Лабораторная работа №4

Вычисление наибольшего общего делителя

дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Койфман Кирилл Дмитриевич Группа: НФИмд-01-25

Цель работы Получение практических навыков реализации алгоритмов, вычисляющих наибольший общий делитель (НОД).

Задачи

1. Реализовать алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида, расширенный бинарный алгоритм Евклида.

---

1 задание

Для решения задачи данной работы была написана программа на C++, реализующая описанные в работе алгоритмы:

```
int algorithmEuclid(int a, int b)
{
    //d=НОД(a, b)
    int d = 0;
    int gamma_prev = a;
    int gamma_curr = b;
    int gamma_next = 0;
    while (true)
    {
        int remainder = gamma_prev % gamma_curr;
        gamma_next = remainder;
        if (gamma_next == 0)
        {
            d = gamma_curr;
            break;
        }
        else
        {
            gamma_prev = gamma_curr;
            gamma_curr = gamma_next;
        }
    }
    return d;
}
```

---

1 задание

```
int binaryAlgorithmEuclid(int a, int b)
{
    //d=НОД(a, b)
    int d = 0;
    int g = 1;
    //until one in pair (a or b) becomes odd
    while (a % 2 == 0 && b % 2 == 0)
    {
        a = a / 2;
        b = b / 2;
        g = 2 * g;
    }
    int u = a;
    int v = b;
    while (u != 0)
    {
        if (u % 2 == 0) u = u / 2;
        if (v % 2 == 0) v = v / 2;
        if (u >= v) u = u - v;
        else v = v - u;
    }
    d = g * v;
    return d;
}
```

---

1 задание

```
EuclidAlgoVars extendedAlgorithmEuclid(int a, int b)
{
    //a * x + b * y = d
    int d = 0;
    int x = 0;
    int y = 0;
    //gamma_0{i-1}
    int gamma_prev = a;
    //gamma_1{i}
    int gamma_curr = b;
    //gamma_{i+1}
    int gamma_next = 0;
    //x_0{i-1}
    int x_prev = 1;
    //x_1{i}
    int x_curr = 0;
    //x_{i+1}
    int x_next = 0;
    //y_0{i-1}
    int y_prev = 0;
```

```
    //y_1{i}
    int y_curr = 1;
    //y_{i+1}
    int y_next = 0;
    while (true)
    {
        int remainder = gamma_prev / gamma_curr;
        int q_curr = remainder;
        //gamma_{i+1} = gamma_{i-1} - q_i * gamma_i
        gamma_next = gamma_prev - q_curr * gamma_curr;
        if (gamma_next == 0)
        {
            d = gamma_curr;
            x = x_curr;
            y = y_curr;
            break;
        }
        else
        {
            x_next = x_prev - q_curr * x_curr;
            y_next = y_prev - q_curr * y_curr;
            gamma_prev = gamma_curr;
            gamma_curr = gamma_next;
            x_prev = x_curr;
            x_curr = x_next;
            y_prev = y_curr;
            y_curr = y_next;
        }
    }
    return EuclidAlgoVars{ d,x,y };
}
```

---

1 задание

```
EuclidAlgoVars binaryExtendedAlgorithmEuclid(int a, int b)
{
    //a * x + b * y = d
    int d = 0;
    int x = 0;
    int y = 0;
    int g = 1;
    //until one in pair (a or b) becomes odd
    while (a % 2 == 0 && b % 2 == 0)
    {
        a = a / 2;
        b = b / 2;
        g = 2 * g;
    }
    int u = a;
    int v = b;
```

```
    int A = 1;
    int B = 0;
    int C = 0;
    int D = 1;
    while (u != 0)
    {
        if (u % 2 == 0)
        {
            u = u / 2;
            if (A % 2 == 0 && B % 2 == 0)
            {
                A = A / 2;
                B = B / 2;
            }
            else
            {
                A = (A + b) / 2;
                B = (B - a) / 2;
            }
        }
        if (v % 2 == 0)
        {
            v = v / 2;
            if (C % 2 == 0 && D % 2 == 0)
            {
                C = C / 2;
                D = D / 2;
            }
            else
            {
                C = (C + b) / 2;
                D = (D - a) / 2;
            }
        }
        if (u >= v)
        {
            u = u - v;
            A = A - C;
            B = B - D;
        }
        else
        {
            v = v - u;
            C = C - A;
            D = D - B;
        }
    }
    d = g * v;
    x = C;
    y = D;
    return EuclidAlgoVars{ d,x,y };
}
```

1 задание

И протестируем работу этих алгоритмов:

```
-------------------------Testing GCD-algorithms-------------------------
TEST-1: a = 14, b = 21
Great Common Divisor(GCD) for pair{a=14, b=21} with [1]Euclid Algorithm: 7
Great Common Divisor(GCD) for pair{a=14, b=21} with [2]Binary Euclid
Algorithm: 7
Great Common Divisor(GCD) for pair{a=14, b=21} with [3]Extended Euclid
Algorithm: 7 with condition that a * x + b * y = d, d = 7, x = -1, y = 1:
14 * -1 + 21 * 1 = 7(true)
Great Common Divisor(GCD) for pair{a=14, b=21} with [4]Binary Extended
Euclid Algorithm: 7 with condition that a * x + b * y = d, d = 7, x = -10,
y = 7:
14 * -10 + 21 * 7 = 7(true)

TEST-2: a = 48, b = 36
Great Common Divisor(GCD) for pair{a=48, b=36} with [1]Euclid Algorithm:
12
Great Common Divisor(GCD) for pair{a=48, b=36} with [2]Binary Euclid
Algorithm: 12
Great Common Divisor(GCD) for pair{a=48, b=36} with [3]Extended Euclid
Algorithm: 12 with condition that a * x + b * y = d, d = 12, x = 1, y =
-1:
48 * 1 + 36 * -1 = 12(true)
Great Common Divisor(GCD) for pair{a=48, b=36} with [4]Binary Extended
Euclid Algorithm: 12 with condition that a * x + b * y = d, d = 12, x =
-5, y = 7:
48 * -5 + 36 * 7 = 12(true)

TEST-3: a = 17, b = 51
Great Common Divisor(GCD) for pair{a=17, b=51} with [1]Euclid Algorithm:
17
Great Common Divisor(GCD) for pair{a=17, b=51} with [2]Binary Euclid
Algorithm: 17
Great Common Divisor(GCD) for pair{a=17, b=51} with [3]Extended Euclid
Algorithm: 17 with condition that a * x + b * y = d, d = 17, x = 1, y = 0:
17 * 1 + 51 * 0 = 17(true)
Great Common Divisor(GCD) for pair{a=17, b=51} with [4]Binary Extended
Euclid Algorithm: 17 with condition that a * x + b * y = d, d = 17, x = 1,
y = 0:
17 * 1 + 51 * 0 = 17(true)

TEST-4: a = 75, b = 250
Great Common Divisor(GCD) for pair{a=75, b=250} with [1]Euclid Algorithm:
25
Great Common Divisor(GCD) for pair{a=75, b=250} with [2]Binary Euclid
Algorithm: 25
Great Common Divisor(GCD) for pair{a=75, b=250} with [3]Extended Euclid
Algorithm: 25 with condition that a * x + b * y = d, d = 25, x = -3, y =
```

```
1:
75 * -3 + 250 * 1 = 25(true)
Great Common Divisor(GCD) for pair{a=75, b=250} with [4]Binary Extended
Euclid Algorithm: 25 with condition that a * x + b * y = d, d = 25, x =
-93, y = 28:
75 * -93 + 250 * 28 = 25(true)
```

Спасибо за внимание!