Student ID: 2004873 Student Name: Lee Chen Hong

Process/Function Name: **Venue Information Module -**

**addVenue,searchVenue,modifyVenue,displayVenue,readVenueInform**

Process Description: **To store,add ,edit and search about**

**venue information for student or staff TARUC**

|  |
| --- |
| **Structure Declaration** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Variable Name** | **Data Type** | **Value/Data Element** | **Structure Type** | **Description** |
| **1** | day | Integer | - | Date | To handle for displaying date for user |
| **2** | Month | Integer | - | Date | To handle for displaying month for user |
| **3** | year | integer | - | Date | To handle for displaying year for user |
|  |  |  |  |  |  |
| **1** | venueID | Character | 10 | VenueInform | To store venue ID for each venue. |
| **2** | venueDes | Character | 20 | VenueInform | To store venue full name of each venue. |
| **3** | venueCategory | Character | 3 | VenuInform | To store venue category by using code. |
| **4** | venueMaxVisitor | Integer | - | VenueInform | To store the maximum number visitor for each venue. |
| **7** | venueChecker | Character | 20 | VenueInform | To store the staff that will check the SOP and the program. |
| **8** | venueStatus | Character | 1 | VenueInform | To store the status of venue (Safe/Closed/Danger) |
| **9** | operationStartTimeHrs | Integer | - | VenueInform | To store the venue opening time in Hour |
| **10** | operationStartTimeMin | Integer | - | VenueInform | To store the venue opening time in Minute |
| **11** | opeationEndTimeHrs | Integer | - | VenueInform | To store the venue closing time in Hour |
| **12** | operationEndTimeMin | Integer | - | VenueInform | To store the venue closing time in Minute |

|  |
| --- |
| **Local Variable Declaration** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Variable Name** | **Data Type** | **Value/Data Element** | **Description** |
|  | count1 | Integer | - | As a counter for array to store data. |
|  | action | Integer | - | To make decision. |
|  | errorCounter | Integer | - | As a error counter for password entering. |
|  | password | Character | 8 | Default//Correct password that pre-set before |
|  | userEnterPassword | Character | 8 | User input for password when using the program. |
|  | result | Integer | - | Result of comparing user input with correct password. |
|  |  |  |  |  |
|  | tempVenueDes | Character | 20 | Temporary Venue Name when adding new venue. |
|  | tempVenueStatus | Character | 1 | Temporary Venue Status when adding new venue. |
|  | tempVenueChecker | Character | 20 | Temporary Venue Checker Name when adding new venue. |
|  | tempVenueCategory | Character | 3 | Temporary Venue Category Code when adding new venue. |
|  | tempStartHrs | Integer | - | Temporary Venue start operation in hour when adding new venue. |
|  | tempStartMin | Integer | - | Temporary Venue start operation in minute when adding new venue. |
|  | tempEndHrs | Integer | - | Temporary Venue end operation in hour when adding new venue. |
|  | tempEndMin | Integer | - | Temporary Venue end operation in minute when adding new venue. |
|  | tempMaxVisitor | Integer | - | Temporary Venue maximum visitor can visit when adding new venue. |
|  | opt1 | Character | 1 | To make decision |
|  | opt2 | Character | 1 | To make decision |
|  |  |  |  |  |
|  | selection | Integer | - | To make decision |
|  | index | Integer | - | As a counter for “For Loop”. |
|  | resultCounter | Integer | - | As a counter for number of results found in the data. |
|  | searchVenueId | Integer | - | The user input for searching same venue id in data. |
|  | selec | Character | 1 | To make decision |
|  | searchStatus | Character | 1 | The user input for searching same venue status in data. |
|  | searchCategory | Character | 3 | The user input for searching same venue category code in data. |
|  |  |  |  |  |
|  | selectedVenueData | Integer | - | The user selected type of venue data that need to edit. |
|  | selectedVenue | Integer | - | The user selected which venue that need to edit. |
|  | counting | Integer | - | As a counter for “For Loop”. |
|  | counting2 | Integer | - | As a counter for “For Loop”. |
|  | choice | Integer | - | To make decision |
|  | editVenueDesc | Character | 20 | The user input for editing venue name |
|  | editVenueChecker | Character | 20 | The user input for editing venue checker name |
|  | editVenueStatus | Character | 1 | The user input for editing venue status |
|  | editVenueCategory | Character | 3 | The user input for editing venue category code |
|  | editMaxVisitor | Integer | - | The user input for editing venue number of maximum visitor can visit |
|  | editVenueHrs | Integer | - | The user input for editing venue operation in hrs |
|  | editVenueMin | Integer | - | The user input for editing venue operation in minute |
|  |  |  |  |  |
|  | place | Integer | - | As a counter for “For Loop”. |

|  |
| --- |
| **Function Declaration** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Variable Name** | **Data Type** | **Value/Data Element** | **Description** |
| 1 | addVenue | void |  | To process the adding function for new venue information. |
| 2 | searchVenue | void |  | To process the searching function to get targeting venue information by user requirement. |
| 3 | modifyVenue | void |  | To process the modifying function such as editing or deleing venue information. |
| 4 | displayVenue | void |  | To process the displaying function of all the venue information that current available. |
| 5 | readVenueInform | void |  | To process the reading information section from text file. |

**Program Code**

#include <stdlib.h>

#include <stdio.h>

#include<string.h>

#include<Windows.h>

#pragma warning (disable :4996)

SYSTEMTIME t;

//Decleration Of Functions

void addVenue();

void searchVenue();

void modifyVenue();

void displayVenue();

void venueMenu();

void readVenueInform(count);

static int venueTotalCount = 10; //As a counter for total number of venue

//Decleration Of structure

struct VenueInfo {

int venueId;

char venueDes[20];

char venueCategory[3];

int venueMaxVisior;

char venueChecker[20];

char venueStatus;

int operationStartTimeHrs;

int operationStartTimeMin;

int operationEndTimeHrs;

int operationEndTimeMin;

}venueInform[15];

//Main Function

void main() {

GetLocalTime(&t);

printf("\t\tYou are Visiting Venue Information Section at %d-%d-%d %d:%.2d Hour\n", t.wDay, t.wMonth, t.wYear, t.wHour, t.wMinute);

venueMenu();

}

//Venue Function

void venueMenu(){

int count1 = 0; //As a counter for reading from Text File

int action; //FOR MAKING DECISION

int errorCounter=0; //As a error counter for password entering

char password[] = "Admin123",userEnterPassword[8]; //Password variable

int result;

//Reading data from text file and store in array

readVenueInform(count1);

printf("\n\t\t================================ Venue Information =========================================\n");

//VENUE MENU FOR USERS//

do {

printf("\t\t--Venue Menu--\n");

printf("\tPress 1 :Add new Venue Information\n");

printf("\tPress 2 :Search Venue\n");

printf("\tPress 3 :Modify/EditVenue Venue Information [Worker/Management Use Only] \n");

printf("\tPress 4 :Generate all venue Information\n");

printf("\tPress 0 :Back To Main Menu\n");

printf("\t----------------------------\n");

//GET DECISION FROM USER//

printf("[Please Enter your choice and enter]:");

scanf("%d", &action);

//DECISION//

switch (action) {

case 1: addVenue(); break;

case 2: searchVenue(); break;

case 3:

do {

printf("\n[Please Enter the password (Press 0 to go back)]:"); //Password:Admin123

rewind(stdin);

scanf("%s", &userEnterPassword);

if (userEnterPassword == '0') {

venueMenu();

}

//To check the password

result = strcmp(&password, &userEnterPassword);

printf("\n\t\tLoading...\n");

//3 Times of entering password wrongly will bring to end of the program

if (result == 0) {

printf("\n\t\t---Access Completed---\n");

modifyVenue();

}

else{

printf("\n\t\t[Password Error,Please Try Again]\n");

errorCounter++;

}

} while (errorCounter<3);

if (errorCounter >= 3) {

printf("\n\t\t[Access Denined]\n");

system("pause");

break;

}

case 4: displayVenue(); break;

case 0: main();

default:printf("\nError,Invalid Input!\n");

}

} while (action < 0 || action > 5); //TO AVOID INVALID DATA GO THOUGH

}

//ADDING FUNCTION FOR VENUE INFO//

void addVenue() {

char tempVenueDes, tempVenueStatus,tempVenueChecker,tempCategory;

int tempStartHrs, tempStartMin, tempEndHrs, tempEndMin;

int tempMaxVisitor;

char opt1,opt2;

//INITIAL FILE POINTER//

FILE\* venuefPtr;

//OPEN RELATED TEXT FILE//

venuefPtr = fopen("VenueInformation.txt", "a");

//CHECKING FILE OPEN ERROR//

if (venuefPtr == NULL) {

printf("File Open Error\nPlease contact management\n");

exit(-1);

}

printf("\t\t================================ ADD VENUE SECTION ================================\n");

do {

printf("\t\t----------- New Adding Section ----------\n");

//Entering NEW VENUE NAME

printf("Enter new venue name(Press '-' to Go Back):");

rewind(stdin);

scanf("%s", &tempVenueDes);

if (tempVenueDes == '-') {

fclose(venuefPtr);

venueMenu();

}

//NEW VEUNE CATEGORY CODE

printf("Enter new venue category[Exp:Sport -> SPT](Press '-' to Go Back):");

rewind(stdin);

scanf("%s", &tempCategory);

if (tempCategory =='-'){

fclose(venuefPtr);

venueMenu();

}

//VENUE MAXIMUM VISITOR

do {

printf("Enter the Venue Maximum Visitor for this venue(Press '-' to Go Back):");

scanf("%d", &tempMaxVisitor);

if (tempMaxVisitor == 34) {

fclose(venuefPtr);

venueMenu();

}

else {

if (tempMaxVisitor <= -2) {

printf("\n\t-- Error Data Input Please Try Again --\n");

}

}

} while (tempMaxVisitor < -2);

//VENUE STATUS

do {

printf("Enter the Venue Status [Opened-> O/Closed-> X] (Press '-' to Go Back):");

rewind(stdin);

scanf("%c", &tempVenueStatus);

tempVenueStatus = toupper(tempVenueStatus);

if (tempVenueStatus == '-') {

fclose(venuefPtr);

venueMenu();

}

else {

if (tempVenueStatus != 'X' && tempVenueStatus != 'O') {

printf("\n\t-- Error Data Input Please Try Again --\n");

}

}

} while (tempVenueStatus != 'X' && tempVenueStatus != 'O');

//VENUE CHECKER

printf("Enter the venue checker(Enter ""---"" for default name):");

scanf("%s", &tempVenueChecker);

//VENUE OPERATION TIME

do {

printf("Enter the venue Operation Time:\n");

printf("Start From (Hrs):");

scanf("%d", &tempStartHrs);

printf(" (Min):");

scanf("%d", &tempStartMin);

printf("To (Hrs):");

scanf("%d", &tempEndHrs);

printf(" (Min):");

scanf("%d", &tempEndMin);

if (tempEndHrs >= 24 || tempStartMin >= 60 || tempEndHrs >= 24 || tempEndMin >= 60) {

printf("Invalid Data input,Please Try Again\n");

}

} while (tempEndHrs >= 24 || tempStartMin >= 60 || tempEndHrs >= 24 || tempEndMin >= 60);

//FINAL CHECKING BEFORE STORE AND RECORD THE NEW VENUE DATA

do {

printf("\t\tCheck List:\n");

printf("\tVenue ID : VNE0%d\n", venueTotalCount + 1);

printf("\tVenue Desciption : %s\n", &tempVenueDes);

printf("\tVenue Category : %s\n", &tempCategory);

printf("\tVenue Maximum Visitor : %d\n", tempMaxVisitor);

printf("\tVenue Checker : %s\n", &tempVenueChecker);

printf("\tVenue Status : %c\n", tempVenueStatus);

printf("\tOperation Time > From %d:%d To %d:%d\n",tempStartHrs,tempStartMin,tempEndHrs,tempEndMin);

printf("Do the Information for new venue correct?(Yes=y No=n):");

rewind(stdin);

scanf("%c", &opt1);

opt1 = toupper(opt1);

//STORE AND PRINT THE DATA IN TEXT FILE

if (opt1 == 'Y') {

venueTotalCount = venueTotalCount + 1;

strcpy(venueInform[venueTotalCount].venueDes, &tempVenueDes);

strcpy(venueInform[venueTotalCount].venueCategory, &tempCategory);

venueInform[venueTotalCount].venueMaxVisior = tempMaxVisitor;

strcpy(venueInform[venueTotalCount].venueChecker, &tempVenueChecker);

venueInform[venueTotalCount].venueStatus = tempVenueStatus;

venueInform[venueTotalCount].operationStartTimeHrs = tempStartHrs;

venueInform[venueTotalCount].operationStartTimeMin = tempStartMin;

venueInform[venueTotalCount].operationEndTimeHrs = tempEndHrs;

venueInform[venueTotalCount].operationEndTimeMin = tempEndMin;

fprintf(venuefPtr, "\n0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d", venueTotalCount, venueInform[venueTotalCount].venueDes,venueInform[venueTotalCount].venueCategory, venueInform[venueTotalCount].venueMaxVisior, venueInform[venueTotalCount].venueChecker, venueInform[venueTotalCount].venueStatus, venueInform[venueTotalCount].operationStartTimeHrs, venueInform[venueTotalCount].operationStartTimeMin, venueInform[venueTotalCount].operationEndTimeHrs, venueInform[venueTotalCount].operationEndTimeMin);

printf("\t\n[Record was saved!]\n");

}

//CLEAR THE TEMPORARY DATA

else if (opt1 == 'N') {

printf("\t\n[Record was cleared!]\n");

}

//INVALID INPUT ON DECISION

else {

printf("\t\n[Invalid data Input! Try again!]\n");

}

} while (opt1 != 'Y' && opt1 != 'N');

//ASKING FOR CONTINUE ON SAME SECTION OR NOT

do {

printf("Do you want to continue add venue?(Yes=Y No=N):");

rewind(stdin);

scanf("%c", &opt2);

opt2 = toupper(opt2);

if (opt2 == 'N') {

fclose(venuefPtr);

venueMenu();

}

else if (opt2 != 'Y' && opt2 != 'N') {

printf("\n\t[Invalid data Input! Try again!]\n");

}

} while (opt2 != 'Y' && opt2 != 'N');

} while (opt2 == 'Y');

}

//SEARCH FUNCTION FOR VENUE INFO//

void searchVenue() {

int selection; //FOR MAKING DECISION

int index; //FOR COUNTING ARRAY IN FOR LOOP PART

int resultCounter = 0; //CALCULATE THE RESULT THAT MATCH WITH USER SEARCH REQUIREMENT

int searchVenueId; //USER ENTER VENUE ID FOR SEARCHING

char selec; //FOR MAING DESICION

char searchStatus; //USER ENTER VENUE STATUS FOR SEARCHING

char searchCategory; //USER ENTER VENUE CATEGORY CODE FOR SEARCHING

printf("\t\t=============================== SEARCH VENUE SECTION ===============================\n");

do {

resultCounter = 0; //CLEAR THE COUNTER AFTER USER EVERY SEARCHING

do {

printf("\n\t---------- New Search Section ----------\n");

printf("\t\tSearch Bar:\n");

printf("\tWhich requirement do you want to search:\n");

printf("\t1.Venue ID\n");

printf("\t2.Venue Status\n");

printf("\t3.Venue Checker that still empty\n");

printf("\t4.Venue that still Open\n");

printf("\t5.Venue Category\n");

printf("\t0.Back\n");

printf("\tEnter your selection:");

scanf("%d", &selection);

switch (selection) {

case 1:

//USER ENTER THE VENUE ID THAT NEED

printf("Enter Venue ID(EXP: VNE001 -> 1):");

scanf("%d", &searchVenueId);

printf("\n\tResult:\n");

for (index = 0; index <= venueTotalCount; index++) {

if (venueInform[index].venueId - searchVenueId == 0) {

printf("\n\t---Venue No %d---\n", resultCounter + 1);

printf("\n\tVenue ID :VNE0%d\n", venueInform[index].venueId);

printf("\tVenue Name : %s\n", venueInform[index].venueDes);

printf("\tVenue Category : %s\n", &venueInform[index].venueCategory);

printf("\tVenue Maximum Visitor : %d\n", venueInform[index].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[index].venueChecker);

if (venueInform[index].venueStatus == 'O') {

printf("\tVenue Status : Opening\n");

}

else {

printf("\tVenue Status : Closed\n");

}

printf("\tVenue Operation Time -> %d:%d To %d:%d", venueInform[index].operationStartTimeHrs, venueInform[index].operationStartTimeMin, venueInform[index].operationEndTimeHrs, venueInform[index].operationEndTimeMin);

resultCounter++;

}

}

if (resultCounter == 0) {

printf("\n\t[No result found]\n\n");

}

printf("\n\t[Total %d result found]\n\n", resultCounter);

break;

case 2:

do {

printf("Enter which status do you wnat to search(Opening ->'O' Closed ->'X'):");

rewind(stdin);

scanf("%c", &searchStatus);

searchStatus = toupper(searchStatus);

if (searchStatus != 'O' && searchStatus != 'X') {

printf("\t[Error,Invalid Data]\n");

}

} while (searchStatus != 'O' && searchStatus != 'X');

printf("\n\tResult:\n");

printf("\t%s %s %s %12s\n", "Venue ID", "Venue Description", "Maximum Visitor", "Venue Checker");

for (index = 0; index <= venueTotalCount; index++) {

if (venueInform[index].venueStatus == searchStatus) {

if (venueInform[index].venueId = searchStatus) {

printf("\n\t---Venue No %d---\n", resultCounter + 1);

printf("\n\tVenue ID :VNE0%d\n", venueInform[index].venueId);

printf("\tVenue Name : %s\n", venueInform[index].venueDes);

printf("\tVenue Category : %s\n", &venueInform[index].venueCategory);

printf("\tVenue Maximum Visitor : %d\n", venueInform[index].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[index].venueChecker);

if (venueInform[index].venueStatus == 'O') {

printf("\tVenue Status : Opening\n");

}

else {

printf("\tVenue Status : Closed\n");

}

printf("\tVenue Operation Time -> %d:%d To %d:%d", venueInform[index].operationStartTimeHrs, venueInform[index].operationStartTimeMin, venueInform[index].operationEndTimeHrs, venueInform[index].operationEndTimeMin);

resultCounter++;

}

}

}

if (resultCounter == 0) {

printf("\n\t[No result found]\n");

}

printf("\t[Total %d record found]\n", resultCounter);

break;

case 3:

printf("\n\tResult:\n");

printf("\t%s %s %s %15s\n","Venue ID","Venue Description","Maximum Visitor","Venue Status");

for (index = 0; index < venueTotalCount; index++) {

if (strcmp(venueInform[index].venueChecker, "---") == 0) {

printf("\n\t---Venue No %d---\n", resultCounter + 1);

printf("\n\tVenue ID :VNE0%d\n", venueInform[index].venueId);

printf("\tVenue Name : %s\n", venueInform[index].venueDes);

printf("\tVenue Category : %s\n", &venueInform[index].venueCategory);

printf("\tVenue Maximum Visitor : %d\n", venueInform[index].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[index].venueChecker);

if (venueInform[index].venueStatus == 'O') {

printf("\tVenue Status : Opening\n");

}

else {

printf("\tVenue Status : Closed\n");

}

printf("\tVenue Operation Time -> %d:%d To %d:%d", venueInform[index].operationStartTimeHrs, venueInform[index].operationStartTimeMin, venueInform[index].operationEndTimeHrs, venueInform[index].operationEndTimeMin);

resultCounter++;

}

}

if (resultCounter == 0) {

printf("\tNo result found\n");

}

printf("\n\t[Total %d Result found]\n\n",resultCounter);

break;

case 4:

for (index = 0; index < venueTotalCount; index++) {

if (venueInform[index].venueStatus == 'O') {

if (venueInform[index].operationStartTimeHrs < t.wHour && venueInform[index].operationEndTimeHrs > t.wHour) {

printf("\n\t---Venue No %d---\n", resultCounter+1);

printf("\n\tVenue Name : %s\n", venueInform[index].venueDes);

printf("\tVenue Category : %s\n", venueInform[index].venueCategory);

printf("\tVenue Maximum Visitor : %d\n", venueInform[index].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[index].venueChecker);

printf("\tVenue Status : Opening\n");

printf("\tVenue Operation Time -> %d:%d To %d:%d", venueInform[index].operationStartTimeHrs, venueInform[index].operationStartTimeMin, venueInform[index].operationEndTimeHrs, venueInform[index].operationEndTimeMin);

resultCounter++;

}

}

}

if (resultCounter == 0) {

printf("\n\t[No result found]\n");

}

printf("\t[Total %d record found]\n", resultCounter);

break;

case 5:

printf("\t\tCategory Code:\n");

printf("\n\t1.Sport -> SPT\n");

printf("\n\t2.Block -> BLK\n");

printf("\n\t3.Faciulity -> FAC\n");

printf("Search For Which Category:");

rewind(stdin);

scanf("%s", &searchCategory);

for (index = 0; index < venueTotalCount; index++) {

if (strcmp(venueInform[index].venueCategory, searchCategory) == 0) {

printf("\n\t---Venue No %d---\n", resultCounter + 1);

printf("\n\tVenue ID :VNE0%d\n", venueInform[index].venueId);

printf("\tVenue Name : %s\n", venueInform[index].venueDes);

printf("\tVenue Category : %s\n", &venueInform[index].venueCategory);

printf("\tVenue Maximum Visitor : %d\n", venueInform[index].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[index].venueChecker);

if (venueInform[index].venueStatus == 'O') {

printf("\tVenue Status : Opening\n");

}

else {

printf("\tVenue Status : Closed\n");

}

printf("\tVenue Operation Time -> %d:%d To %d:%d", venueInform[index].operationStartTimeHrs, venueInform[index].operationStartTimeMin, venueInform[index].operationEndTimeHrs, venueInform[index].operationEndTimeMin);

resultCounter++;

}

}

if (resultCounter == 0) {

printf("\n\t[No result found]\n");

}

printf("\t[Total %d record found]\n", resultCounter);

break;

case 0:

venueMenu();

break;

default:

printf("\t\n[Invalid Data,Please Try Again]\n\n");

}

} while (selection < 0 || selection > 3);

do {

printf("Do you want to continue to search(Yes=Y No=N):");

rewind(stdin);

scanf("%c", &selec);

selec = toupper(selec);

if (selec != 'Y' && selec != 'N') {

printf("\t[Error,Invalid data Input]\n");

}

} while (selec != 'Y' && selec != 'N');

} while (selec == 'Y');

venueMenu();

}

//MODIFFY FUNCTION FOR VENUE INFO//

void modifyVenue() {

int selectedVenueData,selectedVenue;

int counting,counting2;

int choice;

char editVenueDesc,editVenueChecker,editVenueStatus,editVenueCategory;

int editMaxVisitor, editVenueHrs,editVenueMin;

FILE\* EditVenue;

EditVenue = fopen("VenueInformation.txt", "w+");

if (EditVenue == NULL) {

printf("\t[File Open Error]\n\t[Please contact management]\n");

exit(-1);

}

printf("\t\t\n=============================== MODIFY VENUE SECTION ==================================\n\n");

printf("\t---------- New Modify/Eding Section -----------\n");

do {

printf("\t\tModify Menu:\n\n");

printf("\t\t------------\n");

printf("\t1.Venue Description\n");

printf("\t2.Venue Category\n");

printf("\t3.Venue Maximum Visitor\n");

printf("\t4.Venue Checker\n");

printf("\t5.Venue Status\n");

printf("\t6.Venue Operation Time\n");

printf("\t7.Back\n");

printf("\t0.Back To Venue Menu\n\n");

printf("Which Data do you want to modify:");

scanf("%d", &selectedVenueData);

if (selectedVenueData < 0 || selectedVenueData>7) {

printf("\t\n[Invalid data]\n\tPlease Try Again\n");

}

} while (selectedVenueData < 0 || selectedVenueData>7);

printf("Whcich Data do you want to modify(VNE001 -> 1)[Press -1 to Go Back]:");

scanf("%d", &selectedVenue);

if (selectedVenue == -1) {

venueMenu();

}

for (counting = 0; counting <= venueTotalCount; counting++) {

if (venueInform[counting].venueId == selectedVenue) {

printf("\tVenue ID : VNE0%d\n", venueInform[counting].venueId);

printf("\tVenue Desciption : %s\n", venueInform[counting].venueDes);

printf("\tVenue maximum Visitor : %d\n", venueInform[counting].venueMaxVisior);

printf("\tVenue Checker : %s\n", venueInform[counting].venueChecker);

if (venueInform[counting].venueStatus == 'O') {

printf("\tVenue Status : Open\n");

}

else {

printf("\tVenue Status : Close\n");

}

switch (selectedVenueData) {

case 1:

printf("Enter New Venue Description:");

rewind(stdin);

scanf("%[^\n]", &editVenueDesc);

printf("\t[From %s -> %s]\n", &venueInform[counting].venueDes, &editVenueDesc);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

strcpy(venueInform[counting].venueDes, &editVenueDesc);

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

printf("\n\t[Data Changed]\n\n");

}

else {

printf("\n\t[Data Cleared!]\n\n");

}

break;

case 2:

printf("Enter New Category Code:");

scanf("%s", &editVenueCategory);

printf("\n\t[%s -> %s]\n", venueInform[counting].venueCategory, &editVenueCategory);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

printf("\n\t[Data Changed!]\n\n");

strcpy(venueInform[counting].venueCategory, editVenueCategory);

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

}

else {

printf("\n\t[Data Cleared!]\n\n");

}

break;

case 3:

printf("Enter New Maximum Visitor:");

scanf("%d", &editMaxVisitor);

printf("\n\t[%d visitor(s) -> %d visitor(s)]\n", venueInform[counting].venueMaxVisior, editMaxVisitor);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1 && editMaxVisitor >= 0) {

printf("\n\t[Data Changed!]\n\n");

venueInform[counting].venueMaxVisior = editMaxVisitor;

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

break;

}

else {

printf("\n\t[Data Cleared!]\n\n");

break;

}

case 4:

printf("Enter Your Name:");

rewind(stdin);

scanf("%s", &editVenueChecker);

printf("\n\t[%s -> %s]\n", &venueInform[counting].venueChecker, &editVenueChecker);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

printf("\n\t[Data Changed!]\n");

strcpy(venueInform[counting].venueChecker, &editVenueChecker);

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

break;

}

else {

printf("\n\t[Data Cleared!]\n\n");

break;

}

case 5:

printf("Enter New Venue Status:");

rewind(stdin);

scanf("%c", &editVenueStatus);

printf("\n\t[%c -> %c]\n", venueInform[counting].venueStatus, editVenueStatus);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

printf("\n\t[Data Changed]\n");

venueInform[counting].venueStatus = editVenueStatus;

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

}

else {

printf("\n\t[Data Cleared!]\n\n");

}

break;

case 6:

printf("\n\tWhich Operation Time do you want to change:\n");

printf("1.Open Time\n");

printf("2.Close Time\n");

printf("Choice:");

scanf("%d", &choice);

if (choice == 1) {

printf("Hrs:");

scanf("%d", &editVenueHrs);

printf("Min:");

scanf("%d", &editVenueMin);

printf("\n\t[ %d : %d -> %d : %d ]\n", venueInform[counting].operationStartTimeHrs, venueInform[counting].operationStartTimeMin, editVenueHrs, editVenueMin);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

printf("\n\t[Data Changed]\n");

venueInform[counting].operationStartTimeHrs = editVenueHrs;

venueInform[counting].operationStartTimeMin = editVenueMin;

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

}

else {

printf("\n\t[Data Cleared!]\n\n");

}

}

else if (choice == 2) {

printf("Hrs:");

scanf("%d", &editVenueHrs);

printf("Min:");

scanf("%d", &editVenueMin);

printf("\n\t[ %d : %d -> %d : %d ]\n", venueInform[counting].operationEndTimeHrs, venueInform[counting].operationEndTimeMin, editVenueHrs, editVenueMin);

printf("Do you sure the data change is correct(Yes -> 1)?:");

scanf("%d", &choice);

if (choice == 1) {

printf("\n\t[Data Changed]\n");

venueInform[counting].operationEndTimeHrs = editVenueHrs;

venueInform[counting].operationEndTimeMin = editVenueMin;

for (counting2 = 0; counting2 < venueTotalCount; counting2++) {

fprintf(EditVenue, "0%d|%s|%s|%d|%s|%c|%d|%d|%d|%d\n", venueInform[counting2].venueId, &venueInform[counting2].venueDes, &venueInform[counting2].venueCategory, venueInform[counting2].venueMaxVisior, &venueInform[counting2].venueChecker, &venueInform[counting2].venueStatus, venueInform[counting2].operationStartTimeHrs, venueInform[counting2].operationStartTimeMin, venueInform[counting2].operationEndTimeHrs, venueInform[counting2].operationEndTimeMin);

}

}

else {

printf("\n\t[Data Cleared!]\n\n");

}

}

else{

printf("\t\n[Invalid Input]\n");

break;

}

break;

default:

printf("\n\t[Error,Invalid Data]\n");

break;

}

}

}

fclose(EditVenue);

venueMenu();

}

//DISPLAY ALL VENUE INFORMATION AVAILABLE//

void displayVenue() {

int place;

printf("\t=============================== DISPLAY VENUE SECTION ===============================\n\n");

printf("++====================================================================================================================++\n");

printf("||--------------------------------------------------------------------------------------------------------------------||\n");

printf("%s %s %s %s %s %s %s %s %s %s %10s %8s %s %2s %s ||\n", "||","|","Venue Id","|", "Venue Name","|", "Category","|", "Maximum Visitor","|", "Checker","|", "Status","|", "Operation Time");

printf("++--------------------------------------------------------------------------------------------------------------------++\n");

for (place = 0; place < venueTotalCount; place++) {

if (venueInform[place].venueId >= 10) {

if (venueInform[place].venueStatus == 'O') {

printf("||%4d.%s VNE0%d %2s %18s %s %6s %4s %8d %8s %10s %8s %s %s %d:%d - %d:%d ||\n", place + 1, "|", venueInform[place].venueId, "|", venueInform[place].venueDes, "|", venueInform[place].venueCategory, "|", venueInform[place].venueMaxVisior, "|", venueInform[place].venueChecker, "|", "Opening", "|",venueInform[place].operationStartTimeHrs, venueInform[place].operationStartTimeMin, venueInform[place].operationEndTimeHrs, venueInform[place].operationEndTimeMin);

}

else {

printf("||%4d.%s VNE0%d %2s %18s %s %6s %4s %8d %8s %10s %8s %s %2s %s:%s - %s:%s ||\n", place + 1,"|", venueInform[place].venueId,"|", venueInform[place].venueDes,"|", venueInform[place].venueCategory,"|", venueInform[place].venueMaxVisior,"|", venueInform[place].venueChecker,"|", "Closed", "|","--", "--", "--", "--");

}

}

else {

if (venueInform[place].venueStatus == 'O') {

printf("||%4d.%s VNE0%d %3s %18s %s %6s %4s %8d %8s %10s %8s %s %s %d:%d - %d:%d ||\n", place + 1, "|",venueInform[place].venueId,"|", venueInform[place].venueDes,"|", venueInform[place].venueCategory,"|", venueInform[place].venueMaxVisior,"|", venueInform[place].venueChecker,"|", "Opening","|", venueInform[place].operationStartTimeHrs, venueInform[place].operationStartTimeMin, venueInform[place].operationEndTimeHrs, venueInform[place].operationEndTimeMin);

}

else {

printf("||%4d.%s VNE0%d %3s %18s %s %6s %4s %8d %8s %10s %8s %s %2s %s:%s - %s:%s ||\n", place + 1,"|", venueInform[place].venueId,"|", venueInform[place].venueDes,"|", venueInform[place].venueCategory, "|",venueInform[place].venueMaxVisior,"|", venueInform[place].venueChecker,"|", "Closed","|","--", "--", "--", "--");

}

}

printf("||--------------------------------------------------------------------------------------------------------------------||\n");

}

printf("++--------------------------------------------------------------------------------------------------------------------++\n");

venueMenu();

}

void readVenueInform(count) {

FILE\* ReadVenue;

ReadVenue = fopen("VenueInformation.txt", "r");

if (ReadVenue == NULL) {

printf("Error Open File!Please Find Management\n");

exit(-1);

}

while ((fscanf(ReadVenue, "%d|%[^|]|%[^|]|%d|%[^|]|%c|%d|%d|%d|%d", &venueInform[count].venueId, &venueInform[count].venueDes, &venueInform[count].venueCategory, &venueInform[count].venueMaxVisior, &venueInform[count].venueChecker, &venueInform[count].venueStatus, &venueInform[count].operationStartTimeHrs, &venueInform[count].operationStartTimeMin, &venueInform[count].operationEndTimeHrs, &venueInform[count].operationEndTimeMin)) != EOF)

{

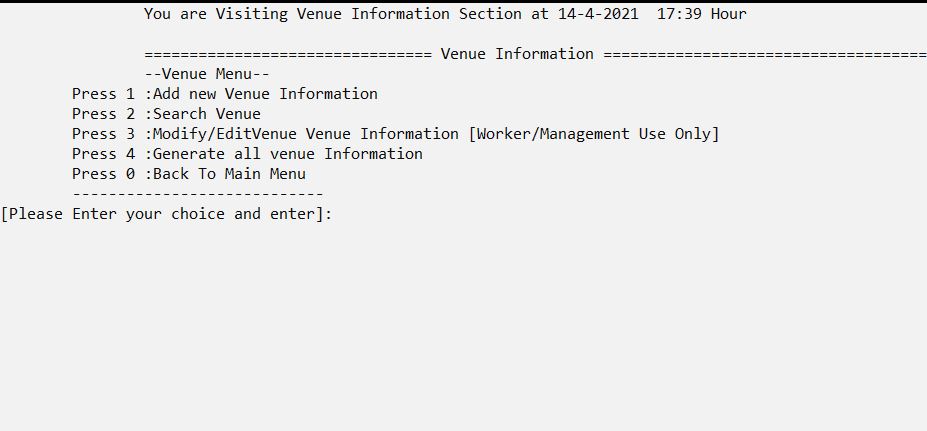
count++;

}

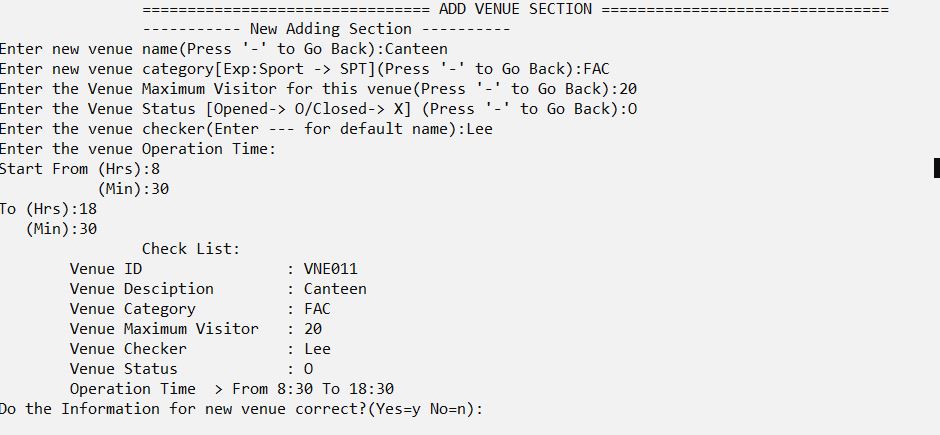
fclose(ReadVenue);

}

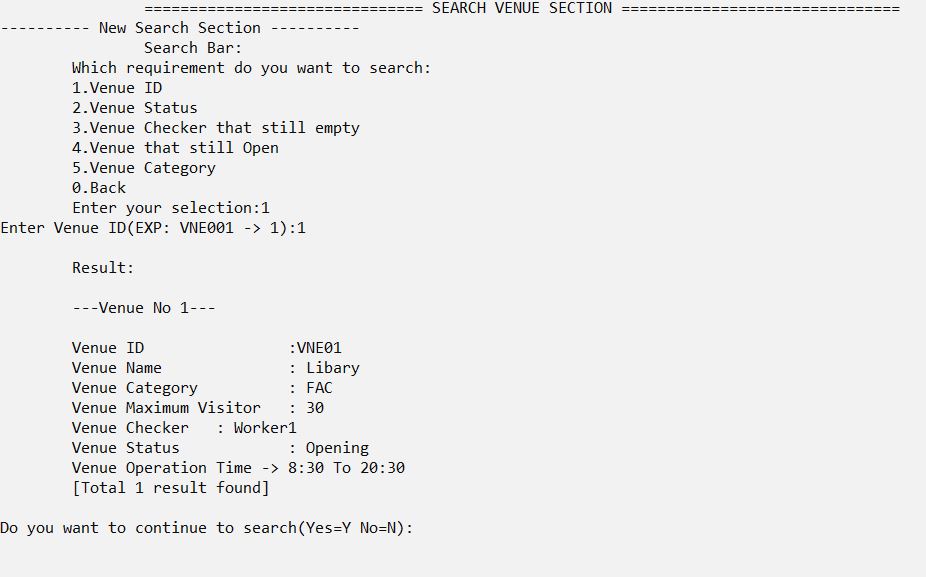
Venue Function

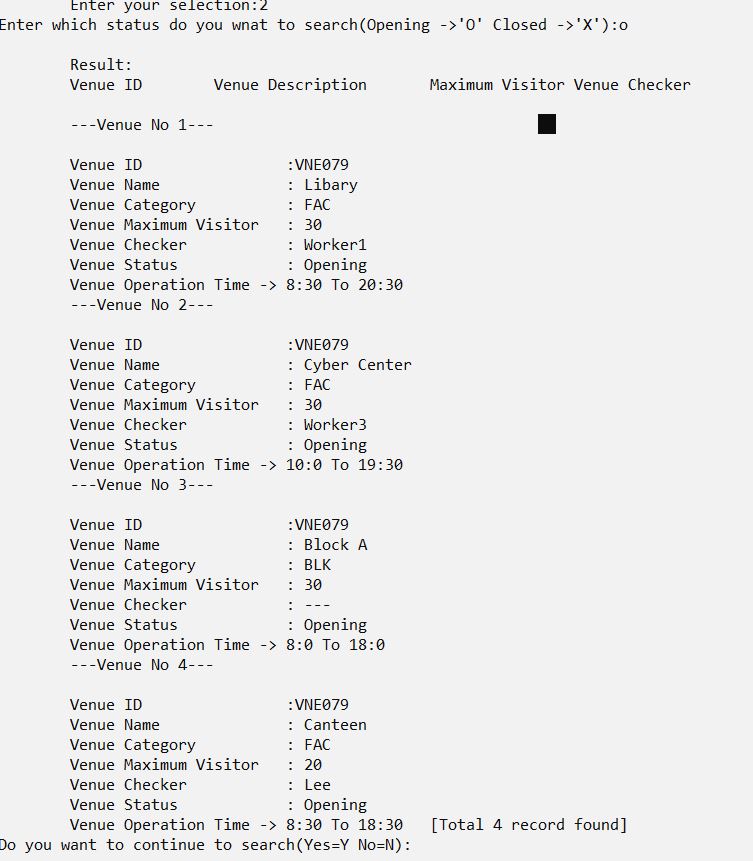


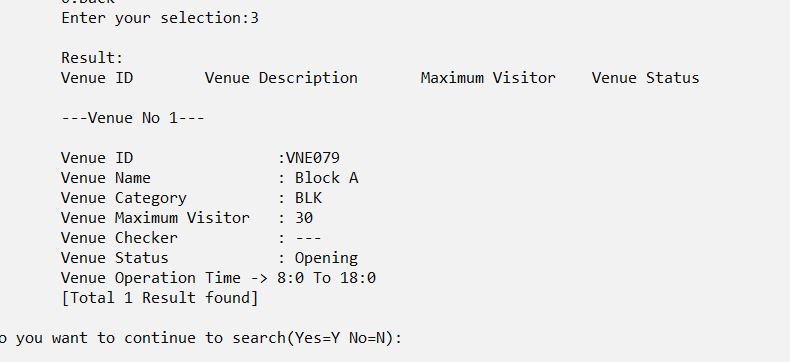
Add Venue Function



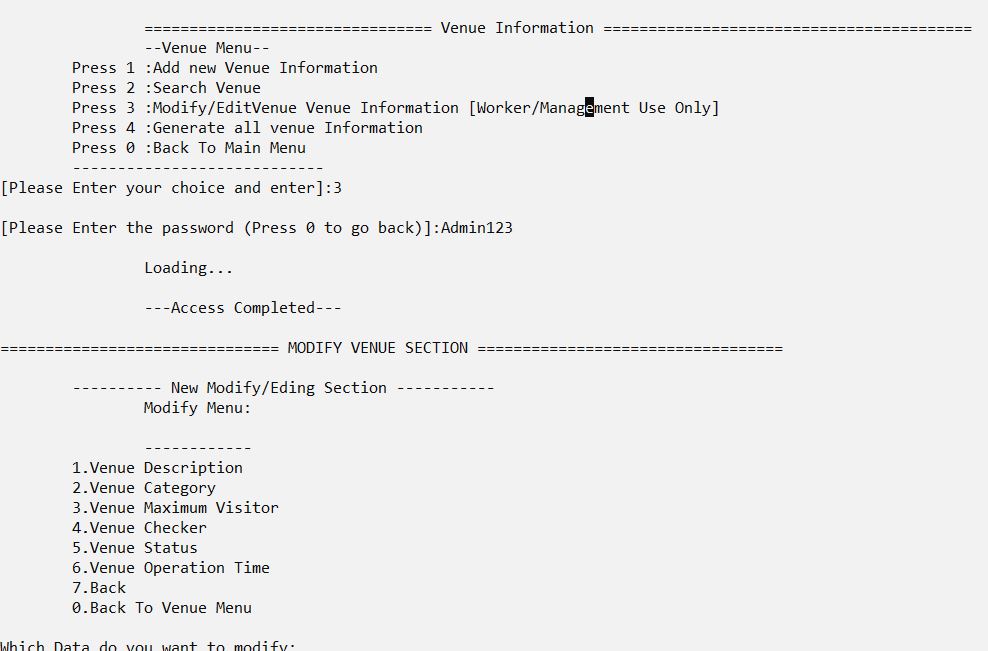
Search Venue Function

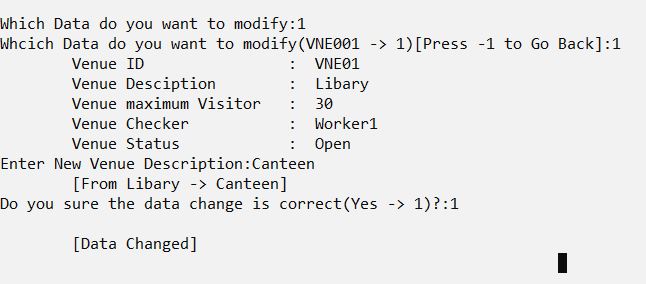






Modify Venue Function





Display Venue Function

