Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Метрология стандартизация и сертификация в информационных технологиях (МСиСвИнфТ)

ОТЧЕТ

по лабораторной работе №3

Тема работы: Расчет метрики сложности потока данных

Выполнили

студенты: гр. 951007 Григорьева О.А.

Воривода М. А.

Проверил: Болтак С.В.

Минск 2020

**Исходный код**

window.onload = function (event) {  
 while (choose()) { }  
 close();  
 function choose() {   
 let a = *1*;  
 a = a \*\* *1*;  
 switch (prompt("Which lab are yo//u want to execute?").toLowerCase()) {  
 case "1.1":  
 lab1\_1();  
 return *true*;  
 case "2.3":  
 lab2\_3();  
 return *true*;  
 case "2.4":  
 lab2\_4();  
 return *true*;  
 case "3.1":  
 lab3\_1();  
 return *true*;  
 case "jilb":  
 jilb();  
 return *true*;  
 case "exit":  
 return *false*;  
 default:  
 return *true*;  
 }  
 }  
  
 function jilb() {  
 enum d {S = *6*, Y = *3*}; enum s {u = *6*};  
 for(let a = *0*; a < *10*; a++) {  
 if(a == *8*) continue;  
 for(let b = *0*; b < *10*; b++) {  
 for(let c = *0*; c < *10*; c++) {  
 for(let d = *0*; d < *10*; d++) {  
 for(let e = *0*; e < *10*; e++) {  
 for(let f = *0*; f < *10*; f++) {  
 switch(f) {  
 case *0*: alert("f = 0"); break;  
 case *2*: alert("f = 2"); break;  
 case *4*: alert("f = 4"); break;  
 case *6*:  
 case *8*: if(f == *6*) {  
 alert("f = 6");  
 } else {  
 alert("f = 8");  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
  
 function lab1\_1() {  
 let invalid: boolean = *true*;   
 let x1: number, x2: number;  
 let y1: number, y2: number;  
 let a = *0xFF*, b = -*0xbb*, c = -*0O142*, d = -*4.e6*, e = *1.5*, f = *1.1*, j = *1.*;  
   
 do {  
 try {  
 x1 = Number.parseInt(prompt("Enter x1:"));  
 x2 = Number.parseInt(prompt("Enter x2:"));  
 y1 = Number.parseInt(prompt("Enter y1:"));  
 y2 = Number.parseInt(prompt("Enter y2:"));  
 invalid = *false*;  
 } catch (e) {  
 alert("INPUT ERROR");  
 }  
 } while (invalid);  
   
 let r1: number = Math.sqrt(x1 \* x1 + y1 \* y1);  
 let r2: number = Math.sqrt(x2 \* x2 + y2 \* y2);  
 const r3 = 'true';  
  
 if (r1 == r2) {  
 alert("Distances to center is equals");  
 } else {  
 alert("Distances to center is not equals");  
 }  
 }   
   
 function lab2\_3() {  
 let n: number = *0*;  
 let invalid: boolean = *true*;  
  
 do {  
 try {  
 n = Number.parseInt(prompt("Enter matrix length (0 < n < 16)"));  
 if (n > *0* && n < *16*) {  
 invalid = *false*;  
 } else {  
 alert("Value is out of bounds");  
 }  
 } catch (e) {  
 alert("Try again");  
 }  
 } while (invalid);  
  
 let matrix: string[][] = new String[n][n];  
   
 for (let i = *0*; i < n; i++) {  
 invalid = *true*;  
 do {  
 try {  
 let temp: string[] = prompt("Enter matrix line:").split(" ", n);  
 for (let j = *0*; j < n; j++) {  
 matrix[i][j] = temp[j];  
 }  
 invalid = *false*;  
 } catch (e) {  
 alert("INPUT ERROR");  
 }  
 } while (invalid);  
 }  
 alert(isIdentity(matrix) ? "Matrix is identity" : "Matrix is not identity");  
  
 function splitToArr(str: string, count: number): string[] {  
 let line: string[] = new String[count];  
 let i: number = *0*;  
  
 while (str.indexOf(" ") > *0* && i < count) {  
 line[i] = str.substring(*0*, str.indexOf(" "));  
 str = str.replace(line[i] + " ", "");  
 i++;  
 }  
  
 line[i] = str;  
 return line;  
 }  
  
 function isIdentity(matrix: string[][]): boolean {  
 for (let row = *0*; row < matrix.length; row++) {  
 let col: number = *0*;  
 while (col < matrix[row].length) {  
 if (col == row) {  
 if (matrix[row][col] == "1") {  
 col++;  
 } else {  
 return *false*;  
 }  
 } else {  
 if (matrix[row][col] == "0") {  
 col++;  
 } else {  
 return *false*;  
 }  
 }  
 }  
 }  
 return *true*;  
 }  
 }  
  
 function lab2\_4() {  
 let a: number[] = inputArray();  
 let b: number[] = inputArray();  
  
 a = sort(a);  
 b = sort(b);  
 alert(arrToStr(sort(concatArray(a, b))));  
  
 function sort(arr: number[]): number[] {  
 for (let i = *0*; i < arr.length; i++) {  
 for (let j = *0*; j < arr.length; j++) {  
 if (arr[i] < arr[j]) {  
 let temp = arr[j];  
 arr[j] = arr[i];  
 arr[i] = temp;  
 }  
 }  
 }  
 return arr;  
 }  
  
 function arrToStr(arr: number[]): string {  
 let answer: string = "";  
 for (let i = *0*; i < arr.length; i++) {  
 answer = answer + arr[i] + " ";  
 }  
 return answer;  
 }  
  
 function concatArray(arrA: number[], arrB: number[]): number[] {  
 let c: number[] = new Number[arrA.length + arrB.length];  
  
 for (let i = *0*; i < arrA.length; i++) {  
 c[i] = arrA[i];  
 }  
 for (let i = arrA.length; i - arrA.length < arrB.length; i++) {  
 c[i] = arrB[i - arrA.length];  
 }  
 return c;  
 }  
  
 function inputArray(): number[] {  
 let invalid: boolean = *true*;  
 let arr: number[];  
 let n: number;  
 do {  
 try {  
 n = Number.parseInt(prompt("Enter array size"));  
 invalid = *false*;  
 } catch (e) {  
 alert("INPUT ERROR");  
 }  
 } while (invalid);  
  
 invalid = *true*;  
 do {  
 try {  
 arr = splitToInt(prompt("Enter array elements").trim(), n);  
 invalid = *false*;  
 } catch (e) {  
 alert("INPUT ERROR");  
 }  
 } while (invalid);  
  
 return arr;  
 }  
  
 function splitToInt(str: string, size: number): number[] {  
 let a: number[] = new Number[size];  
 let i: number = *0*;  
 while (str.indexOf(" ") > *0* && i < size) {  
 a[i] = Number.parseInt(str.substring(*0*,  
 str.indexOf(" ")));  
 str = str.replace(a[i] + " ", "");  
 i++;  
 }  
 a[i] = Number.parseInt(str);  
 return a;  
 }  
 }  
  
 function lab3\_1() {  
 let str1: string = inputString();  
 let str2: string = inputString();  
 let k: number = inputNumber();  
 alert(definition(str1, str2, k) + " - answer")  
  
 function inputString(): string {  
 return prompt("Enter string:");  
 }  
  
 function inputNumber(): number {  
 let invalid: boolean = *true*;  
 let k: number = *0*;  
 do {  
 k = Number.parseInt(prompt("Enter K:"));  
 invalid = *false*;  
 } while (invalid);  
 return k;  
 }  
  
 function definition(s1: string, s2: string, krit: number, ...daring): number {  
 let deletedLength: number = *0*;  
 alert(s1.indexOf(s2));  
 while (krit > *1* && s1.indexOf(s2) > -*1*) {  
 alert(s1 + " " + s2 + " " + krit);  
 deletedLength += s2.length + s1.indexOf(s2);  
 s1 = s1.replace(s1.substring(*0*, s1.indexOf(s2)) + s2, "");  
 krit--;  
 }  
 if (s1.indexOf(s2) < *0*) {  
 return *0*;  
 } else {  
  
 let userInfo: [string, number];  
 return s1.indexOf(s2) + deletedLength + *1*;  
 }  
 let h = *5*;  
 h %= *2*;  
 h ^= *2*;  
 if(h) h+=*2*;  
 }  
 }  
}

**Спен программы**

|  |  |
| --- | --- |
| **Идентификатор** | **Спен** |
| event | 0 |
| str | 13 |
| count | 2 |
| matrix | 7 |
| arr | 15 |
| arrA | 6 |
| arrB | 3 |
| size | 2 |
| s1 | 10 |
| s2 | 9 |
| krit | 3 |
| …daring | 0 |
| a | 16 |
| d | 4 |
| s | 0 |
| b | 7 |
| c | 7 |
| e | 8 |
| f | 5 |
| invalid | 17 |
| x1 | 3 |
| x2 | 3 |
| y1 | 3 |
| y2 | 3 |
| j | 11 |
| r1 | 1 |
| r2 | 1 |
| r3 | 0 |
| n | 11 |
| i | 35 |
| temp | 3 |
| row | 6 |
| col | 6 |
| answer | 3 |
| str1 | 1 |
| str2 | 1 |
| k | 4 |
| deletedLength | 2 |
| userInfo | 0 |
| h | 4 |
| line | 4 |
| Суммарный спен | 239 |

**Метрика Чепина программы**

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа переменных** | | | |
| **P** | **M** | **C** | **T** |
| x1 | answer | str | event |
| x2 | str1 | matrix | count |
| y1 | str2 | arrA | …daring |
| y2 | deletedLength | arrB | s |
| temp |  | size | r3 |
| k |  | s1 | userInfo |
|  |  | s2 | h |
|  |  | krit | line |
|  |  | a |  |
|  |  | d |  |
|  |  | b |  |
|  |  | c |  |
|  |  | e |  |
|  |  | f |  |
|  |  | invalid |  |
|  |  | j |  |
|  |  | r1 |  |
|  |  | r2 |  |
|  |  | i |  |
|  |  | row |  |
|  |  | col |  |
|  |  | n |  |
|  |  | arr |  |
| **6** | **4** | **23** | **8** |

**Q = a1\*P + a2\*M + a3\*C + a4\*T = 1 \* 6 + 2 \* 4 + 3 \* 23 + 0,5 \* 8 = 87**

**Метрика Чепина ввода/вывода**

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа переменных** | | | |
| **P** | **M** | **C** | **T** |
| x1 | answer | matrix |  |
| x2 | str1 | s1 |  |
| y1 | str2 | s2 |  |
| y2 | deletedLength | krit |  |
| temp |  | a |  |
| k |  | b |  |
|  |  | c |  |
|  |  | arr |  |
|  |  | n |  |
| **6** | **4** | **9** | **0** |

**Q = a1\*P + a2\*M + a3\*C + a4\*T = 1 \* 6 + 2 \* 4 + 3 \* 9 + 0,5 \* 0 = 41**

**Фрагмент исходного кода**

let deletedLength: number = *0*;  
let str = prompt("s1");  
let s1 = str;  
str = prompt("s2");  
let s2 = str;  
let krit = Number.parseInt(prompt("Krit:");  
alert(s1.indexOf(s2));  
while (krit > *1* && s1.indexOf(s2) > -*1*) {  
 alert(s1 + " " + s2 + " " + krit);  
 deletedLength += s2.length + s1.indexOf(s2);  
 s1 = s1.replace(s1.substring(*0*, s1.indexOf(s2)) + s2, "");  
 krit--;  
}  
if (s1.indexOf(s2) < *0*) {  
 alert(0);  
} else {  
 let userInfo: [string, number];  
 alert(s1.indexOf(s2) + deletedLength + *1*);  
}

**Спен программы**

|  |  |
| --- | --- |
| **Идентификатор** | **Спен** |
| str | 3 |
| s1 | 10 |
| s2 | 9 |
| krit | 3 |
| deletedLength | 2 |
| Суммарный спен | 27 |

**Метрика Чепина программы**

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа переменных** | | | |
| **P** | **M** | **C** | **T** |
| str | deletedLength | s1 | userInfo |
|  |  | s2 |  |
|  |  | krit |  |
| **1** | **1** | **3** | **1** |

**Q = a1\*P + a2\*M + a3\*C + a4\*T = 1 \* 1 + 2 \* 1 + 3 \* 3 + 0,5 \* 1 = 12,5**

**Метрика Чепина ввода/вывода**

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа переменных** | | | |
| **P** | **M** | **C** | **T** |
| str | deletedLength | s1 |  |
|  |  | s2 |  |
|  |  | krit |  |
| **1** | **1** | **3** | **0** |

**Q = a1\*P + a2\*M + a3\*C + a4\*T = 1 \* 1 + 2 \* 1 + 3 \* 3 + 0,5 \* 0 = 12**