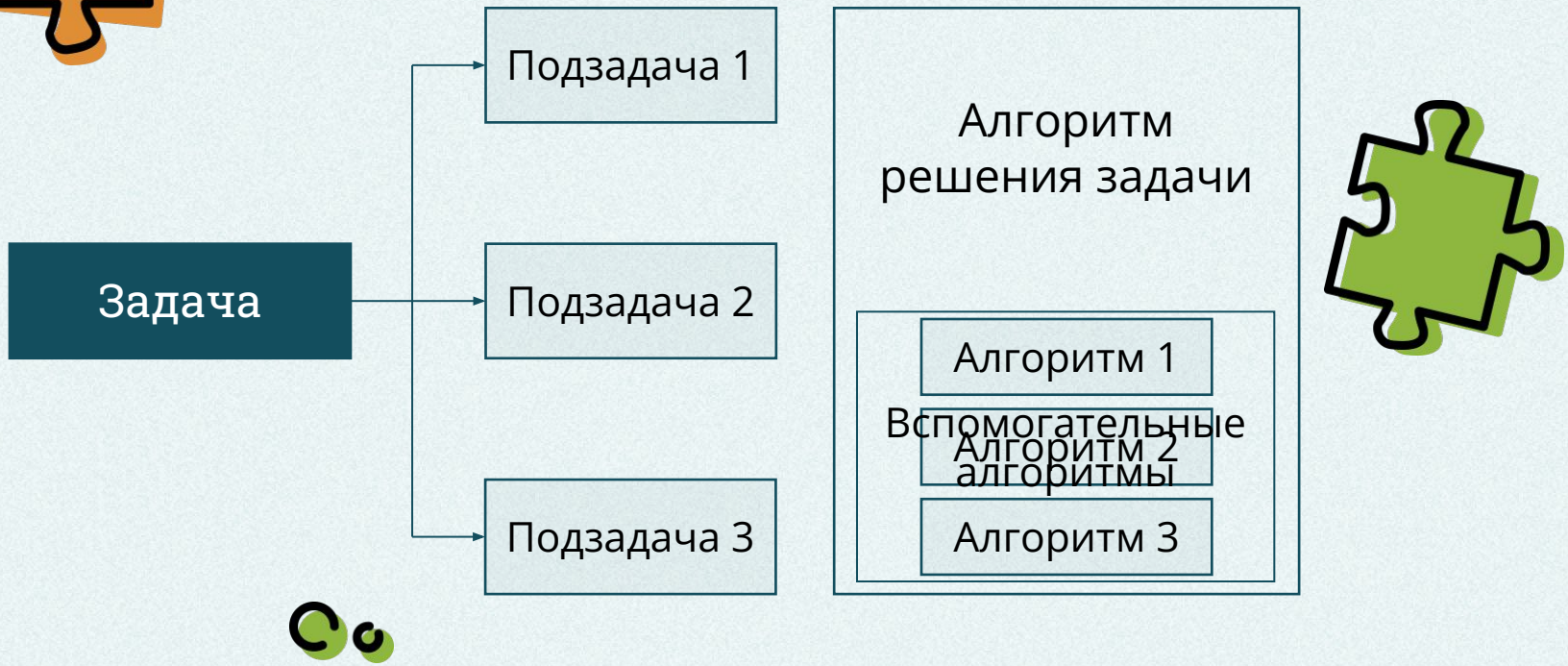
Этапы решения задачи на компьютере:

1. Постановка задачи.
2. Формализация.
3. Анализ математической задачи.
4. Построение алгоритма. – ?
5. Составление программы.
6. Тестирование программы.



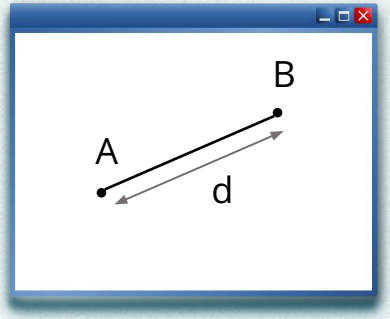
Декомпозиция задачи

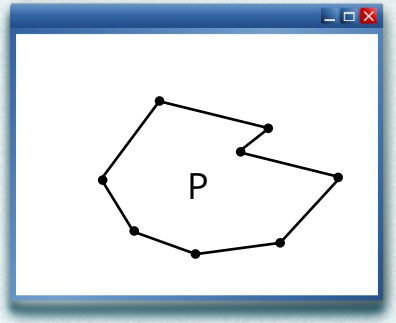
Декомпозиция задачи



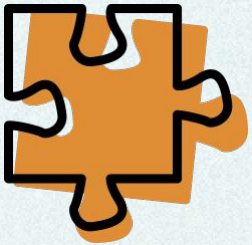
Программирование вспомогательных алгоритмов

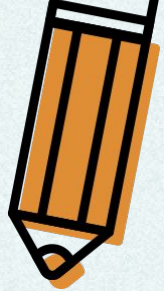
В языках программирования вспомогательные алгоритмы оформляются в виде **подпрограмм**.





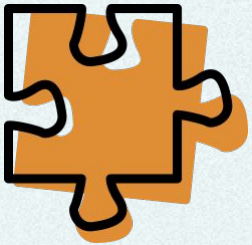
Программирование вспомогательных алгоритмов



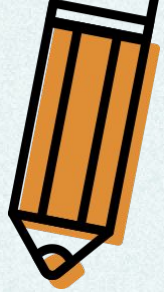


Подпрограммы в языке Pascal

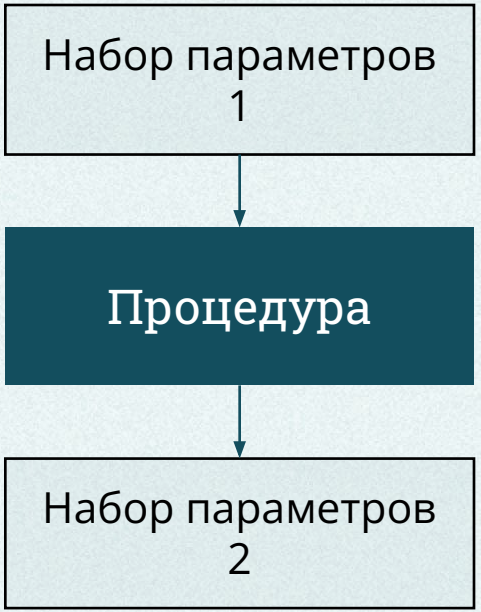
- Процедуры
- Функции

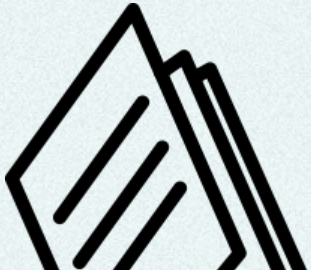


Процедуры

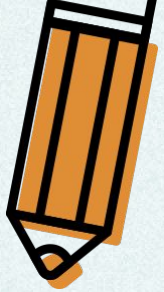


Процедура –
это подпрограмма, которая при выполнении принимает на ввод любое количество параметров, а по завершении работы возвращает также любое количество параметров.





Программирование процедур

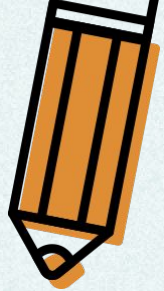


Раздел описания переменных

Раздел описания подпрограмм

Операторный блок

Программирование процедур

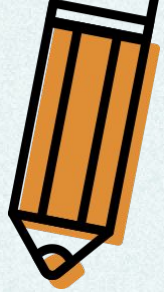


Раздел описания переменных

```
procedure <имя> (<параметры-значения>:
<тип>; var <параметры-переменные>: <тип>);
var
  <дополнительные параметры>: <тип>;
begin
  <оператор 1>;
  <оператор 2>;
  <оператор 3>;
  ...
end;
```

Операторный блок

Программирование процедур



```
procedure <имя> (<параметры-значения>:
<тип>; var <параметры-переменные>: <тип>);
var
  <дополнительные параметры>: <тип>;
begin
  <оператор 1>;
  <оператор 2>;
  <оператор 3>;
  ...
end;
```

Программирование процедур

При вызове процедуры

количество, **порядок следования** и **тип** фактических параметров должны соответствовать формальным параметрам.

```
begin
  <оператор 1>;
  <оператор 2>;
  ...
  <имя>(<фактические п-ры>);
  <оператор 3>;
  <оператор 4>;
  ...
end.
```

```
procedure <имя>(<параметры-значения>:
<тип>; var<параметры-переменные>: <тип>);
var
  <дополнительные параметры>: <тип>;
begin
  <оператор 1>;
  <оператор 2>;
  <оператор 3>;
  ...
end;
```

Области действия параметров

Подпрограмма может обращаться к своим **локальным** и **глобальным** параметрам, но внешняя программа не может обращаться к параметрам подпрограммы.

Глобальные параметры

```
program z1;
var
  k: integer;
  r: real;
begin
  readln (k);
  pr1 (k, r);
  write (r);
end;
```

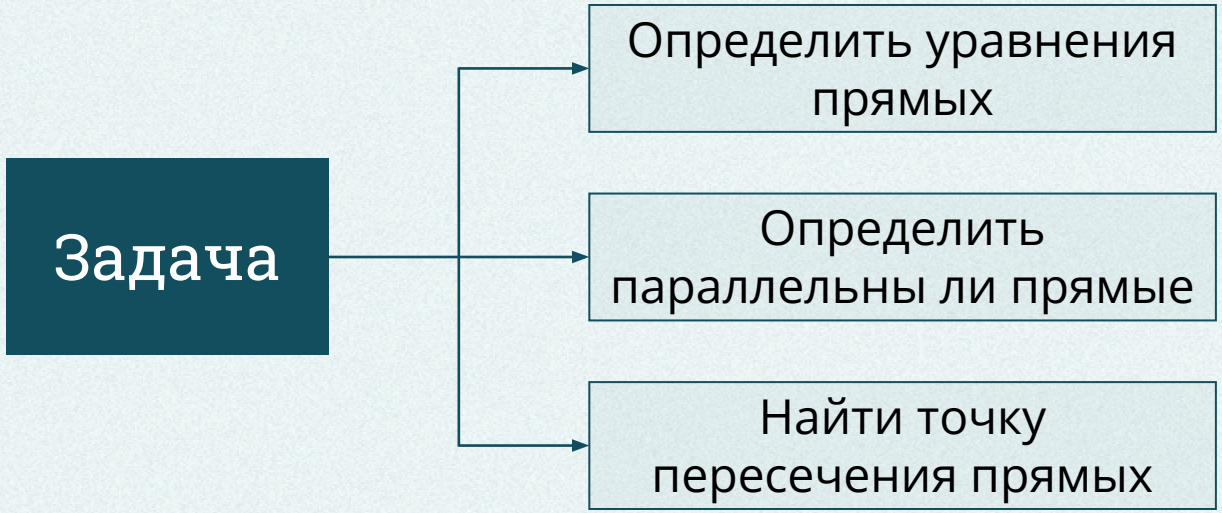
```
procedure pr1 (a: integer; var p: real);
var
  z: boolean;
begin
  z:=a mod 2=0
  p:=a/2;
end;
```

Локальные параметры

Подпрограмма в блок-схеме

Задача

Даны две прямые на координатной плоскости, не совпадающие друг с другом и не параллельные оси **y**. Каждая из них задана координатами 2 точек, лежащих на ней. Найти координаты точки пересечения этих прямых. Если заданные прямые параллельны друг другу – вывести сообщение об этом.



Вывод уравнения прямой

$$y = kx + b$$

$$\begin{cases} y_1 = kx_1 + b \\ y_2 = kx_2 + b \end{cases} \qquad \begin{cases} y_1 - kx_1 = b \\ y_2 - kx_2 = b \end{cases}$$

$$y_1 - kx_1 = y_2 - kx_2$$

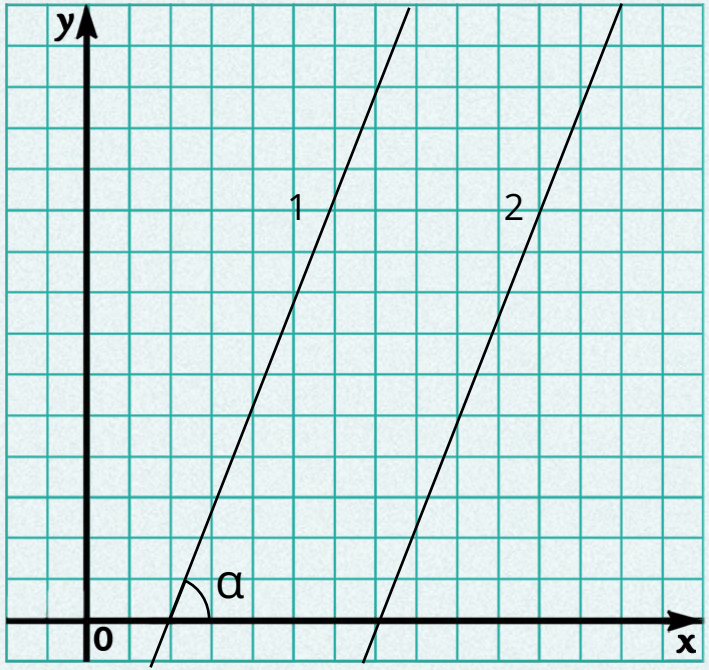
$$kx_2 - kx_1 = y_2 - y_1$$

$$k(x_2 - x_1) = y_2 - y_1$$

$$k = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - kx_1$$

Определение параллельности прямых



$y = k_1x + b_1$
$y = k_2x + b_2$

$k = tg\ \alpha$

$k1 = k2 \Rightarrow 1 \parallel 2$

Поиск точки пересечения прямых

$$\begin{cases} y = k_1x + b_1 \\ y = k_2x + b_2 \end{cases}$$

$$k_1x + b_1 = k_2x + b_2$$

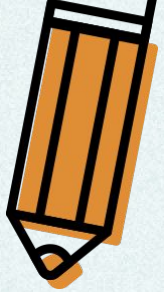
$$k_1x - k_2x = b_2 - b_1$$

$$x(k_1 - k_2) = b_2 - b_1$$

$$x = \frac{b_2 - b_1}{k_1 - k_2}$$

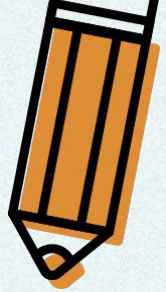
$$y = k_1x + b_1$$

Написание программы



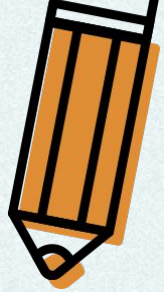
```
program tochka;
var
 x1_1, x1_2, x2_1, x2_2, y1_1, y1_2,
y2_1, y2_2, k1, k2, b1, b2, x, y:
real;

procedure line (x1, y1, x2, y2: real;
var k, b: real);
begin
 k:=(y2-y1)/(x2-x1);
 b:=y1-k*x1;
end;

procedure point ();
begin
 x:=(b2-b1)/(k1-k2);
 y:=k1*x+b1;
end;
```

Написание программы



```
begin
 writeln ('Программа поиска точки пересечения прямых.' );
 writeln ('Введите координаты точек, задающих первую
прямую.');
 write ('x1_1=');
 readln (x1_1);
 write ('y1_1=');
 readln (y1_1);
 write ('x1_2=');
 readln (x1_2);
 write ('y1_2=');
 readln (y1_2);
 writeln ('Введите координаты точек, задающих вторую
прямую.');
 write ('x2_1=');
 readln (x2_1);
 write ('y2_1=');
 readln (y2_1);
 write ('x2_2=');
 readln (x2_2);
 write ('y2_2=');
```

Написание программы



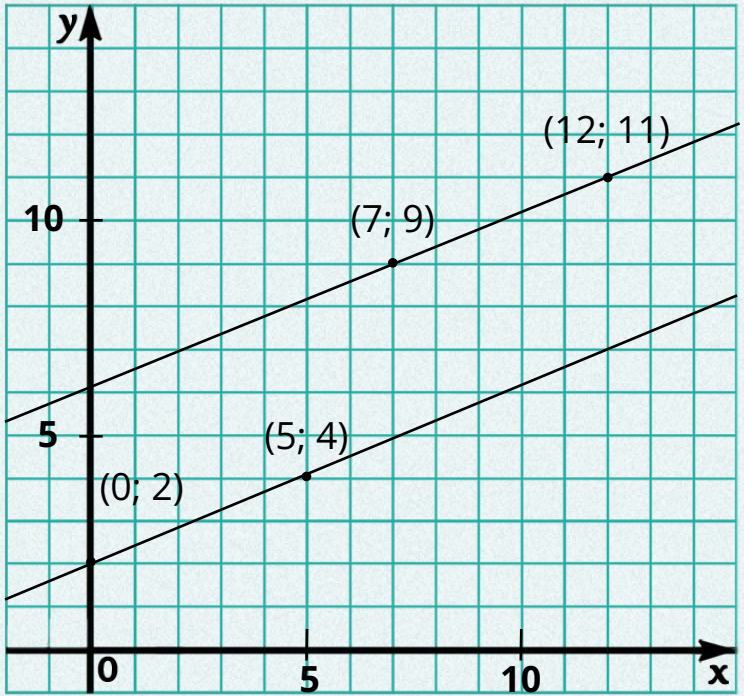
```
line (x1_1, y1_1, x1_2, y1_2, k1, b1);
 line (x2_1, y2_1, x2_2, y2_2, k2, b2);
 if k1=k2
 then write ('Заданные прямые параллельны.')
 else begin
  point ();
  write ('Заданные прямые пересекаются в точке (' , x, '; ', y, ').');
 end;
end.
```

Тестирование программы

Окно вывода

```
Программа поиска точки пересечения прямых.
Введите координаты точек, задающих первую прямую.
x1_1=7
y1_1=9
x1_2=12
y1_2=11
Введите координаты точек, задающих вторую прямую.
x2_1=5
y2_1=4
x2_2=0
y2_2=2
Заданные прямые параллельны.
```

Компиляция прошла успешно (45 строк) Строка 45 Столбец 1

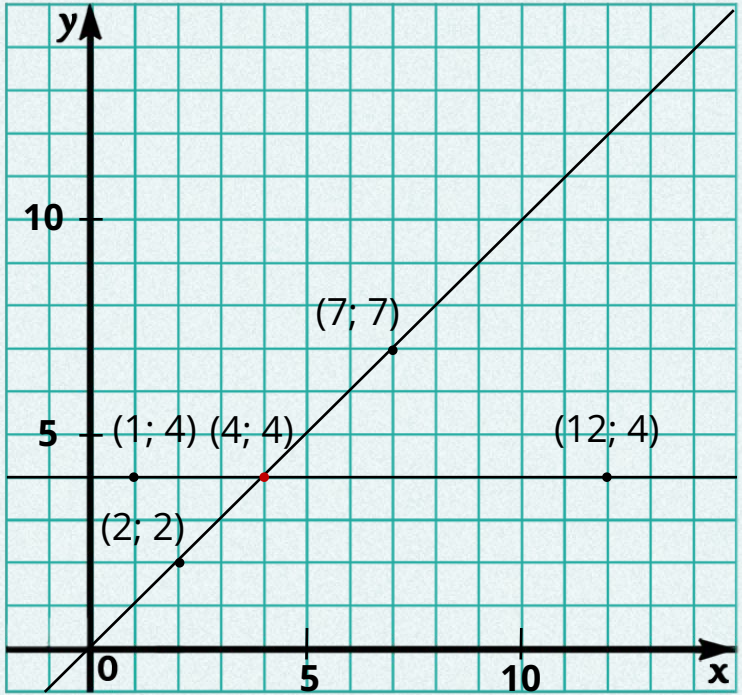


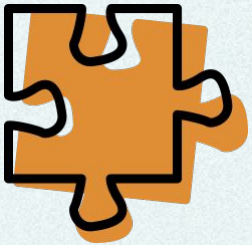
Тестирование программы

Окно вывода

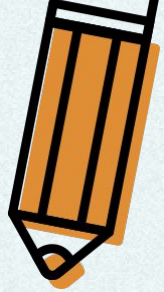
```
Программа поиска точки пересечения прямых.
Введите координаты точек, задающих первую прямую.
x1_1=2
y1_1=2
x1_2=7
y1_2=7
Введите координаты точек, задающих вторую прямую.
x2_1=12
y2_1=4
x2_2=1
y2_2=4
Заданные прямые пересекаются в точке (4; 4).
```

Компиляция прошла успешно (45 строк) Строка 45 Столбец 1





Варианты реализации процедур



Передача данных через локальные параметры

```
procedure line (x1, y1, x2, y2: real;
var k, b: real);
begin
  k:=(y2-y1)/(x2-x1);
  b:=y1-k*x1;
end;
```

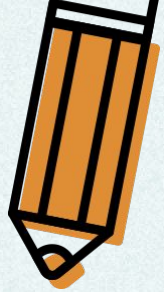
Более универсальный вариант, т. к. такую процедуру можно использовать в разных программах и для различных данных.

Передача данных через глобальные параметры

```
procedure point ();
begin
 x:=(b2-b1)/(k1-k2);
 y:=k1*x+b1;
end;
```

Используется для обработки больших объёмов данных, т. к. экономит оперативную память.

Вспомогательные алгоритмы и подпрограммы



Вспомогательные алгоритмы —

это алгоритмы, которые работают в составе других алгоритмов и используются для решения отдельных подзадач.

Подпрограммы

используются для записи вспомогательных алгоритмов при программировании.

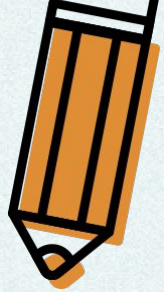
Процедуры и функции —

это два вида подпрограмм в языке Паскаль.

Процедуры —

это подпрограммы, которые могут принимать на вход и возвращать любое количество параметров.

Вспомогательные алгоритмы и подпрограммы



Описание процедуры:

```
procedure <имя> (<параметры-значения>: <тип>;
var <параметры-переменные>: <тип>);
var
  <дополнительные параметры>: <тип>;
begin
  <оператор 1>;
  <оператор 2>;
  <оператор 3>;
  ...
end;
```

Вызов процедуры:

```
<имя> (<фактические параметры>);
```