**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: «Поиск с возвратом»**

Студентка гр. 6381 \_\_\_\_\_\_\_\_\_\_\_\_\_ Лопатина А.С.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Филатов А.Ю.

Санкт-Петербург

2018

**Задание.**

У Вовы много квадратных обрезков доски. Их стороны (размер) изменяются от 1 до N−1, и у него есть неограниченное число обрезков любого размера. Но ему очень хочется получить большую столешницу - квадрат размера N. Он может получить ее, собрав из уже имеющихся обрезков(квадратов).

Например, столешница размера 7×7 может быть построена из 9 обрезков.

Внутри столешницы не должно быть пустот, обрезки не должны выходить за пределы столешницы и не должны перекрываться. Кроме того, Вова хочет использовать минимально возможное число обрезков.

**Входные данные:**

Размер столешницы - одно целое число N(2≤N≤40).

**Выходные данные:**

Одно число K, задающее минимальное количество обрезков (квадратов), из которых можно построить столешницу(квадрат) заданного размера N. Далее должны идти K строк, каждая из которых должна содержать три целых числа x,y и w, задающие координаты левого верхнего угла (1≤x,y≤N) и длину стороны соответствующего обрезка(квадрата).

**Пояснение задания.**

Необходимо, чтобы программа считывала длину стороны исходного квадрата, а затем при помощи перебора с возвратом производила его деление на минимальное количество квадратов.

**Описание алгоритма.**

1. На вход подается длина стороны квадрата. Проверяется её делимость на 2, 5 и 6.
2. Если число четное, кратное 5 или при делении на 6 в остатке дает 3, применяется соответствующий шаблон (в целях сокращения времени работы программы для квадратов с большой длиной стороны). В программе с полным перебором сразу же осуществляется вход в рекурсивную функцию.
3. Если же число простое, происходит заполнение квадратом со стороной (n+1)/2 и двумя квадратами со стороной (n-1)/2 в левый верхний, правый верхний и левый нижний углы соответственно (вновь в целях сокращения работы программы). Затем осуществляется вход в рекурсивную функцию для оставшейся незаполненной части. В программе с полным перебором вход в рекурсию осуществляется без этой вставки квадратов.
4. В рекурсивной функции начинается заполнение пустого места квадратами, начиная с квадратов длиной 1х1 и двигаясь слева направо сверху вниз. Наименьшее число полученных квадратов хранится в отдельной переменной, и как только во время перебора текущее число квадратов становится больше или равно наименьшему, либо заканчиваются пустые ячейки и число квадратов оказывается больше полученного на тот момент наименьшего, решение забывается и делается шаг назад в поисках более оптимального решения. Если же по окончании пустых ячеек квадратов получилось меньше, чем наименьшее их число, полученное на тот момент, программа перезаписывает лучшее решение. В итоге программа получит разделение на наименьшее количество квадратов.

**Описание функций.**

1. **void removeSquare(int\*\* &squareTmp, int color, int x, int y, int n)** – удаление определенного квадрата;
2. **bool installSquare(int\*\* &squareTmp, int color, int size, int x, int y, int n)**– установка квадрата размера size;
3. **bool findEmptySquare(int\*\* &squareTmp, int& x, int& y, int n)** – поиск незаполненной ячейки;
4. **void recurcion(int& min\_square, int n, int\*\* &squareTmp, int\*\* &squareBest, int x, int y, int& temp\_square) –** основная рекурсивная функция;
5. **void print\_square(int x,int y,int size)** – вывод координат левого верхнего угла квадрата и его длины;
6. **void dev5(int size), void dev6(int size), void dev2(int size)** – вывод координат левого верхнего угла квадрата и его длины для квадратов, длина стороны которых кратна 5, при делении на 6 дает остаток 3 и кратна 2 соответственно;

**Тестирование.**

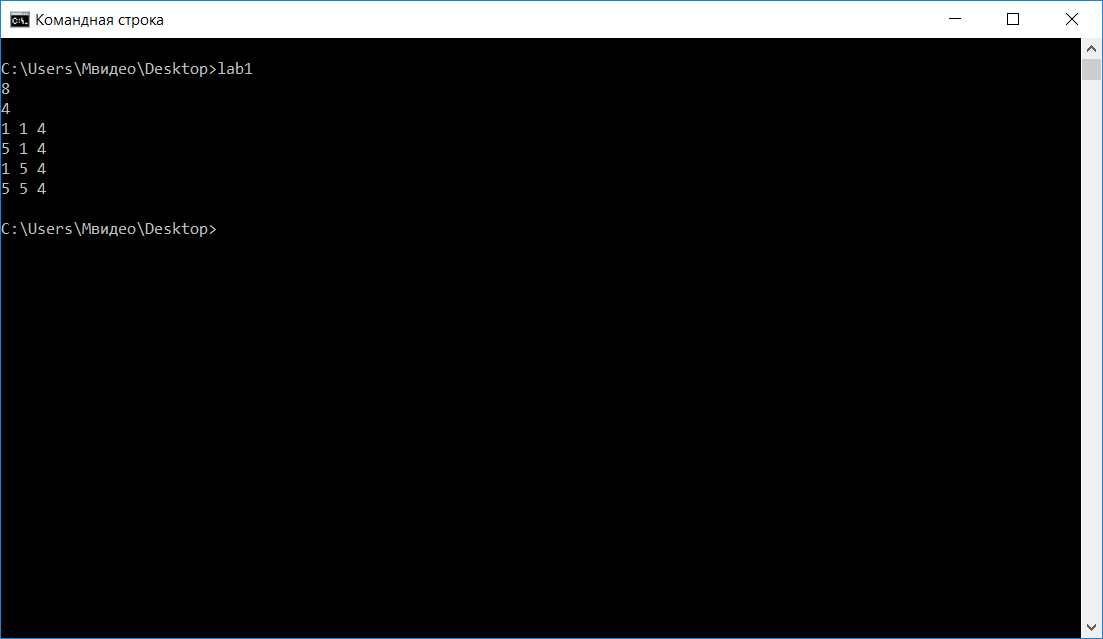


рис. 1

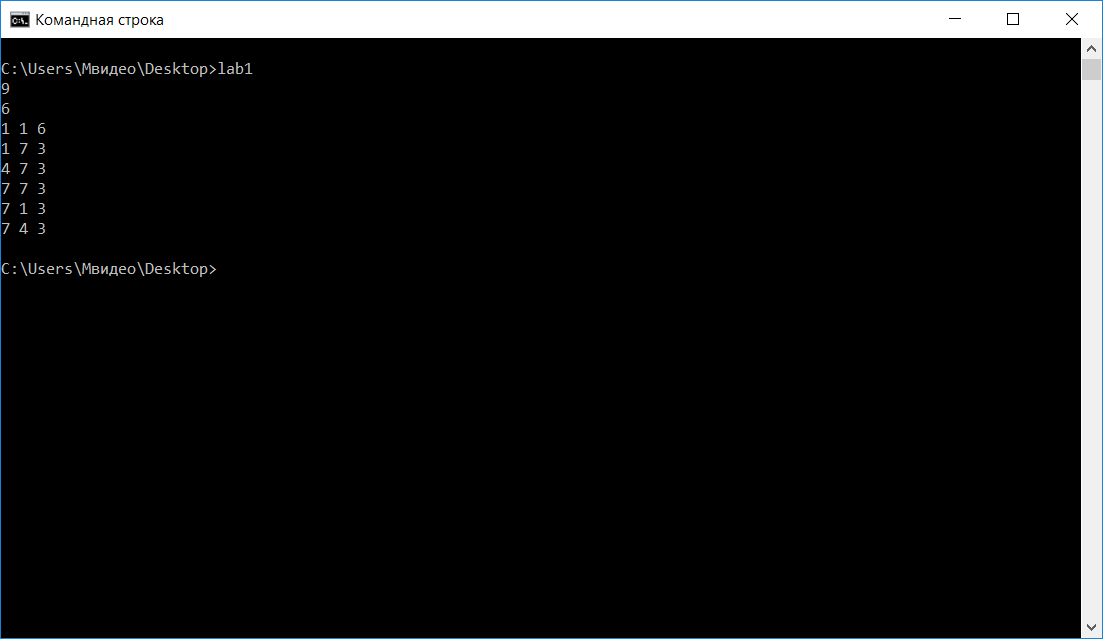


рис. 2

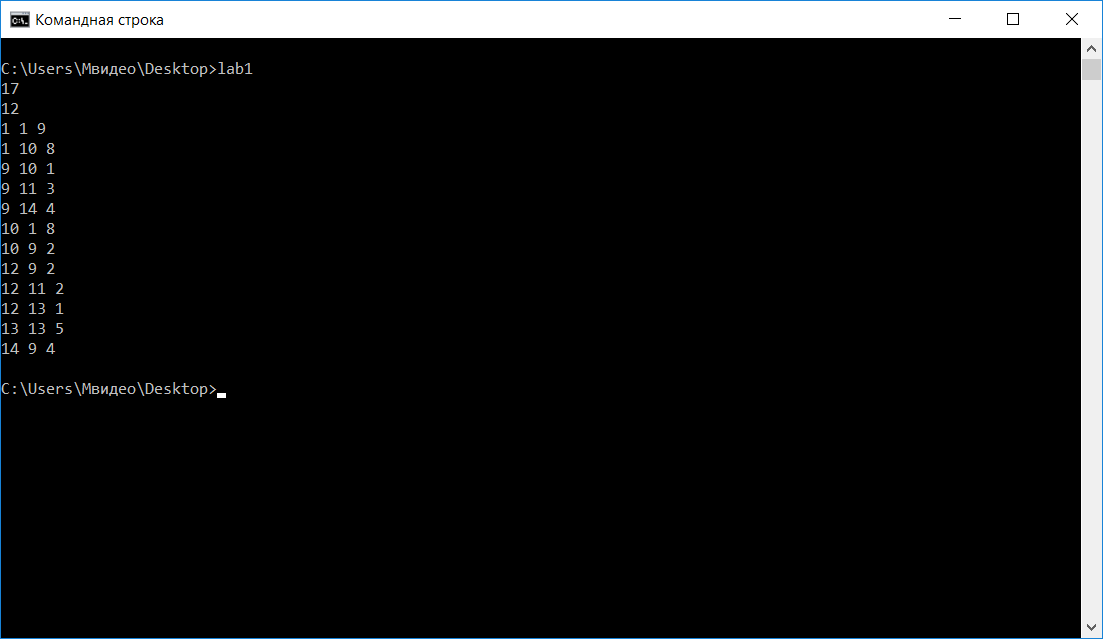


рис. 3

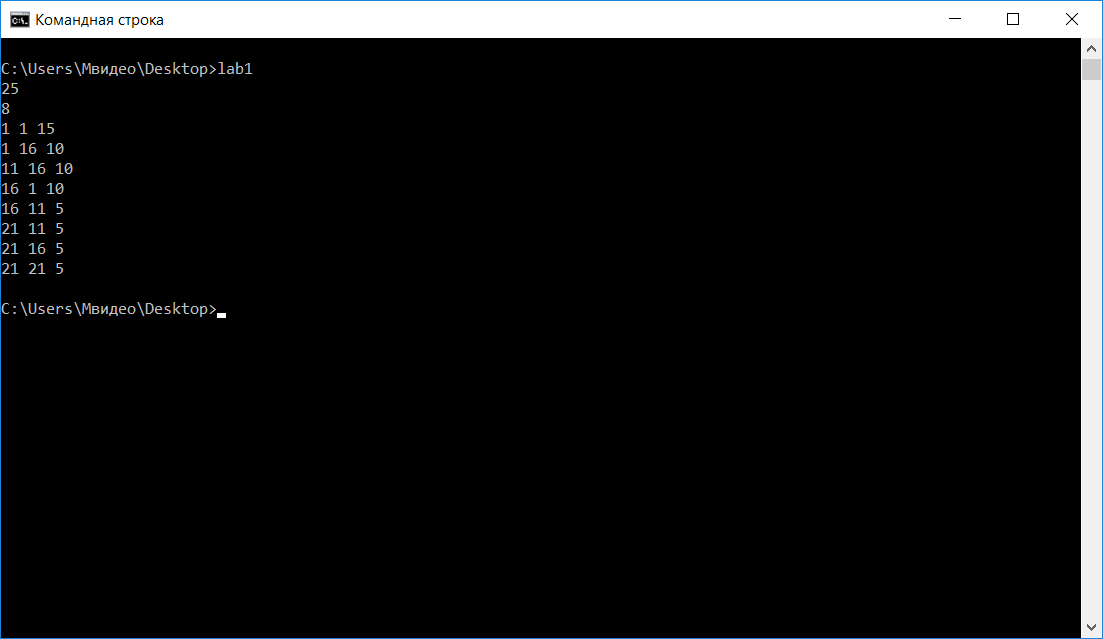


рис. 4

**Выводы.**

В ходе лабораторной работы мы подробно ознакомились с принципом поиска с возвратом. В результате выполнения лабораторной работы была разработана программа, которая вычисляет наименьшее количество квадратов, на которые можно разделить квадрат длиной от 2 до 40.

**Приложение А. Обоснование введения шаблонов.**

Убедимся, что для квадратов с длиной стороны кратной 2, 5 и при делении на 6 дающей остаток 3, можно применить шаблон в целях ускорения времени работы программы и это будет считаться лучшим решением. А также убедимся, что в квадратах с длиной стороны равной простому числу, можно произвести заполнение квадратом со стороной (n+1)/2 и двумя квадратами со стороной (n-1)/2 в левый верхний, правый верхний и левый нижний углы соответственно, а лишь затем выполнить рекурсивный поиск, и что это тоже будет лучшим решением.

Для этого разработаем программу, выполняющую полный перебор и проверим, совпадают ли её решения с решениями программы, содержащей упрощения.

При тестировании этой программы теми же числами получаем:

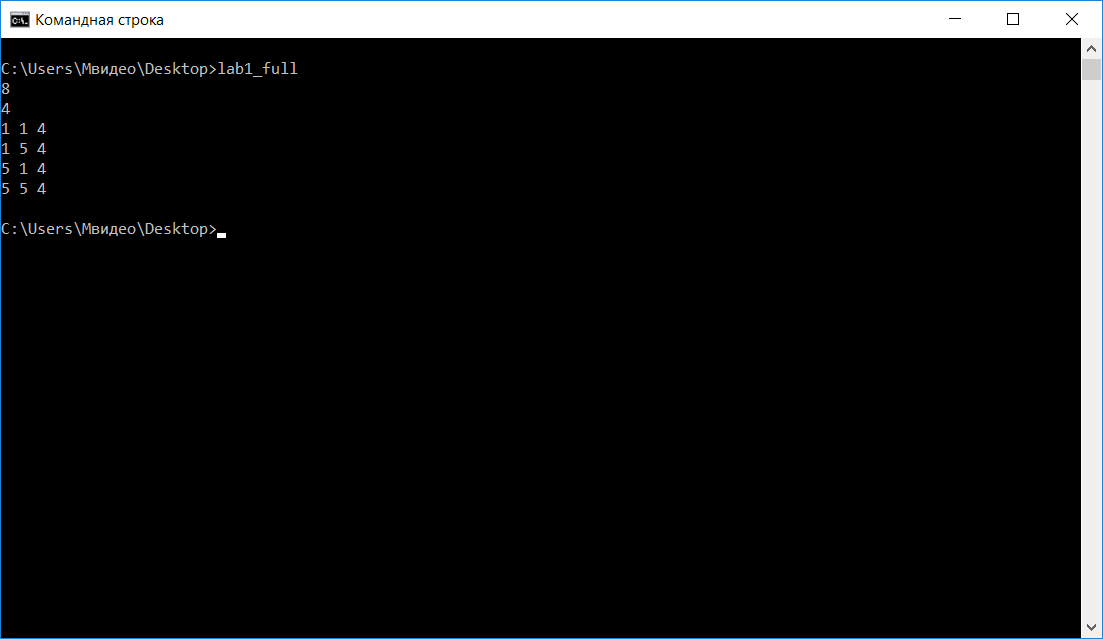


рис. 5

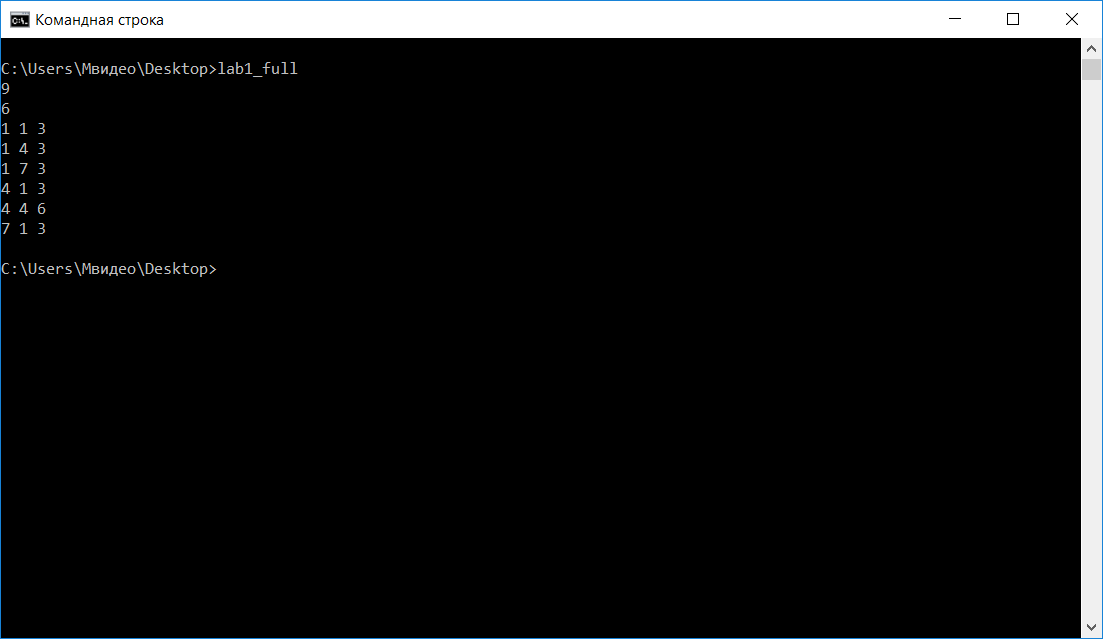


рис. 6

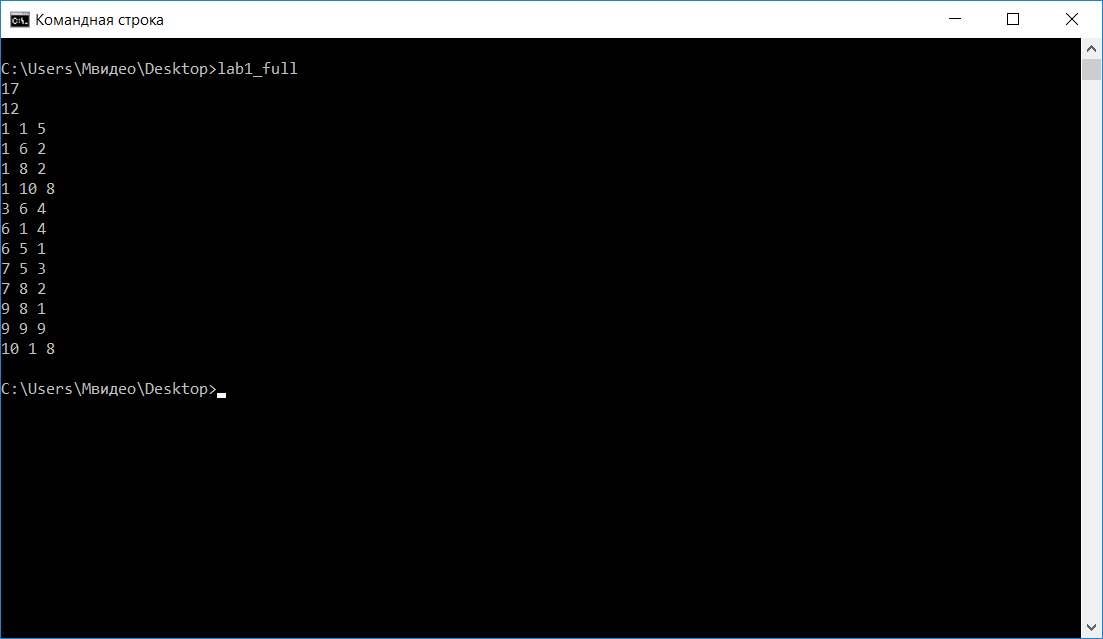


рис. 7

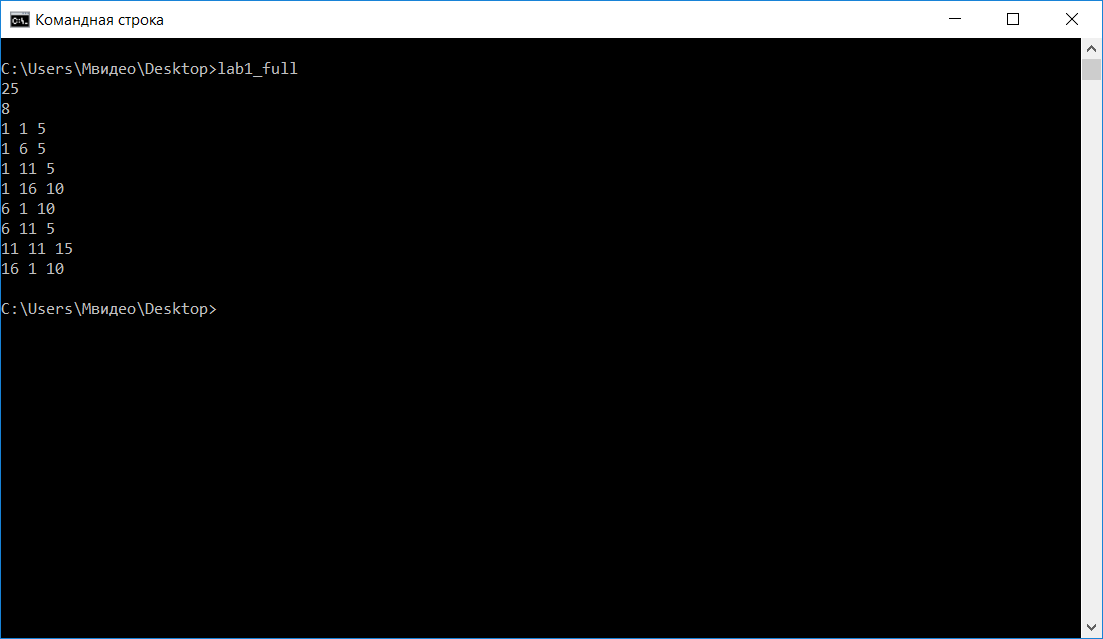


рис. 8

Как можно заметить разложения получились идентичными за исключением положения квадратов, но они не меняют решения. Для демонстрации этого изобразим разрезания квадрата со стороной длины 9. На рис. 9 результат для программы с упрощением, на рис. 10 полным перебором.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

рис. 9

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

рис. 10

Для некоторых простых чисел, такое как 11, например, решения программ могут не совпасть, однако содержать они будут одинаковое количество квадратов. Связано это с тем, что полный перебор выбирает один из вариантов решения с наименьшим числом квадратом, а этих вариантов может быть несколько. Если в программе с полным перебором указать вставку квадратов до размера (n+1)/2, то решения совпадут.

Протестируем программы для n = 11 и изобразим решения:

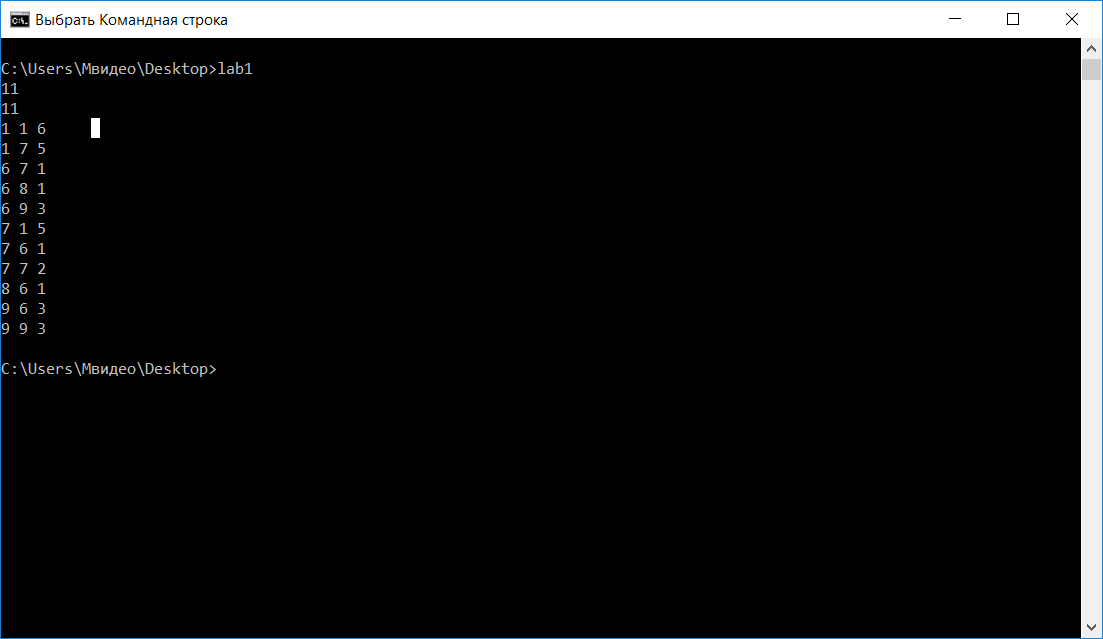


рис 11. Работа программы с упрощениями

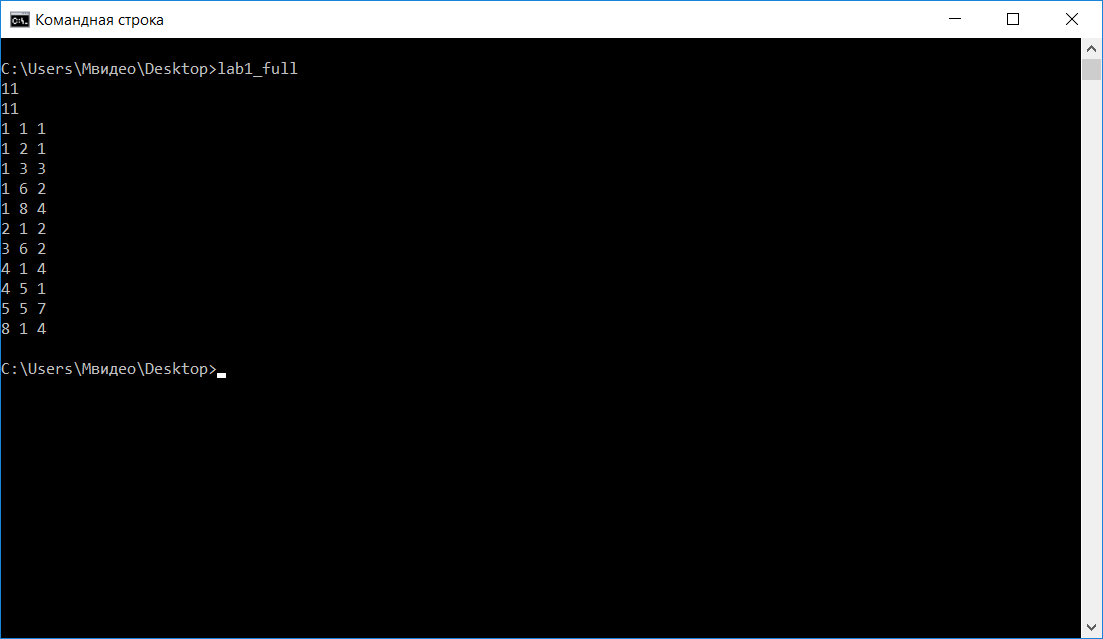


рис 12. Работа программы с полным перебором

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

рис 13. Изображение квадрата, разрезанного программой с упрощениями

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

рис 14. Изображение квадрата, разрезанного программой с полным перебором

Приходим к выводу о том, что используемые шаблоны, а также заполнение начальных трех квадратов в простых числах имеют место быть в программе для уменьшения времени её исполнения.