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 лабораторной работе № 2

Тема работы: Числа Смита

Выполнил

студент: гр. 151003 Барановский Р.А.

Проверил: Фадеева Е.П.

Минск 2021

содержание

[1 Постановка задачи 3](#_Toc85921867)

[2 Текстовый алгоритм решения задачи 4](#_Toc85921868)

[3 Структура данных 6](#_Toc85921869)

[4 Схема алгоритма решения задачи по Гост 19.701-90 7](#_Toc85921870)

[5 Результаты расчетов 8](#_Toc85921871)

[Приложение А 9](#_Toc85921872)

[Приложение Б 14](#_Toc85921873)

# Постановка задачи

 Составное число называется числом Смита, если сумма его цифр равна сумме цифр всех чисел, образующихся разложением исходного числа на простые множители. Число Смита называется супер-числом Смита, если сумма его цифр является числом Смита. Разработать алгоритм и программу поиска супер-числа Смита с номером N, являющегося полным квадратом. (Число называется полным квадратом, если квадратный корень из него является простым числом).

# Текстовый алгоритм решения задачи

Таблица 1 – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
| 1 | Ввод N |
| 2 | N1 := 0 |
| 3 | Num := 4 |
| 4 | Начало цикла A1. Проверка выполнения условия N1 < N. Если условие истинно, то идти к шагу 5, иначе к шагу 25 |
| 5 | Проверка выполнения условия sqrt(Num) = trunc(sqrt(Num)). Если условие истинно, то идти к шагу 6, иначе к шагу 23 |
| 6 | Flag1 := false |
| 7 | Flag2 := false |
| 8 | Flag3 := false |
| 9 | Подсчет суммы цифр простых множителей исходного числа |
| 10 | Подсчет суммы цифр исходного числа |
| 11 | Проверка выполнения условия сумма цифр простых множителей исходного числа сумме цифр исходного числа и исходное число – не простое. Если условие истинно то идти к шагу 12, иначе к шагу 13 |
| 12 | Flag1 := true |
| 13 | Подсчет суммы цифр простых множителей значения, являющегося суммой цифр исходного числа |
| 14 | Подсчет суммы цифр значения, являющегося суммой цифр исходного числа |
| 15 | Проверка выполнения условия сумма цифр простых множителей значения, являющегося суммой цифр исходного числа равна сумме цифр значения, являющегося суммой цифр исходного числа и число, являющееся суммой цифр исходного числа – не простое. Если условие истинно то идти к шагу 16, иначе к шагу 17 |
| 16 | Flag2 := true |
| 17 | Нахождение квадратного корня из исходного числа |
| 18 | Проверка выполнения условия квадратный корень из исходного числа – простое число. Если условие истинно, то идти к шагу 19, иначе к шагу 20 |
| 19 | Flag3 := true |
| 20 | Проверка выполнения условия (Flag1 = true) and (Flag2 = true) and (Flag3 = true). Если условие истинно, то идти к шагу 21, иначе к шагу 22 |
| 21 | N1 := N1 + 1 |
| Продолжение таблицы 1 | |
| 22 | Проверка выполнения условия N1 < > N. Если условие истинно, то идти к шагу 23, иначе к шагу 24 |
| 23 | Num := Num + 1 |
|  |  |
| 24 | Конец цикла А1. Идти к шагу 2 |
| 25 | Вывод Num |
| 26 | Останов. |

# Структура данных

Таблица 2 - Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Num | Integer | Проверяемое на соответствие трем условиям число |
| NumDigitCopy1 | Integer | Копия Num, затем SumEachDigit1 |
| NumDigitCopy2 | Integer | Копия Num, затем SumEachDigit1 |
| NumCopy3 | Integer | Копия Num |
| NumCopy4 | Integer | Копия Num |
| Divider | Integer | Делитель, используемый во всех случаях для нахождения простых множителей числа |
| SumDigit | Integer | Переменная хранящая в себе сумму цифр простого множителя числа до нахождения следующего |
| SumEachDigit1 | Integer | Сумма цифр Num |
| SumEachDigit2 | Integer | Cумма цифр значения, являющегося суммой цифр Num |
| DividerCopy | Integer | Копия Divider |
| MultipliersSum1 | Integer | Сумма простых множителей исходного числа |
| MultipliersSum2 | Integer | Сумма простых множителей значения, являющегося суммой цифр исходного числа |
| N1 | Integer | Счетчик цикла |
| N | Integer | Номер числа Смита |
| NStr | String | Проверка ошибок при вводе |
| Error | Integer | Переменная, равная 1, если пользователь допустил ошибку при вводе, 0, если нет |
| SqrtNum | Real | Корень из Num |
| SqrtNumCopy | Real | Копия SqrtNum |
| Flag1 | Boolean | True, если Num – число Смита |
| Flag2 | Boolean | True, если Num – супер-число Смита |
| Flag3 | Boolean | True, если Num – полный квадрат |
| Checker | Boolean | True, если ошибок при вводе допущено не было, иначе False |

# Схема алгоритма решения задачи по Гост 19.701-90

Рисунок - Схема алгоритма решения задачи по ГОСТ 19.701-90

# Результаты расчетов

Вследствие результатов программы на экран выводятся следующие результаты расчетов:

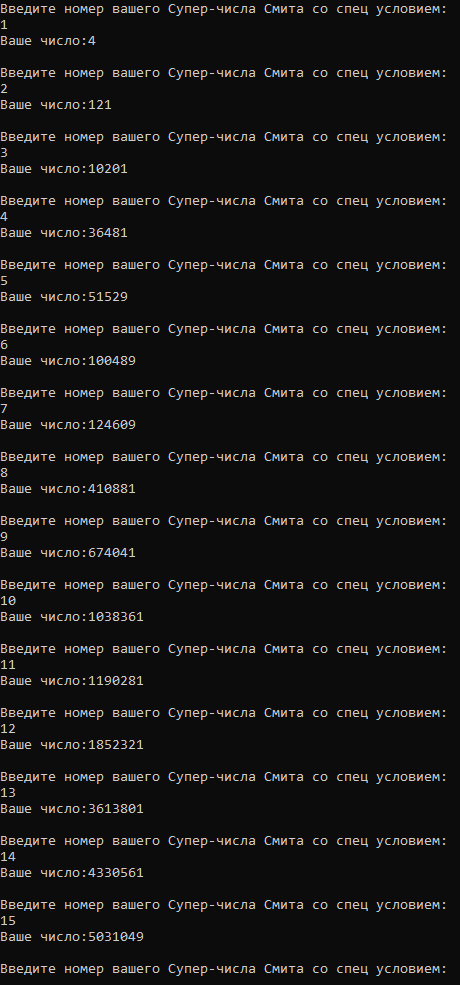


Рисунок – Результаты расчетов 1 программы

Приложение А

(обязательное)

Исходный код программы

program lab2;

//Program finds Smyth number with special conditions

//Console app

{$APPTYPE CONSOLE}

{$R \*.res}

//Modules declaration

uses

System.SysUtils;

//Variables declaration

var Num, NumDigitCopy1, NumDigitCopy2, NumCopy3, NumCopy4, Divider, Sumdigit, SumEachDigit1, SumEachDigit2,

DividerCopy, MultipliersSum1, MultipliersSum2, N1, N,

Error: Integer;

SqrtNum, SqrtNumCopy: real;

Flag1, Flag2, Flag3, Checker: boolean;

NStr: String;

{ Num - potential Smyth number with special conditions

NumDigitCopy1 - Num and then SumEachDigit1 copy

NumDigitCopy2 - Num and then SumEachDigit2 copy

NumCopy3 - Num copy

NumCopy4 - Num copy

Divider - divider in all cases to find all simple

multipliers of certain number

SumDigit - auxiliary variable which contains sum

of digits of multiplier after each division

SumEachDigit1 - sum of digits of original number

SumEachDigit2 - sum of digits of sum of digits of

original number

DividerCopy - Divider copy

MultipliersSum1 - sum of all digits of simple

multipliers of original number

MultipliersSum2 - sum of all digits of simple

multipliers of sum of digits of

original number

N1 - cycle counter

N - number of Smyth number with special conditions

NStr - mistakes checker

Error - variable, which is 1 if user made a mistake and 0

if not

SqrtNum - square root from Num

SqrtNumCopy - SqrtNum copy

Flag1 - Smyth number checker

Flag2 - Super-Smyth number checker

Flag3 - Full square checker

Checker - false if user made a mistake true if not }

//Start the program

Begin

//Repeating the cycle

while(True) do

begin

//Protection from mistakes

Checker := false;

while Checker = false do

begin

writeln('Введите номер вашего Супер-числа Смита со

спец условием:');

readln(NStr);

val(NStr,N,Error);

if (Error = 0) and (N > 0) then

Checker := true

else

writeln('Ошибка ввода');

end;

//Set N1, Num to the original value

N1 := 0;

Num := 4;

//Entering the cycle

while N1 < N do

begin

//Check if square from Num is integer

if sqrt(Num) = trunc(sqrt(Num)) then

begin

//Set Flag1, Flag2, Flag3 to the original value

flag1 := false;

flag2 := false;

flag3 := false;

//Copying Num

NumDigitCopy1 := Num;

NumDigitCopy2 := Num;

NumCopy3 := Num;

NumCopy4 := Num;

//This part of code finds all simple multimpliers

//of Num

Divider := 2;{First divider is 2}

MultipliersSum1 := 0;

while Num > 1 do{While Num is not 1}

begin

if Num mod Divider = 0 then

begin {If Num divides by Divider}

DividerCopy := Divider;

Sumdigit := 0;

//Find sum of all digits of simple multipliers

while DividerCopy <> 0 do

begin

Sumdigit := Sumdigit + (DividerCopy mod 10);

DividerCopy := DividerCopy div 10;

end;

MultipliersSum1 := MultipliersSum1 + Sumdigit;

Num := Num div Divider;

end

else{If Num not divides by Divider}

Divider := Divider + 1;

end;

//This part of code finds sum of all digits

//of original number

SumEachDigit1 := 0;

while NumDigitCopy1<>0 do

begin

SumEachDigit1 := SumEachDigit1 +

(NumDigitCopy1 mod 10);

NumDigitCopy1 := NumDigitCopy1 div 10;

end;

//This part of code checks if Num is a Smyth number

if (SumEachDigit1 = MultipliersSum1) and

(Divider <> NumDigitCopy2) then

Flag1 := true;

//This part of code finds all simple multipliers

//of sum of all digits of original number

NumDigitCopy1 := SumEachDigit1;

NumDigitCopy2 := SumEachDigit1;

Divider := 2;{First divider is 2}

MultipliersSum2 := 0;

while SumEachDigit1 > 1 do

begin {While SumEachDigit is not 1}

if SumEachDigit1 mod Divider = 0 then

begin{If SumEachDigit divides by Divider}

DividerCopy := Divider;

Sumdigit := 0;

//Find sum of all digits of simple multipliers

while DividerCopy<>0 do

begin

Sumdigit := Sumdigit + (DividerCopy mod 10);

DividerCopy := DividerCopy div 10;

end;

MultipliersSum2 := MultipliersSum2 + Sumdigit;

SumEachDigit1 := SumEachDigit1 div Divider;

end

else{If SumEachDigit not divides by Divider}

Divider := Divider + 1;

end;

//This part of code finds sum of all digits of

//sum of all digits of original number

SumEachDigit2 := 0;

while NumDigitCopy1<>0 do

begin

SumEachDigit2 := SumEachDigit2 +

(NumDigitCopy1 mod 10);

NumDigitCopy1 := NumDigitCopy1 div 10;

end;

//This part of code checks if Num is a

//Super-Smyth number

if (SumEachDigit2 = MultipliersSum2) and

(Divider <> NumDigitCopy2) then

Flag2 := true;

//This part of code finds all simple multipliers

//of square root from Num

SqrtNum := sqrt(NumCopy3);{Find square root from

Num}

SqrtNumCopy := SqrtNum;{Memorize square root from

Num}

Divider := 2;{First divider is 2}

while SqrtNum > 1 do

begin{While SqrtNum is not 1}

if trunc(SqrtNum) mod Divider = 0 then{If SqrtNum

divides by

Divider}

SqrtNum := trunc(SqrtNum) div Divider

else{If SqrtNum not divides by divider}

Divider := Divider + 1;

end;

//This part of code checks if Num is a full square

if SqrtNumCopy = Divider then

Flag3 := true;

//If Num satisfies all 3 conditions

if (flag1 = true) and (flag2 = true) and

(flag3 = true) then

N1 := N1 + 1;

//Increment Num

if N1 <> N then

NumCopy4 := NumCopy4 + 1;

Num := NumCopy4;

end

//If square root from Num is not integer

else

Num := Num + 1;

end;

//Output Smyth number with special conditions

writeln('Ваше число:',Num);

//Final operations

writeln;

end;

End.

Приложение Б

(обязательное)

Тестовые наборы

Таблица 3 – Тестовые наборы

|  |  |  |
| --- | --- | --- |
| Тест | Исходные данные и ожидаемый результат | Полученный результат |
| 1 | Исходные данные: Ввод букв в поле для чисел  Ожидаемый результат: Ошибка ввода |  |
| 2 | Исходные данные: Ввод отрицательного значения  Ожидаемый результат:  Ошибка ввода |  |
| 3 | Исходные данные: Ввод не целого значения  Ожидаемый результат:  Ошибка ввода |  |
| 4 | Исходные данные: N = 10  Ожидаемый результат:  Ваше число:1038361 |  |
| 5 | Исходные данные: N = 20  Ожидаемый результат:  Ваше число:105124009 |  |
| 6 | Исходные данные: N = 30  Ожидаемый результат:  Ваше число:640140601 |  |