

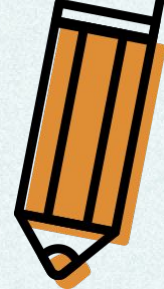
Вспомогательные  
алгоритмы и  
подпрограммы.

Функции

Программирование обработки  
информации



# Вспомогательные алгоритмы и подпрограммы



## Вспомогательные алгоритмы —

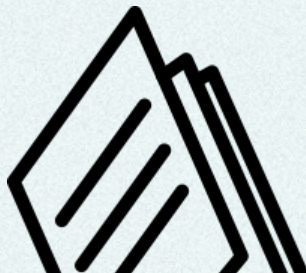
это алгоритмы, которые работают в составе других алгоритмов и используются для решения отдельных подзадач.

## Подпрограммы

используются для записи вспомогательных алгоритмов при программировании.

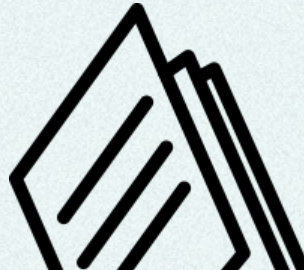
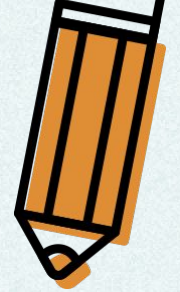
## Процедуры и функции —

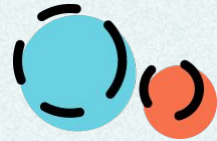
это два вида подпрограмм в языке Паскаль.



# Процедура в программе на языке Pascal

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





# Вспомогательные алгоритмы и подпрограммы

1

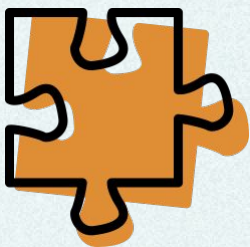
Определение  
функции.

2

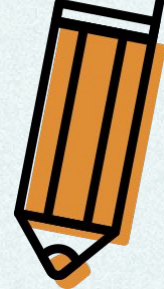
Описание и вызов  
функций в языке  
Pascal.

3

Использование  
функций при  
решении задач.

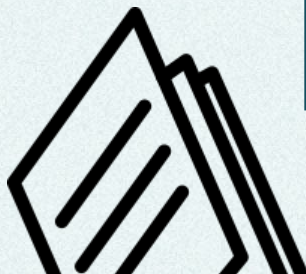
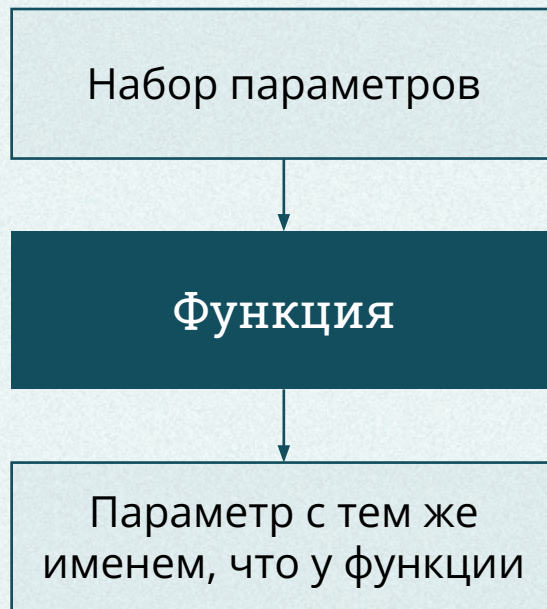


# Функции



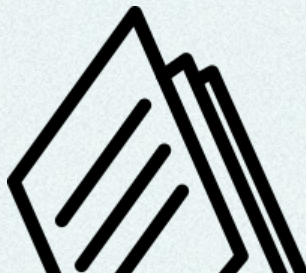
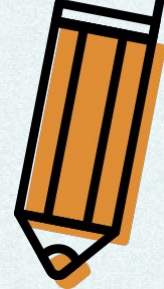
## Функция —



это подпрограмма, которая при выполнении принимает на вход любое количество параметров, а по завершении работы возвращает одно значение в переменную с тем же именем, что и у функции.



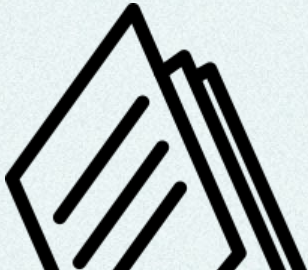
# Описание функции

```
function <имя> (<параметры-значения>: <тип>; var  
<параметры-переменные>: <тип>): <тип значения>;  
var  
  <дополнительные параметры>;  
begin  
  <оператор 1>;  
  <оператор 2>;  
  ...  
  <имя>:=<выражение>;  
  ...  
  <оператор n>;  
end;
```





```
function <имя> (<параметры-значения>: <тип>; var
<параметры-переменные>: <тип>): <тип значения>;
var
  <дополнительные параметры>;
begin
  <оператор 1>;
  <оператор 2>;
  ...
  <имя>:=<выражение>;
  ...
  <оператор n>;
end;
```



# Вызов и исполнение функции

## При вызове функции

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

**begin**

<оператор 1>;

<оператор 2>;

...

t:=<имя> (<фактические п-ры>);

...

<оператор m - 1>;

<оператор m>;

**end.**

```
function <имя> (<параметры-значения>: <тип>; var  
<параметры-переменные>: <тип>); <тип значения>;  
var  
  <дополнительные параметры>;  
begin  
  <оператор 1>;  
  <оператор 2>;  
  ...  
  <имя>:=<выражение>;  
  ...  
  <оператор n>;  
end;
```



## Задача

Написать программу, которая определяет, является ли целое  $n$ , заданное пользователем, простым числом Мерсенна.

$$4 \leq n \leq 2\,000\,000\,000$$

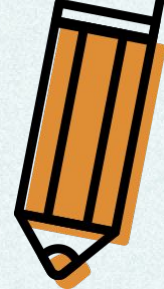
Числа Мерсенна — это числа вида  $M = 2^p - 1$ , где  $p$  — натуральное число.

**Простое число** —

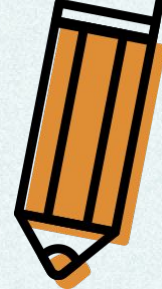
это натуральное число, которое не делится нацело ни на одно целое число, кроме самого себя и единицы.



**Марен Мерсенн**  
(1588 – 1648)

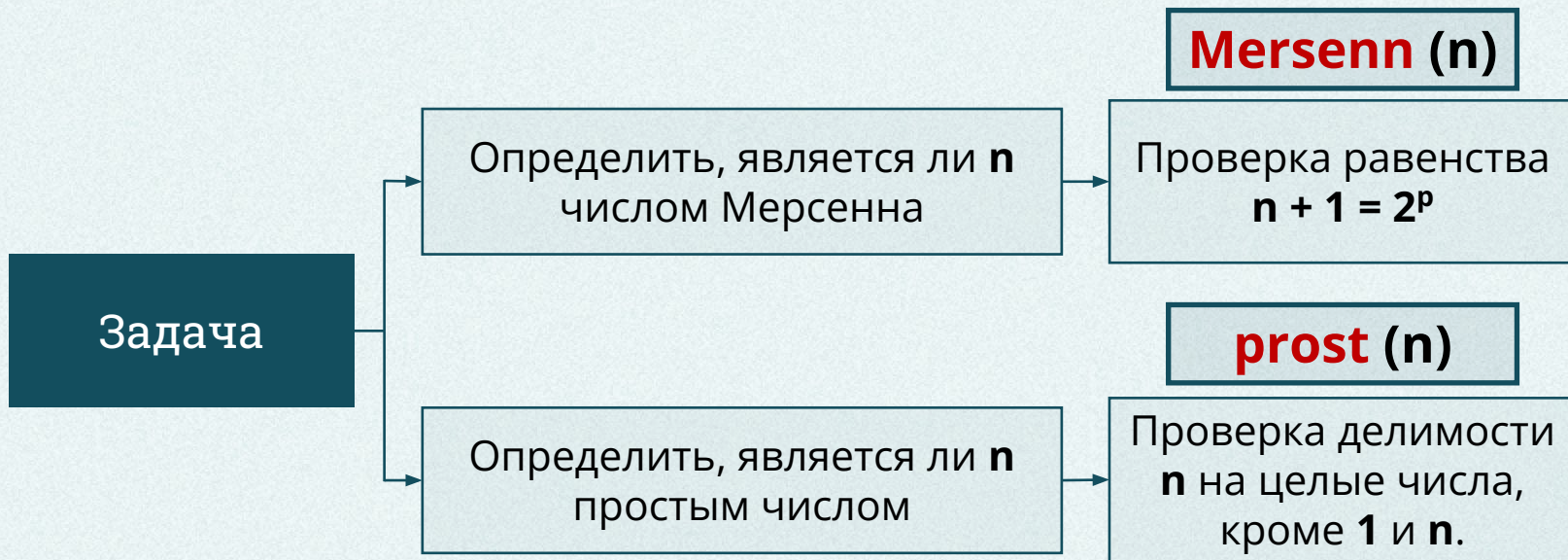


# Задача




Написать программу, которая определяет, является ли целое  $n$ , заданное пользователем, простым числом Мерсенна.

$$4 \leq n \leq 2\,000\,000\,000$$

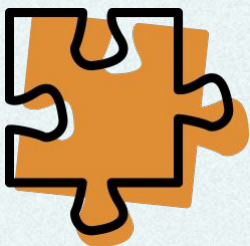


# Написание программы

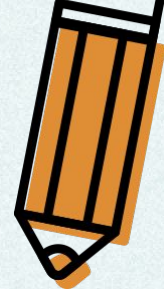


```
program chislo_Mersenna;  
var  
  n: integer;  
  
function Mersenn (n: integer):  
boolean;  
begin  
  n:=n+1;  
  while n mod 2=0 do  
    n:=n div 2;  
  if n=1  
  then Mersenn:=true  
  else Mersenn:=false;  
end;
```

```
begin  
  writeln ('Программа,  
определяющая, является ли целое  
n простым числом Мерсенна.');  write ('n=');  
  readln (n);  
  if Mersenn (n)  
  then if prost (n)  
    then write (n, ' является  
простым числом Мерсенна.')    else write (n, ' не является  
простым числом Мерсенна.')    else write (n, ' не является  
простым числом Мерсенна.');end.
```



## Определение того, является ли число простым



**Простое число** — это целое число, которое не делится нацело ни на одно целое число, кроме самого себя и единицы.

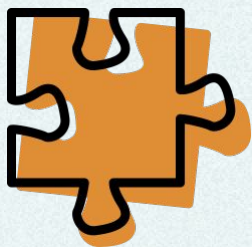
$$\frac{n}{[2; n - 1]}$$

$n$

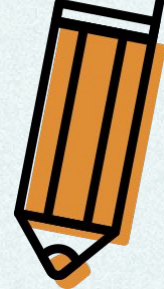
$$n \Rightarrow n = m_1 \times m_2$$

$n$

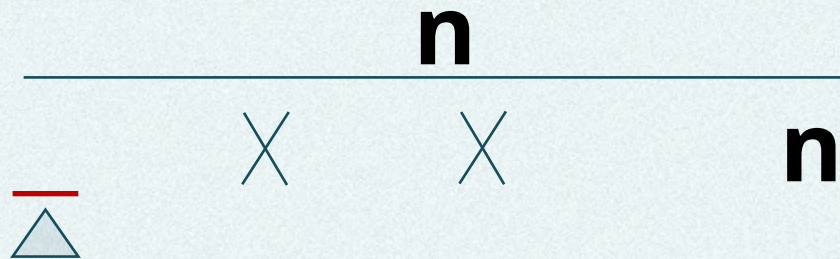
$n$



## Определение того, является ли число простым

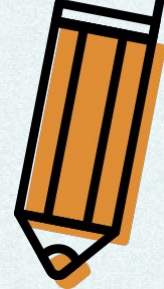


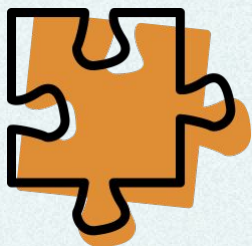
**Простое число** — это целое число, которое не делится нацело ни на одно целое число, кроме самого себя и единицы.



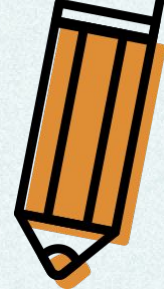
# Написание программы

```
function prost (n: integer): boolean;
var
  i: integer;
  t: boolean;
begin
  t:=n mod 2<>0;
  i:=3;
  while (i<=sqrt(n)+1) and (t=true) do
  begin
    if n mod i=0
    then t:=false;
    i:=i+2;
  end;
  prost:=t;
end;
```



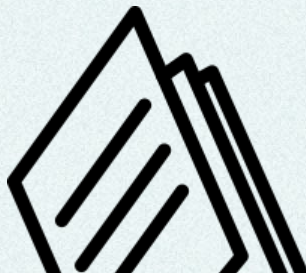


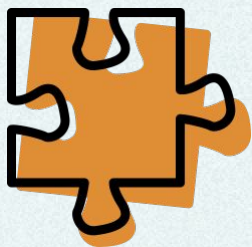
# Вспомогательные алгоритмы и подпрограммы



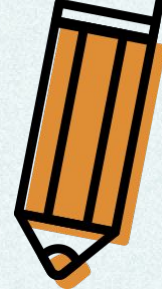
## Функция —

это подпрограмма, которая при выполнении принимает на вход любое количество параметров, а по завершении работы возвращает одно значение в переменную с тем же именем, что и у функции.





# Вспомогательные алгоритмы и подпрограммы



## Описание функции

```
function <имя> (<параметры-значения>: <тип>; var  
<параметры-переменные>: <тип>): <тип значения>;  
var  
  <дополнительные параметры>;  
begin  
  <оператор 1>;  
  <оператор 2>;  
  ...  
  <имя>:=<выражение>;  
  ...  
  <оператор n>;  
end;
```

## Вызов функции

```
begin  
  <оператор 1>;  
  <оператор 2>;  
  ...  
  t:= <имя> (фактические п-ры);  
  ...  
  <оператор m - 1>;  
  <оператор m>;  
end.
```

