Программирование (C++)

§ 23. Как разрабатывают программы?

Программирование (C++)

§ 19. Символьные строки

Что такое символьная строка?

Символьная строка – это последовательность символов.

Хочется:

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

```
string s; // символьная строка
```

строковый тип

```
string s = "Ура!"; // начальное значение
```

```
string s(5, '*'); // "*****"
```

Символьные строки

Присваивание:

```
s = "Вася пошёл гулять";
```

```
string s;
```

Ввод с клавиатуры:

```
cin >> s;
```

до конца слова (до пробела)

```
getline( cin, s );
```

до конца строки

Вывод на экран:

```
cout << s;
```

Длина строки:

```
int n;
n = s.size();
```

метод для строк

точечная запись

Сравнение строк

```
string s;
...
cout << "Введите пароль: ";
cin >> s;
if( s == "sEzAm" )
  cout << "Слушаюсь и повинуюсь!";
else
  cout << "Пароль неправильный";
```

? Какой правильный пароль?

? Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

Сравнение строк

```
string s1, s2;
...
s1 = "паровоз";
s2 = "пароход";
if( s1 < s2 )
  cout << s1 << "<" << s2;
else
  if( s1 == s2 )
    cout << s1 << "=" << s2;
  else
    cout << 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';  // нумерация символов с 0!
```

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

нумерация с 0!

для каждого символа строки

```
for( int i=0; i<s.size(); i++ )
  if( s[i] == 'э' )
     s[i] = 'e';
```

C++11:

```
for( auto& sym: s )
  if( sym == 'э' )
     sym = 'e';
```

Задачи

«A»: Напишите программу, которая вводит строку, состоящую только из точек и букв X, и заменяет в ней все точки на нули и все буквы X на единицы.

Пример:

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

«B»: Напишите программу, которая в символьной строке заменяет все нули на единицы и наоборот. Остальные символы не должны измениться.

Пример:

```
Введите строку: 10a01Bx1010c
Инверсия: 01a10Bx0101c
```

Задачи

«C»: 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 = подстроки):

```
s = "0123456789";
s1 = s.substr(3,5); // "34567"
```

с какого символа

сколько символов

```
s1 = s.substr(3); // "3456789"
```

до конца

Операции со строками

Удаление:

```
s = "0123456789";
s.erase( 3, 6); // "0129"
```

с какого символа

сколько символов

Вставка:

```
s = "0123456789";
s.insert(3, "ABC"); // "012ABC3456789"
```

с какого символа

что

Поиск в строках

```
s = "Здесь был Вася.";
```

где

что

```
int n = s.find('c');
```

no position

```
if( n != string::npos )
  cout << "Номер символа " << n;
else
  cout << "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 и имя:
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

«C»: 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:

```
string s = "12.345бе-бе-бе6789";
int n = stoi( s ); // n = 12
```

до первой ошибки

Вещественное число:

```
string s = "12.345бе-бе-бе6789";
float x = stof( s ); // x = 123.456
```

Преобразования «число» → «строка»

```
int N = 123;
string sN = to_string( N ); // "123"
double X = 123.45;
string sX = to_string( X ); // "123.450000"
```

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 (C++)

§ 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

```
int x, count = 0;
cin >> x;
while( x != 0 ) {
   if( x > 0 )
      count++;
   cin >> x;
   }
cout << count;
```

откуда взять x?

? Что плохо?

Найди ошибку!

```
int x, count = 0;
cin >> x;
while( x != 0 )
   if( x > 0 )
      count++;
   cin >> x;
cout << count;
```

Найди ошибку!

```
int x, count = 0;
cin >> x;
while( x != 0 ) {
   if( x > 0 )
      count++;
   cin >> x;
   }
cout << 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 += число
```

```
if( x % 10 == 5 )
```

Обработка потока данных

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

```
int x, sum = 0;
cin >> x;
while( x != 0 ) {
   if( x % 10 == 5 )
      sum += x;
   cin >> x;
   }
cout << sum;
```

Чего не хватает?

Найди ошибку!

```
int x, sum = 0;
cin >> x; = 0 ) {
   if( x % 10 == 5 )
      sum += x;
   cin >> x;
   }
cout << 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

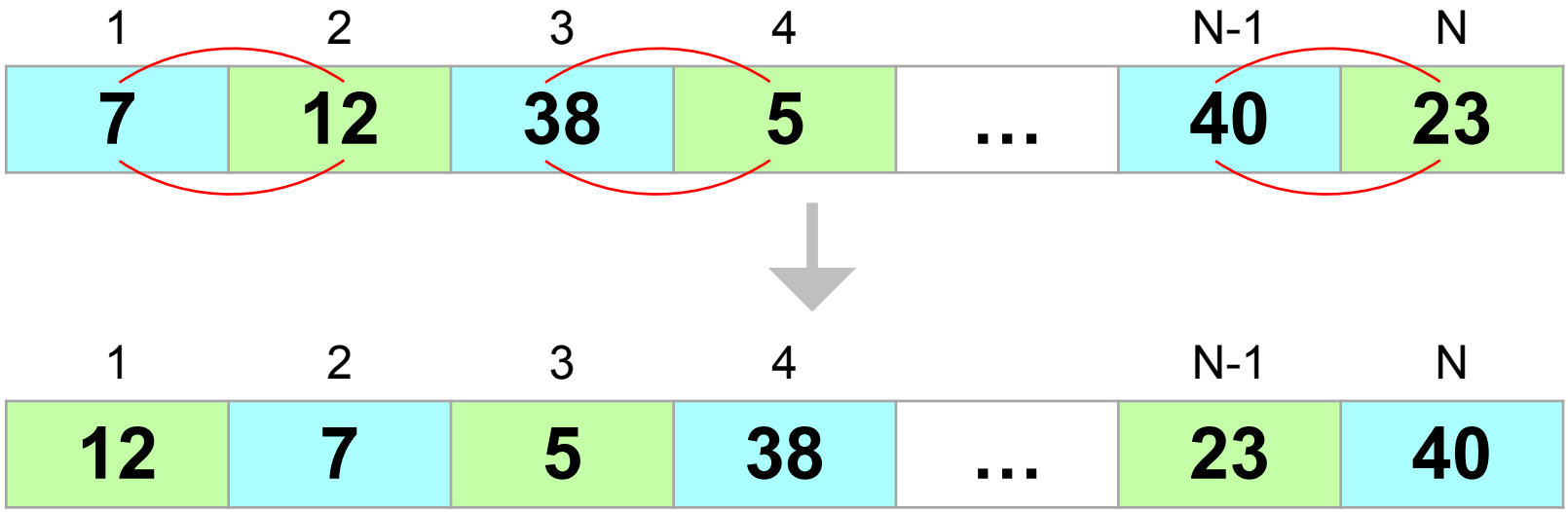
```
c = a;
a = b;
b = c;
```

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(int i=0; i<N; i++) {
   поменять местами A[i] и A[i+1]
   }
```

? Что плохо?

0	1	2	3	4	5
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

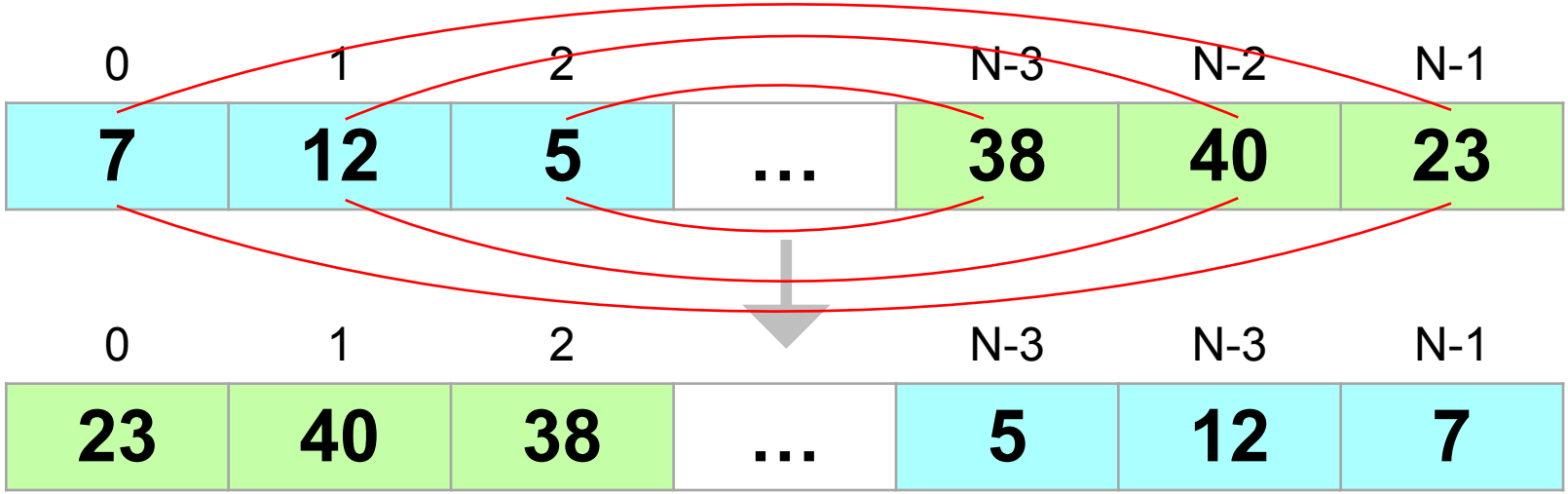
«шагаем» через один

```
for( int i=0; i<N-1; i+=2 ) {
  // переставляем A[i] и A[i+1]
  int c = A[i];
  A[i] = A[i+1];
  A[i+1] = c;
  }
```

A[0]↔A[1], A[2]↔A[3], …, A[N-2]↔A[N-1]

Реверс массива

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

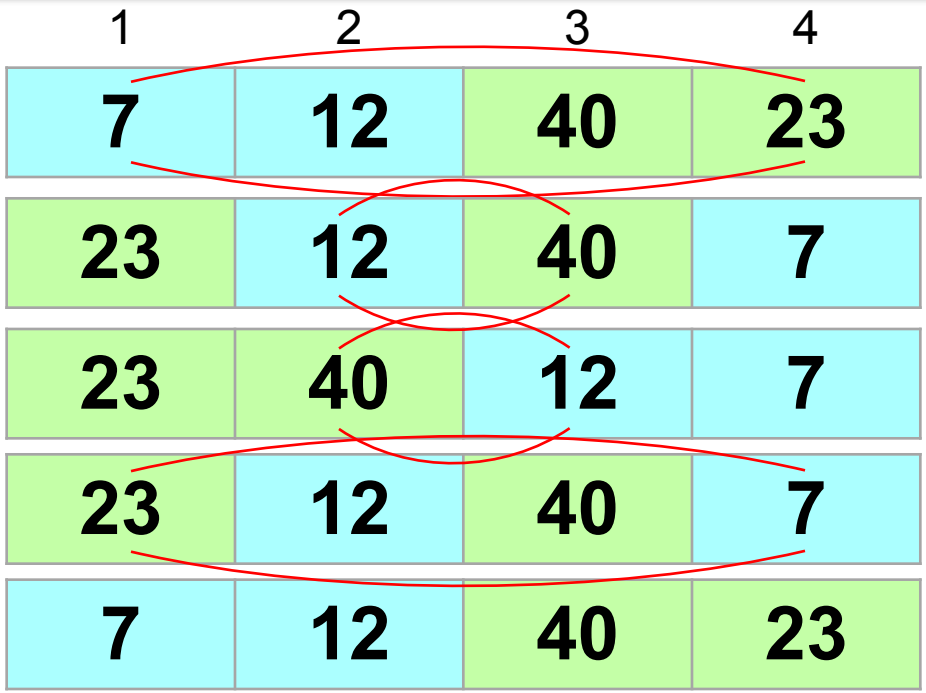



```
A[0]↔A[N-1]      0+N-1 = N-1
A[1]↔A[N-2]      1+N-2 = N-1
A[i]↔A[N-1-i]    i+??? = N-1
A[N-1]↔A[0]      N-1+0 = N-1
```

Реверс массива

```
for(int i=0; i<N/2; i++) {
   поменять местами A[i] и A[N+1-i]
   }
```

? Что плохо?

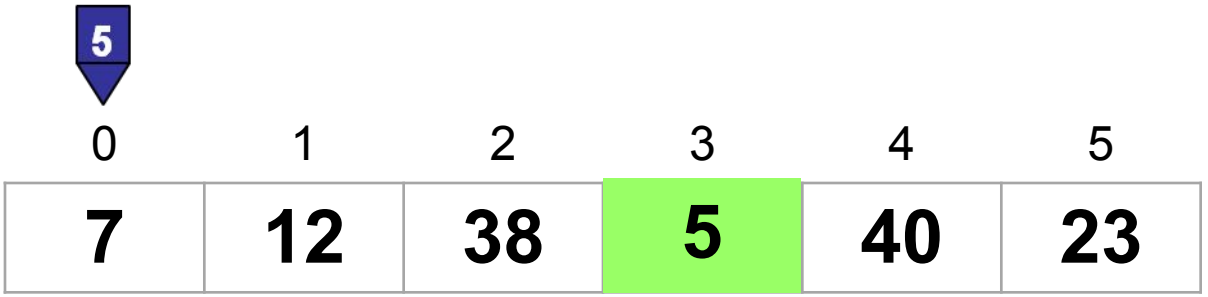


? Как исправить?

Линейный поиск в массиве

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

X = 5




```
int i = 0;
while( A[i] != X )
   i++;
cout << "A[" << i << "]=" << X;
```

? Что плохо?

? Если искать 4?

! Нельзя выходить за границы массива!

Линейный поиск в массиве

не выходим за границу

```
int i = 0;
while( i<N and A[i]!=X )
   i++;
if( i < N )
  cout << "A[" << i << "]=" << X;
else
   cout << "Не нашли!";
```

? Как проверить, нашли или нет?

Досрочный выход из цикла

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

```
int nX = -1; // недействительный номер
for( int i=0; i<N; i++ )
  if( A[i] == X  ) {
    nX = i; // запомнить номер
    break;
    }
if( nX >= 0 )
  cout << "A[" << nX << "]=" << X;
else
  cout << "Не нашли!";
```

нашли!

сразу выйти из цикла

Задачи

«A»: 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
```

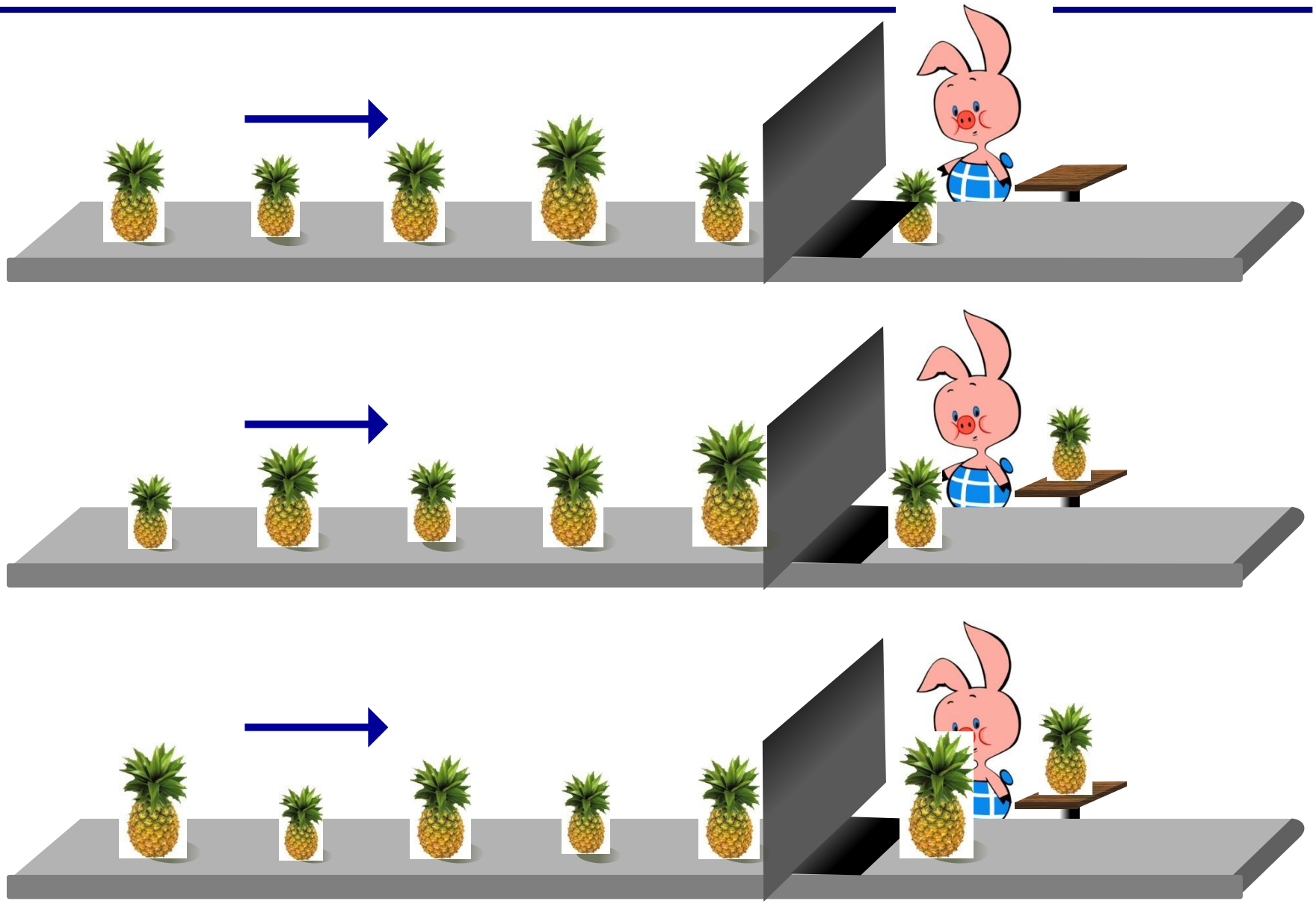
Задачи

«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( int i=0; i<N; i++ )
  if( A[i] > M )
    M = A[i];
cout << M;
```

? Чего не хватает?

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

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

или

2) **M = A[0]** (или любой другой элемент)

максимальный не меньше, чем **A[0]**

Поиск максимального элемента

```
M = A[0];
for( int i=1; i<N; i++ )
   if( A[i] > M )
      M = A[i];
cout << M;
```

начинаем с **A[1]**, так как **A[0]** мы уже посмотрели

? Как найти минимальный?

Номер максимального элемента

Задача. Найти в массиве максимальный элемент и его номер.

Какие переменные нужны?

```
int M = A[0];
int nMax = 0;
for( int i=1; i<N; i++ )
  if( A[i] > M ) {
    M = A[i];
    nMax = i;
    }
cout << "A[" << nMax << "]=" << M;
```

Можно ли убрать одну переменную?

Номер максимального элемента

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

```
int M = A[0];
int nMax = 0;
for( int i=1; i<N; i++ )
  if( A[i] > A[nMax] ) {
    M = A[i];
    nMax = i;
    }
cout << "A[" << nMax << "]="
                    << A[nMax] ;
```

Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M = A[0];
for( int i=1; i<N; i++ )
  if( A[i]<0 and A[i]>M )
    M = A[i];
cout << 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[0];
for( int i=1; i<N; i++ )
  if( A[i]<0 )
    if( M>=0 or A[i]>M )
      M = A[i];
cout << 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?

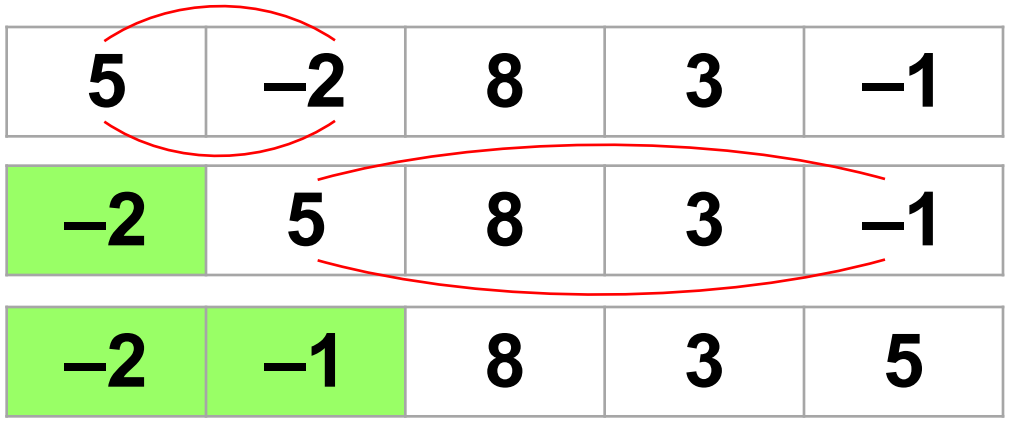
- нашли минимальный, поставили его на первое место

```
c = A[nMin];
A[nMin] = A[0];
A[0] = c;
```

? Как?

? Что дальше?

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



Сортировка выбором

```
for( int i=0; i<N-1; i++ ) {
    // ищем минимальный среди A[i]..A[N-1]

  int nMin = i;
  for( int j=i+1; j<N; j++ )
    if( A[j] < A[nMin] )
      nMin = j;
  // переставляем A[i] и A[nMin]
  int c = A[i];
  A[i] = A[nMin];
  A[nMin] = c;
  }
```

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
```

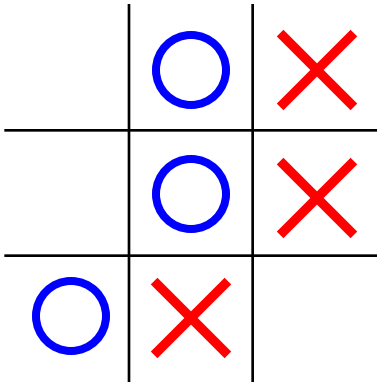
Программирование (C++)

§ 21. Матрицы (двумерные массивы)

Что такое матрица?

нолик

нет знака



крестик

строка 1, столбец 2

? 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 int N = 3, M = 4;
```

строки

столбцы

```
int A[N][M] = {
   { 2, 3, 5,  1},
   { 4, 8, 3, 12},
   {14, 4, 7, 11}
   };
float X[4][M] = { {2, 6, 10} };
bool L[N][2] = {};
```

остальные нули

все **false**

Простые алгоритмы

Заполнение случайными числами:

```
for( int i=0; i<N; i++ ) {
  for( int j=0; j<M; j++ ) {
    A[i][j] = 20 + rand() % 61;
    cout << A[i][j] << " ";
    }
  cout << endl;
  }
```

Вложенный цикл!

в одной строке через пробел

следующий – с новой строки

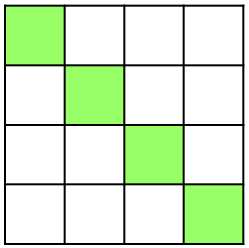
Суммирование:

```
int sum = 0;
for( int i=0; i<N; i++ )
 for( int j=0; j<M; j++ )
   sum += A[i][j];
```

Перебор элементов матрицы

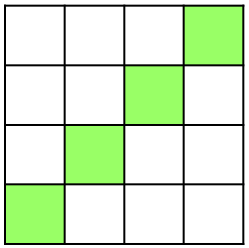
Главная диагональ:

```
for( int i=0; i<N; i++ ) {
  // работаем с  A[i][i]
  }
```



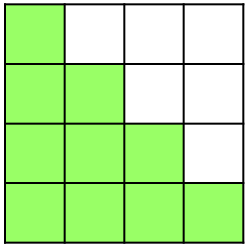
Побочная диагональ:

```
for( int i=0; i<N; i++ ) {
  // работаем с  A[i][N-1-i]
  }
```



Главная диагональ и под ней:

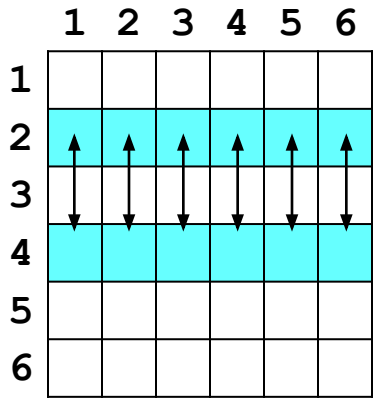
```
for( int i=0; i<N; i++ )
  for( int j=0; j<N; j++ ) {
    // работаем с  A[i][j]
    }
```



Перестановка строк

2-я и 4-я строки:

```
for( int j=0; j<M; j++ ) {
  int c = A[2][j];
  A[2][j] = A[4][j];
  A[4][j] = c;
  }
```



Задачи

«A»: 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 (C++)

§ 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( int i=0; i<N; i++ )
  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( int i=0; i<N-1; i++ ) {
  int nMin = i;
  for( int j=i+1; j<N; j++ )
    if( A[j] < A[nMin] )
      nMin = j;
  int c = A[i]; A[i] = A[nMin]; A[nMin] = c;
  }
```

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 $N \times N$.

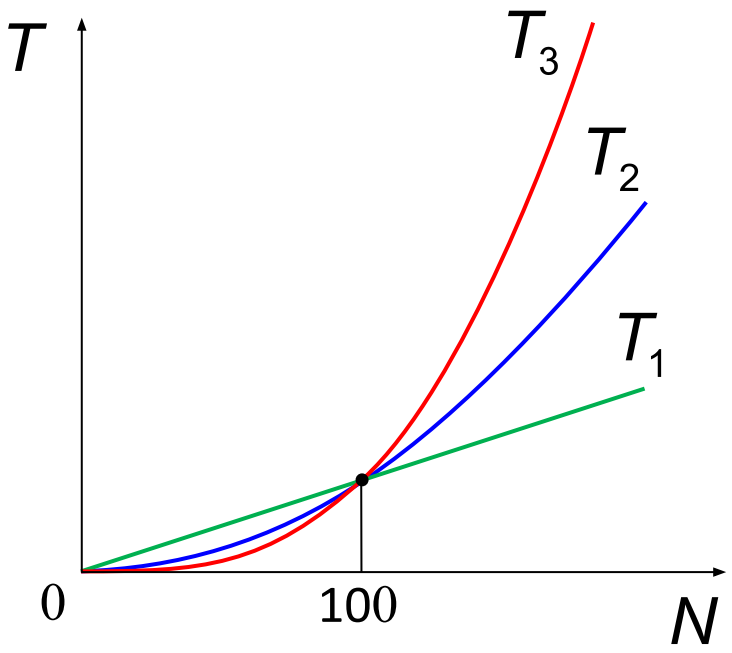
```
int sum = 0;
for( int i=0; i<N; i++ )
 for( int j=0; j<N; j++ )
    sum = sum + A[ij];
```

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

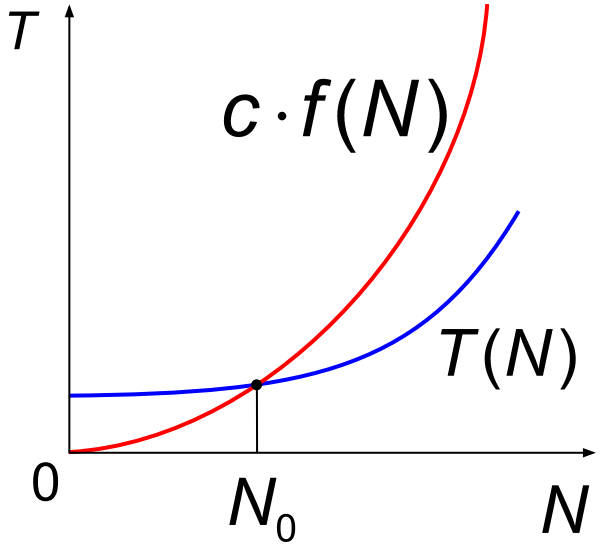
$T(N)$	**время выполнения**
N	100 нс
N^2	10 мс
N^3	0,001 с
2^N	10^{13} лет

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 (C++)

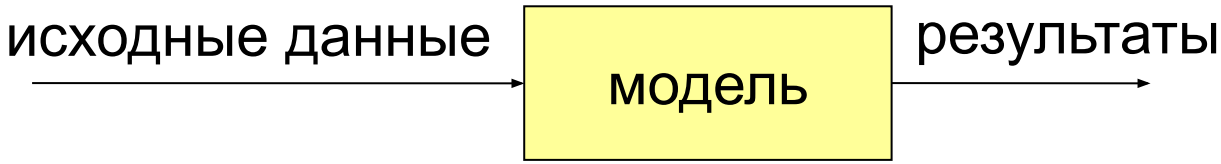
§ 23. Как разрабатывают программы

Этапы разработки программ

I. Постановка задачи

Документ: *техническое задание*.

II. Построение модели



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

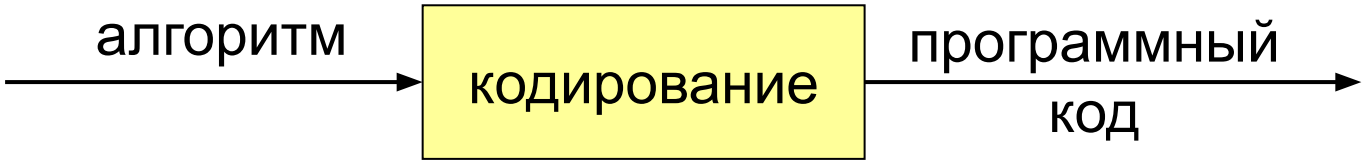
III. Разработка алгоритма и способа хранения данных

«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

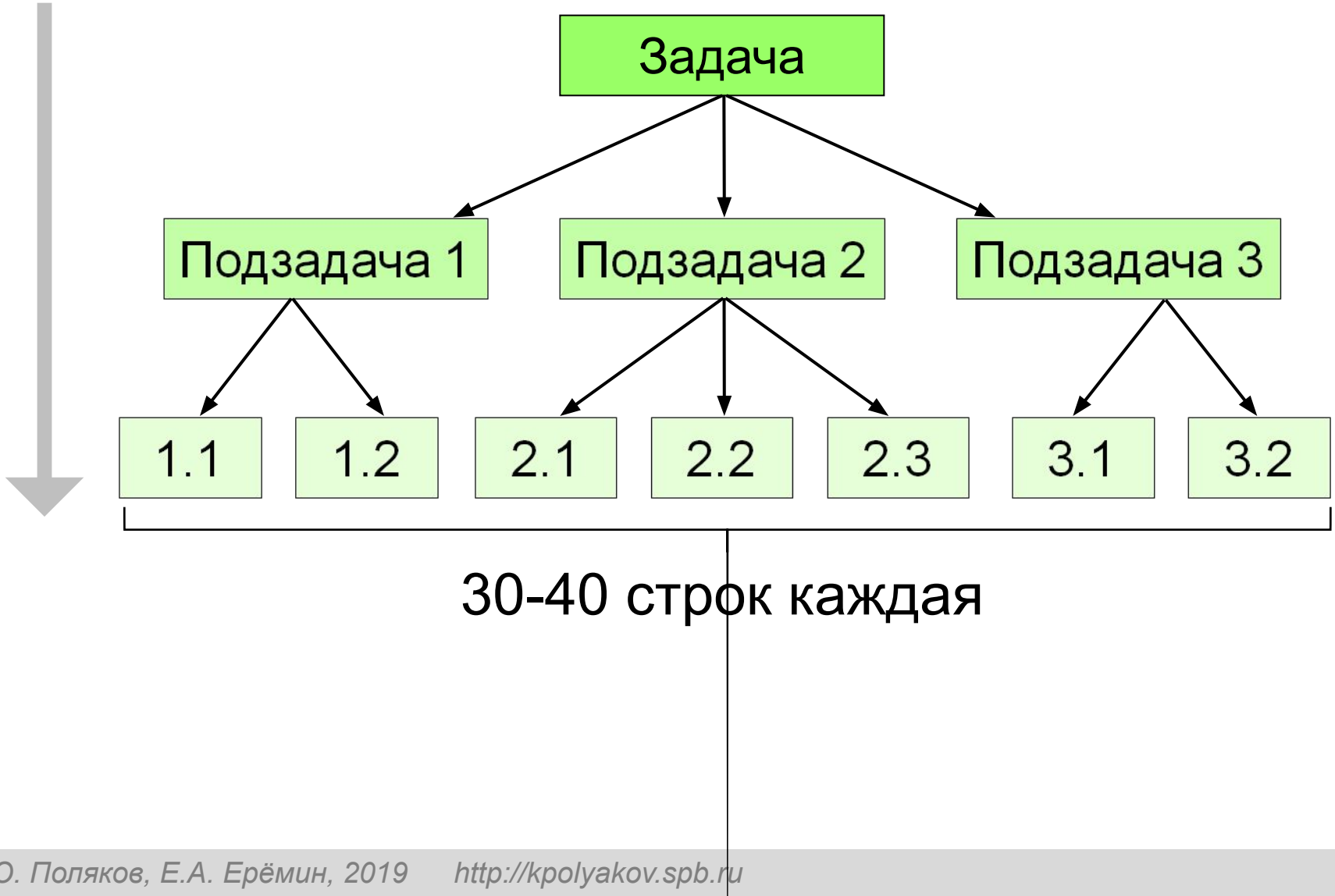
Технические писатели

VIII. Внедрение и сопровождение

- обучение пользователей
- исправление найденных ошибок
- техподдержка

Методы проектирования программ

«Сверху вниз» (последовательное уточнение)



Методы проектирования программ

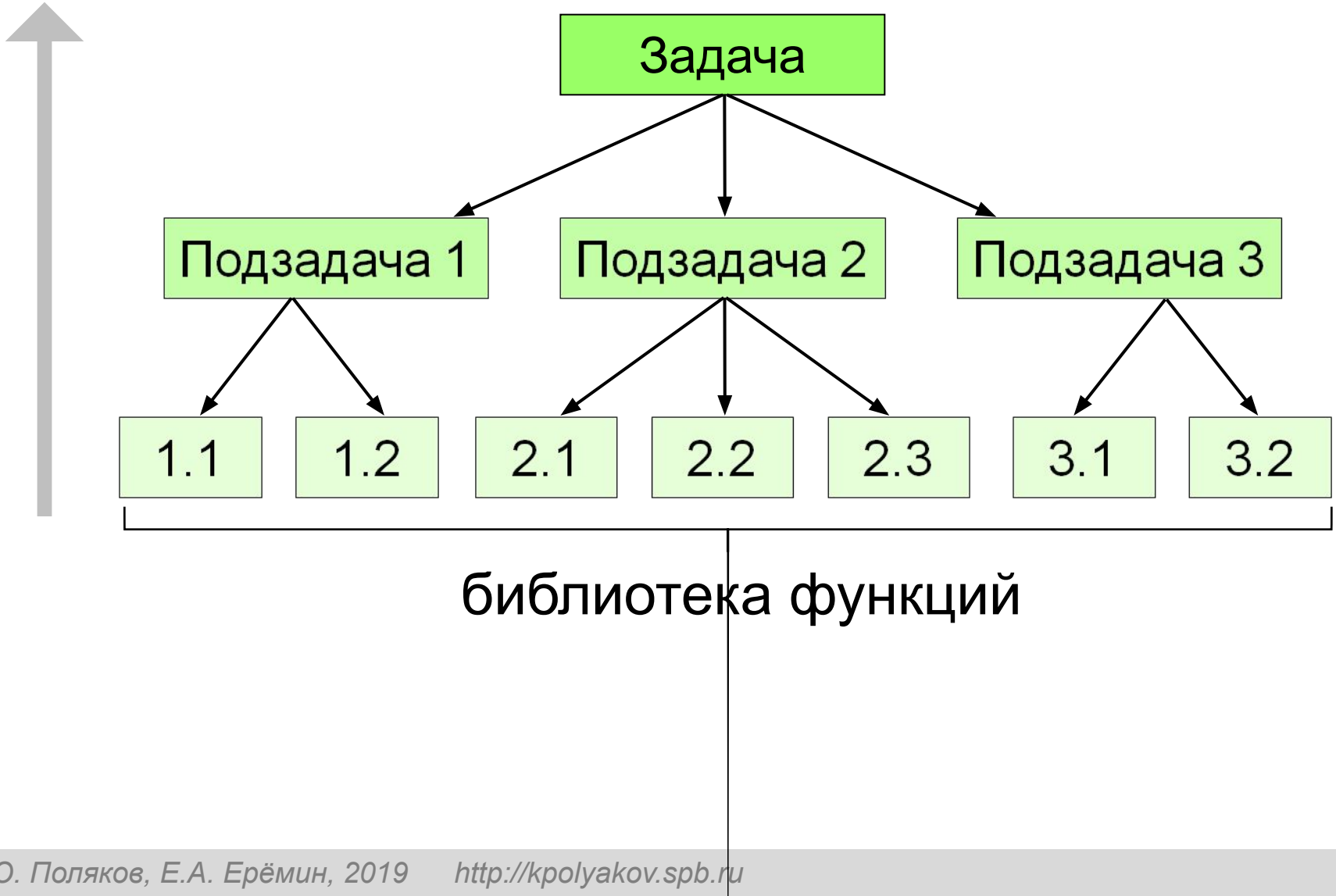
«Сверху вниз» (последовательное уточнение)

- 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!

Пример отладки программы

Программа решения квадратного уравнения

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

```
int main()
   {
   float a, b, c, D, x1, x2;
   cout << "Введите a, b, c: ";
   cin >> a >> b >> c;
   D = b*b - 4*a*a;
   x1 = (-b + sqrt(D))/2*a;
   x2 = (-b - sqrt(D))/2*a;
   cout << "x1=" << x1
        << " x2=" << x2);
   }
```

Тестирование

Тест 1. $a = 1$, $b = 2$, $c = 1$.

Ожидание:

```
x1=-1 x2=-1
```

Реальность:

```
x1=-1 x2=-1
```

Тест 2. $a = 1$, $b = -5$, $c = 6$.

```
x1=3 x2=2
```

```
x1=4.79129 x2=0.208712
```

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.

```
cin >> a >> b >> c;
cout << a << " " << b << " " << c << endl;
D = b*b - 4*a*a;
cout << "D=" << D << endl;
...
```

Результат:

```
Введите a, b, c: 1 -5 6
1 -5 6
D=21
```

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

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

! Одна ошибка найдена!

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
```

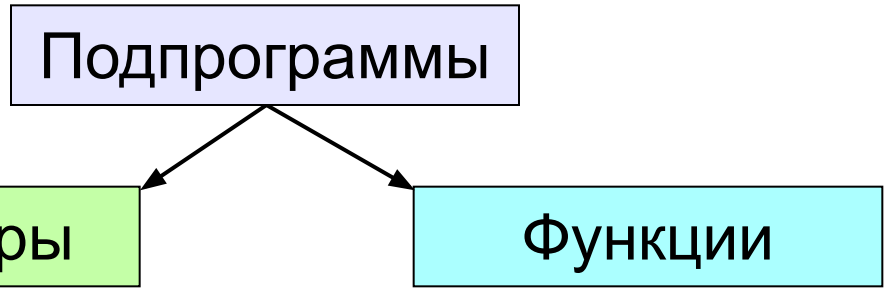
2. Решение уравнения $x^2 + x + 6 = 0$

```
Введите a, b, c: 1 1 6
Нет.
```

Программирование (C++)

§ 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

```
#include <iostream>
using namespace std;

void printLine() {
   cout << "----------" << endl;
   }

int main() {
   ...
   printLine();
   ...
   }
```

Что делает?

какие-то операторы

вызов процедуры

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

Линии разной длины

```
void printLine5() {
  cout << "-----" << endl;
  }
```

? Как улучшить?

```
void printLine10() {
  cout << "----------" << endl;
  }
```

```
void printLine10() {
  for( int i=0; i<10; i++ )
    cout << "-";
  cout << endl;
  }
```

параметр процедуры

```
void printLine10( int n ) {
  for( int i=0; i<n; i++ )
    cout << "-";
  cout << endl;
  }
```

Процедура с параметром

```
#include <iostream>
using namespace std;

void printLine10( int n ) {
   ...
   }

int main() {
   ...
   printLine(10);
   ...
   printLine(7);
   printLine(5);
   printLine(3);
   }
```

Параметр – величина, от которой зависит работа процедуры.

? Что делает?

Аргумент – значение параметра при конкретном вызове.

Несколько параметров

символьная строка

```
void printLine( string c, int n ) {
  for( int i=0; i<n; i++ )
    cout << c;
  cout << endl;
  }
```

? Что изменилось?

? Как вызывать?

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

✓
```
printLine( "+", 5 );
```

✓
```
printLine( "+-+", 5 );
```

В других языках программирования

Python:

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

Паскаль:

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

```
void printBin( int n ) {
  ...
  }
```

Алгоритм перевода через остатки:

```
while( n!=0 ) {
  cout << n % 2;
  n = n / 2;
  }
```

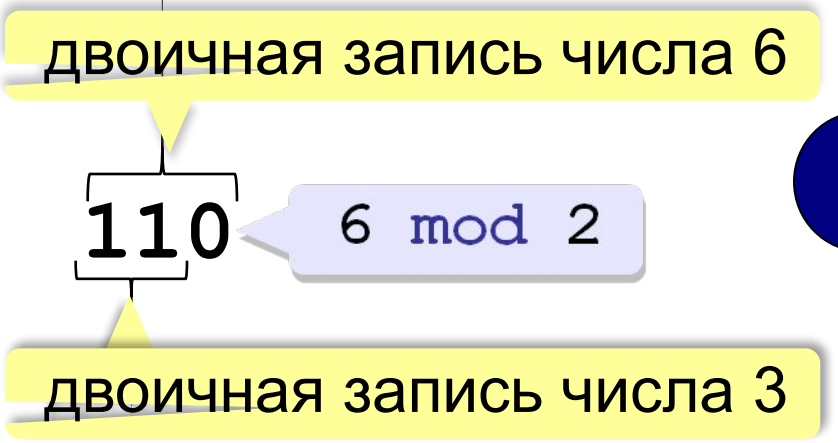
? Что получится при **n = 6**?

011 ✗

в обратном порядке!

Рекурсия

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



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

Это и есть **рекурсия**!

! Чтобы понять рекурсию, нужно понять рекурсию! ☺

Рекурсивная процедура

```
void printBin( int n ) {
  printBin( n / 2 );
  cout << n % 2;
  }
```

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

```
void printBin( int n ) {
  if( n == 0 ) return;
  printBin( n / 2 );
  cout << n % 2;
  }
```

? Что получится?
printBin(6)

```
printBin(6);
    printBin(3);
        printBin(1);
            printBin(0);
        cout << 1 % 2;
    cout << 3 % 2;
cout << 6 % 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
```

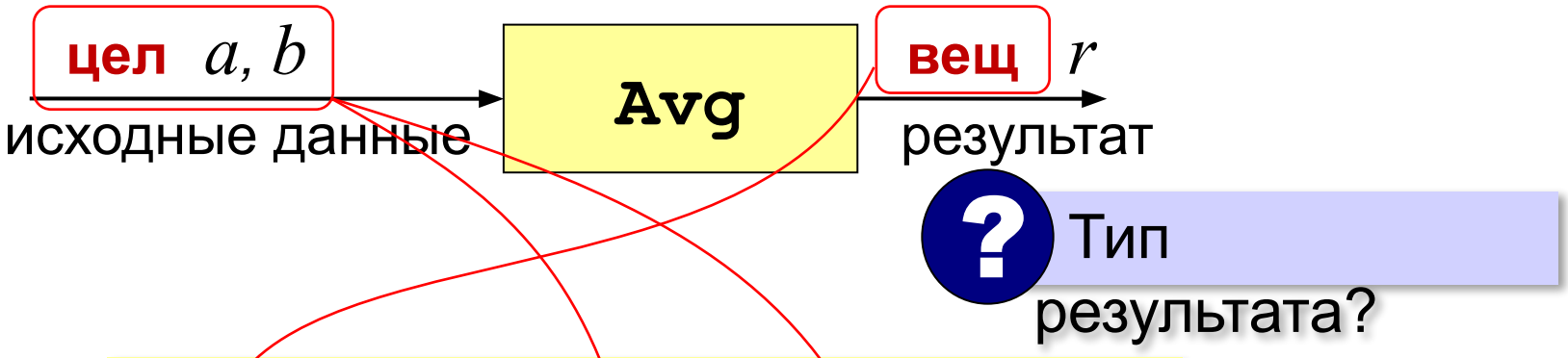
Программирование (C++)

§ 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.




```
float Avg( int a, int b ) {
  float sred =(a+b)/2.;
  return sred;
  }
```

это результат!

Как вызывать функцию?

Запись результата в переменную:

```
float sr;
sr = Avg(5, 8);
```

6.5

```
int x = 2, y = 5;
float sr = Avg(x, 2*y+8);
```

10

? Чему равно?

Вывод на экран:

```
int x = 2, y = 5;
float sr = Avg(x, y+3);
cout << Avg(12, 7);
cout << sr + Avg(x, 12);
```

5
9.5
12

Как вызывать функцию?

Использование в условных операторах:

```
int a, b;
cin >> a >> b;
if( Avg(a,b) > 5 )
   cout << "Да!";
else
   cout << "Нет!";
```

? Когда печатает «Да»?

Как вызывать функцию?

Использование в циклах:

```
int a, b;
cin >> a >> b;
while( Avg(a,b) > 0 ) {
   cout << "Нет!";
   cin >> a >> b;
   }
cout << "Угадал!";
```

? Когда напечатает «Угадал»?

В других языках программирования

Python:

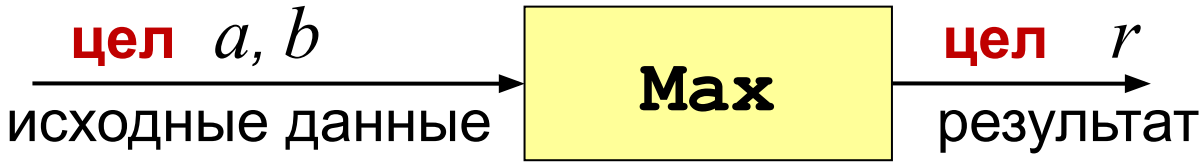
```
def Avg(a, b):
  return(a+b)/2
```

Паскаль:

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

Максимум из двух (трёх) чисел

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




```
int Max( int a, int b ) {
  if( a > b )
    return a;
  else
    return b;
  }
```

? Как с её помощью найти максимум из трёх?

```
int Max3( int a, int b, int c ) {
  return Max( Max(a,b), c );
  }
```

Сумма цифр числа

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

```
int sumDigits( int N ) {
  int d, sum = 0; // накапливаем сумму с 0
  while( N != 0 ) {
    d = N % 10; // выделим последнюю цифру
    sum += d;   // добавим к сумме
    N = N / 10; // удалим последнюю цифру
    }
  return sum; // возвращаем результат
  }
```

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.

```
bool Even( int N ) {
  if( N % 2 == 0 )
    return true;
  else
    return false;
  }
```

```
bool Even( int N ) {
  return (N % 2 == 0);
  }
```

Рекурсивные функции

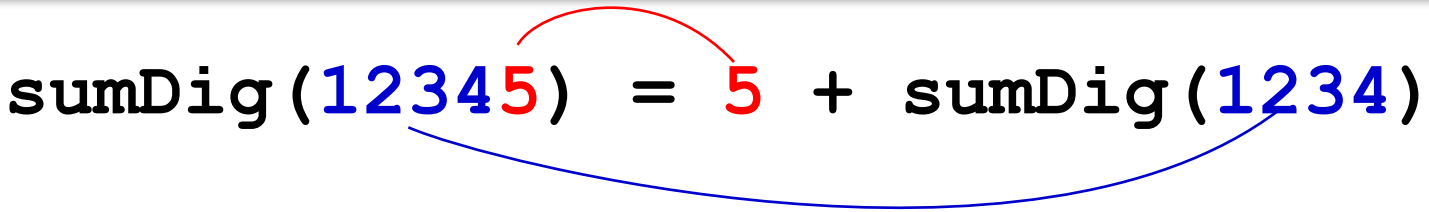
Рекурсивная функция — это функция, которая вызывает сама себя.

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 % 10**

Шаг 2: **M = N / 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)

```
int sumDigRec( int N ) {
  if( N == 0 ) return 0;
  int d = N % 10;
  int sum = sumDigRec(N /10);
  return sum+d;
  }
```

? 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