|  |
| --- |
| **Start Algorithm**  exit 🡨 0 |
| option 🡨 0 |
|  |
| while ( exit 🡨 0) do |
|  |
| string decision 🡨 0 |
| option 🡨 menu() |
| DoCase (option) |
|  |
| case 1 : addClass() |
| break |
| case 2 : removeClass() |
| break |
| case 3 : getClassRpt() |
| break |
| case 4 : mngClass() |
| break |
| case 5 : break |
| EndCase |
|  |
| Write("Would you like to terminate the program ? y/n ") |
| fflush(stdin) |
| Read (decision) |
|  |
| If ( decision 🡨 “y”) |
| exit 🡨 1 |
| else If (decision 🡨 “n”) |
| exit 🡨 0 |
| else |
| EndIf |
|  |
| Write("Sorry the input is not regonised, try using lower case, not upper case letters") |
| Write("Would you like to terminate the program ? y/n ") |
| fflush(stdin) |
| Read (decision) |
|  |
| If ( decision 🡨 “y”) |
| exit 🡨 1 |
| else If (decision 🡨 “n”) |
| exit 🡨 0 |
| EndIf |
| Write (NEWLINE) |
| EndWhile |
| **Stop Algorithm** |

|  |
| --- |
| **Start Algorithm : int menu()** |
| option 🡨 0 |
|  |
| Write("Please chose the number that corresponds with the actions you will like to commit :") |
| Write(NEWLINE) |
| Write("1) Add a class.") |
| Write(NEWLINE) |
| Write("2) Remove a class") |
| Write(NEWLINE) |
| Write("3) Get a class report") |
| Write(NEWLINE) |
| Write("4) Manage class") |
| Write(NEWLINE) |
| Write("5) Undecided") |
| Write(NEWLINE) |
|  |
|  |
| Write("What do wish to do : ") |
| Read(option) |
| return option |
| **Stop Algorithm** |
|  |
| **Start Algorithm: void addClass()** |
|  |
| string cName |
| string tName |
| string tPass |
|  |
| Write(NEWLINE) |
| Write("|\t|\t|\t|\t Add Class \t|\t|\t|\t|") |
| Write(NEWLINE) |
|  |
| Write("Please enter the class's name : ") |
|  |
| Read(cName) |
|  |
| Write("Please enter username : ") |
|  |
| Read(tName) |
|  |
| Write ("Please enter password for the class : ") |
|  |
| Read(tPass) |
|  |
| createClass(cName,tName,tPass) |
| **Stop Algorithm** |
|  |
| **Start Algorithm: void removeClass()** |
|  |
| string className |
| string tUsername |
| string tPassword |
|  |
| Write ("Please enter the name of the class you'd like to delete : ") |
|  |
| Read(className) |
|  |
| strcat(className,".txt") |
|  |
|  |
| OpenFile “className” for Read |
| If (!fp) then |
|  |
| Write("Class not found",className) |
| CloseFile “className” |
|  |
| EndIF |
|  |
| Write("Please enter username : ") |
|  |
| Read(tUsername) |
| Write("Please enter password for class : ") |
|  |
| Read(tPassword) |
| If (fp) then |
|  |
| string readUsername |
| string readPassword |
| Read(readUsername,readPassword) from “className” file |
| Close “className” file |
| If (strcmp(readPassword,tPassword) 🡨 0 and |
| strcmp(readUsername,tUsername) 🡨 0) then |
|  |
| status 🡨 remove(className) |
| Write ("Status :",status) |
| else |
| Write ("User name and/or password is incorrect") |
| else |
|  |
| Write ("File not found",className) |
| EndIf  EndIf  EndIf  EndIf  **Stop Algorithm** |
|  |
|  |
| **Start Algorithm: void getClassRpt()** |
|  |
| string tUsername |
| string tPassword |
| string className |
|  |
| Write (NEWLINE) |
| Write ("|\t|\t|\t|\t Class Report \t|\t|\t|\t|") |
| Write (NEWLINE) |
| Write (NEWLINE) |
|  |
| Write ("Please enter class you'd like to get report from : ") |
|  |
| Read(className) |
|  |
| strcat(className,".txt") |
| OpenFile className for Read |
|  |
| Write ("Please enter usename : ") |
|  |
| Read(tUsername) |
|  |
| Write ("Please enter password : ") |
|  |
| Read(tPassword) |
|  |
| bool isLogged 🡨 loginTeacher(tUsername,tPassword,fp) |
| struct Student students[100] |
| If (isLogged) then |
| x 🡨 0 |
| i 🡨 0 |
|  |
| string heighestAverageStudent |
| heighestAverage 🡨 0 |
|  |
| string heighestEngStudent |
| heighestEng 🡨 0 |
|  |
| string heighestPhysEdStudent |
| heighestPhyEd 🡨 0 |
|  |
| string heighestMathStudent |
| heighestMath 🡨 0 |
|  |
| string heighestInteSciStudent |
| heighestInteSci 🡨 0 |
|  |
| While (x ! 🡨 EOF) |
| struct Student student |
| x🡨Read(student.fullName,student.mathGrade,student.engGrade,student.phyEdGrade,student.inteSciGrade) from className file |
|  |
|  |
| strcpy(students[i].fullName,student.fullName) |
| students[i].mathGrade 🡨 student.mathGrade |
| students[i].engGrade 🡨 student.engGrade |
| students[i].phyEdGrade 🡨 student.phyEdGrade |
| students[i].inteSciGrade 🡨 student.inteSciGrade |
|  |
|  |
| If (students[i].mathGrade > heighestMath) then |
|  |
| heighestMath 🡨 students[i].mathGrade |
| strcpy(heighestMathStudent,students[i].fullName) |
| EndIf |
|  |
| If (students[i].engGrade > heighestEng) then |
|  |
| heighestEng 🡨 students[i].engGrade |
| strcpy(heighestEngStudent,students[i].fullName) |
| EndIf |
|  |
| If (students[i].phyEdGrade > heighestPhyEd) then |
|  |
| heighestPhyEd 🡨 students[i].phyEdGrade |
| strcpy(heighestPhysEdStudent,students[i].fullName) |
| EndIf |
|  |
| If (students[i].inteSciGrade > heighestInteSci) then |
|  |
| heighestInteSci 🡨 students[i].inteSciGrade |
| strcpy(heighestInteSciStudent,students[i].fullName) |
| EndIf |
| i++ |
| EndWhile |
|  |
| i 🡨 i-1 |
| j 🡨 0 |
| Write("Student Name/Math Average Grade/English Average Grade/Integrated Science Average Grade/Physical Education Average Grade/Overall Average) |
| While ( j < i) |
|  |
| averageGrade 🡨 (students[j].mathGrade + students[j].engGrade + students[j].inteSciGrade + students[j].phyEdGrade)/4  EndWhile |
| Write(students[j].fullName,students[j].mathGrade,students[j].engGrade,students[j].inteSciGrade,students[j].phyEdGrade,averageGrade) |
| j++ |
| EndWhile |
|  |
| Write("Heighest average in mathematics ",heighestMath,heighestMathStudent) |
| Write("Heighest average in english",heighestEng,heighestEngStudent) |
| Write("Heighest average in integrated science  ",heighestInteSci,heighestInteSciStudent) |
| Write("Heighest average in physical education ",heighestPhyEd,heighestPhysEdStudent) |
|  |
| EndIf |
|  |
| **Stop Algorithm** |
|  |
| **Start Algorithm: void mngClass()** |
|  |
| option 🡨 0 |
|  |
| Write (NEWLINE) |
| Write ("|\t|\t|\t|\t Manage Class \t|\t|\t|\t|") |
| Write (NEWLINE) |
| Write (NEWLINE) |
|  |
| Write ("Please chose the number that corresponds with the actions you will like to commit :") |
| Write (NEWLINE) |
| Write (NEWLINE) |
|  |
| Write ("1) Add Student to a class.") |
| Write (NEWLINE) |
|  |
| Write ("2) Remove Student from a class.") |
| Write (NEWLINE) |
|  |
| Write ("What would like to do : ") |
| Read(option) |
|  |
| If (option 🡨 1) then |
| addStudents\_Abstract() |
| else If(option 🡨 2) |
| removeStudent() |
| else |
|  |
| Write ("The value you entered is not recognised") |
| Write (NEWLINE) |
| return |
| EndIf |
| |  | | --- | |  | | **Start Algorithm: void createClass( string cName, string tName , string tPass )** | |  | | FILE \*ptr | |  | | string \*fileExtension 🡨 ".txt" | |  | | strcat(cName,fileExtension) | | OpenFile cName for Write | |  | |  | | If (ptr) then | |  | | Write(tName,tPass) to “cName” file | | addStudents(ptr) | | Write("Class Created successfully") | | CloseFile “cName” | | else | |  | | Write (NEWLINE) | | perror("Error") | | Write (NEWLINE) | | exit(1) | | EndIf | | **Stop Algorithm** | |  | | **Start Algorithm: bool loginTeacher( string tName , string tPass ,FILE \*fp)** | |  | |  | | If (fp) then | |  | | string readUsername | | string readPassword | | Read(readUsername,readPassword) from “className” file | |  | | If ( readPassword = tPass AND | | ReadUsername = tName ) then | | return true | | else | |  | | Write ("User name and/or password is incorrect") | | return false | |  | | else | |  | | return false | | EndIf | | EndIf  EndIf | | EndIf  **Stop Algorithm** | |  | | **Start Algorithm: void addStudents(FILE \*fp)** | |  | | size 🡨 0 | |  | | Write ("Please enter the amount of students that are in the class :") | | Read(size) | | struct Student students[size] | | For (i 🡨 0 i < size) do | |  | | x 🡨 i + 1 | | Write ("Please enter student full name : ",x) | |  | | Read(students[i].fullName) | |  | | Write ("Please enter student Mathematics average grade : ",x) | |  | | Read (students[i].mathGrade) | |  | | Write ("Please enter student English average grade : ",x) | |  | | Read (students[i].engGrade) | |  | | Write ("Please enter student Physical Education average grade : ",x) | |  | | Read (students[i].phyEdGrade) | |  | | Write ("Please enter student Integrated Science average grade : ",x) | |  | | Read (students[i].inteSciGrade) | | EndFor | |  | | For ( i 🡨 0 i < size) do | | Write(fp,students[i].fullName,students[i].mathGrade,students[i].engGrade,students[i].phyEdGrade,students[i].inteSciGrade) to “className” file | | EndFor | | **Stop Algorithm** | |  | | **Start Algorithm: void addStudents\_Abstract()** | |  | | string tUsername | | string tPass | | string className | |  | | Write ("Please enter the class you'd like to add the student to : ") | |  | | Read(className) | |  | | strcat(className,".txt") | | OpenFile “className” for Read | |  | | Write ("Please enter username : ") | |  | | Read(tUsername) | |  | | Write ("Please enter password : ") | |  | | Read(tPass) | |  | | bool isLogged 🡨 loginTeacher(tUsername,tPass,fp) | |  | | OpenFile “className” for Appending | |  | | If (isLogged) then   |  |  | | --- | --- | |  | Write("Please enter the amount of students you would like to add.") | |  | numStud 🡨 0 | |  | fflush(stdin) | |  | Read (numStud) | |  |  | |  | struct Student students[numStud] | | |  | | For (i 🡨 0 i < numStud) do | | a 🡨 i + 1  Write ("Please enter student full name : ") | |  | | Read(student.fullName) | |  | | Write ("Please enter student Mathematics average grade : ") | |  | | Read (student.mathGrade) | |  | | Write ("Please enter student English average grade : ") | |  | | Read (student.engGrade) | |  | | Write ("Please enter student Physical Education average grade : ") | |  | | Read (student.phyEdGrade) | |  | | Write ("Please enter student Integrated Science average grade : ") | |  | | Read (student.inteSciGrade) | | WriteFile(fp,student.fullName,student.mathGrade,student.engGrade,student.phyEdGrade,student.inteSciGrade) | | Write ("Student added successfully")    EndFor  CloseFile “className” | |  | |  | | EndIf | | **Stop Algorithm** | |  | | **Start Algorithm: void removeStudent()** | |  | | string tUsername | | string tPass | | string className | | string studentName | |  | | Write ("Please enter the class you'd like to add the student to : ") | |  | | Read(className) | |  | | strcat(className,".txt"); // add .txt to the file extension | | OpenFile “className” Read | |  | | Write ("Please enter username : ") | |  | | Read(tUsername) | |  | | Write ("Please enter password : ") | |  | | Read(tPass) | |  | | Write ("Please enter student that you'd like to remove : ") | |  | | Read(studentName) | |  | | struct Student students[100] | |  | | bool isLogged 🡨 loginTeacher(tUsername,tPass,fp) | | If (isLogged) then | |  | | x 🡨 1 | | i 🡨 0 | |  | | struct Student student | |  | | while (x ! 🡨 EOF) | | x🡨Write(fp,student.fullName,&student.mathGrade,&student.engGrade,&student.phyEdGrade,&student.inteSciGrade) to “className” file | | Write ("Name : i 🡨", student.fullName, i) | |  | | strcpy(students[i].fullName,student.fullName) | | students[i].mathGrade 🡨 student.mathGrade | | students[i].engGrade 🡨 student.engGrade | | students[i].phyEdGrade 🡨 student.phyEdGrade | | students[i].inteSciGrade 🡨 student.inteSciGrade | | i++ | | EndWhile | |  | |  | | CloseFile “className” | | remove(className) | | j 🡨 i | |  | | OpenFile “className” for Write | | Write (tUsername,tPass) to “cName” file | |  | | length 🡨 sizeof(students) / sizeof(students[0]) | | Write (" j :",j) | |  | | For ( i 🡨 0; i < length) do | |  | | If ( strcmp(studentName,students[i].fullName) 🡨 0) | | continue | | else If ( i 🡨 j) | | break | | Write(ptr,,students[i].fullName,students[i].mathGrade,students[i].engGrade,students[i].phyEdGrade,students[i].inteSciGrade) to cName file | | EndFor | |  | | EndIf | | **Stop Algorithm** | |