BÁO CÁO THỰC HÀNH

**Môn học: Lập trình ứng dụng mạng**

**Buổi báo cáo: Lab 01**

**Tên chủ đề:** Bài tập thực hành java cơ bản

*GVHD: Trần Mạnh Hùng*

*Ngày thực hiện: 19/03/2024*

**THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT109.O21.MMCL

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Văn Anh Tuấn | 21522757 | 21522757@gm.uit.edu.vn |
| 2 |  |  |  |
| 3 |  |  |  |

1. **ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình | 3 ngày |
| Link Video thực hiện  *(nếu có)* |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất … |  |
| Điểm tự đánh giá | 10/10 |

BÁO CÁO CHI TIẾT

Trước khi vào bài báo cáo, em xin giải thích qua thư mục **utils**(LAB01/src/utils).

Thư mục này dùng để chứa các hàm tính toán, input, output được sử dụng nhiều lần ở trong chương trình, class Array dùng cho bài 13 để quản lý mảng và class Matrix dùng cho bài 14 để quản lý ma trận.

**Bài 1: Viết chương trình tìm tất cả các số chia hết cho 7 nhưng không phải bội số của 5, nằm trong đoạn 10 và 200 (tính cả 10 và 200). Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng dấu phẩy.**

Giải thuật:

  List<Integer> list = new ArrayList<Integer>();

        for(int i = 10; i <= 200; i++){

            if(i%7==0 && i%5!=0){

                list.add(i);

            }

        }

*Hình 1.1*

Tạo một list kiểu số nguyên để chứa các phần tử tìm được, lặp qua khoảng 10 đến 200, nếu chia hết cho 7 và không chia hết cho 5 thì thêm vào list.

String listInt = list.stream().map(Object::toString).collect(Collectors.joining(", "));

        log.WriteLog("Bai1: " + listInt);

        log.ReadFile();

*Hình 1.2*

Sau khi đã có được list chứa dãy số cần tìm ta duyệt qua list, chuyển nó thành String và thêm dấu phẩy sau mỗi số.

Kết quả:



*Hình 1.3*

**Bài 2: Viết một chương trình tính giai thừa của một số nguyên dương n. Với n được nhập từ bàn phím. Ví dụ, n = 8 thì kết quả đầu ra phải là 1\*2\*3\*4\*5\*6\*7\*8 = 40320.**

Giải thuật:

int result = 1;

        for (int i = 1; i <= n; i++) {

            result \*= i;

        }

*Hình 2.1*

Sử dụng vòng lặp từ 1 đến n và nhân các phẩn tử với nhau, lưu kết quả vào biến result.

Kết quả:



*Hình 2.2*

**Bài 3: Hãy viết chương trình để tạo ra một map chứa (i, i\*i), trong đó i là số nguyên từ 1 đến n (bao gồm cả 1 và n), n được nhập từ bàn phím. Sau đó in map này ra màn hình. Ví dụ: Giả sử số n là 8 thì đầu ra sẽ là: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}.**

Giải thuật:

 int n = scanner.nextInt();

        Map<Integer, Integer> map = new HashMap<Integer, Integer>();

        for(int i = 1; i <= n; i++) {

            map.put(i, i\*i);

        }

*Hình 3.1*

Tạo 1 map với số phần tử là n được nhập từ bàn phím, duyệt qua từ 1, n trong mỗi vòng lặp: gán key = i , value = i\*i

Sau đó :

String str = map.toString().replace("=", ": ");

*Hình 3.2*

Map sẽ tự động hiển thị các cặp key và value với dấu “ = “ , để thay sang dấu “ : “ ta chỉ cần chuyển map sang kiểu String và sử dụng hàm replace().

Kết quả:



*Hình 3.3*

**Bài 4: Viết chương trình chuyển đổi một số tự nhiên ở hệ số 10 thành một số ở hệ cơ số B (1 <= B <= 32) bất kỳ. Giả sử hệ cơ số cần chuyển là 2 <= B <= 16. Số đại điện cho hệ cơ số B > 10 là A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.**

Giải thuật:

System.out.print("Nhap vao 1 so he co so 10: ");

        int n = scanner.nextInt();

        System.out.print("Nhap he co so b can chuyen: ");

        int b = scanner.nextInt();

        int tmp\_n = n;

        String result = "";

        while (tmp\_n > 0) {

            int soDu = tmp\_n % b;

            char chuSo;

            if (soDu >= 10) {

                chuSo = (char) (soDu + 55);

            } else {

                chuSo = (char) (soDu + '0');

            }

            result = (new StringBuilder()).append(chuSo).append(result).toString();;

            tmp\_n /= b;

        }

*Hình 4.1*

Đầu tiên, nhập vào một số ở hệ cơ số mười lưu vào **n** và nhập vào hệ cơ số muốn chuyển đổi lưu vào **b**. Sau đó, lặp cho tới khi **n <= 0**, trong mỗi bước lặp lấy **n** chia lấy dư cho **b**, nếu số dư **lớn hơn bằng 10** chuyển thành chữ cái tương ứng ( A, B, C…). Cuối cùng lưu kết quả vào biến **result** và in ra màn hình.

Kết quả:



*Hình 4.2*

**Bài 5: Dãy số Fibonacci được định nghĩa như sau: F0 = 0, F1 = 1, F2 = 1, Fn = F(n-1) + F(n-2) với n >= 2. Ví dụ: 0, 1, 1, 2, 3, 5, 8, ... Hãy viết chương trình tìm n số Fibonacci đầu tiên.**

Giải thuật:

  System.out.print("Nhap n:");

        int n = scanner.nextInt();

        int f0 = 0, f1 = 1;

        String result = "0, 1";

        for (int i = 2; i < n; i++) {

            int f = f0 + f1;

            result += ", " + f;

            f0 = f1;

            f1 = f;

        }

*Hình 5.1*

Sử dụng vòng lặp for bắt đầu từ 2 tới n. Trong mỗi lần lặp, tính toán số fibonacci tiếp theo bằng cách cộng 2 biến f0, f1 trong đó f0 là f(n-2) và f1 là f(n-1). Lưu số fibonacci tiếp theo vào biến f và Lưu kết quả vào biến result. Cuối cùng hoán vị f0 = f1 , f1 = f.

Kết quả:



*Hình 5.2*

**Bài 6: Viết chương trình tìm ước số chung lớn nhất (USCLN) và bội số chung nhỏ nhất (BSCNN) của hai số nguyên dương a và b nhập từ bàn phím.**

Giải thuật:

Hàm USCLN (Hình 6.1)

 public static int USCLN(int a, int b) {

        if (b == 0) return a;

        return USCLN(b, a % b);

    }

*Hình 6.1*

System.out.print("Nhap vao a,b:");

        int a = scanner.nextInt(), b = scanner.nextInt();

        int uscln = USCLN(a, b);

        int bscnn = (a \* b) / uscln;

*Hình 6.2*

Nhập vào 2 số a, b. Sử dụng hàm USCLN để tính ước số chung lớn nhất lưu vào biến uscln, sau đó tính bscnn bằng cách lấy a \* b tất cả chia cho uscln.

Kết quả:



*Hình 6.3*

**Bài 7: Viết chương trình liệt kê tất cả các số nguyên tố nhỏ hơn n. Số nguyên dương n được nhập từ bàn phím.**

Giải thuật:

Hàm isPrime()

public boolean isPrime(int prime) {

        if (prime < 2) {

            return false;

        }

        for (int i = 2; i <= Math.sqrt(prime); i++) {

            if (prime % i == 0) {

                return false;

            }

        }

        return true;

    }

*Hình 7.1*

System.out.print("Nhap n: ");

        int n = scanner.nextInt();

        String result = "";

        for (int i = 2; i < n; i++) {

            if ((new util()).isPrime(i)) {

                result += Integer.toString(i) + " ";

            }

        }

*Hình 7.2*

Nhập n từ bàn phím, sử dụng vòng lặp từ 2 tới n, nếu là số nguyên tố thì add vào biến result.

Kết quả:



*Hình 7.3*

**Bài 8: Viết chương trình liệt kê tất cả số nguyên tố có 5 chữ số trong java.**

Giải thuật:

Hàm isPrime()

 public boolean isPrime(int prime) {

        if (prime < 2) {

            return false;

        }

        for (int i = 2; i <= Math.sqrt(prime); i++) {

            if (prime % i == 0) {

                return false;

            }

        }

        return true;

    }

*Hình 8.1*

 String result = "";

        for (int i = 10000; i < 100000; i++) {

            if ((new util()).isPrime(i)) {

                result += (i==99991 ? Integer.toString(i) : (Integer.toString(i) + ", "));

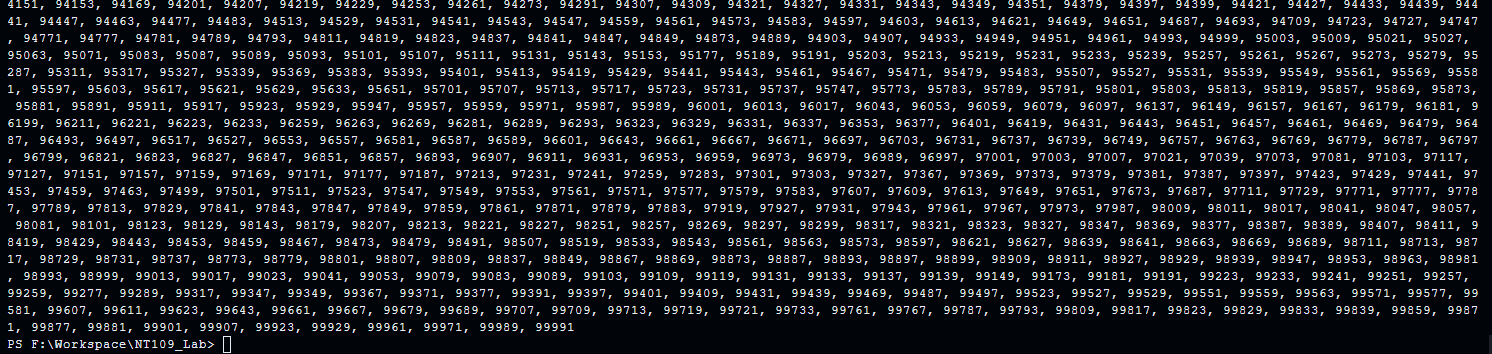
            }

        }

*Hình 8.2*

Duyệt từ 10000 tới 100000, nếu là số nguyên tố thì lưu kết quả vào biến result.

Kết quả:



*Hình 8.3*

**Bài 9: Viết chương trình phân tích số nguyên n thành các thừa số nguyên tố trong java. Ví dụ: 100 = 2x2x5x5.**

 System.out.print("Nhap n: ");

        int n = scanner.nextInt();

        int tmp\_n = n;

        String result = "";

        for (int i = 2; i <= tmp\_n; i++) {

            while (tmp\_n % i == 0) {

                result += (tmp\_n==i ? Integer.toString(i) : (Integer.toString(i) + "x"));

                tmp\_n /= i;

            }

        }

*Hình 9.1*

Nhập vào số n từ bàn phím, lặp từ 2 đến n. Trong mỗi vòng lặp, sử dụng 1 vòng while với điều kiện n chia hết cho i, nếu n % i thì đó là 1 thừa số nguyên tố của n, nếu không tiếp tục vòng for. Lưu kết quả vào biến result.

Kết quả:



*Hình 9.2*

**Bài 10: Nhập một sâu ký tự. Đếm số từ của sâu đó (mỗi từ cách nhau bởi một khoảng trắng có thể là một hoặc nhiều dấu cách, tab, xuống dòng). Ví dụ " hoc java co ban den nang cao " có 7 từ.**

Giải thuật:

Đầu tiên khởi tạo các hằng số chứa dấu cách, tab, xuống dòng.

private static final char SPACE = ' ';

    private static final char TAB = '\t';

    private static final char BREAK\_LINE = '\n';

*Hình 10.1*

System.out.print("Nhap chuoi: ");

        String input = scanner.nextLine();

        int count = 0;

        int size = input.length();

        boolean notCounted = true;

        for (int i = 0; i < size; i++) {

            if (input.charAt(i) != SPACE && input.charAt(i) != TAB

                    && input.charAt(i) != BREAK\_LINE) {

                if(notCounted) {

                    count++;

                    notCounted = false;

                }

            } else {

                notCounted = true;

            }

        }

*Hình 10.2*

Nhập vào chuỗi cần đếm lưu vào biến input. Tạo 1 biến count để đếm số từ. Sử dụng vòng lặp for lặp qua độ dài của input. Trong mỗi vòng lặp đếm số từ cho đến khi gặp lại các hằng số như dấu cách, tab, xuống dòng.

Kết quả:



*Hình 10.3*

**Bài 11: Nhập một sâu ký tự. Liệt kê số lần xuất hiện của các từ của sâu đó.**

Giải thuật:

Khởi tạo các hằng số chứa dấu cách, tab, xuống dòng, dấu phẩy.

private static final char SPACE = ' ';

    private static final char TAB = '\t';

    private static final char BREAK\_LINE = '\n';

    private static final char COMMA = ',';

*Hình 11.1*

Hàm addWord()

 public static void addWord(Map<String, Integer> wordMap, StringBuilder sb) {

        String word = sb.toString();

        if (word.length() == 0) {

            return;

        }

        if (wordMap.containsKey(word)) {

            int count = wordMap.get(word) + 1;

            wordMap.put(word, count);

        } else {

            wordMap.put(word, 1);

        }

    }

*Hình 11.2*

Kiểm tra xem từ đã có trong danh sách chưa, nếu có rồi thì count + 1, nếu chưa thì tạo thêm từ mới , key = word và value = 1.

Hàm countWord()

public static Map<String, Integer> countWords(String input) {

        // khởi tạo wordMap

        Map<String, Integer> wordMap = new TreeMap<String, Integer>();

        if (input == null) {

            return wordMap;

        }

        int size = input.length();

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < size; i++) {

            if (input.charAt(i) != SPACE && input.charAt(i) != TAB

                    && input.charAt(i) != BREAK\_LINE && input.charAt(i) != COMMA) {

                // build một từ

                sb.append(input.charAt(i));

            } else {

                // thêm từ vào wordMap

                addWord(wordMap, sb);

                sb = new StringBuilder();

            }

        }

        // thêm từ cuối cùng tìm được vào wordMap

        addWord(wordMap, sb);

        return wordMap;

    }

*Hình 11.3*

Khởi tạo 1 map<String, Int>, nhập vào chuỗi từ bàn phím, sử dụng vòng lặp for lặp qua độ dài của chuỗi, sử dụng StringBuilder để thêm từng chữ cái vào thành 1 từ cho tới khi gặp các hằng số như dấu cách, tab, xuống dòng , dấu phẩy.

Kết quả:



*Hình 11.4*

**Bài 12: Nhập 2 sâu ký tự s1 và s2. Kiểm tra xem sâu s1 có chứa s2 không?**

Giải thuật:

System.out.print("Nhap s1: ");

        String s1 = scanner.nextLine();

        System.out.print("Nhap s2: ");

        String s2 = scanner.nextLine();

        String result = s1.contains(s2) ? "s1 chua s2" : "s1 khong chua s2";

*Hình 12.1*

Nhập vào 2 chuỗi s1, s2. Sử dụng phương thức contains của lớp String để kiểm tra s1 có chứa s2 hay không.

Kết quả:



*Hình 12.2*

**Bài 13: Viết chương trình nhập vào mảng A có n phần tử, các phần tử là số nguyên lớn hơn 0 và nhỏ hơn 100. Thực hiện các chức năng sau:  
a) Tìm phần tử lớn thứ nhất và lớn thứ 2 trong mảng với các chỉ số của chúng (chỉ số đầu tiên tìm được).  
b) Sắp xếp mảng theo thứ tự tăng dần.  
c) Nhập số nguyên x và chèn x vào mảng A sao cho vẫn đảm bảo tính tăng dần cho mảng A.**

Giải thuật:

Sử dụng class Array đã tạo ở trong thư mục utils để quản lý mảng nhập vào.

Phương thức Input(): Nhập vào n phần tử từ bàn phím.

 public void Input() {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Nhap vao n: ");

        n = scanner.nextInt();

        arr = new ArrayList<>();

        for (int i = 0; i < n; i++) {

            System.out.print("Nhap vao phan tu thu " + i + ": ");

            arr.add(scanner.nextInt());

        }

        scanner.close();

    }

*Hình 13.1*

Phương thức Output(): Xuất mảng ra màn hình

 public String Output() {

        String result = "";

        for (int i = 0; i < arr.size(); i++) {

            result += arr.get(i) + " ";

        }

        return result;

    }

*Hình 13.2*

Phương thức Max(): Tìm số lớn nhất trong mảng

public int Max() {

        int max = arr.get(0);

        for (int i = 1; i < arr.size(); i++) {

            if (arr.get(i) > max) {

                max = arr.get(i);

            }

        }

        return max;

    }

*Hình 13.3*

Phương thức Min(): Tìm số nhỏ nhất trong mảng

public int Min() {

        int min = arr.get(0);

        for (int i = 1; i < arr.size(); i++) {

            if (arr.get(i) < min) {

                min = arr.get(i);

            }

        }

        return min;

    }

*Hình 13.4*

Phương thức SecondMax(): Tìm số lớn thứ 2 trong mảng

public int SecondMax() {

        int max = Max();

        int secondMax = Min();

        for (int i = 0; i < arr.size(); i++) {

            if (arr.get(i) > secondMax && arr.get(i) < max) {

                secondMax = arr.get(i);

            }

        }

        return secondMax;

    }

*Hình 13.5*

Phương thức Sort(): Sắp xếp các phần tử trong mảng theo thứ tự tăng dần

public void Sort() {

        Collections.sort(arr);

    }

*Hình 13.6*

Phương thức Addx(): Thêm X vào mảng và vẫn giữ nguyên thứ tự các phần tử tăng dần

public void Addx(int x){

        arr.add(x);

        n += 1;

        this.Sort();

    }

*Hình 13.7*

Hàm main() :

Array a = new Array();

        MyLog log = new MyLog("Lab01\\21522757.txt");

        a.Input();

        log.WriteLog("Bai13 Input: " + a.Output());

        log.WriteLog("Bai13 a) Max: " + a.Max() + ", Second max: " + a.SecondMax());

        a.Sort();

        log.WriteLog("Bai13 b) Sort: " + a.Output());

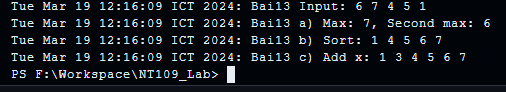
        a.Addx(3);

        log.WriteLog("Bai13 c) Add x: "+ a.Output());

        log.ReadFile();

*Hình 13.8*

Kết quả:



*Hình 13.9*

**Bài 14: Viết chương trình nhập vào ma trận A có n dòng, m cột, các phần tử là số nguyên lớn hơn 0 và nhỏ hơn 100. Thực hiện các chức năng sau:  
a) Tìm phần tử lớn thứ nhất với chỉ số của nó (chỉ số đầu tiên tìm được).  
b) Tìm và in ra các phần tử là số nguyên tố của ma trận (các phần tử không nguyên tố thì thay bằng số 0).  
c) Sắp xếp tất cả các cột của ma trận theo thứ tự tăng dần và in kết quả ra màn hình.  
d) Tìm cột trong ma trận có nhiều số nguyên tố nhất.**

Giải thuật:

Sử dụng class Matrix ở trong thư mục utils để quản lý ma trận\

Phương thức Input(): nhập vào ma trận m\*n từ bàn phím

 public void Input() {

        Scanner scanner = new Scanner(System.in);

        for (int i = 0; i < row; i++) {

            for (int j = 0; j < column; j++) {

                System.out.print("Nhap vao phan tu thu [" + i + "][" + j + "]: ");

                matrix[i][j] = scanner.nextInt();

            }

        }

        scanner.close();

    }

*Hình 14.1*

Phương thức getMatrix(): trả về ma trận

public int[][] getMatrix() {

        return matrix;

    }

*Hình 14.2*

Phương thức FindMax(): Tìm số lớn nhất trong ma trận và in ra vị trí của nó

public String FindMax(){

        int max = matrix[0][0];

        StringBuilder sb = new StringBuilder();

        int x = 0, y = 0;

        for (int i = 0; i < row; i++) {

            for (int j = 0; j < column; j++) {

                if(matrix[i][j] > max){

                    max = matrix[i][j];

                    x = i;

                    y = j;

                }

            }

        }

        sb.append("Max: " + max + " at [" + x + "][" + y + "]");

        return sb.toString();

    }

*Hình 14.3*

Phương thức FindPrime(): in ra ma trận số nguyên tố và thay thế những số không phải số nguyên tố = 0

public String FindPrime(){

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < row; i++) {

            for (int j = 0; j < column; j++) {

                if((new util()).isPrime(matrix[i][j])){

                    sb.append(matrix[i][j] + " ");

                } else {

                    sb.append("0 ");

                }

            }

            sb.append("\n");

        }

        return sb.toString();

    }

*Hình 14.4*

Phương thức SortColumn(): Sắp xếp cột theo thứ tự tăng dần giá trị

 public int[][] SortColumns(){

        for (int i = 0; i < column; i++) {

            int[] temp = new int[row];

            for (int j = 0; j < row; j++) {

                temp[j] = matrix[j][i];

            }

            for (int j = 0; j < row; j++) {

                for (int k = j + 1; k < row; k++) {

                    if(temp[j] > temp[k]){

                        int t = temp[j];

                        temp[j] = temp[k];

                        temp[k] = t;

                    }

                }

            }

            for (int j = 0; j < row; j++) {

                matrix[j][i] = temp[j];

            }

        }

        return matrix;

    }

*Hình 14.5*

Phương thức FindColumnWithMostPrime() : In ra cột có nhiều số nguyên tố nhất

public String FindColumnWithMostPrime(){

        StringBuilder sb = new StringBuilder();

        int[] count = new int[column];

        for (int i = 0; i < column; i++) {

            for (int j = 0; j < row; j++) {

                if((new util()).isPrime(matrix[j][i])){

                    count[i]++;

                }

            }

        }

        int max = count[0];

        int index = 0;

        for (int i = 1; i < column; i++) {

            if(count[i] > max){

                max = count[i];

                index = i;

            }

        }

        for(int i = 0; i < row; i++ ){

            sb.append(matrix[i][index]);

            sb.append("\n");

        }

        return sb.toString();

    }

*Hình 14.6*

Hàm main():

 MyLog log = new MyLog("Lab01\\21522757.txt");

        Matrix matrix = new Matrix(4, 4);

        matrix.Input();

        log.WriteLog("Bai 14 Input:\n" + (new util()).toString(matrix.getMatrix()));

        log.WriteLog("Bai 14 a) "+ matrix.FindMax());

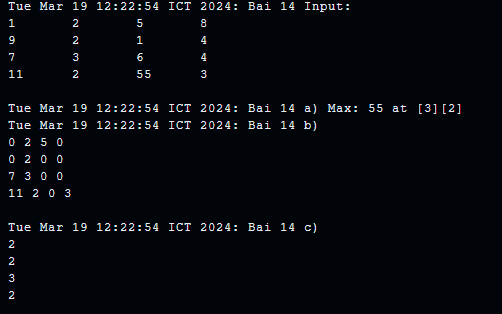
        log.WriteLog("Bai 14 b) \n"+ matrix.FindPrime());

        log.WriteLog("Bai 14 c) \n"+ matrix.FindColumnWithMostPrime());

        log.ReadFile();

*Hình 14.7*

Kết quả:



*Hình 14.8*

**Bài 15: Viết chương trình cho phép người dùng nhập vào 5 mã số, lưu 5 mã số này vào một mảng.**

* **Chương trình sẽ kiểm tra xem trong 5 mã số này có mã số nào sai định dạng hay không (định dạng là “00yLxxxx” với y là số nguyên từ 2 - 5, x là số nguyên từ 0-9).**
* **Nếu có bất kỳ mã số nào sai định dạng thì chương trình in ra “Sai rồi” rồi kết thúc chương trình, ngược lại thì in ra “Đúng rồi”. (Sử dụng biểu thức chính quy để ràng buộc định dạng)**

Giải thuật: Tạo mảng chứa 5 chuỗi nhập vào. Sử dụng vòng lặp for lặp qua các phần tử ở trong mảng, nếu phần tử không matches với regex pattern (“00[2-5]L[0-9]{4}) thì in ra “Sai roi!” nếu matches thì in ra “Dung roi!”.

MyLog log = new MyLog("Lab01\\21522757.txt");

        String[] arr = new String[5];

        (new util()).StringInput(arr);

        String result = "Dung roi!";

        log.WriteLog("Bai 15 Input: " + (new util()).StringOutput(arr));

        for (String s : arr) {

            if(!s.matches("00[2-5]L[0-9]{4}")){

                result = "Sai roi!";

                log.WriteLog("Bai 15 Output: " + result);

                log.ReadFile();

                return;

            }

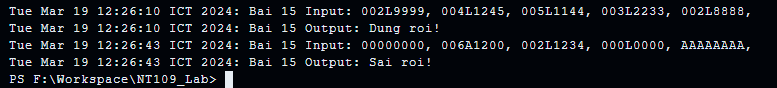
        }

*log.WriteLog("Bai 15 Output: " + result);*

        log.ReadFile();

*Hình 15.1*

Kết quả:



*Hình 15.2*

**HẾT**