САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №0
по курсу «Алгоритмы и структуры данных»
Тема: Введение. Работа с файлами, тестирование
Вариант 4

Выполнил:
Андреюк Николай Ростиславович
К3141

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Ввод-вывод

Необходимо выполнить 4 следующих задачи:

1. Задача a + b. В данной задаче требуется вычислить сумму двух заданных чисел. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-109 \le a, b \le 109$. Выход: единственное целое число — результат сложения a + b.
2. Задача $a + b^2$. В данной задаче требуется вычислить значение $a + b^2$. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-109 \le a, b \le 109$. Выход: единственное целое число — результат сложения $a + b^2$.
3. Выполните задачу a + b с использованием файлов.
    - Имя входного файла: input.txt
    - Имя выходного файла: output.txt
    - Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-109 \le a, b \le 109$.
    - Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.
4. Выполните задачу $a+b^2$ с использованием файлов аналогично предыдущему пункту.

Листинг кода:

```
package lab0;

import java.io.*;
import java.util.Scanner;

public class task1 {
    public static void main(String[] args) throws
IOException {
        Runtime instance = Runtime.getRuntime();
          long usedMemory = instance.totalMemory() -
instance.freeMemory();
        int kb = 1024;
        long startTime = System.nanoTime();
```

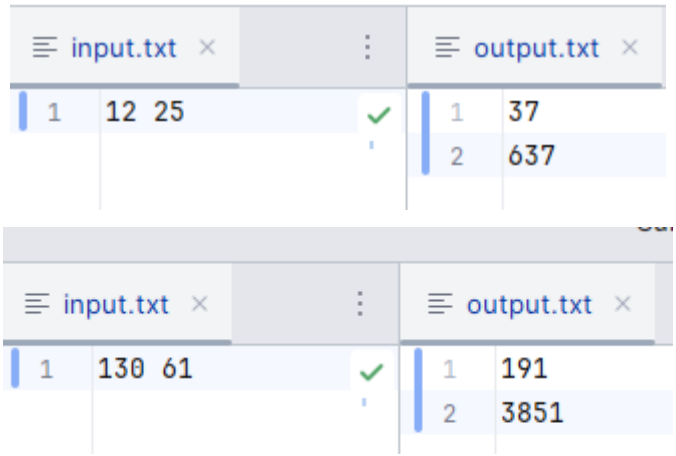
```
        Scanner scanner = new Scanner(System.in);

        long a, b;
                //long  a  =  scanner.nextLong(),  b  =
scanner.nextLong();
        //System.out.println(a + b);
        //System.out.println(a + b * b);

                                FileReader    fr    =    new
FileReader("src/lab0/input.txt");
        Scanner fscan = new Scanner(fr);
        a = fscan.nextLong();
        b = fscan.nextLong();
        fr.close();

FileWriter              fw              =              new
FileWriter("src/lab0/output.txt");
        fw.write((a + b) + "\n");
        fw.write(String.valueOf(a + b * b));
        fw.close();


         System.out.printf("Used memory: %,d (KB)\n",
usedMemory/kb);
          System.out.printf("Runtime:    %,.9f (sec)",
(System.nanoTime() - startTime)/Math.pow(10, 9));
    }
}
```

Описание решения:

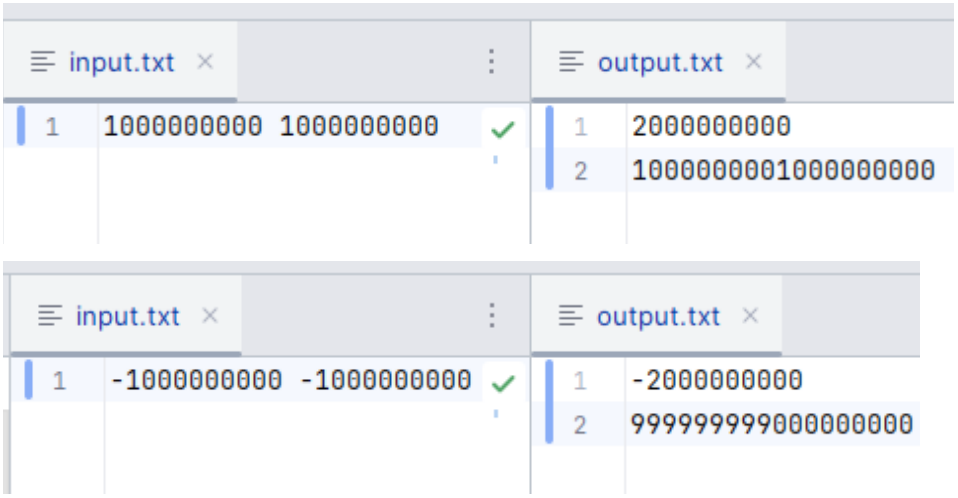
1. Создадим две переменные целочисленного типа для хранения значений a и b. Введем их через консоль и выведем результат их сложения.
2. Дополним алгоритм выводом результата сложения a и b^2. Заметим, что b^2 можно получить двумя способами: умножить число само на себя или использовать функцию из математической библиотеки java.lang.Math.pow(a, b)
3. Создадим объекты для сканирования и записи файлов. Сканируем два значения из файла input.txt и произведем вышеописанную операцию сложения. Запишем полученную сумму в файл output.txt.

4. Аналогично предыдущему пункту выполним сложение числа a и квадрата числа b.

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0,065247100 с	2 908 Кб
Пример из задачи	0,095769800 с	2 621 Кб
Пример из задачи	0,119819400 с	2 621 Кб
Верхняя граница диапазона значений входных данных из текста задачи	0,114055100 с	2.908 Кб

Вывод по задаче:
В результате выполнения задачи я вспомнил, как писать простейшие программы на ввод-вывод данных через терминал и через файл.

Задача №2. Число Фибоначчи

Разработать эффективный алгоритм для подсчета чисел Фибоначчи.

Листинг кода:

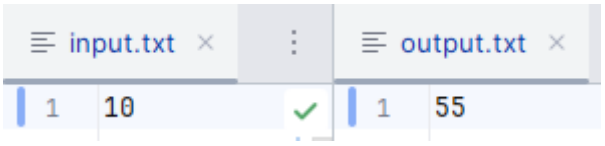
```
package lab0;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class task2 {
    public static void main(String[] args) throws
IOException{
            Runtime instance = Runtime.getRuntime();
             long usedMemory = instance.totalMemory()
- instance.freeMemory();
            int kb = 1024;

                              FileReader   fr   =   new
FileReader("src/lab0/input.txt");
            Scanner scanner = new Scanner(fr);
            int n = scanner.nextInt();
            fr.close();

                              FileWriter   fw   =   new
FileWriter("src/lab0/output.txt");

            if (n <= 0) {
                  System.out.println("please, put the
Natural number from 1 to 45 (there is no 0 in
Fibonacci sequence)");
            } else {
                long startTime = System.nanoTime();
```

```
fw.write(String.valueOf(fibonacci(n)));

                System.out.printf("Runtime:    %,.9f
(c)\n",  (System.nanoTime()  -  startTime)/Math.pow(10,
9));
                System.out.printf("Used memory: %,d
(KB)", usedMemory/kb);
            }
            fw.close();
        }

        public static long fibonacci(int n) {
            long curr = 1, prev = 1;
            for (int i = 0; i < n - 2; i++) {
                long t = curr;
                curr += prev;
                prev = t;
            }
            return curr;
        }
    }
```

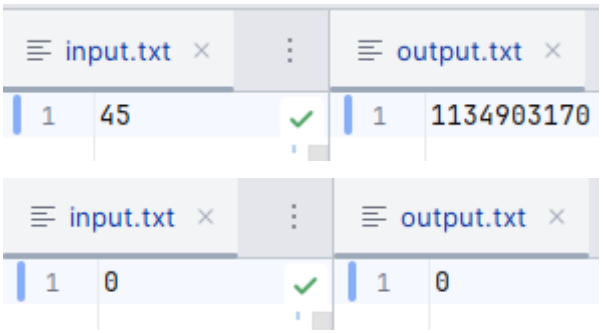
Решение:

Создадим функцию, которая на вход принимает номер числа в последовательности, и возвращает число в последовательности. Алгоритм поиска числа в последовательности заключается в циклическом сложении двух предыдущих чисел (`t` и `prev`) и получении нового числа (`curr`).

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0,091856400 (c)	2 908 (KB)
Пример из задачи	0,061605600 (c)	2 908 (KB)
Верхняя граница диапазона значений входных данных из текста задачи	0,103519800 (c)	2 908 (KB)

Вывод по задаче:

В результате выполнения задачи я узнал, как написать эффективный по памяти алгоритм для поиска чисел в последовательности Фибоначчи, понял, почему рекурсивный алгоритм занимает количество памяти пропорциональное

Задача №3. Еще про числа Фибоначчи

Определение последней цифры большого числа Фибоначчи.

Ограничение по времени: 5 сек.

Ограничение по памяти: 512 Мб

Листинг кода:

```
package lab0;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

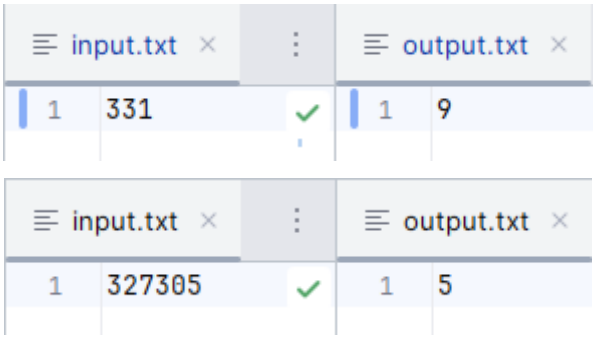
```
    import java.util.Scanner;
    import java.lang.Runtime;

    public class task3 {
       public static void main(String[] args) throws
IOException {
           long startTime = System.nanoTime();
           Runtime instance = Runtime.getRuntime();
                              long    totalMemory    =
instance.totalMemory();
           long freeMemory = instance.freeMemory();
                  long  usedMemory  =  totalMemory  -
freeMemory;
           long maxMemory = instance.maxMemory();
           int kb = 1024;

           //scan number
                           FileReader    fr    =    new
FileReader("src/lab0/input.txt");
           int n = new Scanner(fr).nextInt();
           fr.close();

           int prev = (int) (fibonacci(1) % 10);
           int curr = (int) (fibonacci(2) % 10);

           //keep going in cycle until n
           for (int i = 1; i < n-1; i++) {
               int t = curr;
               curr = (prev + curr) % 10;
               prev = t;
           }

                           FileWriter    fw    =    new
FileWriter("src/lab0/output.txt");
           fw.write(String.valueOf(curr));
           fw.close();

               System.out.printf("Total  memory:  %,d
(KB)\n", totalMemory/kb);
```

```
                System.out.printf("Free memory: %,d
(KB)\n", freeMemory/kb);
                System.out.printf("Used memory: %,d
(KB)\n", usedMemory/kb);
                 System.out.printf("Max memory: %,d
(KB)\n", maxMemory/kb);
                System.out.printf("Runtime:     %,.9f
(sec)", (System.nanoTime() - startTime)/Math.pow(10,
9));
        }

        public static long fibonacci(int n) {
                //разово (циклом) считаем наше число.
числа (n - k) не хранятся в памяти
            long curr = 1, prev = 1;
            for (int i = 0; i < n - 2; i++) {
                long t = curr;
                curr += prev;
                prev = t;
            }
            return curr;
        }
    }
```

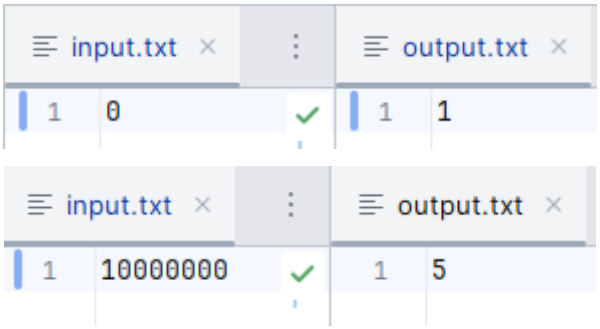
Решение задачи:

Решение отличается от предыдущей задачи только тем, что мы храним в переменных `curr`, `prev` и `t` значение только последней цифры в числе.

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0,064591100 (sec)	2 908 (KB)
Пример из задачи	0,091432700 (sec)	2 908 (KB)
Пример из задачи	0,074422300 (sec)	2 908 (KB)
Верхняя граница диапазона значений входных данных из текста задачи	0,130128900 (sec)	2 908 (KB)

Вывод по задаче:

Я вспомнил, как решать нестандартные проблемы нестандартными путями.

Задача №4. Тестирование ваших алгоритмов

Вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

Дополнительно: вы можете протестировать объем используемой памяти при выполнении вашего алгоритма.

Листинг кода:

```
package lab0;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

```
    import java.util.Scanner;
    import java.lang.Runtime;

    public class task3 {
       public static void main(String[] args) throws
IOException {
           long startTime = System.nanoTime();
           Runtime instance = Runtime.getRuntime();
                              long    totalMemory    =
instance.totalMemory();
           long freeMemory = instance.freeMemory();
                  long  usedMemory  =  totalMemory  -
freeMemory;
           long maxMemory = instance.maxMemory();
           int kb = 1024;

           //scan number
                              FileReader    fr    =    new
FileReader("src/lab0/input.txt");
           int n = new Scanner(fr).nextInt();
           fr.close();

           int prev = (int) (fibonacci(1) % 10);
           int curr = (int) (fibonacci(2) % 10);

           //keep going in cycle until n
           for (int i = 1; i < n-1; i++) {
               int t = curr;
               curr = (prev + curr) % 10;
               prev = t;
           }

                              FileWriter    fw    =    new
FileWriter("src/lab0/output.txt");
           fw.write(String.valueOf(curr));
           fw.close();

               System.out.printf("Total  memory:  %,d
(KB)\n", totalMemory/kb);
```

```
                System.out.printf("Free memory: %,d
(KB)\n", freeMemory/kb);
                System.out.printf("Used memory: %,d
(KB)\n", usedMemory/kb);
                 System.out.printf("Max memory: %,d
(KB)\n", maxMemory/kb);
                System.out.printf("Runtime:     %,.9f
(sec)", (System.nanoTime() - startTime)/Math.pow(10,
9));
        }

        public static long fibonacci(int n) {
                //разово (циклом) считаем наше число.
числа (n - k) не хранятся в памяти
            long curr = 1, prev = 1;
            for (int i = 0; i < n - 2; i++) {
                long t = curr;
                curr += prev;
                prev = t;
            }
            return curr;
        }
    }
```

Решение:

Создадим сущность в которой будет записываться информация о работе программы (`Runtime`). С помощью методов `totalMemory()` и `freeMemory()` мы можем узнать, сколько памяти использует JVM и сколько осталось свободной памяти. Вычтя одну величину из другой, мы можем узнать, сколько памяти использует программа.

Для отслеживания времени исполнения воспользуемся функцией `System.nanoTime()`. Запишем в переменную информацию о времени работы JVM в самом начале программы. Во время вывода информации о работе программы (в конце выполнения задач 1, 2 и 3) мы снова вызываем этот метод и получив разницу со временем старта, узнаем, сколько времени проработала наша программа.

Вывод

В результате выполнения данной лабораторной работы, я вспомнил, как писать простейшие программы на языке Java, написал эффективный алгоритм поиска чисел из последовательности Фибоначчи, а также поиска последней цифры в очень большом числе Фибоначчи. Также я научился использовать функции для анализа работы алгоритмов.