*Logic Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Erick Francisco González Martínez | **Program #** | 7 |

|  |  |
| --- | --- |
| **Class Name** | OutputHandler |

|  |  |
| --- | --- |
| **Method Name** | Double round |

|  |  |
| --- | --- |
| **Parameters** | Double number |

|  |  |  |
| --- | --- | --- |
| If number < 0 | | |
| Return ceil(number – 0.5) | | |
| Return floor(number + 0.5) | | |
| **Method Name** | | String convert |

|  |  |
| --- | --- |
| **Parameters** | Double number |

|  |
| --- |
| string s = to\_string ( round (x \* 100000.0) / 100000.0) |
| int length |
| while (s[s.length() - 1] == '0' && length < s.length()){ |
| length = s.find(".") + 6 |
| } |
|  |
| s.pop\_back(); |
| Return s |

|  |  |
| --- | --- |
| **Method Name** | Void OutputHandler::display |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| cout << "N = " << results[0] << endl; |
| cout << "wk = " << convert(results[1]) << endl; |
| cout << "xk = " << convert(results[2]) << endl; |
| cout << "yk = " << convert(results[3]) << endl; |
| cout<<"------------"<<endl; |
| cout << "b0 = " << convert(results[4]) << endl; |
| cout << "b1 = " << convert(results[5]) << endl; |
| cout << "b2 = " << convert(results[6]) << endl; |
| cout << "b3 = " << convert(results[7]) << endl; |
| cout<<"------------"<<endl; |
| cout << "zk = " << convert(results[8]) << endl; |

|  |  |
| --- | --- |
| **Method Name** | OutputHandler(vector<double> temp):results(temp  {}; |

|  |  |
| --- | --- |
| **Parameters** | vector<double> temp |

|  |
| --- |
| //Inline initialization of the vector |

|  |  |
| --- | --- |
| **Class Name** | InputReader |

|  |  |
| --- | --- |
| **Method Name** | InputReader::InputReader() |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| Results(vector<double>(9, 0)) |
| fileName("") |

|  |  |
| --- | --- |
| **Method Name** | InputReader::InputReader |

|  |  |
| --- | --- |
| **Parameters** | const InputReader &ir |

|  |  |  |
| --- | --- | --- |
| results = ir.results; | | |
| fileName = ir.fileName; | | |
| **Method Name** | | InputReader::handleInput |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| getline(cin, fileName); |

|  |  |
| --- | --- |
| **Method Name** | InputReader::openFile |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| file.open(fileName) |
|  |
| if (file.fail() || file.eof()) |
| file.close() |
| return false |
| End if |
| Return true |

|  |  |
| --- | --- |
| **Method Name** | InputReader::storeValues() |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| double x,y,z,w |
| vector<double> ks, input |
| string line; |
| stringstream ss |
|  |
| getline(file, line) |
| ss = (stringstream)line |
|  |
| for (double i; ss >> i;) { |
| ks.push\_back(i); |
| if (ss.peek() == ',') |
| ss.ignore() |
| } |
| wk = ks[0] |
| xk = ks[1] |
| yk = ks[2] |
|  |
| while (!file.eof()) |
| { |
| getline(file, line); |
|  |
| ss = (stringstream)line; |
|  |
| for (double i; ss >> i;) { |
| input.push\_back(i) |
| if (ss.peek() == ',') |
| ss.ignore() |
| } |
| Data += input; |
| Clear(input) |
| } |
|  |
| N = data.size() |
|  |
| if (N != 0) |
| for (int i = 0; i < N; i++) |
| { |
| wi += data[i][0] |
| xi += data[i][1] |
| yi += data[i][2] |
| zi += data[i][3] |
| w2i += data[i][0]\*\*2 |
| x2i += data[i][1]\*\*2 |
| y2i += data[i][2]\*\*2 |
| wx += data[i][0]\* data[i][1] |
| wy += data[i][0]\* data[i][2] |
| wz += data[i][0]\* data[i][3] |
| xy += data[i][1]\* data[i][2] |
| xz += data[i][1]\* data[i][3] |
| yz += data[i][2]\* data[i][3] |
| } |
| Mat = {{ N , wi , xi , yi , zi }, |
| { wi , w2i , wx , wy , wz }, |
| { xi , wx , x2i , xy , xz } |
| { yi , wy , xy , y2i , yz }} |
|  |
| calculateValues(); |

|  |  |
| --- | --- |
| **Method Name** | InputReader::rrf() |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| double pivot = 1; |
|  |
| for(int i =0 ; i< mat.size(); i++) |
| { |
| for(int k =0; k < mat.size();k++) |
| { |
| if(k != i) |
| { |
| for(int j =i+1; j < mat[i].size();j++) |
| { |
| mat[k][j] = (mat[i][i]\*mat[k][j] - mat[k][i]\*mat[i][j])/pivot |
| } |
| } |
| } |
| pivot = mat[i][i] |
| } |
|  |
| b0 = (mat[0][mat.size()])/pivot |
| b1 = (mat[1][mat.size()])/pivot |
| b2 = (mat[2][mat.size()])/pivot |
| b3 = (mat[3][mat.size()])/pivot |

|  |  |
| --- | --- |
| **Method Name** | InputReader::calculateValues() |

|  |  |
| --- | --- |
| **Parameters** |  |

|  |
| --- |
| rrf() |
| results[0] = N |
| results[1] = wk |
| results[2] = xk |
| results[3] = yk |
| results[4] = b0 |
| results[5] = b1 |
| results[6] = b2 |
| results[7] = b3 |
| results[8] = b0+wk\*b1+xk\*b2+yk\*b3 |