Слово "алгоритм" увековечило известного в далеком прошлом математика аль Хорезми, от имени которого оно и произошло.
В IX веке аль Хорезми сформулировал правила выполнения четырех арифметических действий.
def: **АЛГОРИТМ** – однозначно трактуемая (исполнителем) процедура решения задачи, включающая конечную последовательность точно определенных шагов или операций, для выполнения которых (исполнителю) требуется конечный объем памяти и конечное время.

Действия могут быть любыми, но всегда должны быть определены **начальные условия** и **конечный результат**.
Для любых начальных условий, удовлетворяющих заданию на алгоритм, последний должен дать правильный результат. Это **правильный алгоритм**.

Пирог творожный

Начальные условия:

* мука - 2 стакана
* сливочное масло - 200 грамм
* творог - 200 грамм
* сгущенное молоко - 1 банка
* орехи грецкие

Собственно, алгоритм:

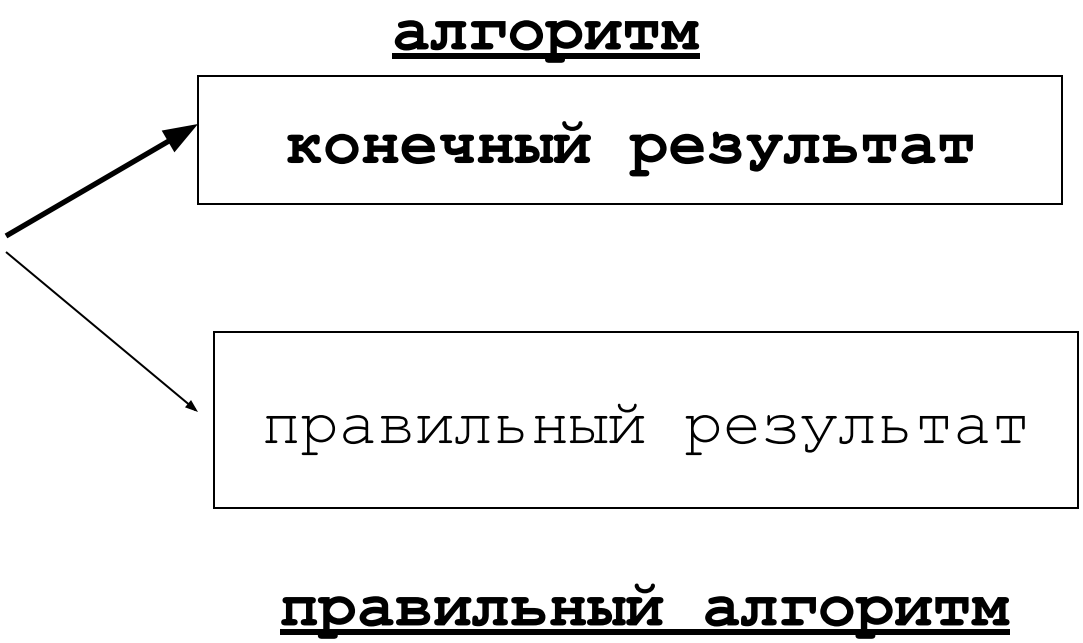
1. размельчить масло
2. тщательно растереть его с творогом
3. добавить муку
4. замесить тесто и оставить его на холоде на 20-30 минут
5. проварить сгущенное молоко 20-30 мин
6. раскатать из теста 4 коржа
7. выпечь коржи
8. смазать коржи горячей сгущенкой
9. посыпать коржи орехами
10. положить коржи один на другой

Вот такой вот пирог.

```
СФОРМИРОВАТЬ СЧЕТ
    ОПРЕДЕЛИТЬ СУММУ ПОКУПКИ
                    в каждой строке счета:
                            сумма = кол-во * цена
                    сложить суммы по всем строкам
    ЕСЛИ СУММА ПОКУПКИ > 1000 руб. ТО
        ОПРЕДЕЛИТЬ СУММУ СКИДКИ
                    определить тип клиента
                    определить величину скидки в процентах
                    сумма скидка = сумма покупки * процент /
100
    ОПРЕДЕЛИТЬ ИТОГ
                    итог = сумма покупки - скидка
НАПЕЧАТАТЬ СЧЕТ
ЗАРЕГИСТРИРОВАТЬ СЧЕТ В ЖУРНАЛЕ
```

Работаем с разными уровнями детализации =>
уменьшаем сложность программирования и количество ошибок =>
ускоряем разработку программ.
Такие простые алгоритмы полезно записывать в комментариях.

ВЫВОД: Алгоритм можно написать на СТРУКТУРНОМ ЕСТЕСТВЕННОМ ЯЗЫКЕ, если Вы его знаете!!!!!!!



Один важный момент, касающийся алгоритмов: алгоритм должен быть понятен исполнителю.

Исполнители могут быть разными -> конкретность алгоритма для исполнителя.

Исполнителя характеризуют:

среда;

система команд;

элементарные действия; (результат выполнения команды)

отказы.(когда команда задана при недопустимом состоянии среды)

СВОЙСТВА алгоритма

Понятность
Дискретность
Определенность
Результативность
Массовость

ФОРМЫ алгоритма

Словесная - рецепт, инструкция
Графическая - блок-схема
Псевдокоды - псевдоязык
Программная - язык программирования

- **Понятность** для исполнителя — т.е. **исполнитель алгоритма должен знать, как его выполнять**.
- **Дискретность** (прерывность, раздельность) — т.е. алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).
- **Определенность** — т.е. каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.
- **Результативность** (или конечность). Это свойство состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов.
- **Массовость.** Это означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется **областью применимости алгоритма.**

Почти всегда составитель алгоритма и исполнитель - разные лица. И Вам надо побеспокоиться о том, чтобы Ваш алгоритм был понятен исполнителю.

Алгоритм может быть записан в любой форме: на естественном языке, на каком-либо символическом языке, на языке схем и т.д.

Например, математические выражения - это тоже алгоритмы для тех кто знает математику:

Пример:

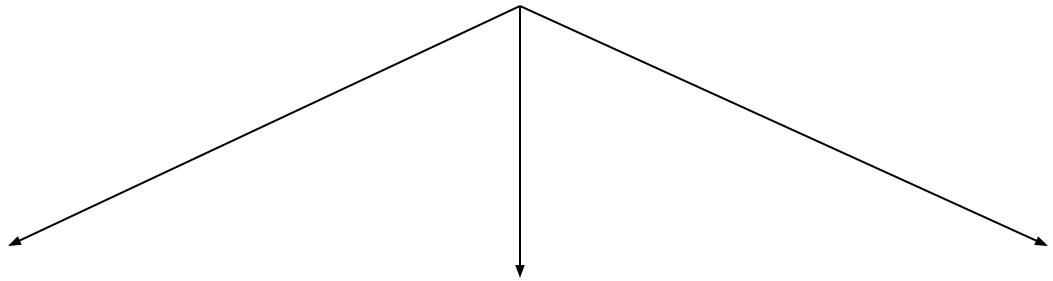
(2^3+4)+2*5

Это простое выражение (алгоритм) с математического языка можно перевести на естественный (словестный) язык:

1. возвести 2 в третью степень (получим 8)
2. к 8 прибавить 4 (получим 12)
3. умножить 2 на 5 (получим 10)
4. к 12 прибавить 10 (получим 22)

Если исполнитель - компьютер, то алгоритм записывается на языке программирования, который данный компьютер «понимает».

перестановка 2-х чисел a и b



```
tmp = a
a   = b
b   = tmp
```

$$a = a + b$$
$$b = a - b$$
$$a = a - b$$

```
a = a xor b
b = a xor b
a = a xor b
```

a=01 b = 10

Исключающее ИЛИ (умножение с переносом)

xor, NEQV, ^

```
01 xor 10 = 11
11 xor 10 = 01
11 xor 01 = 10
```

Целесообразно перед написанием самой программы написать алгоритм ее работы, т.е. определить что и как должна делать программа без углубления в синтаксис конкретного языка программирования. Это позволяет не утонуть в деталях и видеть основной принцип работы.

На практике наиболее распространены следующие формы представления алгоритмов:

словесная (записи на естественном языке);

графическая (изображения из графических символов);

псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);

программная (тексты на языках программирования).

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов

Основные служебные слова

алг (алгоритм)	**сим** (символьный)	**дано**	**для**	**да**
арг (аргумент)	**лит** (литерный)	**надо**	**от**	**нет**
рез (результат)	**лог** (логический)	**если**	**до**	**при**
нач (начало)	**таб**(таблица)	**то**	**знач**	**выбор**
кон (конец)	**нц** (начало цикла)	**иначе**	**и**	**ввод**
цел (целый)	**кц** (конец цикла)	**все**	**или**	**вывод**
вещ (вещественный)	**длин** (длина)	**пока**	**не**	**утв**

Общий вид алгоритма,записанного на псевдокоде:

алг название алгоритма (аргументы и результаты)
  дано условия применимости алгоритма
  надо цель выполнения алгоритма
нач описание промежуточных величин
  последовательность команд (тело алгоритма)
кон

Часть алгоритма от слова **алг** до слова **нач** называется **заголовком**, а часть, заключенная между словами **нач** и **кон** — **телом** алгоритма.

В предложении **алг** после названия алгоритма в круглых скобках указываются:
характеристики (**арг, рез**) и
тип значения (цел, вещ, сим, лит или **лог)**
всех **входных** (**аргументы**) и
выходных (**результаты**) **переменных**.
При описании массивов (таблиц) используется служебное слово **таб**, дополненное **граничными парами** по каждому индексу элементов массива.

Примеры предложений алг:

```
алг Объем и площадь цилиндра (арг вещ R, H, рез вещ V, S)
алг Корни КвУр(арг вещ a, b, c, рез вещ x1, x2, рез лит t)
алг Исключить элемент(арг цел N, арг рез вещ таб A[1:N])
алг Диагональ(арг цел N, арг цел таб A[1:N,1:N], рез лит
Otvet)
```

Предложения **дано** и **надо** не обязательны.
В них рекомендуется записывать утверждения, описывающие **состояние среды исполнителя алгоритма**, например:

алг Число максимумов (**арг цел** N, **арг вещ таб** A[1:N], **рез цел** K)
дано | N>0
надо | K - число максимальных элементов в таблице A

Здесь в предложениях **дано** и **надо** после знака "|" записаны **комментарии**.

Оператор присваивания := служит для вычисления выражений и
присваивания их значений переменным. Для ввода и вывода данных используют команды

Общий вид: **A := B**

ввод имена переменных

вывод имена переменных, выражения, тексты.

Для ветвления применяют команды **если** и **выбор,**

для **организации циклов** — команды **для** и **пока.**

Пример записи алгоритма в псевдокодах

```
алг Сумма квадратов (арг цел n, рез цел S)
   дано | n > 0
   надо | S = 1*1 + 2*2 + 3*3 + ... + n*n
нач
   цел i
   ввод n; S:=0
   нц для i от 1 до n
      S:=S+i*i
   кц
   вывод "S = ", S
кон
```

Графическое описание алгоритмов выполняется с помощью блок-схем, составленных из последовательности блоков, предписывающих выполнение отдельных операций, и связей между ними.

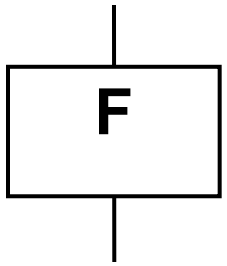
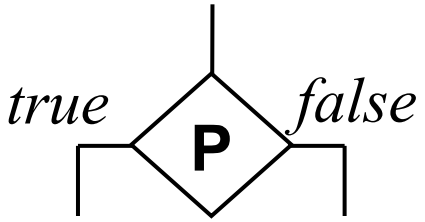
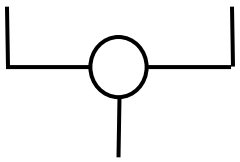
Правила оформления блок-схем

1. внутри блоков помещается информация, поясняющая действия;
2. конфигурация блоков и размеры оговорены ГОСТом;
3. каждый блок снабжается номером в разрыве контура блока в левой верхней части;
4. высота блока (размер ***a***) выбирается из ряда *10, 15, 20…мм*;
5. ширина блока (размер ***b***) выбирается как $b=1,5\times a$;
6. в схемах, не являющихся документацией (плакаты, курсовые работы), можно увеличить ширину блока для удобства записи информации;
7. ход вычислительного процесса изображается линиями связи;
8. линии связи обязательно имеют стрелки, если они направлены снизу вверх или справа налево – в остальных случаях стрелки необязательны.

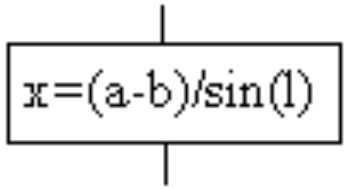
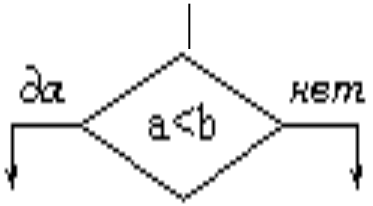
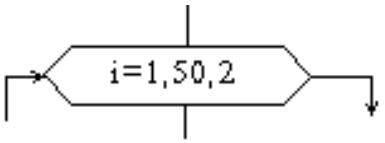
Преимущества графического описания алгоритмов

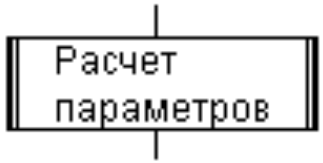
1. наглядность;
2. читаемость;
3. явное отображение управления;
4. имеет стандарт изображения элементов.

- Типы вершин блок-схем алгоритмов
- **Блок-схема** – ориентированный граф, указывающий порядок исполнения команд алгоритма.
  Существует только три типа вершин.

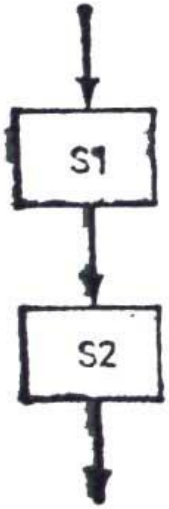
1. Функциональная вершина – имеет один вход и один выход.	
2. Предикатная вершина – имеет один вход и два выхода. Функция *P* передает управление по одной из ветвей в зависимости от значения функции *P*.	
3. Объединяющая вершина – обеспечивает передачу управления от одного из двух входов к выходу	

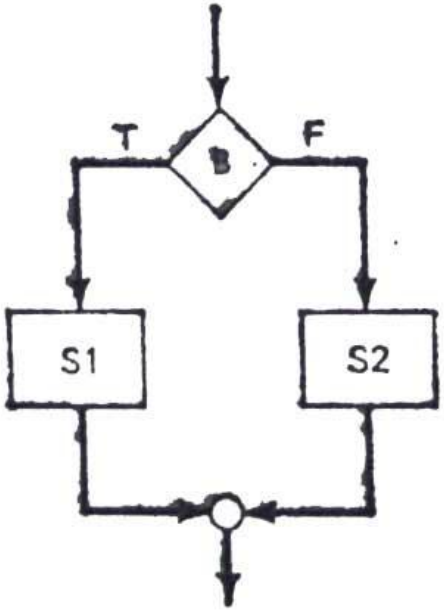
В таблице приведены наиболее часто употребляемые блоки схем алгоритмов.

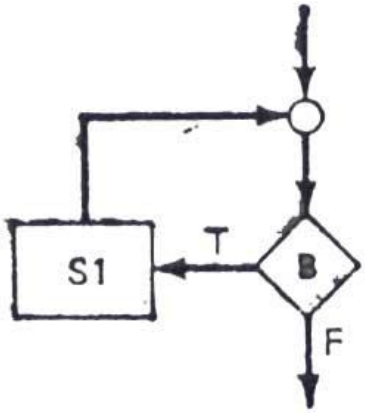
Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модифи-кация		Начало цикла

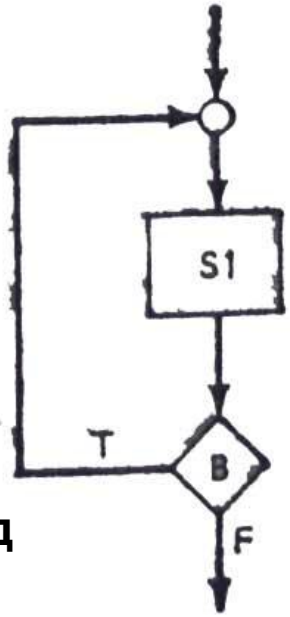
Предопре-деленный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

Принцип структурного программирования
(Теорема Бема-Якопини) –
логическая структура любой программы может быть выражена комбинацией из базовых структур:
1) Следование. 2) Ветвление. 3) Цикл.
Структурная блок схема -композиция из базовых алгоритмических структур









Один вход - один выход

Следование Ветвление (выбор) Цикл
Композиция Альтернатива Итерация
IF THEN ELSE **WHILE DO** **DO WHILE**

1. **Базовая структура <u>следование</u>**. Образуется из последовательности действий, следующих одно за другим:

<u>Псевдокоды</u>

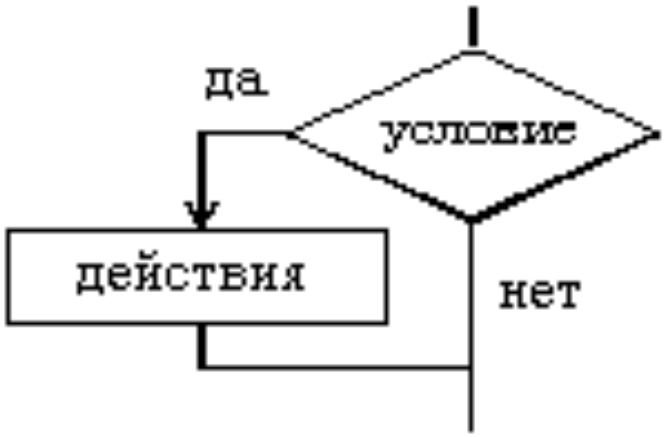
действие 1
действие 2
. . . . . .
действие n

<u>Язык блок-схем</u>

Структура **ветвление** существует в четырех основных вариантах:

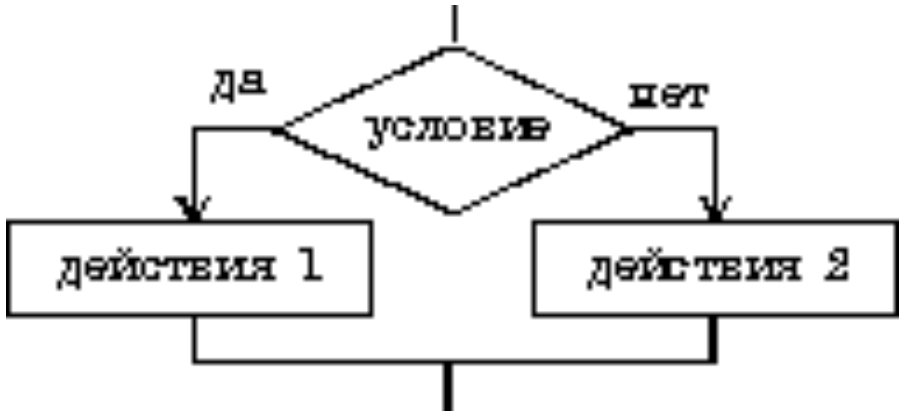
. 1. **если-то**

если условие
то действия
все

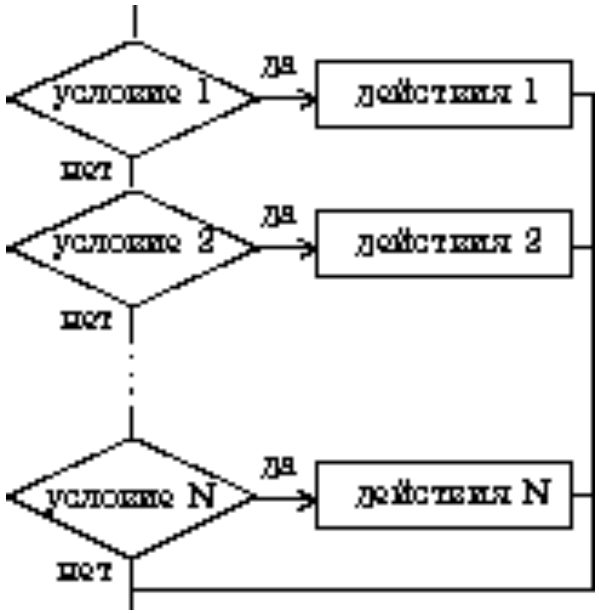


2. **если-то-иначе**

Если условие
то действия 1
иначе действия 2
все

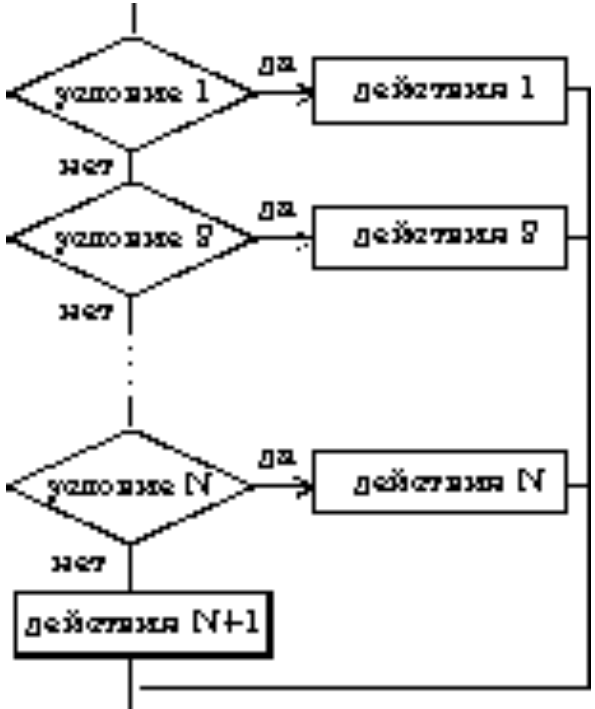


3. **выбор**



выбор
при условие1: действия1
при условие2: действия2
. . . . . .
при условиеN: действияN
все

4. **выбор-иначе**



Выбор
при условие1: действия1
при условие2: действия2
. . . . . .
при условиеN: действияN
иначе действия N+1
все

Способы комбинации структур

Путем *СЛЕДОВАНИЯ* структур друг за другом.

Путем создания *СУПЕРПОЗИЦИЙ* – вложение одной структуры в другую.

Признаки структурного программирования

1. Полное *исключение операторов безусловных переходов*.

2. Модульность.

Модуль – последовательность логически связанных операций, оформленных как отдельная часть программы.

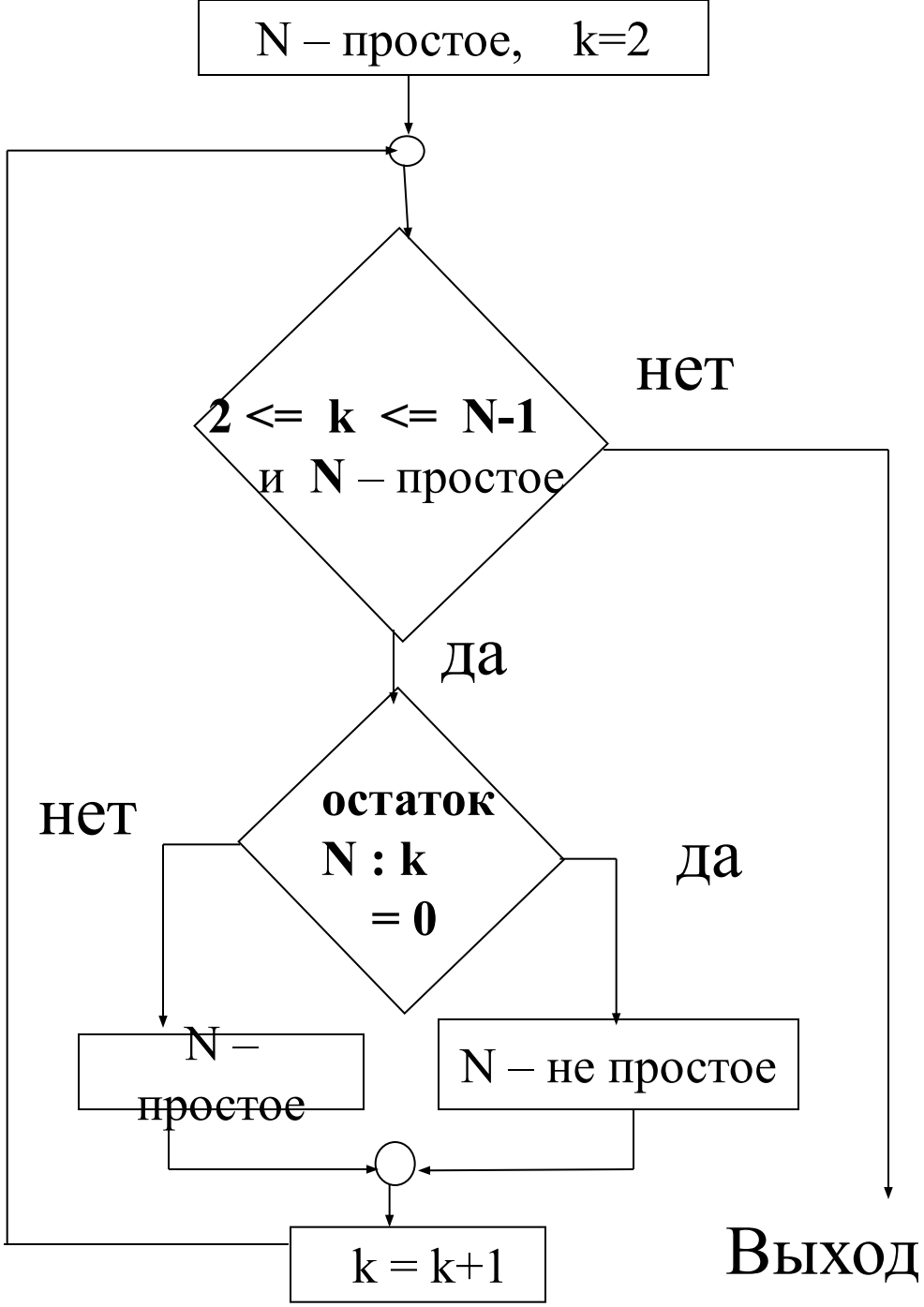
Преимущества модульной структуры:

- возможность разработки программы несколькими программистами;
- простота проектирования и модификации программ;
- упрощение отладки программ: поиска и устранения ошибок;
- возможность использования готовых библиотек подпрограмм и модулей;
- лучшая читаемость программ.

3. Детализация или декомпозиция – нисходящее проектирование программ:

- построение иерархии модулей программ;
- разбиение задач на подзадачи;
- детализация до уровня подзадач, решение которых обеспечивается за 3÷5 строк.

Определить является ли натуральное число N>2 простым ?



Определить является ли
натуральное число N>2 простым ?

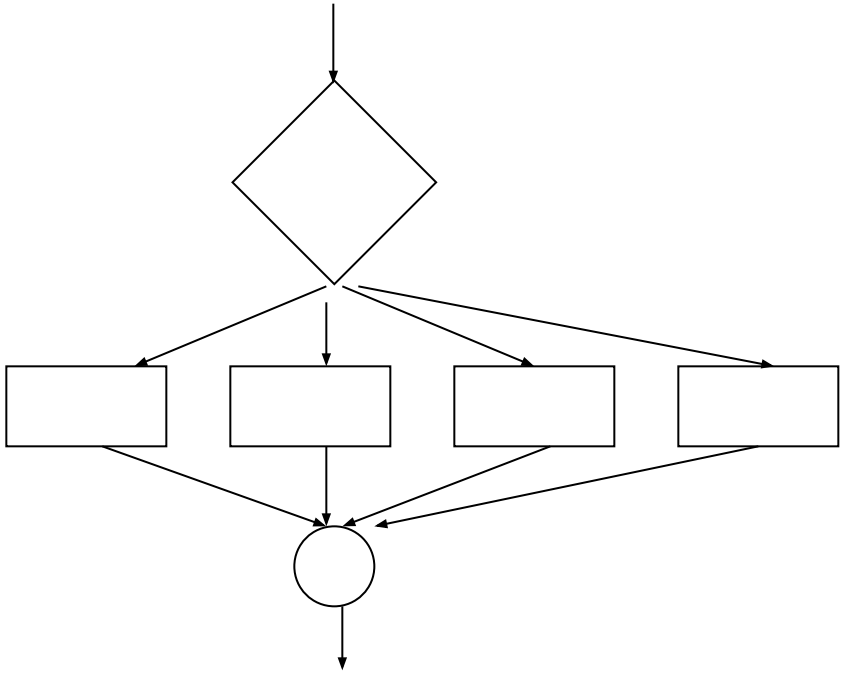
Алгоритм N>2 простое ?

N – простое
Перебор натур.чисел k от 2 до N-1
Если (остаток N / k == 0 и N – простое), то
N – не простое

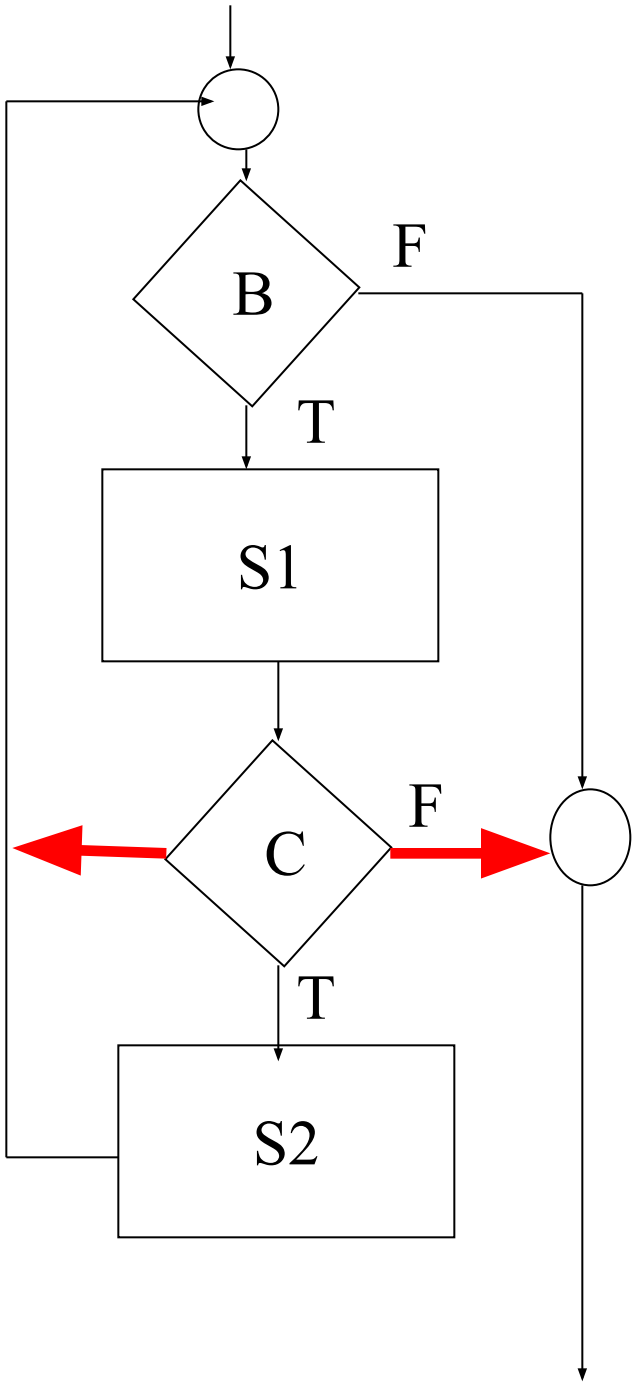
Алгоритм не эффективен, но правилен

Расширения базовых структур

Многовариантный выбор



Прекращение итерации цикла



Досрочный выход из цикла