Open Ended Lab

On

Hospital Management System using C Programming Language

Course No: CSE 1210

Course Name: Computer Fundamentals and Programming Laboratory

Submitted By-

Indrajit Gupta

Roll: 2209057

September 30, 2024

Department of Electronics and Communication Engineering

Khulna University of Engineering & Technology

Khulna-9203, Bangladesh.

Problem Statement

Develop a C programming project or problem solution system as a beginner to gain more C programming skills using the following open-ended features:

Open-ended features:

1. You should use at least **two** user-defined functions with "call by value" and "call by reference".
2. You should use at least **one** array and **one** string in the whole project.
3. The project should have at least **one** if-else statement and **one** loop statement.
4. You also may choose any data types, operators, and expressions.

Result analysis:

1. Analyze the time required to complete the whole project.
2. Did you observe any invalid input or any error during run-time?

Evaluation

Open-ended features	Remarks
user-defined functions	
array and string	
if-else statement and loop statement.	

	Excellent	Above Average	Average	Below Average	Marks (10)
Organization/Presentation					
Necessary data presentation					
Result Analysis					
Conclusion					

Signature of Examiner

Contents

1. Introduction

This hospital management system project was written in C programming language and is a console application. This system was built as a straightforward side project in the Code::Blocks IDE using the GCC compiler. The Hospital Management System console software is a simple tool with no graphics. Similar to how hospital management works on mobile devices. User can add, view, edit, search and delete patient records in this project. A file is maintained for each record that was added or modified. The basis of a hospital management system is the idea of creating personal or organizational records of patients, adding this data and updating it. Here, users can add their patient information securely and quickly. The system makes it easy to keep records of each person. The entire project was developed using the "C" programming language and various variables and strings. Consumers will find this little project easy to use and understand. It is a simple terminal program created in C without using any visual elements.

1.1. Motivation and Background

- Add new patient: User can add new patient by entering details such as name, phone, age, gender.
- Show all patient: This option is used to display a list of information about each patient from the file where they are stored.
- Search for patient: Patients can be searched using their patient id.
- Edit a patient: When adding patient, user can change the patient details.
- Delete a record: User can delete them from the file.
- User can save more patient information.

1.2. Objectives

1. To create a project using c-programming and its feature.
2. To implement features like control statement, structures and file handlings
3. To be familiar with resources reusability by making user defined functions.
4. To make the program easy while it is running.
5. To learn more about different functions included in different header files.
6. To be able to make a project in C programming language.

2. Current works

This project is too much useful cause now a days, we need a helper to save patient details in this modern age. This project will help to create and save multiples patient. This also can delete patient record. If the user wants to delete all and then create a new patient list again or want to edit patient details this project will help to do that. This project will help to search a patient from patient list also.

3. Methodology

The program starts executing from the main() function. It initializes some variables and prints an introductory message.

Problem Statement: The first step in software development is to identify the problem. In this case, the problem is to create a Hospital Management System to efficiently store, retrieve, and manipulate hospital information.

Standard C libraries are used, including stdio.h for input and output, string.h for string manipulation and stdlib.h for memory allocation.

File handling is used to read and write patient data to a text file. The program uses a command line interface to interact with the user. Users can select various options through a menu presented by the program. User input is captured using scanf().

Data Validation: The code includes validation checks for data integrity, such as verifying the length of names, phone numbers, ages, allergies etc. It checks for the uniqueness of names, phone numbers to prevent duplicates.

User Manuals and About Us: The code provides additional features like user guidelines and information about the program developers to enhance the user experience.

Procedural Programming: The code follows a procedural programming paradigm, where the program is organized into functions that perform specific tasks. It is a structured way to write code, making it more readable and maintainable.

File Handling: File handling is an essential concept in this code. The program uses the fopen(), fprintf(), and fclose() functions to read and write patient data to a text file. It is a common method for data storage and retrieval.

User Input and Validation: User input is captured using scanf(). The code performs data validation by checking the length of input strings and ensuring the uniqueness of patient data.

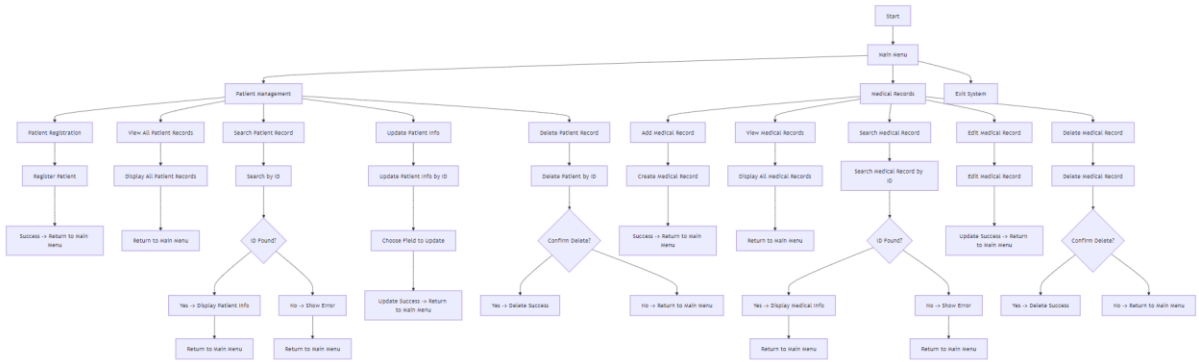
Modular Design: Modular design is employed to break down the program into smaller, manageable functions. Each function handles a specific task, which promotes code reusability and maintainability.

Error Handling: The code includes error handling to provide feedback to the user when issues arise. This ensures that the program can gracefully handle errors and guide the user on corrective actions.

Looping and User Interaction: The code uses a while loop to repeatedly display the main menu and wait for user input. This enables continuous interaction with the user until they choose to exit.

Hospital Data Storage: Patient information (name, phone number, gender) is stored in text files. Patient data is read from and written to the "Patient Report.txt" file. Functions like pat_reg(), viewpatient(), searched(), update (), addmed(), viewmed() and dlt() interact with this data file to perform operations.

3.1 Flow chart



3.2 Hardware and Software Requirements

1. Code::Blocks IDE
2. GCC compiler

4. Implementation and Testing

The program is implemented in the C programming language. Standard C libraries are used, including stdio.h for input and output, for string string.h. stdlib.h is used to manipulation, and for memory allocation. File handling to read and write hospital data to a text file.

4.1 Sample Codes

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>

struct patient
{
   int patientid;
   char name [50];
   int age;
   char gen[20];
   int phn;
   char ins_name [50];
   char allergies [50];

};
struct medi_rec
{
   int medid;
   char name[50];
   int age;
   char gen[10];
   char ins_name [50];
   char allergies [50];
   char prob [50];
   char dname [50];
};
int pat_id = 100;

void pat_reg(struct patient p[],int size)
{
   pat_id=pat_id+1;
   p[size].patientid= pat_id;
   printf("\nEnter Patient's Full Name: ");
   scanf(" %[^\n]",p[size].name);
   printf("Enter Patient's Age: ");
   scanf(" %d",&p[size].age);
   printf("Enter Patient's Gender: ");
   scanf(" %[^\n]",&p[size].gen);
   while(1)
   {
      printf("Enter Patient's Phone Number (Start
after 0) : 0");
```

```
      scanf("%d",&p[size].phn);
      if(p[size].phn>9999999999 ||
p[size].phn<100000000)
      {
         printf("Invalid Phone Number!\n");
      }
      else
      {
         break;
      }
   }
   printf("Enter Insurance Name: ");
   scanf(" %[^\n]",p[size].ins_name);
   printf("Describe the allergies in 50 words or less:
");
   scanf(" %[^\n]",p[size].allergies);


}

void flreports1(struct patient p[],struct medi_rec
m[],int medi_size,int size)
{
   int i;
   FILE *file1;
   file1 = fopen("Patient Report.txt","w");
   fprintf(file1,"%d\n",medi_size);//bondho kore
khulleo jeno paoa jay
   for(i=0; i<size; i++)
   {
      fprintf(file1,"\n");
      fprintf(file1,"Patient's Id :
%d\n",p[i].patientid);
      fprintf(file1,"Patient's Name :
%s\n",p[i].name);
      fprintf(file1,"Patient's Age : %d\n",p[i].age);
      fprintf(file1,"Patient's Gender :
%s\n",p[i].gen);
      fprintf(file1,"Patient's Contact Number:
%d\n", p[i].phn);
      fprintf(file1,"Patient's Insurance :
%s\n",p[i].ins_name);
      fprintf(file1,"Patient's Allergies :
%s\n",p[i].allergies);
```

```
      fprintf(file1,"Patient's Diagnosis :
%s\n",m[i].prob);
      fprintf(file1,"Patient's Appointed Doctor :
%s\n",m[i].dname);
      printf("File Successfully Genarated..\n");
      fprintf(file1,"\n");
   }
   fclose(file1);
}
void flreports2(struct patient p[],struct medi_rec
m[],int size)
{
   int i,medi_size2;

   FILE *file1;
   file1 = fopen("Patient Report.txt","r");
   fscanf(file1,"%d\n",&medi_size2);
   for(i=0; i<medi_size2; i++)
   {
      fscanf(file1,"\n");
      fprintf(stdout,"\n");
      fscanf(file1,"Patient's Id :
%d\n",&p[i].patientid);
      fprintf(stdout,"Patient's Id :
%d\n",p[i].patientid);
      fscanf(file1,"Patient's Name :
%[^\n]\n",p[i].name);
      fprintf(stdout,"Patient's Name :
%s\n",p[i].name);
      fscanf(file1,"Patient's Age : %d\n",&p[i].age);
      fprintf(stdout,"Patient's Age : %d\n",p[i].age);
      fscanf(file1,"Patient's Allergies :
%[^\n]\n",p[i].allergies);
      fprintf(stdout,"Patient's Allergies :
%s\n",p[i].allergies);
      fscanf(file1,"Patient's Diagnosis :
%[^\n]\n",m[i].prob);
      fprintf(stdout,"Patient's Diagnosis :
%s\n",m[i].prob);
      fscanf(file1,"Patient's Appointed Doctor :
%[^\n]\n",m[i].dname);
      fprintf(stdout,"Patient's Appointed Doctor :
%s\n",m[i].dname);
      printf("File Successfully Loaded..\n");
      fscanf(file1,"\n");
```

```
      fprintf(stdout,"\n");
   }
   fclose(file1);
}

int main()
{
   printf("\t\t\tWelcome to INDRA Hospital
Management System....");

   int menu,sub,pat_size=0,med_size=0;
   struct patient pat[1000];

   struct medi_rec medic [1000];
   while(1)
   {
      printf("\n\n\t\t\t\t\t Main Menu\n");
      printf("\n");
      printf("1) Patient Management\n");
      printf("2) Medical Records\n");
      printf("3) Reports & Analytics\n");
      printf("4) Exit\n");
      printf("Please Select an Option from the Menu
to Proceed: ");
      scanf("%d",&menu);
      if(menu==1)
      {
         printf("\n1)Patient Management:\n");
         printf("   1) Patient Registration.\n");
         printf("   2) View All Patient Records.\n");
         printf("   3) Search Patient Record.\n");
         printf("   4) Update Patient
Information.\n");
         printf("   5) Delete Patient Record.\n");
         printf("   0) Return To Main Menu.\n\n");
         printf("Please select an option from the
sub-menu: ");
         scanf("%d",&sub);
         if(sub==0)
         {

            continue;
         }
         else if(sub==1)
```

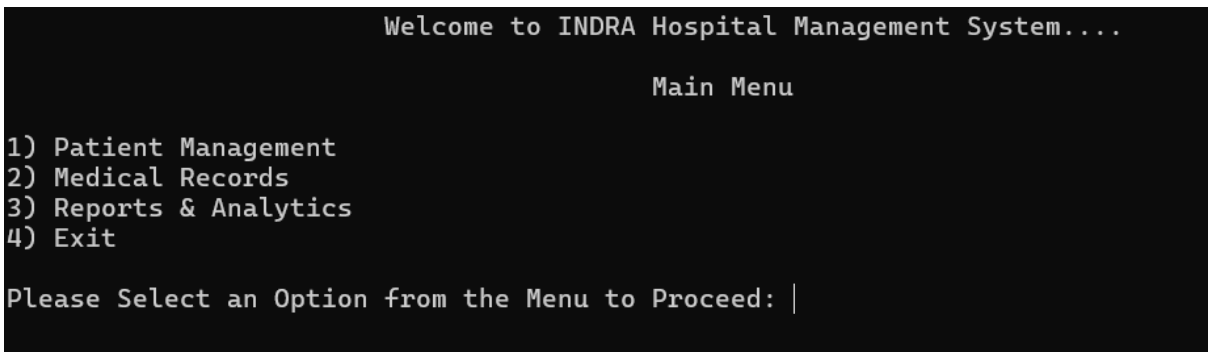
```
        {
           printf("\nPatient Registration: ");
           printf("\n");
           while(1)
           {
              char ch;
              pat_reg(pat,pat_size);
              printf("Patient ID %d Successfully
Added\n",pat_id);
              pat_size++;
              printf("Do You Want To Register?);
        }
        else if (sub==2)
        {
           printf("\n");
           printf("\n");
           viewpatient(pat,pat_size);
        }
        else if (sub==3)
        {
           int userid;
           printf("Enter Patient Id :");
           scanf("%d",&userid);
           printf("\n");
           searched(pat,pat_size,userid);
           printf("\n");

        }
        else if(sub==4)
        {
           int userid2;
           printf("Enter Patient Id :");
           scanf("%d",&userid2);
           printf("\n");
           update(pat,pat_size,userid2);
        }

        else
        {
           printf("Invalid Input!!\n");
        }

     }
     else if(menu==2)
```

```
    {
        printf("\n");
        printf("3) Medical Report:\n");
        printf("   1) Add A New Medical Record.\n");
        printf("   2) View All Medical Record.\n");
        printf("   3) Search a Medical Record\n");
        printf("   4) Edit a Medical Record.\n");
        printf("   0) Return To Main Menu.\n");
        printf("Please select an option from the
sub-menu: ");
        scanf("%d",&sub);
        if(sub==0) {
           continue;
        }

        else if(sub==1){
           printf("\nMedical Record : ");
           printf("\n");
           while(1){
              int uerid;
              char ch3;
              printf("Enter Patient ID to Create a
Medical Record: ");
              scanf("%d",&uerid);

addmed(pat,medic,med_size,uerid,pat_size);
              med_size++;
              printf("Do You Want To Create Another
Patient Record? (Y/N): ");
              scanf(" %c",&ch3);
              if(ch3=='n'||ch3=='N'){
                 printf("\n");
                 printf("\n");
                 break;
              }
              else if(ch3=='y'||ch3=='Y') {
                 continue;
              }
              else{
                 printf("Invalid Input!!\n");

              }
           }
        }
```

```
        else if(sub==2) {
           printf("\n");
           printf("\n");
          viewmed(pat,medic,med_size,pat_size);

        }
        else if(sub==3)
        {
           int uerid1;
           printf("\nEnter Patient Id: ");
           scanf("%d",&uerid1);
           printf("\n");

searchedmed(pat,medic,med_size,uerid1,pat_size
);
           printf("\n");
        }
        else if(sub==4)
        {
           int uerid2;
           printf("\nEnter Patient Id: ");
           scanf("%d",&uerid2);
           printf("\n");

dltmed(pat,medic,med_size,uerid3,pat_size);
           printf("\n");
        }
        else
        {
           printf("Invalid Input!!\n");
        }
     }
     else if(menu==3)
     {
        printf("4) Reports & Analytics:\n");
        printf("   1) Search & Generate a All Patient
Report.\n");
        printf("   2) Load Patient Record. \n");
        printf("   0) Return To Main Menu.\n");
        printf("Please select an option from the
sub-menu: ");
        scanf("%d",&sub);
        if(sub==0)
        {
```

```
            continue;
        }
        else if(sub==1)
        {
            flreports1(pat,medic,med_size,pat_size);
        }
        else if(sub==2)
        {
            flreports2(pat,medic,pat_size);
            printf("\n");

        }
        else
        {
            printf("Invalid Input!!\n");
        }
    }
    else if(menu==4)
    {
        break;
    }
    else
    {
        printf("Invalid Input!!\n");
    }

  }
}
```

4.2 Sample Input Output



```
                      Welcome to INDRA Hospital Management System....

                                      Main Menu

1) Patient Management
2) Medical Records
3) Reports & Analytics
4) Exit

Please Select an Option from the Menu to Proceed: 1

1)Patient Management:
   1) Patient Registration.
   2) View All Patient Records.
   3) Search Patient Record.
   4) Update Patient Information.
   5) Delete Patient Record.
   0) Return To Main Menu.

Please select an option from the sub-menu:
```

```
Please select an option from the sub-menu: 1

Patient Registration:

Enter Patient's Full Name: Abc
Enter Patient's Age: 18
Enter Patient's Gender: M
Enter Patient's Phone Number (Start after 0) : 01234567891
Enter Insurance Name: def
Describe the allergies in 50 words or less: ghi
Patient ID 101 Successfully Added
Do You Want To Register Another Patient (Y/N): y

Enter Patient's Full Name: qwe
Enter Patient's Age: 20
Enter Patient's Gender: F
Enter Patient's Phone Number (Start after 0) : 03698521479
Invalid Phone Number!
Enter Patient's Phone Number (Start after 0) : 01234567895
Enter Insurance Name: ghi
Describe the allergies in 50 words or less: pol
Patient ID 102 Successfully Added
Do You Want To Register Another Patient (Y/N): n
```

```
1)Patient Management:
   1) Patient Registration.
   2) View All Patient Records.
   3) Search Patient Record.
   4) Update Patient Information.
   5) Delete Patient Record.
   0) Return To Main Menu.

Please select an option from the sub-menu: 2



Patient ID: 101
Patient Name: Abc
Patient's age: 18
Patient's Gender: M
Contact Number: 1234567891
Insurance : def
Allergies: ghi

Patient ID: 102
Patient Name: qwe
Patient's age: 20
Patient's Gender: F
Contact Number: 1234567895
Insurance : ghi
Allergies: pol
```

```
1) Patient Management
2) Medical Records
3) Reports & Analytics
4) Exit

Please Select an Option from the Menu to Proceed: 2

3) Medical Report:
   1) Add A New Medical Record.
   2) View All Medical Record.
   3) Search a Medical Record
   4) Edit a Medical Record.
   5) Delete a Medical Record.
   0) Return To Main Menu.
Please select an option from the sub-menu: 1

Medical Record :
Enter Patient ID to Create a Medical Record: 101
Enter Patient's Diagnosis: Fever
Enter Appointed Doctor: Dr. ABCDE
Patient Record Successfully Created
Do You Want To Create Another Patient Record? (Y/N): y
Enter Patient ID to Create a Medical Record: 102
Enter Patient's Diagnosis: Headace
Enter Appointed Doctor: Dr. FGHI
Patient Record Successfully Created
Do You Want To Create Another Patient Record? (Y/N): n
```

```
1) Patient Management
2) Medical Records
3) Reports & Analytics
4) Exit

Please Select an Option from the Menu to Proceed: 3
4) Reports & Analytics:
   1) Search & Generate a All Patient Report.
   2) Load Patient Record.
   0) Return To Main Menu.
Please select an option from the sub-menu: 1
File Successfully Genarated..
File Successfully Genarated..
```

```
4) Reports & Analytics:
   1) Search & Generate a All Patient Report.
   2) Load Patient Record.
   0) Return To Main Menu.
Please select an option from the sub-menu: 2

Patient's Id : 101
Patient's Name : Abc
Patient's Age : 18
Patient's Gender : M
Patient's Contact Number: 1234567891
Patient's Insurance : def
Patient's Allergies : ghi
Patient's Diagnosis : Fever
Patient's Appointed Doctor : Dr. ABCDE
File Successfully Loaded..


Patient's Id : 102
Patient's Name : qwe
Patient's Age : 20
Patient's Gender : F
Patient's Contact Number: 1234567895
Patient's Insurance : ghi
Patient's Allergies : pol
Patient's Diagnosis : Headace
Patient's Appointed Doctor : Dr. FGHI
File Successfully Loaded..
```

5. Conclusion

The Hospital Management System, developed in C, user-friendly solution for efficiently organizing and managing patient information. It addresses the growing need for effective hospital management int today's digital world.

5.1 Future work

There are several potential areas for future work and improvements to enhance its functionality, usability, and robustness. Here are some ideas for future work:

1. Graphical User Interface (GUI): One significant enhancement could be the development of a graphical user interface (GUI). This would make the application more user-friendly, with buttons, forms, and visual feedback.

2. Database Integration: Transition from storing hospitals in text files to using a database management system (e.g., SQLite, MySQL). This would improve data management, querying, and scalability.

3. Cloud Integration: Implement cloud integration to sync and backup hospital data. This would allow users to access their hospitals from multiple devices and provide data security.

4. Advanced Search and Filters: Enhance the search functionality with filters, enabling users to search by various criteria (e.g., name, phone number, email, organization, etc.).

5. Hospital Groups: Implement the ability to group hospitals, allowing users to categorize and manage their hospitals more effectively.

6. Hospital Import/Export: Add features to import patient from popular formats (e.g., vCard) and export patient to different formats for compatibility with other applications.

7. Hospital Image Support: Allow users to associate images with patients, improving visual recognition and personalization.

8. Data Encryption: Implement data encryption to enhance security and protect hospital information.

9. Multi-Language Support: Make the application available in multiple languages to cater to a broader user base.

10. Custom Fields: Allow users to add custom fields to hospitals to accommodate specific information unique to their needs.

11. Reminders and Notifications: Integrate reminder functionality to set important dates and events associated with hospitals.

12. Export and Import Profiles: Enable users to save and switch between different hospital profiles for various use cases.

13. Mobile App and Web Version: Develop a mobile app version for iOS and Android and a web-based version for cross-platform accessibility.

14. User Accounts and Authentication: Implement user accounts with authentication to protect hospital data and allow multiple users to access the same application.

15. Social Media Integration: Enable users to link hospitals to their social media profiles for easier access to additional information.

16. AI and Natural Language Processing: Implement AI features for recognizing and processing hospital data more intelligently, such as auto-completion of hospital details.

17. Data Backup and Recovery: Add features for automatic data backups and the ability to recover deleted hospitals.

18. Feedback and Reporting: Include a feedback and reporting system for users to report issues or suggest improvements.

19. Accessibility Features: Make the application accessible to individuals with disabilities by adhering to accessibility standards.

20. Performance Optimization: Continuously work on optimizing the application's performance to handle a large number of patients efficiently.

Future work on a Hospital Management System involves a combination of software development, user experience improvement, and data management enhancements. The specific directions for future work should align with the goals and needs of the users and the application's intended use cases.

References

[1] Programming in ANSI C by E. Balagurusamy

[2] https://www.tutorialspoint.com