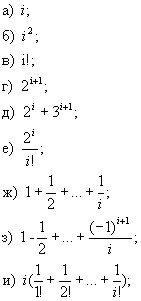
**139**Дано натуральное число n. Получить последовательность b1, ..., bn, где при i = 1, 2, ..., n значение b i равно:



**Type**

Mass = **array** [1..10000] **of** integer;

**var**

i,n:integer;

b:mass;

**begin**

write('n:=');

readln(n);

**for** i:=1 **to** n **do**

**begin**

b[i]:= i\*i;

writeln ('b=',b[i]);

**end**;

**end**

**в текстовый файл**

Mass = **array** [1..10000] **of** integer;

**var**

i,n:integer;

b:mass;

t: text;

**begin**

assign(t,'pyt.txt');

reset (t);

read(t,n);

close (t);

append (t);

//writeln (t,'');

**for** i:=1 **to** n **do**

**begin**

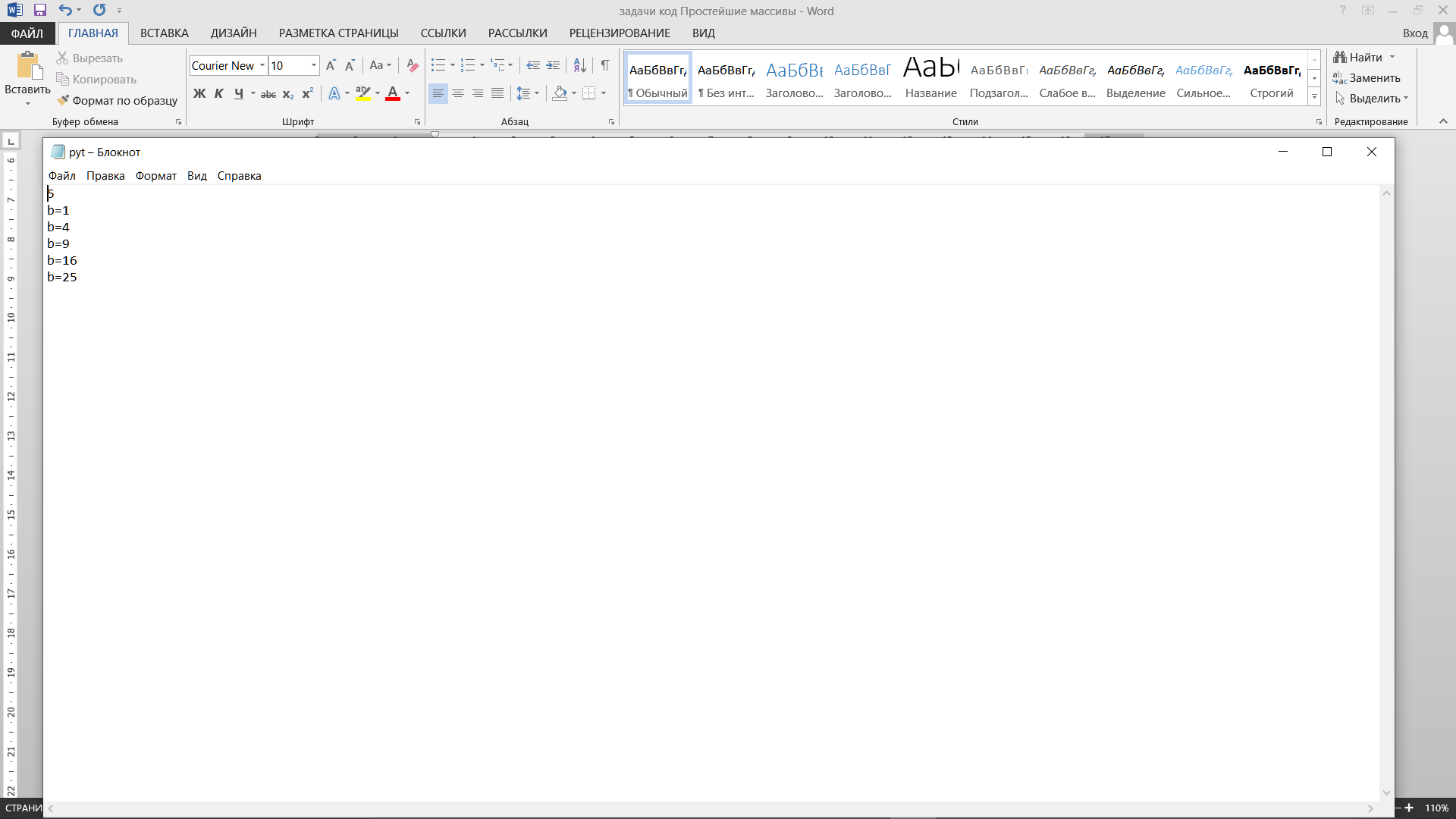
b[i]:= i\*i;

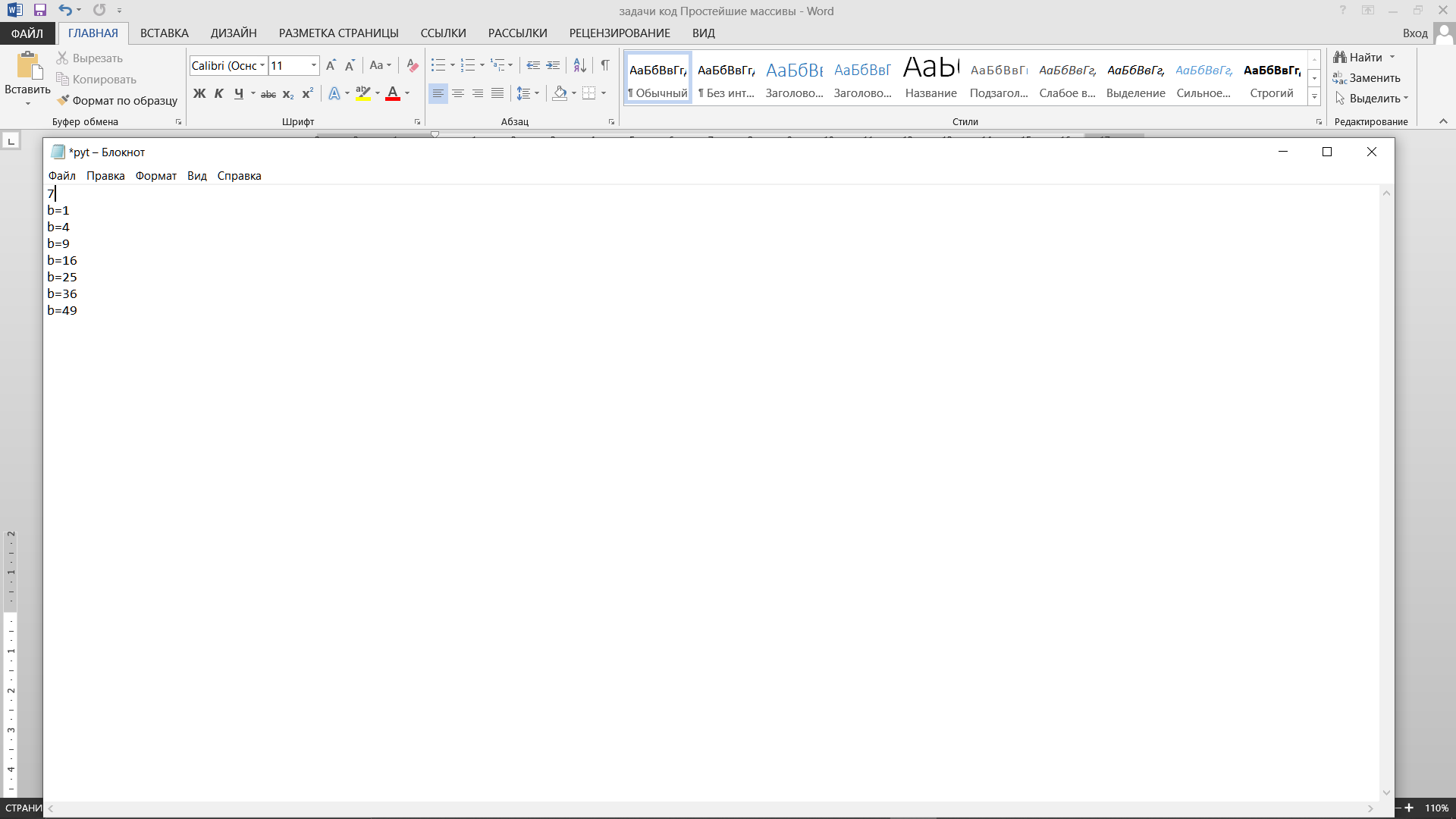
writeln (t,'b=',b[i]);

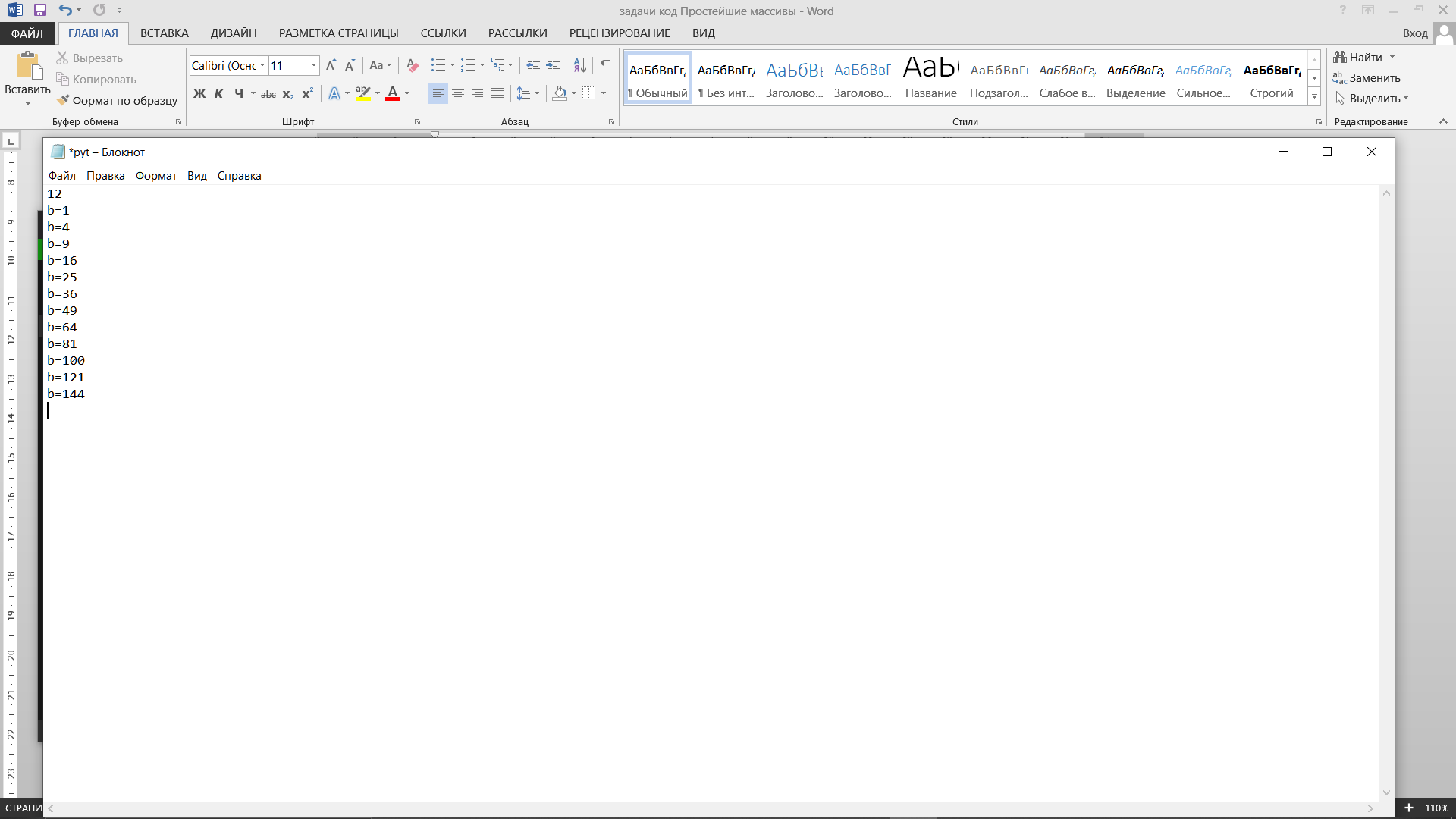
**end**;

close (t);

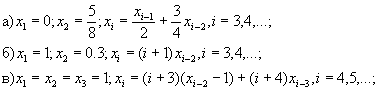
**end**.







**145**Последовательность x1, x2, ...образована по закону:



Получить x1, x2, ..., x20.

**type**

Mass = **array** [1..10000] **of** integer;

**var**

i : integer;

a, b, c : real;

f: mass;

**const** n = 20;

**begin**

(a, b, c) := (1, 1, 1);

write(a, ' ', b, ' ', c, ' ');

**for** i:=4 **to** 20 **do**

**begin**

(a, b, c) := (b, c, (i+3)\*(c-1)+(i+4)\*a);

write(c, ' ');

**end**;

**end**.

Вывод в текстовый файл

**type**

Mass = **array** [1..100] **of** integer;

**var**

i : integer;

a, b, c : real;

f: mass;

ftxt: text;

**begin**

assign (ftxt,'145.txt');

append (ftxt);

writeln (ftxt, '');

f[1]:=1;

f[2]:=1;

f[3]:=1;

write(ftxt, f[1], ' ', f[2], ' ', f[3], ' ');

**for** i:=4 **to** 20 **do**

**begin**

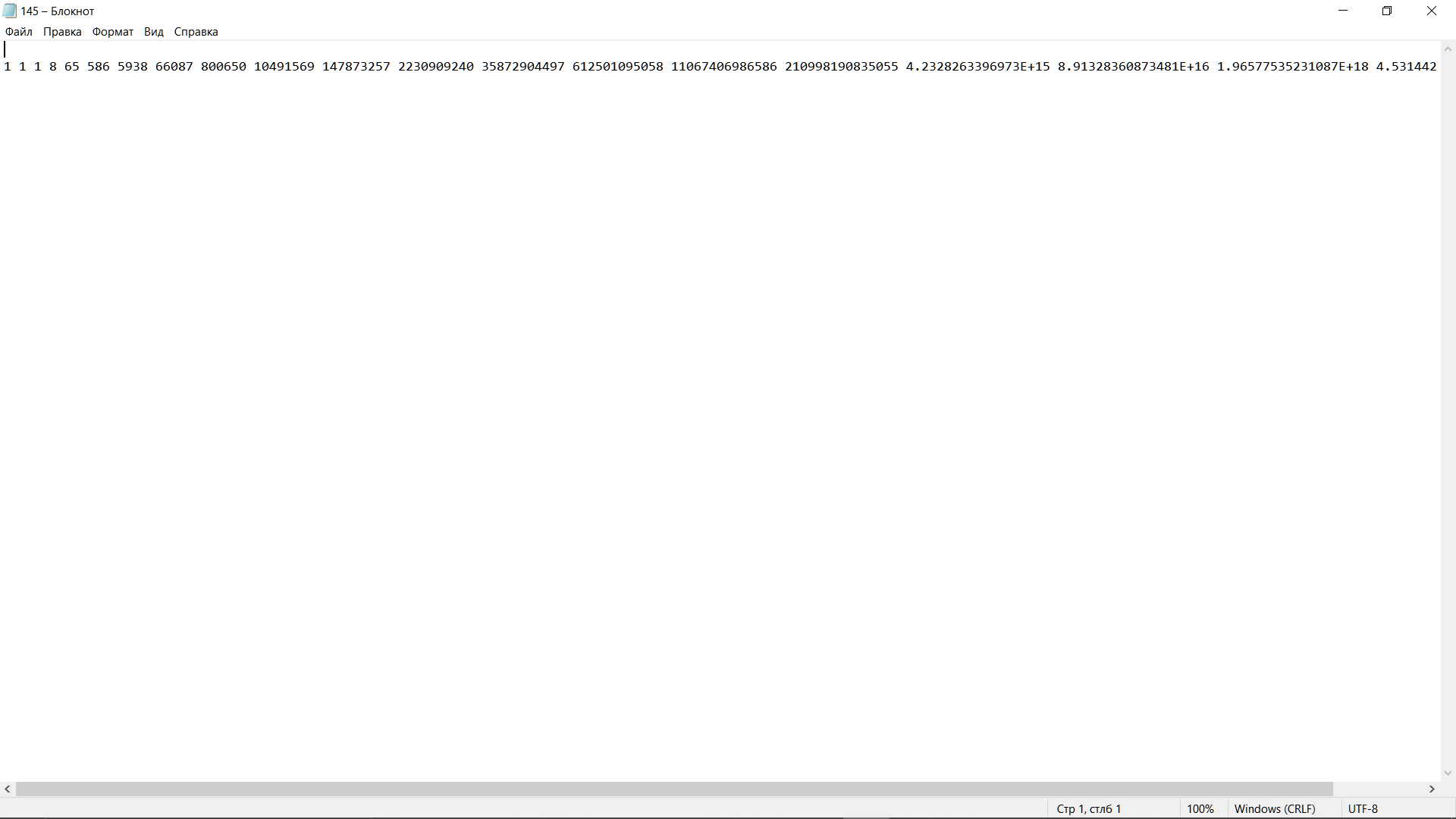
f[I]:= (i+3)\*(f[i-2]-1)+(i+4)\*f[i-3];

write(ftxt, f[i], ' ');

**end**;

close (ftxt);

**end**.



1 1 1 8 65 586 5938 66087 800650 10491569 147873257 2230909240 35872904497 612501095058 11067406986586 210998190835055 4.2328263396973E+15 8.91328360873481E+16 1.96577535231087E+18 4.53144209353026E+19

**164**Даны действительные числа a1,..., a50. Получить "сглаженные" значения a1,..., a50, заменив в исходной последовательности все члены, кроме первого и последнего, по формуле

http://bgu-chita.ru/zadachnik/Glava06/image12.gif  
б) при "сглаживании" используются лишь старые значения членов.

**const** n=50;

**Type**

Mass = **array** [1..100] **of** real;

**var**

a:**array**[1..n] **of** real;

xp,xs:real;

i:byte;

f: mass;

**begin**

randomize;

writeln('Исходный массив:');

**for** i:=1 **to** n **do**

**begin**

a[i]:=10\*random;

write(a[i]:5:1);

**end**;

writeln;

writeln;

writeln('Сглаженный массив:');

write(a[1]:5:1);

**for** i:=2 **to** n-1 **do**

**begin**

xp:=a[i-1];

xs:=a[i+1];

a[i]:=(xp+a[i]+xs)/3;

write(a[i]:5:1);

**end**;

write(a[n]:5:1);

readln

**end**.

**Вывод в текст**

**Type**

Mass = **array** [1..100] **of** real;

**const** n=50;

**var**

xp,xs:real;

i:byte;

f,a: mass;

ftxt: text;

**begin**

assign (ftxt,'164.txt');

append (ftxt);

writeln (ftxt, '');

randomize;

writeln(ftxt,'Исходный массив:');

**for** i:=1 **to** n **do**

**begin**

a[i]:=10\*random;

write(ftxt,a[i]:5:1);

**end**;

writeln;

writeln;

writeln(ftxt,'Сглаженный массив:');

f[1]:=a[1];

f[n]:= a[n];

write(ftxt,f[1]:5:1);

**for** i:=2 **to** n-1 **do**

**begin**

f[i]:=(a[i-1]+a[i]+a[i+1])/3;

write(ftxt,f[i]:5:1);

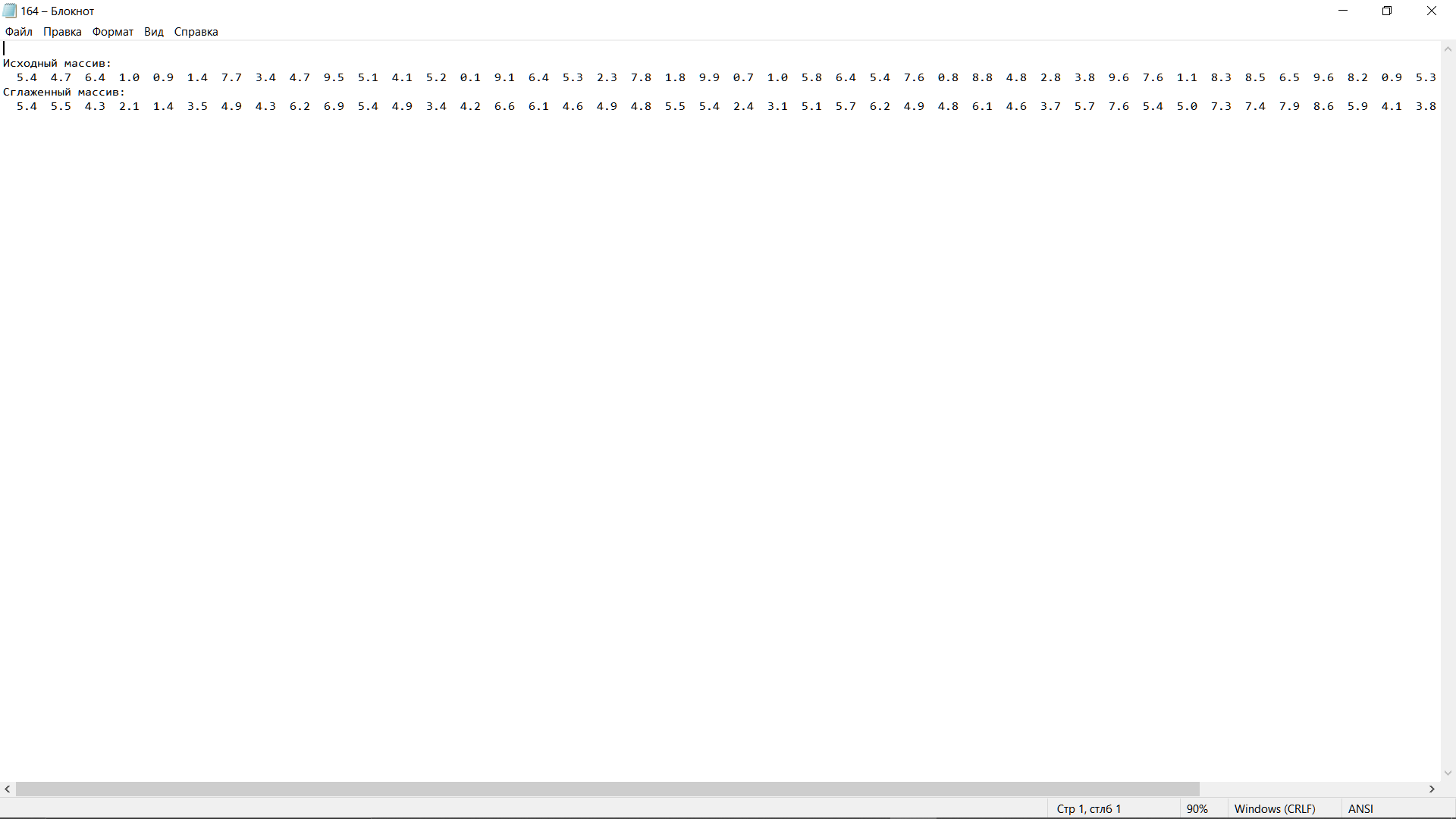
**end**;

write(ftxt,f[n]:5:1);

close (ftxt);

readln

**end**.

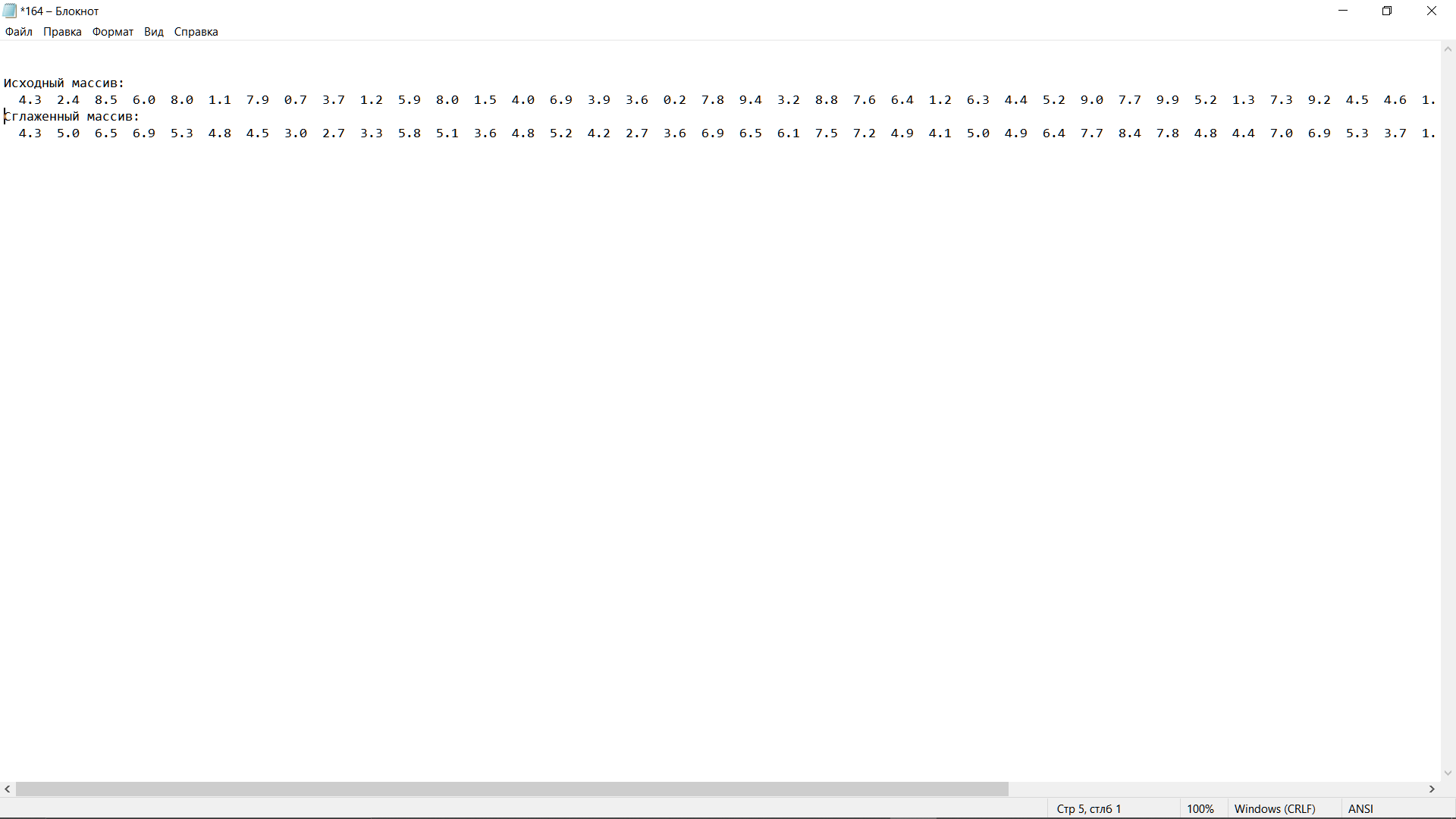


Исходный массив:

5.4 4.7 6.4 1.0 0.9 1.4 7.7 3.4 4.7 9.5 5.1 4.1 5.2 0.1 9.1 6.4 5.3 2.3 7.8 1.8 9.9 0.7 1.0 5.8 6.4 5.4 7.6 0.8 8.8 4.8 2.8 3.8 9.6 7.6 1.1 8.3 8.5 6.5 9.6 8.2 0.9 5.3 2.1 9.2 9.0 4.7 6.1 0.7 6.7 3.9

Сглаженный массив:

5.4 5.5 4.3 2.1 1.4 3.5 4.9 4.3 6.2 6.9 5.4 4.9 3.4 4.2 6.6 6.1 4.6 4.9 4.8 5.5 5.4 2.4 3.1 5.1 5.7 6.2 4.9 4.8 6.1 4.6 3.7 5.7 7.6 5.4 5.0 7.3 7.4 7.9 8.6 5.9 4.1 3.8 5.0 7.7 7.1 6.0 4.3 3.9 4.8 3.9



Исходный массив:

4.3 2.4 8.5 6.0 8.0 1.1 7.9 0.7 3.7 1.2 5.9 8.0 1.5 4.0 6.9 3.9 3.6 0.2 7.8 9.4 3.2 8.8 7.6 6.4 1.2 6.3 4.4 5.2 9.0 7.7 9.9 5.2 1.3 7.3 9.2 4.5 4.6 1.1 0.2 4.3 2.3 2.2 2.2 9.8 9.8 2.0 0.2 7.4 6.9 8.0

Сглаженный массив:

4.3 5.0 6.5 6.9 5.3 4.8 4.5 3.0 2.7 3.3 5.8 5.1 3.6 4.8 5.2 4.2 2.7 3.6 6.9 6.5 6.1 7.5 7.2 4.9 4.1 5.0 4.9 6.4 7.7 8.4 7.8 4.8 4.4 7.0 6.9 5.3 3.7 1.7 2.1 2.9 2.5 2.3 4.8 8.1 6.6 3.0 3.5 6.0 7.0 8.0