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 №6

Тема работы: Длинная арифметика: умножение

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Исходная постановка задачи 3](#_Toc120385740)

[2 Дополненная постановка задачи 4](#_Toc120385741)

[3 Методика решения 5](#_Toc120385742)

[3.1 Описание оператора try..except..end и функции Pos 5](#_Toc120385743)

[3.1.1 Оператор try..except..end 5](#_Toc120385744)

[3.1.2 Функция Pos и использование константной строки 5](#_Toc120385745)

[3.2 Условия ввода 5](#_Toc120385746)

[3.3 Проверка введенных данных и запись числа в массив 6](#_Toc120385747)

[3.3.1 Проверка ввода системы счисления и количества чисел для произведения 6](#_Toc120385748)

[3.3.2 Проверка корректности введенного числа и его запись в массив 6](#_Toc120385749)

[3.4 Произведение чисел 6](#_Toc120385750)

[3.5 Вывод произведения всех чисел 8](#_Toc120385751)

[4 Структура данных 9](#_Toc120385752)

[Приложение А 10](#_Toc120385753)

[Приложение Б 19](#_Toc120385754)

# Исходная постановка задачи

Даны два целых числа в десятичной системе счисления до 50 цифр, которые >=0. Необходимо найти их произведение.

# Дополненная постановка задачи

Вводится необходимая система счисления NS (максимальная система счисления – 20-ая) и количество чисел для умножения n (не больше 50). Числа должны быть целыми, не более 256 символов и >=0. Необходимо найти их произведение.

# [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 произведения 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.

## Проверка введенных данных и запись числа в массив

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

Проверка ввода системы счисления NS и количества чисел для произведения 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.

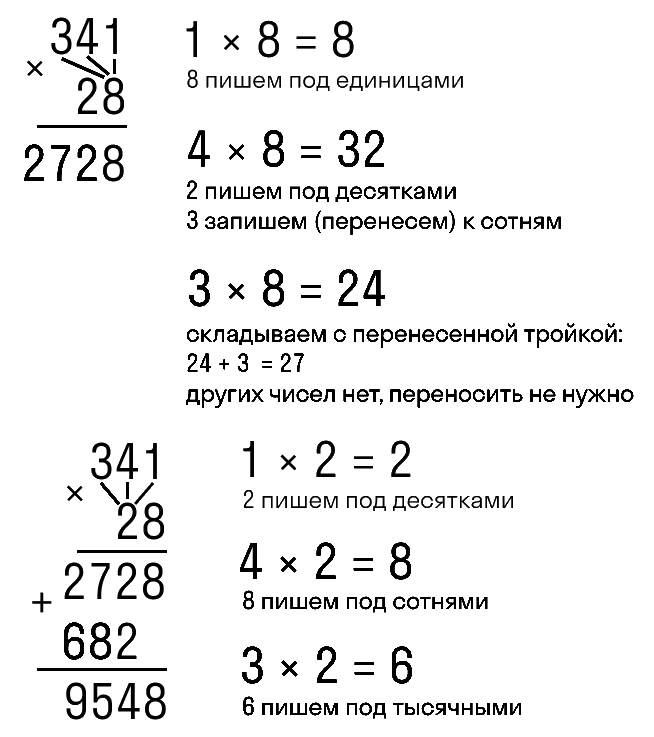
Так в данной задаче необходимо работать с очень большими числами (которые не входят ни в один тип данных), необходимо записывать каждую цифру числа в отдельный элемент массива. Так как в результате умножения количество цифр произведения может быть больше, чем количества цифр изначальных значений, то 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) с помощью оператора if. Если соответствует, то численное значение символа записываем в элемент массива.

## Произведение чисел

Произведение чисел происходит поэтапно: сначала умножаются два числа, потом произведение этих двух чисел умножается с третьим числом, потом произведение предыдущих чисел со следующим числом и так далее.

Произведение двух чисел происходит с помощью метода умножения столбиком. Так как числа записаны с конца, умножать элементы начнем сначала. Умножаем элементы поэтапно: сначало все элементы второго числа с первым элементов первого числа, потом вторым элементом и так далее. Рассмотрим пример:



Для нахождения произведения будем делать следующе действия:

1. Умножим текущие цифры и сложим с переносом для произведения – получим произведение цифр Prod.
2. С помощью операции Prod div NS (где Prod – произведение цифр, NS – выбранная система счисления) найдем перенос для произведения в следующий элемент.
3. Находим сумму всех значений в текущем элементе: текущий элемент в массиве произведения складываем с переносом для суммы и текущим произведением цифр (найденная с помощью операции Prod mod NS (где Prod – произведение цифр, NS – выбранная система счисления)).
4. С помощью операции Sum div NS (где Sum – сумма всех значений в текущем элементе, NS – выбранная система счисления) найдем перенос для суммы в следующий элемент.
5. В массиве произведения в текущий элемент записываем найденное значение с помощью операции Sum mod NS (где Sum – сумма всех значений в текущем элементе, NS – выбранная система счисления).

## Вывод произведения всех чисел

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.

Произведение записана с конца, значит вывод элементов массива следует начинать с конца. Для вывода произведения используем цикл с параметром for. Выводить символы будем из константной строки NSAlphabet по индексу текущего значения элемента из массива произведения, 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 |
| NSAlphabet | String | Набор символов для перевода символа в его численное значение и наоборот. |
| Num1 | Array [1..(MaxSize \* MaxAmount)] of ShortInt | Массив первого числа |
| Num2 | Array[1..MaxSize] of ShortInt | Массив второго числа |
| ProdNums | Array[1..(MaxSize \* MaxAmount)] of SmallInt | Массив произведения двух чисел |
| 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 произведения |
| j | ShortInt | Счетчик цикла |
| Sum | ShortInt | Сумма всех значений в текущем элементе |
| CarrySum | ShortInt | Перенос для суммы в следующий разряд |
| Len1 | Word | Длина первого числа |
| Len2 | Word | Длина второго числа |
| Prod | Word | Произведение цифр текущего разряда |
| CarryProd | Word | Перенос для произведения в следующий разряд |
| PosElement | Word | Текущая позиция в элементе в произведении |
| i | Word | Счетчик цикла |
| k | Word | Счетчик цикла |
| flag | Boolean | Индикатор проверки правильности ввода |

Приложение А

(обязательное)

Исходный код программы

Program WarmUp6;

{

The program should calculate the product 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 \* MaxAmount)] of SmallInt;

Num2 :array[1..MaxSize] of SmallInt;

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

str :string;

NS, Amount : Integer;

j, Sum, CarrySum : ShortInt;

Len1, Len2, i, k, Prod, CarryProd, PosElement : Word;

flag: boolean;

//Num1 - array of digits of the first number

//(further the product of the two previous numbers)

//Num2 - array of digits of the second numbers (with

//which need to multiply the first number)

//ProdNums - array of digits of product of numbers (in

//the process of multiplication)

//str - string variable for writing numbers

//NS - number system

//Amount - amount of numbers

//Sum - the sum of all values in the current element

//(for ProdNums)

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

//(for Sum)

//Len1 - the length of the first number

//Len2 - the length of the second number

//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

//i, j, k - cycle counter

//flag - flag to confirm the correctness of entering

//numbers

Begin

Writeln('Enter number system (maximum number system is

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

calculate their product.');

//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('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;

//The cycle go (Amount-1) times to multiply of all the

//numbers

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

begin

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 length >1, the first digit cannot be 0

//(in the mirrored view it is last)

else if (Len2 > 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 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;

//Multiply the digits of the first number by the

//second number

for i := 1 to Len1 do

begin

for k := 1 to Len2 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:= Num1[i] \* Num2[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 + ProdNums[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 ProdNums

ProdNums[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 = Len2 then

begin

if CarryProd >= 1 then

begin

//c in the next element is equal to the

//carry, since this element is new for

//multiplied

ProdNums[PosElement+1]:=CarryProd;

//Carry is assigned 0 for the next iterations

CarryProd:=0;

end;

if CarrySum = 1 then

begin

//ProdNums in the next element is equal to

//the sum of carry and ProdNums in the next

//element.

ProdNums[PosElement+1]:= CarrySum +

ProdNums[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 = Len1) and (ProdNums[PosElement+1] > 0) then

PosElement:= PosElement + 1;

end;

//For the next iteration, the first number becomes

//the product of the previous.

//And reset the produs of nums for the next iteration

for i := 1 to PosElement do

begin

Num1[i]:= ProdNums[i];

ProdNums[i]:= 0;

end;

//Update the length of the first number

Len1:= PosElement;

//Reset the carryes for the operations

CarrySum:= 0;

CarryProd:= 0;

end;

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

//If the product is 0, then output one 0

if Num1[PosElement] = 0 then

Writeln(Num1[PosElement])

//Else write the answer, mirroring the back

//And transfer to symbolic value (+1 because numbering

//in delphi starts from 1)

else

for i := PosElement downto 1 do

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

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:

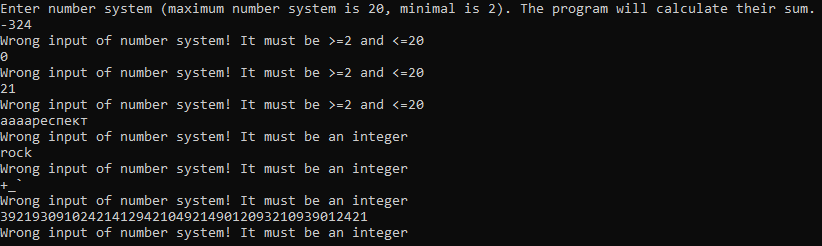


Рисунок 1 – Результаты расчетов

Тест 2

Исходные данные: Некорректный ввод количества чисел для произведения

Ожидаемый результат: Повторная попытка

Полученный результат:

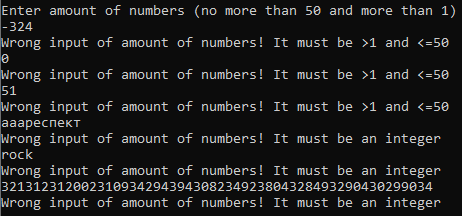


Рисунок 2 – Результаты расчетов

Тест 3

Исходные данные: Некорректный ввод первого числа

Ожидаемый результат: Повторная попытка

Полученный результат:

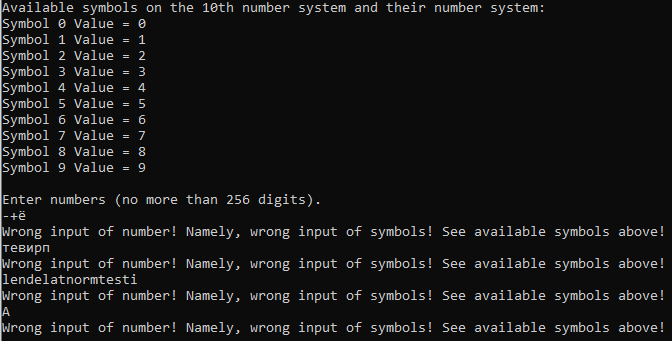


Рисунок 3 – Результаты расчетов

Тест 4

Исходные данные: Некорректный ввод второго числа

Ожидаемый результат: Повторная попытка

Полученный результат:

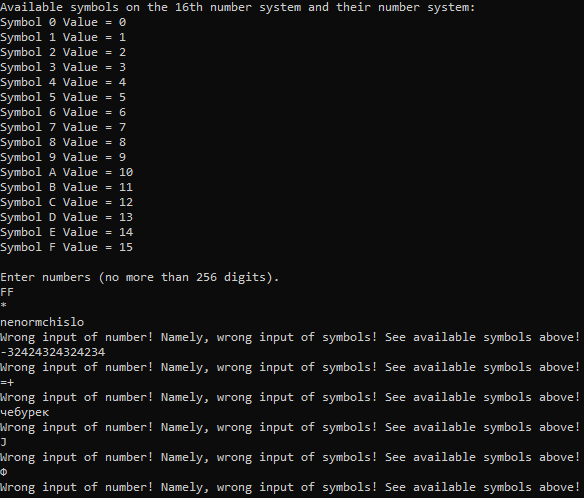


Рисунок 4 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Тест 5

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

Ожидаемый результат:



Рисунок 5 – Ожидаемый результат (ilovecalc)

Полученный результат:

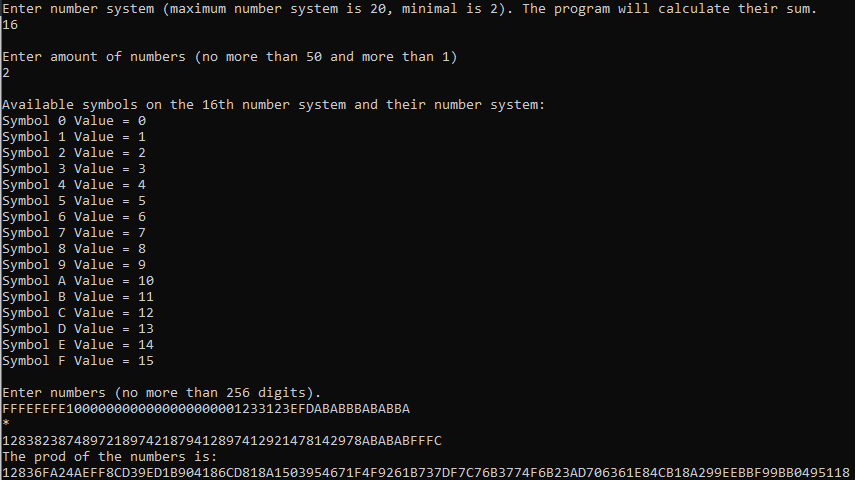


Рисунок 6 – Результаты расчетов

Тест 6

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

Ожидаемый результат:

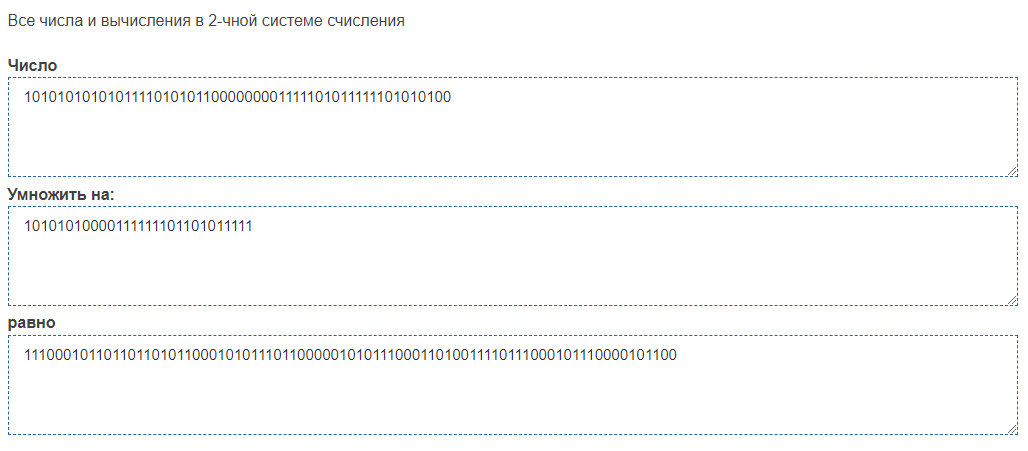


Рисунок 7 – Ожидаемый результат (ilovecalc)

Полученный результат:

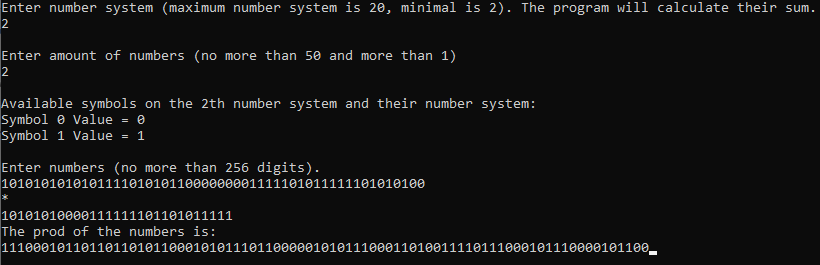


Рисунок 8 – Результаты расчетов

Тест 7

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

Ожидаемый результат:



Рисунок 9 – Ожидаемый результат (ilovecalc)

Полученный результат:

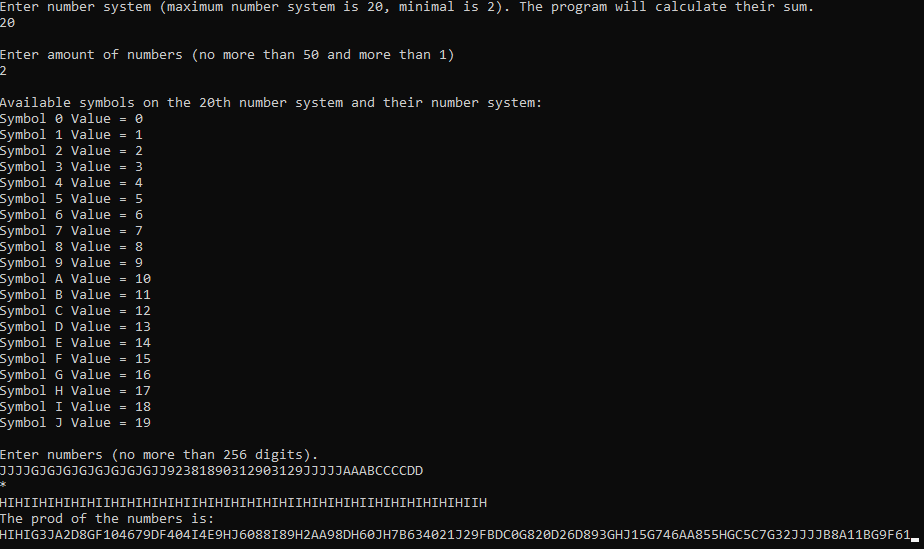


Рисунок 10 – Результаты расчетов

Тест 8

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

Ожидаемый результат:

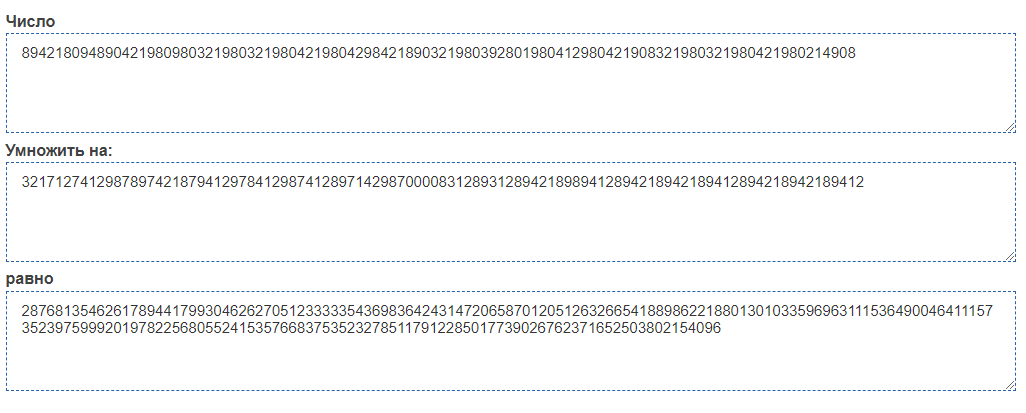


Рисунок 11 – Ожидаемый результат (ilovecalc)

Полученный результат:

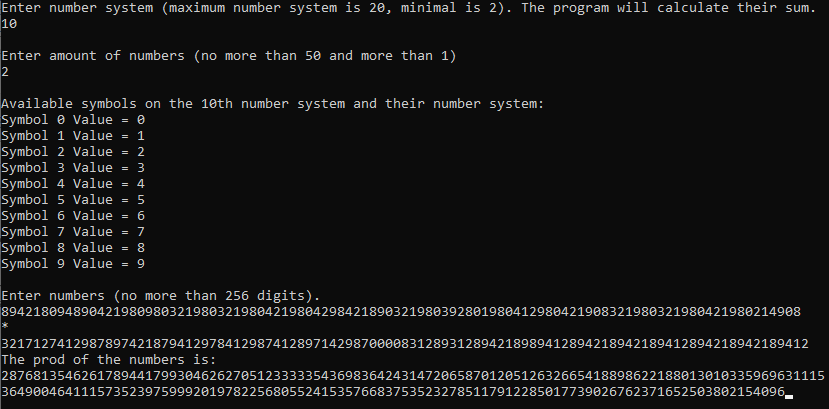


Рисунок 12 – Результаты расчетов

Тест 9

Исходные данные: Система счисления: 10; количество чисел для произведения: 6.

Ожидаемый результат:

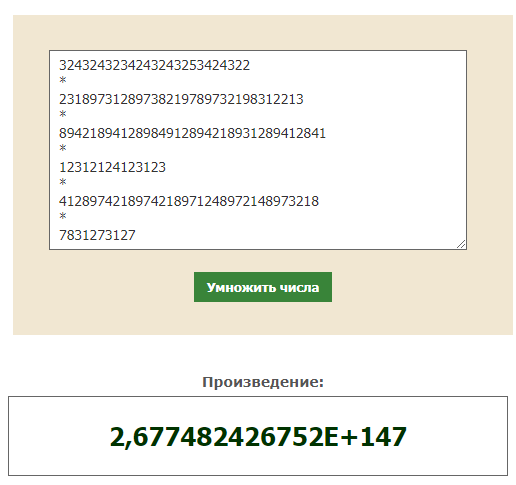


Рисунок 13 – Ожидаемый результат (tools.mega8)

Полученный результат:

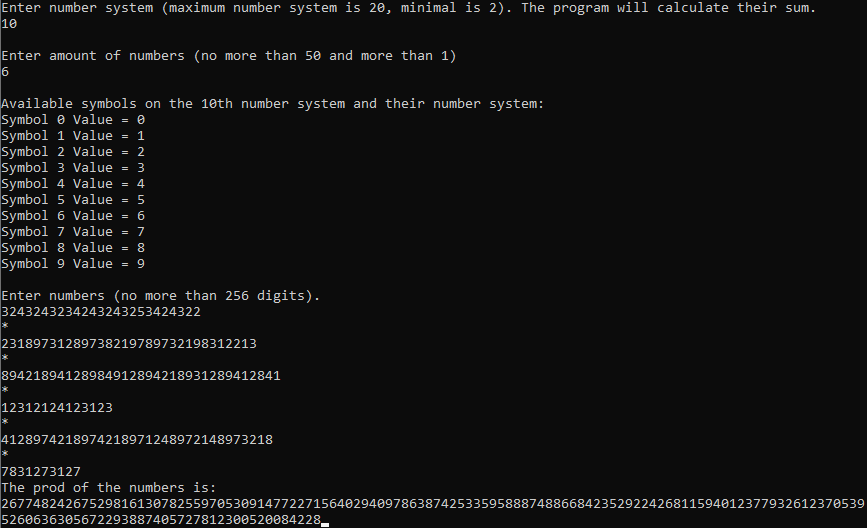


Рисунок 14 – Результаты расчетов