Лабораторная работа №6

Тема: Разложение чисел на множители

Выполнила: Исламова Сания Маратовна

Группа: НПИмд-01-24

Студ.билет: 1132249576

Задача лабораторной работы:

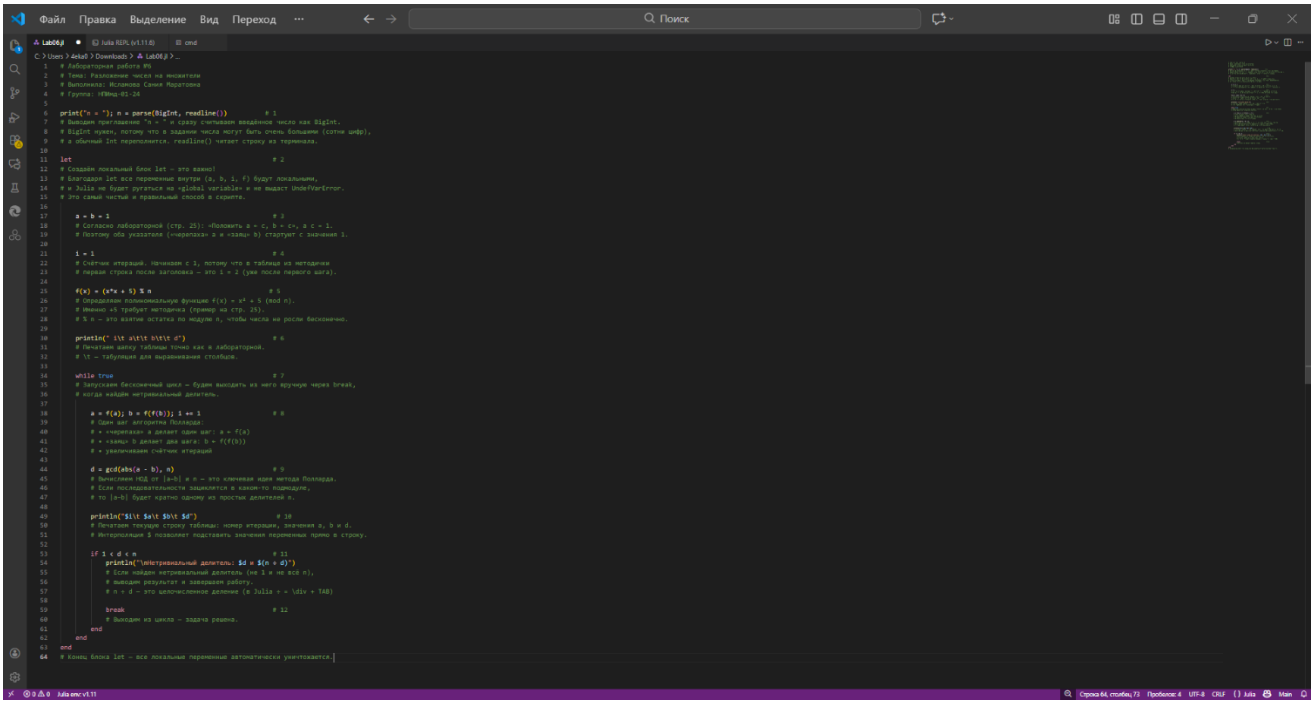
Реализовать рассмотренный алгоритм программно: Алгоритм, реализующий p-метод Полларда

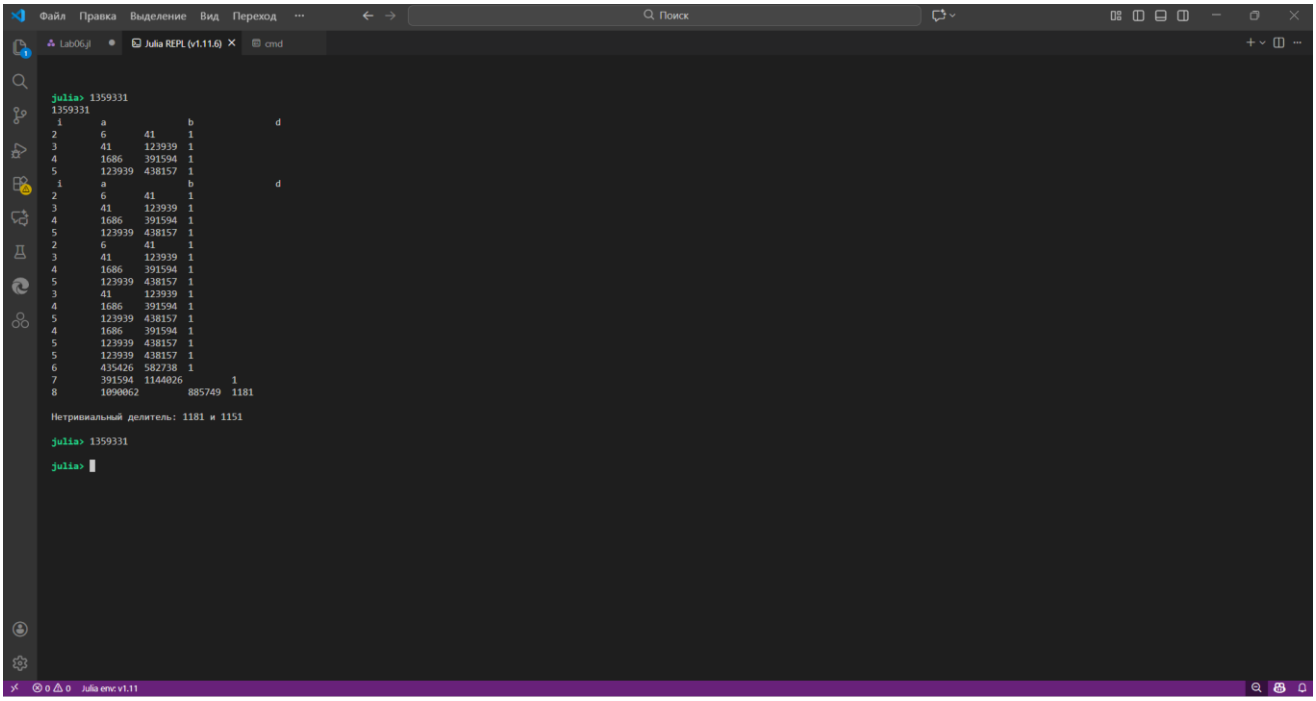
Описание хода выполнения лабораторной работы:

```
print("n = "); n = parse(BigInt, readline())          # 1
# Выводим приглашение "n = " и сразу считываем введённое число как BigInt.
# BigInt нужен, потому что в задании числа могут быть очень большими (сотни
цифр),
# а обычный Int переполнится. readline() читает строку из терминала.

let                                                   # 2
# Создаём локальный блок let — это важно!
# Благодаря let все переменные внутри (a, b, i, f) будут локальными,
# и Julia не будет ругаться на «global variable» и не выдаст UndefVarError.
# Это самый чистый и правильный способ в скрипте.

    a = b = 1                                         # 3
    # Согласно лабораторной (стр. 25): «Положить a ← c, b ← c», а c = 1.
    # Поэтому оба указателя («черепаха» a и «заяц» b) стартуют с значения 1.

    i = 1                                             # 4
    # Счётчик итераций. Начинаем с 1, потому что в таблице из методички
    # первая строка после заголовка — это i = 2 (уже после первого шага).

    f(x) = (x*x + 5) % n                              # 5
    # Определяем полиномиальную функцию f(x) = x² + 5 (mod n).
    # Именно +5 требует методичка (пример на стр. 25).
    # % n — это взятие остатка по модулю n, чтобы числа не росли бесконечно.

    println(" i\t a\t\t b\t\t d")                     # 6
    # Печатаем шапку таблицы точно как в лабораторной.
    # \t — табуляция для выравнивания столбцов.

    while true                                        # 7
    # Запускаем бесконечный цикл — будем выходить из него вручную через break,
    # когда найдём нетривиальный делитель.
```

```
        a = f(a); b = f(f(b)); i += 1                    # 8
        # Один шаг алгоритма Полларда:
        # • «черепаха» a делает один шаг: a ← f(a)
        # • «заяц» b делает два шага: b ← f(f(b))
        # • увеличиваем счётчик итераций

        d = gcd(abs(a - b), n)                           # 9
        # Вычисляем НОД от |a-b| и n — это ключевая идея метода Полларда.
        # Если последовательности зациклятся в каком-то подмодуле,
        # то |a-b| будет кратно одному из простых делителей n.

        println("$i\t $a\t $b\t $d")                      # 10
        # Печатаем текущую строку таблицы: номер итерации, значения a, b и d.
        # Интерполяция $ позволяет подставить значения переменных прямо в
строку.

        if 1 < d < n                                     # 11
            println("\nНетривиальный делитель: $d и $(n ÷ d)")
            # Если найден нетривиальный делитель (не 1 и не всё n),
            # выводим результат и завершаем работу.
            # n ÷ d — это целочисленное деление (в Julia ÷ = \div + TAB)

            break                                        # 12
            # Выходим из цикла — задача решена.
        end
    end
end
# Конец блока let — все локальные переменные автоматически уничтожаются.
```

Результат реализации рассмотренного алгоритма




Файл Правка Выделение Вид Переход
Поиск
Lab06.jl
Julia REPL (v1.11.6)
cmd
julia> 1359331
1359331
i a b d
2 6 41 1
3 41 123939 1
4 1686 391594 1
5 123939 438157 1
i a b d
2 6 41 1
3 41 123939 1
4 1686 391594 1
5 123939 438157 1
2 6 41 1
3 41 123939 1
4 1686 391594 1
5 123939 438157 1
3 41 123939 1
4 1686 391594 1
5 123939 438157 1
4 1686 391594 1
5 123939 438157 1
5 123939 438157 1
6 435426 582738 1
7 391594 1144026 1
8 1090062 885749 1181
Нетривиальный делитель: 1181 и 1151
julia> 1359331
julia>
Julia env: v1.11