

Муниципальное бюджетное образовательное учреждение
высшего образования «Самарская академия
государственного и муниципального управления»

Юридический факультет
Кафедра Общонаучных дисциплин и права

Задание №2
по дисциплине «Программная инженерия»
на тему «Структурное программирование. Циклы»

Выполнил Иванов И.И.,
студент группы 921-Д

Проверила Коробецкая А.А.

Задача

Найти наибольший общий делитель (НОД) и наименьшее общее кратное (НОК) двух целых чисел.

Словесное описание

Пользователь вводит два целых положительных числа A и B . Программа вычисляет их наибольший общий делитель (НОД) по алгоритму Евклида (вычитанием):

1. Если $A = B$, то $\text{НОД} = A = B$.
2. В противном случае из большего числа вычесть меньшее, результат записать вместо большего числа.
3. Повторить с шага 1.

Наименьшее общее кратное (НОК) вычисляется по формуле через НОД.

Математическая модель

Дано:

A, B

Алгоритм Евклида:

$A[0] = A \quad B[0] = B$

Пока

$$\text{НОД}(A;B) = A[n] = B[n]$$

НОК:

$$\text{НОК}(A;B) = \frac{A \cdot B}{\text{НОД}(A;B)} \quad (1)$$

Описание работы программы

При загрузке программы экран очищается, выводится текст задачи.

Пользователю предлагается поочередно ввести числа A и B .

В случае ошибочного ввода выводится сообщение, и программа просит ввести неверное число повторно.

После правильного ввода вычисляется НОД по алгоритму Евклида. Результат выводится на экран.

Вычисляется НОК по формуле через НОД. Результат выводится на экран.

Пользователю предлагается повторить решение или выйти из программы.

Входные данные: целые положительные числа А, В.

Выходные данные: значение НОД(А; В), НОК (А; В).

Интерфейс ввода-вывода: текстовый (командная строка)

Возможные ошибки:

<i>Код</i>	<i>Сообщение</i>	<i>Пояснение</i>	<i>Реакция программы</i>
01	Требуется ввести целое положительное число	Неверный ввод координат (текст, дробное число, отрицательное число)	Сообщение об ошибке и запрос повторного ввода числа

Инструментальные средства

Язык программирования: Pascal

Среда программирования: Free Pascal Compiler 2.6

Системные требования

Операционная система: Windows, DOS.

Аппаратная платформа: персональный компьютер архитектуры x86/64, монитор, клавиатура.

Дополнительное программное обеспечение: Компилятор Free Pascal Compiler 2.6 или новее.

Декомпозиция задачи

Выполним декомпозицию задачи на подпрограммы.

1. Подпрограммы ввода данных:

а. Считывание значения

Параметры: -

Возвращает: целое число

Описание: Выполняется проверка правильности введенного значения (целое положительное число). В случае неверного ввода значение запрашивается повторно (в цикле).

б. Запрос выхода из программы

Параметры: -

Возвращает: логический

Описание: Если нажата клавиша выхода (Escape), возвращает True, в противном случае – False.

2. Подпрограммы обработки данных:

а. Алгоритм Евклида

Параметры: А, В: целые числа

Возвращает: значение НОД: целое число

Описание: Классический вариант алгоритма (описан выше).

б. Вычислить НОД

Параметры: А, В: целые числа

Возвращает: значение НОД: целое число

Описание: Общая подпрограмма для вычисления НОД. Вызывает алгоритм Евклида (в перспективе – любой другой алгоритм).

Добавлено дополнительное условие: если одно из чисел А или В равно 1, то $\text{НОД}(А; В) = 1$. Это значительно ускорит работу алгоритма для таких случаев и незначительно замедлит в случаях с $\text{НОД} \neq 1$.

с. Вычислить НОК

Параметры: А, В, НОД: целые числа

Возвращает: значение НОК: длинное целое число

Описание: Расчет по формуле (1). Если НОД не передается в качестве параметра ($\text{НОД} \leq 0$), то его значение запрашивается из подпрограммы НОД.

3. Подпрограммы вывода данных:

а. Вывод приветствия

Параметры: -

Возвращает: -

Описание: Очищает экран, выводит описание задачи

б. Вывод результатов расчетов

Параметры: имя: строка, значение: целое число

Возвращает: -

Описание: Выводит на экран значение результата.

с. Вывод сообщения об ошибке

Параметры: ошибка: специальный тип

Возвращает: -

Описание: Выводит на экран номер ошибки и пояснение.

Блок-схемы

На рисунке 1 представлена схема тела программы, на рисунках 2-4 – основных подпрограмм. Прочие подпрограммы являются тривиальными и в виде блок-схем не показаны.

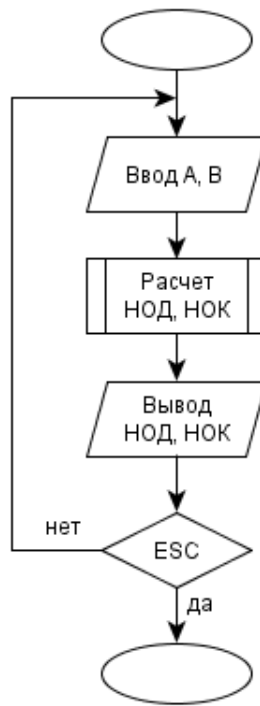


Рис. 1. Блок-схема тела программы

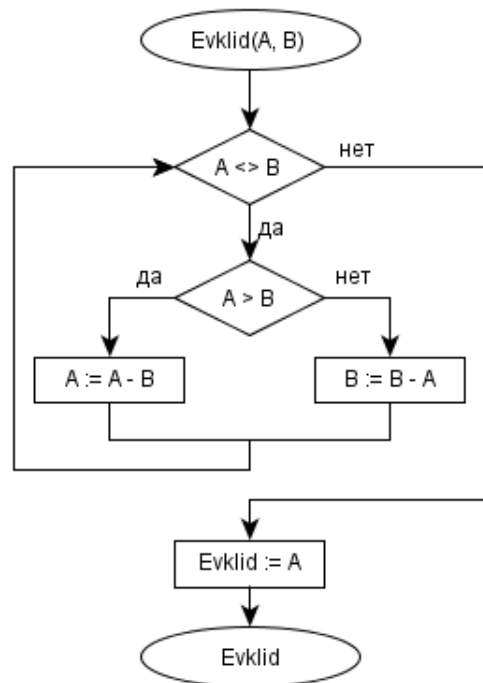


Рис. 2. Блок-схема подпрограммы «Алгоритм Евклида»

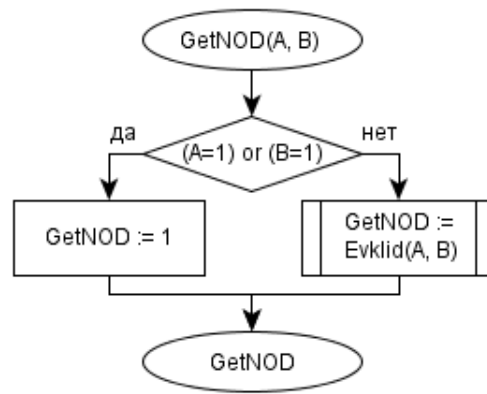


Рис. 3. Блок-схема подпрограммы «Вычисление НОД»

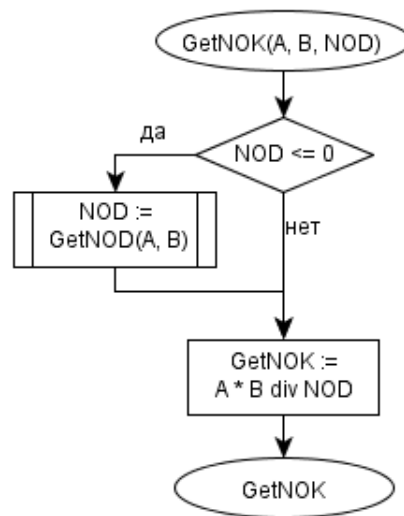


Рис. 4. Блок-схема подпрограммы «Вычисление НОК»

Исходный код

Программный модуль Nod_Nok.pas

```
program GCD;
{Найти наибольший общий делитель (НОД) и наименьшее общее кратное (НОК)
двух целых чисел.}

uses crt;

// =====

type
//сообщение об ошибке
TErrorMsg = record
    Code: Byte; //код ошибки
    Text: String; //текст пояснения
end;

// =====

var
    A, B: Integer; //исходные значения
    NOD: Integer; //наименьший общий делитель
    NOK: LongInt; //наибольшее общее кратное

// =====

const
    Msg_Description =
        'Найти наибольший общий делитель (НОД) и наименьшее общее кратное (НОК)
двух целых чисел.';
    Msg_GetValue =
        'Введите целое положительное число ';
    Msg_Error = 'Ошибка';
    Msg_PressToExit = 'Нажмите ESC для выхода или любую другую клавишу для
повтора.';
    Msg_NOD = 'НОД';
    Msg_NOK = 'НОК';

    ERRORMSG_01_WrongInt: TErrorMsg
        = (Code: 1; Text: 'Требуется ввести целое положительное число');

// =====

//вывод сообщения об ошибке
procedure PrintError(const E: TErrorMsg);
begin
    Writeln(Msg_Error, ' ', E.Code, ': ', E.Text);
end;

//вывод на экран условия задачи
procedure PrintHello;
begin
    ClrScr;

    Writeln('+++++');
    Writeln(Msg_Description);

    Writeln('+++++');
```

```

    Writeln;
end;

//конец работы? да - если нажат ESC, нет - иначе
function ReadEnd: Boolean;
begin
    Writeln;
    Writeln(Msg_PressToExit);
    ReadEnd := (ReadKey = #27);
end;

//ввод числа
function ReadValue(const ValueName: String): Integer;
var
    Code: Integer;
    //запрос ввода с попыткой перевести строку в число
    procedure DoRead(var Value, Code: Integer);
    var
        Str: String;
    begin
        Write(Msg_GetValue, ValueName, ' = ');
        Readln(Str);
        Val(Str, Value, Code);
    end;
begin
    // первое чтение
    DoRead(ReadValue, Code);
    // пока есть ошибка - повторить ввод
    while (Code <> 0) or (ReadValue < 0) do
    begin
        PrintError(ERRORMSG_01_WrongInt);
        DoRead(ReadValue, Code);
    end;
end;

//вывод результата (НОД или НОК)
procedure WriteResult(Name: String; Value: Integer);
begin
    Writeln(Name, ' = ', Value);
end;

// =====

//алгоритм Евклида
function Evklid(A, B: Integer): Integer;
begin
    // пока числа не равны
    while A <> B do
        //из большего вычесть меньшее
        if A > B then    A := A - B
            else        B := B - A;
    // ответ = A = B
    Evklid := A;
end;

// вычисление НОД (вызов алгоритма)
function GetNOD(A, B: Integer): Integer;
begin
    //если одно из чисел = 1, то НОД = 1
    if (A = 1) or (B = 1) then
        GetNOD := 1
    // иначе - по алгоритму Евклида
    else

```



```

    GetNOD := Evklid(A, B);
end;

//вычисление НОК через НОД
function GetNOK(A, B: Integer; NOD: Integer): LongInt;
begin
    //если НОД не задан - вычислить
    if NOD <= 0 then
        NOD := GetNOD(A, B);

    //вычисление
    GetNOK := A * B div NOD;
end;

//=====

begin
PrintHello;

repeat
    //ввод данных
    A := ReadValue('x');
    B := ReadValue('y');

    //расчеты
    NOD := GetNOD(A, B);
    NOK := GetNOK(A, B, NOD);

    //вывод
    WriteResult(Msg_NOD, NOD);
    WriteResult(Msg_NOK, NOK);

    //повторить?
until ReadEnd;

end.

```

Тестовые примеры

Порядок тестирования: ручной ввод входных величин.

№	Входные данные		Результат		Примечание
	А	В	НОД	НОК	
1	1	50	1	50	Без алгоритма Евклида
2	70	80	10	560	
3	13	20	1	260	Через алгоритм Евклида
4	0		Ошибка 01: Требуется ввести целое положительное число		Сразу после ввода А
	676f		Ошибка 01: Требуется ввести целое положительное число		Сразу после ввода А
	15	-5	Ошибка 01: Требуется ввести положительное целое число		Сразу после ввода В
		9	3	45	