Статические массивы

1. Понятие
2. Заполнение значениями;
3. Вычисления;
4. Поиск;
5. Вычислительная сложность алгоритмов;
6. Сортировка.

Статические массивы

Массивы – формальное объединение нескольких однотипных объектов (чисел, символов, строк и т.п.), рассматриваемое как единое целое.

Объявление массива

```
var
    a: array [1..10] of real;
    b: array[0..50] of integer;
    c: array[-3..4] of boolean;
```

Обращение к элементу массива

```
b[17] := 123;
c[-2] := a[1]>a[2];
for k := 1 to 10 do a[k] := 0;
```

Статические массивы

Многомерные массивы

```
var
  a: array[1..3] of array[–2..2] of array[1..5] of real;
```

или более компактная запись

```
var
  a: array[1..3, –2..2, 1..5] of real;
```

Статические массивы

Можно передать все элементы одного массива другому массиву

```
var
  a, b: Vector;
или
  a, b: array[1..5] of real;
...
a := b;
```

```
var
  a: array[1..5] of real;
  b: array[1..5] of real;
...
a := b;   {Здесь возникнет ошибка несоответствия типов}
```

Статические массивы

Над массивами не определены операторы отношения

```
var
  a, b: array[1..5] of real;
  eq: boolean;
  i: integer;
...
eq := true;
for i := 1 to 5 do
  if a[i]<>b[i] then eq := false;
if eq then ...
```

Статические массивы

1. Понятие
2. **Заполнение значениями;**
3. Вычисления;
4. Поиск;
5. Вычислительная сложность алгоритмов;
6. Сортировка.

Статические массивы

Пример: заполнить массив вещественными числами из отрезка [5;10]

Генератор случайного числа:

- Random(maxValue: integer): integer;

  Возвращает случайное целое в диапазоне от 0 до maxValue-1

- Random(a,b: integer): integer;

  Возвращает случайное целое в диапазоне от a до b

- Random: real;

  Возвращает случайное вещественное в диапазоне [0..1)

Статические массивы

Вариант 1 (используя Random(maxValue: integer): integer)

```
const
  N = 10;
var
  m : array [1..N] of real;   // массив
  i : integer;                // счетчик цикла
begin
  // заполнение массива значениями
  for i := 1 to N do
     m[i] := random(50+1)/10 + 5;
  // вывод значений на экран
  for i := 1 to N do
     writeln('m[',i:2,'] =',m[i]:4:1);
end.
```

```
m[ 1] = 9.1
m[ 2] = 9.9
m[ 3] = 6.0
m[ 4] = 5.7
m[ 5] = 7.3
m[ 6] = 5.0
m[ 7] = 9.0
m[ 8] = 9.4
m[ 9] = 5.9
m[10] = 7.0
```

Статические массивы

Вариант 2 (используя Random(a,b: integer): integer)

```
const
  N = 10;
var
  m : array [1..N] of real;   // массив
  i : integer;                // счетчик цикла
begin
  // заполнение массива значениями
  for i := 1 to N do
     m[i] := random(50,100)/10;
  // вывод значений на экран
  for i := 1 to N do
     writeln('m[',i:2,'] =',m[i]:4:1);
end.
```

```
m[ 1] = 7.2
m[ 2] = 6.4
m[ 3] = 5.0
m[ 4] = 5.0
m[ 5] = 6.2
m[ 6] = 5.9
m[ 7] = 8.5
m[ 8] = 7.9
m[ 9] = 7.1
m[10] = 9.5
```

Статические массивы

Вариант 3 (используя Random: real)

```
const
  N = 10;
var
  m : array [1..N] of real;   // массив
  i : integer;                // счетчик цикла
begin
  // заполнение массива значениями
  for i := 1 to N do
     m[i] := random*5 + 5;
  // вывод значений на экран
  for i := 1 to N do
     writeln('m[',i:2,'] =',m[i]:4:1);
end.
```

```
m[ 1] = 6.0
m[ 2] = 5.9
m[ 3] = 5.5
m[ 4] = 6.1
m[ 5] = 7.7
m[ 6] = 9.3
m[ 7] = 9.6
m[ 8] = 9.0
m[ 9] = 7.4
m[10] = 5.2
```

Статические массивы

Замечания по примеру:

- в данном случае можно было использовать только один цикл, в нем формировать и выводить на экран значения. В общем случае на операции формирования, обработки и вывода требуется свой цикл;
- в вариантах 1 и 2 результат вещественный с точностью до десятых, в 3 – 11-12 значимых.

Статические массивы

Пример: заполнение двумерного массива

```
const
  Nr = 2;   Nc = 4;
var
  a, b : array [1..Nr,1..Nc] of integer;
  i,j : integer;
begin
writeln('После объявления массива a');
for i := 1 to Nr do
  begin
     for j := 1 to Nc do write (a[i,j]:3);
     writeln;
  end;
for i := 1 to Nr do
   for j := 1 to Nc do a[i,j] := random(11);
```

Статические массивы

```
writeln('После заполнения массива
a');
for i := 1 to Nr do
  begin
    for j := 1 to Nc do write (a[i,j]:3);
    writeln;
  end;
b := a;
writeln('Содержимое массива b');
for i := 1 to Nr do
  begin
    for j := 1 to Nc do write (b[i,j]:3);
    writeln;
  end;
end.
```

После объявления массива a
 0 0 0 0
 0 0 0 0
После заполнения массива a
 0 9 1 4
 2 7 5 5
Содержимое массива b
 0 9 1 4
 2 7 5 5

Статические массивы

1. Понятие
2. Заполнение значениями;
3. **Вычисления;**
4. Поиск;
5. Вычислительная сложность алгоритмов;
6. Сортировка.

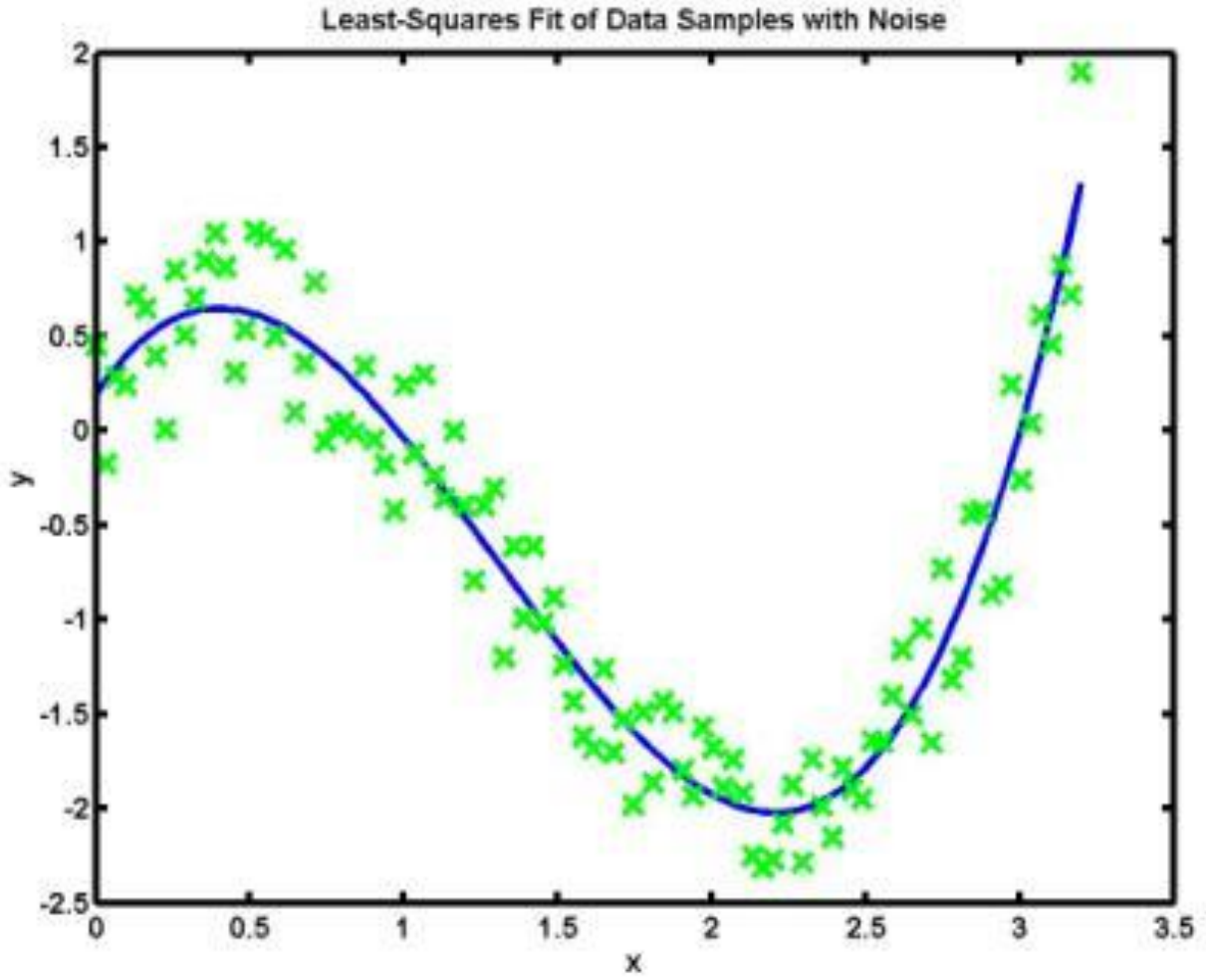
Статические массивы

Задача вычисления:

- среднее арифметическое;
- среднее геометрическое;
- аппроксимация;
- интерполяция;
- и другие математические вычисления

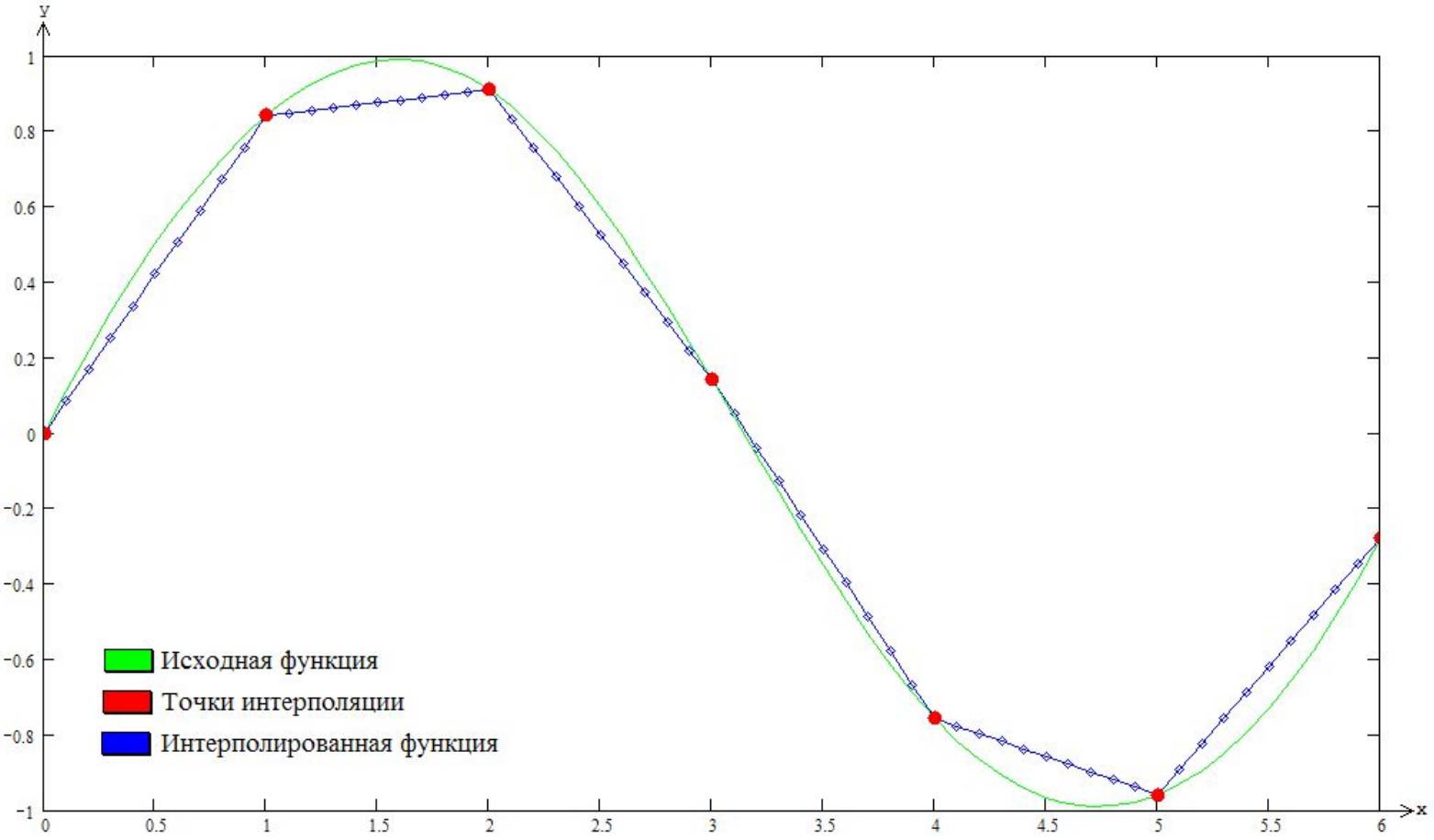
Статические массивы

Аппроксимация – приближенное описание сложной функции более простыми



Статические массивы

Интерполяция - нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений



Статические массивы

Пример: вычислить значение функции в точке заданной таблично

Алгоритм решения:

1. Задать значения функции в виде точек (массив аргументов, массив значений)
2. Задать точку, для которой вычисляется значение функции.
3. Найти в массиве аргументов ближайший к заданной точке.
4. Вычисление коэффициентов полинома первого порядка (y=kx+b)
5. Вычисление значения функции в заданной точке.

Статические массивы

```
const
  N = 5;
var
  mx : array [1..N] of real;    // массив аргументов
  my : array [1..N] of real;    // массив значений
  x,y : real;                   // точка, для которой вычисляется
  k,b : real;                   // коэффициенты полинома
  i : integer;                  // счетчик цикла
  findPoint: boolean;           // маркер поиска точки
```

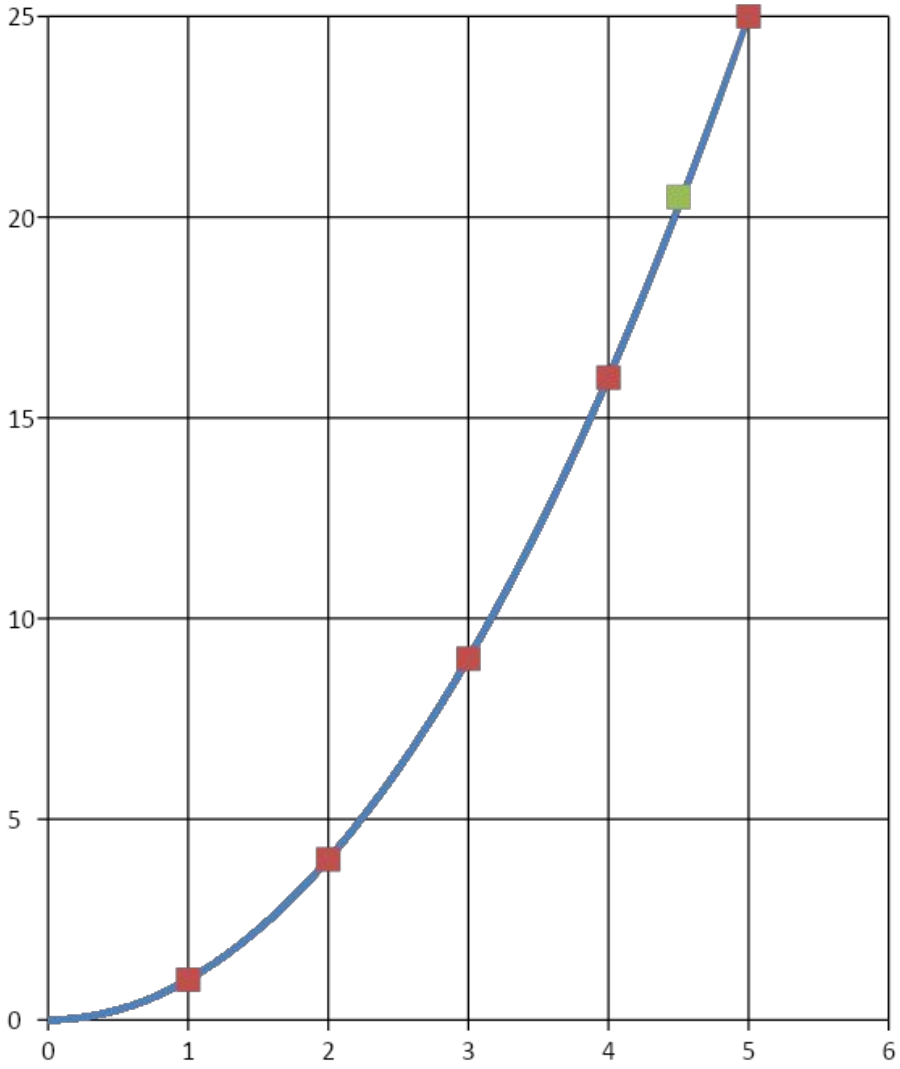
Статические массивы

```
begin
  writeln('Задайте функцию в виде таблицы');
  for i := 1 to N do
    begin
      write('Точка ',i,' (x,y) ');
      read(mx[i], my[i]);
    end;
  write('Введите значение аргумента, для которого необходимо вычислить
значение функции, x=');
  readln(x);
```

Статические массивы

```
//поиск ближайшего аргумента
  i := 1; findPoint := false;
    while not findPoint do // поиск ближайшего аргумента
    begin
     if (x >= mx[i]) and (x < mx[i+1]) then
         begin // ближайший аргумент найден
           findPoint := true;
           k := (my[i+1] - my[i])/(mx[i+1] - mx[i]);
           b := my[i] - k*mx[i];
           y := k*x+b;
           writeln('Значение функции в точке x=',x,' равно y=',y:0:2);
         end;
    if i = N-1 then // точки кончились
        findPoint := true;
              else
                 i := i + 1;
    end;
end.
```

Статические массивы



x= 4.5
Точное значение y=20.25
Интерполированное значение y=20.5
Ошибка – 1.23%

Статические массивы

1. Понятие
2. Заполнение значениями;
3. Вычисления;
4. **Поиск;**
5. Вычислительная сложность алгоритмов;
6. Сортировка.

Статические массивы

- поиск min/max
- поиск по заданному условию
- поиск цепочек

Статические массивы

Пример: Найти и вывести самую длинную цепочку четных элементов массива

Алгоритм решения:

1. Заполнить массив случайными числами
2. Поиск подряд идущих четных элементов
3. Определение самой длинной из них

Статические массивы

```
const
  N = 20;
var
  m : array [1..N] of integer;
  i : integer;
  c, l : integer;   //позиции и длина текущей цепочки
  cm, lm : integer; //позиции и длина самой длинной цепочки
  flag1, flag2 : boolean; // флаги цепочки
begin
  for i := 1 to N do
    begin //заполним и распечатаем массив
      m[i] := random(10 + 1);
      write(m[i],' ');
    end;
Writeln;
```

Статические массивы

```
// зададим начальные значения для текущий
 // позиций цепочки и их макс знач.
 c := 0; l := 0;
 cm := 0; lm := 0;
 flag1 := false; flag2 := false;
 for i := 2 to N do
  begin
   flag2 := (m[i-1] mod 2=0) and (m[i] mod 2=0); // два соседних элемента четные
   if flag2 then
      begin
         if not flag1 then begin c := i-1; l := 2; end //два подряд четных, а пред - нет
            else l := i - c + 1; //больше двух подряд четных
         if l > lm  // длина текущей цепочки больше предыдущей
               then cm := c; lm := l;
      end
     else
      begin
       // сброс параметров текущей цепи в нуль
       l := 0; c := 0;
      end;
   flag1 := flag2;
   writeln(c,'-',l,'|',cm,'-',lm);
  end;
```

Статические массивы

```
if cm <> 0 then
     for i := cm to cm+lm-1 do write(m[i],' ')
  else writeln('Цепочек нет')
end.
```

Позиция	1	2	3	4	5	6	7	8	9	10
Значение	2	8	8	5	1	9	6	6	10	6
Два подряд четных	нет	да	да	нет	нет	нет	Нет	да	да	да
Предыдущие два подряд четных	нет	нет	да	да	нет	нет	Нет	нет	да	да
Текущая цепь	0	1	1	0	0	0	0	7	7	7
Длина текущей цепи	0	2	3	0	0	0	0	2	3	4
Макс цепь	0	1	1	1	1	1	1	1	1	7
Длина макс цепи	0	2	3	3	3	3	3	3	3	4

Статические массивы

Пример: Найти точку на плоскости, ближайшую заданной

Алгоритм решения:

1. Заполнить массивы координатами точек
2. Задать координаты заданной точки
3. Обойти все исходные точки, определить расстояние до каждой их них
4. Запомнить минимальное расстояние

Статические массивы

```
const
  N = 5; //кол-во точек на плоскости
var
  px,py : array [1..N] of real; // координаты исходных точек
  x,y : real;
  i, imin : integer;
  L, Lmin : real; //текущее и минимальное расстояние
begin
writeln('Исходные точки');
  for i := 1 to N do // заполняем координаты точек
      begin
        px[i] := 5 - 10*random;
        py[i] := 5 - 10*random;
        writeln('(',px[i]:5:2,',',py[i]:5:2,')');
      end;
  writeln('Введите координаты точек');
  write('x=');readln(x);
  write('y=');readln(y);
```

Статические массивы

```
  Lmin := sqrt(sqr(x-px[1]) + sqr(y-py[1]));
  imin := 1;
  writeln('Точка #1 расстояние ',Lmin:5:2);
  for i := 2 to N do
     begin
       L := sqrt(sqr(x-px[i]) + sqr(y-py[i]));
       if Lmin > L then begin Lmin := L; imin := i; end;
       writeln('Точка #',i,' расстояние ',L:5:2);
     end;
  writeln('Ближайшая точка #',imin,' расстояние ',Lmin:5:2);
end.
```

Статические массивы

(-4.07,-0.33)
(1.92, 2.72)
(-3.72, 4.01)
(-4.19,-4.36)
(-1.67, 2.33)
Введите координаты точек
x=0
y=0
Точка #1 расстояние 4.08
Точка #2 расстояние 3.33
Точка #3 расстояние 5.47
Точка #4 расстояние 6.05
Точка #5 расстояние 2.87
Наиближайшая точка #5 расстояние 2.87

Статические массивы

1. Понятие
2. Заполнение значениями;
3. Вычисления;
4. Поиск;
5. **Вычислительная сложность алгоритмов;**
6. Сортировка.

Массивы. Вычислительная сложность

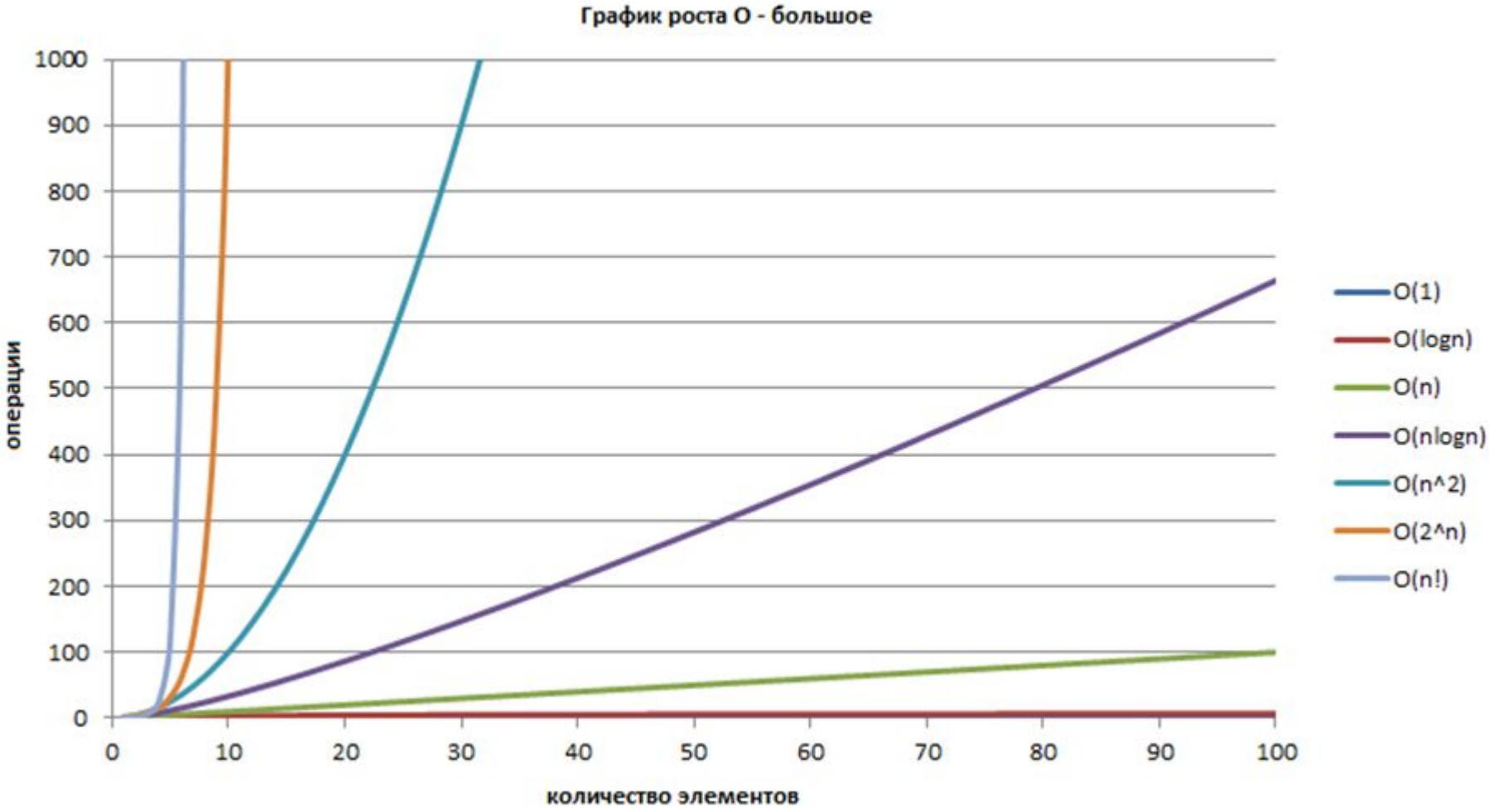
Вычислительная сложность алгоритма — это оценка количества операций, требуемых для достижения результата, в зависимости от количества обрабатываемых элементов.

O(…) – вычислительная сложность «О-большое», значит не хуже чем …

Примеры:

- O(1) – операция взятия значения по индексу
- O(n) – поиск в массиве конкретного значения методом последовательного перебора
- O(logN) – бинарный поиск
- O(N^2) – поиск пары самых близко расположенных точек на плоскости
- O(2^N), O(N!) – unreal

Массивы. Вычислительная сложность



Массивы. Вычислительная сложность

Время вычисления при млн. операций в секунду

Сложность	N=10	N=50	N=100	N=500	N=1000
N log(N)	0.00001	0.00008	0.0002	0.0013	0.003
N^2	0.0001	0.0025	0.01	0.25	1
2^N	0.001	35 лет	10^16 лет	10^137 лет	10^287 лет
N!	3.6	10^50 лет	10^144 лет	Очень много	Нереально много

Массивы. Вычислительная сложность

Какова верхняя оценка O алгоритма поиска минимального числа?

1. начало; поиск минимального элемента массива array из N элементов
2. min := array[1]
3. для i от 2 до N выполнять:
4. если array[i] < min
5. min := array[i]
6. конец; вернуть min

1. N – 1 операция присваивания счетчику цикла i нового значения;
2. N – 1 операция сравнения счетчика со значением N;
3. N – 1 операция сравнения элемента массива со значением min;
4. от 1 до N операций присваивания значения переменной min.

$$O(f(N)) = O(N-1+N-1+N-1+N+1) = O(4N-2) = O(N)$$

Массивы. Вычислительная сложность

Обработка массива размером $N=10^8$

Вариант	**1**	**2**
Сложность	C_1n^2	C_2nlog n
реализация	Ассемблер $C_1 = 2$	Javascript $C_2 = 100$
Вычислитель	Суперкомпьютер	Смартфон
Скорость	10^{11} операций/с	10^8 операций/с
Время		
	Более двух дней	44 минуты

Статические массивы

1. Понятие
2. Заполнение значениями;
3. Вычисления;
4. Поиск;
5. Вычислительная сложность алгоритмов;
6. **Сортировка.**

Статические массивы

Задача сортировки:
для заданной последовательности

$$a_1, a_2, \ldots, a_n$$

найти перестановку её элементов в таком порядке

$$a'_1, a'_2, \ldots, a'_n,$$

что при заданной функции f справедливо отношение:

$$f(a'_1) \le f(a'_2) \le \ldots \le f(a'_n)$$

f(x) – функция упорядочивания, её значение называется ключом

Типы сортировки:

- внутренний (все элементы находятся в памяти машины);
- внешний (часть в памяти, часть на внешних носителях);
- устойчивая сортировка (не меняет относительный порядок сортируемых элементов, имеющих одинаковые ключи)

Основные характеристики:

- время;
- дополнительный объем памяти (без привлечения, с привлечением, необходимо копировать массив).

Сортировка простыми обменами (пузырьком)

564 41 165 815 685 764 827

41 564 165 685 815 764 827

41 165 564 685 764 815 827

41 165 564 685 764 815 827

41 165 564 685 764 815 827

41 165 564 685 764 815 827

41 165 564 685 764 815 827

Статические массивы

```
const
   N = 10;
var
   arr: array[1..N] of integer;
   i, j, k: integer;
begin
   write ('Исходный массив: ');
   for i := 1 to N do begin
      arr[i] := random(101);
      write (arr[i]:4);
   end;
   writeln; writeln;
   for i := 1 to N-1 do
      for j := 1 to N-i do
         if arr[j] > arr[j+1] then begin
            k := arr[j];
            arr[j] := arr[j+1];
            arr[j+1] := k
         end;
    write ('Отсортированный массив: ');
   for i := 1 to N do write (arr[i]:4);
   writeln;
end.
```

Сортировка выбором

1) поиск номера минимального значения в текущем списке;
2) обмен этого значения со значением первой не отсортированной позиции 3) сортируется хвост списка, исключив из рассмотрения уже отсортированные элементы

195 700 949 274 444 108 698

108 700 949 274 444 195 698

108 195 949 274 444 700 698

108 195 274 949 444 700 698

108 195 274 444 949 700 698

108 195 274 444 698 700 949

108 195 274 444 698 700 949

Сортировка вставками

10 3 335 33 355 217 536

3 10 335 33 355 217 536

3 10 335 33 355 217 536

3 10 33 335 355 217 536

3 10 33 335 355 217 536

3 10 33 217 335 355 536

3 10 33 217 335 355 536

Статические массивы

Другие методы:

- шейкер-сортировка;
- метод Шелла;
- быстрая сортировка Хора;
- быстрая сортировка (quicksort);
- и тд

Статические массивы

Алгоритм	Структура данных	Временная сложность		
		Лучшее	В среднем	В худшем
Быстрая сортировка	Массив	O(n log(n))	O(n log(n))	O(n^2)
Сортировка слиянием	Массив	O(n log(n))	O(n log(n))	O(n log(n))
Пирамидальная сортировка	Массив	O(n log(n))	O(n log(n))	O(n log(n))
Пузырьковая сортировка	Массив	O(n)	O(n^2)	O(n^2)
Сортировка вставками	Массив	O(n)	O(n^2)	O(n^2)
Сортировка выбором	Массив	O(n^2)	O(n^2)	O(n^2)
Блочная сортировка	Массив	O(n+k)	O(n+k)	O(n^2)
Поразрядная сортировка	Массив	O(nk)	O(nk)	O(nk)

Хорошо	Приемлемо	Плохо