Программирование (Паскаль)

Программирование (Паскаль)

§ 19. Символьные строки

Что такое символьная строка?

Символьная строка – это последовательность символов.

Хочется:

- строка – единый объект
- длина строки может меняться во время работы программы

```
var s: string; { символьная строка }
```

строковый тип

Символьные строки

Присваивание:

```
s:= 'Вася пошёл гулять';
```

```
var s: string;
```

Ввод с клавиатуры:

```
readln(s);
```

ввод до конца строки

Вывод на экран:

```
writeln(s);
```

Длина строки:

```
var n: integer;
n:= Length(s);
```

Сравнение строк

```
var s: string;
...
writeln('Введите пароль: ');
readln(s);
if s='sEzAm' then
  write('Слушаюсь и повинуюсь!')
else
  write('Пароль неправильный');
```

? Какой правильный пароль?

? Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

Сравнение строк

```
var s1, s2: string;
...
s1:= 'паровоз';
s2:= 'пароход';
if s1 < s2 then
  write(s1, '<', s2)
else
  if s1 = s2 then
    write(s1, '=', s2)
  else
    write(s1, '>', s2);
```

? Что выведет?

паровоз < пароход

первые отличающиеся буквы

Сравниваем с начала:
паровоз
пароход

! в < х!

«в»: код 1074 «х»: код 1093

Посимвольная обработка строк

```
s[4]:= 'a';
```

Задача. Ввести строку и заменить в ней все буквы «э» на буквы «е».

```
var i: integer;
...
for i:=1 to length(s) do
  if s[i]='э' then
     s[i]:='e';
```

для каждого символа строки

Задачи

«A»: Напишите программу, которая заменяет в символьной строке все точки на нули и все буквы X на единицы.

Пример:

```
Введите строку: ..X.XX
Двоичный код: 0010110
```

«B»: Напишите программу, которая выполняет инверсию битов в символьной строке: заменяет в ней все нули на единицы и наоборот.

Пример:

```
Введите строку: 10011010
Инверсия: 01100101
```

Задачи

«С»: Введите битовую строку и дополните её последним битом, который должен быть равен 0, если в исходной строке чётное число единиц, и равен 1, если нечётное (в получившейся строке должно всегда быть чётное число единиц).

Пример:

```
Введите битовую строку: 01101010110
Результат: 011010101100
```

Операции со строками

Объединение *(конкатенация)* :

```
s1:= 'Привет';
s2:= 'Вася';
s  := s1 + ', ' + s2 + '!';
```

'Привет, Вася!'

Срез (выделение части строки):

```
s:= '123456789';
s1:= copy(s,3,5); { '34567' }
```

с какого символа

сколько символов

Операции со строками

Удаление:

```
s:= '123456789';
delete(s, 3, 6); { '129' }
```

с какого символа

сколько символов

Вставка:

```
s:= '123456789';
insert('ABC', s, 3) ; { '12ABC3456789' }
```

что

куда

с какого символа

? Процедуры или функции?

Поиск в строках

```
s:= 'Здесь был Вася.';
          что  где

n:= pos('с', s);
if n > 0 then
  write('Номер символа ', n)
else
  write('Символ не найден.');
```

Находит первое слева вхождение подстроки!

Задачи

«A»: Ввести с клавиатуры в одну строку фамилию и имя, разделив их пробелом. Вывести первую букву имени с точкой и потом фамилию.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр
П. Иванов
```

«B»: Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр Семёнович
П.С. Иванов
```

Задачи

«С»: Ввести адрес файла и «разобрать» его на части, разделенные знаком `'/'`. Каждую часть вывести в отдельной строке.

Пример:

```
Введите адрес файла:
C:/Фото/2015/Байкал/shaman.jpg
C:
Фото
2015
Байкал
shaman.jpg
```

Преобразования «строка» → «число»

Целое число:

```
var r: integer;
...
s:= '123';
val(s, N, r); { N = 123 }
```

номер первого ошибочного символа

Вещественное число:

```
var r: integer;
...
s:='123.456';
val(s, X, r); { X = 123.456}
```

Преобразования «число» → «строка»

```
n:= 123;
str(N, s); { s = '123' }
x:= 123.456;
str(X, s); { s = '1.234560E+002' }
str(X:10:3, s); { s = ' 123.456' }
```

Как объявить переменные?

Задачи

«A»: Напишите программу, которая вычисляет сумму двух чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3
Ответ: 15
```

«B»: Напишите программу, которая вычисляет сумму трёх чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45
Ответ: 60
```

Задачи

«C»: Напишите программу, которая вычисляет сумму произвольного количества чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45+10
Ответ: 70
```

«D»: Напишите программу, которая вычисляет выражение, содержащее целые числа и знаки сложения и вычитания.

Пример:

```
Введите выражение:
12+134-45-17
Ответ: 84
```

Программирование (Паскаль)

§ 20. Обработка массивов

Обработка потока данных

Задача. С клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено положительных чисел.

1) нужен счётчик
2) счётчик увеличивается
3) нужен цикл
4) это цикл с условием (число шагов неизвестно)

Когда увеличивать счётчик?

Какой цикл?

```
счётчик = 0
пока не введён 0:
   если введено число > 0 то
      счётчик:= счётчик + 1
```

Обработка потока данных

```
var x, count: integer;
count:=0;
read( x );
while x <> 0 do begin
  if x > 0 then
    count:= count + 1;
  read( x );
end;
writeln( count );
```

откуда взять **x**?

? Что плохо?

Найди ошибку!

```
var x, count: integer;
count:=0;
read( x );
while x <> 0 do begin
   if x > 0 then
      count:= count + 1;
er read( x );
writeln( count );
```

Найди ошибку!

```
var x, count: integer;
count:=0;
while x = 0 do begin
   if x <> then
      count:= count + 1;
   read( x );
end;
writeln( count );
```

Обработка потока данных

Задача. С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

1) нужна переменная для суммы
2) число добавляется к сумме, если оно заканчивается на "5"
3) нужен цикл с условием

Как это записать?

```
сумма := 0
пока не введён 0:
  если число оканчивается на "5" то
     сумма:= сумма + число
```

```
if x mod 10 = 5 then
```

Обработка потока данных

Задача. С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

```
var x, sum: integer;
sum:=0;
read( x );
while x <> 0 do begin
  if x mod 10 = 5 then
    sum:= sum + x;
  read( x )
end;
writeln( sum );
```

Чего не хватает?

Найди ошибку!

```
var x, sum: integer;
sum:=0;
read( x ); 0 do begin
   if x mod 10 = 5 then
      sum:= sum + x;
   read( x )
end;
writeln( sum );
```

Задачи

«A»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.

«B»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.

Задачи

«C»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти среднее арифметическое всех двузначных чисел, которые делятся на 7.

«D»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.

Перестановка элементов массива

? Как поменять местами значения двух переменных *a* и *b*?

вспомогательная переменная

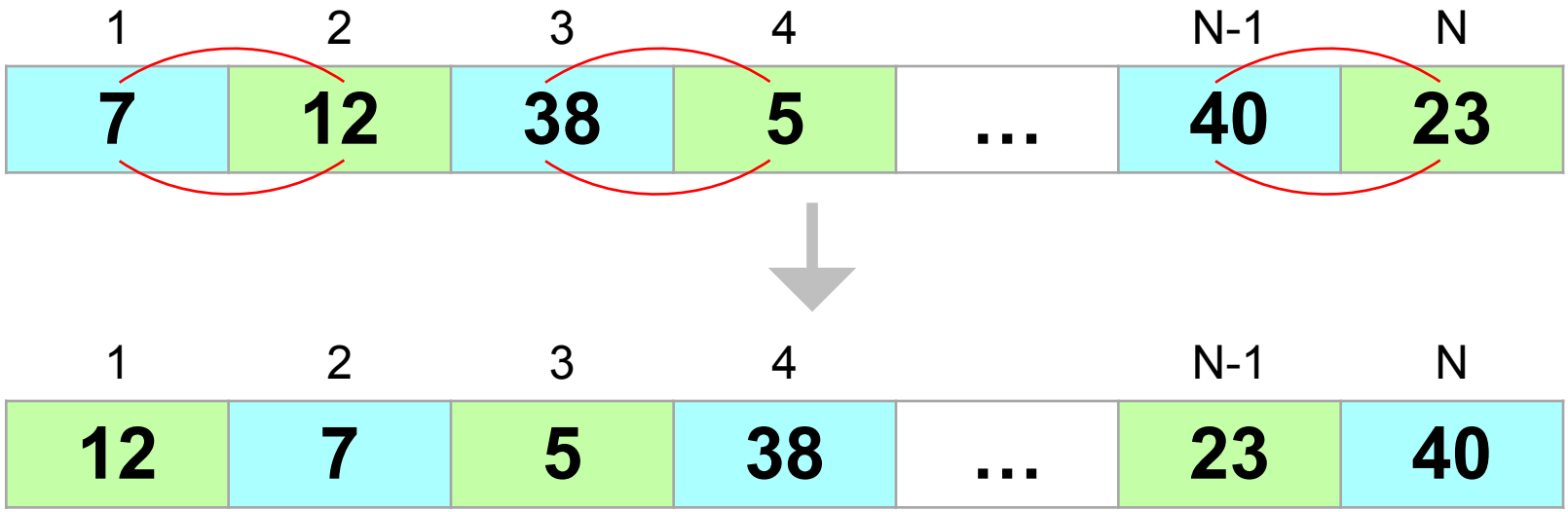
```
c:= a;
a:= b;
b:= c;
```

элементы массива:

```
c:= A[i];
A[i]:= A[k];
A[k]:= c;
```

Перестановка пар соседних элементов

Задача. Массив A содержит чётное количество элементов N. Нужно поменять местами пары соседних элементов: первый со вторым, третий — с четвёртым и т. д.



Перестановка пар соседних элементов

```
for i:=1 to N do begin
  поменять местами A[i] и A[i+1]
end;
```

? Что плохо?

1	2	3	4	5	6
7	12	38	5	40	23
12	7	38	5	40	23
12	38	7	5	40	
12	38	5	7	40	23
12	38	5	40	7	23
12	38	5	40	23	7

выход за границы массива

?

Перестановка пар соседних элементов

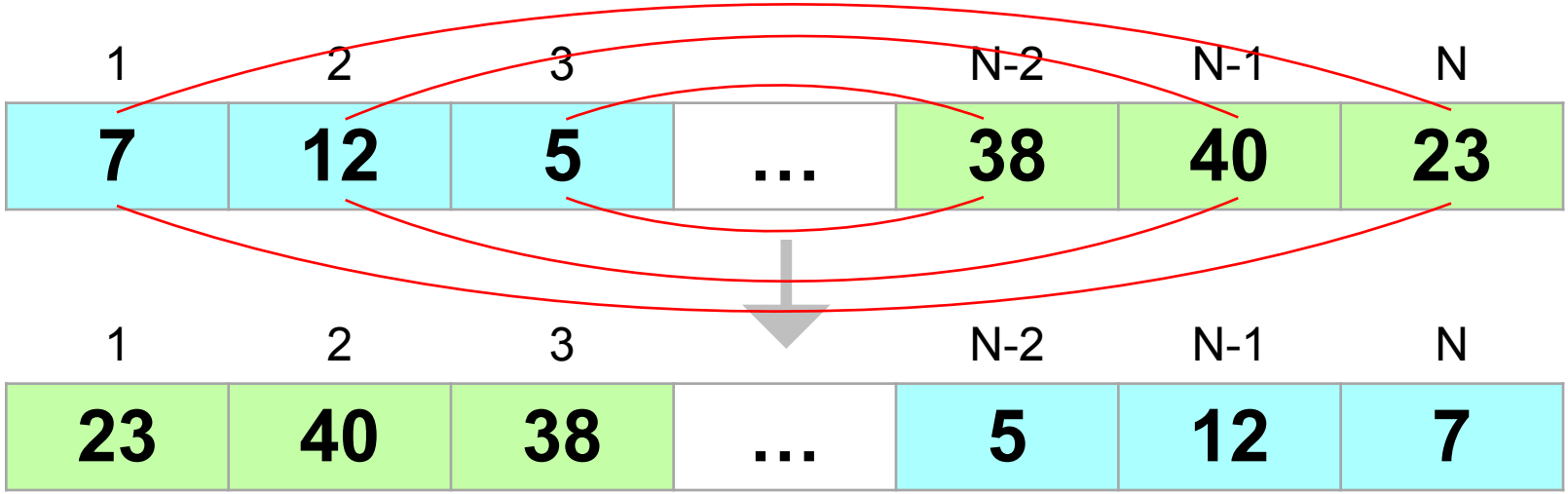
не выходим за границу

```
i:= 1;
while i < N do begin
   { переставляем A[i] и A[i+1] }
   c:= A[i];
   A[i]:= A[i+1];
   A[i+1]:= c;
   i:= i + 2     { к следующей паре }
end;
```

A[1]↔A[2], A[3]↔A[4], …, A[N-1]↔A[N]

Реверс массива

Задача. Переставить элементы массива в обратном порядке (выполнить *реверс*).




```
A[1]↔A[N]          1+N    = N+1
A[2]↔A[N-1]        2+N-1  = N+1
A[i]↔A[N+1-i]      i+???  = N+1
A[N]↔A[1]          N+1    = N+1
```

Реверс массива

```
for i:=1 to N div 2 do begin
  поменять местами A[i] и A[N+1-i]
end;
```

? Что плохо?

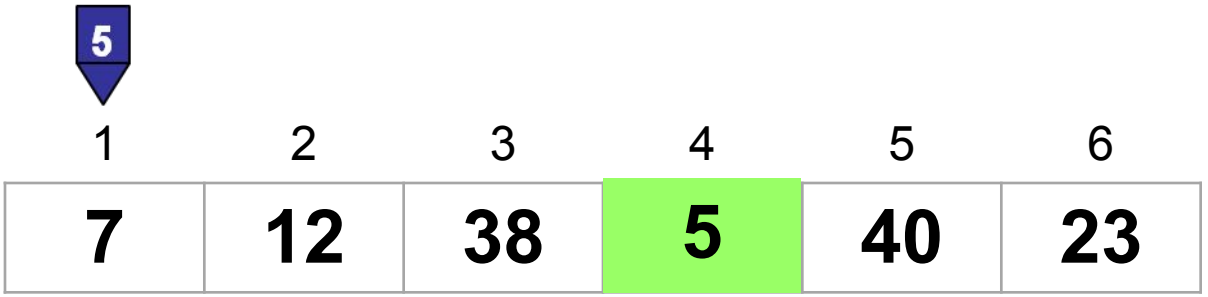
1	2	3	4	
7	12	40	23	i=1
23	12	40	7	i=2
23	40	12	7	i=3
23	12	40	7	i=4
7	12	40	23	

? Как исправить?

Линейный поиск в массиве

Задача. Найти в массиве элемент, равный X, и его номер.

X = 5




```
i:=1;
while A[i]<>X do
  i:= i + 1;
writeln('A[',i,']=',X);
```

? Что плохо?

? Если искать 4?

! Нельзя выходить за границы массива!

Линейный поиск в массиве

не выходим за границу

```
i:=1;
while (i<=N) and (A[i]<>X)
   i:= i + 1;
if i<=N then
   writeln( 'A[',i,']=',X )
else
   writeln( 'Не нашли!');
```

? Как проверить, нашли или нет?

Досрочный выход из цикла

Задача. Найти в массиве элемент, равный X, и его номер.

```
var nX: integer;
```

```
nX:=0; { номер элемента }
for i:=1 to N do
  if A[i]=X then begin
    nX:= i; { запомнить номер }
    break
  end;
if nX > 0 then
  writeln( 'A[',nX,']=', X )
else
  writeln( 'Не нашли!');
```

нашли!

сразу выйти из цикла

? Как объявить **nX**?

Задачи

«А»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20], выводит его на экран, а затем находит индекс первого элемента, равного введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Что ищем: 13
A[4] = 13
```

Задачи

«B»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [-10,10], выводит его на экран, а затем находит индекс **последнего** элемента, равного введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: -5 -6 2 3 -3 0 8 -3 0 9
Что ищем: 0
A[9] = 0
```

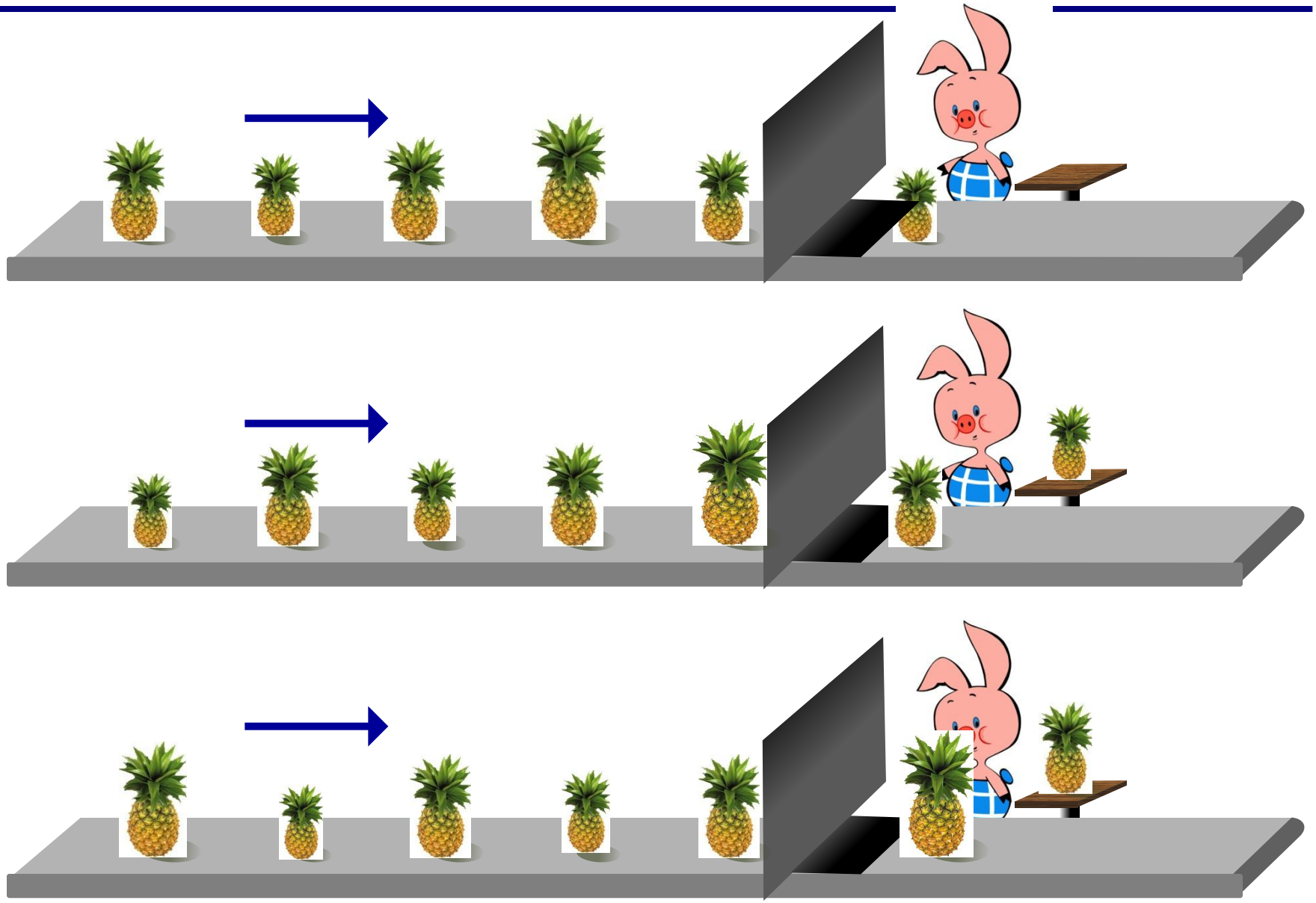
Задачи

«С»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [10,50], выводит его на экран, а затем находит индексы всех элементов, равных введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: 12 45 30 18 30 15 30 44 32 17
Что ищем: 30
A[3] = 30
A[5] = 30
A[7] = 30
```

Поиск максимального элемента



Поиск максимального элемента

? Какие переменные нужны?

```
for i:=1 to N do
  if A[i] > M then
    M:=A[i];
writeln( M );
```

? Чего не хватает?

? Какое начальное значение взять для **M**?

1) **M** – значение, которое заведомо меньше всех элементов массива

или

2) **M = A[1]** (или любой другой элемент)

максимальный не меньше, чем **A[1]**

Поиск максимального элемента

```
M:= A[1];
for i:=2 to N do
  if A[i] > M then
    M:= A[i];
writeln( M );
```

начинаем с **A[2]**, так как **A[1]** мы уже посмотрели

? Как найти минимальный?

Номер максимального элемента

Задача. Найти в массиве максимальный элемент и его номер.

? Какие переменные нужны?

```
M:= A[1]; nMax:= 1;
for i:=2 to N do
  if A[i] > M then begin
    M:= A[i];
    nMax:= i
  end;
writeln( 'A[', nMax, ']=', M);
```

? Можно ли убрать одну переменную?

Номер максимального элемента

Если знаем **nMax**, то **M=A[nMax]**!

```
M:= A[1]; nMax:= 1;
for i:=2 to N do
  if A[i]> A[nMax] then begin
    M:= A[i];
    nMax:= i
  end;
writeln( 'A[', nMax, ']=', A[nMax] );
```

Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M:= A[1];
for i:=2 to N do
  if (A[i]<0) and (A[i]>M) then
    M:=A[i];
writeln( M );
```

? Что плохо?

? Как исправить?

1	2	3	4	5
5	–2	8	3	–1

M = 5

Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M:= A[1];
for i:=2 to N do
  if A[i]<0 then
    if (M>=0) or (A[i]>M) then
      M:=A[i];
writeln( M );
```

сначала записали неотрицательный!

? Если нет отрицательных?

Задачи

«A»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [50; 150] и находит в нём минимальный и максимальный элементы и их номера.

«B»: Напишите программу, которая получает с клавиатуры значения элементов массива и выводит количество элементов, имеющих максимальное значение.

«C»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [100; 200] и находит в нём пару соседних элементов, сумма которых минимальна.

Задачи

«D»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [–100; 100] и находит в каждой половине массива пару соседних элементов, сумма которых максимальна.

Задачи-2 (максимум в потоке)

«A»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное и максимальное среди полученных чисел.

«B»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное число, делящееся на 3, среди полученных чисел.

«C»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Напишите программу, которая находит максимальное двузначное число, заканчивающееся на 6, среди полученных чисел.

Задачи-2 (максимум в потоке)

«D»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Напишите программу, которая находит среди полученных чисел пару полученных друг за другом чисел, сумма которых максимальна.

Сортировка

Сортировка — это расстановка элементов списка (массива) в заданном порядке.

Задача. Отсортировать элементы в порядке **возрастания** (*неубывания* – если есть одинаковые).

Алгоритмы сортировки:

- простые, но медленные (при больших N)
- быстрые, но сложные…

Сортировка выбором

? Где должен стоять минимальный элемент?

- нашли минимальный, поставили его на первое место

```
c:= A[nMin];
A[nMin]:= A[1];
A[1]:= c;
```

? Как?

? Что дальше?

- из оставшихся нашли минимальный, поставили его на второе место и т.д.

5	–2	8	3	–1
–2	5	8	3	–1
–2	–1	8	3	5

Сортировка выбором

```
for i:=1 to N-1 do begin
    { ищем минимальный среди A[i]..A[N] }
  nMin:= i;
  for j:=i+1 to N do
    if A[j] < A[nMin] then
      nMin:= j;
  { переставляем A[i] и A[nMin] }
  c:=A[i];
  A[i]:=A[nMin];
  A[nMin]:=c;
end;
```

не трогаем те, которые уже поставлены

? Почему цикл до N-1?

Задачи

«A»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20] и сортирует его в порядке убывания.

Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Сортировка: 18 16 16 14 13 13 9 5 3 2
```

«B»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [10,100] и сортирует его по возрастанию последней цифры числа (сначала идут все числа, которые заканчиваются на 0, потом все, которые заканчиваются на 1, и т.д.).

Пример:

```
Массив: 12 10 31 40 55 63 28 87 52 92
Сортировка: 10 40 31 12 52 92 63 55 87 28
```

Задачи

«С»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20] и сортирует его в порядке возрастания. На каждом шаге цикла выполняется поиск максимального (а не минимального!) элемента.

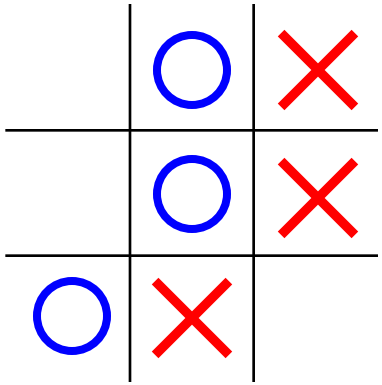
Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Сортировка: 2 3 5 9 13 13 14 16 16 18
```

Программирование (Паскаль)

§ 21. Матрицы (двумерные массивы)

Что такое матрица?



? Как закодировать?

Матрица — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.).

Каждый элемент матрицы имеет два индекса – номера строки и столбца.

Объявление матриц

```
const N = 3, M = 4;
var A: array[1..N, 1..M] of integer;
    X: array[-3..0, -8..M] of real;
    L: a                N, 5 boolean;
```

строки столбцы

строки столбцы

Простые алгоритмы

Заполнение случайными числами:

```
for i:=1 to N do begin
  for j:=1 to M do begin
    A[i,j]:= 20+random(61);
    writeln( A[i,j], ' ')
  end;
  writeln
end;
```

Вложенный цикл!

в одной строке через пробел

следующий – с новой строки

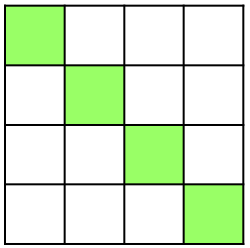
Суммирование:

```
sum:= 0;
for i:=1 to N do
  for j:=1 to M do
    sum:= sum + A[i,j];
```

Перебор элементов матрицы

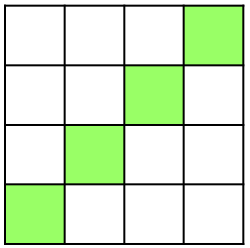
Главная диагональ:

```
for i:=1 to N do begin
  { работаем с  A[i,i] }
end;
```



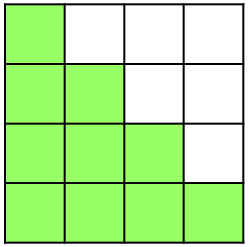
Побочная диагональ:

```
for i:=1 to N do begin
  | работаем с  A[i,N+1-i]
end;
```



Главная диагональ и под ней:

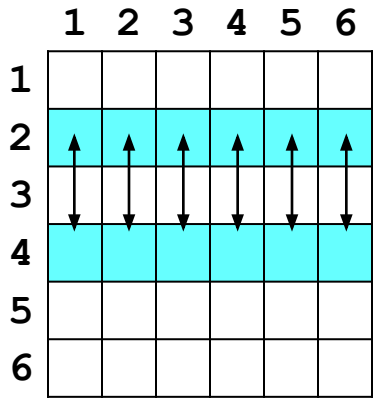
```
for i:=1 to N do begin
  for j:=1 to i do begin
    { работаем с  A[i,j] }
  end;
end;
```



Перестановка строк

2-я и 4-я строки:

```
for j:=1 to M do begin
  c:= A[2,j];
  A[2,j]:= A[4,j];
  A[4,j]:= c
end;
```



Задачи

«А»: Напишите программу, которая заполняет матрицу случайными числами и находит максимальный элемент на главной диагонали квадратной матрицы.

Пример:

```
Матрица A:
12 34 14 65
71 88 23 45
87 46 53 39
76 58 24 92
Результат: A[4,4] = 92
```

Задачи

«B»: Напишите программу, которая заполняет матрицу случайными числами и находит максимальный элемент матрицы и его индексы (номера строки и столбца).

Пример:

```
Матрица A:
12 34 14 65
71 88 23 98
87 46 53 39
76 58 24 92
Максимум: A[2,4] = 98
```

Задачи

«С»: Напишите программу, которая заполняет матрицу случайными числами и находит минимальный из чётных положительных элементов матрицы. Учтите, что таких элементов в матрице может и не быть.

Пример:

```
Матрица A:
16 34 14 65
71 88 23 45
87 12 53 39
76 58 24 92
Результат: A[3,2] = 12
```

Программирование (Паскаль)

§ 22. Сложность алгоритмов

Как сравнивать алгоритмы?

- быстродействие (**временна́я сложность**)
- объём требуемой памяти (**пространственная сложность**)
- понятность

Обычно не бывает все хорошо!

Время работы алгоритма – это количество элементарных операций *T*, выполненных исполнителем.

зависит от количества данных (размера массива *N*)

Функция *T(N)* называется **временно́й сложностью алгоритма**

$T(N) = 2N^3$

Как увеличится время работы при увеличении *N* в 10 раз?

Примеры определения сложности

Задача 1. Вычислить сумму первых трёх элементов массива (при $N \geq 3$).

```
Sum:= A[1] + A[2] + A[3];
```

$T(N) = 3$

2 сложения + запись в память

Задача 2. Вычислить сумму всех элементов массива.

```
Sum:= 0;
for i:=1 to N do
  Sum:= Sum + A[i];
```

$T(N) = 2N + 1$

N сложений, N+1 операций записи

Примеры определения сложности

Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
for i:=1 to N-1 do begin
  nMin:=i;
  for j:=i+1 to N do
    if A[i]<A[nMin] then nMin:=j;
  c:=A[i]; A[i]:=A[nMin]; A[nMin]:=c;
end;
```

Число сравнений:

$$T_c(N) = (N-1) + (N-2) + \ldots + 2 + 1 = \frac{N(N-1)}{2} = \frac{1}{2}N^2 - \frac{1}{2}N$$

Число перестановок: $T_n(N) = N - 1$

Примеры определения сложности

Задача 4. Найти сумму элементов квадратной матрицы размером *N*×*N*.

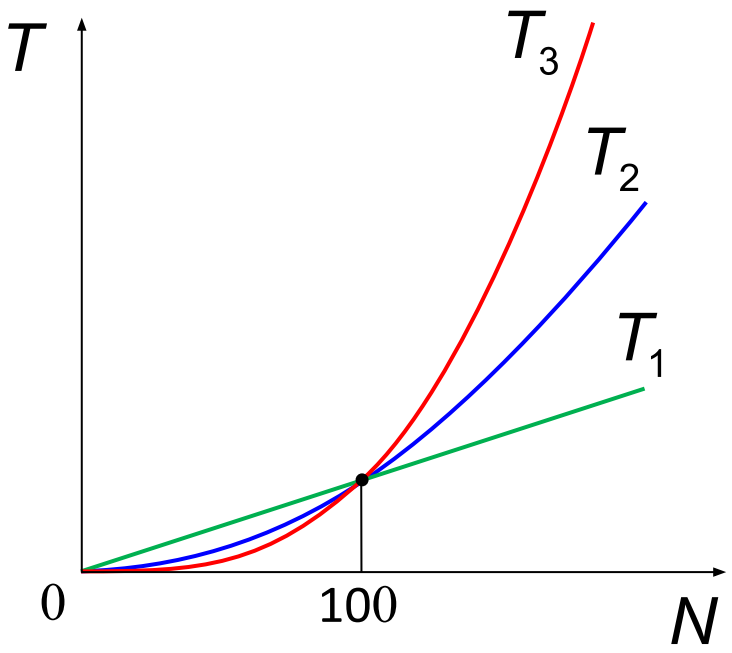
```
Sum:= 0;
for i:=1 to N do
  for j:=1 to N do
    Sum:= Sum + A[i,j];
```

Самостоятельно! ☺

Сравнение алгоритмов по сложности

$T_1(N) = 10000 \cdot N$ $T_2(N) = 100 \cdot N^2$ $T_3(N) = N^3$

? Какой алгоритм выбрать?



при $N < 100$:

$$T_3(N) < T_2(N) < T_1(N)$$

при $N > 100$:

$$T_3(N) > T_2(N) > T_1(N)$$

! Нужно знать размер данных!

Асимптотическая сложность

Асимптотическая сложность – это оценка скорости роста количества операций при больших значениях N.

линейная

постоянная

сложность $O(N)$ $\Leftrightarrow$ $T(N) \le c \cdot N$ для $N \ge N_0$

сумма элементов массива:

$T(N) = 2 \cdot N - 1 \le 2 \cdot N$ для $N \ge 1$ $\Rightarrow$ $O(N)$

квадратичная

сложность $O(N^2)$ $\Leftrightarrow$ $T(N) \le c \cdot N^2$ для $N \ge N_0$

сортировка методом выбора:

$$T_c(N) = \frac{1}{2}N^2 - \frac{1}{2}N \le \frac{1}{2}N^2 \text{ для } N \ge 0 \Rightarrow O(N^2)$$

Асимптотическая сложность

кубичная

сложность $O(N^3)$ ⇔ $T(N) \le c \cdot N^3$ для $N \ge N_0$

сложность $O(2^N)$

сложность $O(N!)$

задачи оптимизации, полный перебор вариантов

Факториал числа *N*: $N! = 1 \cdot 2 \cdot 3 \ldots \cdot N$

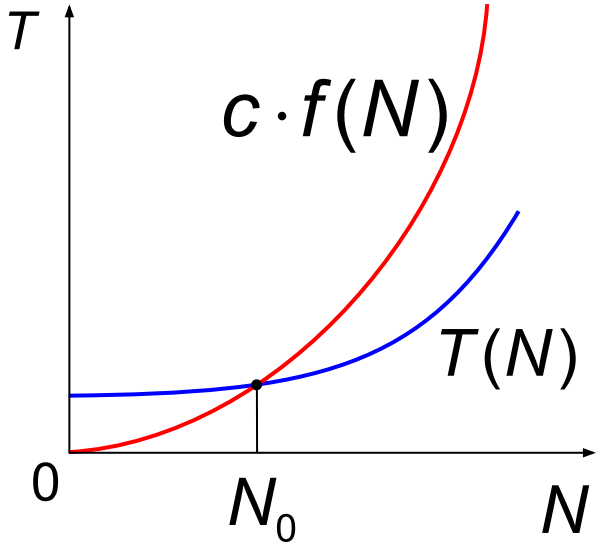
$T(N)$	**время выполнения**
N	100 нс
N^2	10 мс
N^3	0,001 с
2^N	10^{13} лет

N = 100,
1 млрд оп/с

Асимптотическая сложность

Алгоритм относится к классу $O(f(N))$, если найдется такая постоянная c, что начиная с некоторого $N = N_0$ выполняется условие

$$T(N) \le c \cdot f(N)$$



это верхняя оценка!

$$O(N) \Rightarrow O(N^2) \Rightarrow O(N^3) \Rightarrow O(2^N)$$

«Алгоритм имеет сложность $O(N^2)$».

обычно – наиболее точная верхняя оценка!

Программирование (Паскаль)

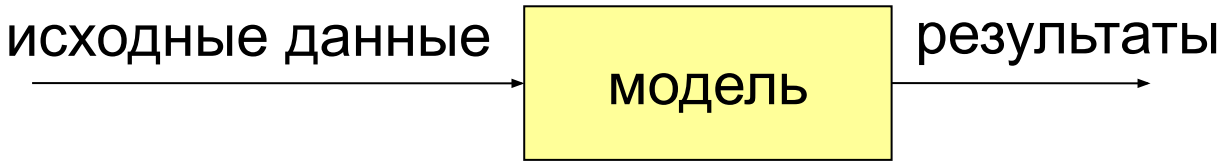
§ 23. Как разрабатывают программы

Этапы разработки программ

I. Постановка задачи

Документ: *техническое задание*.

II. Построение модели



Формализация: запись модели в виде формул (на формальном языке).

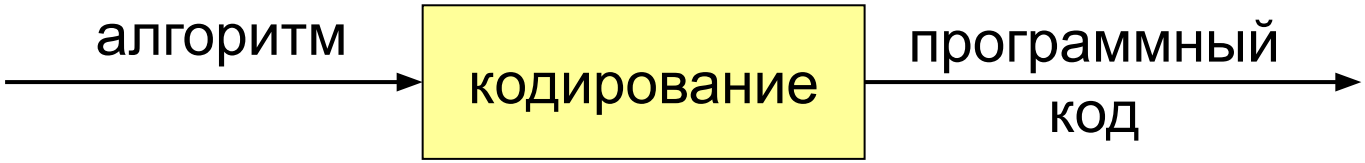
III. Разработка алгоритма и способа хранения данных

«Алгоритмы + структуры данных = программы»
(Н. Вирт)

Этапы разработки программ

IV. Кодирование

Запись алгоритма на языке программирования.



V. Отладка

Поиск и исправление ошибок в программах:

- **синтаксические** – нарушение правил языка программирования
- **логические** – ошибки в алгоритме

могут приводить к **отказам** – аварийным ситуациям во время выполнения *(run-time error)*

Этапы разработки программ

VI. Тестирование

Тщательная проверка программы во всех режимах:

- **альфа-тестирование** – внутри компании (тестировщики)
- **бета-тестирование** – (доверенные) пользователи

VII. Документирование

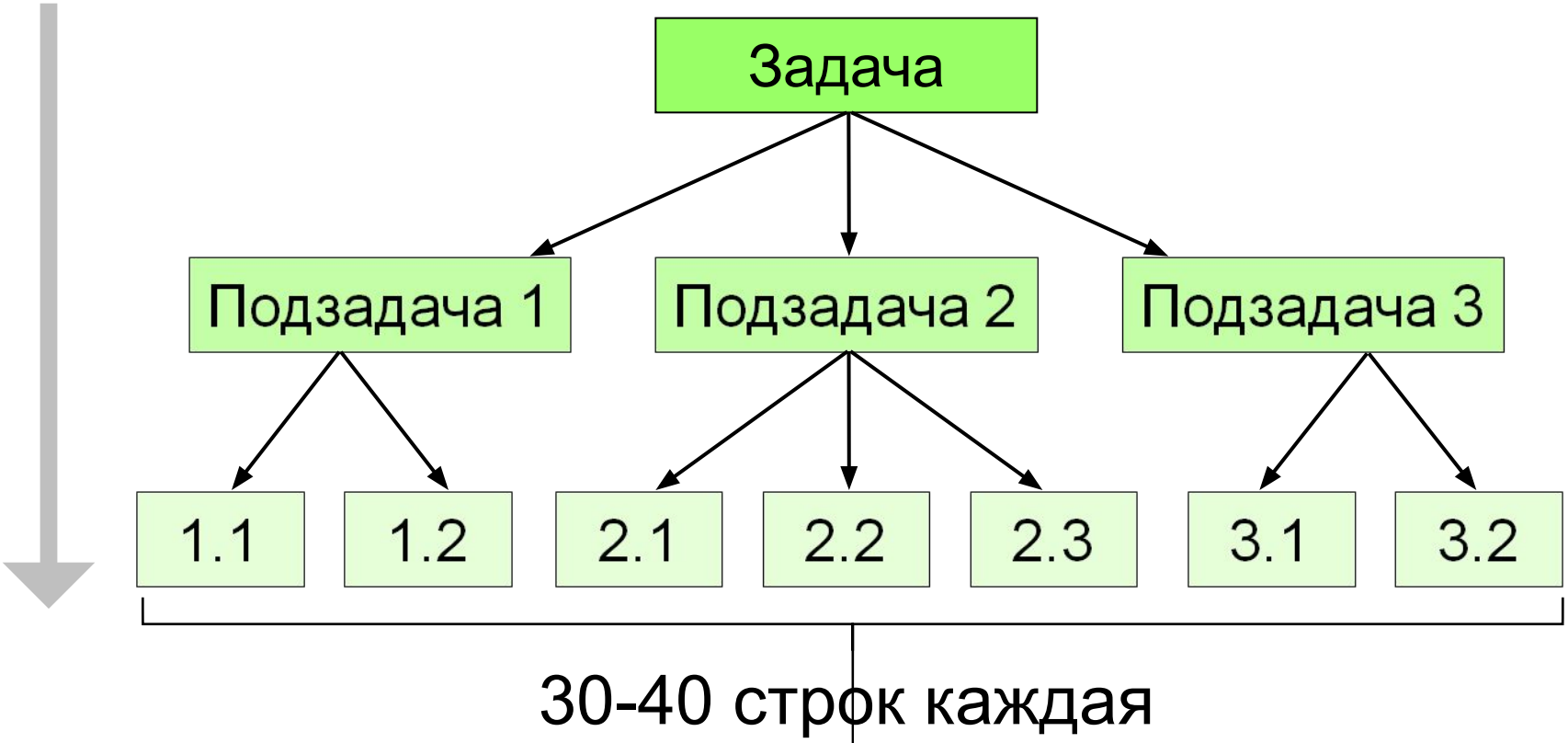
Технические писатели

VIII. Внедрение и сопровождение

- обучение пользователей
- исправление найденных ошибок
- техподдержка

Методы проектирования программ

«Сверху вниз» (последовательное уточнение)



Методы проектирования программ

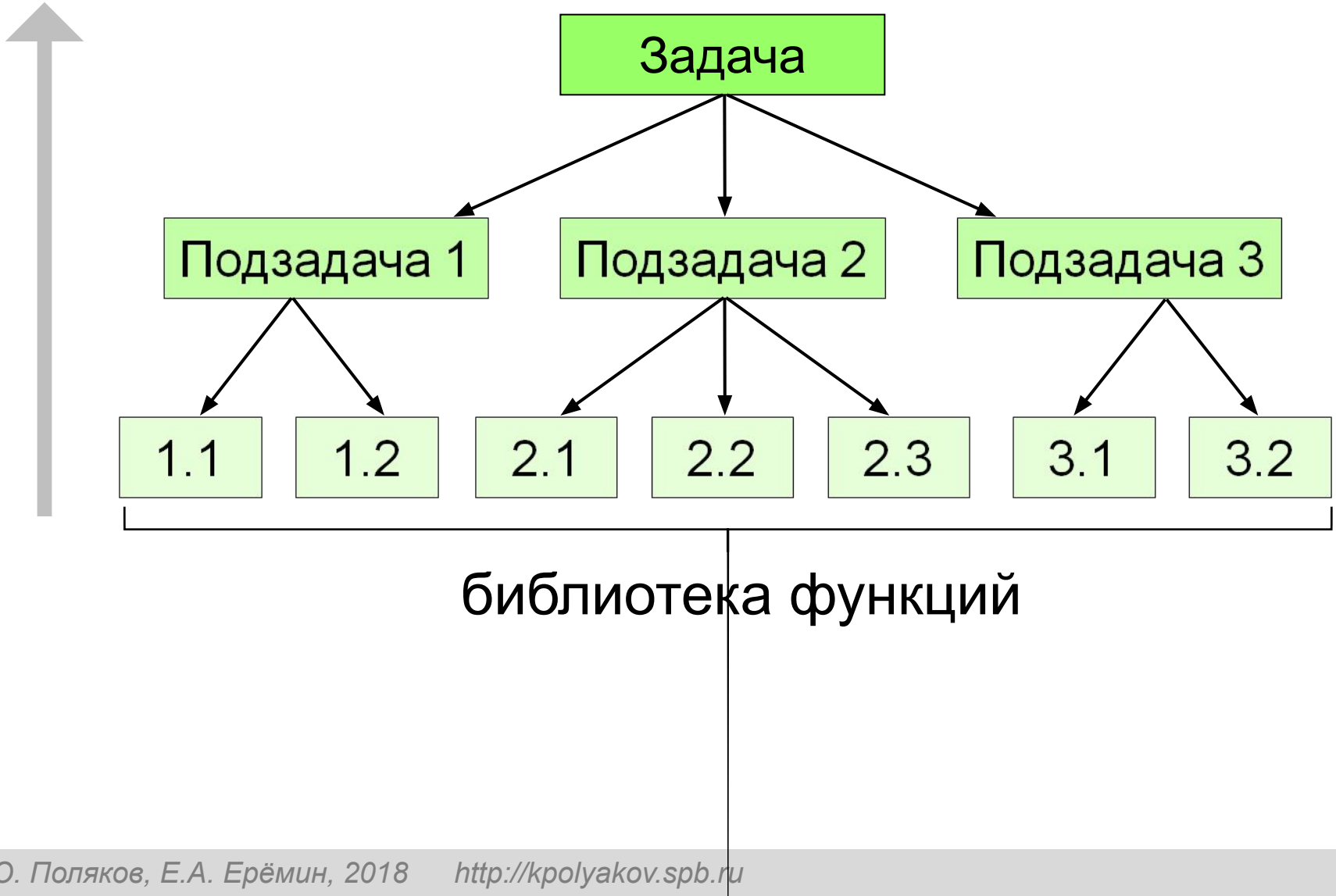
«Сверху вниз» (последовательное уточнение)

- сначала задача решается «в целом»
- легко распределить работу
- легче отлаживать программу (всегда есть полный работающий вариант)

- в нескольких подзадачах может потребоваться решение одинаковых подзадач нижнего уровня
- быстродействие не известно до последнего этапа (определяется нижним уровнем)

Методы проектирования программ

«Снизу вверх» (восходящее)



Методы проектирования программ

«Снизу вверх» (восходящее)

- нет дублирования
- сразу видно быстродействие

- сложно распределять работу
- сложнее отлаживать (увеличение числа связей)
- плохо видна задача «в целом», может быть нестыковка на последнем этапе

Почти всегда используют оба подхода!

Отладка программы

Программа решения квадратного уравнения

$$ax^2 + bx + c = 0$$

```
program SqEq;
var a, b, c, D, x1, x2: real;
begin
  write('Введите a, b, c: ');
  read(a, b, c);
  D:=b*b-4*a*a;
  x1:=(-b+sqrt(D))/2*a;
  x2:=(-b-sqrt(D))/2*a;
  writeln('x1=', x1, ' x2=', x2);
end.
```

Тестирование

Тест 1. $a = 1$, $b = 2$, $c = 1$.

Ожидание:

```
x1=-1.0 x2=-1.0
```

Реальность:

```
x1=-1.0 x2=-1.0
```

Тест 2. $a = 1$, $b = -5$, $c = 6$.

```
x1=3.0 x2=2.0
```

```
x1=4.791 x2=0.209
```

Найден вариант, когда программа работает неверно.
Ошибка **воспроизводится**!

Возможные причины:

- неверный ввод данных
- неверное вычисление дискриминанта
- неверное вычисление корней
- неверный вывод результатов

$$D = b^2 - 4ac$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

Отладочная печать

Идея: выводить все промежуточные результаты.

```
read(a, b, c);
writeln(a, ' ', b, ' ', c);
D:=b*b-4*a*a;
writeln('D=', D);
...
```

Результат:

```
Введите a, b, c: 1 -5 6
1.0 -5.0 6.0
D=21.0
```

$$D = b^2 - 4ac = 25 - 4 \cdot 1 \cdot 6 = 1$$

```
D:=b*b-4*a*c;
```

! Одна ошибка найдена!

Отладка программы

Тест 1. $a = 1$, $b = 2$, $c = 1$.

Ожидание:

```
x1=-1.0 x2=-1.0
```

Реальность:

```
x1=-1.0 x2=-1.0
```

Тест 2. $a = 1$, $b = -5$, $c = 6$.

```
x1=3.0 x2=2.0
```

```
x1=3.0 x2=2.0
```

Программа работает верно?

Тест 3. $a = 8$, $b = -6$, $c = 1$.

```
x1=0.5 x2=0.25
```

```
x1=32.0 x2=16.0
```

```
x1:=(-b+sqrt(D))/(2*a);
x2:=(-b-sqrt(D))/(2*a);
```

Что неверно?

Документирование программы

- назначение программы
- формат входных данных
- формат выходных данных
- примеры использования программы

Назначение:

программа для решения уравнения

$$ax^2 + bx + c = 0$$

Формат входных данных:

значения коэффициентов a, b и c вводятся с клавиатуры через пробел в одной строке

Документирование программы

Формат выходных данных:

значения вещественных корней уравнения; если вещественных корней нет, выводится слово «нет»

Примеры использования программы:

1. Решение уравнения $x^2 - 5x + 6 = 0$

```
Введите a, b, c: 1 -5 6
x1=3 x2=2
```

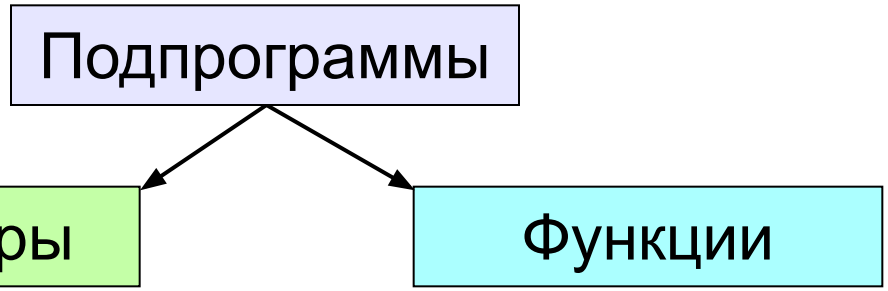
2. Решение уравнения $x^2 + x + 6 = 0$

```
Введите a, b, c: 1 1 6
Нет.
```

Программирование (Паскаль)

§ 24. Процедуры

Два типа подпрограмм



выполняют действия

+ возвращают некоторый результат

? Процедура или функция?

а) рисует окружность на экране
б) определяет площадь круга
в) вычисляет значение синуса угла
г) изменяет режим работы программы
д) возводит число x в степень y
е) включает двигатель автомобиля
ж) проверяет оставшееся количество бензина в баке
з) измеряет высоту полёта самолёта

Простая процедура

```
program withProc;
procedure printLine;
begin
  writeln('----------')
end;
begin
  ...
  printLine;
  ...
end.
```

какие-то операторы

вызов процедуры

Что делает?

- можно вызывать сколько угодно раз
- нет дублирования кода
- изменять – в одном месте

Линии разной длины

```
procedure printLine5;
begin
  writeln('-----')
end;
```

```
procedure printLine10;
begin
  writeln('----------')
end;
```

Как улучшить?

параметр процедуры

```
procedure printLine10;
var i: integer;
begin
```

```
procedure printLine(n: integer);
var i: integer;
begin
  for i:=1 to n do
    write('-');
  writeln
end;
```

Процедура с параметром

```
program withProc;
procedure printLine(n: integer)
begin
  ...
end;
begin
  ...
  printLine(10);
  ...
  printLine(7);
  printLine(5);
  printLine(3);
end;
```

Параметр – величина, от которой зависит работа процедуры.

? Что делает?

Аргумент – значение параметра при конкретном вызове.

Несколько параметров

символьная строка

```
procedure printLine(c: string; n: integer);
var i: integer;
begin
  for i:=1 to n do
    write(c);
  writeln
end;
```

? Что изменилось?

? Как вызывать?

```
printLine( 5, '+' );
```

✓
```
printLine( '+', 5 );
```

✓
```
printLine( '+-+', 5 );
```

В других языках программирования

Python:

```
def printLine (n):
  print('-'*n)
```

C:

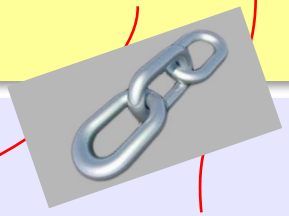
```
void printLine (int n)
{
  int i;
  for (i=1; i<=n; i++)
    putchar('-');
  putchar('\n');
}
```

Как не нужно писать процедуры

```
var x, y: integer;

procedure summa;
begin
  writeln(x+y);
end;

begin
  x := 10; y := 5;
  summa;
end.
```



Что плохо?

- только $x + y$
- не перенести в другую программу

Как исправить?

Как нужно писать процедуры

```
var x, y: integer;
```

```
procedure summa(a, b: integer);
begin
  writeln(a+b);
end;
```

Процедура принимает данные только через параметры!

```
begin
  x := 10; y := 5;
  summa(x, y);
  summa(2*x+y, 15);
end.
```

Задачи

«A»: Напишите процедуру, которая принимает параметр – натуральное число N – и выводит на экран две линии из N символов "–".

Пример:

```
Длина цепочки: 7
-------
-------
```

«B»: Напишите процедуру, которая принимает один параметр – натуральное число *N*, – и выводит на экран прямоугольник длиной *N* и высотой 3 символа.

Пример:

```
Длина прямоугольника: 7
ooooooo
o     o
ooooooo
```

Задачи

«С»: Напишите процедуру, которая выводит на экран квадрат со стороной *N* символов. При запуске программы *N* нужно ввести с клавиатуры.

Пример:

```
Сторона квадрата: 5
ooooo
o   o
o   o
o   o
ooooo
```

Задачи

«D»: Напишите процедуру, которая выводит на экран треугольник со стороной *N* символов. При запуске программы *N* нужно ввести с клавиатуры.

Пример:

```
Сторона: 5
o
oo
ooo
oooo
ooooo
```

Рекурсия

Задача. Вывести на экран двоичный код натурального числа.

```
procedure printBin(n: integer);
begin
...
end;
```

Алгоритм перевода через остатки:

```
while n<>0 do begin
  write(n mod 2);
  n:= n div 2
end;
```

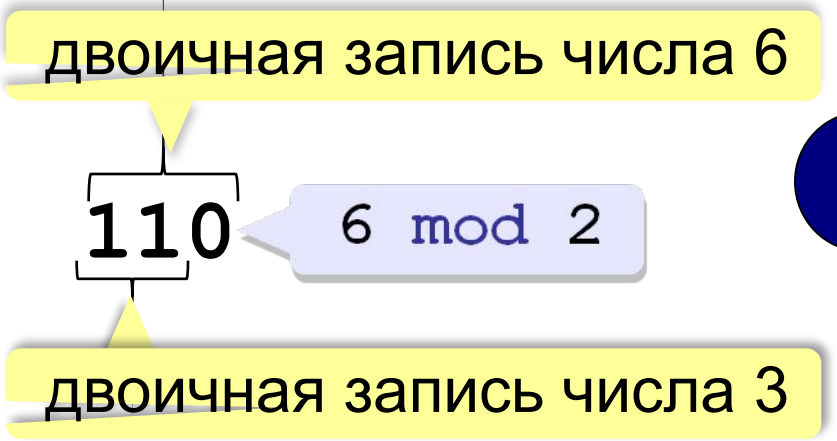
? Что получится при **n = 6**?

011 ✗

в обратном порядке!

Рекурсия

Чтобы вывести двоичную запись числа **n**, нужно сначала вывести двоичную запись числа **(n div 2)**, а затем — его последнюю двоичную цифру, равную **(n mod 2)**.



Чтобы решить задачу, нужно решить ту же задачу для меньшего числа!

Это и есть **рекурсия**!

Чтобы понять рекурсию, нужно понять рекурсию! ☺

Рекурсивная процедура

```
procedure printBin(n: integer);
begin
  printBin(n div 2);
  write(n mod 2)
end;
```

вызывает сама себя!

? Что получится?

```
printBin(6);
```

Рекурсивная процедура — это процедура, которая вызывает сама себя.

```
printBin(6);
  printBin(3);
    printBin(1);
      printBin(0);
        printBin(0);
```

бесконечные вызовы

? Как исправить?

Рекурсивная процедура

```
procedure printBin(n: integer);
begin
  if n = 0 then exit;
  printBin(n div 2);
  write(n mod 2)
end;
```

? Что получится?
printBin(6)

```
printBin(6);
  printBin(3);
    printBin(1);
      printBin(0);
    write(1 mod 2);
  write(3 mod 2);
write(6 mod 2);
```

рекурсия заканчивается!

1 1 0

Задачи

«A»: Напишите рекурсивную процедуру, которая переводит число в восьмеричную систему.

Пример:

```
Введите число: 66
В восьмеричной: 102
```

«B»: Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 9.

Пример:

```
Введите число: 75
Основание: 6
В системе с основанием 6: 203
```

Задачи

«C»: Напишите рекурсивную процедуру, которая переводит число в шестнадцатеричную систему.

Пример:

```
Введите число: 123
В шестнадцатеричной: 7B
```

«D»: Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 36.

Пример:

```
Введите число: 350
Основание: 20
В системе с основанием 20: HA
```

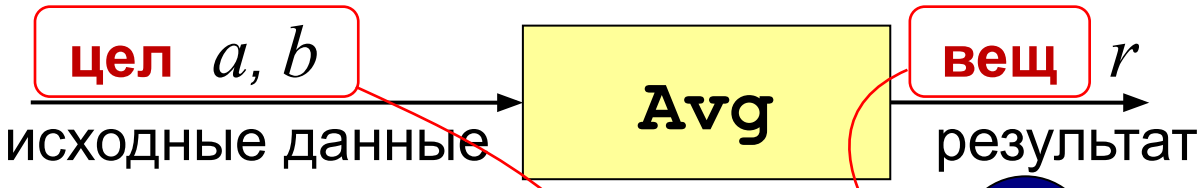
Программирование (Паскаль)

§ 25. Функции

Что такое функция?

Функция — это вспомогательный алгоритм, который возвращает результат (число, строку символов и др.).

Задача. Написать функцию, которая вычисляет среднее арифметическое двух целых чисел.



? Тип результата?

```
function Avg(a, b: integer): real;
begin
  Avg :=(a+b)/2
end.
```

специальная переменная для записи результата функции

Как вызывать функцию?

Запись результата в переменную:

```
var sr: real;
sr:= Avg(5, 8);
```

6.5

```
var x, y: integer;
    sr: real;
x:=2; y:=5;
sr:= Avg(x, 2*y+8);
```

10

? Чему равно?

Вывод на экран:

```
var x, y: integer;
    sr: real;
x:=2; y:=5;
sr:= Avg(x, y+3);
writeln( Avg(12, 7) );
writeln( sr+Avg(x, 12) );
```

5

9.5

12

Как вызывать функцию?

Использование в условных операторах:

```
var a, b: integer;
...
read( a, b );
if Avg(a,b) > 5 then
  writeln('Да!')
else
  writeln('Нет!');
```

? Когда печатает «Да»?

Как вызывать функцию?

Использование в циклах:

```
var a, b: integer;
...
read( a, b );
while Avg(a,b) > 0 do begin
  writeln('Нет!');
  read( a, b )
end;
writeln('Угадал!');
```

Когда напечатает «Угадал»?

В других языках программирования

Python:

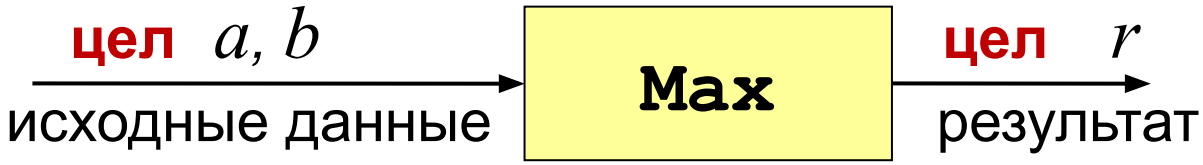
```
def Avg(a, b):
  return(a+b)/2
```

C:

```
float Avg(int a, int b)
{
  return (a+b)/2.0;
}
```

Максимум из двух (трёх) чисел

Задача. Составить функцию, которая определяет наибольшее из двух целых чисел.




```
function Max(a, b: integer):
integer;
begin
  if a>b then
       Max:=a
  else Max:=b
end;
```

? Как с её помощью найти максимум из трёх?

```
function Max3(a, b, c: integer):
                        integer;
begin
  Max3:= Max( Max(a,b), c )
кон
```

Сумма цифр числа

Задача. Составить функцию, которая вычисляет сумму значений цифр натурального числа.

```
function sumDigits(N: integer): integer;
var d, sum: integer;
begin
  sum:=0; { накапливаем сумму с 0 }
  while N<>0 do begin
    d:= N mod 10; { выделим последнюю цифру }
    sum:=sum+d;   { добавим к сумме }
    N:= N div 10  { удалим последнюю цифру }
  end;
  sumDigits:=sum
end;
```

Задачи

«A»: Напишите функцию, которая вычисляет среднее арифметическое пяти целых чисел.

Пример:

```
Введите 5 чисел: 1 2 3 4 6
Среднее: 3.2
```

«B»: Напишите функцию, которая находит количество цифр в десятичной записи числа.

Пример:

```
Введите число: 751
Количество цифр: 3
```

Задачи

«С»: Напишите функцию, которая находит количество единиц в двоичной записи числа.

Пример:

```
Введите число: 75
Количество единиц: 4
```

Логические функции

Логическая функция — это функция, возвращающая логическое значения (**да** или **нет**).

- можно ли применять операцию?
- успешно ли выполнена операция?
- обладают ли данные каким-то свойством?

Логические функции

Задача. Составить функцию, которая возвращает «**True**», если она получила чётное число и «**False**», если нечётное.

```
function Even(N: integer):
                    boolean;
begin
  if N mod 2 = 0 then
    Even:= True
  else
    Even:= False
end;
```

```
function Even(N: integer):
                    boolean;
begin
  Even:=(N mod 2 = 0)
end;
```

Рекурсивные функции

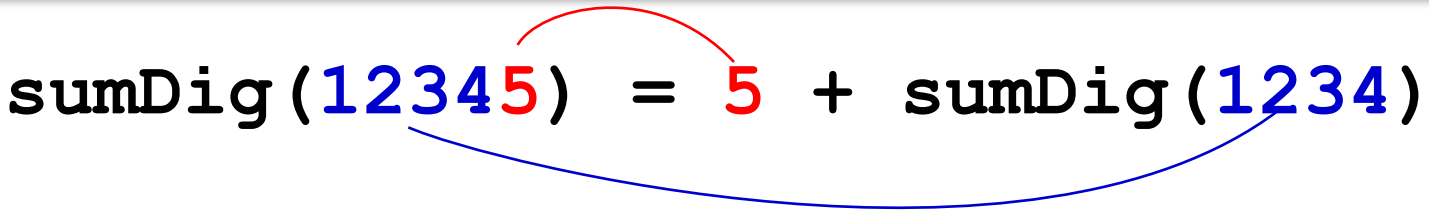
Рекурсивная функция — это функция, которая вызывает сама себя.

Задача. Составить рекурсивную функцию, которая вычисляет сумму цифр числа.

? Как сформулировать решение рекурсивно?

Сумму цифр числа N нужно выразить через сумму цифр другого (меньшего) числа.

Сумма цифр числа N равна значению последней цифры плюс сумма цифр числа, полученного отбрасыванием последней цифры.



Рекурсивная функция

Сумма цифр числа N

Вход: натуральное число **N**.
Шаг 1: **d:= N mod 10**
Шаг 2: **M:= N div 10**
Шаг 3: **s:=** сумма цифр числа **M**
Шаг 4: **sum:= s + d**
Результат: **sum**.

последняя цифра

число без последней цифры

Что забыли?

Когда остановить?

Сумма цифр числа (рекурсия)

```
function sumDigRec(N: integer): integer;
var d, sum: integer;
begin
  if N = 0 then
    sumDigRec:=0
  else begin
    d:=N mod 10;
    sum:=sumDigRec(N div 10);
    sumDigRec:=sum+d
  end
end;
```

? Зачем это?

? Где рекурсивный вызов?

Задачи

«A»: Напишите логическую функцию, которая возвращает значение «истина», если десятичная запись числа заканчивается на цифру 0 или 1.

Пример:

```
Введите число: 1230
Ответ: Да
```

«B»: Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число помещается в 8-битную ячейку памяти.

Пример:

```
Введите число: 751
Ответ: Нет
```

Задачи

«С»: Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число простое (делится только на само себя и на единицу).

Пример:

```
Введите число: 17
Число простое!
```

Пример:

```
Введите число: 18
Число составное!
```

Конец фильма

ПОЛЯКОВ Константин Юрьевич
д.т.н., учитель информатики
ГБОУ СОШ № 163, г. Санкт-Петербург
kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович
к.ф.-м.н., доцент кафедры мультимедийной дидактики и ИТО ПГГПУ, г. Пермь
eremin@pspu.ac.ru

Источники иллюстраций

1. иллюстрации художников издательства «Бином»
2. авторские материалы