**РЕШЕНИЯ ЗАДАЧ**

**Задача 2**

Как может показаться, для решения задачи достаточно описать массив необходимого размера, заполнить его нулями и затем при чтении данных из файла сразу увеличивать элемент массива с индексом, соответствующим значению считываемого элемента, на единицу. После окончания чтения данных из файла остается просмотреть весь массив и вывести в качестве результата те номера индексов массива, которые будут содержать в качестве значения единицы – это и будут элементы, встречающиеся по одному разу. Однако для данной постановки задачи это – неверное решение, так как в условии сказано, что дан одномерный массив, содержащий элементы. Поэтому прежде, чем решать задачу, мы должны обязательно поместить данные из файла в массив (файл здесь выступает лишь для удобства ввода данных в программу).

Допустим, мы учли отмеченный момент, и теперь для решения задачи нам достаточно использовать еще один массив такой же размерности, и уже в него записывать количество раз встречаемости соответствующих элементов исходного массива. Но в этом случае мы будем нерационально использовать память. Значит, надо попытаться отыскать другой способ решения.

Такой способ существует, и заключается в следующем. Мы знаем, что значения элементов массива не превышают *n*. Поэтому задача может быть решена в два прохода по массиву. Во время первого прохода к элементу, номер которого соответствует значению элемента первоначального массива, добавляем значение, равное *n+*1. Во время второго прохода остается проверить полученный массив и выбрать номера элементов, значения которых изменились только на *n+*1. Такие номера и будут значениями элементов первоначального массива, которые встречались только по одному разу (если какие-то элементы не изменялись вообще, то это означает, что ).

Для того, чтобы во время первого прохода по массиву знать первоначальное значение исходного элемента массива (в результате операций суммирования на предыдущих шагах любой элемент исходного массива мог измениться), необходимо рассматривать не сам этот элемент, а остаток от деления этого элемента на значение, равное *n+*1. На рис. 1 показана схема, иллюстрирующая описываемый метод при . При этом на схеме снизу обведены номера индексов, показывающих значения неповторяющихся элементов в массиве: таким образом, во введенном массиве по одному разу встречаются элементы 2, 3, 4, 5, 10 и 12.

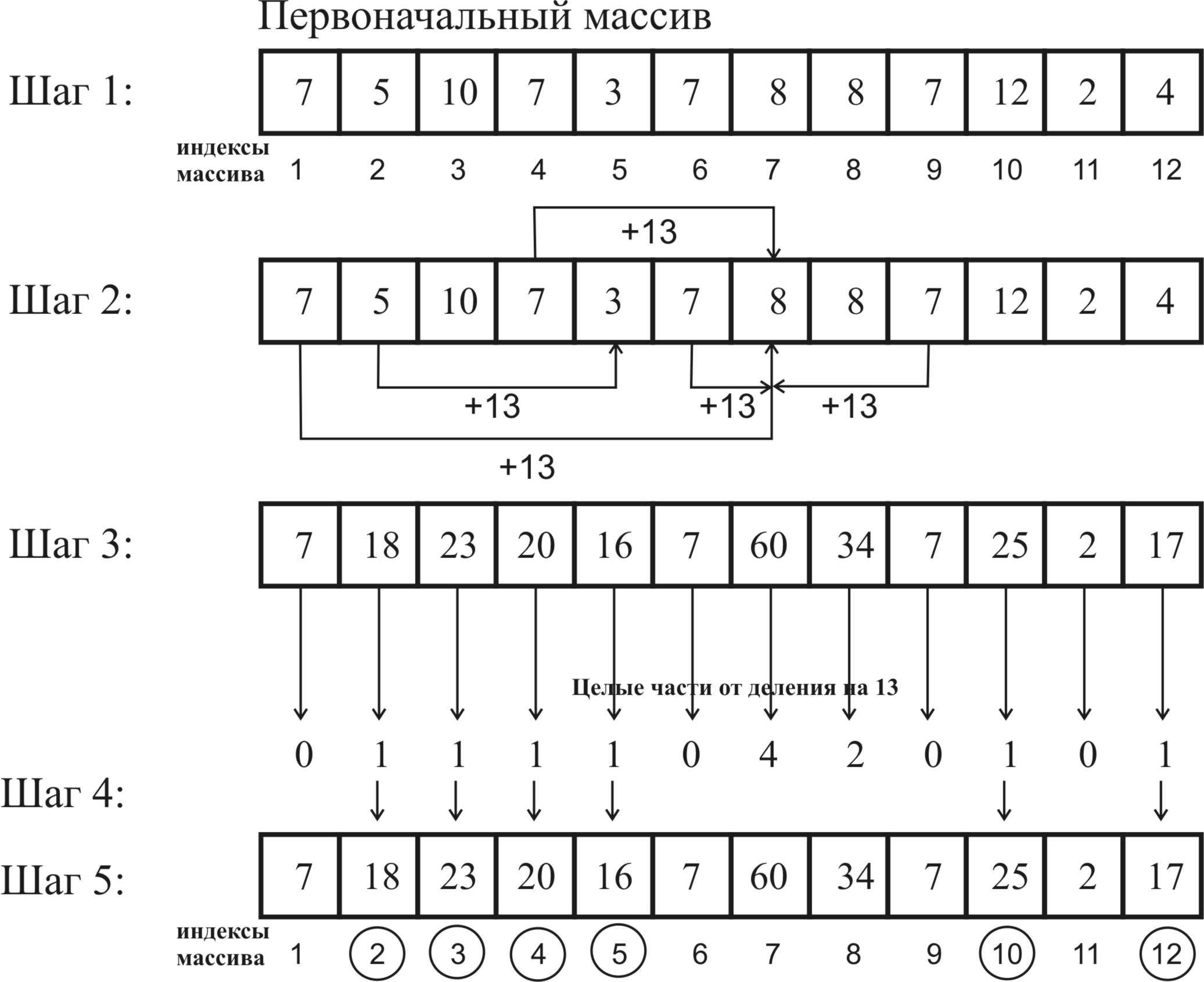


Рис. 1. Иллюстрация метода решения к задаче 1

Исходный код программы на Free Pascal:

*Uses Crt;*

*Type Mas=array [0..1000000] of Longint;*

*Const a: ^Mas=nil;*

*Var i,n: Longint;*

*f: Text;*

*Begin*

*assign(f,'in.txt');*

*reset(f);*

*readln(f,n);*

*GetMem(a,n\*SizeOf(Longint));*

*for i:=0 to n-1 do read(f,a[i]);*

*close(f);*

*for i:=0 to n-1 do inc(a[(a[i] mod (n+1))-1],n+1); // собственно, само решение*

*assign(f,'out.txt');*

*rewrite(f);*

*for i:=0 to n-1 do if (trunc(a[i]/(n+1)) = 1) then write(f,i+1,' ');*

*close(f);*

*FreeMem(a,n\*SizeOf(Longint));*

*a:=nil;*

*End.*

Исходный код программы на С++:

*#include <iostream>*

*#include <fstream>*

*using namespace std;*

*int main()*

*{ ifstream F;*

*long i=0, n;*

*long \*a= new long [n];*

*F.open("IN.txt",ios::in);*

*if (F) { F>>n;*

*while (!F.eof()) F>>a[i++]; } else cout<<"Файл не найден!"<<endl;*

*F.close();*

*for(i=0; i<n; i++) a[a[i]%(n+1)-1]+=(n+1); // собственно, само решение*

*ofstream f;*

*f.open("OUT.txt");*

*for(i=0; i<n; i++) if ((int)a[i]/(n+1) == 1) f<<(i+1)<<" ";*

*f.close();*

*delete[]a; return 0; }*

**Задача 3**

В условии данной задачи, напротив, про массив ничего не сказано. Более того, указано, что строки входного файла и представляют собой журнал службы безопасности. Поэтому задачу можно решать и без записи данных из файла в массив. К возможным способам решения данной задачи можно отнести, например, следующие.

1. Организовать одномерный массив целых чисел, заполненный изначально нулями. После этого при чтении данных из файла значение элемента массива, индекс которого совпадает с прочитанным из файла значением, увеличивать на единицу. По завершении чтения данных просмотреть массив и вывести индекс того элемента, который равен единице – это и будет житель без пары. При таком способе решения крайне неэффективно будет расходоваться память.
2. Осуществить полный перебор всех элементов массива, используя вложенный цикл, – в этом случае трудоемкость алгоритма будет близка к , что приведет к медленной работе алгоритма.
3. Отсортировать исходный массив (например, по возрастанию), и затем исследовать соседние элементы – как только найдется элемент без пары, – задача решена. Такой способ по скорости работы будет сходен с предыдущим.

Тем не менее, оптимальное решение этой задачи не предусматривает использования какого-либо массива, и основано на применении инструкции XOR (исключающее ИЛИ) к элементам, хранящимся в файле. При решении задачи таким способом (да и другими тоже) необходимо исключить из рассмотрения последний элемент в файле, так как он хранит количество элементов, и при решении не требуется.

Исходный код программы на Free Pascal:

*Var a,b: Longint;*

*f: Text;*

*Begin*

*assign(f,'in.txt');*

*reset(f);*

*b:=0;*

*while not eof(f) do*

*begin*

*read(f,a); write(a,' ');*

*if not eof(f) then b:= b xor a;*

*end;*

*close(f);*

*assign(f,'out.txt');*

*rewrite(f);*

*write(f,b);*

*close(f);*

*End.*

Исходный код программы на языке С++:

*#include <iostream>*

*#include <fstream>*

*using namespace std;*

*int main()*

*{ long a, b=0;*

*ifstream F;*

*F.open("IN.txt",ios::in);*

*if (F) {*

*while (!F.eof())*

*{ F>>a;*

*if (!F.eof()) b=b^a; } } else cout<<"Файл не найден!"<<endl;*

*F.close();*

*ofstream f;*

*f.open("OUT.txt");*

*f<<b;*

*f.close(); return 0; }*

Приведенный метод на основе XOR лишен практически всех недостатков, которые могут проявиться при любых других способах решения этой задачи.

**Задача 4**

Основной трудностью при решении этой задачи является разработка алгоритма, позволяющего перебирать все возможные комбинации элементов, т.е., по сути, это задача на комбинаторику. Для таких задач известно, что если количество элементов равно , то количество возможных комбинаций будет равно .

При решении задачи необходимо использовать правило перехода от одного сочетания к другому. Переход к следующему сочетанию осуществляем так: просматриваем текущее сочетание с конца и ищем элемент, который еще можно увеличить. Когда элементы увеличивать уже нельзя, переходим к элементу левее, и увеличиваем его на 1. Оставшуюся часть сочетания заполняем в порядке следования числами, которые больше измененного элемента. Ниже приведены коды программ с повторами комбинаций.

Исходный код программы на Free Pascal:

*var a,b: array[0..30] of integer;*

*i,j,n,m,s,count: integer;*

*st: string;*

*f,ff: Text;*

*begin*

*assign(f,'in.txt');*

*reset(f);*

*readln(f,n);*

*for i:=0 to n-1 do read(f,b[i]);*

*close(f);*

*count:=0;*

*assign(f,'out1.txt');*

*rewrite(f);*

*for m:=1 to n-1 do*

*begin*

*for i:=0 to m do a[i]:=i; {первоначальное сочетание в группе}*

*while i>0 do*

*begin*

*i:=m; s:=0;*

*for j:=1 to m do s:=s+b[a[j]-1]; {подсчёт суммы}*

*if s=0 then begin {сумма = 0, - записываем вариант в файл}*

*writeln(f,''); writeln(f,'');*

*inc(count);*

*for j:=1 to m do write(f,b[a[j]-1],' ');*

*end;*

*while a[i]=n-m+i do dec(i); {ищем элемент для очередного изменения}*

*{(в столбце слева)}*

*inc(a[i]);*

*for j:=i+1 to m do a[j]:=a[j-1]+1; {изменяем правую часть сочетания}*

*end;*

*end;*

*close(f);*

*assign(f,'out.txt');*

*assign(ff,'out1.txt');*

*rewrite(f);*

*reset(ff);*

*writeln(f,count); {записываем количество сумм в начало файла}*

*readln(ff,st);*

*while not eof(ff) do*

*begin*

*readln(ff,st); writeln(f,st);*

*end;*

*close(f); close(ff);*

*erase(ff);*

*end.*

Исходный код программы на языке С++:

*#include <iostream>*

*#include <fstream>*

*using namespace std;*

*int a[31], b[30], i=0, j, n, m, s, count=0;*

*ifstream F;*

*fstream f;*

*int main()*

*{ F.open("IN.txt",ios::in);*

*f.open("OUT.txt", ios::out);*

*if (F) { F>>n;*

*while (!F.eof()) F>>b[i++]; } else cout<<"File not found!"<<endl;*

*F.close();*

*f<<endl; //оставляем место в файле для записи кол-ва вариантов сумм*

*for(m=1; m<n; m++)*

*{ for(i=0; i<=m; i++) a[i]=i; //первоначальное сочетание в группе*

*for( ; i>0; )*

*{ i=m; s=0;*

*for(j=1; j<=m; j++) s+=b[a[j]-1]; //подсчёт суммы*

*if (s==0) { f<<endl<<endl; //сумма = 0, - записываем вариант в файл*

*count++;*

*for(j=1; j<=m; j++) f<<b[a[j]-1]<<" "; }*

*while (a[i]==n–m+i) i--; //ищем элемент для очередного изменения*

*//(в столбце слева)*

*a[i]++;*

*for(j=i+1; j<=m; j++) a[j]=a[j-1]+1; } } //изменяем правую часть*

*//сочетания*

*f.seekg(0);*

*f<<count; //записываем количество сумм в начало файла*

*f.close(); return 0; }*

**Задача 5**

Самый простой вариант решения задачи – это открыть цикл от 10 (например) и до указанного во входном файле числа. В цикле переводить очередное число в строку для более удобного выделения цифр числа, выполнять требуемые операции и искать массы репки:

*program N1;*

*{$APPTYPE CONSOLE}*

*uses SysUtils, DateUtils;*

*var i,n,s,max: longint;*

*a,j,k,nn,c: integer;*

*ss: string;*

*f: text;*

*t1,t2: TDateTime;*

*begin*

*t1:=Now;*

*assign(f,'in.txt');*

*reset(f);*

*read(f,max);*

*close(f);*

*assign(f,'out.txt');*

*rewrite(f);*

*for i:=10 to max do*

*begin*

*s:=0;*

*str(i,ss);*

*nn:=length(ss);*

*for j:=1 to nn do*

*begin*

*val(ss[j],a,c);*

*n:=1;*

*for k:=1 to j do n:=n\*a;*

*s:=s+n;*

*end;*

*if s=i then writeln(f,i);*

*end;*

*close(f);*

*t2:=Now;*

*writeln('Time ',MilliSecondsBetween(t2,t1)/1000:5:3,' s.');*

*write('Program is complete');*

*readln;*

*end.*

Тестирование этой версии программы и следующих проведем на компьютере с процессором Core i3-3240 (3,4 ГГц).

Взяв в качестве входного значения 108, получим время работы 103,6 секунды.

Одной из слабых сторон алгоритма является повторный подсчет степеней цифр, составляющих числа, поэтому введем двумерный массив для хранения предварительно подсчитанных степеней всех цифр. Опишем функцию вычисления степени цифры, а также откажемся от использования процедуры val:

*program N2;*

*{$APPTYPE CONSOLE}*

*uses SysUtils, DateUtils;*

*var i,m,s,max: longint;*

*k,j: integer;*

*a: array[1..10,1..10] of longint;*

*ss: string;*

*f: text;*

*t1,t2: TDateTime;*

*function stepen(x,y: byte): longint;*

*var i: byte; z: longint;*

*begin*

*z:=1;*

*for i:=1 to y do z:=z\*x;*

*stepen:=z;*

*end;*

*begin*

*t1:=Now;*

*assign(f,'in.txt');*

*reset(f);*

*read(f,max);*

*close(f);*

*assign(f,'out.txt');*

*rewrite(f);*

*for i:=0 to 9 do*

*for j:=1 to 10 do a[i+1,j]:=stepen(i,j);*

*for i:=10 to max do*

*BEGIN*

*if i mod 10 <> 0 then*

*begin*

*m:=i;*

*s:=0;*

*str(i,ss);*

*k:=length(ss);*

*while m<>0 do*

*begin*

*j:=m mod 10;*

*m:=m div 10;*

*s:=s+a[j+1,k];*

*k:=k-1;*

*if s>i then m:=0;*

*end;*

*if s=i then writeln(f,i);*

*end;*

*END;*

*close(f);*

*t2:=Now;*

*writeln('Time ',MilliSecondsBetween(t2,t1)/1000:5:3,' s.');*

*write('Program is complete');*

*readln;*

*end.*

Взяв в качестве входного значения 108, получим время работы уже 18,7 секунды, т.е. усовершенствованный код работает примерно в 5,5 раз быстрее.

Можно написать код еще эффективнее, если избавиться от использования в программе процедуры str, которая используется в программе для получения количества цифр числа. Сделать это можно, введя дополнительную переменную v, смысл которой понятен из кода программы:

*program N3;*

*{$APPTYPE CONSOLE}*

*uses SysUtils, DateUtils;*

*var i,m,s,v,max: longint;*

*k,j,n: integer;*

*a: array[1..10,1..10] of longint;*

*f: text;*

*t1,t2: TDateTime;*

*function stepen(x,y: byte): longint;*

*var i: byte; z: longint;*

*begin*

*z:=1;*

*for i:=1 to y do z:=z\*x;*

*stepen:=z;*

*end;*

*begin*

*t1:=Now;*

*assign(f,'in.txt');*

*reset(f);*

*read(f,max);*

*close(f);*

*assign(f,'out.txt');*

*rewrite(f);*

*for i:=0 to 9 do*

*for j:=1 to 10 do a[i+1,j]:=stepen(i,j);*

*n:=2; v:=100;*

*for i:=10 to max do*

*BEGIN*

*if i mod 10 <> 0 then*

*begin*

*m:=i;*

*s:=0;*

*k:=n;*

*while m<>0 do*

*begin*

*j:=m mod 10;*

*m:=m div 10;*

*s:=s+a[j+1,k];*

*k:=k-1;*

*if s>i then m:=0;*

*end;*

*if s=i then writeln(f,i);*

*end else if i mod v = 0 then*

*begin*

*v:=v\*10;*

*n:=n+1;*

*end;*

*END;*

*close(f);*

*t2:=Now;*

*writeln('Time ',MilliSecondsBetween(t2,t1)/1000:5:3,' s.');*

*write('Program is complete');*

*readln; end.*

Взяв в качестве входного значения 108, получим время работы 6,0 секунд, т.е. еще в 3 раза быстрее. В итоге, усовершенствованный код эффективнее первоначального примерно в 17 раз.

**Задача 6**

*Type Mas=array [byte] of byte;*

*Var N,i,L: byte;*

*a: Mas;*

*f: Text;*

*procedure Next(var a: Mas; var L: byte);*

*var i,j:byte;*

*s:word;*

*begin*

*i:=L-1;*

*s:=a[L];*

*while (i>1) and (a[i-1]<=a[i]) do*

*begin s:=s+a[i];*

*i:=i-1;*

*end;*

*a[i]:=a[i]+1;*

*L:=i+s-1;*

*for j:=i+1 to L do a[j]:=1;*

*end;*

*Begin*

*assign(f,'in.txt');*

*reset(f);*

*read(f,N);*

*close(f);*

*assign(f,'out.txt');*

*rewrite(f);*

*L:=N;*

*for i:=1 to L do a[i]:=1;*

*writeln;*

*repeat*

*for i:=1 to L-1 do write(f,a[i],' + ');*

*write(f,a[L]);*

*writeln(f,' ');*

*Next(a,L);*

*until L=1;*

*writeln(f,a[L]);*

*close(f);*

*end.*

**Задача 7**

Решение задачи основано на отбрасывании более высоких степеней неизвестного слагаемого (приближения) при решении уравнения вида , где  – неизвестное приближение,  – некоторое начальное приближение (например, равное 1):



или, отбрасывая степени порядка 2 и 3 при :



откуда .

На каждом следующем этапе присваиваем значению  полученное значение  и повторяем процесс поиска , тем самым уточняя его.

Исходный код программы будет иметь вид:

*var x,x0,a,e,j: double;*

*i: integer;*

*f: Text;*

*begin*

*assign(f,'in.txt');*

*reset(f); read(f,x,e);*

*close(f);*

*x0:=1; i:=0; j:=e;*

*a:=(x-x0\*x0\*x0)/(3\*x0\*x0);*

*if a<0 then e:=-a else e:=a;*

*while e>=1E-10 do*

*begin*

*x0:=x0+a; i:=i+1;*

*a:=(x-x0\*x0\*x0)/(3\*x0\*x0);*

*if a<0 then e:=-a else e:=a;*

*end;*

*i:=0;*

*while j<1 do begin {определяем число знаков после запятой}*

*inc(i); j:=j\*10;*

*end;*

*assign(f,'out.txt');*

*rewrite(f); write(f,x0:0:i);*

*close(f);*

*end.*