|  |  |  |
| --- | --- | --- |
| Draft Version | Author | Start Date & End Date |
| 1.0 | Adrian Harter | 11/24/24; 11/26/24 |

Main Program

Cout << *description of program*

PAUSE

Cin << Y (any key)

FUNCTION (checkWorkingDirectory) {

If ( < 1 time logs || >= 10 time logs)

PAUSE; Cout << *error description*

ELSE *continue*

}

FUNCTION (checkFilenameDuplicate) {

FOR (*filename : directory*)

*filename* 🡪 list

IF (*filename* > 1 in list)

PAUSE; Cout << *error description; include filenames*

ELSE *continue*

}

CHECK FOR VALIDITY

FUNCTION (checkInternalNames) {

FOR (*filename* : *directory*) {

FOR (*lastNamefirstName* : *filename*) {

*lastNamefirstName* 🡪 list

IF (*lastNamefirstName* > 1 in list)

PAUSE; Cout << *error description*; *include filenames*

ELSE *continue*

}

}

}

FUNCTION (checkClassID) {

FOR (*filename* : *directory*) {

FOR (*classID* : *filename*) {

*classID* 🡪 list

IF (*classID* != 1 in list)

PAUSE; Cout << *error description; include filenames, include classID*

ELSE *continue*

}

}

}

FUNCTION (createReport1) {

OPEN (*text file*)

WRITE << *title; classID; people (list); report description*

FOR (*numberOfNames* : *people*)

n++

INSERT *dataframe* 2 x 1n

*dataframe.*column = “names”, “total minutes logged”

*totalMinutes* (list) = FUNCTION (getTotalMinutes) {

FOR (*filename* : *directory*) {

FOR (*minutes* : *filename*)

*accumulate minutes; return in list*

}

}

FOR (*dataframe*.rows)

row = { *people* : *totalMinutes* }

}

FUNCTION (createReport2) {

OPEN (*text file*)

WRITE << *title; classID; people (list); report description*

WRITE << *activityCodes* (*list*)

FOR (*codenames* : *activityCodes*)

n++

INSERT *dataframe* 2 x 1n

*dataframe.*column = “activity code”, “minutes spent by all team members”

*minutesByMembers* (list) = FUNCTION (getMinutesByMembers) {

FOR (*filename* : *directory*) {

FOR (*minutes* : *filename*)

*accumulate minutes per activity code; return as list*

}

}

FOR (*dataframe.rows*)

row = { *activityCodes* : *minutesByMembers* }

}

FUNCTION (createReport3) {

OPEN (*text file*)

WRITE << *title; classID; people (list); report description*

WRITE << *activityCodes* (*list*)

FOR (*codenames* : *activityCodes)*

i++

FOR (*numberOfNames* : *people*)

j++

INSERT *dataframe* i x j

*dataframe.*column = “name”, ‘0’, ‘1’, ‘2’, …

*mapPersonToCodes* {key-values} = FUNCTION (personToCode) {

FOR (*filename* : *directory*) {

FOR (*codes* : *filename*)

*create map b/w person and all activity codes; return as key-value list*

}

}

FOR (*dataframe.*row)

row = { *people* : *mapPersonToCodes* }

}

FUNCTION (createReport4) {

OPEN (*text file*)

WRITE << *title; classID; people (list); report description*

INSERT *dataframe* 2 x 7

*Dataframe.*column = “Day of the Week”, “Total Team Minutes”

*daysOfWeekTotals* {key-values}= FUNCTION (daysOfWeekCounts) {

FOR (*filename* : *directory*) {

FOR (*days* : *filename*) {

VAR *counter*

*scan through each file, row by row. Get “data” data, and cluster by date + 7 so that each date + [0-6] has its own running total (i.e. d+1, d+2, d+3, etc). After d+6 is incremented, cycle back to d + 1*

IF ( *counter* < 6 && !*filename*.row)

EXIT

}

}

}

*daysOfWeekTotals* 🡪 “Monday”, “Tuesday”, “Wednesday” …

FOR (*dataframe*.row)

row = *daysOfWeekTotals*

}

FUNCTION (createReport5) {

OPEN (*text file*)

WRITE << *title; classID; people (list); report description*

FOR (*filename* : *directory*) {

FOR (*teamMeetingCode* : *filename*)

listA += *teamMeetingCode*

*sub-list each example of team meetings, following [date, time1, time2, activity code, meeting count] syntax.*

}

BUBBLE SORT listA

>> IF (listAdate1 > listAdate2) {

SWITCH

ELSE

*continue*

}

IF (listaAtime1 > listAtime1)

SWITCH

ELSE

*Continue*

} <<

FOR (*entry* : *listA*) {

IF (*entry.*time2 < (*entry +* 1).time1) {

REMOVE *entry*

IF ((*entry* +1).time2 < *entry*.time2)

REMOVE

}

}

listB = FUNCTION (meetingTimeTotals) {

FOR (*entry* : *listB*) {

IF (*entry*.time1 – (*entry* + 1).time1 < 0) {

a = || *entry*.time1 – (*entry* + 1).time1||

ELSE

a = *entry*.time1 – (*entry* + 1).time1

}

IF (*entry*.time2 – (*entry* + 1).time2 < 0) {

b = || *entry*.time1 – (*entry* + 1).time1||

ELSE

b = *entry*.time2 – (*entry* + 2).time1

}

IF (*entry*.time1 – (*entry* + 1).time1 > 0 && *entry*.time2 – (*entry* + 1).time2 < 0 ) {

c = b - a

ELIF (*entry*.time1 – (*entry* + 1).time1 < 0 && *entry*.time2 – (*entry* + 1).time2 < 0)

c = *entry*.time2 – (*entry* +1).time1

ELSE

c = (*entry* + 1).time2 – *entry*.time1

}

}

}

listC = { listA : listB}

FOR (listC[1]) {

IF (listC[1] > 2)

*compare minimum time of entry1 for date(x) to maximum time of entry for date(x). If max – min (in HH:MM) < time2 of entry(n – 1) within the bounds of +- 5 minute), where ‘n’ is the last entry for date(x), keep meeting entry*

ELSE

Cout << *error message describing discontinuous meeting*

}

Cout << listB.*sum*

}

Graph1

PAUSE

Cin << *any*

Graph2

PAUSE

Cin << *any*

Graph3

PAUSE

Cin << *any*

Cout << *closing message*