Содержание

[1 Постановка задачи 3](#_Toc121151303)

[1.1 Исходное условие задачи 3](#_Toc121151304)

[1.2 Модифицированное условие задачи 3](#_Toc121151305)

[1.3 Исходные данные 3](#_Toc121151306)

[2 Структура данных 4](#_Toc121151307)

[3 Детали и методика решения задачи 5](#_Toc121151308)

[3.1 Детали задачи 5](#_Toc121151309)

[3.2 Методика решения задачи 5](#_Toc121151310)

[3.3 Проверка исходных данных с помощью цикла с постусловием 6](#_Toc121151311)

[4 Результаты расчетов 8](#_Toc121151312)

[Приложение А 9](#_Toc121151313)

[Приложение Б 14](#_Toc121151314)

# Постановка задачи

## Исходное условие задачи

Есть два целых натуральных числа, состоящих до 50 цифр. Вывести на экран их частное.

## Модифицированное условие задачи

Есть два целых числа (Num\_1 и Num\_2), состоящих до 255 цифр. Вывести на экран частное чисел Rez и остаток, если таковой существует.

## Исходные данные

Программа получает от пользователя :

* делимое;
* делитель.

Вывод данных происходит в следующем виде:

* частное чисел z1, Num\_1 и z2, Num\_2 равно Rez;
* частное чисел z1, Num\_1 и z2, Num\_2 (неполное) равно Rez, остаток ArrayOfDiv;
* z1, Num\_1 / z2, Num\_2 = Rez.

# Структура данных

Таблица 1 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Num\_1 | ShortString | Делимое |
| Num\_2 | ShortString | Делитель |
| Io | ShortInt | Вспомогательная переменная для подсчетов |
| I | ShortInt | Вспомогательная переменная для подсчетов |
| J | SmallInt | Вспомогательная переменная для подсчетов |
| К | ShortInt | Вспомогательная переменная для переноса разряда |
| Ko | ShortInt | Вспомогательная переменная для переноса разряда |
| Count | ShortInt | n-я цифра частного |
| Z1 | ShortString | Знак первого числа |
| Z2 | ShortString | Знак второго числа |
| Rez | ShortString | Частное |
| LengthA | Integer | Длина ArrayOfDiv |
| ArrayOfDiv | Array[1..256] of ShortInt | Массив вычитания Num\_1 и Num\_2 |
| Stop | Boolean | Переменная для проверки на ошибки |

# Детали и методика решения задачи

## Детали задачи

Детали задачи таковы:

* числа должны быть введены без знака, если они положительные;
* числа должны быть целыми;
* числа могут быть отрицательными;
* числа должны быть введены со знаком, если они отрицательные;
* количество цифр в числе должно быть не больше 255;
* второе число может быть больше, чем первое.

## Методика решения задачи

Проанализировав условие и детали задачи, можно сделать вывод, что для решения необходимы следующие пункты:

* ввод числа должен быть в тип Shortstring, т. к. тип Integer может содержать числа в промежутке от -231 до 231 - 1 (7 символов), a Shortstring – 256 символов;
* использование массива типа smallint для подсчётов;
* наличие переменных z1 и z2, которые будут отвечать за знаки (если равна 1 – число положительное; -1 – отрицательное) при выводе начальных значений чисел и подсчётах.

Сами расчёты будут проходить по следующим пунктам:

1. Проверка чисел на их длину, чтобы выявить, стоит ли Num\_1 добавлять разряды и частное начинать с «0.», и определить, с какого разряда Num\_1 начинать вычитание Num\_2.
2. Если разряд первого числа меньше, чем Num\_2, то получение разряда от следующих.
   1. Если следующий – 0, то происходит поиск ненулевого разряда, получая также разряды.
   2. Если нет ненулевого разряда, то процесс вычитания на этом диапазоне разрядов Num\_1 прекращается.
3. Вычитание происходит поразрядно до тех пор, пока разряды Num\_2 не закончатся или не выполнится условие 2.b.
4. Увеличение диапазона вычитания:
   1. На 1 разряд, если длина Num\_1 больше либо равна длине Num\_2.
   2. На длину Num\_2, если его длина больше.
5. Добавление в строку Rez n-й разряд Count частного.
6. Повторение шагов 2-5 до тех пор, пока диапазон вычитания не будет больше, чем:
   1. Длина Num\_1, если длина Num\_1 больше либо равна длине Num\_2.
   2. Длина Rez не будет больше длины Num\_2 на 5 символов.

## Проверка исходных данных с помощью цикла с постусловием и конструкции «try…except…end;»

При проверке исходных данных использовался цикл с постусловием и конструкция «try…except…end;», так как в таком случае при переводе каждого символа строки в тип Integer при возникновении ошибки EConvertError сработает конструкция, а также цикл запуститься хотя бы один раз:

Repeat

Write('Введите первое число: ');

readln(Num\_1);

Write('Введите второе число: ');

readln(Num\_2);

Try

Begin

for I := 1 to length(Num\_1) do

ArrayOfDiv[I] := StrToInt(Num\_1[I]);

for I := 1 to length(Num\_2) do

StrToInt(Num\_2[I]);

Stop := true;

End;

except

on e: EConvertError do

begin

Writeln('Вы ввели не число! Повторите попытку!');

Stop := False;

end;

End;

Until Stop;

С помощью этого цикла мы повторяем вывод фразы «Введите первое число: », получение данных от пользователя, если было правильно введено значение, иначе фразу «Вы ввели не число! Повторите попытку!» до тех пор, пока полученные значения не будет выполнять всем требованиям. Так же происходит и с вводом второго числа.

# Результаты расчетов

Вследствие выполнения программы на экран выводятся следующие результаты расчетов:



Рисунок 1– Результаты расчётов

Приложение А

(обязательное)

Исходный код программы

program Деление\_десятичная;

{

Dividing two numbers (around 255 digit in each)

}

uses

System.SysUtils;

var

Num\_1, Num\_2, Rez, z1, z2: shortstring;

K, Ko, I, io, Count: shortint;

j: smallint;

Stop: boolean;

ArrayOfDiv: array [1 .. 256] of shortint;

// Num\_1, Num\_2 - numbers, which must be divided

// J, I, Io, Count, K, Ko - auxiliary variables for counting

// Rez - result of dividing

// Z1, z2 - variables of signs of numbers

// ArrayOfDiv - array to divide Num\_1 and Num\_2

// Stop - variable to check for errors

const

LengthA = length(ArrayOfDiv);

// LengthA - Length ArrayOfDiv

begin

// Requiring the numbers and checking them for wrong

// signs and minus

Repeat

Write('Введите первое число: ');

readln(Num\_1);

if Num\_1[1] = '-' then

Begin

Delete(Num\_1, 1, 1);

z1 := '-';

End;

Write('Введите второе число: ');

readln(Num\_2);

if Num\_2[1] = '-' then

Begin

Delete(Num\_2, 1, 1);

z2 := '-';

End;

if ((z1='-') and (z2='')) or ((z2='-') and (z1='')) then

Rez := '-';

Try

Begin

// Initialization from first digit to the last

// to array element from the start of it

for I := 1 to length(Num\_1) do

ArrayOfDiv[I] := StrToInt(Num\_1[I]);

// Checking for the wrong signs

for I := 1 to length(Num\_2) do

StrToInt(Num\_2[I]);

Stop := true;

End;

except

on e: EConvertError do

begin

Writeln('Вы ввели не число! Повторите попытку!');

Stop := False;

end;

End;

Until Stop;

// Initializing the 'boarders' of substracting

j := length(Num\_2);

io := j;

// Increasing them, if it needs

if (ArrayOfDiv[1]<StrToInt(Num\_2[1])) or (ArrayOfDiv[j] <

StrToInt(Num\_2[j])) then

io := io + 1;

Stop := False;

// Looking for the needing way of calculating

if (length(Num\_1) > length(Num\_2)) or

((length(Num\_1)=length(Num\_2)) and (Num\_1[1]>=Num\_2[1]))

then

Begin

I := io;

Repeat

Stop := False;

// Subtraction

repeat

repeat

// If we need to get a rank from the next one

If (ArrayOfDiv[I] < StrToInt(Num\_2[j])) and (I > 1)

then

// if the next on is zero and we again

// need a rank

if ArrayOfDiv[I - 1] = 0 then

Begin

Ko := 1;

While (ArrayOfDiv[I-Ko]=0) and ((I-Ko)>0) do

Ko := Ko + 1;

if I - Ko > 0 then

// Trying to find the rank, which isn't zero

while K < Ko do

Begin

K := K + 1;

ArrayOfDiv[I - K] := ArrayOfDiv[I - K] - 1;

ArrayOfDiv[I+1-K] := ArrayOfDiv[I+1-K]+10;

End;

End

Else

Begin

// Getting a rank from the next one and

// substracting it

ArrayOfDiv[I - 1] := ArrayOfDiv[I - 1] - 1;

ArrayOfDiv[I] := ArrayOfDiv[I] + 10;

End;

// If isn’t possible to substracting one time more

if (ArrayOfDiv[I] < StrToInt(Num\_2[j])) or

((ArrayOfDiv[I - j + 1] < StrToInt(Num\_2[1])) and

(ArrayOfDiv[I - j] = 0)) then

Stop := true

Else

Begin

ArrayOfDiv[I] := ArrayOfDiv[I] -

StrToInt(Num\_2[j]);

// if the whole divider was subtracted

if j = 1 then

Count := Count + 1;

End;

// Going to new rank

j := j - 1;

I := I - 1;

K := 0;

until (j = 0) or Stop;

// starting again

j := length(Num\_2);

I := io;

Until Stop;

if not((Rez = '') and (Count = 0)) then

begin

insert(IntToStr(Count), Rez, length(Rez) + 1);

Count := 0;

end;

// Updating 'borders'

j := length(Num\_2);

io := io + 1;

I := io;

Until I > length(Num\_1);

Write('Частное чисел ', z1, Num\_1, ' и ', z2, Num\_2);

if Rez = '' then

Writeln(' не существует')

else

Begin

// Finding a remainder

j := 1;

While (ArrayOfDiv[j] = 0) and (j < LengthA) do

j := j + 1;

if (j <= LengthA) and (ArrayOfDiv[j] = 0) then

Writeln(' равно ', Rez)

Else

Begin

Writeln(' (неполное) равно ', Rez);

write('Остаток равен ');

// Outputting the sum

for I := j to length(Num\_1) do

Write(ArrayOfDiv[I]);

End;

End;

End

Else

Begin

// the pattern of dividing a smaller

// figure by a smaller figure

insert('0.', Rez, length(Rez) + 1);

// Adding missing ranks

if length(Num\_1) < length(Num\_2) then

for I := 1 to (io - 2) do

insert('0', Rez, length(Rez) + 1);

I := io;

Repeat

Stop := False;

// Subtraction

repeat

repeat

// If we need to get a rank from the next one

If (ArrayOfDiv[I] < StrToInt(Num\_2[j])) and (I > 1)

then

// if the next on is zero and we

// again need a rank

if ArrayOfDiv[I - 1] = 0 then

Begin

Ko := 0;

While (ArrayOfDiv[I-Ko]=0) and ((I-Ko)>0) do

Ko := Ko + 1;

if I - Ko > 0 then

// Trying to find the rank, which isn't zero

while K < Ko do

Begin

K := K + 1;

ArrayOfDiv[I - K] := ArrayOfDiv[I - K] - 1;

ArrayOfDiv[I+1-K] := ArrayOfDiv[I+1-K]+10;

End;

End

Else

Begin

// Getting a rank from the next one and

// substracting it

ArrayOfDiv[I - 1] := ArrayOfDiv[I - 1] - 1;

ArrayOfDiv[I] := ArrayOfDiv[I] + 10;

End;

// If isn’t possible to substracting one time more

if (ArrayOfDiv[I] < StrToInt(Num\_2[j])) or

((ArrayOfDiv[I - j + 1] < StrToInt(Num\_2[1])) and

(ArrayOfDiv[I - j] = 0)) then

Stop := true

Else

Begin

ArrayOfDiv[I] := ArrayOfDiv[I] –

StrToInt(Num\_2[j]);

// if the whole divider was subtracted

if j = 1 then

Count := Count + 1;

End;

// Going to new rank

j := j - 1;

I := I - 1;

K := 0;

until (j = 0) or Stop;

// starting again

j := length(Num\_2);

I := io;

Until Stop;

insert(IntToStr(Count), Rez, length(Rez) + 1);

Count := 0;

// Updating 'borders'

j := length(Num\_2);

if length(Num\_1) = length(Num\_2) then

io := io + 1

Else

Begin

io := io + j;

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

insert('0', Rez, length(Rez) + 1);

End;

I := io;

Until length(Rez) > (length(Num\_2) + 5);

Write(z1, Num\_1, ' / ', z1, Num\_2, ' = ', Rez);

End;

readln;

end.

Приложение Б

(обязательное)

Тестовые наборы

Тест 1

Исходные данные:

* первое число – 1250;
* второе число – 5.

Ожидаемый результат:

250

Полученный результат:



Рисунок 2 – Тест 1

Тест 2

Исходные данные:

* первое число – 111111111111111111111111111111111;
* второе число – 999999999999999999999999999999999.

Ожидаемый результат:

0.1111111111111111111111111111111111111

Полученный результат:

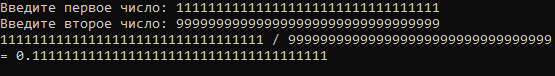


Рисунок 3 – Тест 2

Тест 3

Исходные данные:

* система счисления – 10;
* первое число – 999999999999999999999999999999999;
* второе число – 999999999999999999999999999999999.

Ожидаемый результат:

1

Полученный результат:

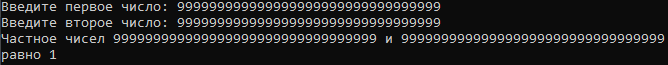


Рисунок 4 – Тест 3

Тест 4

Исходные данные:

* первое число – 100000;
* второе число – 300.

Ожидаемый результат:

* частное: 333;
* остаток: 100.

Полученный результат:

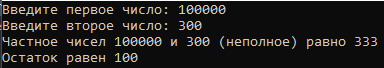


Рисунок 5 – Тест 4

Тест 5

Исходные данные:

* первое число – -32;
* второе число – -1024.

Ожидаемый результат:

0.003125

Полученный результат:



Рисунок 6 – Тест 5

Тест 6

Исходные данные:

* первое число – 1;
* второе число – -999.

Ожидаемый результат:

-0.(001)

Полученный результат:



Рисунок 7 – Тест 6

Тест 7

Таблица 3 – Тестовые ситуации

|  |  |  |
| --- | --- | --- |
| Тест | Тестовые ситуации (вместе с расчётами) | Проверка полученного результата |
| 1. |  | [См. Детали задачи](#_Детали_задачи) |
| 2. |  | [См. Детали задачи](#_Детали_задачи) |