

Отчёт по лабораторной работе №4.

Вычисление наибольшего общего делителя

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Лапшенкова Любовь Олеговна 1032217633

Группа: НФИмд-02-21

Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

2 декабря, 2021, Москва

Цели и задачи работы

Целью данной лабораторной работы является ознакомление с алгоритмами вычисления наибольшего общего делителя, – а так же реализация алгоритмов на произвольном языке программирования.

Реализовать все рассмотренные в инструкции к лабораторной работе алгоритмы нахождения наибольшего общего делителя программно.

Ход выполнения и результаты

Входные данные

```
#ввели числа для поиска НОДа  
a=27  
b=9
```

Figure 1: Входные данные для реализации алгоритмов по нахождению НОД

Алгоритм Евклида. Реализация

```
#алгоритм Евклида
def algorithm_Evklida(a,b):
    '''
    Расписываем пункты 1-4 для алгоритма Евклида
    '''
    r=[]
    r.append(a)
    r.append(b)
    i=1
    while r[i]!=0:
        i+=1
        r.append(r[i-2]%r[i-1])
    d=r[i-1]
    print('НОД(' ,a, ',' ,b, ')=',d)
    algorithm_Evklida(a,b)
```

Figure 2: 2 часть программного кода реализации алгоритма Евклида для нахождения НОД

$$\text{НОД}(27 , 9) = 9$$

Figure 3: Результат реализации алгоритма Евклида для нахождения НОД

Бинарный алгоритм Евклида. 1 способ. Реализация

```
#бинарный алгоритм Евклида
'''
Выпишем два случая реализации программы для бинарного алгоритма Евклида
'''

#1 случай (с использованием алгоритма Евклида и ограничивающих условий)
def bin1_algorithm_Evkliida(a,b):
    def NOD(a,b):
        '''
        Расписываем пункты 1-4 для алгоритма Евклида
        '''

        r=[]
        r.append(a)
        r.append(b)
        i=1
        while r[i]!=0:
            i+=1
            r.append(r[i-2]%r[i-1])
        d=r[i-1]
        return d
```

Figure 4: 1 часть программного кода реализации бинарного алгоритма Евклида 1 способом для нахождения НОД

Бинарный алгоритм Евклида. 1 способ. Реализация

```
if ((b>0)and(a>=b)):
    '''проверка на обязательное условие'''
    if ((a%2==0)and(b%2==0)):
        '''проверка на четность'''
        print('НОД(' ,a , ' , ' ,b , ')= ' ,2*NOD(a//2,b//2))
    if ((a%2!=0)and(b%2==0)):
        '''проверка на то, что a нечетное, а b четное'''
        print('НОД(' ,a , ' , ' ,b , ')= ' ,NOD(a,b//2))
    if ((a%2!=0)and(b%2!=0)and(a>b)):
        '''проверка на то, что оба нечетные'''
        print('НОД(' ,a , ' , ' ,b , ')= ' ,NOD(a-b,b))
    if (a==b):
        '''если числа равны друг другу'''
        print('НОД(' ,a , ' , ' ,b , ')= ' ,a)
else:
    print('Перепроверьте входные данные! Должно быть: (0<b<=a)')
binl_algorithm_Evkliida(a,b)
```

Figure 5: 2 часть программного кода реализации бинарного алгоритма Евклида 1 способом для нахождения НОД

$$\text{НОД}(27 , 9) = 9$$

Figure 6: Результат реализации бинарного алгоритма Евклида для нахождения НОД (1 способ)

Бинарный алгоритм Евклида. 2 способ. Реализация

```
#бинарный алгоритм Евклида
#2 случай (по алгоритму)
def bin2_algorithm_Evklida(a,b):
    '''
    Выполняем пункты 1-6 по алгоритму
    '''
    g=1
    u=a
    v=b
    while ((a%2==0)and(b%2==0)):
        '''
        пока четные выполнять до получения хотя бы одного нечетного
        '''
        a=a//2
        b=b//2
    g=2*g
```

Figure 7: 1 часть программного кода реализации бинарного алгоритма Евклида 2 способом для нахождения НОД

Бинарный алгоритм Евклида. 2 способ. Реализация

```
while (u!=0):  
    '''  
    выполняем пункт 4 алгоритма  
    '''  
    if (u%2==0):  
        u=u//2  
    else:  
        u=u  
    if (v%2==0):  
        v=v//2  
    else:  
        v=v  
    if u>=v:  
        u=u-v  
    else:  
        v=v-u  
d=g*v  
return d  
if ((b>0)and(a>=b)):  
    print('НОД(' ,a , ' , ' ,b , ')=' ,bin2_algorithm_Evkliada(a,b))  
else:  
    print('Перепроверьте входные данные! Должно быть: (0<b<=a)')
```

Figure 8: 2 часть программного кода реализации бинарного алгоритма Евклида 2 способом для нахождения НОД

$$\text{НОД}(27 , 9) = 9$$

Figure 9: Результат реализации бинарного алгоритма Евклида для нахождения НОД (2 способ)

Расширенный алгоритм Евклида. Реализация

```
#расширенный алгоритм Евклида
def rassh_algorithm_Evklida(a,b):
    '''
    выполняем пункты 1-4 по расширенному алгоритму Евклида
    '''

    #пункт 1
    r=[]
    x=[]
    y=[]
    r.append(a)
    r.append(b)
    x.append(1)
    x.append(0)
    y.append(0)
    y.append(1)
    i=1
    while r[i]!=0:
        '''
        пункт 2
        '''
        i+=1
        r.append(r[i-2]%r[i-1])
```

Figure 10: 1 часть программного кода реализации расширенного алгоритма Евклида для нахождения НОД

Расширенный алгоритм Евклида. Реализация

```
if r[i]==0:
    ...
    пункт 3
    ...
    d=r[i-1]
    x=x[i-1]
    y=y[i-1]
else:
    x.append(x[i-2]-((r[i-2]//r[i-1])*x[i-1]))
    y.append(y[i-2]-((r[i-2]//r[i-1])*y[i-1]))
return d,x,y
if ((b>0)and(a>=b)):
    print('НОД(','a',' ','b,')=',rassh_algorithm_Evklida(a,b)[0])#тк три значения в return пишем индекс ответа, который нужен
    print('x=',rassh_algorithm_Evklida(a,b)[1])#тк три значения в return пишем индекс ответа, который нужен
    print('y=',rassh_algorithm_Evklida(a,b)[2])#тк три значения в return пишем индекс ответа, который нужен
else:
    print('Перепроверьте входные данные! Должно быть: (0<b<=a)')
```

Figure 11: 2 часть программного кода реализации расширенного алгоритма Евклида для нахождения НОД

$$\begin{aligned}\text{НОД}(27, 9) &= 9 \\ x &= 0 \\ y &= 1\end{aligned}$$

Figure 12: Результат реализации расширенного алгоритма Евклида для нахождения НОД

Расширенный бинарный алгоритм Евклида. Реализация

```
#расширенный бинарный алгоритм Евклида
def rassh_bin_algorithm_Evklida(a,b):
    '''
    6 пунктов алгоритма к выполнению
    '''
    g=1
    u=a
    v=b
    A=1
    B=0
    C=0
    D=1
    while ((a%2==0)and(b%2==0)):
        '''
        пока четные выполнять до получения хотя бы одного нечетного
        '''
        a=a//2
        b=b//2
        g=2*g
    '''
    пока u не равно 0
    '''
```

Figure 13: 1 часть программного кода реализации расширенного бинарного алгоритма Евклида для нахождения НОД

Расширенный бинарный алгоритм Евклида. Реализация

```
while (u!=0):  
    #пока u четное  
    if (u%2==0):  
        u=u//2  
        #если оба A и B четные  
        if ((A%2==0)and(B%2==0)):  
            A=A//2  
            B=B//2  
        else:  
            A=(A+b)//2  
            B=(B-a)//2  
    else:  
        u=u  
    #пока v четное  
    if (v%2==0):  
        v=v//2  
        #если оба C и D четные  
        if ((C%2==0)and(D%2==0)):  
            C=C//2  
            D=D//2  
        else:  
            C=(C+b)//2  
            D=(D-a)//2  
    else:  
        v=v
```

Расширенный бинарный алгоритм Евклида. Реализация

```
#пункт 4.3
if u>=v:
    u=u-v
    A=A-C
    B=B-D
else:
    v=v-u
    C=C-A
    D=D-B

#пункт 5
d=g*v
x=C
y=D
return d,x,y
if ((b>0)and(a>=b)):
    print('НОД(' ,a , ' ,b, ')=',rassh_bin_algorithm_Evklida(a,b)[0])
    print('x=',rassh_bin_algorithm_Evklida(a,b)[1])
    print('y=',rassh_bin_algorithm_Evklida(a,b)[2])
else:
    print('Перепроверьте входные данные! Должно быть: (0<b<=a)')
```

Figure 15: 3 часть программного кода реализации расширенного бинарного алгоритма Евклида для нахождения НОД

$$\begin{aligned}\text{НОД}(27, 9) &= 9 \\ x &= 0 \\ y &= 1\end{aligned}$$

Figure 16: Результат реализации расширенного бинарного алгоритма Евклида для нахождения НОД

Спасибо за внимание