25 (высокий уровень, время – 20 минут)

Тема: Обработка целых чисел. Проверка делимости

Что проверяется:

Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности (?).

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Что нужно знать:

- в известных задачах этого типа (не олимпиадных) нет ограничения на время выполнения, по крайней мере, оно несущественно для отрезков, заданных для перебора; поэтому можно использовать простой перебор без оптимизации;
- задачи этого типа предлагается решать с электронных таблиц или собственной программы; как правило, написать правильную программу значительно проще
- пусть необходимо перебрать все целые числа на отрезке [a; b] и подсчитать, для скольких из них выполняется некоторое условие; общая структура цикла перебора записывается так (Python):

```
count = 0
for n in range(a, b+1):
  if условие выполнено:
    count += 1
print( count )
```

Pascal:

```
count := 0;
for n:=a to b do
  if условие выполнено then
    count := count + 1;
writeln(count);
```

C++:

```
int count = 0;
for(int n = a; n <= b; n++)
  if( условие выполнено )
    count += 1;
std::cout << count;
```

- проверку условия удобно оформить в виде функции, возвращающей логическое значение (True/False), но можно этого и не делать
- проверить делимость числа n на число d можно с помощью операции взятия остатка от деления n на x: если остаток равен 0, число n делится на x нацело
- проверка делимости на языке Python выглядит так:

```
if n % d == 0:
      print("Делится")
else: print("Не делится")
```

- тоже самое на Pascal

```
if n mod d = 0 then
     writeln('Делится')
else writeln('Не делится')
```

- то же самое на C++

```
if( n % d == 0 )
     std::cout << "Делится";
else std::cout << "Не делится";
```

Количество делителей

- для определения числа делителей натурального числа n можно использовать цикл, в котором перебираются все возможные делители d от 1 до n, при обнаружении делителя увеличивается счётчик делителей:

```
count = 0
for d in range(1, n+1):
  if n % d == 0:
    count += 1
print( count )  # вывести количество делителей
```

- тоже самое на Pascal

```
count := 0;
for d:=1 to n do
  if n mod d = 0 then
    count := count + 1;
writeln( count );
```

- то же самое на C++

```
int count = 0;
for(int d = 1; d <= n; d++)
  if( n % d == 0 ) count ++;
std::cout << count; // вывести количество делителей
```

- если требуется определить не только количество делителей, но и сами делители, нужно сохранять их в массиве
- в языке Python удобно использовать динамический массив: сначала он пуст, а при обнаружении очередного делителя этот делитель добавляется в массив:

```
divs = []
for d in range(1,n+1): # перебор всех возможных делителей
  if n % d == 0:       # если нашли делитель d
    divs.append(d)     # то добавили его в массив
```

- в языках Pascal и C++ проще обойтись без динамического массива; здесь есть два варианта:
  1) выделить массив достаточного размера для хранения всех делителей; например, количество делителей числа n явно не превышает n;
  2) хранить только нужное количество делителей, например, если нас интересуют числа, имеющие 4 делителя, достаточно выделить массив из 4-х элементов, а остальные делители в массив не записывать
- перебор делителей можно оптимизировать, учитывая, что наименьший из пары делителей, таких что $a \cdot b = n$, не превышает квадратного корня из n; нужно только аккуратно обработать случай, когда число n представляет собой квадрат другого целого числа;
- отметим, что для чисел, которые предлагаются в вариантах заданий, такая оптимизация не обязательна; более того, усложнение программы может привести к дополнительным ошибкам…

Простые числа

- простое число n делится только на 1 и само на себя, причём единица не считается простым числом; таким образом, любое простое число имеет только два делителя
- для определения простоты числа можно считать общее количество его делителей; если их ровно два, то число простое, если не два – не простое:

```
nDel = 0              # количество делителей числа
for d in range(1, n+1): # все возможные делители
  if n % d == 0:
    nDel += 1         # нашли ещё делитель
if nDel == 2:
  print( "Число простое" )
else:
  print( "Число составное" )
```

- работу программы можно ускорить: если уже найдено больше двух делителей, то число не простое и можно досрочно закончит работу цикла с помощью оператора **break**:

```
nDel = 0              # количество делителей числа
for d in range(1, n+1): # все возможные делители
  if n % d == 0:
    nDel += 1         # нашли ещё делитель
    if nDel > 2:      # уже не простое число
      break           # досрочный выход из цикла
if nDel == 2:
  print( "Число простое" )
else:
  print( "Число составное" )
```

- другой вариант – считать количество делителей числа на отрезке [2; n–1]; как только хотя бы один такой делитель будет найден, можно завершить цикл, потому что число явно не простое:

```
nDel = 0              # количество делителей на отрезке [2; n-1]
for d in range(2, n):
  if n % d == 0:
    nDel += 1         # нашли делитель
    break             # досрочный выход из цикла
if nDel == 0:
  print( "Число простое" )
else:
  print( "Число составное" )
```

- можно сделать то же самое с помощью логической переменной:

```
prime = True              # сначала считаем число простым
for d in range(2, n):
  if n % d == 0:
    prime = False         # уже не простое
    break                 # досрочный выход из цикла
if prime:
  print( "Число простое" )
else:
  print( "Число составное" )
```

- тоже самое на Pascal

```
prime := True;              { сначала считаем число простым }
for d:=2 to n-1 do
  if n mod d = 0 then begin
    prime := False;   { уже не простое }
    break             { досрочный выход из цикла }
  end;
if prime then
  writeln( 'Число простое' )
```

```
else
  writeln( 'Число составное' );
```

- то же самое на C++

```
bool prime = true;              // сначала считаем число простым
for( int d = 2; d <= n-1); d++ )
  if( n % d == 0 ) {
    prime = false; // уже не простое
    break;         // досрочный выход из цикла
    }
if( count == 2 )
  std::cout << "Число простое";
else
  std::cout << "Число составное";
```

- в этом задании обычно предлагаются большие числа, поэтому проверка делимости на все числа от 2 до n-1 выполняется очень долго, и на устаревших компьютерах время работы приведённого выше алгоритма может быть слишком велико
- программу можно оптимизировать, если вспомнить, что наименьший из пары делителей, таких что $a \cdot b = n$, не превышает квадратного корня из n; поэтому можно закончить перебор значением $\sqrt{n}$, округлив его до ближайшего целого числа; если на отрезке [2; $\sqrt{n}$] не найден ни один делитель, их нет и на отрезке [$\sqrt{n}$ + 1, $n - 1$]
- следовательно, можно существенно ускорить перебор, изменив конечное значение переменной цикла:

```
for d in range(2, round(sqrt(n))+1):
```

- на языке Pascal:

```
for d:=2 to round(sqrt(n)) do
```

- на языке C++:

```
for( int d = 2; d <= round(sqrt(n)); d++ )
```

Пример задания:

P-02 (демо-2021). *Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [174457; 174505], числа, имеющие ровно два различных натуральных делителя, не считая единицы и самого числа. Для каждого найденного числа запишите эти два делителя в таблицу на экране с новой строки в порядке возрастания произведения этих двух делителей. Делители в строке таблицы также должны следовать в порядке возрастания.*

1) если число имеет ровно два делителя, отличных от единицы и самого числа, то произведение этих делителей и есть само число; таким образом, строки в таблице должны быть записаны в порядке возрастания чисел, которые они образуют;
2) чтобы сами делители в одной строке были записаны в порядке возрастания, нужно выполнять перебор от меньшего числа на отрезке к большему;
3) эффективно использовать ускоренный перебор делителей, то есть для числа N перебирать только числа от 2 до $q = \sqrt{N}$ (не включая точный квадратный корень, если он существует); все делители – парные, то есть если a – делитель N, то $b = N / a$ – тоже делитель N
4) программа была написана при разборе задачи P-00, она подходит для любого заданного количества делителей; так как здесь нам нужно выводить все делители, кроме единицы и самого числа, цикле перебора делителей начинаем с 2, и не включаем $q = \sqrt{N}$:

```
from math import sqrt
```

```
divCount = 2  # нужное количество делителей
for n in range(174457, 174505+1):
  divs = []
  q = round(sqrt(n))
  for d in range(2,q):
    if n % d == 0:
      divs = divs + [d, n//d]
      if len(divs) > divCount: break
  if len(divs) == divCount:
    print( *divs )
```

5) поскольку делителей всего 2, сортировать их не нужно – первым всегда будет меньший из делителей

6) Ответ:

3 58153
7 24923
59 2957
13 13421
149 1171
5 34897
211 827
2 87251

7) программа на **Паскале**:

```
const divCount = 2;
var n, count, d, i, j, q: longint;
    divs: array[1..divCount] of longint;
begin
  for n:=174457 to 174505 do begin
    count := 0;
    q := round(sqrt(n));
    for d:=2 to q-1 do
      if n mod d = 0 then begin
        count := count + 2;
        if count <= divCount then begin
          divs[count-1] := d;
          divs[count] := n div d;
        end
        else break
      end;
    if count = divCount then begin
      for i:=1 to divCount do
        write(divs[i], ' ');
      writeln
    end
  end
end.
```

8) программа на **C++**:

```
#include <iostream>
#include <cmath>
int main()
{
```

```
  const int divCount = 2;
  int divs[divCount] = {};
  for( int n = 174457; n <= 174505; n++ ) {
    int count = 0;
    int q = round(sqrt(n));
    for( int d = 2; d <= q-1; d++ )
      if( n % d == 0 ) {
        count += 2;
        if( count <= divCount ) {
          divs[count-2] = d;
          divs[count-1] = n / d;
          }
        else break;
        }
    if( count == divCount ) {
      for( int i = 0; i < divCount; i++ )
        std::cout << divs[i] << ' ';
      std::cout << std::endl;
      }
    }
}
```

Ещё пример задания:

P-01. *Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку* [3532000; 3532160], *простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.*

9) поскольку нужно вывести не только сами числа, но и их порядковые номера, нужно использовать счётчик:

```
count = 0
for n in range(3532000, 3532160+1):
  if число n простое:
    count += 1
    print( count, n )
```

Решение (простой перебор, может работать очень долго):

10) для определения простоты числа ищем полное количество делителей, если оно равно 2, то число простое:

```
count = 0
for n in range(3532000, 3532160+1):
  nDel = 0
  for d in range(1,n+1):
    if n % d == 0: nDel += 1
    if nDel > 2: break
  if nDel == 2:
    count += 1
    print( count, n )
```

11) полная программа а языке Pascal:

```
var n, count, d, nDel: longint;
begin
  count := 0;
```

```
  for n:=3532000 to 3532160 do begin
    nDel := 0;
    for d:=1 to n do
      if n mod d = 0 then begin
        nDel := nDel + 1;
        if nDel > 2 then break;
      end;
    if nDel = 2 then begin
      count := count + 1;
      writeln( count, ' ', n )
    end
  end
end.
```

12) полная программа а языке C++:

```
#include <iostream>
#include <cmath>
int main()
{
  int count = 0;
  for( int n = 3532000; n <= 3532160; n++ ) {
    int nDel = 0;
    for( int d = 1; d <= n; d++ )
      if( n % d == 0 ) {
        nDel += 1;
        if( nDel > 2 ) break;
        }
    if( nDel == 2 ) {
      count++;
      std::cout << count << ' ' << n << std::endl;
      }
    }
}
```

13) Ответ:

1 3532007
2 3532019
3 3532021
4 3532033
5 3532049
6 3532091
7 3532103
8 3532121
9 3532147

Решение (перебор до $\sqrt{n}$):

1) полная программа на языке Python:

```
from math import sqrt
count = 0
for n in range(3532000, 3532160+1):
  prime = True
  for d in range(2, round(sqrt(n))):
```

```
      if n % d == 0:
        prime = False
        break
    if prime:
      count += 1
      print( count, n )
```

2) **(Б.С. Михлин)** ещё один вариант, в котором вместо функции sqrt используется возведение в степень 0,5:

```
count = 0
for n in range(3532000, 3532160+1):
  for d in range(2, round(n**0.5)+1):
    if n % d == 0:
      break
  else: # else относится к циклу "for d ...", а не к "if"
    # блок выполняется, если не сработал "break"
    count += 1
    print( count, n )
```

3) **(Б.С. Михлин)** компактное решение, использующее встроенную функцию **all** – она возвращает логическое значение **True**, если все элементы переданного ей списка равны **True**; возвращает **False**, если хотя бы один из них равен **False**:

```
count=0
for n in range(3532000,3532160+1):
      # если у 'n' нет делителей от 2 до корня из n
      #(т.е. все 'd' дают остаток отличный от нуля):
    if all( n%d!=0 for d in range(2,round(n**0.5)+1) ):
        count+=1
        print(count,n)
```

4) вариант с функцией **isPrime**, которая возвращает логическое значение **True** (истина) для простых чисел и **False** (ложь) для составных:

```
from math import sqrt
def isPrime(n):
  for d in range(2, round(sqrt(n)+1) ):
    if n % d == 0:
      return False
  return True

count = 0
for n in range(3532000, 3532160+1):
  if isPrime(n):
    count += 1
    print( count, n )
```

5) Ответ:

1 3532007
2 3532019
3 3532021
4 3532033
5 3532049
6 3532091
7 3532103
8 3532121

9 3532147

Решение (программа на языке Pascal):

14) обратим внимание на заданный отрезок [3532000; 3532160]; числа в нём превышают 32767 – предел для 16-битных целых чисел типа **integer**; поэтому для того, чтобы программа работала правильно на всех системах, вместо **integer** используем тип **longint**, такие переменные всегда занимают 4 байта (диапазон от –2147483648 до 2147483647)
15) для каждого числа **n** из заданного диапазона в цикле ищем делители; количество найденных простых чисел хранится в переменной **count**:

```
var n, count, d: longint;
    prime: boolean;
begin
  count := 0;
  for n:=3532000 to 3532160 do begin
    prime := True;
    for d:=2 to round(sqrt(n)) do
      if n mod d = 0 then begin
        prime := False;
        break;
      end;
    if prime then begin
      count := count + 1;
      writeln( count, ' ', n )
    end
  end
end.
```

16) вариант с функцией **isPrime**, которая возвращает логическое значение **True** (истина) для простых чисел и **False** (ложь) для составных:

```
var n, count: longint;
function isPrime( n: integer ): boolean;
var d: longint;
begin
  isPrime := True;
  for d:=2 to round(sqrt(n)) do
    if n mod d = 0 then begin
      isPrime := False;
      break;
    end;
end;
begin
  count := 0;
  for n:=3532000 to 3532160 do begin
    if isPrime(n) then begin
      count := count + 1;
      writeln( count, ' ', n )
    end
  end
end.
```

6) Ответ:

1 3532007

2 3532019
3 3532021
4 3532033
5 3532049
6 3532091
7 3532103
8 3532121
9 3532147

Решение (программа на языке C++):

17) для того, чтобы использовать математические функции. нужно подключить заголовочный файл **cmath**:

```
#include <cmath>
```

18) полная программа на языке C++:

```
#include <iostream>
#include <cmath>
int main()
{
  int count = 0;
  for( int n = 3532000; n <= 3532160; n++ ) {
    bool prime = true;
    for( int d = 2; d <= round(sqrt(n)); d++ )
      if( n % d == 0 ) {
        prime = false;
        break;
        }
    if( prime ) {
      count++;
      std::cout << count << ' ' << n << std::endl;
      }
    }
}
```

19) вариант с функцией **isPrime**, которая возвращает логическое значение **true** (истина) для простых чисел и **false** (ложь) для составных:

```
#include <iostream>
#include <cmath>
bool isPrime( int n )
{
  bool prime = true;
  for( int d = 2; d <= round(sqrt(n)); d++ )
    if( n % d == 0 ) {
      prime = false;
      break;
      }
  return prime;
}
int main()
{
  int count = 0;
  for( int n = 3532000; n <= 3532160; n++ )
```

```
      if( isPrime(n) ) {
        count++;
        std::cout << count << ' ' << n << std::endl;
        }

  }
```

20) Ответ:

1 3532007
2 3532019
3 3532021
4 3532033
5 3532049
6 3532091
7 3532103
8 3532121
9 3532147

Ещё пример задания:

P-00. *Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку* [194455; 194500], *числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.*

Решение (простой перебор):

1) поскольку заданный отрезок [194455; 194500] содержит не так много чисел, можно решать задачу простым перебором, особо не заботясь об эффективности вычислений
2) при написании программы на языке Python можно поступить так

```
for для всех чисел n в интервале:
  divs = массив всех делителей n
  if len(divs) == 4:
    вывести массив делителей
```

3) полная программа на языке Python:

```
for n in range(194455, 194500+1):
  divs = []
  for d in range(1,n+1):
    if n % d == 0:
      divs.append(d)
  if len(divs) == 4:
    print( *divs )
```

4) (**Б.С. Михлин**) построить массив делителей на языке Python можно и с помощью генератора списка:

```
for n in range(194455, 194500+1):
  divs = [d for d in range(1, n+1) if n % d == 0]
  if len(divs) == 4:
    print( *divs )
```

Аналогично можно построить массив делителей, удовлетворяющих заданному условию, например, всех чётных делителей:

```
for n in range(194455, 194500+1):
  divs = [d for d in range(1, n+1)
              if n % d == 0 and d % 2 == 0]
```

```
    if len(divs) == 4:
      print( *divs )
```

К сожалению, этот способ сложно использовать в других языках программирования.

5) в качестве оптимизации можно прерывать работу внутреннего цикла, когда найден пятый делитель (число *n* уже точно не подходит), но это не критично:

```
for n in range(194455, 194500+1):
  divs = []
  for d in range(1,n+1):
    if n % d == 0:
      divs.append(d)
      if len(divs) > 4: break
  if len(divs) == 4:
    print( *divs )
```

6) ещё один вариант программы (с функцией, которая возвращает массив делителей):

```
def allDivisors(n):
  divs = []
  for d in range(1,n+1):
    if n % d == 0:
      divs.append(d)
  return divs

for n in range(194455, 194500+1):
  divs = allDivisors(n)
  if len(divs) == 4:
    print( *divs )
```

7) Ответ:

1 5 38891 194455
1 163 1193 194459
1 139 1399 194461
1 2 97231 194462
1 113 1721 194473
1 439 443 194477
1 2 97241 194482
1 43 4523 194489
1 11 17681 194491

Решение (ускоренный перебор):

21) идея состоит в том, чтобы для определения количества делителей числа N перебирать только числа до $q = \sqrt{N}$; если число q целое, его нужно добавить в список делителей, а все остальные делители – парные, то есть если a – делитель N, то $b = N / a$ – тоже делитель N

22) получается такая программа, которая подходит для любого заданного количества делителей:

```
from math import sqrt
divCount = 4  # нужное количество делителей
for n in range(194455, 194500+1):
  divs = []
  q = round(sqrt(n))
  if q*q == n:
    divs.append(q)
  for d in range(1,q):
```

```
      if n % d == 0:
        divs = divs + [d, n//d]
        if len(divs) > divCount: break
    if len(divs) == divCount:
      print( *sorted(divs) )
```

23) Ответ:

1 5 38891 194455
1 163 1193 194459
1 139 1399 194461
1 2 97231 194462
1 113 1721 194473
1 439 443 194477
1 2 97241 194482
1 43 4523 194489
1 11 17681 194491

Решение (программа на языке Pascal):

1) обратим внимание на заданный отрезок [194455; 194500]; числа в нём превышают 32767 – предел для 16-битных целых чисел типа **integer**; поэтому для того, чтобы программа работала правильно на всех системах, вместо **integer** используем тип **longint**, такие переменные всегда занимают 4 байта (диапазон от –2147483648 до 2147483647)
2) поскольку нас интересуют только числа, у которых 4 делителя, можно хранить в памяти только первые 4 найденных делителя, а как только будет найден пятый, заканчивать поиск делителей (число нам точно не подходит); такой подход позволяет отказаться от использования динамических массивов и выделить один массив из 4 элементов:

```
  divs: array[1..4] of longint;
```

3) для каждого числа **n** из заданного диапазона в цикле ищем делители; количество найденных делителей хранится в переменной **count**:

```
  count := 0;
  for d:=1 to n do              { перебор всех возможных делителей }
    if n mod d = 0 then begin   { нашли делитель }
      count := count + 1;
      if count <= 4 then        { сохраняем первые 4 делителя }
        divs[count] := d
      else break                { нашли пятый => выходим }
    end;
```

4) полная программа на языке Pascal:

```
  var n, count, d, i: longint;
      divs: array[1..4] of longint;
  begin
    for n:=194455 to 194500 do begin
      count := 0;
      for d:=1 to n do
        if n mod d = 0 then begin
          count := count + 1;
          if count <= 4 then
            divs[count] := d
          else break
        end;
```

```
    if count = 4 then begin
      for i:=1 to 4 do
        write(divs[i], ' ');
      writeln
    end
  end
end.
```

5) вариант с функцией **divsNumber**, которая возвращает количество делителей числа:

```
var n, i: longint;
    divs: array[1..4] of longint;
function divsNumber(n: longint): longint;
var count, d: integer;  { локальные переменные }
begin
  count := 0;
  for d:=1 to n do
    if n mod d = 0 then begin
      count := count + 1;
      if count <= 4 then
           divs[count] := d
      else break
    end;
  divsNumber := count
end;
begin
  for n:=194455 to 194500 do begin
    if divsNumber(n) = 4 then begin
      for i:=1 to 4 do
        write(divs[i], ' ');
      writeln
    end
  end;
end.
```

8) ускоренный перебор (до $q = \sqrt{N}$); в отличие от программы на Python, нужно вручную делать сортировку массива, поскольку делители записывались в массив не в порядке возрастания:

```
const divCount = 4;
var n, count, d, i, j, q: longint;
    divs: array[1..divCount] of longint;
begin
  for n:=194455 to 194500 do begin
    count := 0;
    q := round(sqrt(n));
    if q*q = n then begin
      count := count + 1;
      divs[count] := q;
    end;
    for d:=1 to q-1 do
      if n mod d = 0 then begin
        count := count + 2;
        if count <= divCount then begin
          divs[count-1] := d;
```

```
            divs[count] := n div d;
          end
          else break
        end;
      if count = divCount then begin
          { сортировка масива divs }
        for i:=1 to divCount do
          for j:=i to divCount-1 do
            if divs[j] > divs[j+1] then
              swap( divs[j], divs[j+1] );
        for i:=1 to divCount do
          write(divs[i], ' ');
        writeln
      end
    end
  end.
```

9) Ответ:

1 5 38891 194455
1 163 1193 194459
1 139 1399 194461
1 2 97231 194462
1 113 1721 194473
1 439 443 194477
1 2 97241 194482
1 43 4523 194489
1 11 17681 194491

Решение (программа на языке C++):

1) при программировании на языке C++ нужно не забыть, что нумерация элементов массивов начинается с нуля
2) полная программа на языке C++:

```
#include <iostream>
int main()
{
  int divs[4] = {};
  for( int n = 194455; n <= 194500; n++ ) {
    int count = 0;
    for( int d = 1; d <= n; d++ )
      if( n % d == 0 ) {
        count ++;
        if( count <= 4 )
          divs[count-1] = d;
        else break;
        }
    if( count == 4 ) {
      for( int i = 0; i < 4; i++ )
        std::cout << divs[i] << ' ';
      std::cout << std::endl;
      }
    }
```

```
}
```

3) вариант с функцией **divsNumber**, которая возвращает количество делителей числа и заполняет переданный ей массив первыми 4-мя делителями:

```
#include <iostream>
int divsNumber( int n, int divs[] )
{
  int count = 0;
  for( int d = 1; d <= n; d++ )
    if( n % d == 0 ) {
      count ++;
      if( count <= 4 )
           divs[count-1] = d;
      else break;
      }
  return count;
}
int main()
{
  int divs[4] = {};
  for( int n = 194455; n <= 194500; n++ ) {
    if( divsNumber(n, divs) == 4 ) {
      for( int i = 0; i < 4; i++ )
        std::cout << divs[i] << ' ';
      std::cout << std::endl;
      }
    }
}
```

4) ускоренный перебор (до $q = \sqrt{N}$); в отличие от программы на Python, нужно вручную делать сортировку массива, поскольку делители записывались в массив не в порядке возрастания:

```
#include <iostream>
#include <cmath>
int main()
{
  const int divCount = 4;
  int divs[divCount] = {};
  for( int n = 194455; n <= 194500; n++ ) {
    int count = 0;
    int q = round(sqrt(n));
    if( q*q == n ) {
      divs[count] = q;
      count ++;
      }
    for( int d = 1; d <= q-1; d++ )
      if( n % d == 0 ) {
        count += 2;
        if( count <= divCount ) {
          divs[count-2] = d;
          divs[count-1] = n / d;
          }
        else break;
```

```
            }
        if( count == divCount ) {
            // сортировка масива divs
          for( int i = 0; i < divCount; i++ )
            for( int j = i; j < divCount-1; j++ )
              if( divs[j] > divs[j+1] ) {
                int temp = divs[j];
                divs[j] = divs[j+1];
                divs[j+1] = temp;
                }
          for( int i = 0; i < divCount; i++ )
            std::cout << divs[i] << ' ';
          std::cout << std::endl;
          }
        }
  }
```

5) Ответ:

1 5 38891 194455
1 163 1193 194459
1 139 1399 194461
1 2 97231 194462
1 113 1721 194473
1 439 443 194477
1 2 97241 194482
1 43 4523 194489
1 11 17681 194491

6) (**Муфаззалов Д.Ф., г. Уфа**) Ускоренный перебор на языке C++ можно осуществлять

без сортировки, если располагать делители в нужном порядке по мере их получения;

без извлечения корня и округления, если преобразовать неравенство d < sqrt(n) по правилам математики;

и без проверки на полный квадрат, так как если число является полным квадратом, то оно имеет нечетное количество делителей.

```
#include <iostream>
int main()
{
    const int divCount =4;
    int divs[divCount],i,d;
    for( int n = 194455; n <= 194500; n++ )
    {
        int count = 0;
        for(  d = 1; d*d < n; d++ )
            if( n % d == 0 )
            {
                divs[count/2] = d;
                divs[divCount-count/2-1]=n/d;
                count+=2;
                if( count > divCount ) break;
            }
        if (count == divCount)
        {
```

```
            for( i = 0; i < divCount; i++ )
                std::cout << divs[i] << ' ';
            std::cout << std::endl;
        }
    }
}
```

7) **(Муфаззалов Д. Ф., г. Уфа)** Сортировки можно избежать и если хранить только половину меньших делителей, а другую половину получать при выводе:

```
#include <iostream>
int main()
{
    const int divCount = 4;
    int divs[divCount/2],i,d;
    for( int n = 194455; n <= 194500; n++ ) {
      int count = 0;
      for( d = 1; d*d < n; d++ )
        if( n % d == 0 ) {
          divs[count/2] = d;
          count += 2;
          if( count > divCount ) break;
          }
      if (count == divCount) {
        for( i = 0; i < divCount/2; i++ )
          std::cout << divs[i] << ' ';
        for(  i--; i>=0; i-- )
          std::cout << n/divs[i] << ' ';
        std::cout << std::endl;
        }
      }
}
```

8) **(Муфаззалов Д. Ф., г. Уфа)** Программа с ускоренным перебором, не зависящая от четности количества делителей

```
#include <iostream>
int main()
{
    const int divCount =4;
    int divs[divCount],i,d;
    for( int n = 194455; n <= 194500; n++  ) {
      int count = 0;
      for(  d = 1; d*d < n; d++ )
        if( n % d == 0 ) {
          divs[count/2] = d;
          count+=2;
          if( count > divCount ) break;
          }
      if( d*d == n ) {
        divs[count/2] = d;
        count++;
        }
      if (count == divCount) {
        for( i = 0; i < divCount/2; i++ )
```

```
            std::cout << divs[i] << ' ';
          if( divCount % 2 )
            std::cout << divs[divCount/2] << ' ';
          for(  i--; i>=0; i-- )
            std::cout << n/divs[i] << ' ';
          std::cout << std::endl;
          }
        }
}
```

9) В задаче на поиск чисел с четырьмя делителями массив не нужен вовсе. Для таких чисел достаточно найти минимальный делитель, отличный от единицы, а остальные будут равны единице, самому числу и частному от деления самого числа на найденный делитель. Если число имеет хотя бы 2 делителя больше единицы и меньше корня из этого числа, то оно не имеет ровно 4 делителя.

```
#include <iostream>
using namespace std;
int main()
{
    int div,d,count;
    for( int n = 194455; n <= 194500; n++ ) {
      for( d = 2,count = 0; d*d < n && count < 2; d++ )
        if( n % d == 0 && !count++ ) div = d;
      if (count == 1)
        cout << 1 << ' ' << div << ' '
             << n/div << ' ' << n << endl;
      }
}
```

Задачи для тренировки:

1) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [126849; 126871], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
2) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [102714; 102725], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
3) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [154026; 154043], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
4) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [209834; 209857], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
5) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [150750; 150764], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
6) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [251811; 251826], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
7) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [113012; 113061], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
8) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [222987; 223021], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
9) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [258274; 258297], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
10) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [338472; 338494], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.
11) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [164700; 164752], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
12) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [193136; 193223], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
13) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [100812; 100923], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
14) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [177399; 177453], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
15) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [180131; 180179], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.

16) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [135743; 135789], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
17) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [157898; 157990], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
18) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [143146; 143215], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
19) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [118811; 118972], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
20) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [100806; 100950], числа, имеющие ровно 6 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
21) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [190201; 190230], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке убывания.
22) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [190201; 190280], числа, имеющие ровно 4 различных **ЧЁТНЫХ** делителя. Выведите эти четыре делителя для каждого найденного числа в порядке убывания.
23) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [190061; 190080], числа, имеющие ровно 4 различных **НЕЧЁТНЫХ** делителя. Выведите эти четыре делителя для каждого найденного числа в порядке убывания.
24) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [11275; 16328], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
25) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [20789; 35672], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
26) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [78920; 92430], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
27) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [125873; 136762], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
28) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [244143; 367821], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
29) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [652938; 744328], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
30) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [904528; 997438], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.

31) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [1820348; 1880927], числа, имеющие ровно 5 различных делителей. Выведите эти делители для каждого найденного числа в порядке возрастания.
32) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [394441; 394505], числа, имеющие максимальное количество различных делителей. Если таких чисел несколько, то найдите **минимальное** из них. Выведите количество делителей найденного числа и два наибольших делителя в порядке убывания.
33) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [286564; 287270], числа, имеющие максимальное количество различных делителей. Если таких чисел несколько, то найдите **максимальное** из них. Выведите количество делителей найденного числа и два наибольших делителя в порядке убывания.
34) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [586132; 586430], числа, имеющие максимальное количество различных делителей. Найдите **минимальное** и **максимальное** из таких чисел. Для каждого из них в отдельной строчке выведите количество делителей и два наибольших делителя в порядке убывания.
35) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [394480; 394540], числа, имеющие максимальное количество различных делителей. Выведите информацию о таких числах, расположив их в порядке возрастания. Для каждого числа выведите его порядковый номер, количество делителей и два наибольших делителя в порядке убывания.
36) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [194441; 196500] числа (в порядке возрастания) с нечётным количеством делителей. Для каждого такого числа выведите его порядковый номер (начиная с единицы), само число, количество его делителей и делитель, квадрат которого равен этому числу.
37) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [248015; 251575] числа (в порядке возрастания) с нечётным количеством делителей, которые не делятся на 2. Для каждого такого числа выведите его порядковый номер (начиная с единицы), само число, количество его делителей и делитель, квадрат которого равен этому числу.
38) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [268220; 270335] число с максимальной суммой делителей, имеющее не более четырех делителей. Для найденного числа выведите сумму делителей, количество делителей и все делители в порядке убывания.
39) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [573213; 575340] число с минимальной суммой делителей, имеющее ровно четыре делителя. Для найденного числа выведите сумму делителей, количество делителей и все делители в порядке убывания.
40) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2943444; 2943529], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
41) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4671032; 4671106], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
42) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4202865; 4202923], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.

43) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [1547341; 1547409], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
44) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4837177; 4837236], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
45) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4301614; 4301717], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
46) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2358827; 2358891], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
47) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4730727; 4730817], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
48) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2484292; 2484370], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
49) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [1371085; 1371134], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
50) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2532421; 2532491], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
51) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3144472; 3144600], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
52) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [4409962; 4410101], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
53) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3614033; 3614116], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
54) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [6638225; 6638322], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
55) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [5336748; 5336834], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
56) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [5962464; 5962581], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
57) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [5408238; 5408389], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.

58) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [6080068; 6080176], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
59) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [7178551; 7178659], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
60) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3532000; 3532160], простые числа. Выведите все найденные простые числа в порядке убывания, слева от каждого числа выведите его номер по порядку.
61) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2532000; 2532160], простые числа. Выведите все найденные простые числа в порядке убывания, слева от каждого числа выведите его номер по порядку.
62) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [1532040; 1532160], простые числа. Выведите все найденные простые числа в порядке убывания, слева от каждого числа выведите его номер по порядку.
63) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2532000; 2532160] первые пять простых чисел. Выведите найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.
64) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2532000; 2532160], простые числа. Найдите все простые числа, которые заканчиваются на цифру 7. Выведите их в порядке возрастания, слева от каждого числа выведите его номер по порядку.
65) (**А.Н. Носкин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [2532000; 2532160], простые числа. Найдите все простые числа, но выведите на экран только каждое третье простое число (то есть числа с порядковыми номерами 1, 4, 7, 10, …). Вывод осуществите в порядке возрастания, слева от каждого числа выведите его собственный порядковый номер среди всех простых чисел.
66) (**Б.С. Михлин**) Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [194441; 196500] простые числа (т.е. числа у которых только два делителя: 1 и само число), оканчивающиеся на 93. Для каждого простого числа выведите его порядковый номер (начиная с единицы) и само число.
67) (**П.Е. Финкель, г. Тимашевск**) Уникальным назовём число, если у него только третья и пятая цифры чётные. Для интервала [33333;55555] найдите количество таких чисел, которые не делятся на 6, 7, 8 и найдите разность максимального и минимального из них.
68) (**П.Е. Финкель, г. Тимашевск**) Уникальным назовём число, если у него только первые две цифры нечётные. Для интервала [57888;74555] найдите количество таких чисел, которые не делятся на 7, 9, 13, и найдите разность максимального и минимального из них.
69) (**П.Е. Финкель, г. Тимашевск**) Уникальным назовём число, если у него только последние три цифры нечётные. Для интервала [64444;77563] найдите количество таких чисел, которые не делятся на 9, 13, 17, и найдите разность максимального и минимального из них.