This C program implements a **Hospital Management System** that allows users to manage patients and doctors using file handling. It provides functionalities to **admit and discharge patients**, **add and list doctors**, and **view patient lists**.

**How the Program Works**

**1. Header Files and Global Variables**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

* stdio.h → Provides standard input-output functions (printf, scanf, fopen, etc.).
* stdlib.h → Contains functions like exit() and memory allocation functions.
* time.h → Used for working with date and time (time() and localtime()).

**2. Structure Definitions**

struct patient{

int id;

char patientName[50];

char patientAddress[50];

char disease[50];

char date[12];

}p;

* Defines a struct patient to store:
  + id: Unique Patient ID
  + patientName: Name of the patient
  + patientAddress: Address of the patient
  + disease: Disease of the patient
  + date: Admission date

struct doctor{

int id;

char name[50];

char address[50];

char specialize[50];

char date[12];

}d;

* Defines a struct doctor to store:
  + id: Unique Doctor ID
  + name: Name of the doctor
  + address: Address of the doctor
  + specialize: Doctor's specialization
  + date: Date of joining

**3. File Pointer**

FILE \*fp;

* fp is a global file pointer to read and write records from files.

**4. main() Function (Menu-Driven Program)**

int main(){

int ch;

while(1){

system("cls");

printf("<== Hospital Management System ==>\n");

printf("1.Admit Patient\n");

printf("2.Patient List\n");

printf("3.Discharge Patient\n");

printf("4.Add Doctor\n");

printf("5.Doctors List\n");

printf("0.Exit\n\n");

printf("Enter your choice: ");

scanf("%d", &ch);

* Displays the main menu and asks the user to choose an option.
* system("cls"); clears the screen (works in Windows).

switch(ch){

case 0:

exit(0); // Exits the program

case 1:

admitPatient();

break;

case 2:

patientList();

break;

case 3:

dischargePatient();

break;

case 4:

addDoctor();

break;

case 5:

doctorList();

break;

default:

printf("Invalid Choice...\n\n");

}

printf("\n\nPress Any Key To Continue...");

getch();

}

return 0;

}

* Calls respective functions based on the user's choice.
* Uses getch(); to pause execution until the user presses a key.

**5. admitPatient() - Adding a New Patient**

void admitPatient(){

char myDate[12];

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

sprintf(myDate, "%02d/%02d/%d", tm.tm\_mday, tm.tm\_mon+1, tm.tm\_year + 1900);

strcpy(p.date, myDate);

* Gets the **current system date** and stores it in p.date.

fp = fopen("patient.txt", "ab"); // Open file in append mode

printf("Enter Patient id: ");

scanf("%d", &p.id);

printf("Enter Patient name: ");

fflush(stdin);

gets(p.patientName);

printf("Enter Patient Address: ");

fflush(stdin);

gets(p.patientAddress);

printf("Enter Patient Disease: ");

fflush(stdin);

gets(p.disease);

printf("\nPatient Added Successfully");

fwrite(&p, sizeof(p), 1, fp); // Write to file

fclose(fp);

}

* Opens patient.txt in **append mode** ("ab") to store patient records.
* Takes patient details from the user.
* Uses fwrite(&p, sizeof(p), 1, fp); to **write the patient struct to the file**.
* Closes the file.

**6. patientList() - Displaying Patients**

void patientList(){

system("cls");

printf("<== Patient List ==>\n\n");

printf("%-10s %-30s %-30s %-20s %s\n", "Id", "Patient Name", "Address", "Disease", "Date");

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

fp = fopen("patient.txt", "rb");

while(fread(&p, sizeof(p), 1, fp) == 1){

printf("%-10d %-30s %-30s %-20s %s\n", p.id, p.patientName, p.patientAddress, p.disease, p.date);

}

fclose(fp);

}

* Opens patient.txt in **read mode** ("rb").
* Reads **each patient record** using fread().
* Displays all records in a formatted manner.

**7. dischargePatient() - Deleting a Patient Record**

void dischargePatient(){

int id, f=0;

system("cls");

printf("<== Discharge Patient ==>\n\n");

printf("Enter Patient id to discharge: ");

scanf("%d", &id);

* Takes the **ID of the patient** to discharge.

FILE \*ft;

fp = fopen("patient.txt", "rb");

ft = fopen("temp.txt", "wb");

* Opens patient.txt in **read mode**.
* Opens temp.txt in **write mode** to store records except the one to be deleted.

while(fread(&p, sizeof(p), 1, fp) == 1){

if(id == p.id){

f=1; // Record found

}else{

fwrite(&p, sizeof(p), 1, ft);

}

}

* Reads all records and **copies all except the matching record** to temp.txt.

if(f==1){

printf("\n\nPatient Discharged Successfully.");

}else{

printf("\n\nRecord Not Found !");

}

fclose(fp);

fclose(ft);

remove("patient.txt");

rename("temp.txt", "patient.txt");

}

* Deletes patient.txt and renames temp.txt as patient.txt, effectively **removing the discharged patient**.

**8. addDoctor() - Adding a New Doctor**

void addDoctor(){

char myDate[12];

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

sprintf(myDate, "%02d/%02d/%d", tm.tm\_mday, tm.tm\_mon+1, tm.tm\_year + 1900);

strcpy(d.date, myDate);

system("cls");

printf("<== Add Doctor ==>\n\n");

fp = fopen("doctor.txt", "ab");

printf("Enter Doctor id: ");

scanf("%d", &d.id);

printf("Enter Doctor Name: ");

fflush(stdin);

gets(d.name);

printf("Enter Doctor Address: ");

fflush(stdin);

gets(d.address);

printf("Doctor Specialize in: ");

fflush(stdin);

gets(d.specialize);

printf("Doctor Added Successfully\n\n");

fwrite(&d, sizeof(d), 1, fp);

fclose(fp);

}

* Similar to admitPatient(), but for **adding doctors**.

**9. doctorList() - Displaying Doctors**

void doctorList(){

system("cls");

printf("<== Doctor List ==>\n\n");

printf("%-10s %-30s %-30s %-30s %s\n", "id", "Name", "Address", "Specialize","Date");

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

fp = fopen("doctor.txt", "rb");

while(fread(&d, sizeof(d), 1, fp) == 1){

printf("%-10d %-30s %-30s %-30s %s\n", d.id, d.name, d.address, d.specialize, d.date);

}

fclose(fp);

}

* Reads doctor records from doctor.txt and displays them.

**Conclusion**

This **Hospital Management System** provides: ✅ **File Handling** (Saving and retrieving records)  
✅ **Data Management** (Adding, viewing, and deleting records)  
✅ **Dynamic Date Handling** (Stores the current date automatically)

This is a **basic version** and can be **expanded** to include more features like **searching, modifying records, and advanced security**.