Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по разминке №7

Тема работы: Длинная арифметика: деление

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Исходная постановка задачи 3](#_Toc120385659)

[2 Дополненная постановка задачи 4](#_Toc120385660)

[3 Методика решения 5](#_Toc120385661)

[3.1 Описание оператора try..except..end и функции Pos 5](#_Toc120385662)

[3.1.1 Оператор try..except..end 5](#_Toc120385663)

[3.1.2 Функция Pos и использование константной строки 5](#_Toc120385664)

[3.2 Условия ввода 5](#_Toc120385665)

[3.3 Проверка введенных данных и запись числа в массив 6](#_Toc120385666)

[3.3.1 Проверка ввода системы счисления и количества чисел для деления 6](#_Toc120385667)

[3.3.2 Проверка корректности введенного числа и его запись в массив 6](#_Toc120385668)

[3.4 Деление чисел 7](#_Toc120385669)

[3.5 Вывод частного и остатка 8](#_Toc120385670)

[4 Структура данных 9](#_Toc120385671)

[Приложение А 11](#_Toc120385672)

[Приложение Б 26](#_Toc120385673)

# Исходная постановка задачи

Даны два целых числа в десятичной системе счисления до 50 цифр, первое число является делителем другого. Делимое >=0, делитель >0. Необходимо найти их частное.

# Дополненная постановка задачи

Вводится необходимая система счисления NS (максимальная система счисления – 20-ая) и количество чисел для деления n (не больше 50). Числа должны быть целыми, не более 256 символов, делимое должно быть >=0, делители >0 и делимое должно быть больше либо равно итоговому делителю. Необходимо найти частное.

# [Методика решения](#_Toc83996305)

## Описание оператора try..except..end и функции Pos

### Оператор try..except..end

Оператор try..except..end имеет вид:

try  
 операторы;  
except  
  блок обработки исключений;  
end;

Выполнение блока начинается с секции try, при отсутствии исключительных ситуаций только она и выполняется. Секция except получает управление в случае возникновения исключительной ситуации. После обработки выполняются операторы, стоящие после end.

### Функция Pos и использование константной строки

Функция Pos имеет вид:

Pos(Substr, S)

Функция возвращает индекс символа, с которого начинается первое вхождение подстроки Substr в строку S. Eсли заданная подстрока не встречается в заданной строке, то функция вернет 0.

В данной задачи в роли подстроки Substr применяется символ из введенной строки; в роли строки S применяется константная строка, в которой индекс элемента уменьшенный на единицу соответствует численному значению символа в этом элементе:

NSAlphabet = ''

## Условия ввода

Условия ввода:

* система счисления NS должна быть целочисленной, больше либо равно 2 и меньше либо равно 20;
* количество чисел для деления n должно быть целочисленным, больше 1 и меньше либо равно 50;
* при вводе чисел, длина должна быть не больше 256 символов и значение каждого символа должно соответствовать введенной системе счисления NS. Также делимое должно быть больше либо равно итоговому делителю, делимое должно быть >=0, делители >0

## Проверка введенных данных и запись числа в массив

### Проверка ввода системы счисления и количества чисел для деления

Проверка ввода системы счисления NS и количества чисел для деления n происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора try..except..end (описание оператора см. [главу 3.1.1](#_Оператор_try..except..end)) проверяем целочисленность; оператором if проверяет принадлежность заданному промежутку.

### Проверка корректности введенного числа и его запись в массив

Проверка корректности введенного числа происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора if проверяем условие: длина должна быть не больше 256 символов.

Так в данной задаче необходимо работать с очень большими числами (которые не входят ни в один тип данных), необходимо записывать каждую цифру числа в отдельный элемент массива. В данной задаче для нахождения частного чисел надо посчитать произведение делителей (подробнее см. [главу 3.4](#_Деление_чисел)). Так как в результате умножения количество цифр произведения может быть больше, чем количества цифр изначальных значений, то будем записывать числа с конца для увеличения размера числа.

Перед началом записи числа в массив заполним все его элементы 0 с помощью цикла с параметром for. Далее для записи числа в массив используем цикл с предусловием while, в котором условие входа – i <= len и flag = false (i – счетчик цикла, len – длина введенной строки, flag – индикатор проверки правильности ввода). В теле цикла для перевода из символа в его численное значение используем константную строку NSAlphabet (в которой индекс элемента уменьшенный на единицу соответствует численному значению символа в этом элементе) и процедуру Pos (описание процедуры и константной строки см. [главу 3.1.2](#_Функция_Pos_и)). Далее проверяем корректность (символ должен соответствовать введенной системе счисления и делители должны <>0) с помощью оператора if. Если соответствует, то численное значение символа записываем в элемент массива.

## Деление чисел

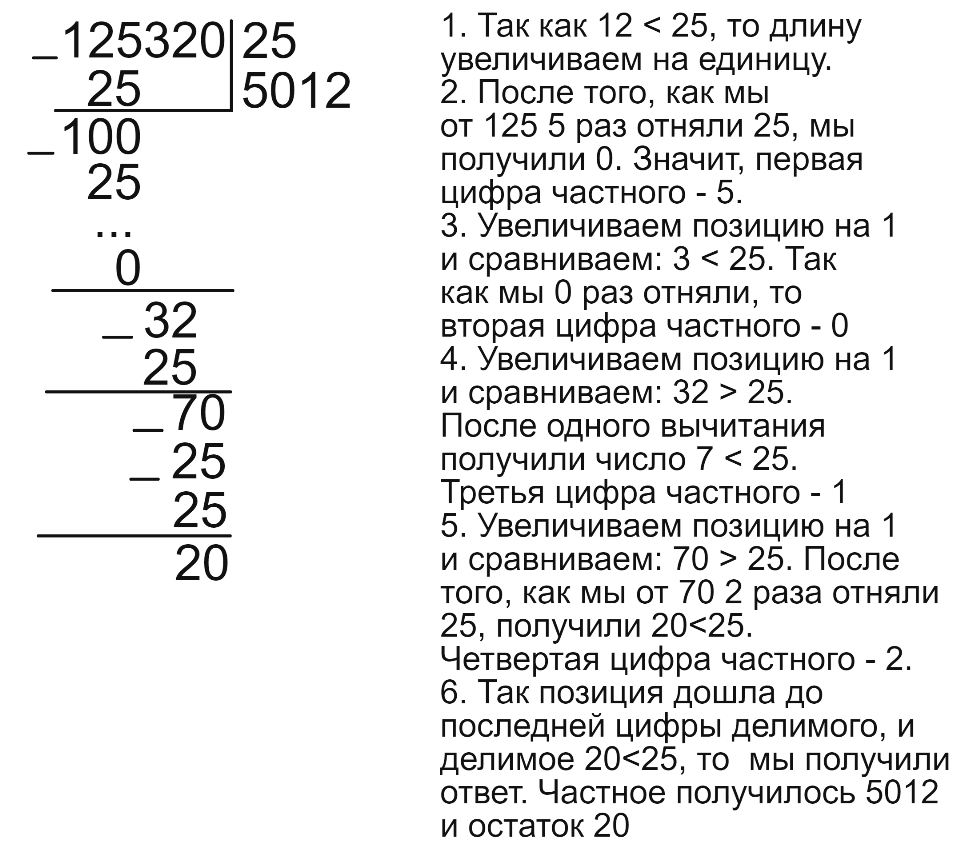
Деление чисел происходит в один этап: первое число делится на произведение делителей. То есть, запись вида a/b/c можно записать a/(b\*c), подробнее об произведении чисел см. [ReportForWarmUp6](../WarmUp6/ReportForWarmUp6.docx).

Деление двух чисел происходит с помощью метода деления столбиком. Перед началом делением проверяем условие: делимое должно быть больше либо равно итоговому делителю.

Так как числа записаны с конца, действовать начнем с конца. Делаем следующие действия:

1. Выделяем длину делителя в делимом и сравнивается. Если выделенное делимое меньше, то длину увеличиваем на единицу.
2. Пока выделенное делимое не станет меньше делителя делаем: от выделенного делимого вычитаем делитель (вычитание чисел см. [ReportForWarmUp5](../WarmUp5/ReportForWarmUp5.docx)). Количество вычитаний – это значение частного.
3. Далее позицию увеличаем на единицу и делаем пункт 2. до тех пора, пока позиция не дойдет до последней цифры делимого и делимое не будет меньше делителя.

Рассмотрим пример:



## Вывод частного и остатка

Для перевода из численного значения символа в символ используем константную строку NSAlphabet, так как данная строка хранит символы, в которых индекс элемента уменьшенный на единицу соответствуют численному значению символа в этом элементе.

Частное записано сначала, значит вывод элементов массива следует начинать сначала. Для вывода частного используем цикл с параметром for. Выводить символы будем из константной строки NSAlphabet по индексу текущего значения элемента из массива частного, увеличенного на единицу.

Остаток делимого записан с конца, значит вывод элементов массива следует начинать с конца. Для вывода остатка используем цикл с параметром for. Выводить символы будем из константной строки NSAlphabet по индексу текущего значения элемента из массива делимого, увеличенного на единицу.

# Структура данных

Таблица 1 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| MaxSize | Integer | Максимальный допустимый размер числа |
| MaxAmount | Integer | Максимальное допустимое количество чисел для деления |
| NSAlphabet | String | Набор символов для перевода символа в его численное значение и наоборот. |
| Num1 | Array[1..MaxSize] of SmallInt | Массив первого числа (делимого) |
| Num2 | Array[1..(MaxSize \* MaxAmount)] of SmallInt | Массив второго числа (делитель) |
| Num3 | Array[1..MaxSize] of SmallInt | Массив третьего числа (множитель делителя) |
| Results | Array[1..(MaxSize \* MaxAmount)] of SmallInt | Массив для хранения произведения чисел, а потом частного |
| str | String | Строка для записи числа в массив |
| MaxNS | ShortInt | Максимальная допустимая система счисления |
| NS | Integer | Введенная система счисления |
| Amount | Integer | Введенное количество чисел для деления |
| j | ShortInt | Счетчик цикла |
| Res | ShortInt | Разность цифр текущего разряда |
| CarryRes | ShortInt | Перенос для разности в следующий разряд |
| Sum | ShortInt | Сумма всех значений в текущем элементе (для произведения) |
| CarrySum | ShortInt | Перенос для суммы в следующий разряд (для произведения) |
| Len1 | Word | Длина первого числа |
| Len2 | Word | Длина второго числа |
| Len3 | Word | Длина третьего числа |
| Prod | Word | Произведение цифр текущего разряда |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| CarryProd | Word | Перенос для произведения в следующий разряд |
| PosElement | Word | Текущая позиция в элементе в произведении |
| CurrElInQuotient | Word | Текущий элемент в частном |
| ResultDiv | Word | Результат деления выделенной части цифр |
| ToPosEl | Word | До какого элемента будем делить |
| CurrPosEl | Word | Текущая позиция в делимом |
| i | Word | Счетчик цикла |
| k | Word | Счетчик цикла |
| p | Word | Счетчик цикла |
| flag | Boolean | Индикатор проверки правильности ввода |
| FoundLarger | Boolean | Индикатор нахождения большего числа |
| DelZero | Boolean | Индикатор удаления ненужный нулей |

Приложение А

(обязательное)

Исходный код программы

Program WarmUp7;

{

The program should calculate the division of n numbers

in the entered number system

}

{$APPTYPE CONSOLE}

Uses

System.SysUtils;

Const

MaxSize=256;

MaxAmount=50;

NSAlphabet = '0123456789ABCDEFGHIJ';

MaxNS = length(NSAlphabet);

//MaxSize - maximum amount of digits for entered

//numbers

//MaxAmount - maximum amount of numbers

//NSAlphabet - transfer between symbols and numbers

//MaxNS - maximum number system

Var

Num1 :array[1..MaxSize] of SmallInt;

Num2 :array[1..(MaxSize \* MaxAmount)] of SmallInt;

Num3 :array[1..MaxSize] of SmallInt;

Results :array[1..(MaxSize \* MaxAmount)] of SmallInt;

str :string;

NS, Amount : Integer;

j, Res, CarryRes, Sum, CarrySum : ShortInt;

Prod, CarryProd, PosElement : Word;

Len1, Len2, Len3, k, i, p : Word;

ResultDiv, ToPosEl, CurrPosEl, CurrElInQuotient: Word;

flag, FoundLarger, DelZero: boolean;

//Num1 - array of digits of the first number

//(numerator)

//Num2 - array of digits of the second numbers

//(denominator)

//Num3 - array of digits of the third numbers

//(multiplier in the denominator)

//Results - at first product of two numbers in the

//denominator, then the result of dividing two numbers

//str - string variable for writing numbers

//NS - number system

//Amount - amount of numbers

//Res - residual of digits in the selected range

//CarryRes - carry for residual

//Sum - sum of all digits in the certain category (for

//ProdNums)

//CarrySum - сarry 1 (if there is) to the next element

//(for Sum)

//Prod - product of the digits of the first and second

//number

//CarryProd - сarry the digit (if there is) to the next

//element (for Prod)

//PosElement - the current position of the element in

//the multiplication

//Len1 - the length of the first number

//Len2 - the length of the second number

//Len3 - the length of the third number

//CurrElInQuotient - current element in quotient

//i, j, k, p - cycle counter

//ResultDiv - result of dividing a part of digits

//ToPosEl - up to which element will divide

//CurrPosEl - current position in the numerator

//flag - flag to confirm the correctness of entering

//numbers

//FoundLarger - when a larger number is found, the

//variable will become true

//DelZero - variable to remove the unnecessary zeros

//(if there is)

Begin

Writeln('Enter number system (maximum number system is

',MaxNS,', minimal is 2). The program will

calculate their quotient.');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Validating the correct input data type

Try

Readln(NS);

Except

Writeln('Wrong input of number system! It must be

an integer');

flag:= True;

End;

//Validate Range

if ((NS > MaxNS) or (NS < 2)) and (flag = False) then

begin

Writeln('Wrong input of number system! It must be

>=2 and <=',MaxNS);

flag:= True;

end;

Until flag = False;

Writeln;

Writeln('Enter amount of numbers (no more than ',

MaxAmount,' and more than 1)');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Validating the correct input data type

Try

Readln(Amount);

Except

Writeln('Wrong input of amount of numbers! It must

be an integer');

flag:= True;

End;

//Validate Range

if ((Amount > MaxAmount) or (Amount <= 1)) and

(flag = False) then

begin

Writeln('Wrong input of amount of numbers! It must

be >1 and <=',MaxAmount);

flag:= True;

end;

Until flag = False;

//Declaring available symbols and their value

Writeln;

Writeln('Available symbols on the ',NS,'th number

system and their number system:');

for i := 0 to (NS - 1) do

Writeln('Symbol ', NSAlphabet[i+1],' Value = ',i);

Writeln;

Writeln('Attention! Numerator must be >= denominator!

And remember that division by zero is

disallowed');

Writeln('Enter numbers (no more than ',MaxSize,'

digits).');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Read the first entered number and check for

//correctness.

Readln(str);

//Find length of the first number

Len1:= length(str);

//Checking the correct length

if Len1 > MaxSize then

begin

Writeln('Wrong input of number! The length of the

number must be no more than ',MaxSize,'

digits.');

flag:= True;

end

//Else if length >1, the first digit cannot be 0 (in

//the mirrored view it is last)

else if (Len1 > 1) and (str[1] = '0') then

begin

Writeln('Wrong input of number! The first digit of

a number cannot be 0');

flag:= True;

end

//Else writing a number to an array and checking for

//valid symbols

else

begin

//Reset the first number for the input

for i := 1 to length(Num1) do

Num1[i]:= 0;

//Write the first entered number in mirrored view

//to an array

i:= 1;

while (i <= Len1) and (flag = False) do

begin

//Transfer to numerical value (-1 because

//numbering in delphi starts from 1)

Num1[i]:= Pos(str[Len1-i+1], NSAlphabet) - 1;

//Checking for correct input in the number system

//Num1[i] will be <0 if the symbol is not in

//NSAlphabet

if (Num1[i] < 0) or (Num1[i] >= NS) then

begin

Writeln('Wrong input of number! Namely, wrong

input of symbols! See available symbols

above!');

flag:= True;

end;

//Modernize i

i:= i + 1;

end;

end;

Until flag = False;

Writeln('/');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Read the second entered number and check for

//correctness.

Readln(str);

//Find length of the second number

Len2:= length(str);

//Checking the correct length

if Len2 > MaxSize then

begin

Writeln('Wrong input of number! The length of the

number must be no more than ',MaxSize,'

digits.');

flag:= True;

end

//Else if the first digit is 0 (in the mirrored view

//it is last), then the input is invalid.

//Next, determine the type of error

else if str[1] = '0' then

begin

if Len2 > 1 then

Writeln('Wrong input of number! The first digit

of a number cannot be 0')

else

Writeln('Wrong input of number! Division by zero

is disallowed');

flag:= True;

end

//Else writing a number to an array and checking for

//valid symbols

else

begin

//Reset the second number for the input

for i := 1 to length(Num2) do

Num2[i]:= 0;

//Write the second number in mirrored view to an

//array

i:=1;

while (i<=Len2) and (flag = False) do

begin

//Transfer to numerical value (-1 because

//numbering in delphi starts from 1)

Num2[i]:= Pos(str[Len2-i+1], NSAlphabet) - 1;

//Checking for correct input in the number system

//Num2[i] will be <0 if the symbol is not in

//NSAlphabet

if (Num2[i] < 0) or (Num2[i] >= NS) then

begin

Writeln('Wrong input of number! Namely, wrong

input of symbols! See available symbols

above!');

flag:= True;

end;

//Modernize i

i:= i + 1;

end;

end;

Until flag = False;

//The cycle go (Amount - 2) times to multiply all the

//numbers in the denominator

for j := 1 to (Amount - 2) do

begin

Writeln('/');

//Cycle with postcondition for entering correct

//data.

Repeat

//Initialize the flag

flag:= False;

//Read the third number and check for correctness.

Readln(str);

//Find length of the third number

Len3:= length(str);

//Checking the correct length

if len3>MaxSize then

begin

Writeln('Wrong input of number! The length of

the number must be no more than

',MaxSize,' digits.');

flag:= True;

end

//Else if the first digit is 0 (in the mirrored

//view it is last), then the input is invalid.

//Next, determine the type of error

else if str[1] = '0' then

begin

if Len3 > 1 then

Writeln('Wrong input of number! The first

digit of a number cannot be 0')

else

Writeln('Wrong input of number! Division by

zero is disallowed');

flag:= True;

end

//Else writing a number to an array and checking

//for valid symbols

else

begin

//Reset the third number for the input

for i := 1 to length(Num3) do

Num3[i]:= 0;

//Write the third number in mirrored view to an

//array

i:= 1;

while (i <= Len3) and (flag = False) do

begin

//Transfer to numerical value (-1 because

//numbering in delphi starts from 1)

Num3[i]:= Pos(str[Len3-i+1], NSAlphabet) - 1;

//Checking for correct input in the NS.

//Num3[i] will be <0 if the symbol is not in

//NSAlphabet

if (Num3[i] < 0) or (Num3[i] >= NS) then

begin

Writeln('Wrong input of number! Namely,

wrong input of symbols! See available

symbols above!');

flag:= True;

end;

//Modernize i

i:= i + 1;

end;

end;

Until flag = False;

//Multiply the digits of the second number by the

//third number

for i := 1 to Len2 do

begin

for k := 1 to Len3 do

begin

//Сalculate at what position in the

//multiplication the element now

PosElement:= k+i-1;

//Starting to multiply the last digits of the

//numbers (in the mirrored view it is first)

//and add the carry (if there is).

Prod:= Num2[i] \* Num3[k] + CarryProd;

//The integer part of dividing by NS is the carry

//that will go to the next element

CarryProd:= Prod div NS;

//Find the sum of digits in a current position

//element

Sum:= (CarrySum + Results[PosElement] +

(Prod mod NS));

//The integer part of dividing by NS is the carry

//that will go to the next element

CarrySum:= Sum div NS;

//The modulo of the Sum by NS is the digit in

//the Results

Results[PosElement] := Sum mod NS;

//If there is a carry on the last digit of the

//second number, then add a carry to the next

//element

if k = Len3 then

begin

if CarryProd >= 1 then

begin

//c in the next element is equal to the

//carry, since this element is new for

//multiplied

Results[PosElement+1]:= CarryProd;

//Carry is assigned 0 for the next iterations

CarryProd:= 0;

end;

if CarrySum = 1 then

begin

//c in the next element is equal to the sum

//of carry and c in the next element,

//since this element was already for added

Results[PosElement+1]:= CarrySum +

Results[PosElement+1];

//Carry is assigned 0 for the next iterations

CarrySum:=0;

end;

end;

end;

//If in the last digits of the two numbers

//there is a number in the next position,

//then the current position element increases

if (i = Len2) and (Results[PosElement+1] > 0) then

PosElement:= PosElement + 1;

end;

//For the next iteration, the first number becomes

//the product of the previous. And reset the product

//of nums (Results) for the next iteration

for i := 1 to PosElement do

begin

Num2[i]:= Results[i];

Results[i]:= 0;

end;

//Update the length of the first number

Len2:= PosElement;

//Reset the carryes for the operations

CarrySum:= 0;

CarryProd:= 0;

end;

//Check that the number of digits in the numerator must

//be >= digits in the denominator

if Len2>Len1 then

flag:= True

else

begin

//Checking the condition: numerator >= denominator

i:= Len1;

FoundLarger:= False;

while (i >= 1) and (FoundLarger = False) do

begin

//If Num1>Num2, exit the cycle

if Num1[i] > Num2[i] then

FoundLarger:= True

//If Num2>Num1, exit the cycle and raise the flag

else if Num2[i] > Num1[i] then

begin

FoundLarger:= True;

flag:= True;

end;

//Modernize i

i:= i - 1;

end;

end;

//Checking the condition numerator >= denominator

if flag = True then

Writeln('Error! The denominator is greater than the

numerator! Restart the program')

else

begin

//Initialize the variables (considering that they

//are written in mirrored view)

ToPosEl:= Len1 - Len2 + 2;

//(+2 since at the beginning of the cycle will

//decrease by 1)

CurrPosEl:= len1;

CurrElInQuotient:=0;

//(0 since at the beginning of the cycle will add 1)

repeat

//Initialize variables

DelZero:= False;

i:=CurrPosEl;

j:= ToPosEl;

//Remove unnecessary zeros in the numerator

//(if there is)

while (i >= j) and (DelZero = False) do

begin

//When found the number (1..NS-1) exit the cycle

if (Num1[i] > 0) then

DelZero:= True

//Else move the current position of the element

//to the left

else

begin

//Also move ToPosEl (one time less than

//CurrPosEl to maintain the logic of division

//into a column)

if (i <> CurrPosEl) then

ToPosEl:= ToPosEl - 1;

//Reduce CurrPosEl

CurrPosEl:= CurrPosEl - 1;

end;

//Modernize i

i:=i - 1;

end;

//After dividing, take out the next digit

ToPosEl:= ToPosEl - 1;

//Initialize the variables

CurrElInQuotient:= CurrElInQuotient+1;

ResultDiv:=0;

FoundLarger:= False;

i:=CurrPosEl;

j:= Len2;

//Compare two numbers: the highlighted part of the

//numerator and denominator

while (i >= ToPosEl) and (FoundLarger = False) do

begin

//Checking conditions for division (the

//highlighted part of the numerator >=

//denominator)

if ((CurrPosEl-ToPosEl+1) > Len2) or

(((CurrPosEl-ToPosEl+1) = Len2)

and (Num1[i] > Num2[j])) or

((j = 1) and (Num1[i] = Num2[j])) then

begin

//Initialize the variables

CarryRes:= 0;

k:= 1;

p:= ToPosEl;

//Cycle to subtract two numbers: the

//highlighted part of the numerator and

//denominator

while p <= CurrPosEl do

begin

//Considering the signs, starting to

//subtract the last digits of the numbers

//(in the mirrored view it is first)

//and consider the carry (if there is).

Res:= Num1[p] - Num2[k] + CarryRes;

//If the difference is less than zero, then

//take 1 (for the current digit it is NS)

//from the next digit

if Res < 0 then

begin

Res:= Res + NS;

CarryRes:= -1;

end

//Else carry = 0

else

CarryRes:= 0;

//Put the result in the residual array c

Num1[p]:= Res;

//Modernize variables

k:= k + 1;

p:= p + 1;

end;

//Since the highlighted part of the numerator

//was taken away from the denominator once,

//perform +1

ResultDiv:= ResultDiv + 1;

//If a digit has been deleted, correct

//CurrPosEl

if (Num1[CurrPosEl] = 0) then

CurrPosEl:= CurrPosEl - 1;

//Reset i and j to initial value for next check

i:= CurrPosEl + 1;

j:= len2 + 1;

end

//Otherwise check the condition - the

//highlighted part of the numerator <

//denominator. Exiting the cycle to consider

//the next highlighted part of the numerator

else if ((CurrPosEl-ToPosEl+1) < Len2) or

(((CurrPosEl-ToPosEl+1) = Len2)

and (Num1[i] < Num2[j])) then

FoundLarger:= True;

//Modernize variables

i:= i - 1;

j:= j - 1;

end;

//Write the result of the current element in

//quotient

Results[CurrElInQuotient]:= ResultDiv;

//Check for output: numerator remainder must be

//< denominator

until (CurrPosEl < Len2) or ((CurrPosEl = Len2)

and (Num1[i+1] < Num2[j+1]));

Writeln('The prod of the numbers is:');

//If the first highlighted part of the numerator was

//< denominator, then zero is not taken out – correct

//it (if there is)

if Results[1] = 0 then

i:= 2

else

i:= 1;

//Сount for final element in quotient (сonsidering

//division method)

CurrElInQuotient:= CurrElInQuotient + ToPosEl - 1;

//Write the answer, mirroring the back

//And transfer to symbolic value (+1 because

//numbering in delphi starts from 1)

while i <= CurrElInQuotient do

begin

Write(NSAlphabet[Results[i]+1]);

i:= i + 1;

end;

Writeln;

Writeln('Remainder after division is:');

//Write the remainder after division, deleting

//unnecessary zeros. CurrPosEl + 1 because

//in a cycle there can be CurrPosEl = 0

DelZero:= False;

for i := (CurrPosEl + 1) downto 1 do

begin

if (DelZero = False) and (Num1[i] > 0) then

DelZero:= True;

if (DelZero = True) or (i = 1) then

Write(NSAlphabet[Num1[i]+1]);

end;

end;

Readln;

End.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

Тест 1

Исходные данные: Некорректный ввод системый счисления

Ожидаемый результат: Повторная попытка

Полученный результат:

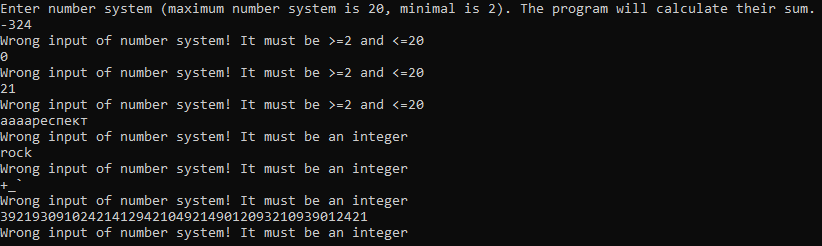


Рисунок 1 – Результаты расчетов

Тест 2

Исходные данные: Некорректный ввод количества чисел для деления

Ожидаемый результат: Повторная попытка

Полученный результат:

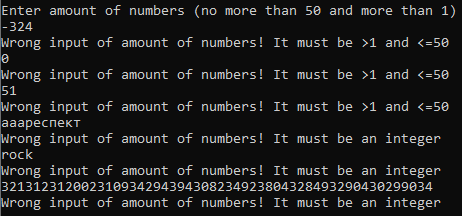


Рисунок 2 – Результаты расчетов

Тест 3

Исходные данные: Некорректный ввод первого числа

Ожидаемый результат: Повторная попытка

Полученный результат:

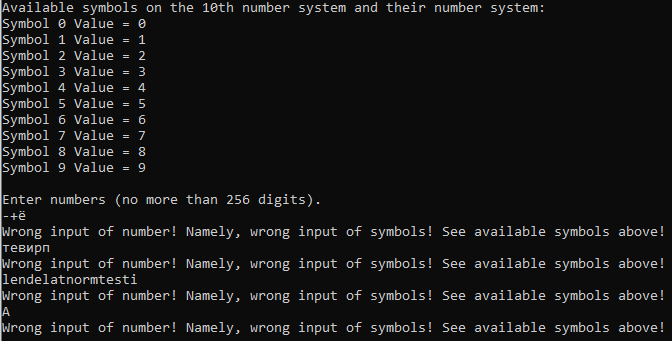


Рисунок 3 – Результаты расчетов

Тест 4

Исходные данные: Некорректный ввод второго числа

Ожидаемый результат: Повторная попытка

Полученный результат:

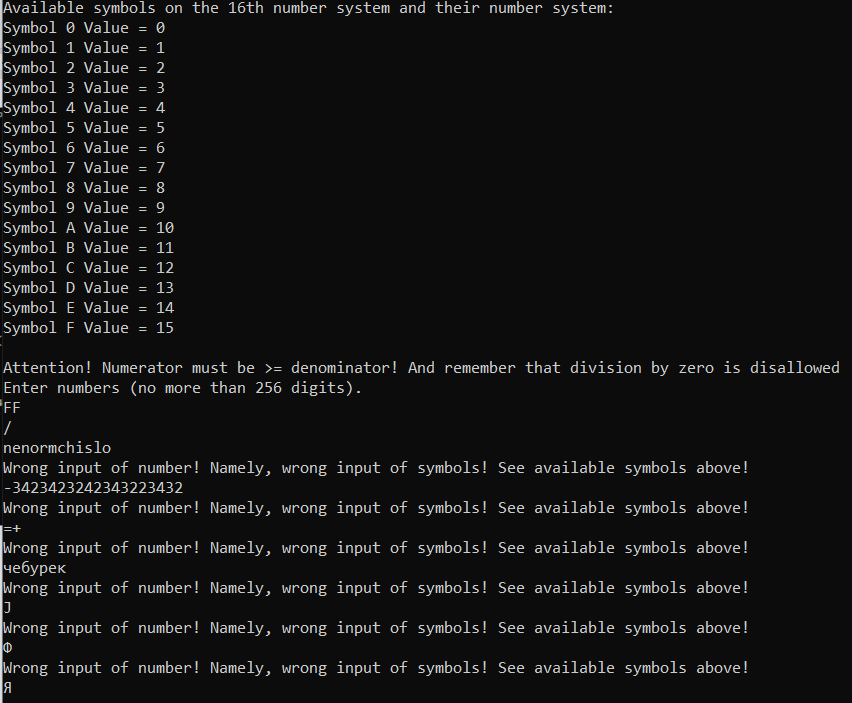


Рисунок 4 – Результаты расчетов

Тест 5

Исходные данные: Некорректный ввод второго числа

Ожидаемый результат: Повторная попытка

Полученный результат:

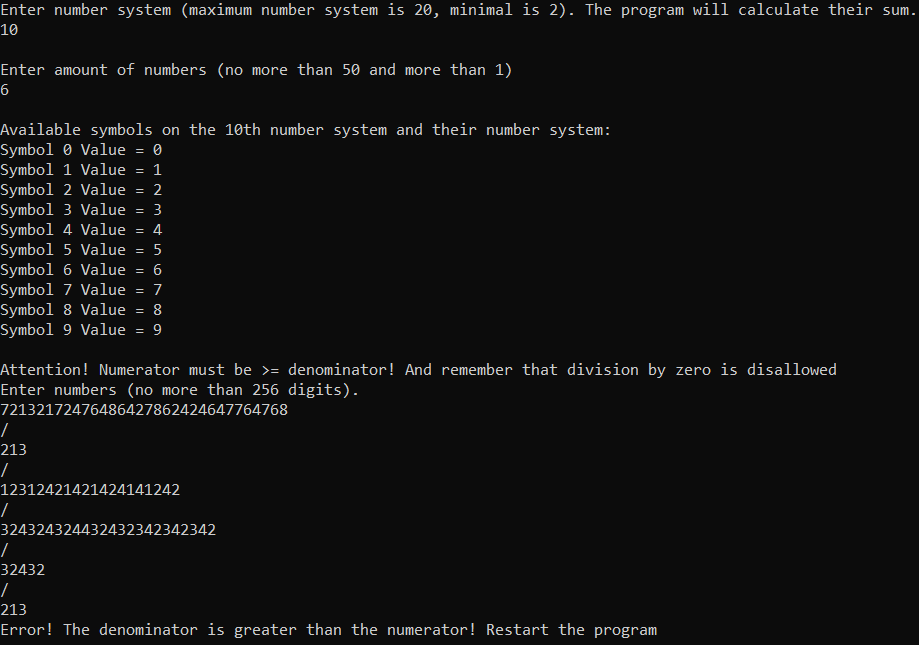


Рисунок 5 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Тест 6

Исходные данные: Система счисления: 16; количество чисел для деления: 2.

Ожидаемый результат:

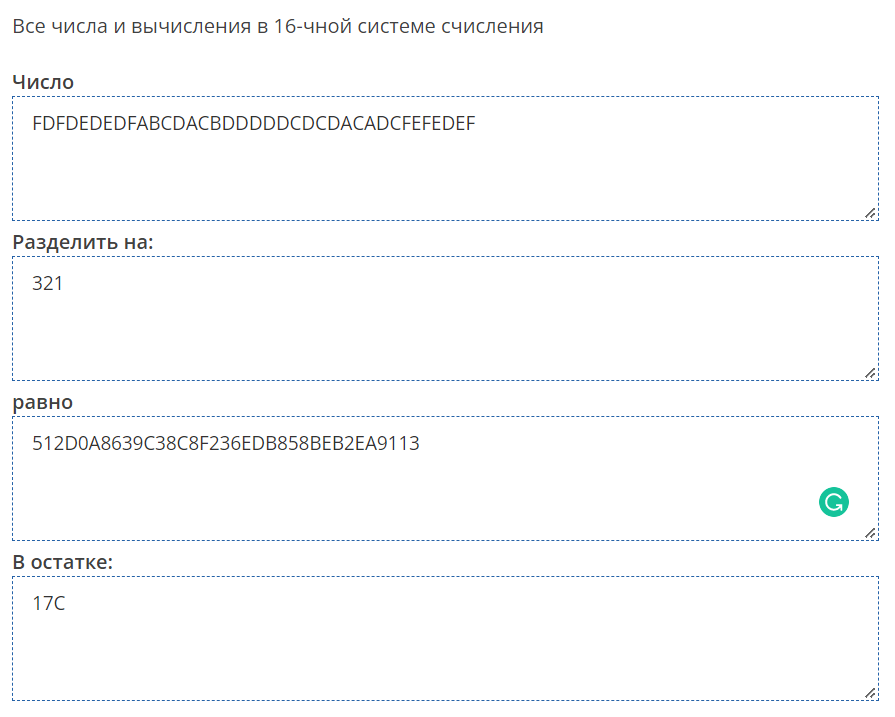


Рисунок 6 – Ожидаемый результат (ilovecalc)

Полученный результат:

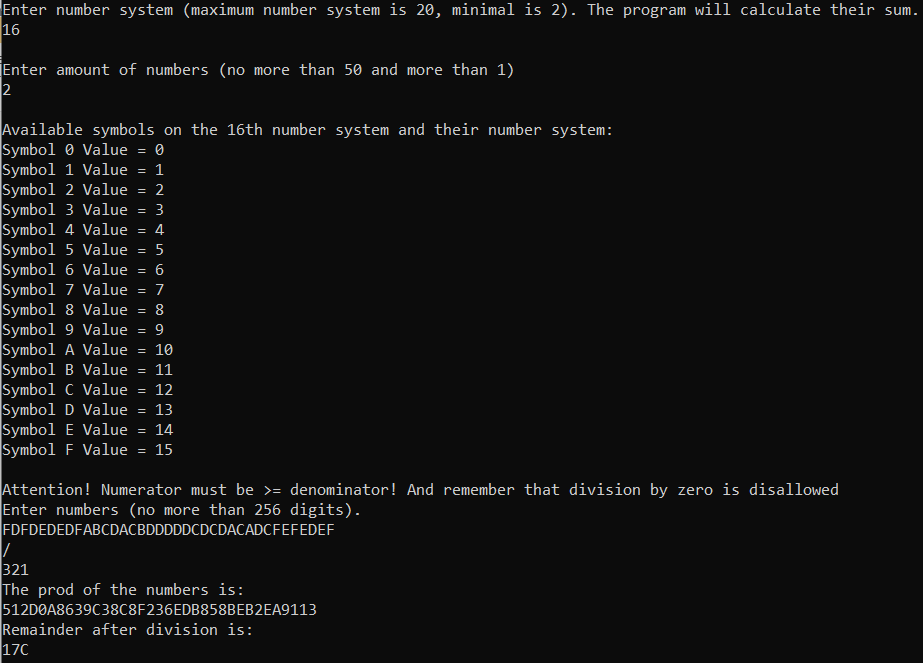


Рисунок 7 – Результаты расчетов

Тест 7

Исходные данные: Система счисления: 2; количество чисел для деления: 2.

Ожидаемый результат:

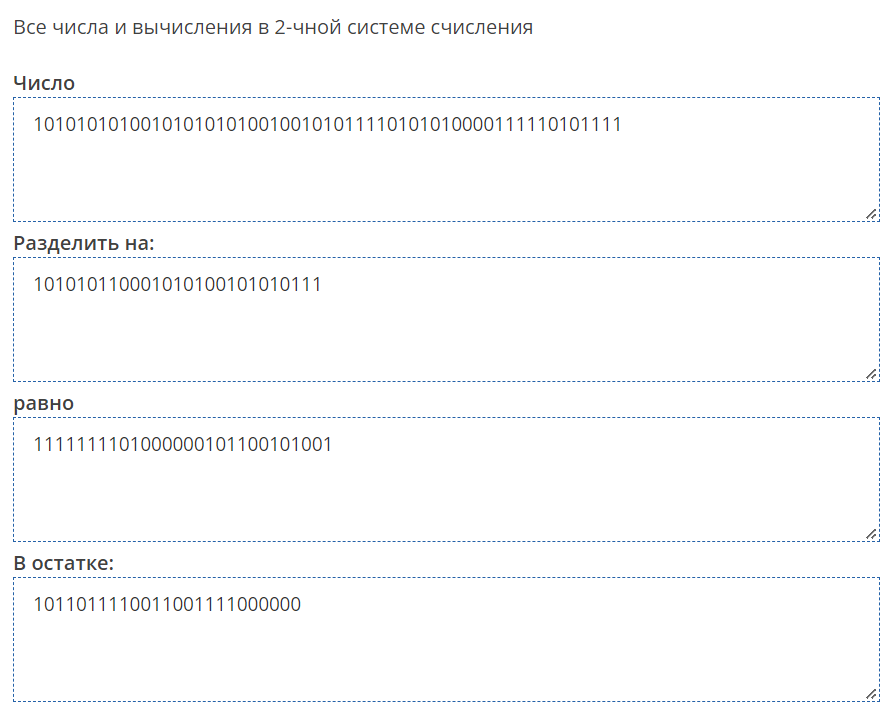


Рисунок 8 – Ожидаемый результат (ilovecalc)

Полученный результат:

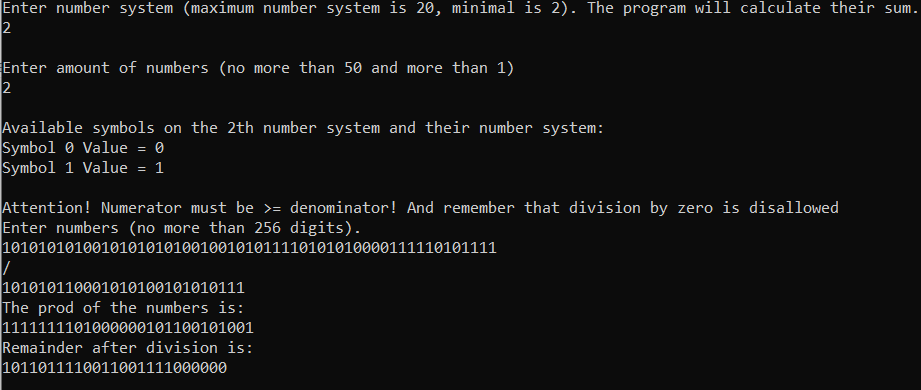


Рисунок 9 – Результаты расчетов

Тест 8

Исходные данные: Система счисления: 20; количество чисел для деления: 2.

Ожидаемый результат:

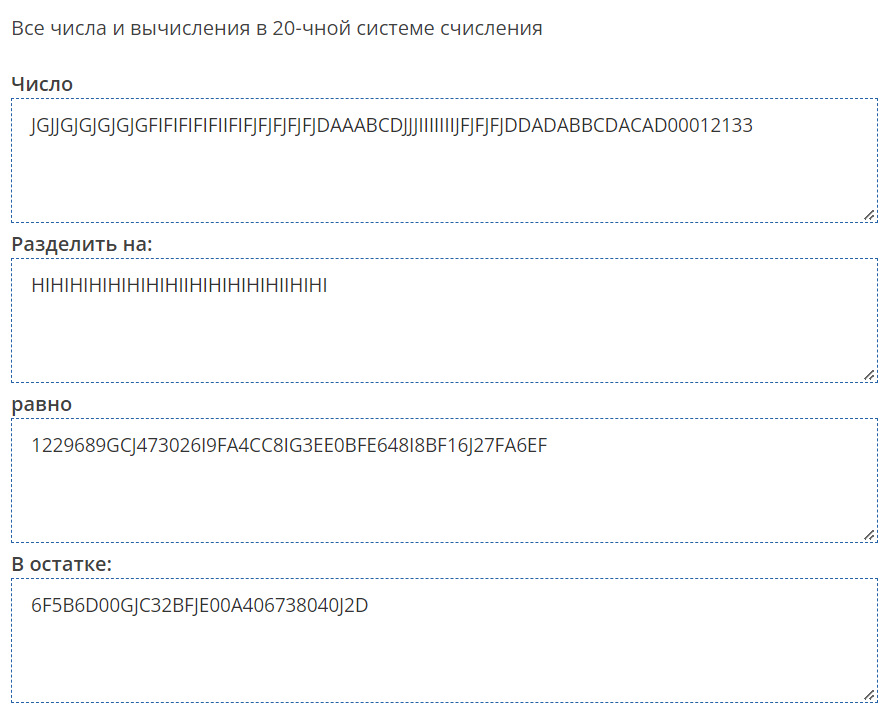


Рисунок 10 – Ожидаемый результат (ilovecalc)

Полученный результат:

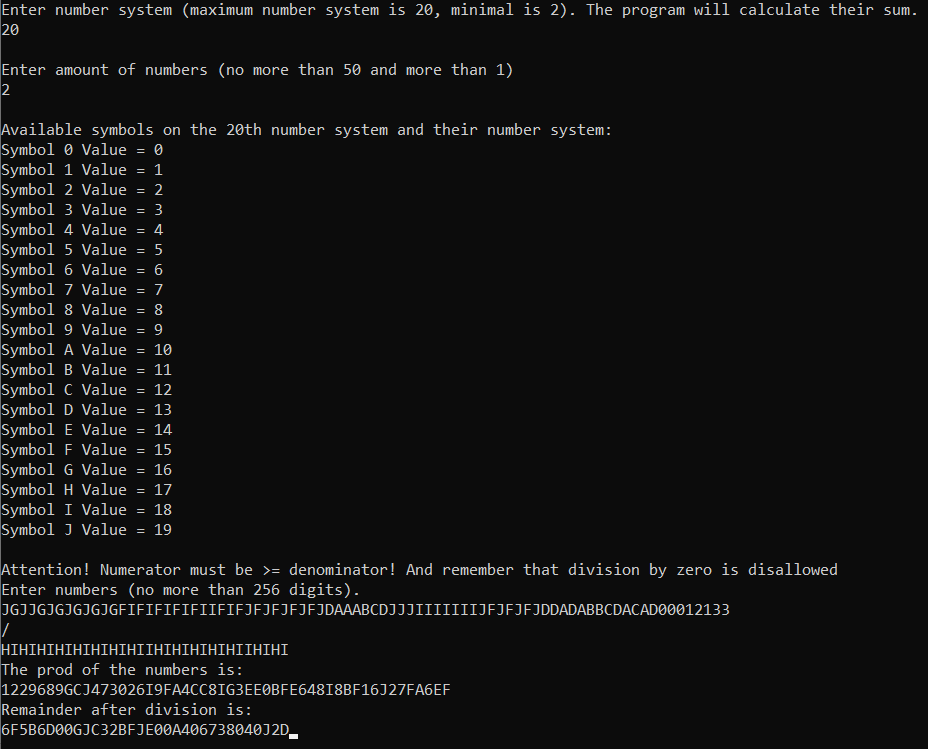


Рисунок 11 – Результаты расчетов

Тест 9

Исходные данные: Система счисления: 10; количество чисел для деления: 2.

Ожидаемый результат:

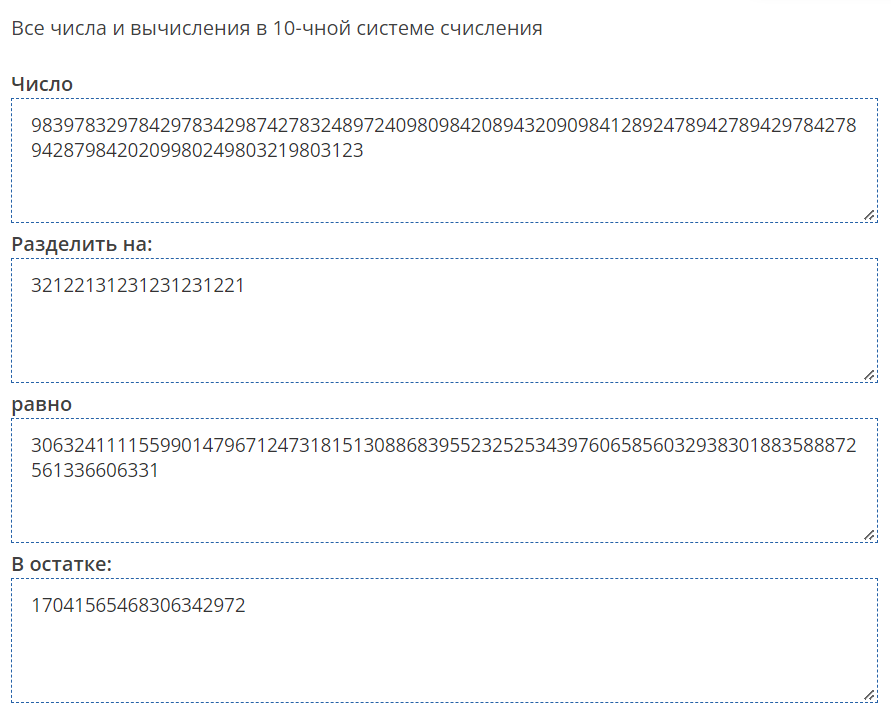


Рисунок 12 – Ожидаемый результат (ilovecalc)

Полученный результат:

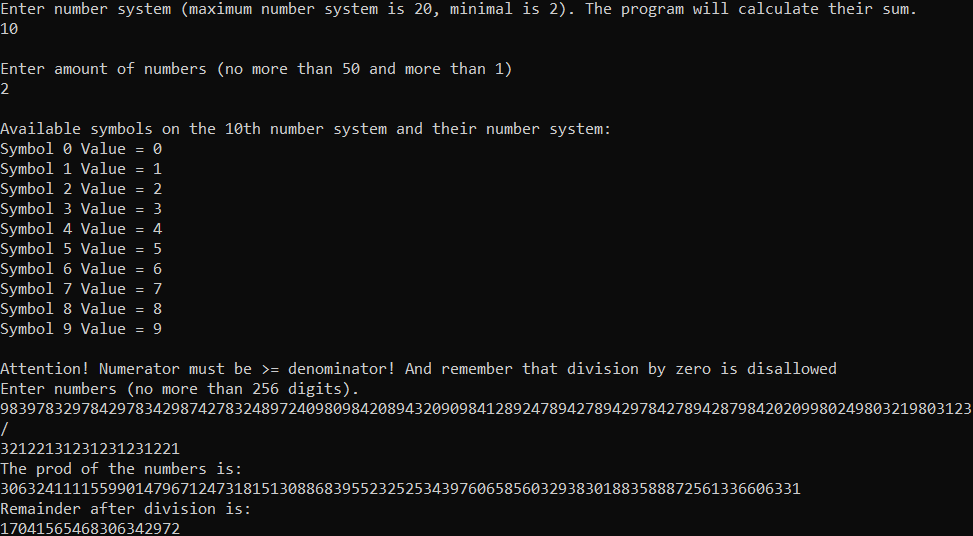


Рисунок 13 – Результаты расчетов

Тест 10

Исходные данные: Система счисления: 10; количество чисел для деления: 8.

Ожидаемый результат:

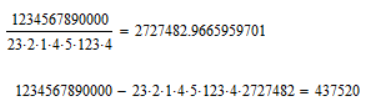


Рисунок 14 – Ожидаемый результат (mathcad)

Полученный результат:

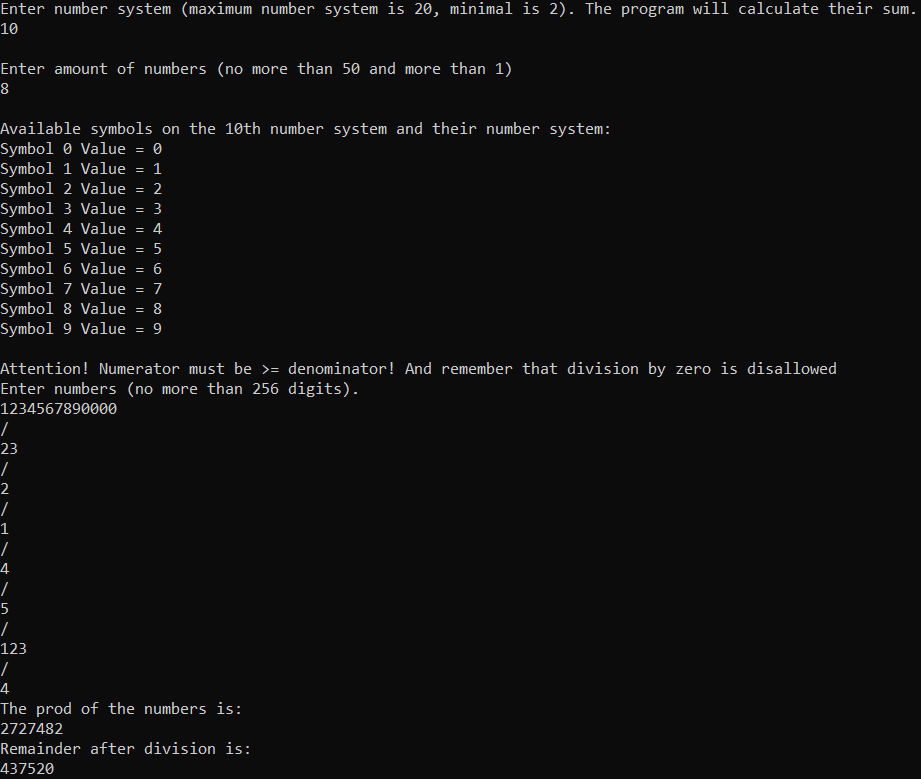


Рисунок 15 – Результаты расчетов