

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины «Алгоритмизация»
Вариант 8

Выполнил:
Данилецкий Дмитрий Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

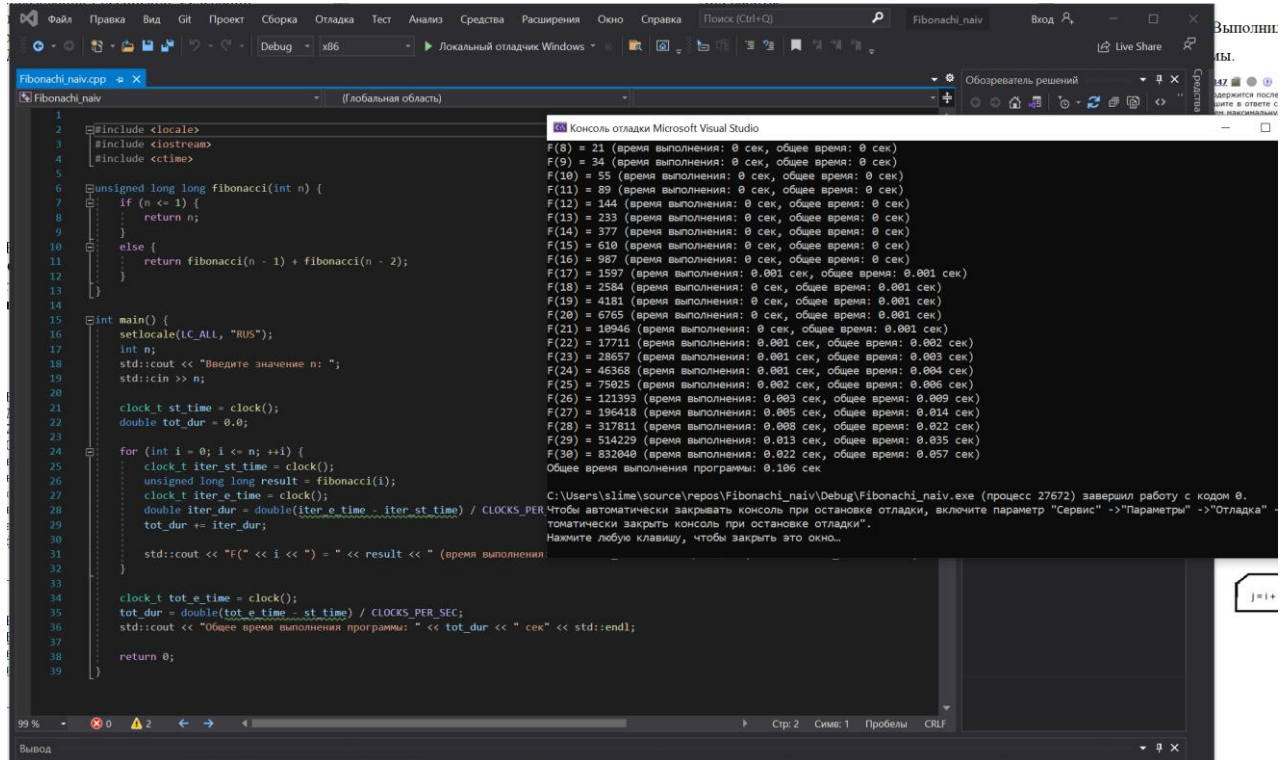
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Написал программу, которая считает числа Фибоначчи через рекурсивную функцию и измерил время за которое она рассчитывает каждое число.



```
1 #include <locale>
2 #include <iostream>
3 #include <ctime>
4
5 unsigned long long fibonacci(int n) {
6     if (n <= 1) {
7         return n;
8     }
9     else {
10        return fibonacci(n - 1) + fibonacci(n - 2);
11    }
12 }
13
14 int main() {
15     setlocale(LC_ALL, "RUS");
16     int n;
17     std::cout << "Введите значение n: ";
18     std::cin >> n;
19
20     clock_t st_time = clock();
21     double tot_dur = 0.0;
22
23     for (int i = 0; i <= n; ++i) {
24         clock_t iter_st_time = clock();
25         unsigned long long result = fibonacci(i);
26         clock_t iter_e_time = clock();
27         double iter_dur = double(iter_e_time - iter_st_time) / CLOCKS_PER_SEC;
28         tot_dur += iter_dur;
29
30         std::cout << "F(" << i << ") = " << result << " (время выполнения: ";
31
32     }
33
34     clock_t tot_e_time = clock();
35     tot_dur = double(tot_e_time - st_time) / CLOCKS_PER_SEC;
36     std::cout << "Общее время выполнения программы: " << tot_dur << " сек" << std::endl;
37
38     return 0;
39 }
```

```
F(0) = 0 (время выполнения: 0 сек, общее время: 0 сек)
F(1) = 1 (время выполнения: 0 сек, общее время: 0 сек)
F(2) = 1 (время выполнения: 0 сек, общее время: 0 сек)
F(3) = 2 (время выполнения: 0 сек, общее время: 0 сек)
F(4) = 3 (время выполнения: 0 сек, общее время: 0 сек)
F(5) = 5 (время выполнения: 0 сек, общее время: 0 сек)
F(6) = 8 (время выполнения: 0 сек, общее время: 0 сек)
F(7) = 13 (время выполнения: 0 сек, общее время: 0 сек)
F(8) = 21 (время выполнения: 0 сек, общее время: 0 сек)
F(9) = 34 (время выполнения: 0 сек, общее время: 0 сек)
F(10) = 55 (время выполнения: 0 сек, общее время: 0 сек)
F(11) = 89 (время выполнения: 0 сек, общее время: 0 сек)
F(12) = 144 (время выполнения: 0 сек, общее время: 0 сек)
F(13) = 233 (время выполнения: 0 сек, общее время: 0 сек)
F(14) = 377 (время выполнения: 0 сек, общее время: 0 сек)
F(15) = 610 (время выполнения: 0 сек, общее время: 0 сек)
F(16) = 987 (время выполнения: 0 сек, общее время: 0 сек)
F(17) = 1597 (время выполнения: 0.001 сек, общее время: 0.001 сек)
F(18) = 2584 (время выполнения: 0 сек, общее время: 0.001 сек)
F(19) = 4181 (время выполнения: 0 сек, общее время: 0.001 сек)
F(20) = 6765 (время выполнения: 0 сек, общее время: 0.001 сек)
F(21) = 10946 (время выполнения: 0 сек, общее время: 0.001 сек)
F(22) = 17711 (время выполнения: 0.001 сек, общее время: 0.002 сек)
F(23) = 28657 (время выполнения: 0.001 сек, общее время: 0.003 сек)
F(24) = 46368 (время выполнения: 0.001 сек, общее время: 0.004 сек)
F(25) = 75025 (время выполнения: 0.002 сек, общее время: 0.006 сек)
F(26) = 121393 (время выполнения: 0.003 сек, общее время: 0.009 сек)
F(27) = 196418 (время выполнения: 0.005 сек, общее время: 0.014 сек)
F(28) = 317811 (время выполнения: 0.008 сек, общее время: 0.022 сек)
F(29) = 514229 (время выполнения: 0.013 сек, общее время: 0.035 сек)
F(30) = 832040 (время выполнения: 0.022 сек, общее время: 0.057 сек)
Общее время выполнения программы: 0.106 сек
```

Рисунок 1. Результат работы программы

2. Написал программу, считающую числа Фибоначчи с записью их в массив для исключения многочисленных повторяющихся вычислений. Также добавил цикл, делающий 1000000 итераций для каждого числа, чтобы появилась возможность измерить время, поскольку программа работала слишком быстро.

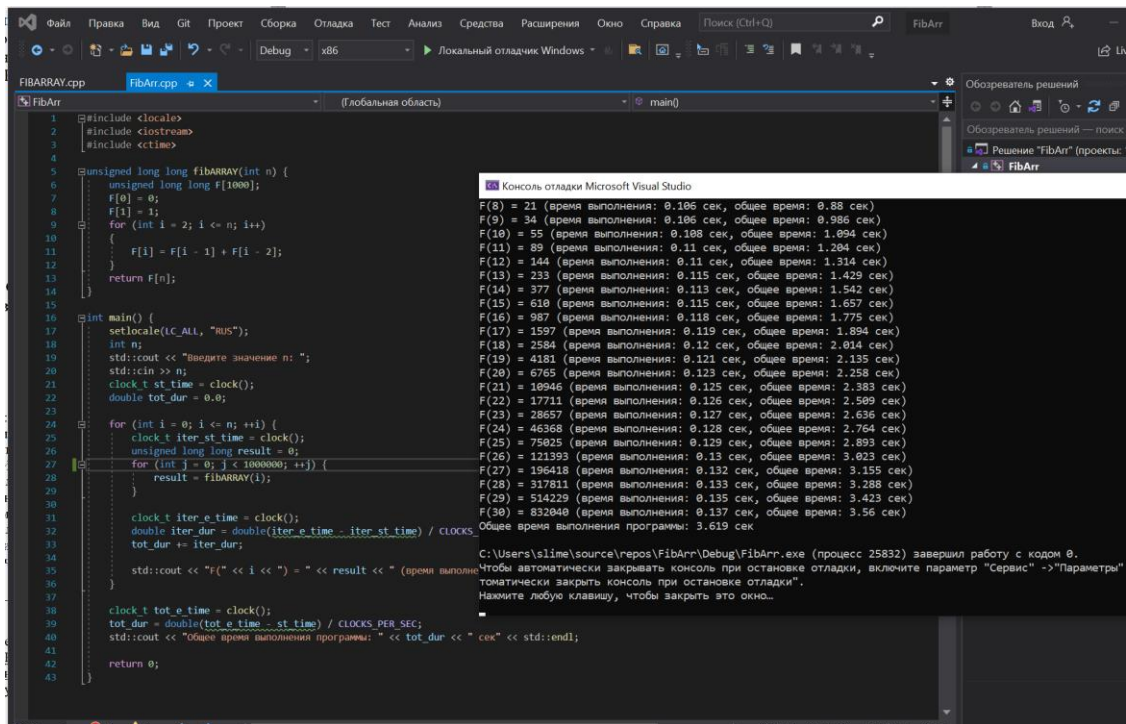


Рисунок 2. Результат программы для первых 30 чисел последовательности

3. Сделал наглядную демонстрацию разницы в эффективности этих двух программ с помощью таблиц excel.

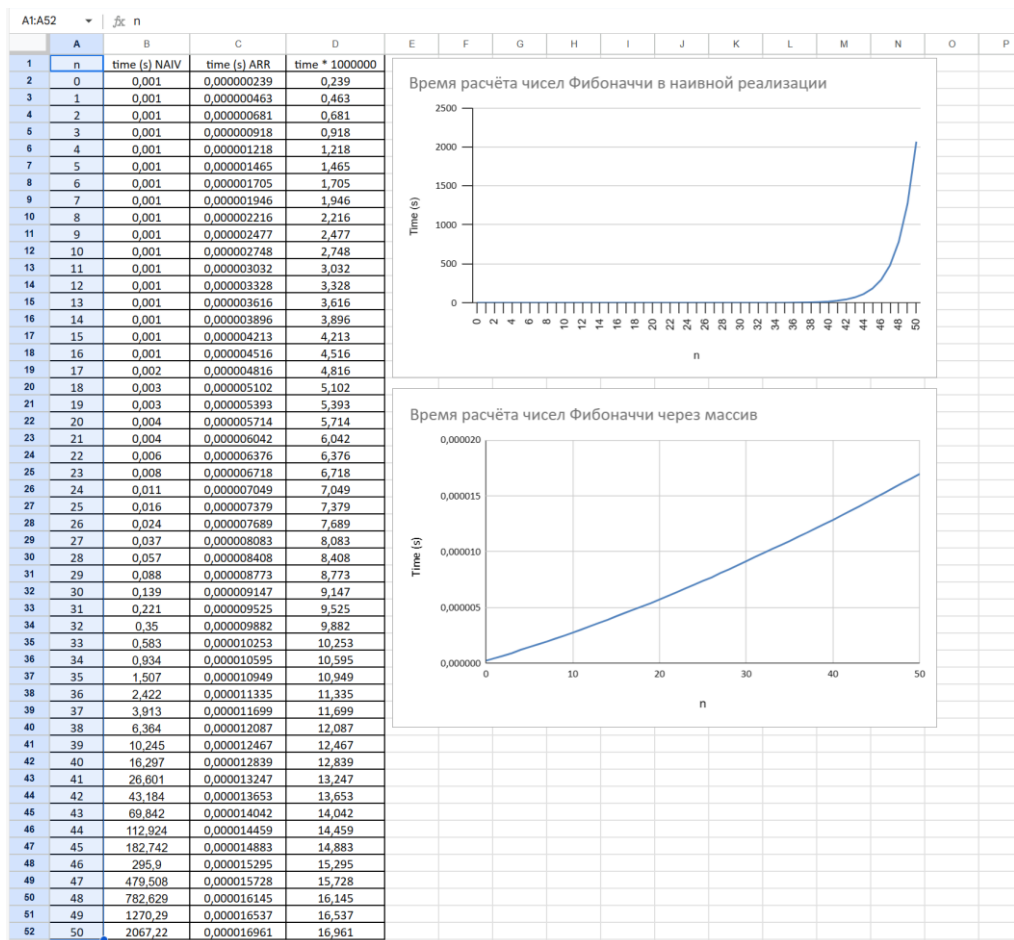


Рисунок 3. Графики для чисел Фибоначчи

4. Написал программу, вычисляющую наибольший общий делитель для двух целых чисел через цикл с перебором всех возможных целых чисел, которые меньше максимального из пары данных.

появилась возможность измерить время.

```

1 #include <iostream>
2 #include <ctime>
3
4 int Naivegcd(int a, int b) {
5     int gcd = 1;
6     int MAX = std::max(a, b);
7
8     for (int d = 2; d <= MAX; d++) {
9         if (a % d == 0 && b % d == 0) {
10             gcd = d;
11         }
12     }
13
14     return gcd;
15 }
16
17 int main() {
18     int a = 3918848;
19     long int b = 1653264;
20     for (int i = 1; i <= 11; i++) {
21         clock_t st_time = clock();
22         long int res = Naivegcd(a, b);
23         clock_t e_time = clock();
24         double time = double(e_time - st_time) / CLOCKS_PER_SEC;
25         std::cout << "ITER " << i << ": Gcd(" << a << ", " << b << ") = " << res << ", time: " << time << " sec" << std::endl;
26         b *= 2;
27     }
28
29     return 0;
30 }

```

Консоль отладки Microsoft Visual Studio

```

ITER 1: Gcd(3918848, 1653264) = 61232, time: 0.006 sec
ITER 2: Gcd(3918848, 3306528) = 122464, time: 0.006 sec
ITER 3: Gcd(3918848, 6613056) = 244928, time: 0.01 sec
ITER 4: Gcd(3918848, 13226112) = 489856, time: 0.02 sec
ITER 5: Gcd(3918848, 26452224) = 979712, time: 0.039 sec
ITER 6: Gcd(3918848, 52904448) = 1959424, time: 0.08 sec
ITER 7: Gcd(3918848, 105808896) = 3918848, time: 0.153 sec
ITER 8: Gcd(3918848, 211617792) = 3918848, time: 0.318 sec
ITER 9: Gcd(3918848, 423235584) = 3918848, time: 0.631 sec
ITER 10: Gcd(3918848, 846471168) = 3918848, time: 1.227 sec
ITER 11: Gcd(3918848, 1692942336) = 3918848, time: 2.414 sec

```

C:\Users\slime\source\repos\ConsoleApplication26\Debug\ConsoleApplication26.exe
 Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры".
 Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 4. Результат работы программы (НОД через перебор)

5. Разработал программу, выполняющую расчёт НОД для двух целых чисел с помощью алгоритма Евклида.

```

1 #include <iostream>
2 #include <ctime>
3
4 int EuclidGCD(int a, int b) {
5     if (a < b) {
6         int d = a;
7         a = b;
8         b = d;
9     }
10
11     while (b != 0) {
12         int d = b;
13         b = a % b;
14         a = d;
15     }
16
17     return a;
18 }
19
20 int main() {
21     int a = 3918848, res;
22     int b = 1653264;
23     for (int i = 1; i <= 11; i++) {
24         clock_t st_time = clock();
25         for (int i = 0; i <= 1000000000; i++) {
26             res = EuclidGCD(a, b);
27         }
28         clock_t e_time = clock();
29         double time = double(e_time - st_time) / CLOCKS_PER_SEC;
30         std::cout << "ITER " << i << ": Gcd(" << a << ", " << b << ") = " << res << ", time: " << time << " sec" << std::endl;
31         b *= 2;
32     }
33
34     return 0;
35 }

```

Консоль отладки Microsoft Visual Studio

```

ITER 1: Gcd(3918848, 1653264) = 61232, time: 1.713 sec
ITER 2: Gcd(3918848, 3306528) = 122464, time: 1.405 sec
ITER 3: Gcd(3918848, 6613056) = 244928, time: 1.47 sec
ITER 4: Gcd(3918848, 13226112) = 489856, time: 1.434 sec
ITER 5: Gcd(3918848, 26452224) = 979712, time: 1.301 sec
ITER 6: Gcd(3918848, 52904448) = 1959424, time: 1.222 sec
ITER 7: Gcd(3918848, 105808896) = 3918848, time: 1.142 sec
ITER 8: Gcd(3918848, 211617792) = 3918848, time: 1.141 sec
ITER 9: Gcd(3918848, 423235584) = 3918848, time: 1.144 sec
ITER 10: Gcd(3918848, 846471168) = 3918848, time: 1.136 sec
ITER 11: Gcd(3918848, 1692942336) = 3918848, time: 1.134 sec

```

C:\Users\slime\source\repos\ConsoleApplication27\Debug\ConsoleApplication27.exe (процесс 12384) заверш
 0.
 Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры".
 Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 5. Результат работы программы (НОД через алгоритм Евклида)

6. Сделал графическую демонстрацию превосходства второго метода над первым.

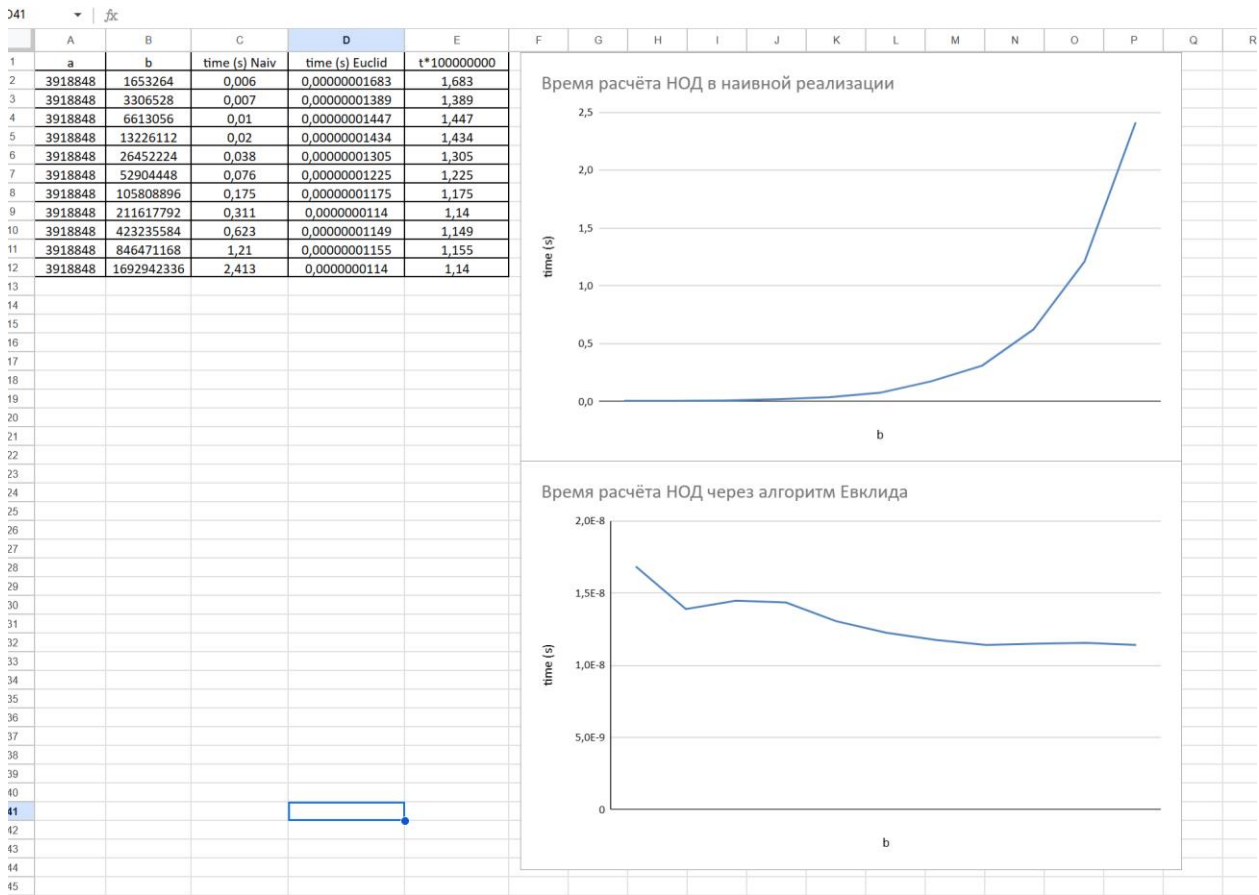


Рисунок 6. Графики для НОД