ИНФОРМАТИКА

7. Алгоритмизация и программирование

1. Понятие алгоритма

2. Свойства алгоритма

3. Формы записи алгоритмов

4. Базовые алгоритмические структуры

5. Сложность алгоритмов

6. Уровень языка программирования

1. Понятие алгоритма

Алгоритм — **заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.**

2. Свойства алгоритма

1. **Понятность для исполнителя** — исполнитель алгоритма должен понимать, как его выполнять.
2. **Дискретность (прерывность, раздельность)** — алгоритм должен представлять решение задачи как последовательное выполнение простых (или ранее определённых) шагов (этапов).
3. **Определённость** — каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола.
4. **Результативность (или конечность)** — за конечное число шагов алгоритм либо должен приводить к решению задачи, либо останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение заданного времени с выдачей промежуточных результатов.
5. **Массовость** — алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для класса задач, различающихся лишь исходными данными.

3. Формы записи алгоритма

- **Словесная** (запись 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 естественно-го языка, общепринятые математические обозначения и др.);
- **Программная** (тексты на языках программирования).

Словесная форма записи алгоритма

Алгоритм в данной форме записи представляет собой последовательность действий:

1) задать два числа;
2) если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
3) определить большее из чисел;
4) заменить большее из чисел разностью большего и меньшего из чисел;
5) повторить алгоритм с шага 2.

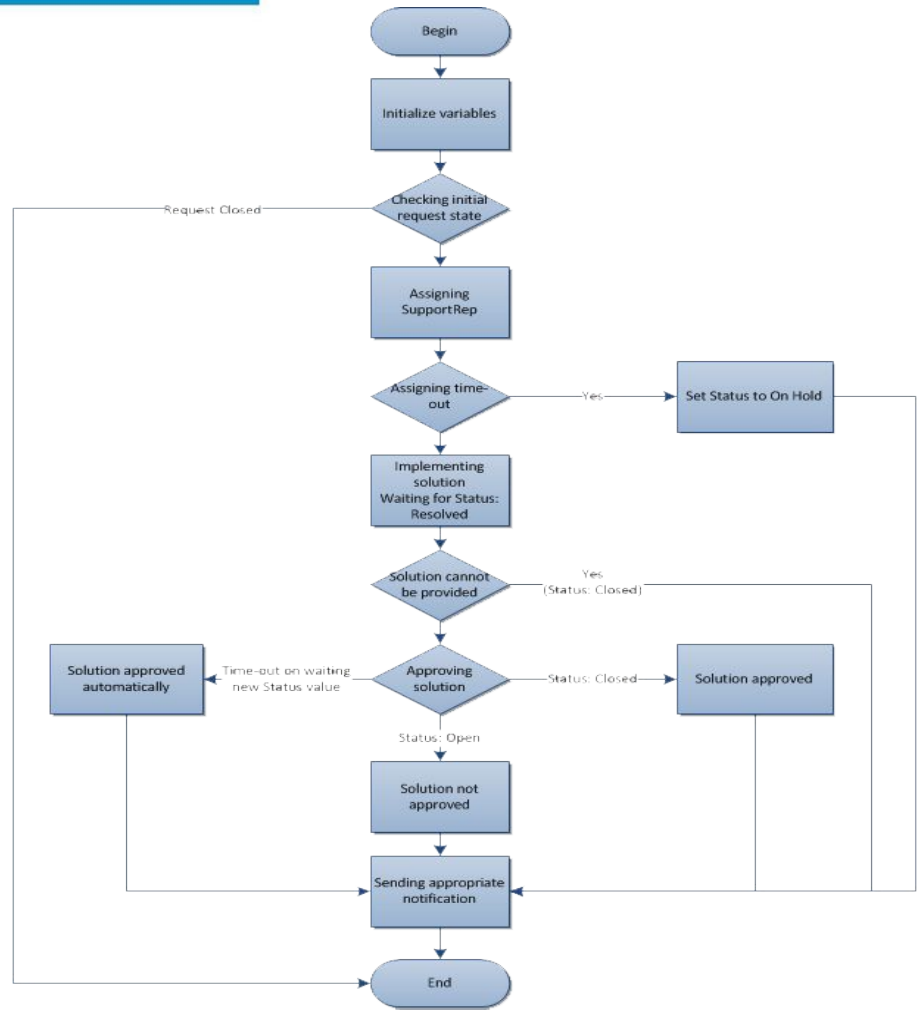
Недостатки словесного способа:

- строго не формализуем для машинного исполнения;
- избыточность;
- допускают неоднозначность толкования отдельных предписаний.

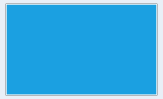
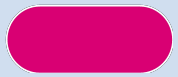
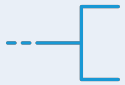
Графическая форма записи алгоритма

- Является более компактной и наглядной по сравнению со словесным;
- Изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий;

Такое графическое представление называется **схемой алгоритма** или **блок-схемой**.



Графическая форма записи алгоритма

Символ	Название	Назначение
	Данные	Обозначение процедуры ввода/вывода данных
	Процесс	Обработка данных в виде операции или группы операций
	Соединитель	Соединение прерванных линий потока
	Преопределенный процесс	Вычисление внутри выделенной подпрограммы/функции/модуле
	Подготовка	Изменение параметров цикла (организация счетчика)
	Решение	Проверка условия и организация ветвления потока
	Терминатор	Вход или выход во внешнюю среду
	Комментарий	Запись пояснений по алгоритму

Псевдокод

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов. В нем не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования.

Примеры обозначений

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

Псевдокод

Примеры записи алгоритмов:

алг Сумма квадратов (**арг цел** n, **рез це**
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

Алгоритм Дейкстры для нахождения кратчайших путей.

DIJKSTRA(G, l, s)
▷ Вход: граф $G(V, E)$ (ориентированный или нет) с неотрицательными
▷ длинами рёбер $\{l_e : e \in E\}$; вершина $s \in V$.
▷ Выход: для всех вершин u, достижимых из s,
▷ $dist[u]$ будет равно расстоянию от s до u (и ∞ для недостижимых).

for всех вершин $u \in V$
$dist[u] \leftarrow \infty$
$prev[u] \leftarrow$ NIL
$dist[s] \leftarrow 0$

$H \leftarrow$ MAKE-QUEUE(V) (в качестве ключей используются значения $dist$)
while $H \neq \emptyset$
$u \leftarrow$ DELETE-MIN(H)
for всех рёбер $(u, v) \in E$
if $dist[v] > dist[u] + l(u, v)$
then $dist[v] \leftarrow dist[u] + l(u, v)$
$prev[v] \leftarrow u$
DECREASE-KEY$(H, v, dist[v])$

Программная форма записи

Примеры программной записи алгоритмов

JavaScript

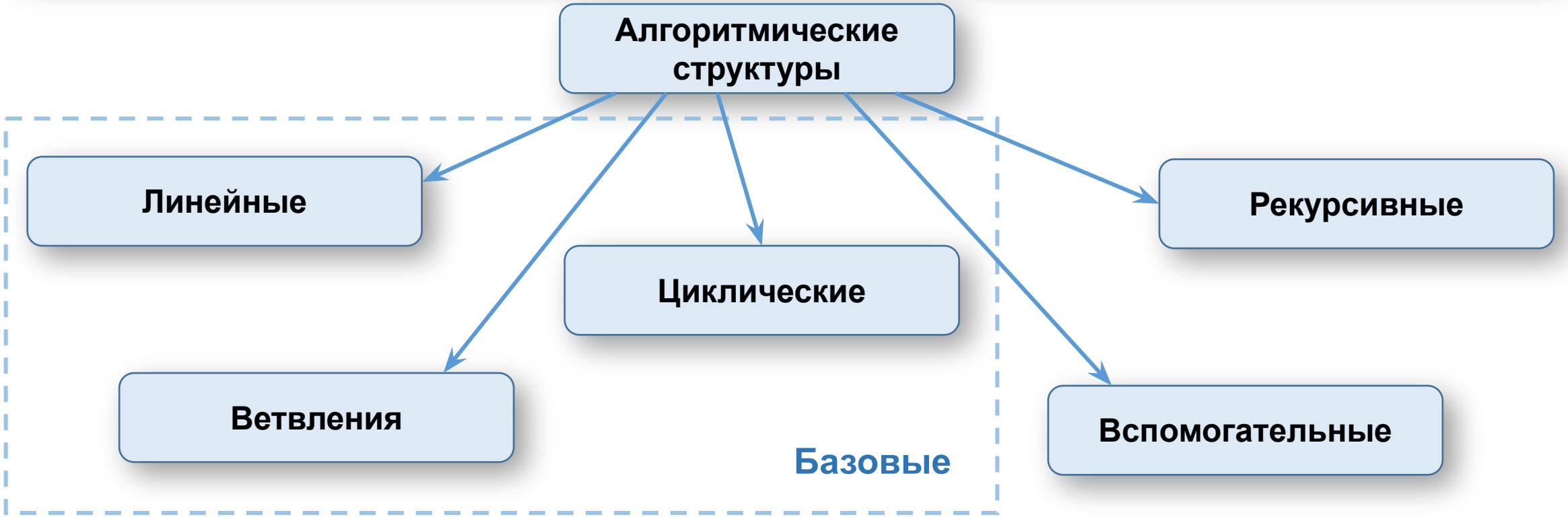
```
function sortBubble(data) {
   var tmp;
   for (var i = data.length - 1; i > 0; i--) {
      var counter=0;
      for (var j = 0; j < i; j++) {
         if (data[j] > data[j+1]) {
            tmp = data[j]; data[j] = data[j+1];
            data[j+1] = tmp; counter++;
         }
      }
      if(counter==0){ break; }
   }
 return data;
};
```

C++

```
void bubble(int* a, int n) {
  for (int i = n - 1; i >= 0; i--) {
    for (int j = 0; j < i; j++) {
      if (a[j] > a[j + 1]) {
        int tmp = a[j];
        a[j] = a[j + 1];
        a[j + 1] = tmp;
      }
    }
  }
}
```

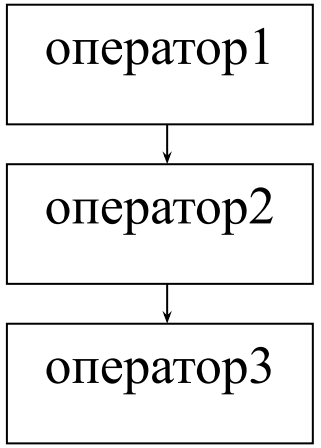
4. Базовые алгоритмические структуры

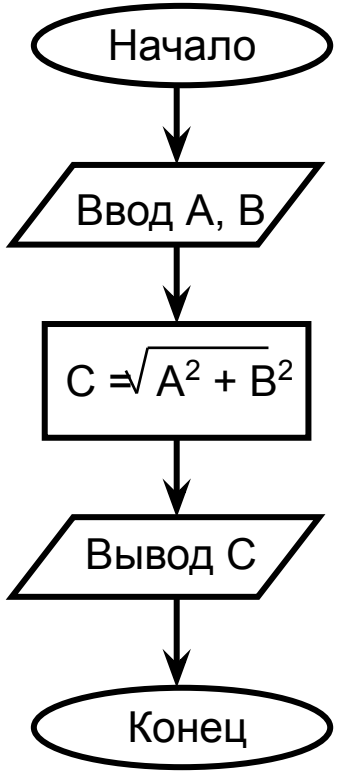
Алгоритмические структуры — типовые группы элементарных шагов алгоритма.



Линейная структура

Линейный алгоритм Р (или его часть) — если каждый шаг алгоритма выполняется только один раз, причем после каждого i-го шага выполняется (i + 1)-й шаг, если i-й шаг — не конец алгоритма.



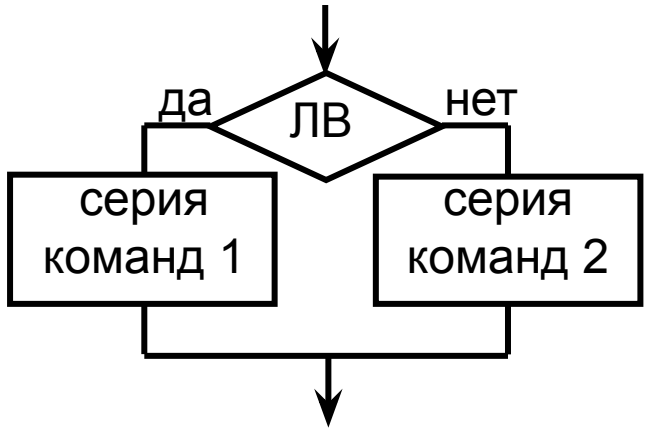


Ветвление

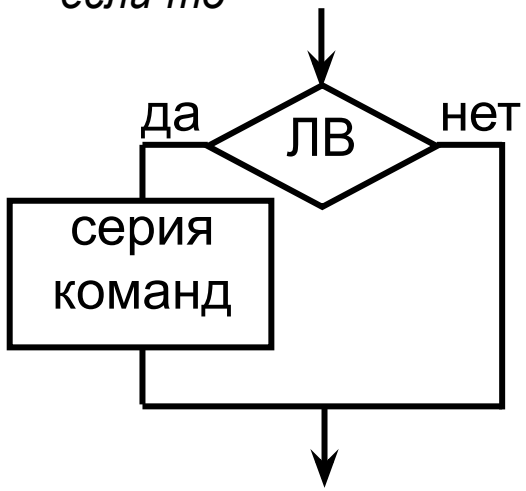
Ветвление — структура, где вычислительный процесс реализуется по одному из нескольких заранее предусмотренных направлений (**ветвей**) в зависимости от выполнения некоторого условия (логического выражения — ЛВ).

Ветвящийся процесс, включающий в себя две ветви, называется ***простым***, более двух ветвей - ***сложным***.

полное ветвление
если-то-иначе



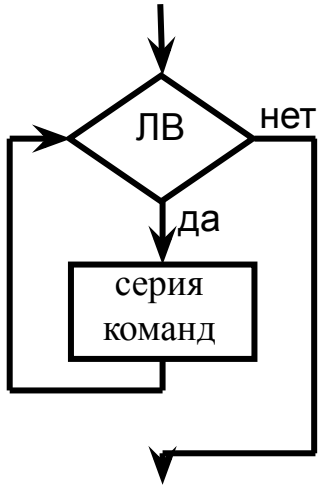
неполный вариант ветвления
если-то



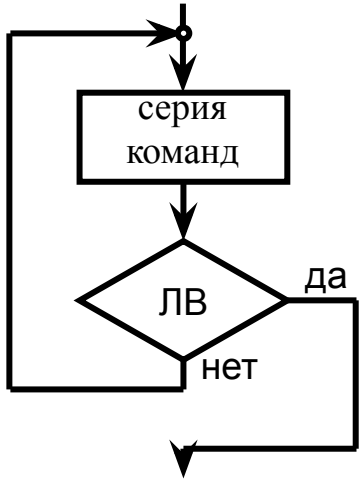
Циклические структуры

Цикл — структура, где подряд идущая группа шагов алгоритма, выполняется несколько раз. Количество повторений либо фиксировано, либо зависит от входных данных алгоритма.

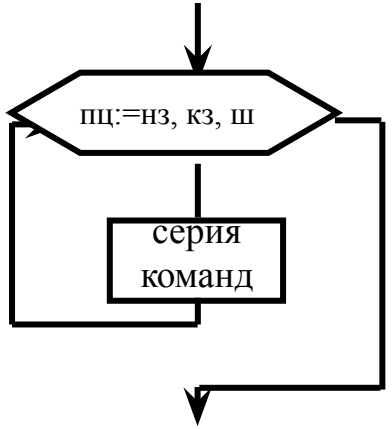
цикл с предусловием



цикл с постусловием



цикл со счетчиком



5. Сложность алгоритмов

Сложность алгоритма $O(f(n))$ — ограничительная функция от размерности задачи *n*, показывающая, что время работы алгоритма (или объём занимаемой памяти) растёт в зависимости от объёма входных данных не быстрее, чем некоторая константа, умноженная на $f(n)$, где $f(n)$ — функция, выражающая скорость роста объема вычислений от увеличения размерности задачи n.

Если $f(n) = n^m$, т.е. имеет линейный или полиномиальный характер, то алгоритм считается «**хорошим**», если экспоненциальный , т.е. $f(n) = e^n$ — «**плохим**».

Примеры: $O(n)$ — линейная сложность: *поиск максимума и минимума*, $O(\log n)$ — логарифмическая сложность: *бинарный поиск*; $O(n2)$ — квадратичная сложность: *алгоритм сортировки вставками*.

6. Уровень языка программирования

Классификация языков программирования по критерию детализации предписаний:

- **машинные** (самого низкого уровня);
- машинно-ориентированные (**ассемблеры**);
- машинно-независимые (**высокого уровня**).

Источники информации

1. Симонович С. В. Информатика. Базовый курс: Учебник для вузов. 3-е изд. Стандарт третьего поколения. — СПб.: Питер, 2011. — 640 с.
2. Макарова Н.В., Волков В.Б. Информатика. — СПб.: Питер, 2012. — 576 с.
3. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург. 2011. — 720 с.
4. Дж. Макконелл. Анализ алгоритмов. Активный обучающий подход. — 3-е дополненное издание. М: Техносфера, 2009. — 416 с.
5. ГОСТ 19781-74. Единая система программной документации. Термины и определения. Утв. пост. Госкомстата № 2051 от 08.05.08.
6. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных систем. Условные обозначения и правила выполнения.

Чесноков Сергей Евгеньевич,
доцент кафедры информатики
ФГБОУ ВО «ПГТУ», г. Йошкар-Ола
kinf@volgatech.net