*Logic Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | José González Ayerdi | **Program #** | 7 |

|  |  |
| --- | --- |
| **Class Name** | Prediccion |

|  |  |
| --- | --- |
| **Design** | OST de la página 1 |
| **References** | FST de la página 1 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Method Name** | Main |

|  |  |
| --- | --- |
| **Parameters** | args[] : string |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| Hacer: |
| Hacer: |
| Desplegar en pantalla: “Introduzca el nombre del archivo” |
| Si no se encontró el archivo desplegar: “No se encontró el archivo.” |
| Mientras no se encuentre el archivo |
| Leer archivo línea por línea |
| Guardar cada línea en una lista |
| Si ocurre un error al leer desplegar mensaje de tipo Excepción de I/O |
| Mientras el clasificador no detecte anomalías |
| betas = clasificador.reducir(matriz) |
| Desplegar N |
| Desplegar wk |
| Desplegar xk |
| Desplegar yk |
| Desplegar b0 |
| Desplegar b1 |
| Desplegar b2 |
| Desplegar b3 |
| Desplegar zk |
|  |

|  |  |
| --- | --- |
| **Method Name** | buscaArchivo |

|  |  |
| --- | --- |
| **Parameters** | Nombre : string |
|  |  |

|  |
| --- |
| Usar un buffer para leer archivos |
| Si ocurrió un error, entonces regresar false |
| Si sí se encontró el archivo, entonces regresar verdadero |
|  |

|  |  |
| --- | --- |
| **Class Name** | Clasificador |

|  |  |
| --- | --- |
| **Design** | OST de la página 1 |
| **References** | FST de la página 2 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Method Name** | clasificar |
| **Parameters** | Info : lista |
|  |  |
|  |  |

|  |
| --- |
| Si el tamaño de la lista es cero entonces desplegar en pantalla: “El archivo está vacío” y regresar falso. |
| aux es un arreglo = cada número individual separado por comas |
| Si el tamaño de aux es distinto de 3 entonces desplegar en pantalla: “El primer renglón debe contener 3 datos”, y regresar falso. |
| Llevar a cabo las asignaciones: |
| wk = Int(aux[0]) |
| xk = Int(aux[1]) |
| yk = Int(aux[2]) |
| Si ocurre un error entonces desplegar en pantalla: “Todos los valores deben de ser numéricos. Asegúrese también de que el formato de los números sea correcto.” Y regresar falso |
| Si xk < 0 ó wk < 0 ó yk < 0) entonces desplegar en pantalla: “Los primeros 3 datos deben ser mayores o iguales a cero”, y regresar falso. |
| Para todo i = 1 hasta n |
| aux = cada renglón a partir del segundo separado por coma |
| Si no se encuentran comas o bien hay menos de 4 elementos por línea entonces desplegar en pantalla: “El formato es incorrecto. El archivo debe contener cuádruplos a partir del segundo renglón, estos valores deben estar separados por comas” |
| Revisar que los 4 valores de aux sean número, si no lo son entonces desplegar en pantalla: “Todos los valores deben de ser numéricos. Asegúrese también de que el formato de los números sea correcto.” Y regresa falso |
| Si cualquiera de los valores de aux es menor que cero entonces desplegar en pantalla: “Uno o más datos son negativos, deben ser mayores o iguales a cero”, y regresa falso. |
| Añadir a las listas de w, x, y sus correspondientes valores a partir de aux, si ocurre algún error, desplegar en pantalla: “Todos los valores deben de ser numéricos. Asegúrese también de que el formato de los números sea correcto.” Y regresa falso |
| Termina ciclo |
| Regresa verdadero |
|  |

|  |  |
| --- | --- |
| **Method Name** | sum |
| **Parameters** | nums : lista |
|  |  |
|  |  |

|  |
| --- |
| resultado = 0 |
| Desde i = 0 hasta i < nums.size |
| resultado +=nums.get(i) |
| Terminar |
| Regresar restultado |
|  |
|  |

|  |  |
| --- | --- |
| **Method Name** | sum2 |
| **Parameters** | nums : lista |
|  |  |
|  |  |

|  |
| --- |
| Regresa la sumatoria desde i = 0 hasta lista.size de los pow(elementos, 2) de la lista |
|  |

|  |  |
| --- | --- |
| **Method Name** | sumMult |
| **Parameters** | nums1 : lista |
|  | nums2 : lista |
|  |  |

|  |
| --- |
| Regresa la sumatoria desde i = 0 hasta lista1.size de nums1(i) \* nums2(i) |
|  |

|  |  |
| --- | --- |
| **Method Name** | getMat |
| **Parameters** | Ninguno |
|  |  |
|  |  |

|  |
| --- |
| Arma una matriz con los siguientes elementos en orden: |
| getN(), sum(getW()), sum(getX()), sum(getY()), sum(getZ()), sum(getW()), sum2(getW()), sumMult(getW(), getX()), sumMult(getW(), getY()), sumMult(getW(), getZ()), sum(getX()), sumMult(getW(), getX()), sum2(getX()), sumMult(getX(), getY()), sumMult(getX(), getZ()), sum(getY()), sumMult(getW(), getY()), sumMult(getX(), getY()), sum2(getY()), sumMult(getY(), getZ()) |
| Regresa la matriz |
|  |

|  |  |
| --- | --- |
| **Class Name** | Gauss |

|  |  |
| --- | --- |
| **Design** | OST de la página 1 |
| **References** | FST de la página 3 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Method Name** | reducir |

|  |  |
| --- | --- |
| **Parameters** | matValores : double[][] |
|  |  |

|  |
| --- |
| Desde i = 0 hasta 3 |
| aux = matValores[i][i] |
| Desde j = 0 hasta j = 4 |
| Si abs(matvalores[j][i]) > abs(aux) |
| aux = matValores[j][k] |
| pos = j |
| Terminar |
| Si el valor más grande fue cero entonces desplegar en pantalla: “No hay solución.” |
| Si se encontró un número mayor (usar valor absoluto) en la columna entonces intercambiar los renglones en base a la variable posición. |
| divisor = matValores[i][i] |
| Para cada columna de la matriz |
| matValores[i][j] /= divisor |
| Terminar |
| Desde j = 1 hasta 4 – i |
| Factor = matValores[j + i][i] |
| Desde k = 0 hasta 5 – k |
| matValores[j + i][k + i] += matValores[i][k + i] \* -1 \* factor |
| Terminar |
| Terminar |
| Terminar |
| vars[0] = matValores[n - 1][n] / matValores[n - 1][n - 1] |
| Desde j = 2 hasta n |
| val = matValores[n - j][n] |
| Desde k = 0 hasta j – 1 |
| val -= matValores [n - j][n – k - 1] \* vars[k] |
| Terminar |
| vars[j - 1] = val / matValores[n - j][n - j] |
| Terminar |
| Desde j = 0 hasta n |
| betas[j] = vars[n – j - 1] |
| Regresar betas |