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: Нахождение НОДа

Выполнил

Проверил: Фадеева Е.П.

Минск 2021

содержание

[1 Постановка задачи 3](#_Toc88933738)

[2 Методика решения 4](#_Toc88933739)

[2.1 Использование алгоритма Евклида 4](#_Toc88933740)

[2.2 Использование циклов 4](#_Toc88933741)

[3 Текстовый алгоритм решения задачи 5](#_Toc88933742)

[4 Структура данных 6](#_Toc88933743)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 7](#_Toc88933744)

[6 Результаты расчетов 9](#_Toc88933745)

[Приложение А 10](#_Toc88933746)

[Приложение Б 12](#_Toc88933747)

# Постановка задачи

Разработать алгоритм и программу нахождения наибольшего общего делителя (двух и более чисел).

Вывести на печать результаты расчётов в виде:

= значение

# Методика решения

## Использование алгоритма Евклида

Чтобы найти НОД двух чисел, необходимо использовать алгоритм Евклида, который звучит так:

*«Если A > B, то A= A - B. Если нет, то B = B – A. Повторять до тех пор, пока А или В не станут равны нулю, в таком случае НОД = А + В».*

Алгоритм Евклида имеет несколько недостатков: большое количество итераций, невозможность работать с отрицательными числами.

Чтобу улучшить алгоритм, нужно привнести в него несколько модификаций:

1. Деление с остатком более эффективно, чем вычитание. В таком случае возникает проблема деления на ноль – его необходимо игнорировать, так как НОД нуля и любого числа равен этому числу.
2. Зная, что НОД отрицательных чисел равен НОДу их модуля, необходимо каждое отрицательное число взять с противоположным знаком.
3. По завершении вычисления НОДа двух чисел неизвестно, какое из них является НОДом, а какое – нулём. Из-за этого было необходимо выводить сумму двух элементов. Это можно исправить, если вместо вычитания первого числа из второго сделать сортировку двух чисел по возрастанию. Тогда НОДом всегда будет второе число, а нулём – первое.

## Использование циклов

НОД трёх чисел рассчитывается, как НОД НОДа двух чисел и третьего числа. Аналогично для четырёх и более чисел.

Так как заранее неизвестно, сколько чисел введёт пользователь, необходимо реализовать алгоритм Евклида для двух элементов массива с индексами I и I+1. При этом I должен изменяться от 1 до N-1, чтобы не выйти за границу массива.

# Текстовый алгоритм решения задачи

Таблица – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
|  | Ввод N |
|  | Ввод A[1..N] |
|  | I:=1 |
|  | Начало цикла А1. Проверка выполнения условия (I < N). Если условие истинно, перейти к шагу 5, иначе – к шагу 9 |
|  | Проверка выполнения условия (A[I] < 0). Если условие истинно, перейти к шагу 6, иначе – к шагу 7 |
|  | A[I]:= -A[I] |
|  | I:=I+1 |
|  | Конец цикла А1. Идти к шагу 4 |
|  | I:=1 |
|  | Начало цикла А2. Проверка выполнения условия (I <= N-1). Если условие истинно, перейти к шагу 11, иначе – к шагу 19 |
|  | Начало цикла А3 |
|  | Проверка выполнения условия (A[I]<>0). Если условие истинно, перейти к шагу 13, иначе – к шагу 17 |
|  | Проверка выполнения условия (A[I]<=A[I+1]). Если условие истинно, перейти к шагу 15, иначе – к шагу 14 |
|  | Swap(A[I], A[I+1]) |
|  | A[I]:=A[I] mod A[I+1] |
|  | Конец цикла А3. Проверка выполнения условия (A[I]=0). Если условие истинно, перейти к шагу 11, иначе – к шагу 17 |
|  | I:=I+1 |
|  | Конец цикла А2. Идти к шагу 10 |
|  | Вывод А[N] |
|  | Останов. |

# Структура данных

Таблица – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| A | Array[1..1000] of Integer | Массив, который вводит пользователь |
| N | Integer | Количество элементов массива, которое будет запрошено у пользователя при вводе |
| I | Integer | Параметр цикла |
| Temp | Integer | Переменная для сохранения значений при обмене элементов массива местами |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

|  |
| --- |
|  |
| Рисунок 1 – Схема алгоритма решения задачи по ГОСТ 19.701-90 |

|  |
| --- |
|  |
| Рисунок 2 – Схема алгоритма решения задачи по ГОСТ 19.701-90 |

# Результаты расчетов

Вследствие результатов программы на экран выводятся следующие результаты расчетов:

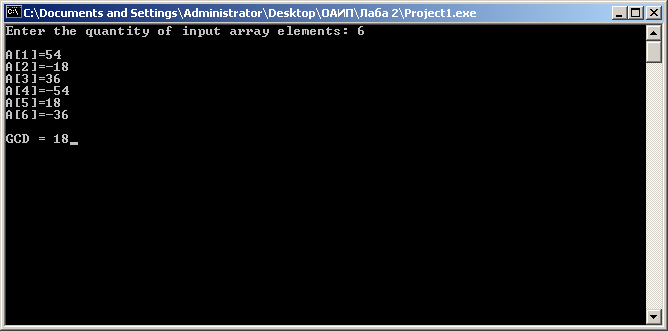


Рисунок 3 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

Program Lab2;

{For a given numbers, calculate it's gcd}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare vars

Var

A: Array[1..1000] of Integer;

Error, Temp, N, I: Integer;

Input: String;

//A - array of numbers to find gcd

//N - array size

//I - cycle parameter

//Temp - needed to swap

//Input - input string

//Error - operator for checking input

Begin

//Checking for the correct input

Repeat

Write('Enter the quantity of input array elements: ');

ReadLn(Input);

Val(Input, N, Error);

If (N <= 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (N > 0) And (Error = 0);

WriteLn;

//Enter every element of array

For I:= 1 To N Do

Begin

//Checking for the correct input

Repeat

Write('A[', I, ']=');

ReadLn(Input);

Val(Input, A[I], Error);

If Error <> 0 Then

WriteLn('Invalid input. Enter another number.');

Until Error = 0;

End;

//GCD(-A,-B) = GCD(A,B)

For I:= 1 to N do

If A[I] < 0 then

A[I]:= -A[I];

WriteLn;

For I:= 1 to N - 1 do

Begin

//Searching for the GCD of two numbers

Repeat

//Mustn't divide zero

If A[I] <> 0 then

Begin

//Euclid algorithm

If A[I] <= A[I + 1] then

A[I + 1]:= A[I + 1] mod A[I]

Else

Begin

//Swap A[I] and A[I+1]

Temp:= A[I];

A[I]:= A[I + 1];

A[I + 1]:= Temp;

End;

End;

Until (A[I] = 0);

End;

//Displaying the GCD

Write('GCD = ', A[N]);

ReadLn;

End.

Приложение Б

(обязательное)

Тестовые наборы

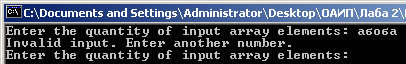
Тестовая ситуация 1: проверка ввода данных

Тестовая ситуация: проверка, сообщит ли программа о некорректном вводе, если введено буквенное или дробное значение.

Исходные данные: N = абоба.

Ожидаемый результат: Invalid input. Enter another number.

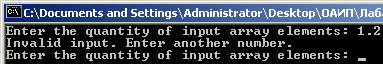
Полученный результат:



Исходные данные: N = 1.2.

Ожидаемый результат: Invalid input. Enter another number.

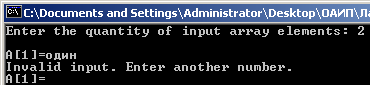
Полученный результат:



Исходные данные: A[1] = один

Ожидаемый результат: Invalid input. Enter another number.

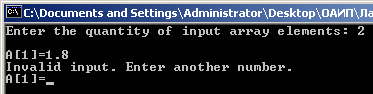
Полученный результат:



Исходные данные: A[1] = 1.8

Ожидаемый результат: Invalid input. Enter another number.

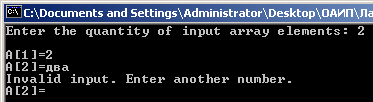
Полученный результат:



Исходные данные: A[2] = два

Ожидаемый результат: Invalid input. Enter another number.

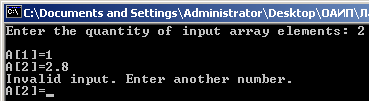
Полученный результат:



Исходные данные: A[2] = 2.8

Ожидаемый результат: Invalid input. Enter another number.

Полученный результат:



Тестовая ситуация 2: проверка вычислений

Тестовая ситуация для проверки вывода правильного НОДа при положительных, отрицательных числах и нулях.

Исходные данные: A[1] = 0; A[2] = 18; A[3] = -18; A[4] = 54; A[5] = -54; A[6] = 36; A[7] =-360;

Ожидаемый результат: НОД = 18.

Полученный результат:

